

Journal of WSCG

An international journal of algorithms, data structures and techniques for computer graphics and visualization, surface meshing and modeling, global illumination, computer vision, image processing and pattern recognition, computational geometry, visual human interaction and virtual reality, animation, multimedia systems and applications in parallel, distributed and mobile environment.

EDITOR – IN – CHIEF

Václav Skala

Journal of WSCG

Editor-in-Chief: Vaclav Skala
c/o University of West Bohemia
Faculty of Applied Sciences
Univerzitni 8
CZ 306 14 Plzen
Czech Republic
<http://www.VaclavSkala.eu>

Managing Editor: Vaclav Skala

Printed and Published by:
Vaclav Skala - Union Agency
Na Mazinach 9
CZ 322 00 Plzen
Czech Republic

Published in cooperation with the University of West Bohemia
Univerzitní 8, 306 14 Pilsen, Czech Republic

Hardcopy: **ISSN 1213 – 6972**
CD ROM: **ISSN 1213 – 6980**
On-line: **ISSN 1213 – 6964**

Journal of WSCG

An international journal of algorithms, data structures and techniques for computer graphics and visualization, surface meshing and modeling, global illumination, computer vision, image processing and pattern recognition, computational geometry, visual human interaction and virtual reality, animation, multimedia systems and applications in parallel, distributed and mobile environment.

EDITOR – IN – CHIEF

Václav Skala

Journal of WSCG

Editor-in-Chief: Vaclav Skala
c/o University of West Bohemia
Faculty of Applied Sciences
Univerzitni 8
CZ 306 14 Plzen
Czech Republic
<http://www.VaclavSkala.eu>

Managing Editor: Vaclav Skala

Printed and Published by:
Vaclav Skala - Union Agency
Na Mazinach 9
CZ 322 00 Plzen
Czech Republic

Published in cooperation with the University of West Bohemia
Univerzitní 8, 306 14 Pilsen, Czech Republic

Hardcopy: **ISSN 1213 – 6972**
CD ROM: **ISSN 1213 – 6980**
On-line: **ISSN 1213 – 6964**

Journal of WSCG

Editor-in-Chief

Vaclav Skala

c/o University of West Bohemia
Faculty of Applied Sciences
Department of Computer Science and Engineering
Univerzitni 8, CZ 306 14 Plzen, Czech Republic
<http://www.VaclavSkala.eu>

Journal of WSCG URLs: <http://www.wscg.eu> or <http://wscg.zcu.cz/jwscg>

Editorial Board

Baranoski,G. (Canada)	Oliveira,Manuel M. (Brazil)
Benes,B. (United States)	Pasko,A. (United Kingdom)
Biri,V. (France)	Peroche,B. (France)
Bouatouch,K. (France)	Puppo,E. (Italy)
Coquillart,S. (France)	Purgathofer,W. (Austria)
Csebfalvi,B. (Hungary)	Rokita,P. (Poland)
Cunningham,S. (United States)	Rosenhahn,B. (Germany)
Davis,L. (United States)	Rossignac,J. (United States)
Debelov,V. (Russia)	Rudomin,I. (Mexico)
Deussen,O. (Germany)	Sbert,M. (Spain)
Ferguson,S. (United Kingdom)	Shamir,A. (Israel)
Goebel,M. (Germany)	Schumann,H. (Germany)
Groeller,E. (Austria)	Teschner,M. (Germany)
Chen,M. (United Kingdom)	Theoharis,T. (Greece)
Chrysanthou,Y. (Cyprus)	Triantafyllidis,G. (Greece)
Jansen,F. (The Netherlands)	Veltkamp,R. (Netherlands)
Jorge,J. (Portugal)	Weiskopf,D. (Germany)
Klosowski,J. (United States)	Weiss,G. (Germany)
Lee,T. (Taiwan)	Wu,S. (Brazil)
Magnor,M. (Germany)	Zara,J. (Czech Republic)
Myszkowski,K. (Germany)	Zemcik,P. (Czech Republic)

Journal of WSCG

Board of Reviewers

2024

Abdul Karim, S. (Malaysia)
Adzhiev, V. (United Kingdom)
Aguirre-Lopez, M. (Mexico)
ARORA, R. (United States)
Baranoski, G. (Canada)
Barton, M. (Spain)
Benes, B. (United States)
Benger, W. (Austria)
Benziane, S. (Algeria)
Birra, F. (Portugal)
Bouatouch, K. (France)
Cabiddu, D. (Italy)
Capobianco, A. (France)
De Martino, J. (Brazil)
Drakopoulos, V. (Greece)
Eisemann, M. (Germany)
Fedorov, V. (Russia)
Feito, F. (Spain)
Fu, Z. (China)
Fuenfzig, C. (Germany)
Galo, M. (Brazil)
Gavrilova, M. (Canada)
Gdawiec, K. (Poland)
Gerrits, T. (Germany)
giannini, f. (Italy)
Goncalves, A. (Portugal)
Grajek, T. (Poland)
Gudukbay, U. (Turkey)
Gunther, T. (Germany)
Hast, A. (Sweden)
Hauenstein, J. (United States)
Heil, R. (Sweden)
Hitschfeld, N. (Chile)
Hu, S. (China)
Hu, C. (Taiwan)
Chaudhuri, P. (India)

Juan, M. (Spain)
Kaczmarek, A. (Poland)
Kerdvibulvech, C. (Thailand)
Klimaszewski, K. (Poland)
Kurasova, O. (Lithuania)
Kurt, M. (Turkey)
Lee, J. (United States)
Lefkovits, S. (Romania)
Lengauer, S. (Austria)
Liu, S. (China)
Lobachev, O. (Germany)
Magdalena-Benedicto, R. (Spain)
Manoharan, P. (India)
Marco, C. (Brazil)
Marques, R. (Spain)
Meyer, A. (France)
Miandji, E. (Sweden)
Miller, M. (Germany)
Montrucchio, B. (Italy)
Müller, S. (Germany)
Nawfal, S. (Iraq)
Nguyen, S. (Viet Nam)
Oliveira, J. (Portugal)
Pagnutti, G. (Italy)
Pan, R. (China)
Papaioannou, G. (Greece)
Pedrini, H. (Brazil)
Pereira, J. (Portugal)
Perez-Diaz, S. (Spain)
Pintus, R. (Italy)
Platis, N. (Greece)
Pombinho, P. (Portugal)
Puig, A. (Spain)
Radouane, K. (France)
Raidou, R. (Austria)
Ray, B. (India)
Renaud, c. (France)

RESHETOV, A. (United States)
Ritter, M. (Austria)
Rodrigues, N. (Portugal)
Rodrigues, J. (Portugal)
Romanengo, C. (Italy)
Sabharwal, C. (United States)
Sacco, M. (Italy)
Sardana, D. (United States)
Satpute, V. (India)
Semwal, S. (United States)
Seracini, M. (Italy)
Shendryk, V. (Ukraine)
Schmalstieg, D. (Germany)
Sirakov, N. (United States)
SLUZEK, A. (Poland)
Sousa, A. (Portugal)

Tandianus, B. (Singapore)
Thalmann, D. (Switzerland)
Tokuta, A. (United States)
Tourre, V. (France)
Tytkowski, K. (Poland)
Wang, L. (United States)
Westermann, R. (Germany)
Wu, S. (Brazil)
Wunsche, B. (New Zealand)
Yoshizawa, S. (Japan)
Zwettler, G. (Austria)

Journal of WSCG

Vol.32, No.1-2, 2024

Contents

Automatic construction of fractal structures with locally controlled lacunarity Bordeaux,B., Gentil,C.	1
End-to-End Move Prediction System for Indoor Rock Climbing: Beta Caller Cardenas,K., Semwal,S., Maher,J.	13
Automated Surface Extraction: Adaptive Remeshing Meets Lagrangian Shrink- Wrapping Cavarga, M.	21
CNN-based Game State Detection for a Foosball Table Hagens,D., Knaup,J., Hergenroether,E., Weinmann,A.	31
MinBackProp – Backpropagating through Minimal Solvers Sungatullina,D., Pajdla,T.	41
Fish Motion Estimation Using ML-based Relative Depth Estimation and Multi-Object Tracking Fang,L., Albadawi,M., Dolereit,T., Kuijper,A., Vahl,M.	51
Reflection probe interpolation for fast and accurate rendering of reflective materials Gojkovic,K., Lesar,Z., Marolt,M.	61
GPU Cache Flush Minimization In Render Graph Systems Sandu, R., Shcherbakov, A.	71
Exploitation of local adjacencies for parallel construction of a Reeb graph variant: cerebral vascular tree case Lepaire, C., Belhaouari, H., Pascual, R., Meseure, P.	79
A proposal of anomaly detection method based on natural data augmentation in the Eigenspace Murakami,N., Hiramatsu,N., Kobayashi,H., Akizuki,S., Hashimoto,M.	91
Fast Training Data Acquisition for Object Detection and Segmentation using Black Screen Luminance Keying Pöllabauer, T., Knauth, V., Boller, A., Kuijper, A., Fellner, D.	101
First Results on Using Transformer for Extroversion Personality Trait Recognition Goncalves, A., Carvalho,J.A.M., Ramos, G.J.J., Paiva, V.V.P.	111

Automatic construction of fractal structures with locally controlled lacunarity

Boris Bordeaux
LIB, Université de
Bourgogne
9 Avenue Alain Savary
BP 47870
21000, Dijon, France
boris.bordeaux@u-bourgogne.fr

Christian Gentil
LIB, Université de
Bourgogne
9 Avenue Alain Savary
BP 47870
21000, Dijon, France
christian.gentil@u-bourgogne.fr

ABSTRACT

Lacunar fractal structures reduce the material quantity and weight while improving some physics properties, such as heat transfers, and preserving good mechanical properties. Nowadays, it is possible to construct such shapes thanks to additive manufacturing. This paper focuses on automatically generating subdivision rules for fractal lacunar structures with local topology control. The first main difficulty is guaranteeing topological consistency while assembling different cells to build a complicated multi-lacuna structure. The second is the adaptation of such shapes to geometric constraints like imposed boundaries. We address these questions throughout the formalism of the Boundary Controlled Iterated Function System. Then, we analyze the lacunarity and complexity of these structures from various geometric, topologic, and fractal measures.

Keywords

Fractals, Geometric Modeling, Lacunar Control, Iterative Modeling, Subdivision

1 INTRODUCTION

Saving energy by designing lighter objects while maintaining high physical properties is a crucial issue for the industry. The global structure must respond to several constraints, like mechanical resistance, energy absorption, heat transfer, and soundproofing, using a minimum quantity of material.

Porous metallic material is a pertinent solution for producing lighter objects. One can obtain metal foams from different techniques: mixtures of gas bubbles and a molten alloy or with the sintering and dissolution process [JZSL05] for instance. Because of the complexity and the randomness of the manufacturing process, it is challenging to guarantee expected properties [Ban06, JWZ07]. Then, the definition, the characterization, and the measure of the porosity arise in different domains to study their relations with physical properties [Esp12, AC15, JWZ07].

Thanks to additive manufacturing, we can produce lighter structures by designing controlled and structured geometry like lattice [TMV⁺16]. These structures generally fill a given volume with the minimum of matter. The most straightforward approach generates a strict periodic lattice as if one directly cuts the volume in a lattice block. The lattice comes from an initial periodic primary cell (thickened edges of a cube, 3D cross, gyroid) duplicated along the three 3D axes. This approach induces discontinuities at the boundaries of the filled volume. A parameter (thickness of beams, for instance) can also control the geometry of the initial cell to modulate its density and generate lattices with variable density to respond to specific physical properties [LJP⁺19]. Numerous surveys on this topic are available; see [PHL20, CLLZ21, FFL⁺18]. Softwares are available like [AKA21] to generate such lattices using optimal surfaces like gyroid as primary cells with variable density. An improvement of this method is to consider the geometry of the volume to adapt the lattice to its boundary to avoid discontinuities. In [KT10], Kou et al. introduce randomness to obtain irregular porous structures to produce variable density. More recently, McNulty et al. [MBZ⁺20] and Levo et al. [LVSZ21] propose bio-inspired approaches introducing multi-scale structures. Topological optimization also uses the multi-scale aspect. A second optimization step introduces a micro-lattice structure in

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

under-constrained parts of the macro lattice, resulting from the first optimization step (see left Figure 1). However, the hierarchy level is generally shallow (one level for topological optimization and generally two levels for bio-inspired approaches). Furthermore, they are provided from ad-hoc implementation without formalizing the topological subdivision process and could be challenging to adapt to any situation.

We propose a new approach based on multi-scale lacunar structures resulting from fractal geometry, as shown in Figure 1 (middle and right). The motivations are: fractals appear as an optimal solution [LC19, AALS17] or are used as a model of bio-inspired structure for optimization [BRBH19]; fractal models can generate a great variety of topologies with different kinds of lacunarities and consequently interesting multi-physics properties such as heat and mass transfer [Pen10, Zha11, ZZ13], energy absorption [MGM⁺18] or acoustical properties [AJS18, SHR97]; fractal modeling encompasses standard NURBS and subdivision surfaces, standard lattice structures, and multi-scale lattices [GGS21].

Producing fractal structures is easy by programming recursive algorithms. However, controlling the resulting shapes is more complicated. Some models like L-system or Boundary Controlled Iterated Function System, associated with topological constraints, allow accurate geometry control [TGM⁺09, GGS21] and enable us to design varieties of lacunar structures with specific fractal topology.

Generating a lightweight structure by an assembly of fractal structures is a challenge. Difficulties come from the different topologies and geometries arising from fractal modeling while ensuring continuity to produce well-defined printable objects. Controlling the boundary topologies of cells makes the assembly process straightforward. For our purpose, we use the BC-IFS model. This model has the advantage of coding the fractal topology with adjacency and incidence constraints, controlling the geometry and the topology independently, and defining free-form shapes according to a set of control points.

As said before, this formalism provides powerful tools for designing various complicated multi-scale lacunar structures. However, currently, designers have to describe all the topological constraints by hand, which could be tedious. Nevertheless, once the topology is defined, the model guarantees the topological coherence, regardless of the geometric realization, which we adjust by control points and the iteration level.

The contribution of this paper is to provide parametrized algorithms that automatically generate a BC-IFS model of various multi-scale 2D structures with different lacunar topological complexities. Section 2 introduces the needed definitions and

properties for the BC-IFS model. Section 3 shows a method to automatically define the subdivision rule of a 2D cell (a face) depending on the fractal topology of its boundary. We complete the lacunarity control with delay subdivisions in Section 4. In Section 5, we propose some measures to characterize the lacunarity and the complexity of fractal structures. Section 6 concludes this article and suggests future works.



Figure 1: Left: Example of a topological optimization with two levels of lattice structures (©Altair Engineering). Middle: Elephant structure (from CGAL Computing Library at <https://github.com/CGAL/cgal>) built with multi-scale lacunas. Right: part of a mechanical system redesigned with multi-scale lacunar components shown in Figure 2.

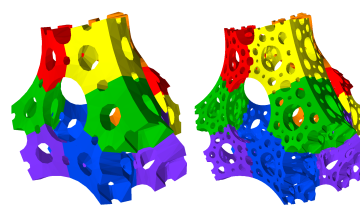


Figure 2: Subdivision of a truncated tetrahedron into some truncated tetrahedrons

2 BACKGROUND: BC-IFS MODEL

The Boundary Controlled Iterated Function System (BC-IFS) [SGGM15] model is based on Iterated Function System [Hut81, Bar90]. A set of contractive transformations defines an object by describing its self-similarity property, i.e., how this set of contractions subdivides the object into smaller copies of itself. By iterating on this set of transformations from an initial object (named the primitive), the obtained sequence converges to a unique shape called the attractor. In practice, the number of iterations is finite, giving an approximation of the attractor.

Recurrent IFS [BEH89] or C-IFS [ZT96] control the subdivision process using an automaton or a directed graph specifying the subdivision of an attractor into copies of itself or other attractors. Each state of the automaton defines a sub-attractor, and the outgoing transitions define the associated subdivision rules. By defining attractors in a barycentric space, C-IFS provide free-form fractal shapes.

Finally, Boundary C-IFS (BC-IFS) [SGGM15, GGS21] uses additional concepts to encode the cellular decomposition with incidence constraints: fractal volumes

bounded by fractal surfaces bounded by fractal edges bounded by vertices. With a set of adjacency constraints between subdivided cells, we define the fractal topology of the attractor (see Figure 3). Incidence and adjacency constraints induce constraints on the BC-IFS transformations. They determine the topology, while the remaining degrees of freedom control the geometry.

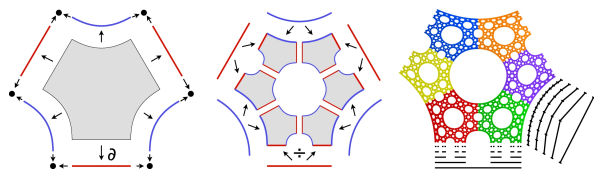


Figure 3: Example of definition 2D fractal topology. Left: the cellular decomposition. Middle: the topology subdivision. Right: the resulting attractor.

3 AUTOMATIC SUBDIVISIONS OF 2D CELLS

To build structures composed of lacunar cells with the BC-IFS model (see middle of Figure 1), one has to define a global shape that assembles different cells. The assembly uses adjacency constraints to ensure a consistent fractal topology at the junction of each cell with another. Then, one has to define the subdivision rule of each cell adapted to the wanted lacunarity (see Figures 3 and 4 for an example in 2D).



Figure 4: Subdivision of a hexahedron into six hexahedrons

Defining the subdivision process of each cell (including incidence and adjacency constraints) could be challenging for many cells. We want to automatize the definition of the subdivision process. To facilitate the assembly of fractal structure, we change the design paradigm by deducing face subdivision from imposed edges. We automatically define the subdivision of a face from its edges with a lacunarity control and deduce all incidence and adjacency constraints.

3.1 Notation

We classify two types of edge topology: B_n edges and C_n edges. The B_n edges subdivide into n connected sub-edges of type B_n (see Figure 5 for an example). The C_n edges subdivide into n disconnected sub-edges of type B_n (see Figure 6 for an example). In practice, B_n edges are Bézier edges, and C_n edges are Cantor edges.

We denote a fractal face by $\mathcal{F}(\partial, E, \mathcal{A})$ with ∂ its boundary, E a tuple of three edge topologies characterizing subdivision process, and \mathcal{A} its subdivision algorithm.

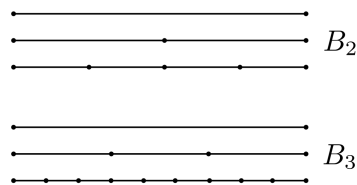


Figure 5: Example of subdivisions of B_2 and B_3 edges.

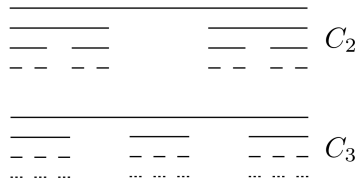


Figure 6: Example of subdivisions of C_2 and C_3 edges.

The parameter ∂ is a tuple of the face's edges in the counter-clockwise direction. We use the following notation $\partial = \{e_0, e_1, \dots, e_N\}$, e_i being the i -th edge. The order of the edges in ∂ is significant, up to a circular permutation. We use this specific notation to facilitate the comparison between boundaries. For instance if $\partial_1 = \{C_3, B_2, C_4, B_2, C_2, B_2\}$ and $\partial_2 = \{B_2, C_2, B_2, C_3, B_2, C_4\}$, they represent the same boundary because $\partial_1 = \partial_2$ with a circular shift of three elements.

The parameter E contains a tuple of three types of edges (E_a, E_l, E_c). The subdivision algorithm uses these edges to define the boundary of the sub-faces. Figure 7 shows where these edges appear after a subdivision. The choice of these edge types directly impacts the lacunarity.

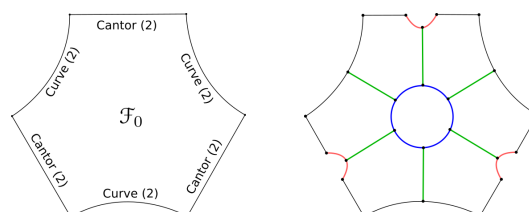


Figure 7: Example of a subdivision process. The adjacencies of the sub-faces are made along the edges E_a in green. The edges that form the central lacuna are the edges E_l in blue. The edges E_c in red are added between two C_n sub-edges.

In this paper, we present several algorithms to define the subdivision of a face. The parameter \mathcal{A} defines the algorithm used.

3.2 Algorithm

We present the algorithm \mathcal{A}_1 to automatize the definition of the subdivision rules of a fractal face from its edges. We present other algorithms in Section 4.3. Notice that the algorithm does not operate the subdivision process. It defines the rules of subdivision and all the adjacency and incidence constraints of the BC-IFS model. Figure 8 describes all the steps of the algorithm \mathcal{A}_1 . It creates one sub-face for each pair of sub-

edges in the corner of the face and one for each sub-edge that is not in a corner. The algorithm defines the incidence constraints for all sub-faces with the first (or the two firsts) edge(s) in the boundary. It also defines adjacency constraints between all adjacent subcells on their E_a edges. The sub-faces have the same parameters as the face, but their boundary differs. The algorithm works as follows:

1. Subdivide the boundary (all the edges) of the fractal face (a).
2. Selection of the sub-edges. The selection step takes, in the sub-edges of the face, the next sub-edge not in a corner, or the next two sub-edges in a corner. Each selection leads to a sub-face creation. The boundary of the sub-face starts with the selection. Define incidence constraints between the sub-edge of the face and the edges in selection.
3. Add to the boundary the E_c edge if the last edge in the current sub-face boundary is a C_n edge (c).
4. Add to the boundary the edges E_a , E_l and E_a in this order (d).
5. Add the E_c edge if the first edge in the sub-face boundary is a C_n edge.
6. Repeat from step 2 until all the sub-faces are created (no more selection).
7. Identify all sub-faces by their edges (i).
8. Define adjacency constraints between all sub-faces on their E_a edges.

We execute the algorithm on an example with the fractal face $\mathcal{F}(\partial, E, \mathcal{A}_1)$, having a boundary $\partial = \langle C_4, B_2, C_2, B_2, C_3, B_2 \rangle$, and having $E = (C_2, B_2, B_2)$. Figure 8 details step-by-step the algorithm for this example. The first step subdivides the boundary. The second step is selection. The first selection is, for instance, the top right corner of the cell: the two sub-edges are B_2 and C_4 . Hence, the boundary of the sub-face starts with $\partial_1 = \langle B_2, C_4 \rangle$. The third step adds an E_c edge since the last edge was a C_n edge: $\partial_1 = \langle B_2, C_4, B_2 \rangle$. The fourth step fills the boundary of the sub-face with the edges E_a , E_l and E_a : $\partial_1 = \langle B_2, C_4, B_2, C_2, B_2, C_2 \rangle$. The fifth step adds a E_c edge if the first edge in the boundary is a C_n edge (not the case for ∂_1). We continue to the next sub-face. The next selection is a sub-edge not in a corner: a C_4 sub-edge. Hence, the boundary of the sub-face starts with $\partial_2 = \langle C_4 \rangle$. Since the last edge in the boundary is a C_n edge, the third step adds the E_c edge: $\partial_2 = \langle C_4, B_2 \rangle$. The fourth step adds the edges E_a , E_l and E_a to the boundary: $\partial_2 = \langle C_4, B_2, C_2, B_2, C_2 \rangle$. The fifth step adds the E_c edge since the first edge is a C_n edge. Finally,

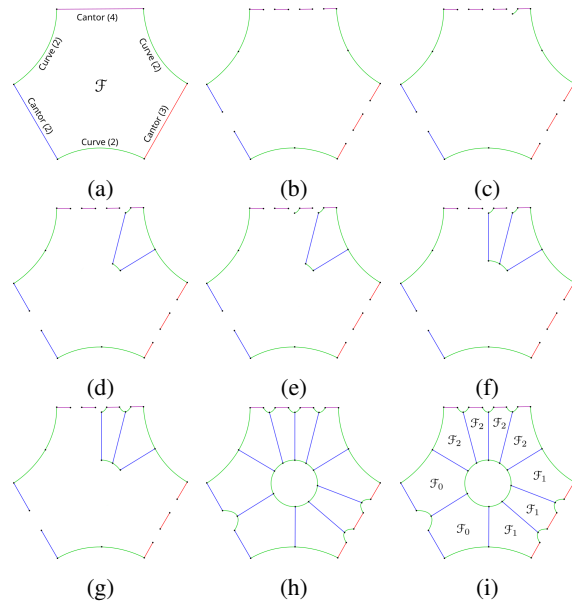


Figure 8: Algorithm \mathcal{A}_1 to define the subdivision rule of a face from its edges.

$\partial_2 = \langle C_4, B_2, C_2, B_2, C_2, B_2 \rangle$, and we continue so on for all the sub-faces.

Figure 9 details five steps to intuitively understand the \mathcal{A}_1 algorithm's logic. An intuitive idea allows us to quickly create other algorithms based on the same principle. We introduce some algorithms in Subsection 4.3. The idea of the algorithm is the following:

1. Subdivide all edges (b).
2. Add two E_c edges between the sub-edges of the C_n edges (c).
3. Add E_l edges in the center of the face to create the lacuna (d).
4. Add the E_a edges between the external boundary and the central lacuna (e).
5. Identify all sub-faces by their edges (f).

3.3 Convergence

The algorithm defines the subdivision rule of a face into sub-faces. It can induce sub-faces with unspecified subdivision rules. Applying the algorithm recursively provides the subdivision rules for all sub-faces that appear during the process. For any initial face, a finite number of faces can appear over the recursion. All sub-faces contain only edges from the base face or the parameter E of the face (E_a , E_l , and E_c). Each sub-face contains a limited number of edges. The maximum is seven: a pair (C_n, C_n) of sub-edges in the corner, two edges E_c , and the three interior edges (two E_a and one E_l). The minimum is four: one B_n sub-edge (not in a corner) and

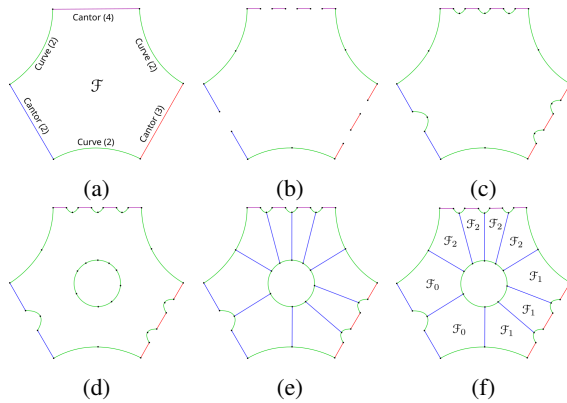


Figure 9: Intuitive algorithm to define the subdivision process of a face from its edges.

the three interior edges. Since there are a finite number of edge types and each sub-face has a finite number of edges, there are a finite number of different sub-faces. Hence, the recursion is guaranteed to terminate.

3.4 Lacunarity control

One can control the lacunarity by choosing the types of edges E_a , E_l , and E_c . Depending on that choice, one face often appears that subdivides into copies of itself. For instance the face $\mathcal{F}_0(\partial_0, E, \mathcal{A}_1)$ with $\partial_0 = \langle C_2, B_2, C_2, B_2, C_2, B_2 \rangle$ and $E = (C_2, B_2, B_2)$ always appears in the sub-faces of all faces $\mathcal{F}(\partial, E, \mathcal{A}_1)$ after some iterations, regardless the boundary ∂ . Figure 10 shows the face \mathcal{F}_0 subdivision.

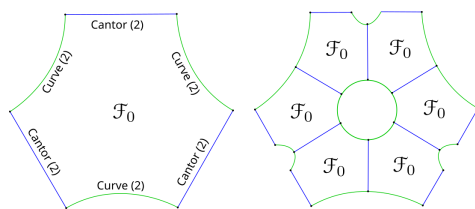


Figure 10: Subdivision of the face $\mathcal{F}_0(\partial_0, E, \mathcal{A}_1)$ with $\partial_0 = \langle C_2, B_2, C_2, B_2, C_2, B_2 \rangle$ and $E = (C_2, B_2, B_2)$. It subdivides into six faces \mathcal{F}_0 .

In the other cases, two or more faces appear, each subdividing into copies of themselves and copies of the other faces. Let be the faces $\mathcal{F}_1(\partial_1, E, \mathcal{A}_1)$ and $\mathcal{F}_2(\partial_2, E, \mathcal{A}_1)$ with $\partial_1 = \langle C_2, B_2, C_2, B_2, C_2, C_2 \rangle$, $\partial_2 = \langle C_2, C_2, C_2, C_2, B_2, C_2, C_2 \rangle$, and $E = (C_2, B_2, C_2)$. It is an example where \mathcal{F}_1 subdivides into four sub-faces \mathcal{F}_1 and two sub-faces \mathcal{F}_2 , whereas \mathcal{F}_2 subdivides into five sub-faces \mathcal{F}_2 and two sub-faces \mathcal{F}_1 . There is no face subdividing into several copies of itself, but there is a cycle between two faces. These faces appear in the sub-faces of all faces $\mathcal{F}(\partial, E, \mathcal{A}_1)$ after some iterations, regardless the boundary ∂ .

4 DELAYED SUBDIVISIONS

We introduce a delay in subdivisions to have structures with faces with different local iteration levels. Without delay (left of Figure 11), the size of the lacunas might be tiny on the small face compared to the ones of the bigger faces. The delay allows us to have a uniform lacunarity when the size of the cells is not uniform (right of Figure 11). On the other hand, when faces have the same size, the delay induces a different lacunarity (see Figure 14). Hence, we propose delay subdivisions and some improvements in the presented algorithm to handle them.

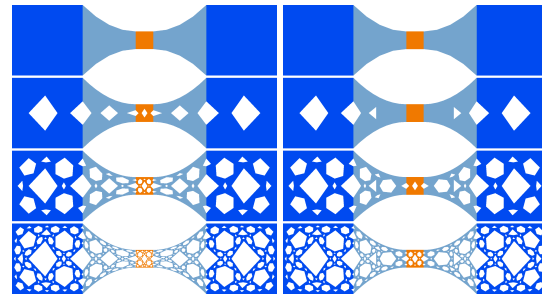


Figure 11: A fractal structure made of five faces without delay (left) and with delay (right). The central face is smaller than the others, but with a delay in its subdivision, its lacunas are not very small compared to the other lacunas.

4.1 Definition

A face with a delay in its subdivision is a face that induces no lacuna after its subdivision but has only its edges subdivided to preserve the consistency of the structure if the face shares an edge with another face. We denote the delay by an exponent $d \geq 2$. The face $\mathcal{F}^d(\partial, E, \mathcal{A})$ subdivides into only one face $\mathcal{F}^{d-1}(\partial', E, \mathcal{A})$ with the boundary ∂' that contains the subdivided edges of ∂ . The sub-edges of C_n edges are unconnected, so we add two E_c edges between each sub-edges. The Figure 12 explains the delay subdivision of the face \mathcal{F}^2 with a boundary $\partial = \langle C_4, B_2, C_2, B_2, C_3, B_2 \rangle$. The sub-face boundary $\partial' = \langle C_4, B_2, B_2, C_4, B_2, B_2, C_4, B_2, B_2, C_4, B_2, B_2, C_2, B_2, B_2, C_2, B_2, B_2, C_3, B_2, B_2, C_3, B_2, B_2, C_3, B_2, B_2 \rangle$.

We also introduce the notion of delay in the subdivision process of edges, also denoted by an exponent $d \geq 2$. The edge subdivides into only one sub-edge of type X_n^{d-1} . Hence, the geometry of the edge does not change over iterations while the edge has a delayed subdivision (i.e., while $d \geq 2$).

4.2 Example

One can combine a delayed subdivision on a face with a delayed subdivision on all its edges to create a global delay in the face subdivision process. It creates a sub-face that does not change the face's shape since it adds

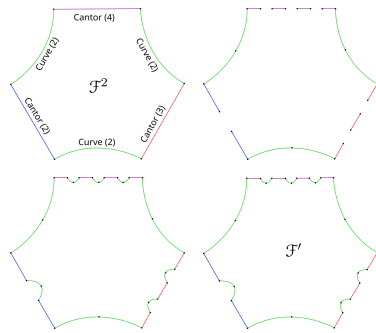


Figure 12: Algorithm to find the subdivision of a delayed face depending on its edges.

no lacuna inside the face, and the boundary does not change because of the delay on each edge. One can also mix delayed and non-delayed edges for one face to make a junction between a face having a global delay and a face having no delay at all.

For instance, we build a structure with three faces $\mathcal{F}_0^2(\partial_0', E, \mathcal{A}_1)$, $\mathcal{F}_1(\partial_1, E, \mathcal{A}_1)$ and $\mathcal{F}_0(\partial_0, E, \mathcal{A}_1)$ with $\partial_0 = \langle C_2, B_2, C_2, B_2, C_2, B_2 \rangle$, ∂_0' the same as ∂_0 but with a delay for each edge, $\partial_1 = \langle C_2, B_2, C_2^2, B_2, C_2^2, B_2, C_2^2, B_2 \rangle$, and $E = (C_2, B_2, B_2)$. Figure 13 shows a scheme of this structure. The faces \mathcal{F}_0^2 and \mathcal{F}_0 behave as the same face, but \mathcal{F}_0^2 has a delay of one iteration level because there is a delay in the face subdivision and also in all its edges subdivisions. The middle face \mathcal{F}_1 connects the faces \mathcal{F}_0^2 and \mathcal{F}_0 with its shared edge C_2 with the face \mathcal{F}_0 and its shared edge C_2^2 with the face \mathcal{F}_0^2 .

At the first iteration level of the structure, the face \mathcal{F}_0 subdivides into six sub-faces \mathcal{F}_0 (see Figure 10). However, the face \mathcal{F}_0^2 subdivides into one sub-face \mathcal{F}_0 ; a delay appears here. Figure 14 shows the structure's third, fourth, and fifth iteration levels. We use adjacency constraints to make the junctions between the faces. Each face has a consistent subdivision thanks to the definition of all adjacency and incidence constraints (by the algorithm). Hence, the structure is consistent at all iteration levels.

4.3 Other subdivision algorithms

The presented algorithm \mathcal{A}_1 creates sub-faces with a shared edge (E_a) between two adjacent sub-faces. The adjacency edges start from the lacuna to the junction of two B_n edges (see Figure 9), but there are other solutions. For instance, Figure 15 shows several ways to subdivide the face \mathcal{F}_0 . The sub-faces depend on how the algorithm places the adjacency edges from the central lacuna on the boundary. The algorithm \mathcal{A}_1 places the adjacency edges at the junction between two B_n edges (middle of Figure 15). The algorithm \mathcal{A}_2 places the same adjacency edges as \mathcal{A}_1 but also on all corners (right of Figure 15). When some edges of a face have a

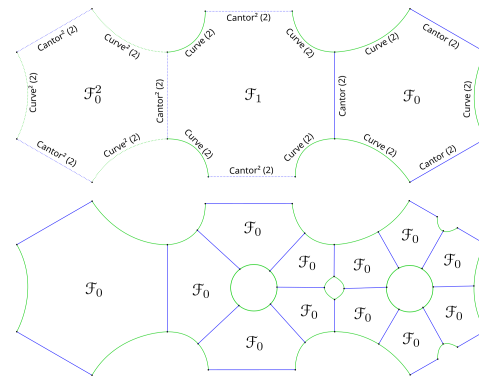


Figure 13: (Above) Scheme of a structure with three faces. The left one is the same as the right one but with a delay in its subdivision. The middle face makes the links between them. (Below) The structure's subdivision.

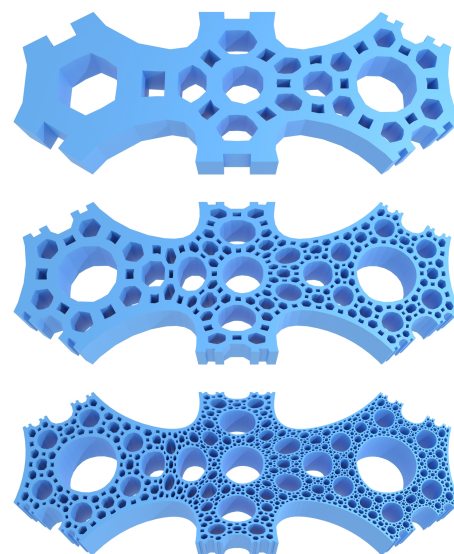


Figure 14: From top to bottom, the second, third, and fourth iteration levels of the extruded structure presented in Figure 13.

delayed subdivision, there are also several ways to subdivide the face, as shown in Figure 16. The algorithm \mathcal{A}_1 places the adjacency edges at the junction between two B_n edges (top right of Figure 16). The algorithm \mathcal{A}_2 places the same adjacency edges as \mathcal{A}_1 but also on all corners (bottom left of Figure 16). The algorithm \mathcal{A}_3 places the same adjacency edges as \mathcal{A}_1 but also on corners surrounding an edge with a delay subdivision (bottom right of Figure 16). When a face has no delay edges, \mathcal{A}_3 and \mathcal{A}_1 generate the same subdivision rule.

Choosing the suitable algorithm would depend on the specific application. However, to limit the number of face types, one has to stick to the same algorithm across all iterations rather than changing it at each level. Alternatively, one can use one algorithm when dealing with faces containing edges that have a delay in their subdivision and another algorithm for faces without such

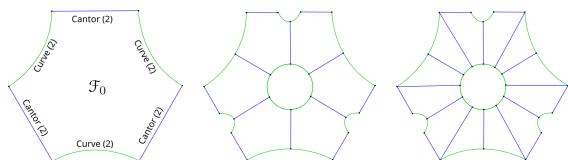


Figure 15: Different subdivisions of the face \mathcal{F}_0 with a boundary $\partial_0 = \{C_2, B_2, C_2, B_2, C_2, B_2\}$.

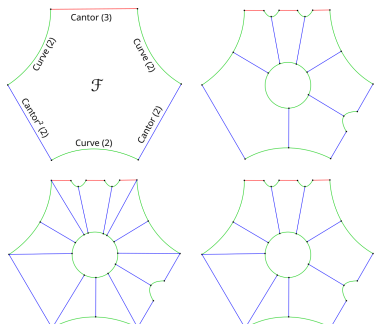


Figure 16: Different subdivisions algorithm of a face \mathcal{F} with a boundary $\partial = \{C_3, B_2, C_2^2, B_2, C_2, B_2\}$.

edges. There are a lot of possibilities to define the subdivision algorithm. The key is to ensure that when the algorithm is applied repeatedly, it leads to already-known faces (no creation of new face topologies). Note that for a structure containing several faces, one can use a different algorithm for each face in order to vary the subdivision. Figure 17 shows an example of two faces having the same boundary but a different algorithm.

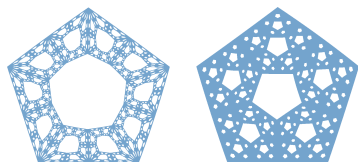


Figure 17: Two faces $\mathcal{F}_1(\partial, E, \mathcal{A}_1)$ and $\mathcal{F}_2(\partial, E, \mathcal{A}_2)$ with the same boundary $\partial = \{B_2, B_2, B_2, B_2, B_2\}$ and the same $E = (B_2, B_2, B_2)$. The algorithm \mathcal{A}_2 (left) creates more lacunas than \mathcal{A}_1 (right).

5 MEASURES ON FRACTAL STRUCTURES

To evaluate the variability of the lacunarity, one can use several measures like relative density [JWZ07], area, or length of boundary (perimeter). However, these measures strongly depend on the iteration level. To characterize fractal structures, evaluating the evolution of the area and the perimeter is convenient from one iteration to the following. The fractal dimension is especially recommended for such a structure. We propose an additional measure that characterizes the topological complexity of such structures. For the rest of the article, \mathcal{F}_0 is the face $\mathcal{F}_0(\partial_0, E, \mathcal{A}_1)$ with $\partial_0 = \{C_2, B_2, C_2, B_2, C_2, B_2\}$ and $E = (C_2, B_2, B_2)$. Its geometric realization is in Figure 18.

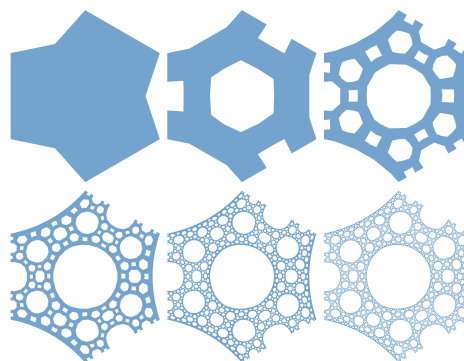


Figure 18: First five iteration levels of the face \mathcal{F}_0 we use for all geometry based measures. Top-left image is the primitive.

5.1 Area and perimeter

For lacunar fractal structures, the area A_i decreases, and the perimeter P_i increases with each iteration. We look at how these values are changing through iteration levels. For instance, with the face \mathcal{F}_0 , we compute A_i and P_i for all iterations. To normalize the results and see the factor F_A (respectively F_P) multiplying the area (respectively the perimeter) at each iteration, we compute A_i/A_0 and P_i/P_0 . Table 1 contains the values for the area and the perimeter of the face \mathcal{F}_0 for the iteration levels 0 to 5. Figure 19 (respectively 20) shows the variation of the area (respectively the perimeter) through iteration levels. We have $F_A = 0.76$ and $F_P = 1.97$, meaning the area decreases by 24% while the perimeter increases by 97% for each iteration level.

i	A_i	P_i	$\ln(A_i/A_0)$	$\ln(P_i/P_0)$
0	234.90	61.61	0	0
1	174.49	97.96	-0.30	0.46
2	132.50	185.63	-0.57	1.10
3	100.92	381.78	-0.84	1.82
4	76.92	807.93	-1.12	2.57
5	58.64	1725.87	-1.39	3.33

Table 1: Area and perimeter of the face \mathcal{F}_0 for each iteration level i .

5.2 Relative density

We compute the relative density [JWZ07] of a fractal structure at different scale levels by using an image of that structure and a mask with variable size. The idea is to have a local evaluation of the matter quantity. We drag the mask over the image for each pixel of the structure and count the number of pixels belonging to the structure after the mask application. Then, we divide the value by the surface of the mask. Finally, we color the image's pixel at the mask's center depending on the relative density found (between 0 and 1) using the colormap in Figure 21. Figure 22 contains the resulting image for the third iteration level of the face \mathcal{F}_0 for dif-

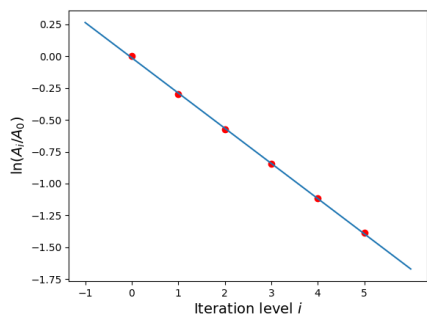


Figure 19: For each iteration level, the logarithm of the area over the area of the primitive for the face \mathcal{F}_0 . The area decreases by 24% for each iteration level.

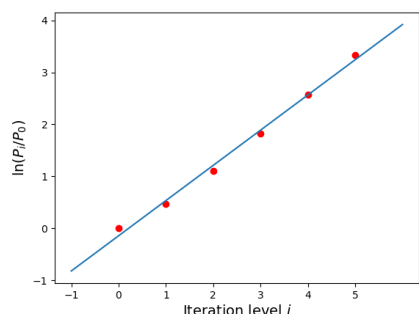


Figure 20: For each iteration level, the logarithm of the perimeter over the perimeter of the primitive for the face \mathcal{F}_0 . The perimeter increases by 97% for each iteration level.

ferent mask sizes. Figure 23 contains the relative density for the structure in Figure 13. The structure's left side has a higher density than the right side because of the delay in subdivision. The additional iteration adds more lacunas on the right side, reducing the relative density.



Figure 21: The colormap we use for the relative density. A value of 0 is mapped on the blue and a value of 1 is mapped on the red. Any other value between 0 and 1 is mapped linearly between blue and red.

5.3 Fractal dimension

Using the box-counting algorithm, we compute the fractal dimension and note it D_{box} [Man85]. The fractal dimension is the slope of the line when we plot the values of $\ln(N_\epsilon)$ on the Y-axis against the value of $\log(1/\epsilon)$ on the X-axis. We note ϵ the size of the boxes and N_ϵ the number of boxes of size ϵ that cover the geometry. For a given geometry, we set it in a squared grid of size one by one. We start with a box size of 1/16 since bigger box sizes are insignificant. We use the geometry of the face \mathcal{F}_0 (in Figure 18) to compute all geometry-based measures. With the fifth

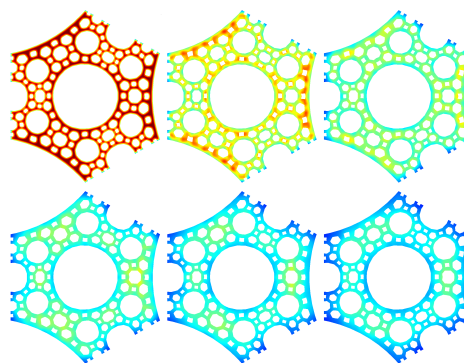


Figure 22: Relative density on the third iteration level of the face \mathcal{F}_0 for different mask sizes. In the reading direction, the mask sizes are 21, 51, 101, 151, 201 and 251 pixels. The image has a size of 825×953 pixels.

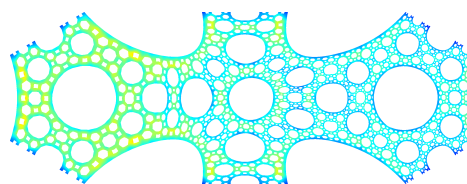


Figure 23: Relative density on the fractal structure in Figure 13 for a mask size of 101 pixels. The image has a size of 2607×953 pixels.

iteration level, we compute that the face has a fractal dimension of $D_{box} = 1.65$. Table 2 contains the values for each size of boxes, and Figure 24 represents the corresponding line.

ϵ	$\ln(1/\epsilon)$	$\ln(N_\epsilon)$
1/16	2.77	4.96
1/32	3.47	6.09
1/64	4.16	7.28
1/128	4.85	8.46
1/256	5.55	9.62
1/512	6.24	10.71
1/1024	6.93	11.80

Table 2: Box-counting method on the fifth iteration of the face \mathcal{F}_0 .

5.4 Topological measure of fractals

One cannot use the Euler-Poincaré characteristic to characterize the topology of a fractal structure because of the infinite increase of the number of lacunas. To address this question, we suggest computing the ratio limit between the number of lacunas and the sum of the number of lacunas and sub-faces. This measure does not depend on the geometric realization. For a given iteration level, we consider the space as a partition composed of sub-faces and lacunas. We quantify the ratio of the space lacunas occupy. Then, the topological measure of a fractal is the limit of this ratio when the iteration level tends to infinity. This ratio gives information on the number of created lacunas

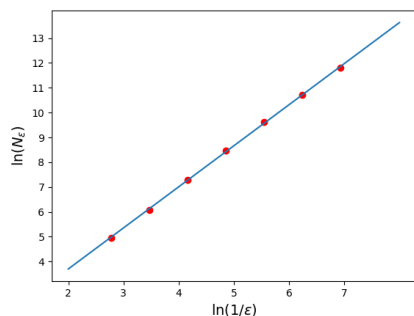


Figure 24: For the fifth iteration of the face \mathcal{F}_0 , the slope of the curve gives $D_{box} = 1.65$.

as a function of created sub-faces along the iterative process.

For a given iteration level i , we note the number of lacunas (respectively number of sub-faces) N_i^L (respectively N_i^C). We compute them recursively. For the primitive (when the iteration level is zero), there is no lacuna ($N_0^L = 0$). For the first iteration level, a no-delayed face has a central lacuna that counts as one lacuna. It can also have lacunas in its boundary due to C_n edges. We consider that each C_n edge creates $n - 1$ half lacunas. When the iteration level is more than zero, N_i^L is the face's number of lacunas at the first iteration level plus the number of lacunas of each sub-face at the previous iteration level.

About the number of sub-faces, when the iteration level is zero, there is one face. For any other iteration level, the number of sub-faces is the sum of the number of sub-faces for each sub-face at the previous iteration level. When the subdivision of a face \mathcal{F} gives n sub-faces \mathcal{F} , we compute the number of sub-faces with the equation 1 and the number of lacunas with the equation 2.

$$N_i^C = n^i \tag{1}$$

$$N_i^L = N_1^L + n \times N_{i-1}^L \tag{2}$$

For instance, we compute the values of N_i^C and N_i^L for the face \mathcal{F}_0 . Figure 10 describes this face subdivision. When there is no iteration, $N_0^L = 0$ and $N_0^C = 1$. The face has three C_2 edges, each of which induces one-half lacuna at the next iteration level. Hence at the first iteration level, N_1^L of $\mathcal{F}_0 = 1 + 3 \times 0.5 = 2.5$. The subdivision of \mathcal{F}_0 gives six sub-faces \mathcal{F}_0 , hence N_1^C of $\mathcal{F}_0 = 6$. For the second iteration level, we have N_2^L of $\mathcal{F}_0 = 2.5 + 6 \times 2.5 = 17.5$ and N_2^C of $\mathcal{F}_0 = 6 \times 6 = 36$. Since the subdivision of \mathcal{F}_0 gives six sub-faces \mathcal{F}_0 , we could compute N_2^C as 6^2 that is also 36.

Let the ratio R_i be:

$$R_i = \frac{N_i^L}{N_i^C + N_i^L}$$

Table 3 indicates the topology-based measures of \mathcal{F}_0 for the first five iteration levels. The curve of the ratio R_i of \mathcal{F}_0 for the first ten iteration levels is in Figure 25. Notice that the ratio has a finite limit. We found that the limit depends only on the parameter E of the face. The algorithm creates more and more characteristic faces that depend on these parameters. Hence, the limit R of R_i when i tends to the infinity of any face depends on the limit R of the specific faces that appear due to the subdivision algorithm. For a characteristic face that subdivides into itself and from equations 1 and 2, we have the following formula:

$$R = \lim_{i \rightarrow +\infty} R_i = \frac{N_1^L}{N_1^L + N_1^C - 1}$$

For instance, with the face \mathcal{F}_0 , the limit is:

$$R = \frac{2.5}{2.5 + 6 - 1} = \frac{1}{3}$$

i	N_i^L	N_i^C	R_i
0	0	1	0
1	2.5	6	0.294
2	17.5	36	0.327
3	107.5	216	0.332
4	647.5	1296	0.333
5	3887.5	7776	0.333

Table 3: Topology-based measures N_i^L , N_i^C and R_i of the face \mathcal{F}_0 for the iteration levels i from 0 to 5.

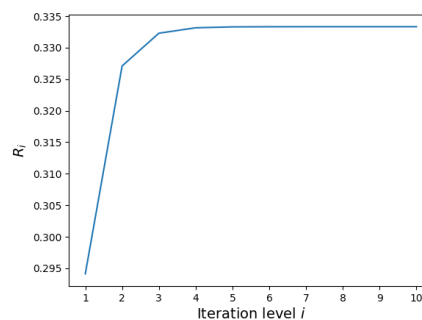


Figure 25: The ratio R_i of the face \mathcal{F}_0 for the iteration levels from 0 to 10.

5.5 Discussion

We created some faces that subdivide in themselves to study the impact of the algorithms and their parameter E in the lacunarity. The faces are the following:

- \mathcal{F}_0 .
- $\mathcal{F}_1(\partial_1, E_1, A_1)$ with $\partial_1 = \{C_2, C_2, C_2, C_2, C_2, C_2, C_2\}$ and $E_1 = (C_2, C_2, C_2)$.
- $\mathcal{F}_2(\partial_2, E_2, A_1)$ with $\partial_2 = \{B_2, B_2, B_2, B_2, B_2\}$ and $E_2 = (B_2, B_2, B_2)$.

- $\mathcal{F}_3(\partial_3, E_1, A_2)$ with $\partial_3 = \{C_2, C_2, C_2, C_2, C_2\}$.
- $\mathcal{F}_4(\partial_4, E_2, A_2)$ with $\partial_4 = \{B_2, B_2, B_2, B_2\}$.
- $\mathcal{F}_5(\partial_5, E_3, A_2)$ with $\partial_5 = \{B_3, B_3, B_3, B_3\}$ and $E_3 = (B_3, B_3, B_3)$.

Figure 26 contains the faces at the fourth iteration level. Table 4 lists those faces with the value of N_1^L , N_1^C , $\lim_{i \rightarrow +\infty} R_i$, the factors F_A and F_P , and the fractal dimension at the fourth iteration level of the cells.

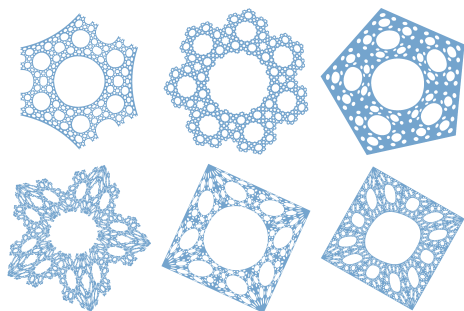


Figure 26: Some characteristic faces that subdivide into themselves. (Top left) The face \mathcal{F}_0 . (Top middle) The face \mathcal{F}_1 . (Top right) The face \mathcal{F}_2 . (Bottom left) The face \mathcal{F}_3 . (Bottom middle) The face \mathcal{F}_4 . (Bottom right) The face \mathcal{F}_5 .

Face	N_1^L	N_1^C	R	F_A	F_P	D_{box}
\mathcal{F}_0	2.5	6	0.33	0.76	1.97	1.72
\mathcal{F}_1	4.5	7	0.43	0.76	2.09	1.74
\mathcal{F}_2	1	5	0.20	0.85	1.69	1.82
\mathcal{F}_3	3.5	10	0.28	0.79	2.58	1.77
\mathcal{F}_4	1	8	0.12	0.75	1.92	1.72
\mathcal{F}_5	1	12	0.08	0.81	2.40	1.77

Table 4: List of measures for each face in Figure 26. There are the number of lacuna and the number of sub-faces at the first iteration, the R value that is the limit of R_i when i tends to infinity, the factors F_A and F_P , and the fractal dimension.

Globally, the set of parameters we used to provide the different fractal faces impacts our topological measure R . The values depend on the subdivision algorithm and the parameter E . They reflect the specificity of the face lacunarity.

On the other hand, geometry measures are more delicate to interpret because they depend on geometric realization. However, we can observe that for two faces having an identical value of F_A , the value of F_P is higher when the parameter E contains the most C_n edges. Intuitively, C_n edges induce lacunas while B_n edges induce no lacunas.

Generally, the higher the fractal dimension, the higher the F_A and the lower the F_P , even if the fractal dimension is relatively homogeneous for these examples.

However, for an identical value of F_A , the fractal dimension increases with the value of F_P .

6 CONCLUSION AND FUTURE WORK

Automatic construction of fractal structures while controlling the lacunarity is challenging because of the variety and complexity of possible topologies.

We propose an automatic method defining the topology of fractal faces's subdivision, depending on the topology of their boundary and additional parameters controlling the lacunarity. Our method presents three main advantages. First, a few intuitive sets of parameters define the type of lacunarity. Second, we base our construction on the BC-IFS model, which codes the topology of the fractal structure. It ensures the consistency of the limit fractal topology and over iterations. The resulting model can be exported in STL format and directly printed in additive manufacturing (see Figure 27). This topological approach allows us to adapt the lacunar geometry to intended applications. Third, as we define a lacunar face from its boundary, we can directly create complicated assemblies without additional constraints, with each face being defined from the edges of its neighborhood faces.

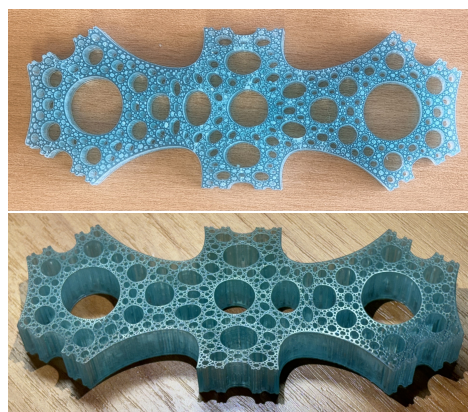


Figure 27: 3D print of the fractal structure presented in Figure 13 and 14.

We complete this approach by introducing a delay in the subdivision process, allowing the cohabitation in an assembly of different iteration levels of an identical lacunar fractal. We use this extension to produce a homogeneous lacunarity over faces with different sizes or to manage variation lacunarity. We can extend this approach by proposing additional subdivision algorithms and introducing more than one main lacunar, for example.

We analyze the lacunarity of the resulting structures by evaluating and proposing different measures for geometric and topologic characteristics.

Future works focus on an automatic process to subdivide volumes. It is more complicated than faces since

the adjacency of two volumes is on faces, with more topology possibilities than edges. As for faces, we can subdivide each volume edge (faces) and create sub-volumes for each sub-face. However, there could be several adjacency faces and lacuna faces for each sub-volume. The complexity induced by the topological cellular decomposition of 3D fractal structures needs deep analysis to identify, parametrize, and automatize the subdivision process while guaranteeing topological consistency.

7 REFERENCES

- [AALS17] Niels Aage, Erik Andreassen, Boyan S Lazarov, and Ole Sigmund. Giga-voxel computational morphogenesis for structural design. *Nature*, 550(7674):84–86, 2017.
- [AC15] Lawrence M. Anovitz and David R. Cole. Characterization and Analysis of Porosity and Pore Structures. *Reviews in Mineralogy and Geochemistry*, 80(1):61–164, 2015.
- [AJS18] Diab W. Abueidda, Iwona Jasiuk, and Nahil A. Sobh. Acoustic band gaps and elastic stiffness of pmma cellular solids based on triply periodic minimal surfaces. *Materials & Design*, 145:20–27, 2018.
- [AKA21] Oraib Al-Ketan and Rashid K. Abu Al-Rub. MSLattice: A free software for generating uniform and graded lattices based on triply periodic minimal surfaces. *Material Design and Processing Communications*, 3(6):1–10, 2021.
- [Ban06] J. Banhart. Metal Foams: Production and Stability. *Advanced Engineering Materials*, 8(9):781–794, September 2006.
- [Bar90] Anthony Barcellos. Fractals everywhere. by michael barnsley. *The American Mathematical Monthly*, 97(3):266–268, 1990.
- [BEH89] Michael F. Barnsley, John H. Elton, and Douglas P. Hardin. Recurrent Iterated Function Systems. *Constructive Approximation*, pages 3–31, 1989.
- [BRBH19] Brent R Bielefeldt, Gregory W Reich, Philip S Beran, and Darren J Hartl. Development and validation of a genetic l-system programming framework for topology optimization of multifunctional structures. *Computers & Structures*, 218:152–169, 2019.
- [CLLZ21] Liang-Yu Chen, Shun-Xing Liang, Yujing Liu, and Lai-Chang Zhang. Additive manufacturing of metallic lattice structures: Unconstrained design, accurate fabrication, fascinated performances, and challenges. *Materials Science and Engineering: R: Reports*, 146:100648, October 2021.
- [Esp12] Laura Espinal. Porosity and Its Measurement. In *Characterization of Materials*, pages 1–10. John Wiley & Sons, Ltd, 2012. _eprint: <https://onlinelibrary.wiley.com/doi/pdf/10.1002/0471266966>
- [FFL⁺18] Jiawei Feng, Jianzhong Fu, Zhiwei Lin, Ce Shang, and Bin Li. A review of the design methods of complex topology structures for 3d printing. *Visual Computing for Industry, Biomedicine, and Art*, 1(1):1–16, 2018.
- [GGS21] Christian Gentil, Gilles Gouaty, and Dmitry Sokolov. *Geometric Modeling of Fractal Forms for CAD*. John Wiley & Sons, 2021.
- [Hut81] John E Hutchinson. Fractals and self similarity. *Indiana University Mathematics Journal*, 30(5):713–747, 1981.
- [JWZ07] Bin Jiang, Zejun Wang, and Naiqin Zhao. Effect of pore size and relative density on the mechanical properties of open cell aluminum foams. *Scripta Materialia*, 56(2):169–172, January 2007.
- [JZSL05] B Jiang, N Zhao, C Shi, and J Li. Processing of open cell aluminum foams with tailored porous morphology. *Scripta Materialia*, 53(6):781–785, September 2005.
- [KT10] X.Y. Kou and S.T. Tan. A simple and effective geometric representation for irregular porous structure modeling. *Computer-Aided Design*, 42(10):930–941, October 2010.
- [LC19] Yuan Liang and Gengdong Cheng. Topology optimization via sequential integer programming and canonical relaxation algorithm. *Computer Methods in Applied Mechanics and Engineering*, 348:64–96, 2019.
- [LJP⁺19] Y. Li, H. Jahr, P. Pavanram, F.S.L. Bobbert, U. Puggi, X.-Y. Zhang, B. Pouran, M.A. Leeftang, H. Weinans, J. Zhou, and A.A. Zadpoor. Additively manufactured functionally graded biodegradable porous iron. *Acta Biomaterialia*, 96:646–661, September 2019.
- [LVSZ21] Nikita Letov, Pavan Velivela, Siyuan Sun, and Yaoyao Zhao. Challenges and Opportunities in Geometric Modelling of Complex Bio-Inspired 3D Objects Designed for Additive Manufacturing. *Amer-*

- ican Society of Mechanical Engineers*, page 10, 2021.
- [Man85] Benoit B Mandelbrot. Self-affine fractals and fractal dimension. *Physica scripta*, 32(4):257, 1985.
- [MBZ⁺20] T. McNulty, D. Bhate, A. Zhang, M. A. Kiser, L. Ferry, A. Suder, S. Bhattacharya, and P. Boradkar. A framework for the design of biomimetic cellular materials for additive manufacturing. *Solid Freeform Fabrication 2017: Proceedings of the 28th Annual International Solid Freeform Fabrication Symposium - An Additive Manufacturing Conference, SFF 2017*, (2):2188–2200, 2020.
- [MGM⁺18] Mehrdad Mohsenizadeh, Federico Gasbarri, Michael Munther, Ali Beheshti, and Keivan Davami. Additively-manufactured lightweight metamaterials for energy absorption. *Materials & Design*, 139:521–530, 2018.
- [Pen10] Deborah Pence. The simplicity of fractal-like flow networks for effective heat and mass transport. *Experimental Thermal and Fluid Science*, 34(4):474–486, 2010.
- [PHL20] Chen Pan, Yafeng Han, and Jiping Lu. Design and optimization of lattice structures: A review. *Applied Sciences*, 10(18), 2020.
- [SGGM15] Dmitry Sokolov, Gilles Gouaty, Christian Gentil, and Anton Mishkinis. Boundary controlled iterated function systems. In *Curves and Surfaces: 8th International Conference, Paris, France, June 12-18, 2014, Revised Selected Papers 8*, pages 414–432. Springer, 2015.
- [SHR97] Bernard Sapoval, Olivier Haeberlé, and Stephanie Russ. Acoustical properties of irregular and fractal cavities. *The Journal of the Acoustical Society of America*, 102(4):2014–2019, 1997.
- [TGM⁺09] Olivier Terraz, Guillaume Guimberteau, Stéphane Mérillou, Dimitri Plemenos, and Djamchid Ghazanfarpour. 3Gmap L-systems: An application to the modelling of wood. *Visual Computer*, 25(2):165–180, 2009.
- [TMV⁺16] Mary Kathryn Thompson, Giovanni Moroni, Tom Vaneker, Georges Fadel, R Ian Campbell, Ian Gibson, Alain Bernard, Joachim Schulz, Patricia Graf, Bhrihu Ahuja, et al. Design for additive manufacturing: Trends, opportunities, considerations, and constraints. *CIRP annals*, 65(2):737–760, 2016.
- [Zha11] Li-Zhi Zhang. Heat and mass transfer in a randomly packed hollow fiber membrane module: a fractal model approach. *International Journal of heat and mass transfer*, 54(13-14):2921–2931, 2011.
- [ZT96] Chems Eddine Zair and Eric Tosan. Fractal modeling using free form techniques. *Computer Graphics Forum*, 15(3):269–278, 1996.
- [ZZ13] Fuzong Zhou and Xinrong Zhang. Assessment of heat transfer in an aquifer utilizing fractal theory. *Applied thermal engineering*, 59(1-2):445–453, 2013.

End-to-End Move Prediction System for Indoor Rock Climbing: Beta Caller

Kevin Cardenas
University of Colorado
Colorado Springs
1420 Austin Bluffs Pkwy
Colorado Springs, CO 80907
kcarden4@uccs.edu

Sudhanshu Semwal
University of Colorado
Colorado Springs
1420 Austin Bluffs Pkwy
Colorado Springs, CO 80907
ssemwal@uccs.edu

James Maher
United States Air Force
Academy
2345 Fairchild Dr
USAFA, CO 80840
james.maher@afacademy.af.edu

ABSTRACT

We developed Beta Caller, an end-to-end system supporting the sport of rock climbing for climbers with visual impairment. Beta Caller provides real-time, audible instructions containing a prediction for the climber's next move while they are actively climbing a rock wall. This system leverages computer vision techniques to collect key information about the climber's environment, enabling Beta Caller to make move predictions on climbing walls it has never encountered before. Neural networks are used to predict where the climber should move next, based on information provided by the computer vision models. The predicted move is translated into a verbal message guiding the climber to the next hold and then transmitted via wireless headphones using a text-to-speech model. This novel idea makes one of the fastest growing sports in the world even more appealing and approachable to climbers with visual impairment, however, this tool can be utilized by all climbers to improve their climbing skills. Beta Caller achieved 80.08% accuracy predicting which limb the climber should move next and, when predicting the location of the next hold, Beta Caller achieved a bounding box error of only 6.79%. These results pioneer a strong foundation shaping the future landscape of rock climbing prediction tools for visually impaired climbers.

Keywords

artificial intelligence, computer vision, human computer interaction, object detection, pose estimation, human motion prediction, rock climbing

1 INTRODUCTION

Rock climbing is one of the fastest growing sports in the world, exploding in popularity in the last few years. With inspiring and alluring documentaries like *The Dawn Wall* [Low17a], *Free Solo* [Vas18a], *Valley Uprising* [Mor14a], and many more, coupled with the sport's Olympic debut in the 2020 Summer Olympics, rock climbing grew exponentially in the last few years and captivated the interest of millions of people worldwide. Even before these catalyzing events, the sport of rock climbing garnered interest across the globe due to the sport's ability to concurrently challenge participants physically and mentally. The sport initially interested a niche group of outdoorsy, free-spirited people, but now captivates the attention of a wide-spread, diverse audience. For those less familiar with this sport, definitions

of several rock climbing terms whose meaning may not be innately understood are provided in Table 1.

Hold	3D plastic grip for a climber to grab with their hand or step on with their foot
Move	Transition of hand or foot to the next hold
Route	A sequence of moves, typically using a single hold color, to get to the top of the wall
Beta	Information about how to complete a route
Caller	Someone providing instructions to a climber

Table 1: Rock Climbing Terminology

A large population of athletes with varying disabilities competitively participate in sports, and rock climbing is no exception. Climbers with disabilities have found many innovative and unique ways to push and pull their body up a wall to get to the top. There have been many technological advancements in prosthetics and various rock climbing gear to make the sport more approachable, inclusive and enjoyable to a wider scope of abilities. However, technology assisting visually impaired climbers remains grossly under-researched. The most common and "advanced" way for a visually impaired climber to participate in the sport is to have someone, who is typically a specially trained coach, on the ground verbally coaching the climber (known as the "caller")

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

by providing them information (commonly referred to as “beta”) about how to ascend the wall. In addition to hearing the commands for the next move, visually impaired climbers use their palms to scan the wall for rock holds. This current approach is exceedingly challenging, time consuming and may be intimidating to a new climber and to the caller who is coaching the climber up the wall. The caller’s task is arguably the most important and requires significant climbing experience, spatial estimation skills and quick thinking in order to process the climber’s body position, what holds are available to them, where they should move next, and finally call out a command to guide the climber to the next hold.

This paper introduces the application of computer vision and mobile media technology to a revolutionary sensory substitution device, Beta Caller, that leverages Artificial Intelligence (AI) to aid rock climbers of all abilities to better enjoy and compete in the sport. Beta Caller combines two computer vision models to understand key information about the climber’s environment (where holds are located on the wall and the climber’s body pose), as well as a combination of neural networks to provide a real-time prediction of the climber’s next move. Beta Caller uses a text-to-speech model to transmit an audible message containing coaching information in a timely and useful manner to enable the climber to quickly and successfully complete the next move and ascend to the top of the wall. In addition to this novel system, our paper contributes a rock climbing movement dataset that, to the best of our knowledge, stands as the sole resource of its kind.

2 RELATED WORK

Despite the recent interest in rock climbing, research within the sport is fairly limited in quantity, depth and scope. A small amount of rock climbing research focuses on better understanding how rock climbers move their bodies [Pfe11a][Ouc10a][Sib07a][Wei14a], and additional research describes a handful of real-time climbing tools to make the sport more enjoyable by helping climbers efficiently move through a route [Kos17a][Kos17b]. A majority of climbing research, however, aims to provide performance feedback to the climber after their workout session concludes in order to evaluate how well they trained instead of providing feedback while the climber is on the rock wall [Lad13a][Bre23a][Kos15a].

One example of a performance feedback tool was created by Ekaireb et al. to assess video footage of a climber using computer vision and machine learning to provide a report of how well the person climbed a route [Eka22a]. Their research conducts image segmentation creating a mask of the raw image only containing the climbing wall and excluding the background. Additionally, information about the climber’s environment is

gathered using an object detection model to locate all of the holds on the climbing wall and then a pose estimation model identifies all of the climber’s joint keypoints. This information is collected for each video frame and, after the climb is completed, these frames are evaluated to provide the climber with feedback about how well they climbed the route. Beta Caller improves the system created by Ekaireb et al. by collecting information about the climber’s environment and training a model to provide climbers with real-time movement predictions on any indoor rock climbing wall.

2.1 Assistive Technology for Visually Impaired Rock Climbers

Richardson et al. developed a real-time, assistive climbing system to guide visually impaired climbers [Ric22a]. Climb-o-Vision uses a helmet-mounted, computer vision system that identifies where holds are located on the climbing wall and provides tactile feedback on the climber’s tongue to guide them to a hold. This system is able to identify holds only where the climber’s helmet is facing and does not provide guidance to the climber where the next best hold is located. Electrotactile tongue interfaces have advanced the capabilities of vision substitution, however this research did not provide any conclusions evaluating the effectiveness of the tongue interface during climbing to guide a climber’s hand to holds.

Ramsay and Chang [Ram20a] developed a real-time climbing tool to assist visually impaired climbers as they are actively climbing using a body pose sonification system. They developed a tool to provide the climber with an auditory command containing the location of the next hold on the wall. Additionally, the tool produces a tone for the climber that changes pitch based on the distance from the climber’s hand to the next hold. However, this tool is limited to a specific, standardized climbing wall called a MoonBoard. A MoonBoard is a small (7.5ft wide by 10.5ft tall) climbing wall used with the sole purpose of training. This board is filled with evenly-spaced holds each with an LED light. When lit, the LED lights identify the holds the climber is allowed to use creating the intended route. The primary benefit of this board is that there are over 6,000 pre-programmed routes in a small space. The MoonBoard was created to provide advanced rock climbers with a training tool. The easiest route on a MoonBoard is significantly more challenging than the most routes in an indoor rock climbing gym, therefore it is not intended for use by a majority of the climbing community.

Ramsay and Chang’s research [Ram20a] capitalized on the known, uniform placement of holds on the MoonBoard, eliminating the need for object detection to identify the location of holds on the rock climbing wall. Additionally, the use of a MoonBoard automatically provides known move sequences, so there is no need to

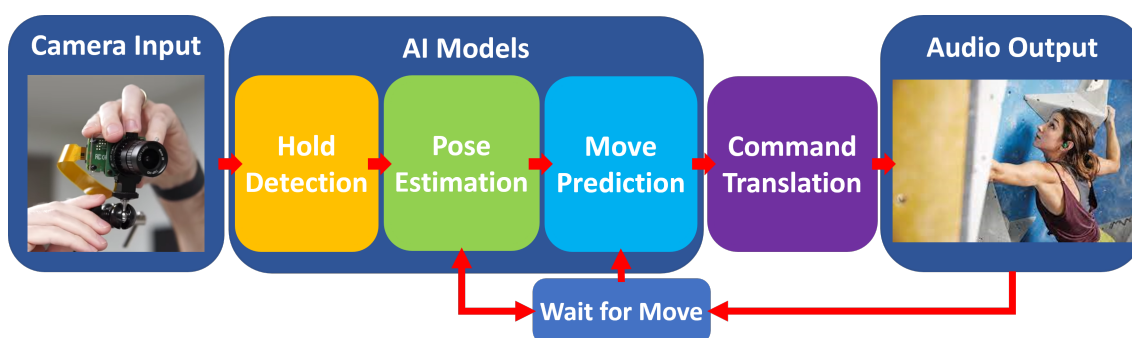


Figure 1: Beta Caller Architecture

create a move prediction model to intelligently predict where the climber should move next or which limb they should move. Ramsay and Chang solely focused on using a pose estimation model to identify the climber’s joint keypoints in order to provide the climber with an auditory command of how to move to the next known hold. Their results were successful when demonstrated on a vertical Twister game mat representing a simulated climbing wall. Beta Caller significantly augments this work outside of the confines of the MoonBoard by generalizing the nature of the system to work on any indoor rock climbing wall. This requires Beta Caller to be able to accurately identify hold locations in addition to the climber’s pose, as well as predict the next best hold for the climber to move to and which hand should move to that hold. To the best of our knowledge, there does not exist any research into real-time systems that predict where a climber should move next.

3 METHODOLOGY

Beta Caller gathers information about the climber’s environment and provides real-time, audible commands containing a prediction for the climber’s next move while they are actively climbing. In practice, the proposed system allows a climber to point a camera at the climbing wall to guide them to the top by providing a series of commands to accomplish each next move.

3.1 System Architecture

Beta Caller comprises three primary components: video camera input, a suite of AI models, and a text-to-speech model providing an audible command to the climber via wireless headphones (Figure 1).

Beta Caller runs on a laptop which is stationed on the ground with a camera facing the climber on the climbing wall. The first two models utilize computer vision techniques to gather key information about the climber’s environment. The first model employs object detection to identify where the holds are located on the climbing wall. The second model uses pose estimation to locate the climber’s pose by identifying joint keypoints on the climber’s body. The last model, the core

of Beta Caller, uses information about the climber’s environment to predict where the climber should move next. Specifically, a combination of neural networks were built to predict which limb the climber should move and, if either hand is predicted to move next, to which hold the climber should move that limb.

After a move is predicted, simple trigonometry is used to calculate the direction the climber should move their hand to the next hold. Additionally, the distance between the climber’s wrist and the next hold is calculated. Using this information, Beta Caller calls out which hand the climber should move, the angle which the climber should move their hand, and the distance from their wrist to the predicted hold (e.g. “Right hand. Two o’clock. About two feet.”). This command is played using a text-to-speech engine, *pyttsx3*, and enables the climber to listen to the output using wireless headphones connected to the laptop. Once the climber receives this command, Beta Caller waits for the climber to complete the move by continuously tracking the climber’s pose. Once the move is completed, Beta Caller makes another prediction where the climber should move to and transmits that command to the climber. This process is repeated until the climber finishes the entire route.

3.2 Data Pipeline

To the best of our knowledge, there does not exist a rock climbing dataset suitable for predicting climbing movement, so we constructed a dataset with over 4,100 images collected as image sequences from over 250 videos where each frame is saved after a completed move. The images are named according to the video they correspond to followed by the time sequence (e.g. 01-01.jpg). These images are sent through a data pipeline to gather features and labels to later be used to train the move prediction model.

The first step in the data pipeline is to run inference on each image using two computer vision models to gather key information about the climbing wall and the climber’s body position. Beta Caller uses one of the leading and most widely used object detection models, YOLOv8 [Red16a], to train a custom object detection

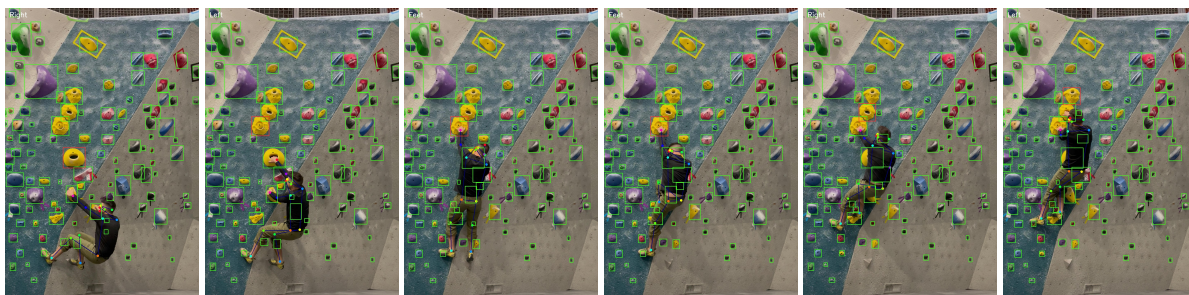


Figure 2: Results of Data Pipeline for Image Labeling

model to identify indoor rock climbing holds. In addition to object detection, Beta Caller utilizes the state of the art pose estimation model, ViTPose [Xu22a]. ViTPose is a vision transformer model that has an encoder to extract image features and a lightweight decoder to conduct pose estimation. Beta Caller uses ViTPose to get the x - y coordinates of 17 joint keypoints from the climber's body. The data collected from each of these computer vision models is illustrated in Figure 3.



Figure 3: Data Output from Computer Vision Models

Hold predictions can be drawn with bounding boxes around each hold found in the image and joint keypoints are drawn as dots accompanied by lines connecting the keypoints, providing a stick figure of the climber. The illustrations of the computer vision models' results shown in Figure 4 are used solely for demonstrative and validation purposes because Beta Caller will not need to visually display the holds and pose, rather use that information to provide an audible command to the climber.



Figure 4: Computer Vision Models' Results

After running inference on an image, the output data is added to a CSV file. The data output from these models are two data structures containing normalized x - y coordinates of the holds within the image and the climber's joint keypoints. The holds data structure contains a row for each hold detected and four columns representing the bounding box coordinates ($x_{min}, y_{min}, x_{max}, y_{max}$). The pose data structure contains a row for each of the 17 joint keypoints and two columns representing the keypoint's x and y coordinates.

The data pipeline uses these two data structures for both the current image and the previous image from the image sequence to compute and identify the labels for the previous image. There are five labels in the dataset: which limb moved and the bounding box coordinates of the hold the climber moved their limb to. Beta Caller combines both feet together into a single class because the primary focus of calling moves to a climber is to direct them to the best hand holds. Focusing on accurate hand predictions provides the climber more time, using less energy, to find the best holds for their feet to prepare them for the next hand movement. Additionally, providing specific guidance for four limbs instead of two might easily become overwhelming for a climber. Previous experience climbing with people with visual impairment confirms the necessity of predicting the correct hand to move as well as when the climber is required to move their feet up to prepare for the next hand movement. Therefore, the limb label will be assigned the value left hand, right hand or feet.

In order to validate the data pipeline, labels are drawn on each image. The top left corner of the image contains text for which limb will move and a red bounding box is drawn around which hold the climber will move their limb to. For example, the first image in the Figure 2 sequence is labeled indicating the climber will move their right hand to the red box because this outcome is observed in the subsequent image from the sequence.

The final dataset prepared for the move prediction models is in CSV format containing 34 features, representing the normalized x - y coordinates for each of the 17 joint keypoints, and 5 labels ($limb, x_{min}, y_{min}, x_{max}, y_{max}$) for over 4,100 images. The dataset was split into two subsets: training and testing. Specifically, 90% of the

data was used for training the model and 10% was reserved for testing its performance. During training, 30% of the training data was used for validation.

3.3 Move Prediction

Two dense neural networks are trained to predict the climber's next move. The first is a multi-class classification network used to predict which hand the climber should move or if they should move their feet. The model architecture is illustrated in Figure 5. The input layer contains 34 input neurons matching the number of features in the dataset. The first hidden layer contains 256 neurons and the second contains 128 neurons and they both use the ReLU activation function. Finally, the output layer contains three neurons (left, right, feet) and uses the Softmax activation function to create simulated prediction probabilities for each class. This model uses the Adam optimizer, the Categorical Cross-Entropy loss function, as well as prediction accuracy for the performance metric.

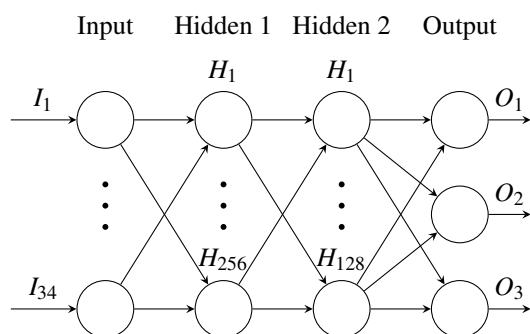


Figure 5: Limb Prediction Neural Network

Combining both the left and right foot movement into a single class causes a class imbalance where each hand represents 25% of the training dataset and the feet class contains 50%. In order to maintain the largest number of data points, Synthetic Minority Over-sampling Technique (SMOTE) [Cha02a] is used to balance the three classes. This technique creates additional, synthetic samples similar to the left and right hand observations to match the number of feet observations.

The second neural network is a multi-output regression network used to predict a bounding box ($x_{min}, y_{min}, x_{max}, y_{max}$) where the climber should move their hand next. Similar to the limb prediction model, the input layer contains 34 input neurons and this model uses five hidden layers. The hidden layers contain 256, 128, 64, 32 and 16 neurons, all using the ReLU activation function. The output layer consists of four neurons ($x_{min}, y_{min}, x_{max}, y_{max}$) activated by the Sigmoid function. This model uses the Adam optimizer, the Mean Squared Error (MSE) loss function, and Root Mean-Squared Error (RMSE) as the performance metric.

3.4 Command Translation

After the next move is predicted, Beta Caller continues in one of two ways depending on which limb is predicted. If the climber's feet are predicted to move, Beta Caller immediately communicates to the climber to move their feet upwards to a nearby hold. If one of the climber's hands is predicted to move, a series of computations must occur first to translate the move predictions into a usable command. For a majority of visually impaired climbers, a usable command contains which hand they should move along with what direction and how far they should move that hand.

To translate a prediction, Beta Caller first maps the predicted hold location to the nearest actual hold on the climbing wall found by the object detection model. Using this hold's center point and the climber's nose keypoint, the angle between the two coordinates is calculated and converted into an hour hand on a clock. The 12 o'clock position represents 0 degrees and each additional 30 degrees increments the hour hand by one.

In addition to the direction the climber should move their hand, the distance between the climber's predicted hand and the next hold is calculated using the climber's wrist keypoint and the hold's center point. The hold location and climber's joint keypoints are all represented by pixel values, so the pixel distance can be calculated easily. However, physical distance is required. In order to convert the pixel distance into physical distance, a known physical distance within the image must be used to create a conversion factor.

The distance between the climber's eyes, known as pupillary distance, is a distance on the human body that has the least amount of variance between human subjects. Limb length, waist or shoulder width, and essentially all other human body measurements vary greatly from human to human. However, the average pupillary distance is 2.5 inches and only ranges from 2-3 inches for all humans [Whi22a]. Therefore, the average pupillary distance is used as a known physical distance to create a conversion factor for calculated pixel distances. Despite the climber facing away from the camera, ViT-Pose is able to make an accurate prediction of where the person's eyes are located.

The exact distance, in feet and inches, is not the easiest for a climber to process, understand and move their body. Beta Caller converts the distance to the next hold to a more usable approximation of distance in order to simplify the command and not overwhelm the climber. Table 2 provides examples of these simplifications. Once the distance is known, all three parts of a command (hand, direction, and distance) are transmitted to the climber using a text-to-speech conversion library, *pyttsx3*. For example, a climber might hear "Right hand. Two o'clock. About two feet."

Feet	Inches	Move Distance
0	6	6 inches
2	0 - 3	About 2 feet
2	4 - 9	2 and a half feet
2	10 - 11	About 3 feet

Table 2: Distance Simplification Examples

4 RESULTS

An experiment was conducted for both the limb prediction and hold prediction neural networks with the goal of creating the best network architecture by trying three pose estimation models and changing the number of hidden layers and neurons.

The three pre-trained pose estimation models used in the experiment are MediaPipe [Goo24a], YOLOPose [Maj22a], and ViTPose [Xu22a]. The accuracy of the pose estimation model is of utmost importance as its outputs serve as crucial inputs for both the limb and hold prediction models. In order to find the best network architecture, the number of hidden layers was varied from one to five layers. Each additional hidden layer contains half of the number of neurons from the preceding hidden layer. The experiment includes networks where the first hidden layer contains 256 neurons, as well as networks that start with 128 neurons in the first hidden layer. In total, thirty different configurations were tested to find the best limb prediction model and ten configurations for the hold prediction model.

The results of the experiment are summarized in Table 3. The most accurate configurations obtain 80.08% limb prediction accuracy and both use ViTPose joint keypoints for limb prediction. One network contains two hidden layers with 256 and 128 neurons and the other network has four hidden layers containing 128, 64, 32, and 16 neurons. With the exception of one model architecture, using ViTPose for features for limb prediction always led to the highest accuracy. As a result, ViTPose data was used for all models in the hold prediction experiment. The best hold prediction model configuration reached a bounding box RMSE of 0.0679. The input and output data use normalized pixels between 0 and 1, so this RMSE represents the percentage of error from the predicted bounding box to the actual hold. The most accurate neural network contains five hidden layers with 256, 128, 64, 32 and 16 neurons and achieves a bounding box prediction error of 6.79%.

In order to validate the move predictions made by both neural networks, the ViTPose data from an image is used for inference with the two trained models. The predicted limb is written in the top left corner of the image and an yellow box is drawn to show the predicted coordinates where the climber should move their hand. A red box is drawn to indicate the closest hold to the predicted coordinates. The final inference results can be visualized in Figure 6. In the left image, the move

prediction models predict the climber will move their left hand to the yellow box which is mapped to the hold outlined with a red box. The right image confirms an accurate prediction that the climber did move their left hand to that predicted hold.

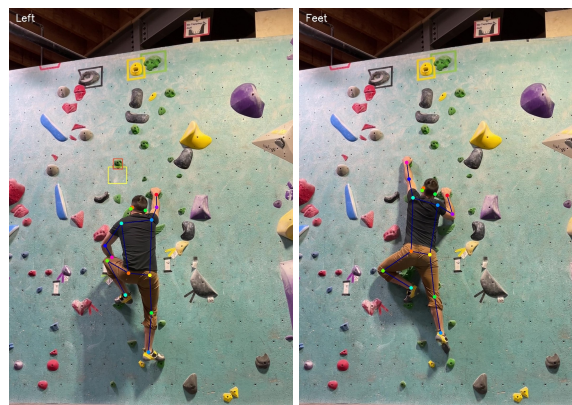


Figure 6: Beta Caller Prediction Results

Quantitative metrics describing the models' ability to accurately predict which limb the climber should and to where the climber should move their limb should not be the primary focus of evaluating the effectiveness of Beta Caller. Rock climbing, like dancing, can be more of an art than a science at times and provides climbers the option of approaching a route differently depending on the climber's body type, climbing style, flexibility, and numerous other factors. This results in a single route being climbed in innumerable diverse ways and establishes the fact that a correct sequence of moves does not exist, rather there are better singular moves than others.

In practice, a human caller will provide an "incorrect" command for which limb the climber should move or where they should move that limb, or both. However, this is merely the result of the caller's climbing style and how they would approach the route. Fortunately, the "incorrect" command will not impede the climber from continuing to move up the wall. Therefore, when Beta Caller's limb prediction model incorrectly predicts a limb to move, it is no different than the human caller and the climber will still be able to find a hold to move to even with the incorrectly predicted limb.

The hold prediction model's error is greatly reduced when Beta Caller maps the predicted bounding box to the closest hold found by the object detection model. However, even with the known location of a hold it is challenging for *any* climber to hear the command "Right hand. Two o'clock. About two feet." and move their hand to the exact location of the hold because the clock direction is rounded to the nearest hour and the distance is rounded to the nearest half foot. This style of command provides an appropriate amount of information without overwhelming the climber with too much

Hidden Layers	Hidden Neurons	Limb Accuracy			Hold RMSE
		MediaPipe	YOLOPose	ViTPose	ViTPose
1	256	69.74	68.94	73.18	0.0696
	128	69.30	68.32	67.05	0.0702
2	256, 128	72.81	72.67	80.08	0.0686
	128, 64	71.93	68.94	73.95	0.0687
3	256, 128, 64	69.30	69.57	77.78	0.0680
	128, 64, 32	72.81	68.94	79.31	0.0684
4	256, 128, 64, 32	71.49	69.57	79.69	0.0682
	128, 64, 32, 16	72.37	65.97	80.08	0.0690
5	256, 128, 64, 32, 16	71.49	65.97	77.78	0.0679
	128, 64, 32, 16, 8	71.93	64.60	78.16	0.0692

Table 3: Limb Prediction and Hold Prediction Experiment Results

detail. Fortunately, visually impaired climbers are especially talented at scanning the wall with their palms to find a hold nearby.

Without existing systems for comparison, these results provide a solid foundation, establish strong baselines, and pave the way for future rock climbing prediction systems.

5 CONCLUSION

This paper provided an operational prototype of an end-to-end system to revolutionize rock climbing for people of all skill levels, especially those with visual impairment. Beta Caller leverages a compilation of AI models to assist rock climbers up a wall to better enjoy and compete in the sport. Beta Caller successfully combines two state-of-the-art computer vision models for object detection and pose estimation and uses two neural networks to predict which limb the climber should move, as well as the next best hold for a climber to move to. Achieving limb prediction accuracy of 80.08% and predicting the next hold with only 6.79% error, Beta Caller creates a vital audible command containing the predicted move’s information in a timely and useful way to enable the climber to make the next move and ascend to the top of the rock climbing wall.

6 FUTURE WORK

Current accuracy for the limb prediction and hold prediction neural networks are sufficient to demonstrate proof of concept and create a robust starting point for rock climbing movement prediction. However, these results have the potential to be more accurate. Dense neural networks are the simplest of feed-forward networks due to their connections from all neurons to all neurons between each layer. The images used as input for the dense neural networks are currently treated as individual, independent observations and they exclude the information observed from the previous frame or multiple frames. The sport of rock climbing is inherently sequential. For example, if a climber moves both

of their feet, the next move will likely be one of their hands, not their feet again. In order to more accurately model this sequential nature of rock climbing, a recurrent neural network or transformer network can be used to leverage the capability of making strong sequence-to-sequence predictions.

Beta Caller was built on a smaller dataset, so a promising avenue for future work involves collecting more data to fine-tune the object detection, pose estimation, and move prediction neural networks. This iterative fine-tuning process holds the potential to significantly enhance the accuracy and versatility of these models, ensuring their effectiveness across various climbing environments and accommodating diverse body types and climbing styles.

In order to establish a conversion ratio for estimating physical distance from pixel distance, Beta Caller utilizes the climber’s pupil distance. It is imperative to validate the accuracy of these estimates through empirical testing. While this conversion methodology holds promise for its simplicity and applicability, its real-world performance should be evaluated across various rock climbing walls. Testing should assess the accuracy of the derived physical distances compared to ground truth measurements. Additionally, the robustness of the conversion ratio should be tested under different lighting conditions, camera angles, and camera distances away from the wall.

Another opportunity for future research could involve a comparative study evaluating the efficacy of Beta Caller, contrasted with the performance of a human caller providing beta to the same climber on the same routes. Speed, accuracy, and ease of understanding could be used as metrics to assess the climber’s response to Beta Caller predictions compared to human caller predictions. Such a study would not only offer insights into the system’s effectiveness but also shed light on the climber’s interaction with inclusive technologies and enhance our understanding of the capabilities and adaptations within the visually impaired community.

7 ACKNOWLEDGMENTS

Our thanks to the United States Paralympic Triathlon Team for teaching us how to guide a climber with visual impairment up a wall and to CityROCK for providing a safe and welcoming gym space to collect data and for allowing one of the authors to climb amazing routes.

8 REFERENCES

- [Bre23a] Breen, M. Reed, T., Nishitani, Y., Jones, M., Breen, H.M., and Breen, M.S.. Wearable and Non-Invasive Sensors for Rock Climbing Applications: Science-Based Training and Performance Optimization. *Sensors*, Vol. 23, No. 11, pp. 5058, MDPI, 2023.
- [Cha02a] Chawla, N., Bowyer, K., Hall, L., and Kegelmeyer, W.P.. SMOTE: Synthetic Minority Over-Sampling Technique. *Journal of Artificial Intelligence Research*, Vol. 16, pp. 321-357, 2002.
- [Eka22a] Ekaireb, S., Khan, M.A., Pathuri, P., Bhatia, P.H. and Sharma, R. and Manjunath-Murkal, N.. Computer Vision Based Indoor Rock Climbing Analysis. 2022.
- [Goo24a] Google. Pose Landmark Detection Guide. MediaPipe, https://developers.google.com/mediapipe/solutions/vision/pose_landmarker. 2024.
- [Kos15a] Kosmalla, F., Daiber, F., Krüger, A.. Climb-sense: Automatic Climbing Route Recognition using Wrist-Worn Inertia Measurement Units. *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pp. 2033-2042, 2015.
- [Kos17a] Kosmalla, F., Zenner, A., Speicher, M., Daiber, F., Herbig, N., Krüger, A.. Exploring Rock Climbing in Mixed Reality Environments. *Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems*, pp. 1787-1793, 2017.
- [Kos17b] Kosmalla, F., Daiber, F., Wiehr, F., Krüger, A.. Climbvis: Investigating In-Situ Visualizations for Understanding Climbing Movements by Demonstration. *Proceedings of the 2017 ACM International Conference on Interactive Surfaces and Spaces*, pp. 270-279, 2017.
- [Lad13a] Ladha, C., Hammerla, N.Y., Olivier, P., and Plötz, T.. ClimbAX: Skill Assessment for Climbing Enthusiasts. *Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pp. 235-244, 2013.
- [Low17a] Lowell, J. and Mortimer, P.. *The Dawn Wall*. Red Bull Media House and Sender Films, 2017.
- [Maj22a] Maji, D., Nagori, S., Mathew, M., and Poddar, D.. YOLO-Pose: Enhancing YOLO for Multi Person Pose Estimation Using Object Keypoint Similarity Loss. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2637-2646, 2022.
- [Mor14a] Mortimer, P. and Rosen, N.. *Valley Uprising*. Sender Films, 2014.
- [Ouc10a] Ouchi, H., Nishida, Y., Kim, I., Motomura, Y. and Mizoguchi, H.. Detecting and Modeling Play Behavior using Sensor-Embedded Rock-Climbing Equipment. *Proceedings of the 9th International Conference on Interaction Design and Children*, pp. 118-127, 2010.
- [Pfe11a] Pfeil, J., Mitani, J. and Igarashi, T.. Interactive Climbing Route Design using a Simulated Virtual Climber. *SIGGRAPH Asia 2011 Sketches*, pp. 1-2, 2011.
- [Ram20a] Ramsay, J. and Chang, H.J.. Body Pose Sonification for a View-Independent Auditory Aid to Blind Rock Climbers. *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 3414-3421, 2020.
- [Red16a] Redmon, J., Divvala, S., Girshick, R., and Farhadi, A.. You Only Look Once: Unified, Real-Time Object Detection. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 779-788, 2016.
- [Ric22a] Richardson, M., Petrini, K., and Proulx, M.. Climb-o-Vision: A Computer Vision Driven Sensory Substitution Device for Rock Climbing. *CHI Conference on Human Factors in Computing Systems Extended Abstracts*, pp. 1-7, 2022.
- [Sib07a] Sibella, F., Frosio, I., Schena, F. and Borghese, N.A.. 3D Analysis of the Body Center of Mass in Rock Climbing. *Human Movement Science*, Vol. 26, No. 6, pp. 841-852, 2007.
- [Vas18a] Vasarhelyi, E. and Chin, J.. *Free Solo*. National Geographic Documentary Films, 2018.
- [Wei14a] Wei, E.. *Indoor Rock Climbing Wall Route Displayer*. 2014.
- [Whi22a] Whitten, C. Pupillary Distance: Types and How to Measure. *WebMD*. 2022.
- [Xu22a] Xu, Y., Zhang, J., Zhang, Q., and Tao, D.. ViTPose: Simple Vision Transformer Baselines for Human Pose Estimation. *Advances in Neural Information Processing Systems*, Vol. 35, No. 6, pp. 38571-38584, 2022.

Automated Surface Extraction: Adaptive Remeshing Meets Lagrangian Shrink-Wrapping

Martin Čavarga
Comenius University Bratislava,
Mlynská dolina 5692, 841 04, Bratislava, Slovak Republic,
martin.cavarga@fmph.uniba.sk

Abstract

Fairing methods, frequently used for smoothing noisy features of surfaces, evolve a surface towards a simpler shape. The process of shaping a simple surface into a more complex object requires using a scalar field defined in the ambient space to drive the surface towards a target shape. Practical implementation of such evolution, referred to as Lagrangian Shrink-Wrapping (LSW), on discrete mesh surfaces presents a variety of challenges. Our key innovation lies in the integration of adaptive remeshing and curvature-based feature detection, ensuring mesh quality and proximity to target data. We introduce the Equilateral Triangle Jacobian Condition Number metric for assessing triangle quality, and introduce trilinear interpolation for enhanced surface detailing to improve upon existing implementations. Our approach is tested with point cloud meshing, isosurface extraction, and the elimination of internal mesh data, providing significant improvements in efficiency and accuracy. Moreover we extend the evolution to surfaces with higher genus to shrink-wrap even more complex data.

Keywords

fairing, surface evolution, Lagrangian Shrink-Wrapping, adaptive remeshing, feature detection, mesh quality, point cloud meshing, isosurface extraction, mesh simplification

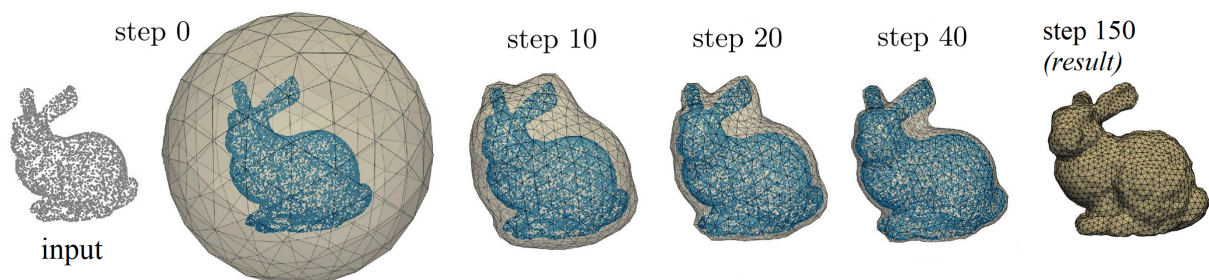


Figure 1: Lagrangian shrink-wrapping (LSW) evolution with the help of *adaptive remeshing* of input point cloud sampling of *bunny* with 12K vertices over 150 steps with an *icosphere* starting surface with subdivision level $s = 2$.

1 INTRODUCTION

Shape extraction from diverse input, including image data, point clouds, or mesh surfaces, is ubiquitous in modern geometry processing. A critical challenge in this domain is maintaining the quality of mesh elements extracted from these representations, which is essential for downstream processing applications such as rendering or numerical simulations. Traditional methods of direct triangulation of point clouds often require point normals, or otherwise struggle to produce watertight manifold surfaces. Unlike direct processing techniques, evolutionary methods make use of principles like diffusion described by partial differential equations which can be equipped with spatial information of input data, such as distance fields, to evolve a simple starting sur-

face with known topological properties, for example, a sphere, into a shape with the desired amount of detail.

1.1 Contributions

This work introduces a robust and stable shrink-wrapping tool designed for voxel, mesh, and point cloud data (see Fig. 1), employing a triangle surface mesh. Our contributions include:

- In contrast with prior approaches, the surface is no longer subject to the stretching of linear discretization elements (mesh triangles), essentially pushing the solution closer to how a smooth surface would evolve. This places our shrink-wrapping algorithm among the state-of-the-art methods which also employ some form of adaptive remeshing.

- Compared to existing implementations, we use trilinear interpolation of (possibly truncated) distance voxel values instead of nearest neighbor for shape precision.
- Extension of the tool’s applicability to initial surfaces constructed from distance contours broadens its utility to higher-genus target data¹.
- The use of customized equilateral triangle quality metric, validating the production of high-quality meshes.
- We adopt a curvature-based feature detection technique, facilitating a more accurate representation of intricate geometric features.

2 RELATED WORK

Our exploration stands on the contributions in significant areas of application: *fairing*, *remeshing*, *image segmentation*, *surface reconstruction*, and *geometry simplification*.

The fairing approaches utilize the key diffusion term in the evolutionary equations. In its simplest form it involves smoothing out all features of an input surface as performed using a semi-implicit formulation of the standard *mean curvature flow* (MCF) by Mikula et al. [2], and minimizing *Willmore energy* by Crane et al. [3]. This leads also towards unknotting tangled higher-genus surfaces by Yu et al. [4] who also considered a point cloud or mesh obstacle repelling an evolving surface. The outputs of such obstacle problem can be stacked on top of each other in the form of *nested cages* studied by Sach et al. [5].

Surface extraction from image data involves augmenting the standard MCF by *advection* which was also explored by Mikula et al. [2] in Section 3.2. These methods date back all the way to perhaps the most widespread brain extraction tomography application called BET2 first published by Smith in 2002 [6]. BET2 extracts a surface from an initial icosphere mesh which evolves towards the boundary of human brain, driven by advection evaluated from the thresholds computed within the histogram of the tomography image.

We put significant emphasis on the development of point cloud reconstruction tools developed by Daniel et al. [7] in collaboration with the authors of [2]. A suitable example of recent evolutionary approaches to surface reconstruction from point cloud data is Point2Mesh which also uses neural networks to evolve convex hull starting surfaces [8].

¹ We consider the genus of a surface envelope of a triangle soup or a point cloud target set in the sense of Hurtado et al. [1].

Since, as we mentioned, evolutionary methods can use simpler initial surfaces, they can also serve as simplification tools for enveloping meshes with undesired internal cavities and non-manifold vertices or edges [1].

This leads us to the use of evolutionary shrink-wrapping for the purposes of *remeshing*, as done by Kobbelt et al. [9], and extended to outward-evolving quad surface patches by Huska et al. [10].

3 METHODOLOGY

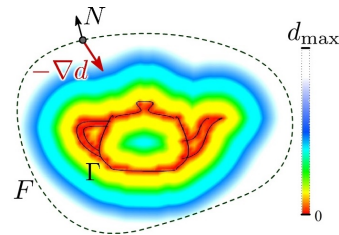


Figure 2: Slice of distance field d to surface Γ Utah Teapot with resolution 120^3 and an evolving surface F driven by fields d and $-\nabla d$. Source: [11].

3.1 Surface Evolution

Let X be a Riemannian 2-manifold. For a time interval $[0, T]$, the map $F : [0, T] \times X \rightarrow \mathbb{R}^3$ defines surface evolution in \mathbb{R}^3 , governed by:

$$\partial_t F = v_N + v_T, \quad (1)$$

where v_N and v_T are the normal and tangential components of velocity respectively. To ensure that the decomposition into normal and tangential components at $F(x) \in \mathbb{R}^3$ is well-defined at each point $x \in X$, $F^t = F(t, \cdot)$ must be an immersion of X into \mathbb{R}^3 for all $t \in [0, T]$. Equation (1) is accompanied by an initial immersion $F^0 = F(0, \cdot)$. First-order Laplacian smoothing follows $\partial_t F = \Delta_{g_F} F = -2HN$, where $H = (\kappa_{\min} + \kappa_{\max})/2$ is the mean curvature and N the outward unit normal. As demonstrated in Section 4.2 of [11], numerical simulations using the Laplace-Beltrami formulation with respect to the metric g_F of immersion F converge to the behavior of mean curvature flow (MCF) for the shrinking sphere solution $r(t) = \sqrt{r_0^2 - 4t}$, beginning from radius r_0 .²

Now let $\Gamma \subset \mathbb{R}^3$ be a *target set*, potentially a non-manifold surface (see Fig.2) or a point cloud. Let $d : \mathbb{R}^3 \rightarrow \mathbb{R}$ be the distance field of Γ , taking either the unsigned d^+ or signed d^\pm form, the latter of which distinguishes the interior $\text{Int}(\Gamma)$ from the exterior $\text{Ext}(\Gamma)$ of Γ with negative and positive sign respectively.

The target set Γ generates its distance field d in its ambient space \mathbb{R}^3 , and for this reason, it can affect the evolution of F in the following advection-diffusion model:

$$\partial_t F = \varepsilon \Delta_{g_F} F + \eta N + \rho v_T, \quad F(0, \cdot) = F^0, \quad (2)$$

² This alignment underscores the practical equivalence of the Laplacian smoothing approach to MCF in numerical settings.

where ε, ρ , and η are control functions, namely

$$\varepsilon(d) := C_1(1 - e^{-d^2/C_2}), \quad C_1, C_2 > 0, \quad (3)$$

$$\rho(d) := A(1 - e^{-d^2}), \quad A > 0, \quad (4)$$

$$\eta(d) := D_1 d(-\nabla d \cdot N) - D_2 \sqrt{1 - (\nabla d \cdot N)^2}, \quad (5)$$

$$D_1 > 0, \quad D_2 \geq 0,$$

inspired by [10] using the distance field d to target Γ . With ε we ensure that MCF slows down to zero as F approaches Γ , we also use a similar control function ρ to slow down the effects of tangential redistribution near Γ , and finally η controls the orientation of normal N to surface F (see Fig. 2). To make sure that F does not evolve past Γ under the influence of MCF, we can use modified weight functions ε^+ and η^+ which are non-zero only for positive values $d > 0$.

We also use the *angle-based tangential redistribution* inspired by [10] for computing velocity v_T with preferred weight $\omega = 0.05$ throughout our experiments.

3.2 The Discrete Picture

The evolving surface F is represented by a triangle mesh (K, V) where K is an abstract simplicial complex and $V \subset F[X] \subset \mathbb{R}^3$ the vertex set, such that its *geometric realization* — in the sense of Hoppe et al. [12] — is a compact manifold surface without boundary. Scalar and vector fields (d and $-\nabla d$) are sampled on regular voxel grids $G \subset \mathbb{R}^3$ containing Γ with cell size $c_G > 0$ using *trilinear interpolation*. In our experiments, we compute cell size as $c_G = \beta_{\min}/40$, that is: 40 voxels per minimal dimension of the target set's bounding box.

We use the *icosphere* subdivision surface (see "step 0" in Fig.1) for simulations with spherical starting surfaces, and for another set of simulations we use *isosurfaces* of the distance field $S_{d_0} = \{\mathbf{x} \in \mathbb{R}^3 \mid d(\mathbf{x}) = d_0\}$ reconstructed via the *Marching Cubes* algorithm [13] after which it is processed with a single step of *adaptive remeshing* by Dunyach et al. [14] to achieve more uniform vertex density³. The adaptive remeshing algorithm is then used during evolution when necessary⁴.

Distance field d is computed using a multi-step approach from Section 2.1 in [11] accelerated by spatial data structures like AABB tree and Octree, and using the *Fast-Sweeping* algorithm by Zhao [16]. The gradient ∇d is computed as a central difference of respective neighboring cell values for each cell.

Non-linear parabolic equation 2 is discretized using finite 2-volumes (areas) V_i surrounding mesh vertices

³ The PMP library [15] allows us to use only one iteration of `pmp::Remeshing::adaptive_remeshing` if the resulting mesh quality suffices.

⁴ when the measured polygon quality drops below the desired level.

$F_i, i = 1, \dots, N_V = |V|$ (see Fig. 3). The concept of these, so called, *co-volumes* is explained in more detail by Meyer et al. [17] and Mikula et al. [2]. In particular, the Laplace-Beltrami operator in (2) is discretized using a cotangent scheme when compiling the underlying sparse linear system of dimension $N_V \times N_V$. Analogously to [10], the advection terms in our case $\eta N + \rho v_T$ correspond to the state of the previous time step during evolution, and thus are added to the system's right-hand side.

The linear system is solved for each coordinate of vertex F_i and time step $t \in [0, T]$. Our preferred solver is BiCGStab, but according to [2] the sparse SOR method can also be used.

3.3 Numerical Stability

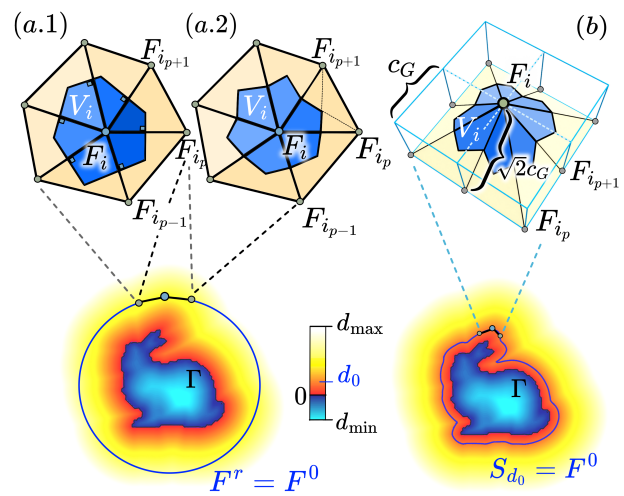


Figure 3: Two distinct types of Laplacian co-volumes evaluated for icosphere starting surface F^r : *Voronoi* (a.1) and *barycentric* (a.2) [17]. The starting surface can also be computed as a level set S_{d_0} of the distance field d where the maximum expected size of the (barycentric) co-volume can be estimated from 4 neighboring voxels with side $c_G > 0$ (b).

The stability of the sparse linear system derived from (2) relies heavily on the spectral radius $\rho(\mathbf{A})$ of the system matrix \mathbf{A} with non-negative diagonal entries. According to Section 3 in [18], the semi-implicit finite volume approach for curves in \mathbb{R}^2 leads to stability constraint $\tau \approx h^2$ where $h > 0$ is a spatial step and $\tau > 0$ a time step. In the co-volume formulation on F this translates to $\tau \approx \mu(V)$ where $\mu(V)$ is the measure (area) of a representative co-volume V . Large deviations result in the formation of singularities⁵ in F . However, variability of measures $\mu(V)$ in unstructured meshes allows only an approximate control ensuring $\rho(\mathbf{A}) \leq 1$.

⁵ At first glance, we consider these singularities qualitatively different from those handled by Kazhdan et al. [19].

The simplest available curvature-preserving heuristic involves uniform scaling the evolving surface F and its corresponding distance field d to target set Γ by factor

$$\phi = (\tau / (\sigma \mu_0(V)))^{1/2}, \quad (6)$$

where $\sigma > 0$ is a, so called, *shrink factor* which accounts for the decrement in average $\mu(V)$. Whenever one requires a result F^t in the original size we simply scale by the inverse of (6). The mean initial co-volume measure $\mu_0(V)$ is estimated from the starting geometry assuming uniform vertex density over the surface. In the case of an icosphere with radius $r > 0$, we put $\mu_0(V) = 4\pi r^2 / N_V^s$, where

$$N_V^s = (N_E^0(4^s - 1) + 3N_V^0) / 3 \quad (7)$$

is the number of vertices under subdivision level $s \geq 0$ with $N_V^0 = 12$ vertices and $N_E^0 = 30$ edges of the base icosahedron. For the proof of formula (7) see Chapter 5 in [20]. With initial isosurface $F^0 = S_{d_0}$, we estimate $\mu_0(V) = c_G^2(4\sqrt{2})/3$ for voxel size $c_G > 0$ (see Fig. 3 (b)).

Throughout our experiments with the shrink-wrapping evolution we keep track of the value $\mu(V)$ per vertex as well as its bounds and average.

3.4 Remeshing and Feature Preservation

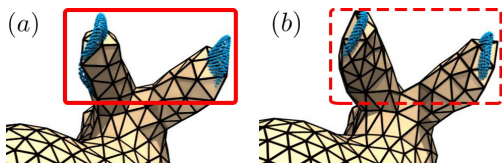


Figure 4: A trade-off between local triangle quality and feature preservation: Results after using cosine-based (a) and curvature-based (b) feature detection. Points of the target set Γ are shown in blue.

Adaptive remeshing [14] introduces deviations from the evolving surface F when applied. This can be mitigated using control functions ε^+ and η^+ with strictly positive support for signed distance fields that can even be shifted by some value $\tilde{d}_0 > 0$ to achieve better results. Sparse point cloud sets Γ , however, do not provide values $d < 0$, and require the use of different techniques for freezing points which should not sink below their respective feature.

In mesh processing, features refer to distinctive subsets of the mesh surfaces, such as sharp corners, creases, and boundaries, which carry important information about the shape of the object being represented. We need the sensitivity to *true features*⁶, and on the other hand avoid marking false positives at *convex-dominant saddle* (CDS) *points* (see Fig.5 (b)), that is: saddle points with much higher positive curvature.

⁶ Stemming from the shape of generating set Γ .

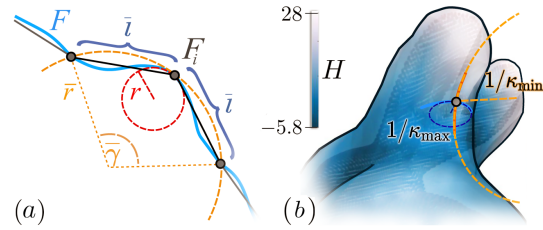


Figure 5: (a): An estimate of mean curvature angle $\bar{\gamma}$ at vertex F_i . (b): The imbalance of principal curvatures κ_{\max} and κ_{\min} at a *convex-dominant saddle vertex*. The color values show mean curvature H .

Since we require an automatic process during surface evolution, we propose a vertex-based detection evaluating the *angle of mean curvature* $\gamma = Hl$ where l is the arc length of smooth surface F . In the discrete setting we evaluate $\bar{\gamma}_i$ at vertex $\bar{F}_i = F_i$ using *neighborhood mean edge length*⁷: $\bar{l} = \frac{1}{m} \sum_{p=1}^m \|F_{i_p} - F_i\|$ where m is the valence of vertex F_i , and the cotangent estimate of mean curvature \bar{H} from [17].

Using $\bar{\gamma}$ we convert mean curvature H to a scale-invariant quantity. Vertices with $\bar{\gamma} = 2\bar{l}\bar{H} < \gamma_{\text{crit}}$ are marked as feature. Unfortunately, this alone leads to a decrease of mesh quality in CDS points – saddle points where $|\kappa_{\max}| < K|\kappa_{\min}|$ for curvature imbalance factor $K > 1$. Such feature elements can be false positives, and we propose not to mark them. Additionally, we should also not mark vertices with valence $m > 6$ because the following adaptive remeshing steps will avoid fixing them.

Considering the extent of sizes β_{\min} , β_{\max} of axis-aligned bounding box of Γ and box expansion factor $\zeta > 0$ for the distance field d we use empirical estimate

$$r \approx 0.4(\beta_{\min} + (0.5 + \zeta)\beta_{\max}) \quad (8)$$

for the radius of the starting icosphere with subdivision level s . The sizing for remeshing [14] is computed as

$$l_{\min} = 2\lambda_{\min}r \sin(\gamma_{\text{ico}}2^{-(s+1)}), \quad l_{\max} = \lambda_{\max}l_{\min}, \quad (9)$$

where $\gamma_{\text{ico}} = 2\pi/5$ is the angular segmentation of the base icosahedron, and $\lambda_{\max} > \lambda_{\min} > 0$ are controllable scale factors. Throughout experiments in Section 4 we empirically put $\lambda_{\min} = 0.14$, and $\lambda_{\max} = 4$. For the isosurface evolution we change the minimum edge length factor to $\lambda_{\min} = 0.4$ and put

$$l_{\min} = \lambda_{\min}\sqrt{2}c_G, \quad l_{\max} = \lambda_{\max}l_{\min}, \quad (10)$$

where c_G is the cell size of the distance field grid. Combined with error $\varepsilon_{\text{rem}} = \max\{\|F - \bar{F}\|\}$ between linear approximation \bar{F} and surface arc F , we obtain adaptive sizing values for all edges [14] (we set this value to $(l_{\min} + l_{\max})/4$). For our evolving surface, the sizing needs to be adapted to stabilized scale ϕ .

⁷ Also used by [6] to estimate the local radius of curvature for a surface update term.

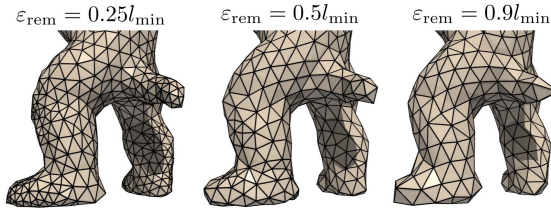


Figure 6: The effect of changing approximation error ϵ_{rem} after 80 time steps for adaptive remeshing as a multiple of minimum edge length $l_{min} = 0.0799$ of *Armadillo* (in stabilized scale $\phi = 0.0261$).

It should also be noted that strong (low- ϵ_{rem}) adaptivity may adversely affect co-volume sizing necessary for stability (see Fig. 6), especially for feature vertices.

To achieve higher level of detail for surface F close to Γ , the mesh sizing parameters λ_{min} and λ_{max} can be decreased for chosen time steps to achieve the effect seen in Fig. 1. The time step size τ must be adjusted accordingly, to ensure stability as described in Section 3.3.

3.5 Triangle Quality Metrics

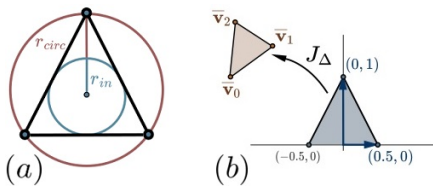


Figure 7: Two quality metrics with preference for equilateral triangles: double inradius over circumradius ratio (a), and Jacobian of the unit equilateral triangle (b) to the planar representation $T = \{v_0, v_1, v_2\}$ in triangle plane: \mathcal{P}_T .

As mentioned in Section 3.3, the semi-implicit formulation of (2) places restrictions on sizes of co-volumes and their uniformity across the surface. The resulting surface F^T can then be used, for example, in mechanical FEM analysis [21]. Most importantly, however, our motivation to ensure that most triangles are as close to equilateral as possible stems from the possibility of fast visualization. The rendering of almost-equilateral triangles can then be accelerated by treating them as discs, i.e.: *fish-scale mesh*⁸ [22]. More information on the quality of linear elements can be found in Section 1.1.2 of [23].

We choose two metrics measuring "equilateralness", namely

1. $2r_{in}/r_{circ} \in [0, 1]$ where r_{in} is the *inradius* and r_{circ} is the *circumradius* of the triangle, implemented in MeshLabTM. For an equilateral triangle $2r_{in} = r_{circ}$.
2. Our customized *condition number* $\kappa(J_{\Delta})$ of the *equilateral triangle Jacobian* $J_{\Delta} : \mathbb{R}^2 \rightarrow \mathbb{R}^2$ inspired by [24].

⁸ It is faster to compute disc-ray intersection than triangle-ray intersection.

When expressing the planar vertex position in the respective basis, we normalize the edge lengths with respect to basis vectors, shown in Fig.7, to account for scaling.

3.6 Experimental Setup

We implemented our framework in C++ with the help of the PMP Library⁹ [15] which provides a versatile half-edge (*surface*) mesh representation [26]. As mentioned in Section 3.2, we chose the BiCGSTAB solver with IncompleteLUT preconditioner from the Eigen library¹⁰.

3D visualization of the VTK polydata output is accomplished using ParaView from KitwareTM. Since VTK supports only vertex values, we average metrics from Section 3.5 across triangles adjacent to each vertex.

First, we show the validity of the stabilization heuristic from Section 3.3 by measuring $\mu_{max} = \max_{i=1, \dots, N_V} \mu(V_i)$, $\mu_{min} = \min_{i=1, \dots, N_V} \mu(V_i)$ and the mean $\mu(V) = \frac{1}{N_V} \sum_{i=1}^{N_V} \mu(V_i)$ for all time steps $t \in [0, T]$. We also observe how the use of remeshing reduces the range of values $[\mu_{min}, \mu_{max}]$.

Since our implementation currently supports two types of starting surface — icosphere and isosurface (see Fig.3) — we naturally choose the latter option to shrink-wrap target data with higher genus. To compare the efficiency of shrink-wrapping in other approaches, we select the closest predecessor [7] and the latest work on Repulsive Surfaces by Yu et al. [4]. Since these methods achieve a somewhat "incomplete" wrapping result, we distinguish between two scenarios

- *obstacle*, with $D_2 = 0$,
- and *full-wrap*, with $D_2 > 0$,

where D_2 is from the advection control function (5).

We judge the experiments based on two criteria:

1. *Distance-based*: How well the result F^T approximates the target set Γ .
2. *Quality-based*: What is the triangle quality distribution of F^T according to Section 3.5.

The first criterion can be measured by simply sampling d on its domain per vertex as $d|_{F^T}$ using trilinear interpolation, or we can approximate the *Hausdorff distance*:

$$d_H(F^t, \Gamma) := \max \left\{ \sup_{\mathbf{p} \in F^t} d(\mathbf{p}, \Gamma), \sup_{\mathbf{q} \in \Gamma} d(F^t, \mathbf{q}) \right\}, \quad (11)$$

where $d(\mathbf{p}, \Gamma) \approx d|_{F^t}$ and $d(F^t, \mathbf{q})$ is the value of the distance field to surface F^t at a point $\mathbf{q} \in \Gamma$ for all $t \in [0, T]$.

⁹ The library is inspired by the book with the same name [25].

¹⁰ PMP also uses Eigen internally for matrix and vector representations.

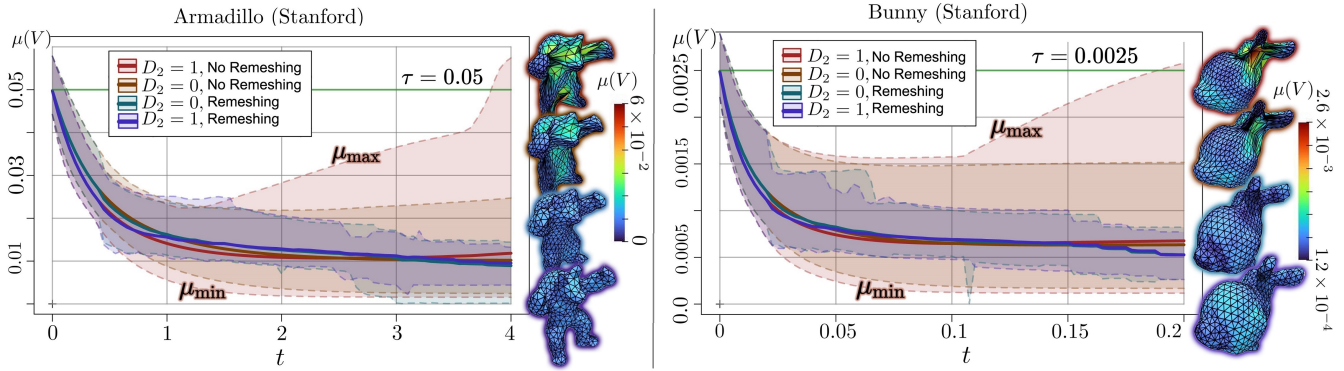


Figure 8: The range of co volume measures $[\mu_{\min}, \mu_{\max}]$ tested for two target meshes (Armadillo and Bunny), for four different settings (of constant D_2 and with or without adaptive remeshing) with resulting meshes $F^{80\tau}$ on the right.

Note that compared to [27] and [28] we choose to utilize trilinear interpolation within already computed distance fields over regular grids.

Furthermore, we divide our experiments into the *comparative* and *demo* categories, with the latter being focused solely on mesh simplification.

It can be observed that evolutionary algorithms (mentioned in Section 2) have a large variety of parameters to fine-tune. We also deem our overall surface extraction algorithm to have too many parameters to cover in this paper. We shall henceforth only mention some of them while, regarding the rest, we refer to our source code on GitHub [29] which will also contain the parameter settings which will yield the comparative, as well as, "most representative" results shown in Section 4.

4 RESULTS

4.1 Stability

Since scaling (6) can be weighed by shrink factor $\sigma > 0$, we first assume $\sigma = 1$, and observe the values of co-volume measures $\mu(V)$ for each vertex during $N_t = 80$ steps. From measures shown, for example, in Fig. 8 we deduce $\mu(V) \approx \tau/5$ most of the time, so we put $\sigma = 1/5$. It is evident that without remeshing, values $\mu(V)$ fluctuate even above the time step size τ .

4.2 LSW for Surface Reconstruction

For the comparative evaluation we choose the Bunny mesh uniformly sampled¹¹ to 12K vertices. Comparing the resulting values in three distinct histogram evaluations for the final time step (see Fig. 9), yields a clear improvement of our method when compared with that of Daniel et al. [7] in terms of both face quality metrics due to remeshing. As mentioned in Section 3.6, it was also important to see how far away the resulting surface

stays from the target point cloud Γ . Clearly, the distribution of distances is smallest for our *full wrap* configuration with the maximum fit. However, in order to improve the effectiveness of feature detection (see Section 3.4) for point cloud targets, we chose value $\tilde{d}_0 = \frac{3\sqrt{3}}{4}c_G$ which is the 1.5-multiple of the voxel diagonal half-length, which is evident in the shift of the histogram of $d|_F$ when compared to the results from [7] and [4] (see Fig. 9).

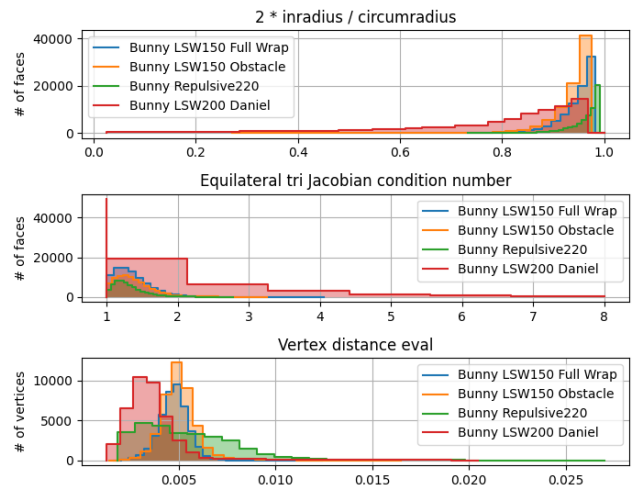


Figure 9: Comparison of four different LSW experiments via histograms of per-face (see Section 3.5) and per-vertex metric $d|_F$ for the last time step.

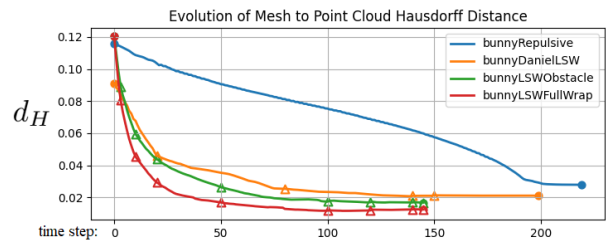


Figure 10: Evaluation of (11) throughout evolution.

¹¹ From the uniform distribution over all mesh vertex indices from the original file.

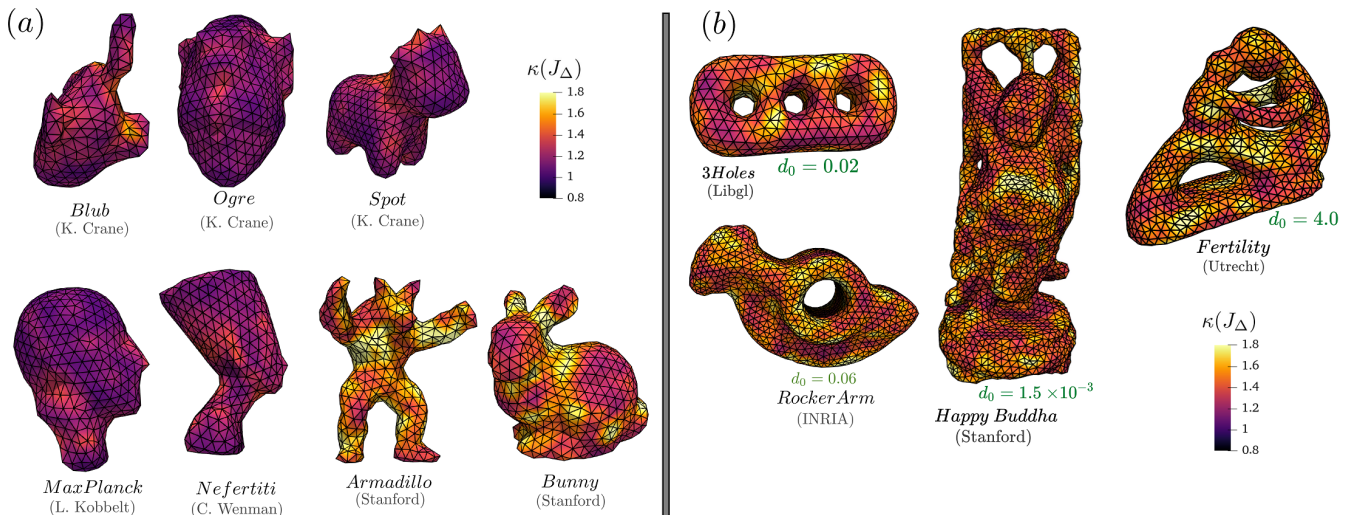


Figure 11: Evaluation of $\kappa(J_\Delta)$ (see Section 3.5) on the shrink-wrapping results from *icosphere* setup (a), and the higher-genus *isosurface* starting surface with given isolevel values $d_0 > 0$.

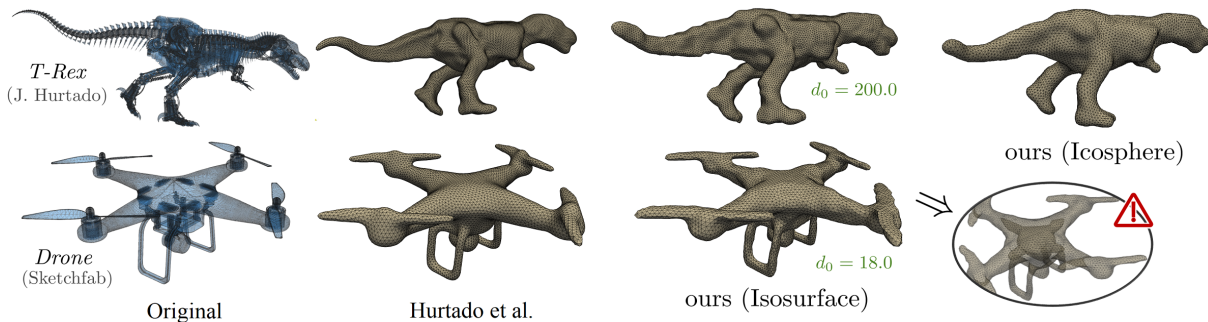


Figure 12: Comparison of our isosurface setup with the "Enveloping CAD" approach by Hurtado et al.[1].

When examining Hausdorff distance (11), we observe fast and stable convergence of F towards the target point cloud Γ in both our Obstacle and FullWrap configurations. Due to the lack of the corrective term with coefficient D_2 in (5), the implementation by Daniel et al. [7] hangs without reaching the surface with significant oscillations. The completely different global method by Yu et al. [4] finds it difficult to converge to the target altogether even after more time steps (see Fig. 10).

We also mark the time steps when mesh sizing changes with symbol Δ . In case of [7] it takes place of a 4:1 triangle subdivision, whereas we use a decay factor of 0.7 for λ_{\min} (see Section 3.4) which is, analogously to [7], applied to time step size τ to ensure stability. In the case of our implementation and that of Yu [4], remeshing takes place whenever necessary for all time steps.

4.3 High-Quality Simplification

Treating our model as an enveloping tool for mesh simplification presents its own benefits and challenges. Most importantly, the simpler topology of the starting

icosphere surface F^0 yields higher simplification ratio considering meshes with internal simplices, that is: those embedded inside the enveloping boundary of the surface, as in Hurtado et al. [1]. On the flip side, the shrink-wrapping of datasets with higher-genus enveloping boundaries leads to self-intersections of F . Without the ability to fix self-intersections by incrementing the surface genus, we are bound to the genus of the starting surface F^0 . That being said, if we construct F^0 from a higher-genus isosurface of the target set distance field d , we are bound to keep internal cavities if the data already has some larger than voxel size $c_G > 0$ (see Fig. 12).

In this section, we also present quality assessment via $\kappa(J_\Delta)$ of the resulting surfaces F^T for a wide variety of datasets as shown in Fig. 11. Even with a few iterations of adaptive remeshing whenever the range of $\kappa(J_\Delta)$ per triangle exceeds our chosen range in our experiments: $[1, 1.5]$ (which slightly exceeds that recommended in [24]), we are able to maintain the quality of almost all faces for all time steps. For meshes with significant features, such as *Armadillo*, *Bunny*, and the higher-genus

isosurface evolution datasets (Fig. 11 (b)) the use of curvature-based feature detection as described in Section 3.4 inhibits adaptive remeshing on vertices marked as feature.

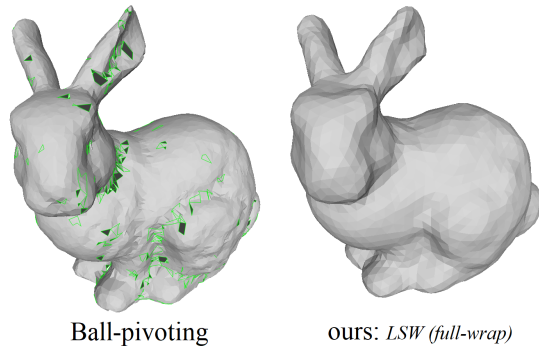


Figure 13: Comparison of point cloud reconstruction results from the Ball-pivoting algorithm and our full-wrap LSW with sizing 0.005. Boundary edges of the non-watertight resulting surface are highlighted in green.

5 DISCUSSION AND FUTURE WORK

Our scale-based heuristic shows stable results with icosphere starting surfaces F^0 (see Section 4.1). However, adjusting parameters is necessary for isosurface F^0 to ensure stability. The LSW method excels in providing the best fit for full-wrap configurations, surpassing other surface evolution techniques in Hausdorff distance evaluation, demonstrating LSW’s ability to closely approximate target meshes without sacrificing surface integrity.

When compared to shrink-wrapping techniques like Yu et al. [4], our approach offers competitive face quality and significantly better fit via distance-based evaluation, particularly excelling in the full-wrap configuration (see Section 4.2).

In addition to the results presented in Section 4, we must also qualitatively compare our reconstruction approach to the *Ball-Pivoting Algorithm* (BPA) [30] which, unlike the widely-used *Poisson Reconstruction* [31], does not require normals for the input point cloud (see Fig. 13). Unlike BPA, which can introduce holes, LSW with the full-wrap configuration stands out in producing watertight surface reconstruction without the need for point normals. It should be noted, that unlike BPA, shrink-wrapping approaches merely approximate the original points.

As a mesh simplification tool, our model offers both benefits and challenges. The use of an icosphere F^r as a starting surface simplifies topologies effectively, yet produces self-intersections for higher-genus target data Γ . This stems from the fact that the *comparison principle* (see Huisken [32]) no longer holds for the forced MCF (2). The use of isosurface S_{d_0} with isolevel $d_0 > 0$,

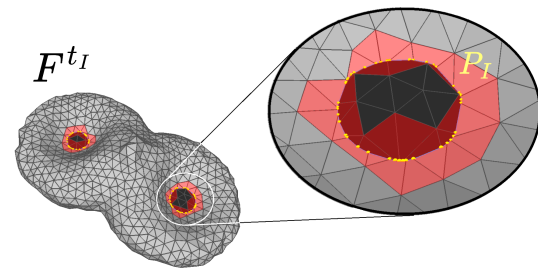


Figure 14: The formation of self-intersection polylines P_t at time $t_t > 0$ when shrink-wrapping a double torus. Faces intersecting at least one other face from surface F^{t_t} are highlighted in red.

on the other hand, does not solve this problem for general target data Γ because it preserves undesired cavities (see Fig. 12).

In our future work we focus on solving the issue of compatibility with higher-genus data by removing surface patches with inverted normals, and properly connecting the remaining faces (see Fig. 14). This will automatically increment the genus of the evolving surface F whenever self-intersections are detected at time $t_t > 0$. This approach is the time-inverse of the surgery technique by Kovács [33], but requires only combinatorial adjustment when the self-intersection polyline P_t is computed.

We shall also examine the *stopping criterion* for LSW to achieve full versatility. Evolution can, for example, terminate when at least 98% of the points no longer move, or change their distance d to Γ . Such criteria need to be formulated for general target data with arbitrary genus and features.

Furthermore, our feature detection techniques are insufficient for very sharp features in Γ (see the *T-Rex* dataset in Fig. 12), and require further inquiry with various projection-based techniques.

When it comes to the applications, we consider using our method for progressively streaming very large mesh files only by passing well-sampled subset of its vertex table and then reconstructing the sample as a point cloud. Feature-sensitive stochastic sampling has been done by Li et al. [34], yet may benefit from our approach which does not require normals in the input data Γ . We propose that this progressive vertex-focused file parsing provides reasonable preview for very large data, often acquired by modern 3D scans.

6 CONCLUSION

In this work, we have introduced a novel approach to automated surface extraction that significantly enhances the efficiency and accuracy of shaping complex objects from simple surfaces. By integrating adaptive remeshing and curvature-based feature detection into the process of Lagrangian Shrink-Wrapping (LSW), we have shown that our method not only maintains high

mesh quality but also ensures proximity to target data across various configurations.

Our findings demonstrate that the scale-based heuristic provides a stable foundation for the evolution of icosphere starting surfaces, albeit requiring nuanced parameter adjustments for other surfaces like isosurfaces. Notably, our method achieves superior surface fitting in "full-wrap" configurations, outperforming existing evolutionary methods when assessed using Hausdorff distance. This precision in approximation without sacrificing surface integrity highlights the potential of our approach for applications requiring accurate and high-quality surface reconstructions.

Moreover, our method presents a competitive alternative to existing surface reconstruction and mesh simplification techniques, such as those proposed by Yu et al. [4], especially in terms of face quality and the preservation of intricate geometric features. The ability of LSW to produce watertight tessellations without the need for normals in the input data, unlike the Ball-Pivoting Algorithm, further underscores its utility for seamless model generation from point cloud data.

As an enveloping tool for mesh simplification, our approach exhibits a comparable simplification ratio for meshes with internal simplices, aligning with the observations of Hurtado et al. [1]. However, we also recognize the challenges posed by datasets with higher-genus enveloping boundaries introducing self-intersections during evolution, which will be the focus of our future work.

Looking forward, we aim to address the limitations encountered with higher-genus data and explore the development of more robust feature detection techniques to capture sharper features accurately. The potential application of our method for progressive streaming of large mesh files presents an exciting avenue for research, promising a significant impact on the field of geometry processing and beyond.

ACKNOWLEDGMENTS

We extend our gratitude to Patrik Daniel and Mariana Sarkoci Remesikova, the authors of [7], for the source code of their application, as well as to Jan J. Hurtado, the co-author of [1], for providing the executable of the article's application (with the T-Rex mesh included). We also express our appreciation to Keenan Crane and Chris Yu, the co-authors of [4], and Robert Bohdal, for their assistance in providing experimental results from the *Repulsive Surfaces* framework.

Special thanks are extended to the creators of the test meshes used in our studies: the Stanford 3D Model Repository (Armadillo, Bunny, Happy Buddha), Keenan Crane (Blub, Ogre, Spot), teams of INRIA, Libgl, Utrecht (RockerArm, 3Holes, Fertility), Leif

Kobbelt and the team at RWTH Aachen (Max Planck), and Cosmo Wenman (Nefertiti), for generously sharing their high-quality models. Additionally, we acknowledge the development teams of the PMP library led by Daniel Seiger and Mario Botsch, and the Eigen Library, for their indispensable tools that facilitated our research.

Our work has also benefited from the use of other datasets not explicitly mentioned earlier (e.g.: Drone from Sketchfab), and the use of ParaView from Kitware™ and MeshLab from ISTI™ for visualization. We recognize the contributions of these and all other data providers who have indirectly supported our research.

Finally, we would like to thank the anonymous reviewers for analyzing our work meticulously.

7 REFERENCES

- [1] Jan Hurtado Jauregui, Anselmo Montenegro, Marcelo Gattass, Felipe Carvalho, and Alberto Raposo. Enveloping cad models for visualization and interaction in xr applications. *Eng. Comput.*, 38(1):781–799, 2022.
- [2] Karol Mikula, Mariana Remešiková, Peter Sarkoci, and Daniel Ševčovič. Manifold evolution with tangential redistribution of points. *SIAM J. Sci. Comput.*, 36:A1384–A1414, 2014.
- [3] Keenan Crane, Ulrich Pinkall, and Peter Schröder. Robust fairing via conformal curvature flow. *ACM Trans. on Graphics (TOG)*, 32(4):1–10, 2013.
- [4] Chris Yu, Caleb Brakensiek, Henrik Schumacher, and Keenan Crane. Repulsive surfaces. *ACM Trans. Graph.*, 40(6), 2021.
- [5] Leonardo Sacht, Etienne Vouga, and Alec Jacobson. Nested cages. *ACM Trans. Graph.*, 34:1–14, 2015.
- [6] Stephen Smith. Fast robust automated brain extraction. *Human brain mapping*, 17:143–55, 2002.
- [7] Patrik Daniel, Matej Medl'a, Karol Mikula, and Mariana Remešiková. Reconstruction of surfaces from point clouds using a lagrangian surface evolution model. page 589–600, 2015.
- [8] Rana Hanocka, Gal Metzer, Raja Giryes, and Daniel Cohen-Or. Point2mesh: A self-prior for deformable meshes. *ACM Trans. Graph.*, 39(4):126:1–126:12, 2020.
- [9] Leif Kobbelt, Jens Vorsatz, Ulf Labsik, and Hans-Peter Seidel. A shrink wrapping approach to remeshing polygonal surfaces. *Comput. Graph. Forum*, 18:119–130, 2000.
- [10] Martin Huska, Matej Medl'a, Karol Mikula, and Serena Morigi. Lagrangian evolution approach

- to surface-patch quadrangulation. *Appl. Math.*, 66:1–43, 2021.
- [11] Martin Čavarga. Advection-driven shrink-wrapping of triangulated surfaces. In *Proc. of the 26th Central European Seminar on Computer Graphics : CESC*, page 95–104, 2022.
- [12] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. Mesh optimization. In *Proc. of the 20th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '93*, page 19–26, 1993.
- [13] William E Lorensen and Harvey E Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *Seminal graphics: pioneering efforts that shaped the field*, pages 347–353. 1998.
- [14] Marion Donyach, David Vanderhaeghe, Loïc Barthe, and Mario Botsch. Adaptive remeshing for real-time mesh deformation. In *Eurographics 2013 - Short Papers*, page 29–32, 2013.
- [15] Daniel Sieger and Mario Botsch. The polygon mesh processing library. <http://www.pmp-library.org>, 2019.
- [16] Hongkai Zhao. A fast sweeping method for eikonal equations. *Math. Comput.*, 74:603–627, 2005.
- [17] Mark Meyer, Mathieu Desbrun, Peter Schröder, and Alan Barr. Discrete differential-geometry operators for triangulated 2-manifolds. page 37–57, 2003.
- [18] Karol Mikula, Daniel Ševčovič, and Martin Balázovjeh. A simple, fast and stabilized flowing finite volume method for solving general curve evolution equations. *Communications in Computational Physics*, 7:195–211, 2010.
- [19] Michael Kazhdan, Jake Solomon, and Mirela Ben-Chen. Can mean-curvature flow be modified to be non-singular? In *Computer Graphics Forum*, volume 31, pages 1745–1754. Wiley Online Library, 2012.
- [20] Martin Čavarga. Mesh primitive counting formulas for subdivision surfaces. In *Proc. of the 9th Slovak-Czech Conference on Geometry and Graphics 2023*, pages 67–76, 2023.
- [21] Dan Neumayer, Madhukar Chatiri, and Matthias Hörmann. Drop test simulation of a cooker including foam packaging and pre-stressed plastic foil wrapping. 2006.
- [22] Radim Sara and Ruzena Bajcsy. Fish-scales: Representing fuzzy manifolds. In *Sixth International Conference on Computer Vision (IEEE Cat. No. 98CH36271)*, pages 811–817. IEEE, 1998.
- [23] S.W. Cheng, T.K. Dey, and J. Shewchuk. *De-launay Mesh Generation*. Chapman & Hall/CRC Computer and Information Science Series. CRC Press, 2016.
- [24] Sandia National Laboratories. Metrics for triangular elements. https://www.sandia.gov/files/cubit/15.3/help_manual/WebHelp/mesh_generation/mesh_quality_assessment/triangular_metrics.htm, 2017.
- [25] Mario Botsch, Leif Kobbelt, Mark Pauly, Pierre Alliez, and Bruno Lévy. *Polygon Mesh Processing*. AK Peters / CRC Press, 2010.
- [26] Daniel Sieger and Mario Botsch. Design, implementation, and evaluation of the surface_mesh data structure. In *Proc. of the 20th International Meshing Roundtable*, pages 533–550, 2012.
- [27] Dejun Zhang, Fazhi He, Soonhung Han, Lu Zou, Yiqi Wu, and Yilin Chen. An efficient approach to directly compute the exact hausdorff distance for 3d point sets. *Integrated Computer-Aided Engineering*, 24:261–277, 2017.
- [28] Michael Bartoň, Iddo Hanniel, Gershon Elber, and Myung-Soo Kim. Precise hausdorff distance computation between polygonal meshes. *CAGD*, 27(8):580–591, 2010.
- [29] Martin Čavarga. Lsw mesh flow. <https://github.com/MCInversion/LSWMeshFlow>, 2022.
- [30] Fausto Bernardini, Joshua Mittleman, Holly Rushmeier, Cláudio Silva, and Gabriel Taubin. The ball-pivoting algorithm for surface reconstruction. *IEEE trans. on visualization and computer graphics*, 5(4):349–359, 1999.
- [31] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In Alla Sheffer and Konrad Polthier, editors, *Proc. of the 4th Eurographics Symposium on Geometry Processing, Cagliari, Sardinia, Italy, June 26-28, 2006*, volume 256 of *ACM International Conference Proceeding Series*, pages 61–70, 2006.
- [32] Gerhard Huisken. Flow by mean curvature of convex surfaces into spheres. *J. Differential Geom.*, 20:237–266, 1984.
- [33] Balázs Kovács. Numerical surgery for mean curvature flow of surfaces. *SIAM J. Sci. Comput.*, 46(2):A645–A669, 2024.
- [34] Yuanqi Li, Jianwei Guo, Xinran Yang, Shun Liu, Jie Guo, Xiaopeng Zhang, and Yanwen Guo. Deep point cloud simplification for high-quality surface reconstruction. *ArXiv*, abs/2203.09088, 2022.

CNN-based Game State Detection for a Foosball Table

David Hagens
ORDIX AG
Karl-Schurz-Straße
19a
33100, Paderborn,
Germany
ddh@ordix.de

Jan M. Knaup
ORDIX AG
Karl-Schurz-Straße
19a
33100, Paderborn,
Germany
jmk@ordix.de

Elke Hergenröther
Darmstadt
University of Applied
Sciences
Schöfferstraße 3
64295, Darmstadt,
Germany
elke.hergenroether@h-
da.de

Andreas Weinmann
Algorithms for
Computer Vision,
Imaging and Data
Analysis Group,
Hochschule Darmstadt,
Germany
andreas.weinmann@h-
da.de

ABSTRACT

The automation of games using Deep Reinforcement Learning Strategies (DRL) is a well-known challenge in AI research. While for feature extraction in a video game typically the whole image is used, this is hardly practical for many real world games. Instead, using a smaller game state reducing the dimension of the parameter space to include essential parameters only seems to be a promising approach. In the game of Foosball, a compact and comprehensive game state description consists of the positional shifts and rotations of the figures and the position of the ball over time. In particular, velocities and accelerations can be derived from consecutive time samples of the game state. In this paper, a figure detection system to determine the game state in Foosball is presented. We capture a dataset containing the rotations of the rods which were measured using accelerometers and the positional shifts were derived using traditional Computer Vision techniques (in a laboratory setting). This dataset is utilized to train Convolutional Neural Network (CNN) based end-to-end regression models to predict the rotations and shifts of each rod. We present an evaluation of our system using different state-of-the-art CNNs as base architectures for the regression model. We show that our system is able to predict the game state with high accuracy. By providing data for both black and white teams, the presented system is intended to provide the required data for future developments of Imitation Learning techniques w.r.t. to observing human players.

Keywords

Game State Detection, Computer Vision, Deep Learning, Foosball, Deep Reinforcement Learning, Imitation Learning

1 INTRODUCTION

State detection based on Computer Vision techniques has been used in the automation of games using Reinforcement Learning, cf. [22, 25]. The common way is to take an input image stream, e.g. the whole screen in a video game cf. [22], and predict a next action based on a DRL system. In contrast to video games, the usage of whole images is often not practical when automating real-world games. In this case, using a lower dimension abstraction of the game, for which we employ the term game state, can be advantageous for the DRL training and prediction. The game of Football is a good example for the automation of complex real-world games [20].

A better accessible scenario is found in the game of Foosball, which provides a stable, more controllable environment, cf. [28, 29]. More concrete, the game state can be defined as the positional shift and the rotations of the figure rods plus the position of the ball as a function of time. In particular, the velocities of the figure rods and the velocity of the ball can be approximately calculated using multiple consecutive time samples. We note that, specifically in Foosball, using the described game state instead of an overall image reduces the parameter space for a DRL agent significantly. In order to provide the required state space data, a game state detection system is needed to extract the game state and provide the data to the DRL agent.

In this paper, we present a game state detection system for the Foosball table shown in Fig. 1. The Foosball table is automated using industrial motors on the black team while the other, white team is human-controllable. At the top of the table, a Logitech BRIO webcam is mounted which captures the playing field in a top down perspective. Currently, the motors report their shifts and rotations, so the game state of the black figures is

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.



Figure 1: The physical Foosball table. The black team is controlled by industrial linear and rotary motors while the white team is controlled by humans. A Logitech BRIO webcam captures the playing field in a top down perspective.

available as a potential ground truth. Additionally, there exists work on the detection of the position and velocity of the ball [5]. The game state of the white figures is unknown. Our system developed in this work is able to detect the game state of all figures (black and white) of the Foosball table using Deep CNN and Computer Vision. We utilize the available game state data for the

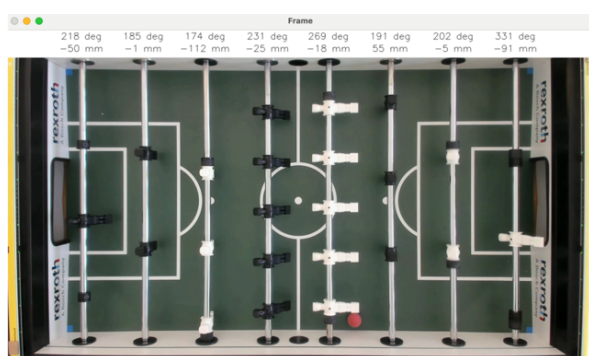


Figure 2: A working example of our figure detection system. The predicted positional shift and rotation angle of each rod is printed above the rod. The captured data can be used by a DRL agent through a ZeroMQ based data provisioning system.

black figures as a ground truth to validate our system. A working example of our system is shown in Fig. 2. As all figures are included in the game state, our system can also be used to capture the state of non-automated Foosball tables. In this respect, we are preparing and contributing to future Imitation Learning experiments (based on matches of only human players) where the whole state space is planned to be captured by Computer Vision techniques.

The paper is structured as follows: Section 1 introduces this paper and provides a brief overview of related work. In section 2 we describe our approach to game state detection. We evaluate our system and present results in section 3. In section 4 we conclude the paper and present aspects of future work.

1.1 Related Work

Several studies conducted research in automating a Foosball table. The sub-process of game state detection is a key part of the automation process [11, 13, 29]. While the detection of the ball is a necessity for a rudimentary automation of a Foosball table, Enos et al. [7] and Gashi et al. [9] note the importance of the detection of the figures to enable a dynamic game play. Due to the focus on the basic automation, most studies do not attempt the figure detection in their work. Bambach et al. [3] and Horst et al. [15] only examined the game state detection without addressing the actual automation process. In contrast, Gashi et al. [9], Rohrer et al. [24] and Zhang et al. [31] only addressed the automation by extending already existing automated Foosball tables and using the established state detection methods.

The automation Hardware is usually built around linear and rotary motors to control the rods with additional sensors to measure the shift and rotation, cf. [5, 17, 21, 29]. Over time, the research shifted from rule based algorithms, cf. [29, 30], towards DRL based machine learning models, cf. [5, 9, 24]. Imitation Learning methods were also used to improve the early rule based approaches [31].

All above studies use a camera and computer vision techniques to detect the game state. Mohebi [21] describes additional methods for the ball state detection, including the usage of Bluetooth, a touch sensible screen as the playing field and a grid of light emitters and detectors. Most studies, however, rely on color segmentation based approaches to detect a distinct colored ball, cf. [3, 7, 11, 18, 26]. The state of the figures is often not considered. Weigel et al. [29] implemented a rudimentary figure detection system which they discarded in their next iteration [28]. Bošnjak et al. [4] proposed the usage of visual marker patterns to detect the rotation of the rods. Janssen et al. [17] used a MRI scanner to detect figures in

a 3D scan but only used this information to enhance the ball detection system. Mohebi [21] proposed the usage of additional sensors for the figure detection. Other studies addressed the figure detection by using a similar color segmentation approach as already used for the ball detection, cf. [1, 21]. In contrast, Horst et al. [15] concentrated their work on detecting the game state of an automated Foosball table by implementing a proof-of-concept for the midfield, human-played figures. Their setup includes the utilization of the YOLOX object detector [10] to find the individual figures and a custom regressor network to predict the rotation angle.

1.2 Contributions

In this work, we present a figure detection system for the physical automated Foosball table used by Horst et al. [15], De Blasi et al. [5] as well as [9, 24], the later two references focusing on the DRL part. Our work extends previous work of Horst et al. [15] who derived a first proof-of-concept using CNN-based Computer Vision methods.

Our contributions can be summarized as follows. (a) We created and verified a ground truth dataset for the training of a CNN-based detection model; (b) We derive an improved figure detection system by basing on end-to-end learning; further, the new system is able to detect all white figures and additionally the black, computer controlled figures. (c) In the developed end-to-end setup, we trained and evaluated multiple feature extractor backbones, including ResNet [12], MobileNet [16] and EfficientNet [27]; (d) We propose a data provisioning system based on ZeroMQ [14]. Concerning (a) we use accelerometers and the motors of the Foosball table to measure the rotations of the rods. Additionally, we derive the shift by utilizing traditional CV techniques. While [15] captured all data in one video using OLED screens to display the rotation of the rods, we discard the screens and OCR step by directly reading from the micro-controller, therefore reducing complexity and fixing noted issues. Concerning (b) we discard the YOLOX object detector [10] which was used in [15] to detect each individual figure. Instead, we aim for an end-to-end regressor network for shift and rotation detection on a per-rod basis. Additionally, we extend and refine the ideas of [15] to include all figures of the Foosball table and to fix issues noted there. Furthermore, we contribute to the DRL research started by De Blasi et al. [5] and extended by Rohrer et al. [24] and Gashi et al. [9]. In particular, by deriving the whole state space (including the black figures), we are preparing for Imitation Learning approaches which require a system to extract the game state of pre-recorded human matches (including both black and white figures).

In contrast to other previous research, color based approaches, cf. [1, 3, 26, 29], are not applicable since

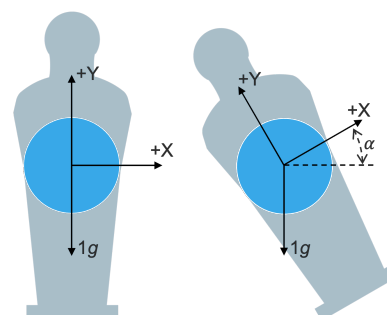


Figure 3: Measuring the rotational shift relative to the ground (α) using a two-axis accelerometer. Since the gravitational force of $1g$ is fixed and due to the orthogonal alignment of the X and Y axis, the measured accelerations on those axis are proportional to the sine and cosine of α .

the figures are not colored with distinguishable colors. While we also use additional sensors for creating a ground truth, a similar approach to Mohebi [21] is not applicable due to a non-permanent hardware modification constraint. We use state of the art deep CNN for the game state detection which removes the need for a calibration step ahead of time and creates a more robust and stable system in contrast to traditional CV techniques.

2 GAME STATE DETECTION

In the following, we present our approach to detecting the game state of a Foosball game. Relying on previous work on detecting the ball [5], we focus on detecting the figures. More precisely, we detect the positional shifts and rotations of the figures. This amounts to detecting the shift and rotations of the corresponding rods since the relative position and the angles do not change within the rod. The velocities of the figures can be calculated through consecutive shifts and rotations. First, we describe our approach to creating a ground truth dataset using accelerometers and the motors of the Foosball table. Afterwards, we develop end-to-end regressor models basing on widely used CNN backbones as feature extractor networks.

2.1 Dataset Creation

The training of our end-to-end regressor network requires the presence of a dataset consisting of the shifts and rotations of all rods of our Foosball table. For the white, human controlled rods, the shift and rotation values are not available. In contrast, the motors which control the black rods report their state. We use the reported state as a ground truth to verify our dataset creation system.

The shift of the white rods can be calculated using traditional Computer Vision techniques. In contrast, the

rotation cannot be validated with traditional CV methods due to different perspectives and no ground truth being available. Therefore, our approach includes the usage of accelerometers to measure the rotation in the real world. An accelerometer measures the linear acceleration on a defined axis. When fixed to a specific point, the gravitational force is a linear acceleration which can be measured by an accelerometer. Using two orthogonal axis, the relative tilt to the ground can be calculated, cf. [8, 23]. As shown in Fig. 3, the measured acceleration A is related to the angle α : $\frac{A_{X,OUT}}{A_{Y,OUT}} = \frac{1g \times \sin(\alpha)}{1g \times \cos(\alpha)} = \tan(\alpha)$ which results in $\alpha = \tan^{-1}\left(\frac{A_{X,OUT}}{A_{Y,OUT}}\right)$ with $A_{X,OUT}$ as the measured acceleration on the X axis and $A_{Y,OUT}$ as the measured acceleration of the Y axis. The default rotation is defined as a figure standing vertically with the head up. The rotation is measured as 0 degrees in the default rotation while the motors of the black figures report 180 degrees as default. Therefore, the measured rotations are shifted to also use a default rotation at 180 degrees.

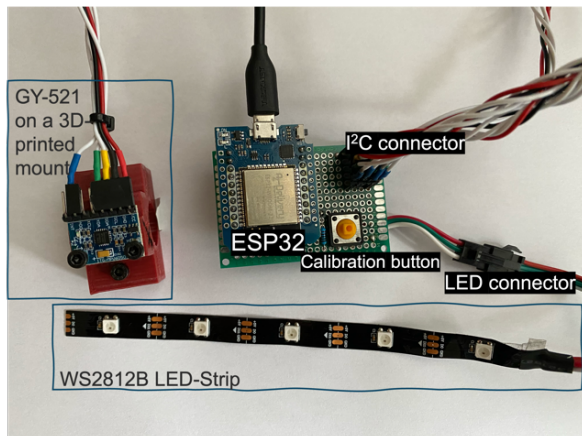


Figure 4: The hardware used to measure the rotation of the white figures. Four GY-521 modules with MNPU6050 accelerometers on 3D-printed mounts were screwed on top of the rods. The accelerometers are connected to an ESP32 based micro-controller via an I²C connector. Additionally, we included a WS2812B-based LED strip to indicate the internal timing and a button to calibrate the zero-point of the accelerometers.

We validated two different types of accelerometers, namely the MPU6050 on GY-521 breakout boards and the ADXL345. By measuring the rotation of the black rods with those sensors we could verify the measurements and estimate errors between the actual rotation and the measurements. We found that the MPU6050 sensor measured the rotation with a higher accuracy but still had a mean absolute error of around 5 degrees with outliers between 20 and 25 degrees. We also observed that some of the sensors tested had calibration errors which were visible in a systematic deviation between

the real and measured rotation angle. To minimize the deviation between the measurements and the real rotation in our ground truth, we rejected the corresponding sensors. Fig. 4 shows an image of our measurement hardware.

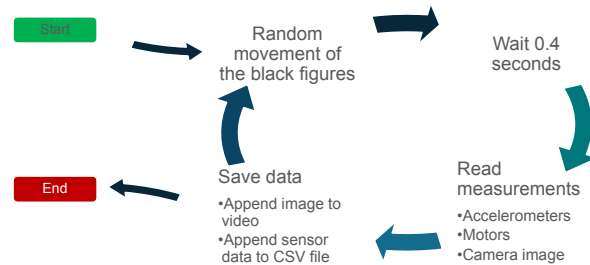


Figure 5: Our proposed dataset capturing process. One iteration of the process corresponds to one frame taken by the camera. While the black figures are moved automatically, the white figures need to be moved manually during the dataset capturing to get a versatile set of different shifts and rotations.

The capturing process, as illustrated in Fig. 5, consists of four steps which are repeated. First, we move and rotate the black rods randomly to get diverse states. As the white rods are not connected to motors, those were moved manually by hand. Next, the system waits 0.4 seconds to let the motors finish their movements. This reduces the risk of motion blur and a faulty state reported by the motors, as they report their final position and not their current shift. Afterwards, the measurements of the motors and the accelerometers are retrieved and a snapshot of the webcam is captured. In the last step, the camera image is appended to an MP4-encoded video and the measurements are saved in a CSV file. This process is repeated N times to get N individual game states with a corresponding camera image in the dataset. The resulting dataset contains the positions and rotations of the black rods, the rotations of the white rods and the camera image.

For deriving the positional shifts of the white rods in the data setup we use a traditional Computer Vision approach. We start by defining a one pixel wide column in the center of each rod and applying a binary thresholding. We search for connected groups of black pixels in this column to find the rubber stoppers which are mounted at each end of the rods. Using those stoppers we can find the center of each rod. We calculate the actual shift by converting the offset between the center of the table and the center of the rod from pixels to mm. We validated this approach by calculating the shifts of the black rods. We observed, that the reported shift sometimes differs from our calculations. As illustrated in Fig. 6, our calculations were more accurate. We conclude, that the rods were still moving when the image was taken resulting in an offset between the reported and the actual shift. The observed motion blur

pattern, as shown in Fig. 6, confirms this assumption. While the images are not blurred, the rods contain a vertical motion blur but no horizontal blurring. Since this motion blur will also be present when detecting the game state of a real Foosball game, we keep the faulty images in the dataset but use our calculated shift as our ground truth.

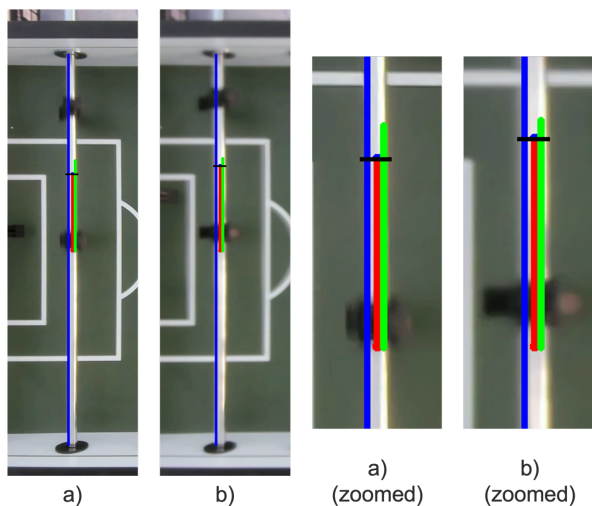


Figure 6: Outliers with a high deviation between the calculated shift (red line) and the reported shift (green line). While the motors report the final shift of the rod, regardless if the rod is still moving, the calculation is based on the current image. Therefore, the calculated shifts are more accurate.

Summing up, the derived data set consists of 500 images containing the corresponding shifts and rotations of the black and white figures. Due to our automated data capturing process using the accelerometers for the measurement of the rotations and an automated CV process for the shift calculation, we were able to avoid a manual labeling process. We note however, that in some occasions, minor manual adaptations were necessary to accommodate small deviations in the camera angle. The positions and rotations of the black figures are uniformly distributed. Due to the manual movement of the white rods, the positions and rotations could not reach a uniform distribution and include some bias towards specific values, c.f. Fig. 7. Albeit not attempting to detect the ball, we included it in the dataset to get more realistic images.

2.2 End-to-End Regressor Networks

For the game state detection on our Foosball table, we propose the utilization of common state-of-the-art Image Classification models like ResNet [12], EfficientNet [27] and MobileNet [16] as a backbone for feature extraction. As the shifts and rotations of the rods are continuous numerical variables, the problem at hand constitutes a regression problem. Thus, classification

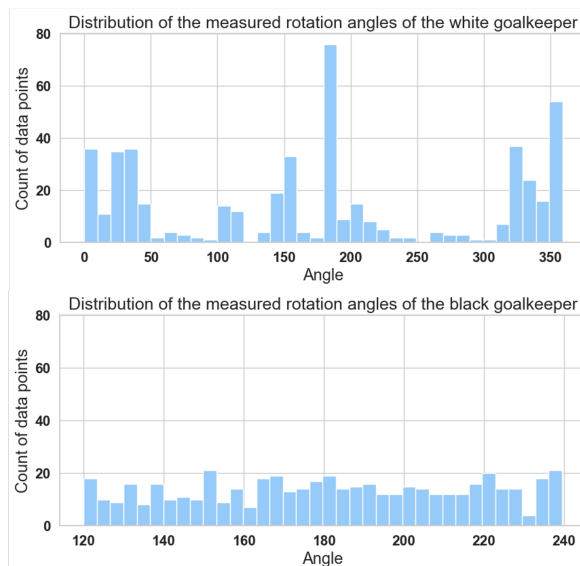


Figure 7: Distribution of the rotation of the black and white goalkeepers. The black goalkeeper was moved randomly by the computer which results in a uniform distribution of measured rotation angles. In contrast, the white goalkeeper was moved manually by hand resulting in a non-uniform distribution with a bias to specific angles. This effect is also present on the other white rods.

models/networks which typically consist of a feature extractor backbone and a classification head cannot be used directly. However, we may reuse the feature extractor (cf. [2]; which mostly consists of convolutional layers) and replace the classification head with a customized regression head. This approach fits within the idea of fine-tuning on a downstream task using parts of a neural network trained for another task on a large set of training data; in particular, we here use already proven feature extractor architectures trained for classification on ImageNet [6].

To account for the periodicity of angles, we decided to use $e^{i\varphi} = (\cos \varphi, \sin \varphi)$ as variable for regression, i.e., we predict the sine and cosine of the angles. Additionally, the shift is scaled to a range of -1 to 1 which balances the influence of the shift, the sine and the cosine. Ultimately, as a regression head, we used a linear layer with an output dimension of three and no activation function to predict the values $(s, \cos \varphi, \sin \varphi)$ where the symbol s denotes the shift.

To mitigate the problem of perspective distortion, which occurs due to the wide field-of-view of the camera, we use an individual regressor model per rod, resulting in 8 models which are trained and inferred sequentially. Each regressor uses a pre-defined cutout of the overall camera image. We custom-trained the 8 models using four different base architectures from ResNet18, ResNet50 [12], EfficientNetV2 [27] and

MobileNetV3 [16] as feature extractors. The weights of the feature extractors are initialized using transfer learning with the provided ImageNet1K [6] weights of the base models from PyTorch. The custom regression head is initialized randomly. Each regressor is trained over 50 epochs using the MSE loss function and the Adam optimizer [19] with a fixed learning rate of 0.001. During the training process, we used 80 % of our dataset (400 images) to train the CNNs and 20 % (100 images) for validation.

2.3 Data Provisioning System

As our work contributes to the automation of our Foosball table using DRL or Imitation Learning techniques, a data provisioning system needs to be implemented which should (a) use standardized formats; (b) be easily accessible with a minimal need of further Hard- or Software; (c) not introduce high latency into the system. We propose a publish-subscribe messaging system based on ZeroMQ [14] which satisfies those requirements.

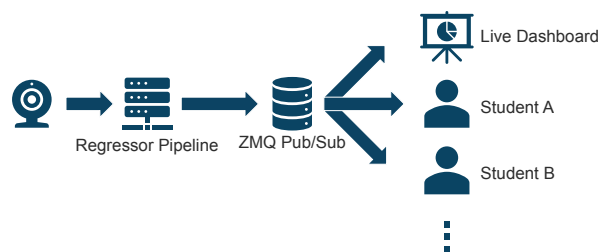


Figure 8: The proposed game state detection pipeline including the data provisioning system. Due to the utilization of ZeroMQ, multiple clients can connect and receive the game state data simultaneously.

The data producer side of the pipeline, in which the shift and rotation is predicted, publishes the JSON-formatted results through a ZeroMQ TCP socket. A client can then connect to this socket, subscribe to the messages and decode the JSON data. An illustration of the data provisioning system with multiple example clients is shown in Fig. 8. While this approach introduces some latency, it also has other major advantages. Firstly, multiple clients can connect to the socket simultaneously. Secondly, the TCP-based socket can also be shared through a network. This enables the possibility of the permanent installation of a dedicated game state detection server accessible to researchers. This would free resources for possible DRL experiments, as the game state detection must not be executed on the same hardware. The option of multiple connected clients allows for different simultaneous experiments as long as the actual Foosball table hardware is not required.

3 EVALUATION

As we utilized multiple feature extractor backbones for our regression model, we evaluate the performance of

the overall system per feature extractor based on the prediction accuracy measured as Mean Absolute Error (MAE) and the inference time. We define the following criteria:

- a) The MAE for the shift should not exceed 11 mm. The feet of the figure is 22 mm wide, so an error in the positional shift of ± 11 mm would still result in a straight shot. Since every rod has different shift limits, the percentage range per rod varies between ± 8.5 % for the defender and ± 20 % for the midfield rod.
- b) The MAE for the rotation should not exceed 42 degrees. A figure is able to block a shot if the feet of the figure are below the highest point of the ball. This is given at a maximum rotation of ± 47 degrees. Due to small rotations of the figure on impact with the ball, we deduct an offset of 5 degrees. Therefore, a figure which is predicted to stand vertically would be able to stop the ball up to an error of ± 42 degrees.
- c) The inference time should be lower than 16.6 ms. Since our present camera is only able to record at 60 FPS, this is the limit which we consider to be real-time. Ultimately, the system should be as fast as possible to provide as much information to the DRL systems as possible. A shorter inference time is therefore generally preferred.

3.1 Quantitative Evaluation

Evaluation of the Prediction Quality. We compare the results using a ResNet18, a ResNet50 [12], an EfficientNetV2 [27] and a MobileNetV3 [16] where we use an individual regressor model per rod as detailed in Section 2.2.

Table 1 shows our evaluation results for the shift detection. Considering the MAE averaged w.r.t. all rods, all feature extractor backbones passed the defined objective – they were below the predefined thresholds above. The shift detection is more accurate on rods which have a smaller movement range (e.g. the midfield rod) than on rods with a high movement range (e.g. the defense rod). Further, the predictions for the white rods have a higher accuracy than those for the black rods. We believe that a possible explanation for this finding can be that, in the training data, the shifts of the black rods follow a uniform distribution while the white rods (which were moved manually by hand) are distributed non-uniformly. Examining each rod individually, both regression networks based on the ResNet18 and on the ResNet50, stay within the predefined error tolerance, whereas the networks based on the MobileNetV3 and EfficientNetV2 backbones exceed the requirement on the white defender. Further, the ResNet18 based model

Feature Extractor	Black Rods				White Rods				Average
	Goal	Defense	Midfield	Striker	Goal	Defense	Midfield	Striker	
ResNet18	2.94	4.02	1.23	1.6	2.94	7.42	2.2	8.68	3.88
ResNet50	2.4	4.02	1.35	2.77	4.5	6.49	1.91	7.83	3.91
MobileNetV3	7.68	9.35	3.01	7.32	9.06	11.94	3.89	6.78	7.38
EfficientNetV2	9.82	5.97	2.06	3.75	5.5	12.73	4.47	8.53	6.6

Table 1: Mean absolute error of the predicted shift in mm per rod and feature extractor.

Feature Extractor	Black Rods				White Rods				Average
	Goal	Defense	Midfield	Striker	Goal	Defense	Midfield	Striker	
ResNet18	1.23	1.47	0.88	1.38	12.64	13.93	4.96	10.93	5.93
ResNet50	1.34	1.43	0.97	1.33	8.33	5.44	4.06	9.91	4.10
MobileNetV3	3.46	2.18	1.72	4.35	17.86	22.96	7.69	21.21	10.18
EfficientNetV2	6.31	2.14	2.29	1.69	14.26	25.32	13.59	28.68	11.79

Table 2: Mean absolute error of the predicted rotation angle in degrees per rod and feature extractor.

performed slightly better on average than the ResNet50 based model. In our opinion, these observations indicate that further training data is necessary to use the full potential of deeper networks.

In Table 2, we provide the mean absolute errors of the rotation angle prediction. Overall, all models achieved the required maximum average error of ± 42 degrees having large margins to the defined threshold. Similar to the shift prediction, the estimate for the white rods had higher MAEs compared to the black rods. In connection with this, we observe that, in addition to the non-uniform sampling, the rotation angle of the black rods is restricted to the range between 120 and 240 degrees (motor constrains) while the white rods can rotate freely. Therefore, the white rods have a higher range of possible values which can also explain the worse prediction accuracy. Furthermore, the ground truth data contains a mean error of ± 5 degrees due to the measurements with the accelerometers. As already observed in the shift detection, the ResNet based models yield lower MAE values compared to the MobileNetV3 and EfficientNetV2 based models.

Evaluation of the Inference Time. The inference times of the models were evaluated on four different systems: *System A*: Apple MacBook Pro 2018 with Intel Core i7 processor and AMD Radeon Pro 560X GPU; *System B*: Apple MacBook Pro 2021 with Apple M1 Pro processor; *System C*: PC with AMD Ryzen 9 5900X processor and NVIDIA RTX 3080 GPU; *System D*: Cloud VM with AMD EPYC-Milan processor and NVIDIA A100 80G PCIe GPU. All systems use the GPUs, either through the Metal Performance Shader (MPS) backend or through NVIDIA CUDA. The models were trained using PyTorch 2.1.0 and Torchvision 0.16.0 in a Python 3.9 environment.

Table 3 summarizes the mean and median inference times on the different systems. The inference times are measured as overall inference time and the infer-

ence time per rod. The overall time should roughly be about 8 times the inference time per rod, as all rods are inferred sequentially. Additionally, the corresponding FPS are calculated. Overall, the ResNet18 based model performed the best with the lowest inference time of 86.18 ms median on system C, thus achieving only 11.6 FPS which is significantly lower than the desired 60 FPS. The other evaluated CNN architectures could not achieve at least 10 FPS on average. The slightly better prediction quality of the ResNet50 based regressor is -in our opinion- not enough to justify the higher inference time of an additional 33.14 ms on the same hardware compared to the ResNet18 based model. If the sequential execution would be parallelized, the ResNet18 based model would probably achieve the desired 60 FPS considering the median inference time per rod of 10.77 ms on system C. The ARM-based MacBook (system B) would then also achieve the goal with 13.38 ms median inference time.

Albeit providing a higher performing GPU, system D could not reach a better performance compared to the other systems. Our conjecture is that the system is CPU-bottle-necked due to the sequential execution of the rods. We observed an average GPU utilization of 40 % on system C and 7 % on system D which supports this assumption. While all systems except the cloud VM showed only small deviations of 0 to 3 % between the median and mean inference times, system D resulted in high deviations of 9.51 % for the EfficientNetV2 based model and 14 to 18 % for the other models. The outliers in the inference times could indicate lower performing storage resulting in more time needed to move data between RAM and the GPU.

3.2 Qualitative Evaluation

In a qualitative evaluation and live tests on the Foosball table, we observed overall accurate results, e.g. illustrated in Fig. 2. There were several issues, in particular

Backbone	System	Inference Time (ms)		Inf. per Rod (ms)		FPS	
		Mean	Median	Mean	Median	Mean	Median
ResNet18	MacBook Pro 2018 (Sys. A)	288.51	286.27	36.06	35.78	3.47	3.49
	MacBook Pro 2021 (Sys. B)	108.03	107.11	13.50	13.38	9.26	9.34
	Gaming PC (Sys. C)	88.88	86.18	11.11	10.77	11.25	11.60
	Cloud VM (Sys. D)	126.23	106.94	15.78	13.36	7.92	9.35
ResNet50	MacBook Pro 2018 (Sys. A)	581.58	573.85	72.69	71.72	1.72	1.74
	MacBook Pro 2021 (Sys. B)	182.79	182.45	22.84	22.80	5.47	5.48
	Gaming PC (Sys. C)	120.83	119.32	15.10	14.91	8.28	8.38
	Cloud VM (Sys. D)	130.77	114.60	16.34	14.32	7.65	8.73
MobileNetV3	MacBook Pro 2018 (Sys. A)	497.21	501.49	62.14	62.58	2.01	1.99
	MacBook Pro 2021 (Sys. B)	156.76	156.44	19.59	19.55	6.38	6.39
	Gaming PC (Sys. C)	117.00	115.93	14.62	14.29	8.55	8.63
	Cloud VM (Sys. D)	122.81	105.14	15.35	13.14	8.14	9.51
EfficientNetV2	MacBook Pro 2018 (Sys. A)	1538.96	1496.68	192.36	187.08	0.65	0.67
	MacBook Pro 2021 (Sys. B)	402.34	400.27	50.29	50.03	2.49	2.50
	Gaming PC (Sys. C)	249.31	245.97	31.16	30.74	4.01	4.07
	Cloud VM (Sys. D)	235.70	215.24	29.46	26.90	4.24	4.65

Table 3: Mean and Median inference time for different feature extractor backbones on the different systems. All systems utilized GPU acceleration through either NVIDIA CUDA or Apple MPS. While the Cloud VM (system D) has the highest theoretical GPU performance, not all backbones could benefit from this. Instead, the lower performing RTX 3080 from system C resulted in a shorter median inference time for the small ResNet18 based model and almost equal times for the ResNet50 and MobileNetV3 based models. All systems except the cloud VM showed only small deviations between the mean and median inference time.

concerning lightning conditions and blur which we discuss next.

Dependence of the Prediction Quality on the Lightning Conditions. As illustrated in Fig. 9, the prediction accuracy is highly dependent on the lighting conditions which cannot be controlled. Our training data was captured with natural light in the summer while we tested the system in the winter with predominantly artificial light. As seen in Fig. 9, the artificial light (in the right images of each example) results in harder shadows with a brighter playing field, especially at the edges of the field. In contrast, the natural light showed only soft shadows. A more diverse dataset with different lighting conditions including natural and artificial light would improve the consistency of the system and provide a more generalized prediction model.

Dependence of the Prediction Quality on Blur. We observed an influence of blur in the images on the prediction accuracy. Since we use a standard webcam without any modifications, the exposure time and focus cannot be fixed. Commonly, a webcam uses a variable exposure time to control the brightness of the image, as the aperture is fixed. Therefore, in a darker scenario, the webcam uses a longer exposure time to get a bright image. The long exposure time can, however, result in motion blur especially on the fast moving figures of the Foosball table. Additionally, the variable focus leads to an automatic re-focusing of the camera which results in overall blurred images. In Fig. 10, the predicted rotation on the blurred, right images deviates about 11

degrees in *a*) and 20 degrees in *b*) compared with the same physical rotation in the non blurred counterparts in the left images.

Additionally, we observed a high influence of occluded figures and / or general unknown items in the images, e.g. by placing a hand inside the camera frame. This risk cannot be eliminated and should be addressed in the future.

4 CONCLUSION, DISCUSSION AND FUTURE WORK

In this paper, we presented a CNN based figure detection system for a semi-automatic Foosball table: the black team was controlled by motors and the white team was controlled by human players. More precisely, we first created and verified a ground truth dataset for training CNN-based detection models. Then, we have derived an improved figure detection system by basing on end-to-end learning. This contrasts the previous work [15] where an intermediate object detection step utilizing a YOLO detector was used. Further, we included the detection of all white figures and additionally the black figures in the detection system. We trained and evaluated our approach using various feature extractor backbones, including ResNets, MobileNets and EfficientNets. Finally, we proposed a data provisioning system based on ZeroMQ.

We demonstrated that our system is able to detect the shifts and rotations of all figures based on a camera

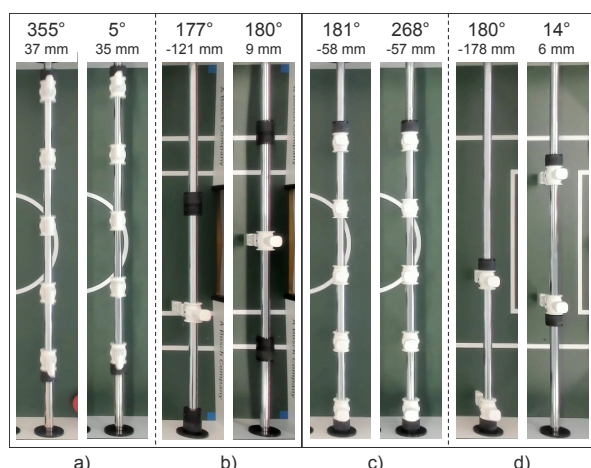


Figure 9: Examples for the influence of lighting conditions on the prediction accuracy. Each example is shown with natural light conditions (the same as in the training data) on the left and artificial light conditions on the right with a similar rotation angle. The artificial lighting results in harder shadows and a brighter background compared to the natural light. While the images in *a)* and *b)* predict the correct rotation in both lighting conditions, the examples in *c)* and *d)* show extreme examples of a deviation from the actual rotation in the artificial lighting conditions. In all cases, the position is predicted within the given boundaries.

image in a top-down perspective. The shift and rotation detection of our system satisfy our defined requirements for all figures. Using the ResNet18-based model, we achieved mean absolute errors of 3.88 mm for the position and 5.93 degrees for the rotations of all figures.

While the overall accuracy of the system satisfies the requirements we defined, some limitations remain. First, concerning stability, we observed that the lighting conditions and image blur can have significant influence on the prediction accuracy. Second, the present system does not achieve a game state detection in real-time, i.e. at 60 FPS. Means to address these issues could be as follows: Concerning speed, one possible solution is the parallelization of the regression models which are currently inferred sequentially. As shown in this paper, the inference time for one rod would achieve a real-time detection at 60 FPS. Concerning the stability of the system, one potential approach is the modification of the hardware of the Foosball table. It appears to be most promising to switch to another, manually controllable camera, as the current webcam shows clear drawbacks.

A further limitation is found in the Foosball table itself. The motors controlling the black rods are subject to a rotation limitation between 120 and 240 degrees on the driver level. This safety feature cannot be overwritten without modifying the hardware and should remain to reduce the maximum velocity of the ball. A manually played Foosball table does not have this issue. The

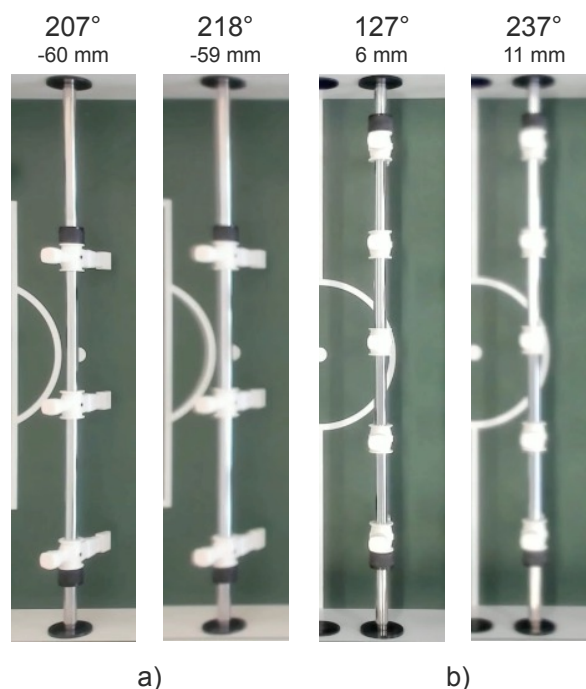


Figure 10: Two extreme examples for the influence of blurred images on the prediction quality. The left images of *a)* and *b)* are non blurred versions with the same positions and rotations as in the right images. The blurred images are a result of the camera performing a re-focusing of the whole image. We observed this effect to occur every 30-60 seconds.

same approach as described in Sec. 2.1 could be applied to generate a corresponding dataset including full 360 degree rotational movement of the black rods.

One aspect of future research is to further develop the system w.r.t. real-time capabilities and robustness such that it can be directly used in the automation process, i.e., provide the necessary information on the game state to the future RL agent controlling the non-human player in a robust way in real-time. Another line of future research is to employ (developments of) the proposed system for Imitation Learning. To this end, we plan to employ our game state detection system for capturing real Foosball games played by humans. This would improve the training of a DRL agent by reducing the need for On-Policy and / or simulation data. We note that for the creation of an Imitation Learning dataset, real-time detection is not necessary.

5 REFERENCES

- [1] Michael Aeberhard, Shane Connelly, Evan Tarr, and Nardis Walker. Single player foosball table with an autonomous opponent. *Georgia Tech, Elect. and Comp. Engineering*, 2007.
- [2] Neena Aloysius and M. Geetha. A review on deep convolutional neural networks. In *2017 International Conference on Communication and Signal Processing (ICCSP)*, pages 0588–0592, 2017.

- [3] Sven Bambach and Stefan Lee. Real-time foosball game state tracking. Technical report, School of Informatics and Computing, Indiana University, 2012.
- [4] Matevž Bošnjak and Gregor Klančar. Fast and reliable alternative to encoder-based measurements of multiple 2-dof rotary-linear transformable objects using a network of image sensors with application to table football. *Sensors*, 20(12), 2020.
- [5] Stefano De Blasi, Sebastian Klöser, Arne Müller, Robin Reuben, Fabian Sturm, and Timo Zerrer. Kicker: An industrial drive and control foosball system automated with deep reinforcement learning. *Journal of Intelligent & Robotic Systems*, 102(1):20, 2021.
- [6] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.
- [7] Nathaniel Enos, Patrick Fenelon, Skyler Goodell, and Nicholas Phillips. Football operator and optical soccer engine (foose). *University of Central Florida, Department of Electrical and Computer Engineering*, 2012.
- [8] Christopher J Fisher. Using an accelerometer for inclination sensing. *AN-1057, Application note, Analog Devices*, pages 1–8, 2010.
- [9] Adriatik Gashi, Elke Hergenröther, and Gunter Grieser. Efficient training of foosball agents using multi-agent competition. In Kohei Arai, editor, *Intelligent Computing. SAI 2023*, pages 472–492, Cham, 2023. Springer Nature Switzerland.
- [10] Zheng Ge, Songtao Liu, Feng Wang, Zeming Li, and Jian Sun. Yolox: Exceeding yolo series in 2021. *arXiv*, 2021.
- [11] Juan David Gutierrez-Franco, John Inlow, Jesse Graham, and Larry Huang. Automated foosball table. *California Polytechnic State University, Mech. Eng. Dep.*, 2013.
- [12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *arXiv*, 2015.
- [13] Nicolás Hernández, Javier Rebolledo, Javier Torres, Gonzalo Carvajal, and Francisco Vargas. Design of an experimental platform for the automation of the goalkeeper of a foosball table. In *2019 IEEE CHILECON*, pages 1–7. IEEE, 2019.
- [14] Pieter Hintjens. *ZeroMQ: messaging for many applications*. "O'Reilly Media, Inc.", 2013.
- [15] Ronny Horst, David Hagens, Elke Hergenröther, and Andreas Weinmann. Real time state detection of a foosball game using cnn-based computer vision. *SAI Computing Conference, London (accepted)*, 2024.
- [16] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, Quoc V. Le, and Hartwig Adam. Searching for mobilenetv3. *arXiv*, 2019.
- [17] Rob Janssen, Jeroen de Best, and René van de Molengraft. Real-time ball tracking in a semi-automated foosball table. In *RoboCup 2009: Robot Soccer World Cup XIII 13*, pages 128–139. Springer, 2010.
- [18] Rob Janssen, Mark Verrijt, Jeroen de Best, and René van de Molengraft. Ball localization and tracking in a highly dynamic table soccer environment. *Mechatronics*, 22(4):503–514, 2012.
- [19] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.
- [20] Hiroaki Kitano, Minoru Asada, Yasuo Kuniyoshi, Itsuki Noda, Eiichi Osawa, and Hitoshi Matsubara. Robocup: A challenge problem for ai. *AI Magazine*, 18(1), 1997.
- [21] Dani Mohebi. The study of semi-automated foosball table. *Tampere University of Applied Sciences, Degree Program in Mechanical Engineering*, 2022.
- [22] OpenAI, Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Dębiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, Rafal Józefowicz, Scott Gray, Catherine Olsson, Jakub Pachocki, Michael Petrov, Henrique Pondé de Oliveira Pinto, Jonathan Raiman, Tim Salimans, Jeremy Schlatter, Jonas Schneider, Szymon Sidor, Ilya Sutskever, Jie Tang, Filip Wolski, and Susan Zhang. Dota 2 with large scale deep reinforcement learning. *arXiv*, 2019.
- [23] Duc An Pham and Trung Nghia Pham. Determination of a tilt angle for the automatic balancing system with the inertial measurement unit mpu6050. In *Proceedings of the AMAS2021*, pages 349–354, Cham, 2022. Springer.
- [24] Tobias Rohrer, Ludwig Samuel, Adriatik Gashi, Gunter Grieser, and Elke Hergenröther. Foosball table goalkeeper automation using reinforcement learning. *LWDA*, pages 173–182, 2021.
- [25] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, Jan 2016.
- [26] Jim R Stefani, Alex J Herpy, Brett Gordon Jaeger, Kevin S Haydon, and Derek Alan Hamel. Automated foosball table. *California Polytechnic State University, Mech. Eng. Dep.*, 2014.
- [27] Mingxing Tan and Quoc V. Le. Efficientnetv2: Smaller models and faster training. *arXiv*, 2021.
- [28] Thilo Weigel. Kiro - a table soccer robot ready for the market. *Proceedings IEEE Int. Conf. Robotics and Automation*, 2005:4266–4271, 2005.
- [29] Thilo Weigel and Bernhard Nebel. Kiro – an autonomous table soccer player. In G. Kaminka, Pedro Lima, and Raúl Rojas, editors, *RoboCup 2002: Robot Soccer World Cup VI*, pages 384–392. Springer, 2003.
- [30] Thilo Weigel, Dapeng Zhang, Klaus Rechert, and Bernhard Nebel. Adaptive vision for playing table soccer. In Susanne Biundo, Thom Frühwirth, and Günther Palm, editors, *KI 2004: Advances in AI*, pages 424–438. Springer, 2004.
- [31] Dapeng Zhang and Bernhard Nebel. Learning a table soccer robot a new action sequence by observing and imitating. In *Proceedings of the Third AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 3, pages 61–66. AAAI Press, 2007.

MinBackProp – Backpropagating through Minimal Solvers

Diana Sungatullina^{1,2}
diana.sungatullina@cvut.cz

Tomas Pajdla¹
pajdla@cvut.cz

¹ CIIRC, CTU in Prague, Jugoslavských partyzanu 1580/3, 16 000, Prague, Czech Republic

² FEE, CTU in Prague, Technicka 2, 160 00 Prague, Czech Republic

ABSTRACT

We present an approach to backpropagating through minimal problem solvers in end-to-end neural network training. Traditional methods relying on manually constructed formulas, finite differences, and autograd are laborious, approximate, and unstable for complex minimal problem solvers. We show that using the Implicit function theorem (IFT) to calculate derivatives to backpropagate through the solution of a minimal problem solver is simple, fast, and stable. We compare our approach to (i) using the standard autograd on minimal problem solvers and relate it to existing backpropagation formulas through SVD-based and Eig-based solvers and (ii) implementing the backprop with an existing PyTorch Deep Declarative Networks (DDN) framework [GHC22]. We demonstrate our technique on a toy example of training outlier-rejection weights for 3D point registration and on a real application of training an outlier-rejection and RANSAC sampling network in image matching. Our method provides 100% stability and is 10 times faster compared to autograd, which is unstable and slow, and compared to DDN, which is stable but also slow.

Keywords

minimal solvers, epipolar geometry, backpropagation, outlier removal, implicit function theorem

1 INTRODUCTION

Recently, minimal problem solvers [Nis04, SNKS05, KBP08, LOÅ⁺18, MVP22] have been incorporated into end-to-end machine learning pipelines in camera localization [BKN⁺17], image matching [WPS⁺23], and geometric model estimation by RANSAC [BR19]. The key problem with using minimal problem solvers in end-to-end neural network training is to make them differentiable for backpropagation.

Early attempts to backpropagate through minimal problem solvers used explicit derivative formulas [IVS15, DYH⁺18, RK18] and finite differences [BKN⁺17] to compute derivatives for backpropagation. Recent work [WPS⁺23] proposed computing the derivatives using autograd. These are valid approaches, but they can still be considerably improved. Manual differentiation is laborious and must be done repeatedly for every new problem. Finite differences are approximate and are prone to numerical errors. Using autograd is also limited to relatively simple minimal problem solvers since, for more complex

solvers with large templates [MVP22], differentiating the templates becomes unstable due to, e.g., vanishing of the gradients [BSF94].

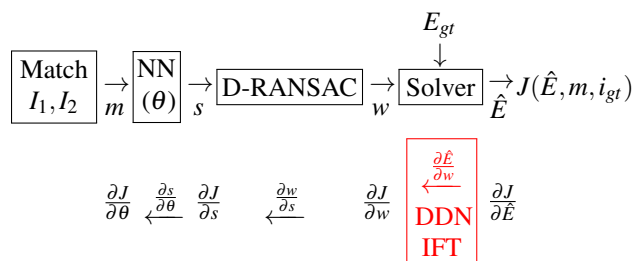


Figure 1: A typical end-to-end training pipeline with a minimal problem Solver [WPS⁺23] that trains a Neural Network (NN) to predict correct matches. Forward pass: Tentative handcrafted matches m between images I_1, I_2 are clarified using NN [ZGZ⁺21] parametrized by trained weights θ , and scores s for these matches are computed. Differentiable D-RANSAC selects a minimal data sample w using the scores s ; the Solver computes a model \hat{E} , which is scored by the loss J using the correct matches $m(i_{gt})$ with the ground truth inlier indicator i_{gt} . The groundtruth E_{gt} is passed to the Solver to choose the closest model. Backward pass: Gradient $\frac{\partial J}{\partial \theta}$ for training weights θ is computed by the chain rule. *The key issue is robustly and efficiently backpropagating through the Solver. We propose to use the Implicit function theorem (IFT) directly or implement the backpropagation via the PyTorch Deep Declarative Network (DDN) machinery [GHC22].*

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

Let us illustrate using the minimal problem solvers in end-to-end neural network training for calibrated image matching [WPS⁺23], Fig. 1. First, a pair of images I_1, I_2 the RootSIFT [AZ12] pre-detected features are used to produce tentative matches m , and the Neural Network $s = \text{NN}(\theta; m)$ computes scores s of the matches m . The network is parameterized by weights θ , the learned pipeline parameters. Next, the differentiable $w = \text{D-RANSAC}(m, s)$ selects a minimal data sample w from m using the s scores. The sample w is passed to the Essential matrix solver $E = \text{Solver}(w)$, which computes up to 10 E candidates and selects the best \hat{E} that is closer to the known ground truth E_{gt} . The Solver can thus be seen as solving a discrete optimization problem consisting of first solving a set of polynomial equations and then choosing the optimal one w.r.t. E_{gt} . Having \hat{E} , a projection-based loss $J(\hat{E}, M, i_{gt})$ is calculated from all tentative matches m and the indicator i_{gt} of the known correct (inlier) matches. As a function to be optimized, the pipeline is the composition

$$J(w; E_{gt}, i_{gt}) = J(\hat{E}; m, i_{gt}) \circ \hat{E}(w; E_{gt}) \circ w(s; m) \circ s(\theta; m)$$

where E_{gt} , i_{gt} , and m can be seen as parameters provided by training data.

To learn weights θ , the (transposed) gradient

$$\frac{\partial L}{\partial \theta} = \frac{\partial L}{\partial \hat{E}} \frac{\partial \hat{E}}{\partial w} \frac{\partial w}{\partial s} \frac{\partial s}{\partial \theta}$$

of J w.r.t. θ has to be calculated. Calculating $\frac{\partial L}{\partial \hat{E}}, \frac{\partial w}{\partial s}, \frac{\partial s}{\partial \theta}$ is straightforward and efficient using autograd. Calculating $\frac{\partial \hat{E}}{\partial w}$ could also be attempted by autograd. We show that it often fails even for a simple minimal problem of computing the calibrated relative camera pose from five image matches [Nis04]. Our main contribution is to show how to calculate $\frac{\partial \hat{E}}{\partial w}$ robustly by the existing PyTorch machinery of Deep Declarative Networks (DDN) [GHC22] and *robustly an efficiently* using the Implicit function theorem (IFT).

1.1 Motivation

Minimal problem solvers can be complex. Symbolic-numeric minimal problem solvers [EM05, Nis04, LOÅ⁺18, MVP22] consist of potentially very large templates (formulas combined with the Gauss-Jordan elimination) to construct a matrix whose eigenvectors provide the solutions. The size of the templates is, for systems with a finite number of solutions and more than three unknowns, proportional to a polynomial in D^n , where D is the mean degree of input constraints and n is the number of unknowns [HL11]. In practice, it is not always necessary to construct a full Groebner basis [CLO15], but practical automatic solver generation algorithms [KBP08, LKZ17, MVP22] still

produce very large templates for problems with more variables. See, for example, the sizes of the templates in [MVP22] and the related discussion there.

Therefore, backpropagating through such templates using autograd on the fully expanded computational graph, as in [WPS⁺23], is generally slower and potentially less stable. Another issue is the backpropagation stability through large eigenvector decompositions [DYH⁺18]. Backpropagation through constraints at the optimum using the Implicit function theorem (for active KKT constraints in semi-algebraic cases) is much simpler, with linear complexity in the size of the input constraints, and is stable.

1.2 Contribution

We investigate backpropagation methods through minimal problem solvers in end-to-end neural network training. We show that using the Implicit function theorem to calculate derivatives to backpropagate through the solution of a minimal problem solver is simple, fast, and stable. We provide a direct implementation of IFT backpropagation that is stable and fast. We also show how a stable but slower backpropagation can be implemented using the existing PyTorch Deep Declarative Networks framework [GHC22]. This second approach may be helpful for quick testing of functionality before a more efficient IFT backpropagation is implemented.

We compare our approach to using the standard autograd on minimal problem solvers [WPS⁺23] and relate it to the existing formulas for backpropagating through SVD-based and Eig-based solvers [IVS15, DYH⁺18].

We demonstrate our technique on toy examples of training outlier-rejection weights for 3D point registration and on a real application of training an outlier-rejection and RANSAC sampling network [WPS⁺23] in two image matching and camera relative pose computation. Our IFT backpropagation provides 100% stability and is 10 times faster than unstable autograd and DDN.

2 PREVIOUS WORK

The interest in backpropagating through minimal problem solvers started with attempting to develop a differentiable version of RANSAC that could be used in end-to-end learning pipelines.

In seminal work [BKN⁺17], a differentiable pipeline for camera localization, including RANSAC and PNP solver, has been suggested. The PNP solver was so efficient that it was possible to estimate the derivatives of its output by numerical central differences. We show in Example 1 how to easily estimate the derivatives using the Implicit function theorem.

In [RK18], a method for robust estimation of fundamental matrices embedded in an end-to-end training pipeline was proposed. The goal was to gradually learn

weights for scoring tentative matches to suppress mismatches in a sequence of weighted least-squares problems. Here, the fundamental matrices are computed via SVD. Backpropagating through SVD uses derivatives computed by explicit formulas derived in [IVS15]. Computing derivatives explicitly by formulas is the best approach, but it can be laborious when done manually, as in [IVS15]. We use this example to show in Section 6.1 that our approach based on DDN/IFT provides equally correct results.

Recent work [WPS⁺23] presents another differentiable pipeline for image matching that includes a differentiable version of RANSAC and minimal problem solvers for fundamental and essential matrix computation. Here, the solvers may use minimal data samples, i.e., 7 and 5 correspondences, respectively, and solve for the matrices using symbolic-numeric algebraic minimal problem solvers, e.g. [Nis04]. To backpropagate through the minimal problem solvers, [WPS⁺23] uses autograd on the templates and the following Eigendecomposition. As explained above, this is a possible approach for small templates but should be replaced by our approach whenever possible. We show in experiments, Section 6.2, that using autograd often fails even for the relatively simple minimal problem solver of essential matrix estimation. In contrast, our approach based on using the Implicit function theorem delivers 100% stable results and is 10 times faster.

Our approach relies on using the Implicit function theorem [Rud76], which was already used to implement backpropagation for SVD and Eigendecomposition [IVS15, DYH⁺18] computation blocks of learning pipelines. These approaches manually compute formulas for Jacobians and gradients and implement backward passes for the modules. It is the simplest, fastest, and most robust approach currently possible. It should be used in all final implementations.

To provide a simple but slow error-prone engineering solution for implementing backpropagation through algebraic solvers, we exploit the DDN [GHC22] that implement a framework to backpropagate through algebraic and semi-algebraic optimizers. A similar approach has been suggested in [ZGM⁺20], but we prefer [GHC22] since it provides PyTorch framework implementation.

2.1 IFT end-to-end learning

The implicit function theorem (IFT) was used in several interesting works related to using geometric optimization in end-to-end learning. None of the works deals with minimal problems, but they are very related to the tasks that are solved with minimal problem solvers, and thus we comment on them here.

In [CPC⁺20], IFT is used to backpropagate through the least-squares PNP optimization solver for end-to-end

learnable geometric vision. Explicit formulas for the Jacobians of the PNP least-squares solvers are derived, and it is shown that they lead to accurate and stable backpropagation. This is an interesting work because it presents a particular generalization of polynomial problems to rational problems but concentrates on one specific geometrical problem.

In [CWW⁺22], there is a method for backpropagation through a probabilistic PNP. Technically, this approach does not use IFT since the PNP in this formulation is not a solution to equations, but a function providing a distribution over the domain of camera poses. The derivatives of the output are thus readily computable by autograd or by explicit formulas.

In [AK17], the OptNet layer for propagating through optimization problems solved by layers of small quadratic programs is developed using IFT. The aim is to provide an efficient and stable backpropagation that can be implemented on GPU. This is an interesting work since it shows how to backpropagate through an important class of optimization problems. However, it can't address general minimal problems since it only addresses the problems specified as a quadratic program with linear constraints.

3 USING THE IMPLICIT FUNCTION THEOREM

Let us now explain and demonstrate how to compute derivatives of the output of a minimal problem solver using the Implicit function theorem.

Technically, minimal problem solvers generate solutions $\mathbf{x}(\mathbf{a})$ to systems of polynomial equations $f_1(\mathbf{a}), \dots, f_K(\mathbf{a})$ based on input parameters \mathbf{a} . To backpropagate through the minimal problem solvers, it is necessary and sufficient to compute the derivatives $\frac{\partial \mathbf{x}}{\partial \mathbf{a}}$ at each solution \mathbf{x} w.r.t. parameters \mathbf{a} .

We now explain how the Implicit function theorem [Rud76] can be used to implement simple, efficient, and numerically stable computation of the derivatives of \mathbf{x} w.r.t. \mathbf{a} . Let us start with the formulation of the theorem itself, and then consider a simple example.

Theorem 1 (Differentiating roots of a polynomial system). *Let $H(\mathbf{a}) = h_1(\mathbf{x}, \mathbf{a}), \dots, h_K(\mathbf{x}, \mathbf{a})$ be a sequence of K complex polynomials in N unknowns $\mathbf{x} = x_1, \dots, x_N$ and M unknowns $\mathbf{a} = a_1, \dots, a_M$. Let $\mathbf{x}(\mathbf{b})$ be an isolated multiplicity-one solution to $H(\mathbf{a})$ in \mathbf{x} for $\mathbf{a} := \mathbf{b} \in \mathbb{C}^M$. Then,*

$$\left[\frac{\partial x_n(\mathbf{a})}{\partial a_m}(\mathbf{b}) \right] = - \left[\frac{\partial h_k(\mathbf{x}, \mathbf{a})}{\partial x_n}(\mathbf{x}(\mathbf{b}), \mathbf{b}) \right]^+ \left[\frac{\partial h_k(\mathbf{x}, \mathbf{a})}{\partial a_m}(\mathbf{x}(\mathbf{b}), \mathbf{b}) \right]$$

with $k = 1, \dots, K$, $m = 1, \dots, M$, $n = 1, \dots, N$, and $[A]^+$ denoting the pseudoinverse [Mey01] of the (Jacobian) matrix $[A]$.

Proof. Informal statement: The Implicit function theorem is a very standard fact. Here we use it in a special situation when functions are polynomials, since we focus on the Computer Vision problems. The main issue is thus ensuring that the assumptions of the Implicit function theorem are satisfied. For polynomials, which are continuous, infinitely differentiable functions, the assumptions of the Implicit function theorem are satisfied iff the solutions to the polynomials are isolated and have multiplicity one. See Supplementary Material, Section 9 for more details. \square

Let us demonstrate the above theorem with an example of the classical computer vision P3P minimal problem of solving the absolute pose of the calibrated camera [FB81].

Example 1. Consider a general configuration of three 3D points $\mathbf{A}_1, \mathbf{A}_2, \mathbf{A}_3 \in \mathbb{R}^3$, their respective calibrated image projections represented by homogeneous coordinates $\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3 \in \mathbb{R}^3$ and the vector $\mathbf{x} = [x_1, x_2, x_3]^\top \in \mathbb{C}^3$ of the depths of the points. Stack all 3D points and image points in a single vector $\mathbf{a} = [\mathbf{A}_1; \mathbf{A}_2; \mathbf{A}_3; \mathbf{a}_1; \mathbf{a}_2; \mathbf{a}_3] \in \mathbb{R}^{18}$. The 3D points, their projections, and depths are related by

$$\begin{aligned} 0 &= h_1(\mathbf{x}, \mathbf{a}) = \|\mathbf{A}_1 - \mathbf{A}_2\|^2 - \|x_1 \mathbf{a}_1 - x_2 \mathbf{a}_2\|^2 \\ 0 &= h_2(\mathbf{x}, \mathbf{a}) = \|\mathbf{A}_2 - \mathbf{A}_3\|^2 - \|x_2 \mathbf{a}_2 - x_3 \mathbf{a}_3\|^2 \\ 0 &= h_3(\mathbf{x}, \mathbf{a}) = \|\mathbf{A}_3 - \mathbf{A}_1\|^2 - \|x_3 \mathbf{a}_3 - x_1 \mathbf{a}_1\|^2 \end{aligned}$$

The calibrated absolute camera pose computation solves for the depths $\mathbf{x}(\mathbf{a})$ as functions of the parameters \mathbf{a} . In this example, $K = 3$, $N = 3$, and $M = 18$. The polynomial system above generically has 8 multiplicity-one solutions $\mathbf{x}_s(\mathbf{a}), s = 1, \dots, 8$ [FB81]. Now, we compute the derivatives of the solutions for a generic parameter vector \mathbf{b} . Let us first compute the derivatives of the functions $h_k(\mathbf{x}, \mathbf{a})$ w.r.t \mathbf{x} and \mathbf{a}

$$\begin{aligned} J_{\mathbf{x}}(\mathbf{x}, \mathbf{a}) &= \left[\frac{\partial h_k(\mathbf{x}, \mathbf{a})}{\partial x_n} \right] = \begin{bmatrix} \frac{\partial h_1}{\partial x_1} & \frac{\partial h_1}{\partial x_2} & \frac{\partial h_1}{\partial x_3} \\ \frac{\partial h_2}{\partial x_1} & \frac{\partial h_2}{\partial x_2} & \frac{\partial h_2}{\partial x_3} \\ \frac{\partial h_3}{\partial x_1} & \frac{\partial h_3}{\partial x_2} & \frac{\partial h_3}{\partial x_3} \end{bmatrix} (\mathbf{x}, \mathbf{a}) = \\ &= 2 \begin{bmatrix} -\mathbf{a}_1^\top (x_1 \mathbf{a}_1 - x_2 \mathbf{a}_2) & \mathbf{a}_2^\top (x_1 \mathbf{a}_1 - x_2 \mathbf{a}_2) & 0 \\ 0 & -\mathbf{a}_2^\top (x_2 \mathbf{a}_2 - x_3 \mathbf{a}_3) & \mathbf{a}_3^\top (x_2 \mathbf{a}_2 - x_3 \mathbf{a}_3) \\ \mathbf{a}_1^\top (x_3 \mathbf{a}_3 - x_1 \mathbf{a}_1) & 0 & -\mathbf{a}_3^\top (x_3 \mathbf{a}_3 - x_1 \mathbf{a}_1) \end{bmatrix} \\ J_{\mathbf{a}}(\mathbf{x}, \mathbf{a}) &= \left[\frac{\partial h_k(\mathbf{x}, \mathbf{a})}{\partial a_m} \right] = \begin{bmatrix} \frac{\partial h_1}{\partial a_1} & \dots & \frac{\partial h_1}{\partial a_{18}} \\ \vdots & \ddots & \vdots \\ \frac{\partial h_3}{\partial a_1} & \dots & \frac{\partial h_3}{\partial a_{18}} \end{bmatrix} (\mathbf{x}, \mathbf{a}) = [K_1 | K_2] \\ K_1 &= 2 \begin{bmatrix} \mathbf{A}_1^\top - \mathbf{A}_2^\top & \mathbf{A}_2^\top - \mathbf{A}_1^\top & 0 \\ 0 & \mathbf{A}_2^\top - \mathbf{A}_3^\top & \mathbf{A}_3^\top - \mathbf{A}_2^\top \\ \mathbf{A}_1^\top - \mathbf{A}_3^\top & 0 & \mathbf{A}_3^\top - \mathbf{A}_1^\top \end{bmatrix} \\ K_2 &= 2 \begin{bmatrix} -x_1 (x_1 \mathbf{a}_1 - x_2 \mathbf{a}_2) & x_2 (x_1 \mathbf{a}_1 - x_2 \mathbf{a}_2) & 0 \\ 0 & -x_2 (x_2 \mathbf{a}_2 - x_3 \mathbf{a}_3) & x_3 (x_2 \mathbf{a}_2 - x_3 \mathbf{a}_3) \\ x_1 (x_3 \mathbf{a}_3 - x_1 \mathbf{a}_1) & 0 & -x_3 (x_3 \mathbf{a}_3 - x_1 \mathbf{a}_1) \end{bmatrix} \end{aligned}$$

Next we evaluate $J_{\mathbf{x}}(\mathbf{x}, \mathbf{a})$ and $J_{\mathbf{a}}(\mathbf{x}, \mathbf{a})$ at a generic parameter vector $\mathbf{b} = [0, 0, 3, 2, 0, 3, 0, 6, 3, -1/3, -1/3, 1,$

$1/3, -1/3, 1, -1/3, 5/3, 1]$ and, e.g., the first solution $\hat{\mathbf{x}}(\mathbf{b}) = [3, 3, 3]$

$$\begin{aligned} J_{\mathbf{x}}(\hat{\mathbf{x}}(\mathbf{b}), \mathbf{b}) &= \begin{bmatrix} -1.33 & -1.33 & 0.00 \\ 0.00 & -5.33 & -21.33 \\ -4.00 & 0.00 & -20.00 \end{bmatrix} \\ J_{\mathbf{a}}(\hat{\mathbf{x}}(\mathbf{b}), \mathbf{b}) &= \begin{bmatrix} -4 & 0 & 0 & 4 & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & 0 & 0 & 4 & -12 & 0 & \dots & 12 & -36 & 0 \\ 0 & -12 & 0 & 0 & 0 & 0 & \dots & 0 & -36 & 0 \end{bmatrix} \end{aligned}$$

and compute

$$\begin{aligned} \left[\frac{\partial \hat{x}_n(\mathbf{a})}{\partial a_m} \right] (\mathbf{b}) &= -J_{\mathbf{x}}(\hat{\mathbf{x}}(\mathbf{b}), \mathbf{b})^{-1} J_{\mathbf{a}}(\hat{\mathbf{x}}(\mathbf{b}), \mathbf{b}) = \\ &= \begin{bmatrix} 1.66 & 1.33 & 0 & \dots & 1.25 & 0.24 & 0 \\ 1.33 & -1.33 & 0 & \dots & -1.25 & -0.24 & 0 \\ -0.33 & 0.33 & 0 & \dots & -0.25 & -1.75 & 0 \end{bmatrix}. \end{aligned}$$

4 IMPLEMENTING MINBACKPROP WITH IFT

The first option to implement the backpropagation for a minimal problem solver is to use the Implicit function theorem in a straightforward way (as in Example 1). This has been done before, e.g., in [IVS15] for SVD and EIG solvers. To use the Implicit function theorem directly, one can construct a system of polynomial equations, compute the derivatives of the system w.r.t. inputs and outputs manually or with the help of a computer algebra system [CSC], and follow the theorem 1 to compute the Jacobian of the output with respect to the input. In case the system of equations is overdetermined, one can write down the Lagrange multipliers method and then use the IFT for the new system.

Applying the Implicit function theorem involves inverting the Jacobian of the polynomial system w.r.t. the output, so before inverting the Jacobian one can check if the matrix is full-rank using SVD. If it is not, one can randomly choose a full-rank Jacobian from the batch to invert.

5 IMPLEMENTING MINBACKPROP WITH DDN

The second option to backpropagate through a minimal problem solver is to use a fully automatic method to compute the derivatives. The Deep Declarative Networks (DDN) framework, introduced in [GHC22], provides a PyTorch implementation allowing backpropagation through optimization solvers, including algebraic, semi-algebraic (i.e., with inequalities), and non-polynomial problems. Next, we explain how to use the DDN framework for implementing backpropagation for minimal problem solvers.

Declarative node: The DDN framework introduces declarative nodes that take input w into the optimal solutions

$$\hat{y}(w) = \arg \min_{y \in C} f(w, y)$$

where C is a constraint set.

Constraint set C : For algebraic minimal problem

solvers, the set $C = \{y: h_1(y) = 0, \dots, h_K(y) = 0\}$ is an algebraic variety [CLO15] consisting of a finite number of points. Hence, in this case, the algebraic variety is a finite set of solutions to some system of polynomial equations $H = [h_1, \dots, h_K]$. Assuming the genericity of data, which is often guaranteed by having noisy measurements, solutions to H have all multiplicity equal to one. Hence it is easy to compute derivatives of the solution to H w.r.t. parameters as demonstrated in Section 3.

Loss f : The low-level loss depends on a particular problem. DDN framework implements backpropagation for a general declarative node and thus expects a loss function $f(w, y)$ with non-degenerate derivatives. For instance, in Section 6.2, thus, one must “invent” such a loss function for minimal problem solvers since they provide exact solutions that always satisfy equations H exactly. A natural choice is to use the sum of squares of the equation residuals from a subset of H , i.e., $f = \sum_{h_i \in G \subseteq H} h_i^2$, such that the derivatives of f are non-degenerate. A possible choice is to use a random linear combination of some h_i ’s. For instance, in Section 6.2, we use the sum of squares of the linear constraints only to keep f simple and non-degenerate.

It is important to mention that in the pipeline of [WPS⁺23], the “Solver” not only computes the solutions y to a polynomial system H but also selects the best solution \hat{y} by finding \hat{y} that is closest to the ground-truth y_{gt} in $\|\hat{y} - y_{gt}\|^2$ sense. This discrete optimization step does not influence the derivatives of \hat{y} w.r.t w since, locally, small changes of w do not influence the result of this discrete optimization as small changes to w generically do not change the result of the discrete optimization.

End-to-end training formulation: End-to-end training of a pipeline with a declarative node is in [GHC22] formulated as the following bi-level optimization problem

$$\begin{aligned} & \min_w J(w, \hat{y}(w)) \\ & \text{s. t. } \hat{y}(w) \in \arg \min_{y \in C} f(w, y) \\ & \quad \text{with } C = \{y: h(y) = 0\}, \end{aligned} \quad (1)$$

where $J(w, y)$ is an upper-level loss, $f(w, y)$ is a low-level loss, $h(y)$ are constraints, w is the input parameters to the declarative node, y is the model to be estimated, and C is a constraint set. The relation $w \mapsto \hat{y}$ is defined as a solution to the low-level optimization problem provided by the declarative node.

Gradients for minimal problem solvers: To minimize the loss $J(w, \hat{y})$ in (1) via the gradient descent,

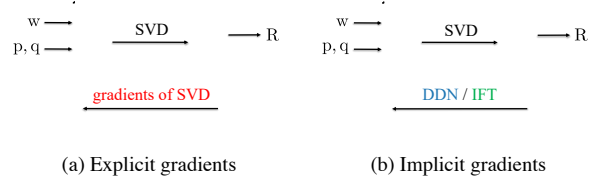


Figure 2: The figure demonstrates the backpropagation for the toy example for the 3D point registration with an outlier. The forward pass is the Kabsch algorithm [Kab76] (SVD) and remains the same for both explicit and implicit methods. The backward pass is performed explicitly via the closed-form gradients of SVD [IVS15] (a) and implicitly, using the Deep Declarative Networks (DDN) [GHC22] and the Implicit function theorem (IFT) (b).

we need to compute $DJ(w, \hat{y})^1$. In our formulation, the loss function $J(w, \hat{y})$ does not depend on w explicitly. Therefore

$$DJ(w, \hat{y}) = D_y J(w, \hat{y}) D\hat{y}(w). \quad (2)$$

The gradients $D_y J(w, \hat{y})$ are the gradients computed before the declarative block and the $D\hat{y}(w)$ are the gradients through the solution of the low-level optimization problem.

Declarative node specification: The important feature of the DDN framework is that one does not have to provide formulas for the derivatives explicitly. It is enough to specify a declarative node by defining the loss $f(w, y)$ and the constraints $H = [h_1, \dots, h_K]$. The backward pass based on the derivatives is then automatically computed by the DDN framework. This makes the implementation of backpropagation through minimal problem solvers easy and robust.

6 EXPERIMENTS

In our experiments, we compute the $D_y(w)$ for a solution of a minimal problem in three ways. First, we compute the gradients using the closed-form solution and backpropagate explicitly using autograd. Secondly, we compute the gradients $D_y(w)$ via the Implicit function theorem and backpropagate through a solution by using them directly. Finally, we use the Deep Declarative Networks framework to implement the backpropagation by specifying a suitable optimization problem. Our experiments show that the derivatives of a solution

¹ Here we use the notation from [GHC22] where $DJ(w, \hat{y})$ denotes the total derivative $\frac{dJ(w, \hat{y}(w))}{dw}$ of loss $J(w, \hat{y}(w))$ w.r.t w when $\hat{y}(w)$ is considered a function of w , $D_w J(w, \hat{y})$ means partial derivatives $\frac{\partial J(w, \hat{y})}{\partial w}$ of $J(w, \hat{y})$ w.r.t. w when \hat{y} is considered fixed, and $D_{\hat{y}} J(w, \hat{y})$ means partial derivatives $\frac{\partial J(w, \hat{y})}{\partial \hat{y}}$ of $J(w, \hat{y})$ w.r.t. \hat{y} when w is consider fixed.

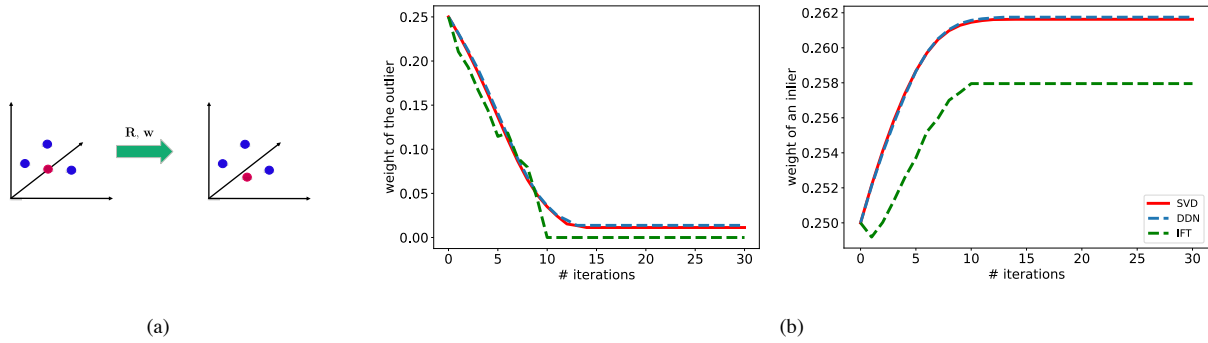


Figure 3: A toy example for the 3D point registration problem with an outlier: (a) visualization of four points before and after rotation R with inlier/outlier weights w , blue points denote inliers, and the magenta point is the outlier; (b) the weight for the outlier (left) and the weight for an inlier (right) during the optimization. The red color indicates explicit computation of the gradients, the blue color denotes backpropagation with the DDN, and the green color is backpropagation with the IFT. One can see that the gradients computed with the DDN and the IFT approximate the gradients of SVD computed in closed form.

to the minimal problem can be computed implicitly; they accurately approximate the derivatives computed explicitly, leading to more stable results.

We consider two types of experiments to show the effectiveness of the proposed method. Firstly, we consider a toy example for the 3D point registration with an outlier. We chose this task since a solution to the minimal problem during the forward pass can be computed using SVD, and the gradients of SVD can be computed both explicitly, using the closed-form solution, and implicitly, using the Implicit function theorem and the Deep Declarative Networks. Secondly, we incorporate our MinBackProp into the existing state-of-the-art epipolar geometry framework, backpropagate through it, and compare the results on the real data.

Our main goal is to show that we obtain equivalent behavior of the whole learning pipeline as the baseline [WPS⁺23] but in a more stable and efficient way. Note that we do not aim to improve the learning task of [WPS⁺23] but demonstrate better computational machinery for solving it.

6.1 3D Point Registration with an Outlier

Given two sets P and Q of 3D points, a point $p_i \in P$ corresponds to the point $q_i \in Q$, $i = 1, \dots, N$, where N is the number of points. We want to find a rotation matrix $R \in \mathbb{R}^{3 \times 3}$: $q_i = R p_i$, $\forall i$. Given the ground-truth rotation matrix $R_{\text{true}} \in \mathbb{R}^{3 \times 3}$, we define the weights w_i , $i = 1, \dots, N$, for each correspondence $p_i \leftrightarrow q_i$ to mark inliers and outliers. To find R and $w = [w_1, \dots, w_N]$, let us consider a bi-level optimization problem:

$$\begin{aligned} \min_w J(\hat{R}(w)) \\ \text{s. t. } \hat{R}(w) \in \arg \min_{R \in \text{SO}(3)} f(w, R) \\ \text{s. t. } h(R) = 0. \end{aligned} \quad (3)$$

Where $J(\hat{R}(w))$ is the upper-level loss:

$$J(\hat{R}(w)) = \arccos \left(\frac{\text{tr}(\hat{R}(w) R_{\text{true}}^T) - 1}{2} \right), \quad (4)$$

measuring the angle of the residual rotation [HZ03] and

$$f(P, Q, w, R) = f(w, R) = \frac{1}{N} \sum_{i=1}^N w_i \|R p_i - q_i\|_2^2, \quad (5)$$

is the low-level loss, and $h(R)$ is as the constraint

$$h(R) = R^T R - I = 0. \quad (6)$$

The low-level minimization problem (5) is known as Wahba's problem [Wah65].

Forward pass. During the forward pass, we solve the low-level optimization problem and compute the value of $J(\hat{R}(w))$. The low-level optimization problem is solved by the Kabsch algorithm [Kab76], which includes the computation of SVD of a matrix constructed from coordinates P and Q . The forward pass is the same for both DDN/IFT and SVD layers (Fig. 2).

Backward pass. During the backward pass, we compute all the derivatives with respect to the input and parameters via the chain rule and update the vector w . The backward pass is different for the SVD and the DDN/IFT layers, and we compare these three methods on the toy example below.

A toy example. Let us consider a toy example to compare the gradients computed explicitly and implicitly given a forward pass fixed for both methods. Given four random points in 3D space, we transformed the points with an identity transform R_{true} and corrupted one correspondence, for instance $p_1 \leftrightarrow q_1$. We want to find the rotation matrix R along with inlier/outlier mask w (see Fig. 3 (a)) given the correspondences $p_i \leftrightarrow q_i$.

Method	AUC@5°	AUC@10°	AUC@20°
∇-RANSAC [WPS+23]	0.41	0.45	0.5
MinBackProp DDN	0.4	0.44	0.49
MinBackProp IFT	0.41	0.45	0.5

Table 1: The average AUC scores for essential matrix estimation of ∇-RANSAC [WPS+23] and MinBackProp (ours) over 12 scenes of the PhotoTourism dataset [JMM+20], under different thresholds. Our method shows the same result as ∇-RANSAC.

We start with the uniform distribution of weights w and initialize them with $1/N$. The developments of the values of w for an inlier and the outlier during the optimization are shown in Fig. 3 (b). We optimize the weights w using the gradient descent with the learning rate of 0.1 for 30 iterations. The values of w for the outlier decrease to 0 and the values of w for inliers increase. One can see that the gradients calculated implicitly approximate the gradients of SVD computed in closed form well.

One can find one more toy example for the fundamental matrix estimation (8-point) in the Supplementary Material, section 10.

6.2 Training Outlier Detection in Epipolar Geometry Estimation

Our second experiment is incorporating our MinBackProp into the existing state-of-the-art epipolar geometry pipeline with real data. For that purpose, we consider a fully-differentiable framework ∇-RANSAC [WPS+23] that exploits the 5-point algorithm [Nis04] for essential matrix estimation and evaluates it on the RootSIFT [AZ12] features of the PhotoTourism dataset [JMM+20]. Figure 1 illustrates the architecture of ∇-RANSAC and the way how we integrate our MinBackProp in it. We adopt all the formulations and processing of [WPS+23], except for the backpropagation through the minimal problem solver (red rectangle in Fig. 1).

Given a pair of images I_1 and I_2 and a set of tentative correspondences $m = [q_i, \tilde{q}_i]$ computed from the RootSIFT features, the inlier indicator i_{gt} , and the ground truth essential matrix E_{gt} . The goal is to train a Neural Network (NN) [ZGZ+21] with parameters θ that predicts importance scores s during inference. In Fig. 1, D-RANSAC indicates differentiable Gumbel Softmax sampler, w denotes a minimal sample used as input into the Solver, and J is the loss function.

In accordance with eq. (1), we formulate our bilevel optimization problem as follows. The upper-level objective J is the symmetric epipolar distance over the inlier set i_{gt} of matches $m_i = [q_i, \tilde{q}_i]$, \hat{E} is the predicted model

Method	stable runs (%)	backward time (s)
∇-RANSAC [WPS+23]	20	34.4
∇-RANSAC*	30	34.5
MinBackProp DDN (ours)	100	37.6
MinBackProp IFT (ours)	100	3.6

Table 2: Stability metrics for 1K training of the baseline ∇-RANSAC [WPS+23] and our MinBackProp methods for the essential matrix estimation. ∇-RANSAC* denotes implementation of ∇-RANSAC [WPS+23] without dropping non-real values in the solver during the forward pass. Our MinBackProp is much more stable compared to the ∇-RANSAC and ∇-RANSAC* and MinBackProp IFT ten times faster than the baseline.

to be estimated, and $\tilde{l}_i = \hat{E} q_i$, $l_i = \hat{E}^T \tilde{q}_i$ are the epipolar lines

$$J(\hat{E}, m, i_{gt}) = \frac{1}{|i_{gt}|} \sum_{i \in i_{gt}} \left(\frac{1}{l_{i1}^2 + l_{i2}^2} + \frac{1}{\tilde{l}_{i1}^2 + \tilde{l}_{i2}^2} \right) (\tilde{q}_i^T \hat{E} q_i)^2, \quad (7)$$

while the low-level objective is the “invented” loss consisting of the algebraic error of the constraints (which is always evaluated to zero but provides non-singular Jacobians, see the discussion in section 5)

$$\begin{aligned} \hat{E}(w) \in \arg \min_{E \in \mathbb{R}^{3 \times 3}} \frac{1}{5} \sum_{i=1}^5 (\tilde{q}_i^T E q_i)^2 \\ \text{s. t. } 2EE^T E - \text{tr}(EE^T) E = 0, \\ \|E\|^2 = 1, \end{aligned} \quad (8)$$

where $w_i = [q_i, \tilde{q}_i]$ is a pair of matches from a minimal sample normalized by the intrinsic matrix pixel coordinates, and E is an essential matrix. The same equations eq. (8) are used to construct the polynomial system for the Implicit function theorem and compute the derivatives w.r.t. \hat{E} and $[q_i, \tilde{q}_i]$ and the Jacobian $\frac{\partial \hat{E}}{\partial w}$, respectively. See Supplementary Material, section 11 for more details.

As in the section 6.1, we retain the forward pass for the MinBackProp unchanged. However, we modify the computation of gradients during the backward pass. Instead of computing the gradients explicitly by the autograd as in ∇-RANSAC, we exploit the Deep Declarative Networks and the Implicit function theorem.

For training, as in ∇-RANSAC, we use correspondences of the St. Peter’s Square scene of the PhotoTourism dataset [JMM+20], consisting of 4950 image pairs, with split 3 : 1 into training and validation, while the rest 12 scenes remain for testing.

To show the effect of our method, we perform two types of experiments. First, we want to show that our method does not decrease the baseline quality. Secondly, we want to show that computing the gradients via the Implicit function theorem is more stable and does not cause runtime errors during the backward pass when autograd tries to invert a singular matrix.

For the first experiment, the testing protocol is as follows: the scores s are predicted by the trained NN, and the model E is estimated by MAGSAC++ [BNIM20]. Table 1 shows the Area Under the Recall curve (AUC), AUC scores are averaged over 12 testing scenes for different thresholds. The essential matrix E is decomposed into R and t using SVD and maximum error between R and R_{gt} and t and t_{gt} is reported. We train all three approaches for 10 epochs. One can see in Table 1 that our MinBackProp obtained the same scores as the ∇ -RANSAC, showing that it does not decrease the quality of the baseline.

The second experiment aims to demonstrate the stability of the proposed MinBackProp compared to the ∇ -RANSAC. For this purpose, we perform the following experiment. Both methods are trained for 10 epochs, and 10 random runs are performed. We measure the percentage of runs which finished 10 epochs training without any runtime errors during the backpropagation. The results are reported in Table 2. Our method is 100% stable compared to the baseline, which is 30% stable, and it is also $\times 10$ faster. ∇ -RANSAC* denotes the implementation of the original ∇ -RANSAC without dropping non-real solutions during the forward pass of the solver since to backpropagate the gradients with the DDN, we need to find at least one solution to the optimization problem, so in case the solver can not find any real solution, we return the one from the previous iteration. We also measured the stability of the ∇ -RANSAC* to be sure that this minor change does not contribute to the stability of our method.

All the experiments were conducted on NVIDIA GeForce GTX 1080 Ti with CUDA 12.2 and Pytorch 1.12.1 with Adam optimizer [KB15], learning rate 10^{-4} and batch size 32.

7 CONCLUSION

We have presented a new practical approach to backpropagating through minimal problem solvers. Our MinBackProp allows backpropagation either using the Deep Declarative Networks machinery, which brings stability and easy use, or using the Implicit function theorem directly, which on top of stability, also speeds up significantly the backward pass. Through synthetic examples, we have shown the applicability of our method on a wide variety of tasks. Furthermore, we compared MinBackProp against the state-of-the-art relative pose estimation approach from [WPS⁺23] and have shown 100% stability compared to 70 – 80% failure rate of the autograd approach, while speeding up the computation 10 times. Our method opens up a promising direction for efficient backpropagation through hard minimal problems.

See Supplementary Material for additional technical details. The code is available at <https://github.com/disungatullina/MinBackProp>.

ACKNOWLEDGMENTS

This work has been supported by the EU H2020 No. 871245 SPRING project.

8 REFERENCES

- [AK17] Brandon Amos and J. Zico Kolter. Optnet: Differentiable optimization as a layer in neural networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 136–145. PMLR, 2017. 3
- [AZ12] Relja Arandjelović and Andrew Zisserman. Three things everyone should know to improve object retrieval. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2911–2918, 2012. 2, 7
- [BKN⁺17] Eric Brachmann, Alexander Krull, Sebastian Nowozin, Jamie Shotton, Frank Michel, Stefan Gumhold, and Carsten Rother. Dsac – differentiable ransac for camera localization. pages 2492–2500, 07 2017. 1, 2
- [BNIM20] Daniel Barath, Jana Noskova, Maksym Ivashchkin, and Jiri Matas. MAGSAC++, a fast, reliable and accurate robust estimator. In *Conference on Computer Vision and Pattern Recognition, 2020*. 8
- [BR19] Eric Brachmann and Carsten Rother. Neural-guided ransac: Learning where to sample model hypotheses. pages 4321–4330, 10 2019. 1
- [BSF94] Yoshua Bengio, Patrice Y. Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Trans. Neural Networks*, 5(2):157–166, 1994. 1
- [CLO98] David Cox, John Little, and Donald O’Shea. *Using Algebraic Geometry*. Springer, 1998. 1
- [CLO15] David A. Cox, John Little, and Donald O’Shea. *Ideals, Varieties, and Algorithms: An Introduction to Computational Algebraic Geometry and Commutative Algebra*. Springer, 2015. 2, 5

- [CPC⁺20] Bo Chen, Álvaro Parra, Jiewei Cao, Nan Li, and Tat-Jun Chin. End-to-end learnable geometric vision by backpropagating pnp optimization. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 8097–8106. Computer Vision Foundation / IEEE, 2020. 3
- [CSC] Ltd. Cybernet Systems Co. Maple. <http://www.maplesoft.com/products/maple/>. 4
- [CWW⁺22] Hansheng Chen, Pichao Wang, Fan Wang, Wei Tian, Lu Xiong, and Hao Li. Epropnp: Generalized end-to-end probabilistic perspective-n-points for monocular object pose estimation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pages 2771–2780. IEEE, 2022. 3
- [DYH⁺18] Zheng Dang, Kwang Moo Yi, Yinlin Hu, Fei Wang, Pascal Fua, and Mathieu Salzmann. Eigendecomposition-free training of deep networks with zero eigenvalue-based losses. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, editors, *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part V*, volume 11209 of *Lecture Notes in Computer Science*, pages 792–807. Springer, 2018. 1, 2, 3
- [EM05] Mohamed Elkadi and Bernard Mourrain. *Symbolic-numeric methods for solving polynomial equations and applications*, pages 125–168. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005. 2
- [FB81] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, 1981. 4
- [GHC22] Stephen Gould, Richard Hartley, and Dylan Campbell. Deep declarative networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(8):3988–4004, 2022. 1, 2, 3, 4, 5
- [Har95] R. I. Hartley. In defence of the 8-point algorithm. In *Proceedings of the Fifth International Conference on Computer Vision, ICCV '95*, page 1064, USA, 1995. IEEE Computer Society. 2
- [HL11] AMIR HASHEMI and DANIEL LAZARD. Sharper complexity bounds for zero-dimensional groebner bases and polynomial system solving. *International Journal of Algebra and Computation*, 21(05):703–713, 2011. 2
- [HZ03] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge, 2nd edition, 2003. 6
- [IVS15] Catalin Ionescu, Orestis Vantzos, and Cristian Sminchisescu. Matrix backpropagation for deep networks with structured layers. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 2965–2973, 2015. 1, 2, 3, 4, 5
- [JMM⁺20] Yuhe Jin, Dmytro Mishkin, Anastasiia Mishchuk, Jiri Matas, Pascal Fua, Kwang Moo Yi, and Eduard Trulls. Image matching across wide baselines: From paper to practice. *International Journal of Computer Vision*, 2020. 7, 3
- [Kab76] W. Kabsch. A solution for the best rotation to relate two sets of vectors. *Acta Crystallographica Section A*, 32(5):922–923, Sep 1976. 5, 6
- [KB15] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. 8
- [KBP08] Zuzana Kukelova, Martin Bujnak, and Tomas Pajdla. Automatic generator of minimal problem solvers. In *European Conference on Computer Vision (ECCV)*, 2008. 1, 2
- [LKZ17] Viktor Larsson, Zuzana Kukelova, and Yinqiang Zheng. Making minimal solvers for absolute pose estimation compact and robust. In *International Conference on Computer Vision (ICCV)*, 2017. 2
- [LOÅ⁺18] Viktor Larsson, Magnus Oskarsson, Kalle Åström, Alge Wallis, Zuzana Kukelova, and Tomás Pajdla. Beyond grobner bases: Basis selection for minimal solvers. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 3945–3954, 2018. 1, 2
- [Mey01] Carl D. Meyer. *Matrix Analysis and Applied Linear Algebra*. SIAM: Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2001. 3, 1
- [MVP22] Evgeniy Martynushev, Jana Vráblíková,

- and Tomáš Pajdla. Optimizing elimination templates by greedy parameter search. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pages 15733–15743. IEEE, 2022. 1, 2
- [Nis04] David Nistér. An efficient solution to the five-point relative pose problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26:756–770, 2004. 1, 2, 3, 7
- [RK18] Rene Ranftl and Vladlen Koltun. Deep fundamental matrix estimation. In *The European Conference on Computer Vision (ECCV)*, 2018. 1, 2
- [Rud76] Walter Rudin. *Principles of Mathematical Analysis*. McGraw-Hill, 1976. 3, 1
- [SNKS05] Henrik Stewénus, David Nistér, Fredrik Kahl, and Frederik Schaffalitzky. A minimal solution for relative pose with unknown focal length. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2005), 20-26 June 2005, San Diego, CA, USA*, pages 789–794. IEEE Computer Society, 2005. 1
- [SR13] I.R. Shafarevich and M. Reid. *Basic Algebraic Geometry 1: Varieties in Projective Space*. SpringerLink : Bücher. Springer Berlin Heidelberg, 2013. 1
- [Wah65] Grace Wahba. A least squares estimate of satellite attitude. *SIAM Review*, 7(3):409–409, 1965. 6
- [WPS⁺23] Tong Wei, Yash Patel, Alexander Shekhovtsov, Jiri Matas, and Daniel Barath. Generalized differentiable ransac. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 17649–17660, October 2023. 1, 2, 3, 5, 6, 7, 8
- [ZGM⁺20] Qianggong Zhang, Yanyang Gu, Mateusz Michalkiewicz, Mahsa Baktashmotlagh, and Anders P. Eriksson. Implicitly defined layers in neural networks. *CoRR*, abs/2003.01822, 2020. 3
- [ZGZ⁺21] Chen Zhao, Yixiao Ge, Feng Zhu, Rui Zhao, Hongsheng Li, and Mathieu Salzmann. Progressive correspondence pruning by consensus learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision.*, 2021. 1, 7

Fish Motion Estimation Using ML-based Relative Depth Estimation and Multi-Object Tracking

Lintao Fang
Fraunhofer IGD-R
Germany 18059, Rostock
fang.lintao@igd-r.fraunhofer.de

Mohamad Albadawi
mohamedbadawi9@gmail.com

Tim Dolereit
Fraunhofer IGD-R
Germany 18059, Rostock
tim.dolereit@igd-r.fraunhofer.de

Arjan Kuijper
Fraunhofer IGD
Germany 64283, Darmstadt
arjan.kuijper@igd.fraunhofer.de

Matthias Vahl
Fraunhofer IGD-R
Germany 18059, Rostock
matthias.vahl@igd-r.fraunhofer.de

ABSTRACT

Fish motion is a very important indicator of various health conditions of fish swarms in the fish farming industry. Many researchers have successfully analyzed fish motion information with the help of special sensors or computer vision, but their research results were either limited to few robotic fishes for ground-truth reasons or restricted to 2D space. Therefore, there is still a lack of methods that can accurately estimate the motion of a real fish swarm in 3D space. Here we present our Fish Motion Estimation (FME) algorithm that uses multi-object tracking, monocular depth estimation, and our novel post-processing approach to estimate fish motion in the world coordinate system. Our results show that the estimated fish motion approximates the ground truth very well and the achieved accuracy of 81.0% is sufficient for the use case of fish monitoring in fish farms.

Keywords

fish activity index, multi-object tracking, absolute depth map reconstruction, post-processing approach, motion estimation, fish swarm

1 INTRODUCTION

The level of fish activity serves as an important indicator in fish farming, providing biologists with insights into the condition of a fish swarm, such as hunger or sickness. For example, reduced activity is often observed in hungry fish, leading to a noticeable decline in their overall speed. Summarizing the motion characteristics of a fish swarm is one of many approaches to reflect their entire activity level. Some researchers in recent years have used computer vision [HZL⁺20, WML⁺21] or Doppler-based technique [HFPA20, HFPA19, HFU⁺22] to estimate real fish speed, but they cannot evaluate the error of their method because of the lack of real fish speed as the reference. Other researchers [WWX15, WLZ⁺16] found robotic fishes with special sensors could be an alternative because these sensors can provide the

ground truth as a reference, but their experimental results are limited to very few fishes rather than fish swarms. Overall, these attempts consistently fail to reflect the true motion information of fish swarms in 3D space, leading to low persuasiveness and reliability.

In our work, we address the aforementioned limitations in two steps. First, it is technically difficult to obtain the real speed of a fish swarm in the real world, so we use the Unity game engine [JBT⁺18] to simulate different fish swarm scenes in the real world. By undertaking this approach, we can not only replicate real-world fish swarm scenarios but also utilize Unity to generate accurate fish swarm motion data and the ground truth. Second, we propose a novel algorithm as shown in Fig. 1 to robustly estimate fish motion in a simple, reproducible, and inexpensive manner. This algorithm consists of three main components, a multi-object tracking module, a monocular depth (mono-depth) estimation module, and a post-processing module. In our experiments, our unique post-processing method proved to be effective in overcoming the significant challenges caused by complex fish swimming motion in 3D space. The estimated average speed of the fish swarm is located in the world coordinate system, so our result can reflect the real state of fish motion. We use this average speed of the fish swarm as an index to represent the overall activ-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

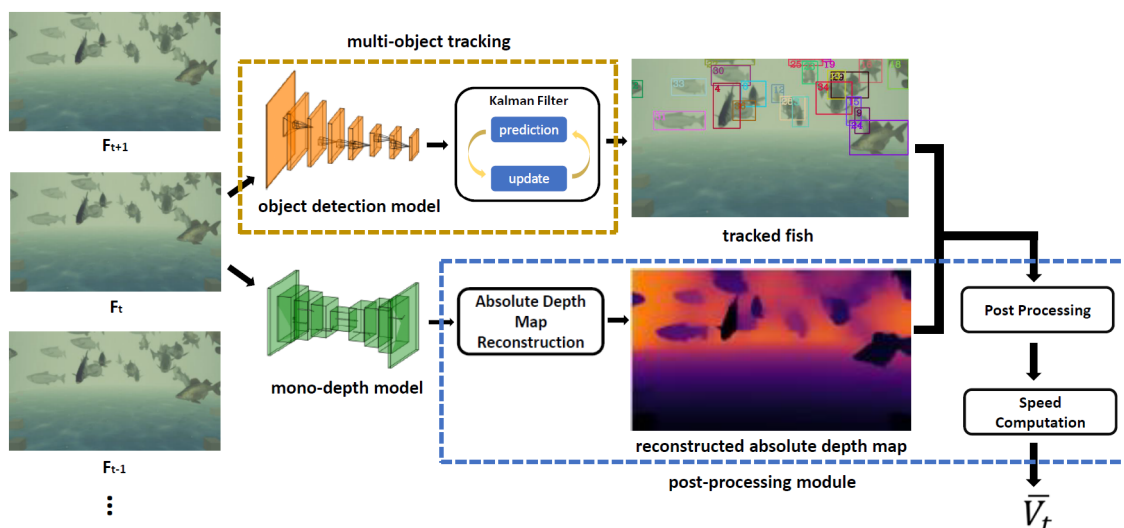


Figure 1: The overview of our Fish Motion Estimation (FME) algorithm. In a given frame F_t , we employ an object detector and Kalman Filter [Kal60] for fish tracking, assigning a unique ID to each fish. Simultaneously, a mono-depth model processes this image F_t and estimates the relative inverse depth map. From this, we reconstruct the absolute depth map. Afterward, we utilize our post-processing approach to process the tracking result and absolute depth map, and further we use this processing result to compute the fish swarm's average speed \bar{V}_t in the world coordinate system.

ity level of the fish swarm. Although we study our Fish Motion Estimation (FME) algorithm in a synthetic environment, we believe that our approach can be easily and effectively transferred to the real world. Therefore, all experimental steps in the synthetic environment remain strictly consistent with those in the real world.

We structure this paper as follows. In Section 2, we summarize the recent approaches for fish motion estimation. In Section 3, we describe our approach to absolute depth map reconstruction, the post-processing approach, and speed computation in detail. In Section 4, we show the detailed experiment results in multi-object tracking, mono-depth estimation, and fish motion estimation. In Section 5, we summarize our work and present the future plan to improve our fish motion estimation algorithm.

Our main contributions include:

- Introduction of a new algorithm that accurately estimates how fishes move in 3D space.
- Creation of a novel post-processing method to handle the challenges posed by the way fishes swim, making our approach stand out.
- A step forward by looking at how fishes move not just in 2D but also in 3D space, offering a more complete picture.

2 RELATED WORK

Fish studies in the early days relied on video systems to analyze fish swimming movements from video recordings, such as FICASS [PSW⁺97]. Later on, some

researchers used camera-based computer-controlled devices to perform real-time processing of fish kinematic information, including swimming acceleration and velocity [WZ07, CXG⁺09]. However, their research results are only based on few fishes. Other researchers use the Doppler principle-based technique to measure the motion information of fish swarms in sea cages [HFPA20, HFPA19, HFU⁺22]. Unlike the Doppler-based technique, some researchers used CNN-based methods to continuously track multiple fishes and further use the tracking results to estimate the fish swimming speed [LXH⁺21, BPPLAM23]. However, their experimental findings are less reliable as they cannot compare their results with ground truth, and their works are limited in 2D space. Zhang et al. [ZZH⁺23] utilized a binocular camera to project the 2D tracking space into 3D space and compare their estimated fish swimming speed with their ground truth in that space, but their ground truth is derived from pixel coordinates rather than from the real world.

To enhance the reliability of experimental results, subsequent researchers replaced the real fish with the controllable robotic fish because they can obtain the real swimming speed of the robotic fish as the ground truth. However, these robot fish-based methods require additional assistance, such as the optimal information fusion decentralized Kalman filter algorithm [WWX15], pressure sensor [WLZ⁺16, ZZX14], and the deep reinforcement learning controller [DSAR24, DNSR23]. Nevertheless, their experimental subjects only involve very few robotic fishes rather than fish swarms.

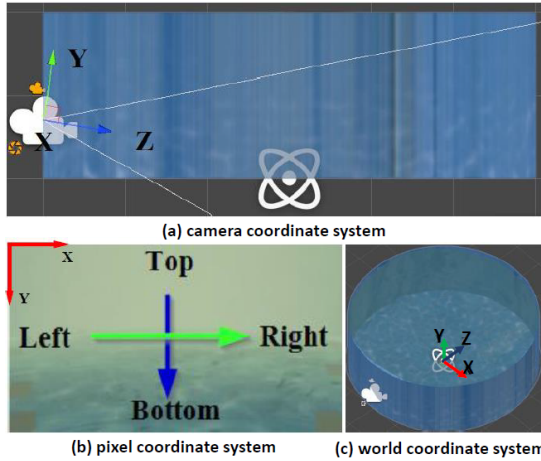


Figure 2: Three types of coordinate systems. (b) The arrows represent the fish's swimming directions.

To overcome these dilemmas, we use 3D fish models and the Unity game engine to simulate real-world fish swarm scenes. Under these settings, we can obtain real fish speed in 3D space as the ground truth. Since our synthetic fish scenes are in the world coordinate system, our work extends fish motion estimation from 2D to 3D space.

3 METHOD

3.1 Absolute Depth Map Reconstruction

To estimate the fish motion information in World Coordinate System (WCS) shown in Fig. 2 (c), we first need to acquire fish position information in WCS. As shown in Fig. 1, we use MiDaS Hybrid [RLH⁺22] as the mono-depth model to estimate relative inverse depth R_{inv} , and use an object detector to estimate fish position in the Pixel Coordinate System (PCS) shown in Fig. 2 (b). The depth R_{inv} predicted by MiDaS Hybrid is scaled and shifted to a unit scale $s = 1$ and zero translation $t = 0$, so it loses the absolute depth information of the objects in the scene and only presents the relative positional relationship between objects. However, we need the absolute depth information to convert fish position information from PCS to that in WCS. To solve this problem, we use an optimized transform algorithm as shown in Equation (1) to convert R_{inv} to absolute depth D_{abs} .

$$s^*, t^* \leftarrow \operatorname{argmin} L_2(\mathbf{D}_{dv}, \mathbf{R}_{dv}) \quad (1)$$

D_{dv} is the known absolute depth values in the scene, while R_{dv} is the relative inverse depth values in the depth map R_{inv} . This algorithm requires at least two known absolute depth values D_{dv} and two relative inverse depth values R_{dv} to solve s^* and t^* .

We place ten reference objects in the fish pond to provide the known absolute depth values from the fish

scene. We take the center point on the front surface from each reference object as the reference point p_i . We locate the coordinate of each reference point $p_i = (w_x^i, w_y^i, w_z^i)$ in WCS. Since Unity game engine can offer camera extrinsic parameter \mathbf{E} and intrinsic parameter \mathbf{I} , we can convert these reference points in WCS to those in the Camera Coordinate System (CCS) shown in Fig. 2 (a). Afterward, we collect the absolute depth values D_{dv} from these reference points. We continue to convert these reference points in CCS to those in PCS so that we use the pixel coordinates of these reference points to extract R_{dv} from the relative inverse depth map R_{inv} . In the end, we use the known absolute depth values D_{dv} and relative inverse depth values R_{dv} to solve the optimal solution s^* and t^* . With s^* and t^* , we can reconstruct the absolute depth D_{abs} by using following the Equation (2).

$$\mathbf{D}_{abs} \leftarrow \mathbf{R}_{inv} \times s^* + t^* \quad (2)$$

3.2 Post Processing

Fish swimming movements are random, unpredictable, and easily influenced by their states as well as the surroundings [RPM22, HZL⁺20]. These special features eventually cause two problems, which are the fluctuation in the position of the BBox center point and the abrupt jump in the depth value at that point. These two problems will cause large errors in speed computation because we estimate fish speed based on the BBox center points.

We assume that the location of a fish at the current frame f is based on its location at previous frames. Therefore, our idea is to summarize the past location information of a fish and use it to update its location information in the current frame f . We use the Exponentially Weighted Moving Average (EWMA) to achieve it.

3.2.1 Fluctuation problem of BBox center point

Due to the various fish swimming postures or fish-tail swinging movements, these factors could cause the BBox dimensions to change abnormally between successive sequences. For example, the width of BBox may suddenly become larger or smaller in two successive frames $t - 1$ and t when a fish is swimming. This leads to the fluctuation problem on the BBox center point, and it can negatively affect the estimation results of fish speed. To overcome this problem, we create two detectors as described in Equation (3) to monitor abnormal changes in the BBox dimensions. We use the average variation $\mathbf{Avg}(\cdot)$ of the BBox dimensions in the past 30 frames as the reference to determine if the current change in the BBox dimension is abnormal.

$$\begin{aligned} \text{detector}_w &= \mathbf{abs}(w_t - w_{t-1}) > \mathbf{Avg}(\mathbf{D}_w, \text{Len}(\mathbf{D}_w)) \\ \text{detector}_h &= \mathbf{abs}(h_t - h_{t-1}) > \mathbf{Avg}(\mathbf{D}_h, \text{Len}(\mathbf{D}_h)). \end{aligned} \quad (3)$$

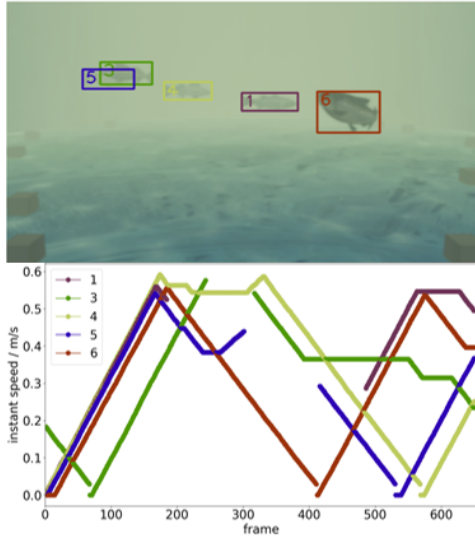


Figure 3: Annotation. The real-time speed of each fish in WCS.

$\mathbf{D}_w = \{(w_i - w_{i-1})\}_{i=t-28}^{t-1}$ is the list containing the difference between the BBox width w in every two adjacent frames, while the $\mathbf{D}_h = \{(h_i - h_{i-1})\}_{i=t-28}^{t-1}$ represents the difference in BBox height h . If we detect an abnormal change in the BBox dimensions at frame t when a fish swims, we collect its BBox dimensions in the previous 5 frames. Afterward, we use EWMA to update the BBox dimensions at the current frame t and further compute the BBox center point $c_t = (c_t^x, c_t^y)$.

3.2.2 Jump problem on the depth value

Abrupt changes in depth value at the BBox center points occur mainly when a fish is partially occluded by others during fish interactions. This occlusion problem causes the fish BBox center point to jump onto other fishes in the next frame making the depth value inaccurate. To tackle this problem, we first create a detector as shown in Equation (4) to monitor each fish and detect if its depth value at the BBox center point jumps. We assume that the change in depth value for each fish in successive frames is smooth when a fish swims alone. $\mathbf{abs}(d_t - d_{t-1})$ is the absolute depth difference in successive frames. We experimentally found that $\delta = 0.6$ m delivers good results. Together with the BBox center point $c_t = (c_t^x, c_t^y)$ at current frame t , the absolute depth map \mathbf{D}_{abs}^t and the set of depth values in past 30 frames $\mathbf{Dep} = \{d_i^*\}_{i=t-29}^{t-1}$, we use EWMA to update the depth value at the current frame t . Eventually, we obtain the updated absolute depth value d_t^* .

$$detector_{dep} = \mathbf{abs}(d_t - d_{t-1}) \geq \delta. \quad (4)$$

3.3 Speed computation

In order to estimate the average speed of the fish swarm, we start with a single fish. We use the BBox center

Hyperparameter	Symbol	Search Space
Kernel Size	k	$[3 \times 3, 5 \times 5, 7 \times 7]$
Detection Thres	T_d	$[30\%, 40\%, 50\%, 60\%, 70\%]$
Tracking Thres	T_t	$[30\%, 40\%, 50\%, 60\%, 70\%]$
Fusion Thres	T_f	$[30\%, 40\%, 50\%, 60\%, 70\%]$
Untracked Thres	T_{unt}	$[30\%, 40\%, 50\%, 60\%, 70\%]$
Unconfirmed Thres	T_{unc}	$[30\%, 40\%, 50\%, 60\%, 70\%]$
IoU Thres	T_{iou}	$[30\%, 40\%, 50\%, 60\%, 70\%]$

Table 1: Grid search space for hyper-parameters

points $\mathbf{C} = \{(c_i^x, c_i^y)\}_{i=1}^t$ and the absolute depth values $\mathbf{Dep} = \{d_i^*\}_{i=1}^t$ to firstly project BBox center points \mathbf{C} in PCS to WCS, and then we compute fish speed $\mathbf{V} = \{v_i\}_{i=1}^t$. The time interval is 1/30 seconds, so $fps = 30$. Next, we still use the above computation steps to calculate the speed for each fish in a fish swarm. With our multi-object tracker, we can identify each fish in the next frames. For each frame, we use Equation (5) to compute the average speed \bar{v}_t of all tracked fishes. Here N is the total number of fishes at the frame t . After traversing all frames T , we obtain the average speed $\bar{\mathbf{V}} = \{\bar{v}_t\}_{t=1}^T$ as the final result.

$$\bar{v}_t = \frac{1}{N} \sum_{o=1}^N v_t^o, \quad (5)$$

4 EXPERIMENTS

4.1 Synthetic Fish Dataset

There are some fish datasets available for different computer vision tasks, such as [UKT20] and [SLK⁺20], but none of them provides fish motion information. Therefore, we use Unity game engine [JBT⁺18] to design our synthetic fish dataset and automatically generate annotation as shown in Fig. 3. This synthetic dataset contains 163 videos which have 89901 images in total. We split this dataset into training dataset with 99 videos, validation dataset with 32 videos, and testing dataset with 32 videos. The frame per second in this dataset is 30 fps.

Unity captures the ground truth speeds based on the gravity center of the 3D fish model. It is very difficult to visually determine the position of the gravity center of a swimming fish, so we choose to approximate the ground truth speed by the speed of the BBox center point. This is expected to introduce inherent errors, primarily stemming from two distinct factors. Firstly, the likelihood of the gravity center aligning precisely with the center point of the BBox diminishes when a fish is

Trackers	k	T_d	T_t	T_f	T_{unt}	T_{unc}	T_{iou}
T_{hrnet}	3×3	30%	70%	70%	70%	30%	—
T_{retina}	—	30%	50%	—	70%	30%	50%
$T_{fasterrcnn}$	—	50%	50%	—	70%	40%	50%

Table 2: Grid search result for multi-object trackers

Trackers	IDF1 \uparrow	MOTA \uparrow	MOTP \uparrow	fps \uparrow
T_{hrnet}	80.1%	81.1%	87.4%	16.7
T_{retina}	79.4%	80.5%	86.7%	20.39
$T_{fasterrcnn}$	82.4%	81.8%	87.4%	21.15

Table 3: Multi-object tracking results

navigating through three-dimensional space. Secondly, the depth value at the BBox center point consistently corresponds to the outer surface of the fish, whereas the gravity center resides within the fish. Consequently, the depth value at the BBox center point is almost always smaller than the value at the gravity center.

4.2 Multi-Object Tracking

We adopt the tracking-by-detection approach to achieve multiple fish tracking in our synthetic fish swarm scenes. We prepare three detector-based multi-object trackers, including HRNet-based tracker T_{hrnet} , RetinaNet-based tracker T_{retina} , and FasterRCNN-based tracker $T_{fasterrcnn}$. We use the same training strategy to train these object detectors. The batch size is set to 10. We optimize these models with the Adam optimizer [KB14] for 40 epochs. The learning rate γ is $1.25e^{-4}$, and it is decayed by half for every 10 epochs. After the training stage, we use each well-trained object detector and Kalman Filter [Kal60] to compose a multi-object tracker. Since Kalman Filter involves some hyperparameters, we create a search space for each tracking hyperparameter shown in Table 1 and use the grid search approach to find the optimal tracking parameter for each tracker. We show the result in Table 2.

We use the evaluation metrics that are commonly adapted in the community to measure tracking performance. The tracking result is shown in Table 3. T_{hrnet} has comparable overall performance with T_{retina} , but $T_{fasterrcnn}$ outperforms other trackers in tracking performance and inference speed. Therefore, we will

Reconstruction Algorithms		$RMSE/m \downarrow$	$\&1.25 \uparrow$	$\&1.25^2 \uparrow$
mono-depth models	ML			
MiDaS Hybrid	LR	0.625	0.73	0.94
	PR	27.36	0.72	0.925
	DT	0.66	0.695	0.915
	KNN	0.655	0.7	0.92
MiDaS Large	LR	0.93	0.58	0.825
	PR	42.915	0.605	0.84
	DT	0.79	0.575	0.855
Boosting MiDaS	KNN	0.785	0.59	0.85
	LR	0.725	0.675	0.9
	PR	118.3	0.66	0.885
	DT	0.71	0.665	0.9
TCMiDaS	KNN	0.7	0.67	0.9
	LR	1.675	0.355	0.535
	PR	2934.985	0.44	0.62
	DT	1.485	0.38	0.6
	KNN	1.415	0.375	0.61

Table 4: Comparison of different absolute depth map reconstruction algorithms

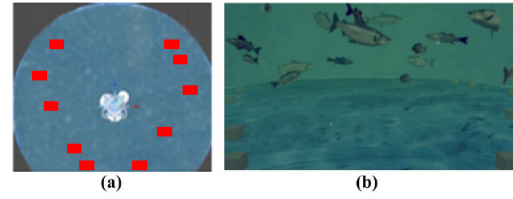


Figure 4: Random layout of reference objects. (a) The red boxes are the positions of reference objects. (b) The reference objects from the front view.

use the FasterRCNN-based tracker to achieve multiple fish tracking in the following experiments.

4.3 Absolute Depth Map Reconstruction

Since it is technically difficult to generate precise depth maps as ground truth in underwater environments, we cannot train any mono-depth models in the real world. Similarly, we do not train any mono-depth models on our synthetic datasets, and we use an already well-trained model MiDaS [RLH⁺22] with high generalization ability.

4.3.1 Transformation Algorithms

We choose four open-source mono-depth variants based on MiDaS as our experimental candidates, and they are MiDaS Hybrid [RLH⁺22], MiDaS Large [RLH⁺22], BoostingMiDaS [MDM⁺21] and TCMiDaS [LLZ⁺21]. We use four Machine Learning (ML) algorithms to reconstruct absolute depth maps from the relative inverse depth maps, including Linear Regression (LR), second-order Polynomial Regression (PR), Decision Tree (DT), and KNN. There are two primary reasons for this operation. Firstly, these models lack the capability to directly predict absolute depth maps. Secondly, [RLH⁺22, MDM⁺21, LLZ⁺21] do not put forth any default transformation algorithms

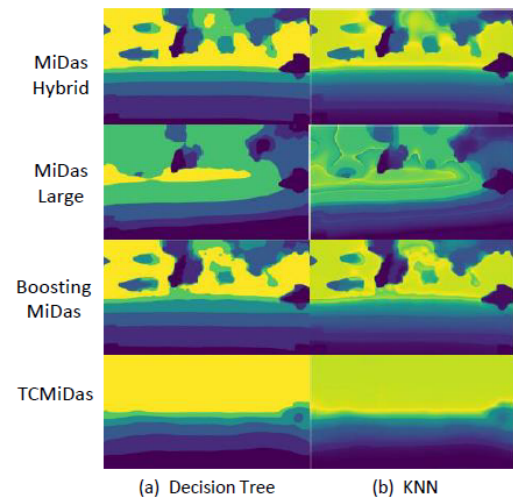


Figure 5: Examples of reconstructed absolute depth maps

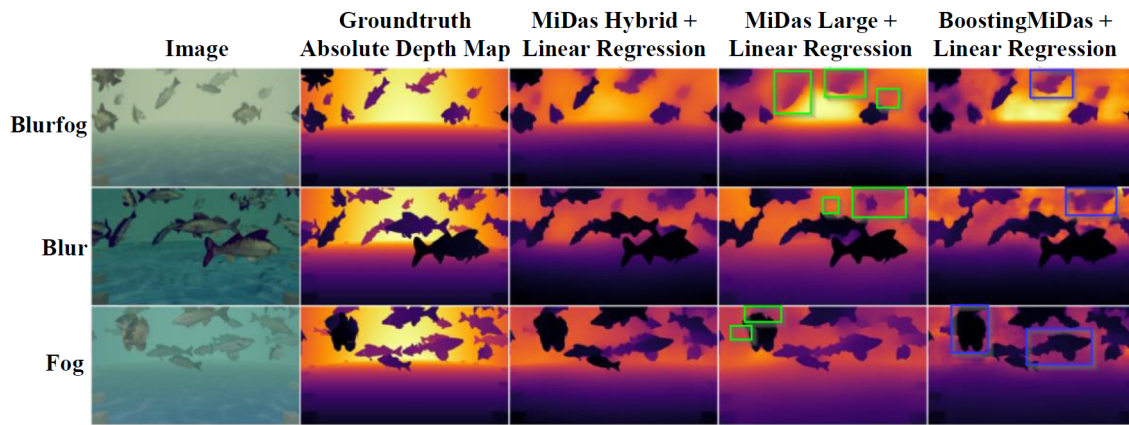


Figure 6: Examples of the absolute depth maps reconstructed by linear regression in different fish scenes. **The green rectangle** marks the ignored fish. **The blue rectangle** marks fish with the same depth values in a small fish swarm.

that can effectively convert a relative inverse depth map into an absolute depth map. We use the Root Mean Squared Error (RMSE) to measure the deviation between the estimated depth values and the ground truth. We also use $\times 1.25$ and $\div 1.25^2$ to evaluate their precision. Since absolute depth map reconstruction requires some known depth values from the scene, we tried different layouts of the reference objects and came up with the one that performed best, as shown in Fig. 4 (a).

A depth estimation algorithm is deemed effective if it achieves continuous values in the predicted depth map without introducing layering effects, while also maintaining good metrics for both the background and foreground objects. We combine four mono-depth variants and four ML algorithms and evaluate their absolute depth map reconstruction results. The result is shown in Table 4. We find that polynomial regression ends up with a very large error in RMSE. This primarily stems from the x^2 term in polynomial regression, which tends to magnify large depth values during the depth map reconstruction process. Thus, it is unsafe to use polynomial regression to conduct depth map transformation. As shown in Fig. 5, the decision tree and KNN tend to reconstruct absolute depth maps with obvious layering effects, so it is not acceptable in our case. Additionally, the reconstructed depth maps based on TCMiDaS lose most of the depth information. It is probably because the training strategy in Li et al. [LLZ⁺21] does not maintain the original generalization capability of MiDas. Thus, we will not consider this model in our work. Unlike other ML algorithms, linear regression maintains the most balanced performance in all metrics and with all Midas variants. Therefore, linear regression is the most suitable transformation algorithm.

We further compare the absolute depth maps reconstructed by linear regression in different fish scenes. As shown in Fig. 6, MiDas Large tends to miss some fishes

that are relatively far away in the scene when estimating depth maps. BoostingMiDaS can provide depth information for all fishes, but it tends to predict the same depth values for neighboring fishes forming a group in the image. The reason may be that the fish's appearance and the boundaries between these fishes become blurred in these scenes, so BoostingMiDaS incorrectly recognizes their contextual cues as the same. Such a situation makes it difficult for BoostingMidas to distinguish multiple fishes with blurred boundaries. MiDas Hybrid and linear regression do not have the problems mentioned above, so we will use MiDas Hybrid and linear regression to reconstruct absolute depth maps for our fish scenes.

4.3.2 Layout of Reference Objects

The spatial layout of reference objects is another important factor in absolute depth map reconstruction. As shown in Fig. 7, we prepare seven different spatial layouts and restrict the size of all reference objects to $0.1 \times 0.1 \times 0.1m^3$. We assume that the reference objects in this size are too small to affect fish swimming movements.

The result is shown in Table 5. We find that the increasing number of reference objects from layouts A to

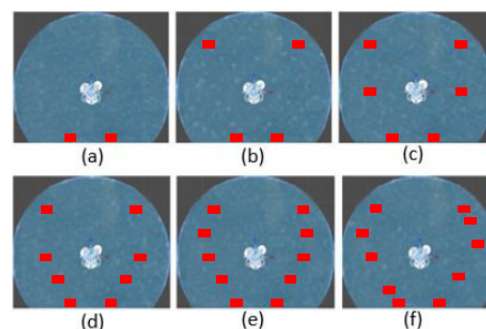


Figure 7: Different layouts of reference objects. The red boxes are the positions of reference objects

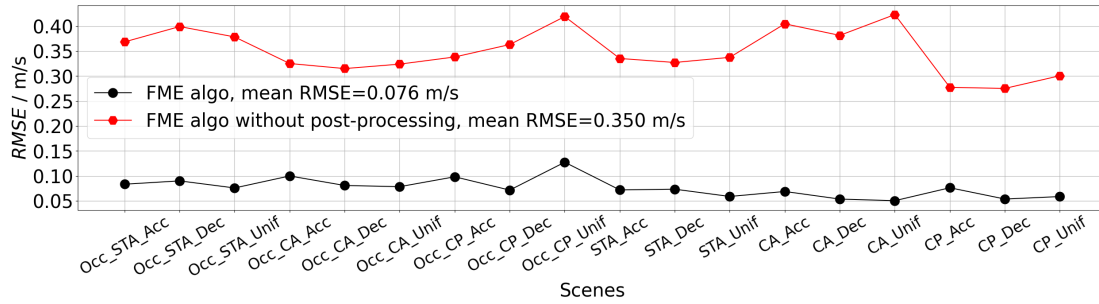


Figure 8: Post-processing performance. The **Occ** means occlusion.

D improves the overall accuracy in absolute depth map reconstruction and decreases errors in reconstructed absolute depth values. The reason is that placing more reference objects in the depth direction of the fish pond can provide more depth information about the pond. However, as shown in Table 5, placing two more objects in layout E shown in Fig. 7 ends up with a very similar evaluation result to that of layout D. This result indicates that placing more objects is less helpful in improving the accuracy of absolute depth map reconstruction. To solve this problem, we created layout F by slightly shifting each reference object in different directions by a small distance. In Table 5, the overall evaluation result in layout F delivers better results than other layouts. Compared to layout E, layout F uses the same number of reference objects to obtain a more complete depth range in the fish pond. Therefore, we will arrange reference objects in our fish ponds according to layout F.

4.4 Post Processing

To investigate the effectiveness of our post-processing method against the fluctuation problem of BBox center point and the jump problem on depth value, we created simple scenes where a fish swims in pre-designed movements. These scenes include three types of motion and three types of swimming trajectories. The three types of motion include acceleration (Acc), deceleration (Dec), and uniform (Unif). The three types of trajectories include the Coordinate Axis direction (CA), the Coordinate Plane direction (CP), and Spatial Turn-Around movements (STA). In detail, the CA direction includes directions in $\vec{d} = (1, 0, 0)$, $\vec{d} = (0, 1, 0)$, and

$\vec{d} = (0, 0, 1)$, while the CP direction includes directions in $\vec{d} = (1, 1, 0)$, $\vec{d} = (1, 0, 1)$, and $\vec{d} = (0, 1, 1)$. Since a fish is either occluded or not occluded when swimming in a fish swarm, we divide these scenes into the occluded and not-occluded ones. In total, we have 18 different scenes. Also, we feed the ground truth BBox and depth map to our post-processing approach rather than the tracking result and reconstructed absolute depth maps as shown in Fig. 1.

As shown in Fig. 8, our FME algorithm can produce better results in occluded and non-occluded scenes than FME algorithm without the post-processing approach. Our post-processing approach reduces $\nabla RMSE$ by $0.274m/s$. This result implies that our post-processing approach effectively addresses the disturbance caused by fish swimming movement. Following, two examples are given to demonstrate the improvements in fish motion estimation.

Since fish swimming movements always cause the drift problem on BBox center point, we make a single fish swimming in the direction $\vec{d} = (1, 0, 0)$ as shown in Fig. 9 in order to investigate the influence of this problem on the fish motion. The fish swimming along this trajectory can always be at the same distance from the

Layout	RMSE/m ↓	&1.25 ↑	&1.25 ² ↑
A	1.84	0.1	0.525
B	0.65	0.68	0.89
C	0.59	0.735	0.93
D	0.565	0.755	0.95
E	0.57	0.75	0.955
F	0.55	0.77	0.95

Table 5: Absolute depth map reconstruction in different layouts

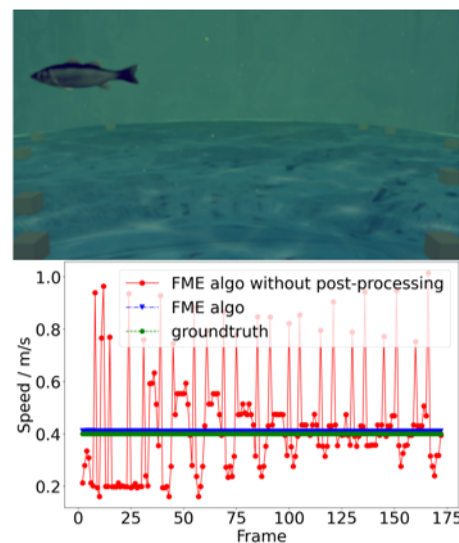


Figure 9: Post processing on BBox

camera. As shown in Fig. 9, the estimated result based on our post-processing approach remains exactly the same as the ground truth while the method without the post-processing generates a very fluctuating result. This comparison indicates that our post-processing approach can overcome the drift problem on BBox center points. When a fish is swimming behind another fish as shown in Fig. 10, it causes the depth value at the BBox center point to abruptly jump in two consecutive frames. However, our post-processing approach can overcome this problem and maintain smooth changes in depth values on successive frames.

4.5 Fish motion estimation

We have prepared two types of fish scenes, including the single-fish scene and the fish swarm. Our initial focus is on the single-fish scene. This choice allows us to fully isolate the effects of interaction between fishes and concentrate on assessing the influence of the spatial swimming movements of an individual fish on our FME algorithm. We compare our FME algorithms based on four different input sources that include Ground Truth BBox (GTBBox), Ground Truth depth (GTDepth), Predicted BBox (PredBBox), and Estimated absolute Depth map (EstDepth). We consider the result from the algorithm based on GTBBox>Depth as the benchmark.

We utilize RMSE to evaluate the deviation between the estimated speed and the ground truth speed. We also use distance correlation (DistCorr) to quantify the similarity of tendency between them. Unlike the Spearman correlation and the Pearson correlation, DistCorr can measure the nonlinear association between two variables with non-monotonicity, and it does not impose any restrictions on the distribution of these two variables.

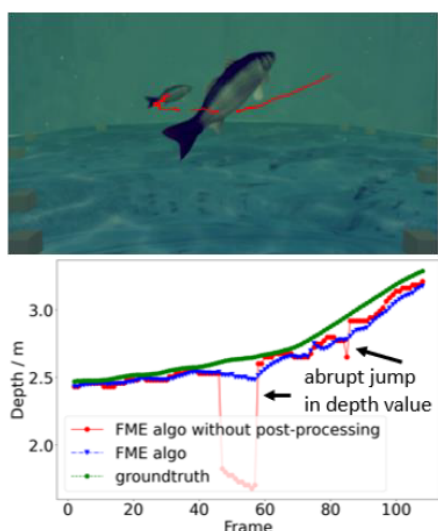


Figure 10: Post processing on depth. The red line on the top figure represents the fish swimming trajectory.

4.5.1 Single-Fish Scene

As shown in Table 6, the benchmark result ends up with an error of $RMSE = 0.05m/s$, while the correlation reaches up to $DistCorr = 92.0\%$. Since we only use ground truth depth maps and BBox to estimate fish speed, we think this error mainly comes from the inherent error as we discussed in Section 4.1 and our post-processing approach.

The estimation result in group C is very close to the benchmark, giving an error of $RMSE = 0.06m/s$ and the correlation of $DistCorr = 92.0\%$. The reason for this difference is that BBoxes predicted by our tracking algorithm deviate a bit from the ground truth BBox, so it leads to a localization error in the fish positions. However, its performance remains very close to the benchmark as shown in Fig. 11 (b). We notice that the result in Group B is worse than that in the benchmark. This is mainly because the reconstructed absolute depth map provides less accurate depth values for fish speed computation. In group D, the combination of our tracking and absolute depth reconstruction algorithms can generate nearly the same performance as that in group B. It implies that our absolute depth estimation algorithm is the main error source in single-fish motion estimation. However, the estimated speed shown in Fig. 11 (c) approximates the ground truth well enough. Thus, we believe that our algorithm can robustly estimate fish motion in single-fish scenes with an error of $RMSE = 0.08m/s$ and the correlation of $DistCorr = 89.0\%$.

4.5.2 Fish Swarm Scene

As shown in Table 7, we notice that the benchmark result has an error of $RMSE = 0.03m/s$, but the correlation between the estimated average speed and the ground truth is $DistCorr = 92.0\%$. The estimated result in Group D has an error of $RMSE = 0.06m/s$. By comparing with the benchmark, our tracking and absolute depth map reconstruction algorithms together lead to an increasing error in the estimated average speed by $\nabla RMSE = 0.03m/s$. Also, its correlation of $DistCorr = 81.0\%$ differs from the benchmark by $\nabla DistCorr = 11.0\%$. The reasons are two-fold. First, fish swarm scenes make a more difficult task for our tracker because of occlusion, which makes it more difficult to continuously and stably track a fish with the same tracking ID, and increases the chances of losing some tracked targets or ID switches while tracking

Group	Input Source	$RMSE(m/s) \downarrow$	$DistCorr \uparrow$
A	Benchmark	0.05	92.0%
B	GTBBox&EstDepth	0.08	90.0%
C	PredBBox>Depth	0.06	92.0%
D	PredBBox&EstDepth	0.08	89.0%

Table 6: Fish motion estimation in single-fish scenes

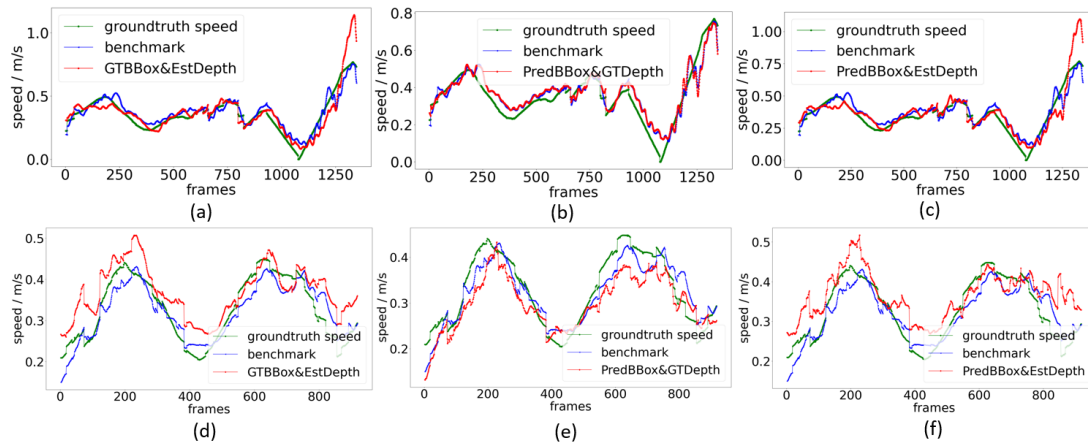


Figure 11: Fish motion estimation. (a)-(c): single fish scenes. (d)-(f): fish swarm scenes.

many fishes. This problem is highly dependent on the complexity of fish interaction in a fish swarm. The imperfection of our tracking algorithm on motion estimation becomes more obvious in fish swarm scenes. Second, fish swarm scenes increase the difficulty of reconstructing accurate absolute depth information for the fish. These two aspects eventually lead to the deviation in fish speed computation. However, the estimated speed curve shown in Fig. 11 (f) closely mirrors the trend of the ground truth curve. Thus, we believe that our tracking model and mono-depth estimation algorithm can adequately estimate fish motion in fish swarm scenes with $RMSE = 0.06m/s$ and $DistCorr = 81.0\%$. Our biologist partners believe that this error is acceptable as long as the estimated speed curve is close to the ground truth curve.

Additionally, we adopt a 95% confidence interval to measure the error range of the average speed in a fish swarm at a given frame f computed by our FME algorithm, and the range is $[-0.167m/s, 0.1m/s]$. This error range is measured under the frame rate of 30 fps.

5 CONCLUSION

In our research, we tackled challenges in fish motion studies by addressing the lack of ground truth information, and the complexity of fish swarm scenes. We created synthetic fish data and introduced a novel fish motion estimation algorithm, incorporating multi-object tracking, a mono-depth model, and an innovative post-processing approach for 3D motion estimation. Our research successfully extends fish studies from 2D to 3D space. Our experiments show that our post-processing

Group	Input Source	$RMSE(m/s) \downarrow$	$DistCorr \uparrow$
A	Benchmark	0.03	92.0%
B	GTBBox&EstDepth	0.05	88.0%
C	PredBBox>Depth	0.04	92.0%
D	PredBBox&EstDepth	0.06	81.0%

Table 7: Fish motion estimation in fish swarm scenes

method significantly reduces errors in fish speed computation, and our FME algorithm can estimate adequately accurate fish motion in different fish swarm scenes.

Our work demonstrates the feasibility of accurately computing fish motion using a multi-object tracker and a mono-depth model. Looking ahead, we aim to design a neural network model for direct fish speed prediction.

6 ACKNOWLEDGMENT

This project was funded by the Federal Ministry of Education and Research under the funding code 03ZU1107MB.

7 REFERENCES

- [BPPLAM23] Faezeh Behzadi Pour, Lorena Parra, Jaime Lloret, and Saman Abdanan Mehdizadeh. Measuring and evaluating the speed and the physical characteristics of fishes based on video processing. *Water*, 15(11):2138, 2023.
- [CXG⁺09] Jiujun Chen, Gang Xiao, Fei Gao, Hongbin Zhou, and Xiaofang Ying. Vision-based perceptive framework for fish motion. In *2009 International Conference on Information Engineering and Computer Science*, pages 1–4. IEEE, 2009.
- [DNSR23] Palmani Duraisamy, Manigandan Nagarajan Santhanakrishnan, and Amirtharajan Rengarajan. Design of deep reinforcement learning controller through data-assisted model for robotic fish speed tracking. *Journal of Bionic Engineering*, 20(3):953–966, 2023.
- [DSAR24] Palmani Duraisamy, Manigandan Nagarajan Santhanakrishnan, Rengarajan Amirtharajan, and Sudha Ramasamy. Real-time implementation of deep reinforcement learning controller for speed tracking of robotic fish through data-assisted modeling. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, 238(2):572–585, 2024.

- [HFPA19] Waseem Hassan, Martin Føre, Magnus Os-haug Pedersen, and Jo Arve Alfredsen. A novel doppler based speed measurement technique for individual free-ranging fish. In *2019 IEEE sensors*, pages 1–4. IEEE, 2019.
- [HFPA20] Waseem Hassan, Martin Føre, Magnus Os-haug Pedersen, and Jo Arve Alfredsen. A new method for measuring free-ranging fish swimming speed in commercial marine farms using doppler principle. *IEEE Sensors Journal*, 20(17):10220–10227, 2020.
- [HFU⁺22] Waseem Hassan, Martin Føre, Henning A Urke, John B Ulvund, Eskil Bendiksen, and Jo A Alfredsen. New concept for measuring swimming speed of free-ranging fish using acoustic telemetry and doppler analysis. *biosystems engineering*, 220:103–113, 2022.
- [HZL⁺20] Fangfang Han, Junchao Zhu, Bin Liu, Baofeng Zhang, and Fuhua Xie. Fish shoals behavior detection based on convolutional neural network and spatiotemporal information. *IEEE Access*, 8:126907–126926, 2020.
- [JBT⁺18] Arthur Juliani, Vincent-Pierre Berges, Ervin Teng, Andrew Cohen, Jonathan Harper, Chris Elion, Chris Goy, Yuan Gao, Hunter Henry, Marwan Mattar, et al. Unity: A general platform for intelligent agents. *arXiv preprint arXiv:1809.02627*, 2018.
- [Kal60] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. 1960.
- [KB14] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [LLZ⁺21] Siyuan Li, Yue Luo, Ye Zhu, Xun Zhao, Yu Li, and Ying Shan. Enforcing temporal consistency in video depth estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1145–1154, 2021.
- [LXH⁺21] Xianghui Li, Xin Xia, Zhuhua Hu, Bing Han, and Yaochi Zhao. Intelligent detection of underwater fish speed characteristics based on deep learning. In *2021 5th Asian Conference on Artificial Intelligence Technology (ACAIT)*, pages 182–189. IEEE, 2021.
- [MDM⁺21] S Mahdi H Miangoleh, Sebastian Dille, Long Mai, Sylvain Paris, and Yagiz Aksoy. Boosting monocular depth estimation models to high-resolution via content-adaptive multi-resolution merging. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9685–9694, 2021.
- [PSW⁺97] RJ Petrell, X Shi, RK Ward, A Naiberg, and CR Savage. Determining fish size and swimming speed in cages and tanks using simple video techniques. *Aquacultural Engineering*, 16(1-2):63–84, 1997.
- [RLH⁺22] René Ranftl, Katrin Lasinger, David Hafner, Konrad Schindler, and Vladlen Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(3), 2022.
- [RPM22] Stefan Reiser, Erick Cantu Perez, and Alexandra Meier. Automated detection of fish activity in recirculating aquaculture systems. 2022.
- [SLK⁺20] Alzayat Saleh, Issam H Laradji, Dmitry A Konovalov, Michael Bradley, David Vazquez, and Marcus Sheaves. A realistic fish-habitat dataset to evaluate algorithms for underwater visual analysis. *Scientific Reports*, 10(1):14671, 2020.
- [UKT20] Oguzhan Ulucan, Diclehan Karakaya, and Mehmet Turkan. A large-scale dataset for fish segmentation and classification. In *2020 Innovations in Intelligent Systems and Applications Conference (ASYU)*, pages 1–5, 2020.
- [WLZ⁺16] Wei Wang, Yuan Li, Xingxing Zhang, Chen Wang, Shiming Chen, and Guangming Xie. Speed evaluation of a freely swimming robotic fish with an artificial lateral line. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4737–4742. IEEE, 2016.
- [WML⁺21] Guangxu Wang, Akhter Muhammad, Chang Liu, Ling Du, and Daoliang Li. Automatic recognition of fish behavior with a fusion of rgb and optical flow data based on deep learning. *Animals*, 11(10):2774, 2021.
- [WWX15] Chengcai Wang, Wei Wang, and Guangming Xie. Speed estimation for robotic fish using onboard artificial lateral line and inertial measurement unit. In *2015 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 285–290. IEEE, 2015.
- [WZ07] GuanHao Wu and LiJiang Zeng. Video tracking method for three-dimensional measurement of a free-swimming fish. *Science in China Series G: Physics, Mechanics and Astronomy*, 50(6):779–786, 2007.
- [ZZH⁺23] Lanyue Zhang, Guilin Zhai, Bo Hu, Zhi Qiao, and Peizhen Zhang. Fish target detection and speed estimation method based on computer vision. In *2023 IEEE 6th International Conference on Electronic Information and Communication Technology (ICEICT)*, pages 1330–1336. IEEE, 2023.
- [ZZX14] Qixin Zhu, Kun Zhong, and Guangming Xie. Speed estimation for robotic fish based on pressure sensor. In *The 26th Chinese Control and Decision Conference (2014 CCDC)*, pages 2714–2718. IEEE, 2014.

Reflection probe interpolation for fast and accurate rendering of reflective materials

Katarina Gojković
University of Ljubljana
Faculty of Computer and
Information Science
Večna pot 113
1000 Ljubljana, Slovenia
kg4775@student.uni-lj.si

Žiga Lesar
University of Ljubljana
Faculty of Computer and
Information Science
Večna pot 113
1000 Ljubljana, Slovenia
ziga.lesar@fri.uni-lj.si

Matija Marolt
University of Ljubljana
Faculty of Computer and
Information Science
Večna pot 113
1000 Ljubljana, Slovenia
matija.marolt@fri.uni-lj.si

ABSTRACT

In this paper, we aim to improve rendering reflections using environment maps on moving reflective objects. Such scenarios require multiple reflection probes to be positioned at various locations in a scene. During rendering, the closest reflection probe is typically chosen as the environment map of a specific object, resulting in sharp transitions between the rendered reflections when the object moves around the scene. To solve this problem, we developed two convolutional neural networks that dynamically synthesize the best possible environment map at a given point in the scene. The first network generates an environment map from the coordinates of a given point through a decoder architecture. In the second approach, we triangulated the scene and captured environment maps at the triangle vertices – these represent reflection probes. The second network receives at the input three environment maps captured at the vertices of the triangle containing the query point, along with the distances between the query point and the vertices. Through an encoder-decoder architecture, the second network performs smart interpolation of the three environment maps. Both approaches are based on the phenomenon of overfitting, which made it necessary to train each network individually for specific scenes. Both networks are successful at predicting environment maps at arbitrary locations in the scene, even if these locations were not part of the training set. The accuracy of the predictions strongly depends on the complexity of the scene itself.

Keywords

reflection probes, interpolation, convolutional neural network

1 INTRODUCTION

Rendering reflective materials presents a formidable challenge in real-time computer graphics. Our goal is to render reflections that faithfully mirror the surroundings of a moving reflective object while ensuring fast rendering. Various techniques address this issue, yet the balance between speed and accuracy remains ever-present.

At one end of the spectrum, we encounter techniques yielding highly precise results but demanding considerable time and computational resources, often unsuitable for real-time applications. Path tracing, notably improved by the recent advancements in denoising techniques, serves as a prominent example.

Conversely, there exist techniques capable of fast but less precise outcomes. By far the most common approach in practice are environment maps. However, environment maps capture only the surroundings of a single point in space (i.e. the reflection probe), which may result in inaccurate reflections if the location of the reflection point differs from that of the probe. To some extent, this issue can be addressed by placing multiple reflection probes in a scene and then selecting the closest one at runtime based on the location of the reflective object. This leads to sharp transitions between reflections when the selected reflection probe changes, which can be mitigated by linearly interpolating neighboring reflection probes. Linear interpolation, however, leads to noticeable inaccuracies and distortions in interpolated reflections.

To address these issues, we present two techniques based on convolutional neural networks (CNNs). In the first approach, a CNN with a decoder architecture takes the location of a reflective object as input and outputs an interpolated environment map at that location. The second technique uses a CNN with an encoder-decoder architecture. First, the set of reflection probes is triangu-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

lated, yielding a triangle mesh covering the scene. During runtime, three reflection probes that form a triangle encompassing the reflective object are selected. Their corresponding environment maps, along with interpolation weights determined by the distance of the reflective object from the reflection probes, are input into the CNN. The CNN then outputs the interpolated environment map for the given location within the scene. Training both CNNs involves overfitting to specific scenes, necessitating separate training sessions for each scene. We evaluate the techniques on several scenes to demonstrate their performance and accuracy. Results show that both techniques produce more accurate reflections than simple linear interpolation, at the cost of computational complexity.

2 RELATED WORK

Rendering reflections in computer graphics is a richly explored domain crucial for achieving realistic scenes. While sophisticated algorithms like path tracing offer physically accurate illumination, there is a parallel focus on less computationally intensive methods leveraging ambient images and reflection probes.

Foundational work by Blinn and Newell [3] introduced techniques for applying textures and reflections to curved surfaces, elevating rendering quality using digital signal processing theory and curved surface mathematics. By incorporating a reflection term inspired by Phong's work [9], the authors aimed to simulate more realistic lighting effects, particularly on highly polished surfaces. This involves accurately modeling surface properties and employing precise normal vectors, along with a subdivision algorithm to facilitate the simulation of mirror reflections from curved surfaces.

Structured importance sampling, detailed by Agarwal et al. [11], offers a technique for illumination rendering based on surrounding images. It includes algorithms for sampling lighting textures with visibility consideration and hierarchical layering for surrounding image sampling, optimizing reflection rendering while reducing the required samples.

Ramamoorthi and Hanrahan's contributions [10] introduced efficient reflection representation with spherical harmonic reflection maps, allowing for accurate sampling frequency determination based on error analysis and fast prefiltering methods using spherical harmonic transformations. Further advancements in reflection appearance were made through the prefiltering of ambient images. Kautz et al. [4] presented three algorithms that unify prefiltering methods, including fast hierarchical filtering, machine-accelerated prefiltering, and anisotropic BRDF model prefiltering. The first algorithm provides approximately 10 times faster prefiltering than the brute force method, while the second, op-

timized for real-time usage, provides accelerated pre-filtering. Lastly, the third enables the application of environment map techniques to anisotropic BRDF models for the first time.

Ashikhmin and Ghosh [2] simplified reflection creation with simple blurred reflections using OpenGL capabilities, while Křivánek and Colbert [6] developed filter importance sampling to reduce aliasing errors in real-time rendering.

Manson and Solan's fast filtering method [7] utilized hardware-accelerated trilinear sampling for efficient cube map prefiltering, optimizing coefficients to maintain high-quality results while reducing complexity.

McGuire et al. [8] introduced a novel data structure called light field probes, and two algorithms for real-time global illumination computation in static environments. The ideas from screen-space and voxel cone tracing techniques were applied to this data structure to efficiently sample radiance on world space rays, with correct visibility information, directly within a pixel and compute shaders. The approach improves traditional techniques by eliminating artifacts like light leaking and enabling complex illumination effects in real-time rendering scenarios.

Xia and Kuang [12] presented a novel method for non-uniform probe placement in probe-based global illumination algorithms, aiming to reduce memory waste and improve shading accuracy. The algorithm dynamically adjusts probe positions based on illumination information, by calculating irradiance errors between probes and shading points and employing gradient descent. The method achieves similar rendering quality to existing techniques like DDGI but with fewer probes. However, the algorithm is currently limited to static scenes and light sources.

Rodriguez et al. [11] presented a method for global illumination rendering using illumination textures and reflection probes, optimizing memory consumption with adaptive parametrization and introducing reflection probes to store light paths in the scene for specular reflections.

Finally, Kopanas et al. [5] introduced a novel approach for rendering scenes with curved reflectors, termed Neural Point Catacaustics, using a point-based representation and a neural warp field to model reflection trajectories. By leveraging neural rendering techniques and efficient point splatting, complex specular effects can be synthesized from casually captured input photos. Key contributions include the explicit representation of reflections with reflection and primary point clouds, enabling interactive high-quality renderings of novel views with accurate reflection flow.

3 INTERPOLATION NETWORKS

Our goal was to devise methods for generating accurate environment maps at arbitrary locations in a scene, used for rendering reflective objects. In this section, we present two approaches based on CNNs. Both approaches intentionally leverage overfitting. Our strategy involved training each model with a rich set of scene-specific data, which facilitated the learning of intricate details and subtleties within test scenes, thus enabling the models to adapt effectively to the test scenes.

3.1 First approach: point-based prediction

In the first approach, we provided the CNN with global coordinates (x, y, z) of a reflective object at its input, and tasked it with predicting a corresponding environment map for the given location. The CNN adopts a decoder architecture, transforming global coordinates into an RGB image. First, the input passes through 4 fully connected layers, which expand to 384 output neurons. Subsequently, the CNN features 16 convolutional layers, with batch normalization incorporated after each layer to ensure stable learning. To achieve the desired output image resolution of 512×256 pixels, 5 upsampling layers are inserted after every third convolutional layer, doubling the spatial resolution. ReLU activation functions are applied after each fully connected or convolutional layer. A schematic of the architecture is shown in [Figure 1](#).

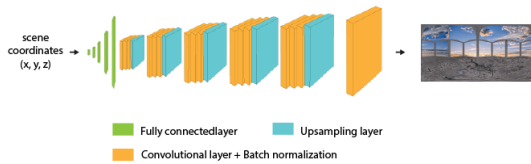


Figure 1: Schematic of the CNN used in the first approach.

3.2 Second approach: triangulation-based prediction

In the second approach, we aimed to enrich the network's input with additional scene context, providing three environment maps along with interpolation weights based on the distance from the reflection probes.

Interpolation weights w_1 , w_2 and w_3 for a point \mathbf{x} in a triangle defined by reflection probe locations \mathbf{p}_1 , \mathbf{p}_2 and \mathbf{p}_3 are computed as follows:

$$\begin{aligned} w'_1 &= (\|\mathbf{p}_1 - \mathbf{x}\| + \varepsilon)^{-1} \\ w'_2 &= (\|\mathbf{p}_2 - \mathbf{x}\| + \varepsilon)^{-1} \\ w'_3 &= (\|\mathbf{p}_3 - \mathbf{x}\| + \varepsilon)^{-1} \\ w_1 &= w'_1 / (w'_1 + w'_2 + w'_3) \\ w_2 &= w'_2 / (w'_1 + w'_2 + w'_3) \\ w_3 &= w'_3 / (w'_1 + w'_2 + w'_3) \end{aligned}$$

We used $\varepsilon = 10^{-9}$ to avoid numerical errors.

3.2.1 Architecture

The CNN in the second approach adopts an auto-encoder architecture. The encoder comprises 5 convolutional layers paired with max pool layers for each of the three input images. The interpolation weights are passed through 4 fully connected layers followed by 5 convolutional layers. Subsequently, the encoded data is passed through the decoder, consisting of 7 convolutional layers. In the decoder, upsampling layers are inserted after each convolution, except the last two, as the desired resolution, which was the same as in the first approach, was achieved at that point. Similar to the first approach, ReLU activation functions are applied after each fully connected or convolutional layer. A schematic of the architecture is shown in [Figure 2](#).

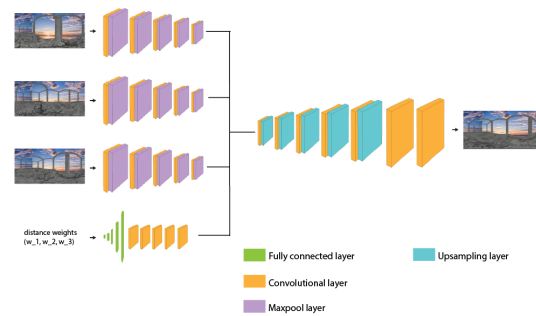


Figure 2: Schematic of the CNN used in the second approach.

3.2.2 Scene triangulation

Each scene has been triangulated based on a manually selected set of reflection probe locations. Delaunay triangulation has been used due to its tendency to avoid sliver triangles. The selection of reflection probe locations varied based on the size and complexity of each scene. In the largest and simplest scene, spanning from $(-400m, -400m)$ to $(400m, 400m)$, 25 reflection probes were determined, while the second and third scenes, although smaller in scale but more intricate, necessitated a denser placement of reflection probes. Specifically, the second scene, measuring $(-5m, 6m)$ by $(7m, 23m)$, had 21 reflection probes, while the third scene, sized $(2m, 25m)$ by $(32m, 50m)$, featured 33 reflection probes. Similarly, in the fourth scene, which spanned from $(5.7, -1.9)$ to $(11.7, -28.9)$, 25 reflection probes were uniformly placed. The fifth scene, although bigger, spanning from $(-45, 70)$ to $(100, -140)$, had its main focus between coordinates $(-45, 8)$ to $(100, -11)$ and $(23, -11)$ to $(40, -140)$. Consequently, there was a denser placement of 25 reflection probes within this area to capture the scene's intricacies. Similarly, In the first scene, reflection probes are densely concentrated in the central region, as it is the most relevant area of the scene, whereas, in

the subsequent three scenes, we aim to distribute the reflection probes as uniformly as possible across the entirety of the scene. [Figure 3](#) shows how the scenes have been triangulated.

At each selected reflection probe, we captured the environment map required as input data for the CNN.

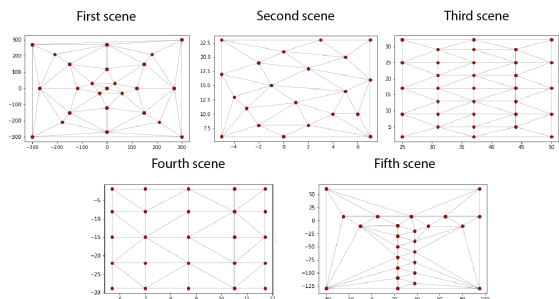


Figure 3: Triangulation of used scenes.

3.3 Learning data

Training the CNNs requires a diverse dataset encompassing input-output pairs for every scene.

For both approaches, panoramic environment maps serve as the reference output data, captured at uniform grid intervals adjusted for scene complexity and size. Specifically, 600 locations were determined for the first scene, 250 for the second scene, 805 for the third scene and 196 for the fourth scene where environment maps were captured. The fifth scene, however, was an exception, with 1210 locations not determined uniformly across the whole scene but only in the most relevant part, which was from $(-45, 8)$ to $(100, -11)$ and $(23, -11)$ to $(40, -140)$.

In the second approach, reflection probe locations were also determined, and panoramic images captured at these points served as input alongside interpolation weights for the locations of reflective objects.

Careful curation of the training set ensures an accurate representation of the depicted scene, facilitating effective network learning.

3.4 Learning procedure

We begin by outlining the training process for both CNNs before delving into a detailed presentation of their performance in [Section 4](#).

During each iteration of the network, input data were first fed through the model to generate predictions in the form of image data. Then the Mean Square Error, which was used as a loss function, was computed from predicted images provided on the CNN's output and true images for the given input coordinates provided in the training set. The Adam optimizer was then employed to minimize the model's error.

For both developed CNNs, three key parameters were determined: the learning rate, batch size, and the number of epochs. Throughout the training of both networks, the learning rate was set to 0.0005. For the first network, the batch size was set to 32, and the number of epochs was set to 16,384, while these hyperparameters were halved for the second network due to the increased complexity of the input data, which included three additional images.

Both models for all three scenes were trained using an AMD Ryzen Threadripper 1950X 16-Core CPU in conjunction with an NVIDIA TITAN V GPU.

The learning procedures for each scene are shown in [Figure 4](#). The loss function values of the last steps are presented in [Table 1](#) and the training duration for both CNNs and each scene is presented in [Table 2](#).

Las Value of Loss Function	First CNN	Second CNN
First Scene	129.688	81.607
Second Scene	192.072	119.955
Third Scene	391.773	333.901
Fourth Scene	109.982	70.636
Fifth Scene	194.705	*

Table 1: Loss function values for the last step by scenes.

	Training time of first CNN	Training time of second CNN
First scene	20 h 50 min	26 h 59 min
Second scene	9 h 10 min	10 h 40 min
Third Scene	28 h 2 min	36 h 21 min
Fourth Scene	6 h 53 min	8 h 58 min
Fifth Scene	42 h 9 min	54 h 30 min

Table 2: Training duration for each scene.

4 RESULTS

Our evaluation relies on both quantitative metrics, such as Mean Squared Error (MSE), and subjective assessments, leveraging the Learned Perceptual Image Patch Similarity (LPIPS) metric for a comprehensive understanding. Both CNNs were trained and tested across five distinct scenes: a simplistic building environment, a detailed room [\[3\]](#), a forest [\[4\]](#), a space ship corridor [\[5\]](#), and a city [\[6\]](#) (see [Figure 5](#)).

² Some initial training steps with significantly different values are excluded from the graph for better visualization.

³ <https://sketchfab.com/3d-models/the-king-s-hall-d18155613363445b9b68c0c67196d98d>

⁴ <https://www.cgtrader.com/3d-models/exterior/landscape/forest-house-scene>

⁵ <https://www.cgtrader.com/free-3d-models/science/other/sci-fi-corridor-6b2e3194-8a54-4846-a290-5bf473a88a74>

⁶ <https://www.cgtrader.com/free-3d-models/architectural/architectural-street/city-scene-719d674f-97b6-49a1-8399-de7722a60f5e>

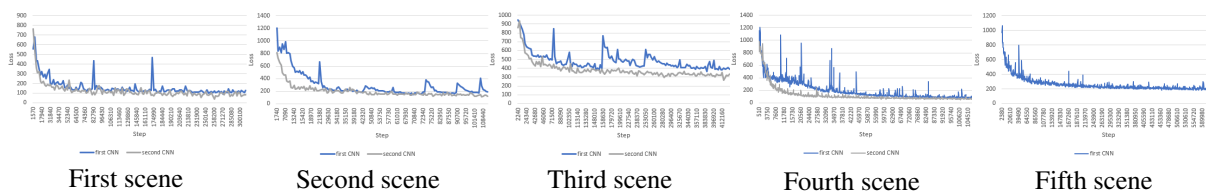


Figure 4: Change in the loss function during the training of the first and second CNNs for each scene. The training data for the second CNN for the fifth scene is not presented due to corruption.

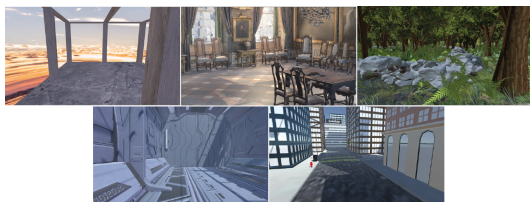


Figure 5: Five scenes on which we trained and tested CNNs.

4.1 Evaluation of predicted environment maps

To measure the accuracy of the predicted environment maps, we compared them against actual scenes at specific points in the room. We assessed both approaches and also generated interpolated images of reflection probes using the same spatial triangulation as described in Section 3. These interpolated images were created by weighting the pixels of three reflection probes.

For each scene, we selected four distinct points within the scene for evaluation. Two points were from the training set (Point A and Point B), and two were outside it (Point C and Point D). This evaluation process provided insights into the models' predictive accuracy across various scenarios.

Models were tested on Intel(R) Core(TM) i7-10510U 4-Core CPU and NVIDIA GeForce MX250 GPU.

4.1.1 First Scene: Simple Building

The first scene depicts a simple environment with minimal details—a building comprising a floor and eleven columns against a sky backdrop.

Mean Squared Error

The Mean Squared Error (MSE) values for the first scene are shown in Table 3.

Point	First CNN	Second CNN	Interpolated Reflection Probes
Point A	100.38	61.59	418.28
Point B	235.40	247.20	1085.67
Point C	718.93	744.58	1152.46
Point D	428.81	443.50	1451.75

Table 3: MSE values according to the real image of the surroundings for the first scene.

The first CNN shows slightly higher MSE values for points inside the training data set, while the second CNN exhibits marginally larger errors for points outside of it. Both CNNs significantly outperform interpolated reflection probes.

To visually represent the disparities between predicted and actual images, Figure 6 presents error maps, illustrating discrepancies for one point within the learning set and one outside. These maps delineate areas of inconsistency between predictions and real surroundings. Correct values are depicted in gray, while errors are indicated by varying colors, with greater intensity denoting larger errors. Error maps reveal fewer errors for both CNN approaches compared to the interpolated reflection probes. The first CNN demonstrates less error, particularly noticeable on the ground and distant towers. However, slight variations in sky color are noticeable in the predictions of the first CNN.

Learned Perceptual Image Patch Similarity

Table 4 shows the Learned Perceptual Image Patch Similarity (LPIPS) values for the first scene.

Point	First CNN	Second CNN	Interpolated Reflection Probes
Point A	0.0584	0.0418	0.0868
Point B	0.1172	0.1187	0.1870
Point C	0.1694	0.1795	0.1893
Point D	0.1302	0.1275	0.1993

Table 4: LPIPS values against the real image for the first scene.

Both CNNs exhibit comparable performance in terms of perceptual properties such as color, texture, and shape, outperforming interpolated reflection probes.

4.1.2 Second Scene: Room

The second scene depicts an intricately detailed room adorned with various decorations.

Mean Squared Error

The MSE values for the second scene are presented in Table 5.

The second CNN shows lower errors in most cases, especially within the training set. Both networks outperform interpolated reflection probes.

Point	First CNN	Second CNN	Interpolated Reflection Probes
Point A	134.81	114.39	952.968
Point B	178.43	134.51	1410.76
Point C	1075.77	995.61	1273.71
Point D	613.60	633.00	1197.21

Table 5: MSE values for the second scene.

Visualizations of error maps confirm the superiority of our approaches over interpolated reflection probes, particularly within the training set. Although the advantage diminishes slightly for scenarios outside the training set, our approaches consistently outperform the interpolated probes. The correlation between the error maps and mean squared error values further validates the robustness of our approaches.

Learned Perceptual Image Patch Similarity

Table 6 displays the LPIPS values for the second scene.

Point	First CNN	Second CNN	Interpolated Reflection Probes
Point A	0.1750	0.1538	0.2494
Point B	0.1673	0.1448	0.2401
Point C	0.2535	0.2418	0.2074
Point D	0.2238	0.2030	0.2565

Table 6: LPIPS values for the second scene.

Our approaches show a significant perceptual advantage over interpolated reflection probes on training set data. However, this advantage decreases with out-of-sample data, especially in the third case, where the LPIPS metric for interpolated reflection probes is lower than for our methods.

4.1.3 Third Scene: Forest

The third scene depicts a forest teeming with trees of various sizes, low shrubbery, and scattered rocks. Dominated by green hues, the scene boasts intricate details.

Mean Squared Error

The MSE values for the third scene are listed in Table 7.

Point	First CNN	Second CNN	Interpolated Reflection Probes
Point A	517.92	498.21	1876.48
Point B	367.71	338.59	1269.22
Point C	686.12	799.88	1203.10
Point D	805.69	832.67	1961.62

Table 7: MSE values for the third scene.

The second CNN performs better within the training set, while the first CNN is better for points outside the training set, contrary to visual impressions favoring the second approach. Both outperform interpolated reflection probes.

Analyzing the error maps reaffirms the superior performance of our approaches over the interpolated reflectance probes, although this distinction is less apparent in the actual images, especially within the forested regions.

Learned Perceptual Image Patch Similarity

Table 8 presents the LPIPS values for the third scene.

Point	First CNN	Second CNN	Interpolated Reflection Probes
Point A	0.3055	0.2653	0.2802
Point B	0.2720	0.2491	0.2043
Point C	0.3491	0.3284	0.2769
Point D	0.3316	0.2998	0.3088

Table 8: LPIPS values for the third scene.

The LPIPS values show less pronounced deviations, with interpolated reflection probes performing comparably to the CNNs.

4.1.4 Fourth Scene: Spaceship Corridor

The fourth scene depicts a small spaceship corridor characterized by predominant grayscale hues.

Mean Squared Error

The MSE values for the fourth scene are shown in Table 9.

Point	First CNN	Second CNN	Interpolated Reflection Probes
Point A	129.08	57.90	1031.44
Point B	111.29	67.39	1413.32
Point C	1303.07	1519.17	1394.56
Point D	856.83	1001.12	1321.07

Table 9: MSE values for the fourth scene.

The second CNN shows lower MSE for points within the training set. Surprisingly, the first CNN presents a marginally lower MSE for points outside the training set. Moreover, in these cases, the MSE of interpolated reflection probes is not substantially greater.

Error maps reveal more errors in the first CNN's predictions than the second's within the training set, and even more in interpolated reflection probes. Outside the training set, error maps are similar for all three approaches.

Learned Perceptual Image Patch Similarity

Table 10 shows the LPIPS values for the fourth scene.

The second CNN demonstrates superior perceptual performance, with both CNNs exhibiting better perceptual characteristics than interpolated reflection probes.

Point	First CNN	Second CNN	Interpolated Reflection Probes
Point A	0.1309	0.0878	0.2248
Point B	0.0991	0.0663	0.2193
Point C	0.2361	0.2334	0.1886
Point D	0.2317	0.1908	0.2405

Table 10: LPIPS values for the fourth scene.

4.1.5 Fifth Scene: City

The fifth scene depicts a cityscape with diverse buildings and two main thoroughfares, which were the focal points for both network training endeavors.

Mean Squared Error

The MSE values for the fifth scene are presented in [Table 11](#).

point	first CNN	second CNN	interpolated reflection probes
Point A	168.45	214.49	2470.10
Point B	236.60	265.06	2303.81
Point C	578.39	476.49	3016.37
Point D	468.92	337.91	1169.55

Table 11: MSE values for the fifth scene.

The first CNN outperforms the second CNN for points within the training set, while the second CNN shows slightly better performance for points outside the training set. Both CNNs significantly outperform the interpolated reflection probes across all points.

For this scene as well, error maps were calculated, revealing fairly similar results between our first and second approaches for points within and outside the training set. Conversely, error regions for interpolated reflection probes are considerably larger and more intense for all points compared to our approaches.

Learned Perceptual Image Patch Similarity

[Table 12](#) shows the LPIPS values for the fifth scene.

point	first CNN	second CNN	interpolated reflection probes
Point A	0.1261	0.1395	0.3036
Point B	0.1733	0.1607	0.3338
Point C	0.2074	0.2120	0.2697
Point D	0.1936	0.1933	0.2564

Table 12: LPIPS values for the fifth scene.

From a perceptual standpoint, both of our approaches are fairly equivalent across all points, exhibiting significantly superior performance compared to interpolated reflection probes.

4.2 Performance Evaluation Results

In addition to assessing prediction performance, we gathered runtime data for both CNNs to evaluate their

suitability for rendering reflections in real-time. Total execution time (including pre-processing and post-processing) and prediction time were measured over 1000 consecutive predictions, from which the average time per prediction was calculated.

Regarding speed, the first network exhibited superior performance, with an average total execution time of approximately 173 milliseconds with a prediction time of about 95 milliseconds. In contrast, the second network had a total execution time of around 333 milliseconds, with a prediction time of approximately 295 milliseconds.

We also compared their performance in real-time execution in the Unity game engine. Specifically, we evaluated rendering speed, measured in frames per second (FPS) [Table 13](#).

Speed Renderings	First CNN	Second CNN	Reflection Probe
	6-10 FPS	0.7-2 FPS	5-150 FPS

Table 13: Rendering speed comparison for different reflection rendering approaches.

The second approach's CNN proved to be unsuitable for real-time applications due to its slow performance. While the real-time reflection probe demonstrated excellent efficiency in the simplest scene, achieving a consistent rendering speed of approximately 150 FPS, it struggled in more complex scenes. In these scenarios, the first CNN performed comparably to the real-time reflection probe, but with smoother transitions between reflections, making it the preferred option for real-time applications.

5 DISCUSSION

In this section, we critically analyze the outcomes and metrics discussed in [Section 4](#). We explore the advantages and drawbacks of each approach, consider their respective applications, and propose avenues for future enhancement.

Both approaches were trained and tested on five different scenes. We initiated the development of both approaches with the first and simplest scene. This scene provided insight into the promising direction of our approach, as we successfully trained the networks for a straightforward and repetitive environment. For the second scene, we selected a room filled with intricate details to assess the ability of the networks to learn such features. The third scene depicted a diverse environment rich in detail but predominantly characterized by a single color. Here, we aimed to evaluate the approaches' effectiveness in reproducing details in such environments. With the same objective, we trained the fourth scene, albeit significantly smaller than the third and featuring a different dominant color. This enabled us to assess the influence of dominant color and scene

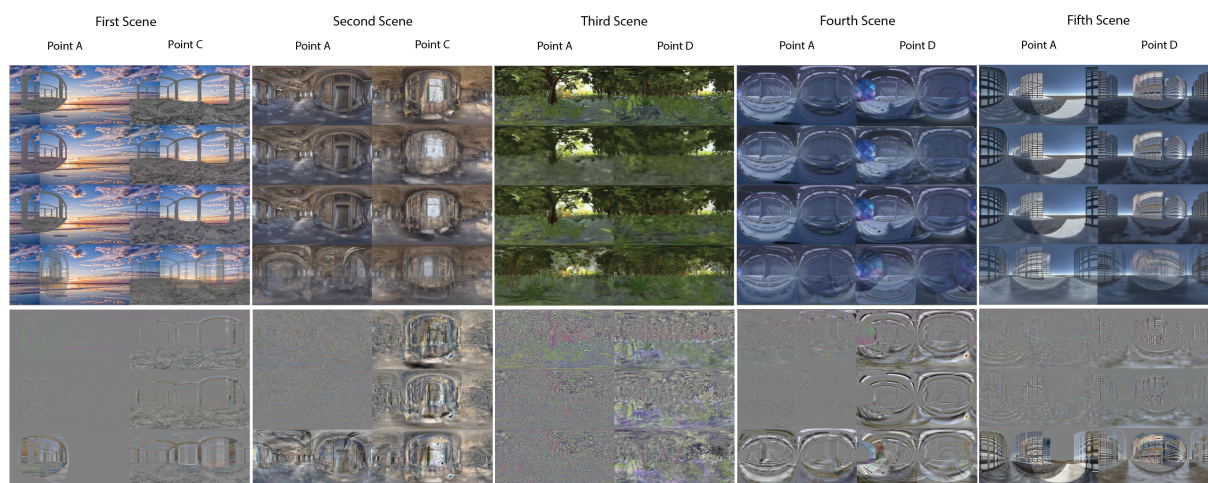


Figure 6: The upper half illustrates the prediction results of the two networks alongside the actual environment map and the interpolated reflection probes for all five scenes. In each scene, the first column showcases the outcomes for the point included in the training set, while the second column displays the outcomes for the point not included in it. The lower half displays error maps corresponding to the rendered results.

size on the results. Finally, the fifth scene contains numerous details, which are uniformly distributed and exhibit regular shapes throughout the scene (unlike the second scene, where details are unevenly distributed and irregular). Here, we aimed to investigate the impact of detail distribution and shape on the results.

5.1 Evaluation of results

We analyzed the effectiveness of the networks in various aspects, including prediction performance, color accuracy, and real-time processing.

Initially, we examined the fidelity of the surroundings' depiction, assessing whether objects were accurately represented and the level of detail in each depiction. Notably, the images generated by the second approach rendered buildings, especially those outside the training set, less accurately compared to the first network (Figure 7). Despite this, both networks performed similarly in rendering objects in other scenes. However, the second network's predictions exhibited more detail, likely due to the additional input data.

The first approach demonstrated superior proficiency in predicting details of rectilinear shapes, particularly when such shapes were in stark contrast with their surroundings, which was most pronounced in the first and fifth scenes.

Furthermore, the second network demonstrated superior color prediction, particularly in scenes with similar color tones between objects and the background (Figure 8). For instance, in the first scene, the sky and building's colors closely resemble each other, with the second network providing more accurate predictions. Conversely, the first network struggled with color nuances, especially apparent in the forest scene. The second net-



Figure 7: The figures illustrate the rendering of the first scene at the point, which was not part of the training set. The first image is the real image at that point, the second is the prediction of the first CNN, and the third is the image predicted by the second CNN.

work's enhanced color perception also contributed to its ability to capture finer details.



Figure 8: The images in the first column depict actual surroundings, while those in the second column represent predictions made by the first CNN, and those in the third column depict predictions made by the second CNN. The images reveal an error in recognizing the correct lower hues by the first network.

In general, both approaches exhibited fairly comparable success in predicting environment maps at points outside the training set. However, the small amount of training data was evident in the second and fourth scenes, where the first approach performed relatively poorly. Consequently, it can be inferred that an undesirable degree of overfitting occurred. This assertion is further supported by the MSE values, which are very small for points within the training set but significantly larger for points outside the training set.

Effective utilization of our approaches requires careful scene design and consideration of lighting placement before data capture and network training. Retraining the CNN after any scene changes is crucial to maintaining accurate and realistic reflections. Additionally, careful placement of static objects and lighting effects ensures their influences are accurately captured in the training data and reflected in predictions.

Despite these considerations, a significant drawback of our approaches remains the absence of reflective object shadows in rendered reflections. Since the training data is captured without it being present in the scene, its shadows are not included, preventing the rendering of self-reflections and reflections between multiple reflective objects. This limitation persists even with meticulous scene design and data capture planning. Moreover, reflections of other non-static objects in the scene and their shadows are not present in the captured training data, as we cannot predict their spatial positions in advance at the time of data acquisition.

From a speed perspective, the first network outperformed the second due to processing less data. While the second network's additional input led to more detailed predictions, it came at the cost of slower processing.

Overall, both approaches have demonstrated the capability to predict and depict the surroundings of a given point in the scene fairly accurately, albeit sometimes with a lack of detail. Compared to traditional approaches such as using the nearest reflection probe or interpolating between reflection probes in the scene to depict reflections, our approaches predict and depict the environment of the object more accurately. However, it is notable that these predictions sometimes portray fewer details, resulting in slightly blurred surroundings. Regarding speed, both approaches are slower than traditional methods, although the first approach has a sufficiently fast rendering speed. When compared to real-time reflection probes, which also accurately depict the environment of the object and produce cleaner renderings with more details, our approaches excel particularly in complex environments where real-time reflection probes may falter, especially during the movement of reflective objects, where transitions between different rendered reflections are highly noticeable and sharp, a limitation not present in our approaches.

5.2 Usability of models

The CNN developed in the second approach proves unsuitable for real-time applications. Conversely, the first approach's network is well-suited for real-time applications like gaming, where heavy computational tasks are minimal, allowing for swift network predictions. On the other hand, both CNNs are valuable for rendering reflections in animations or design tools, offering quality results within reasonable processing times.

From the perspective of the quality of predicted environment maps and rendered reflections, our approaches are most useful for reflective objects that lack pronounced reflective properties (such as wrinkled or rough surfaces), where finer details in the rendered reflections are not crucial. However, their reflections contribute to creating a realistic appearance of the scene, where our approaches excel in providing generally accurately depicted surroundings in reflections and smooth, natural transitions between different areas of the environment as the reflective object moves through the scene.

Both approaches excel in environments with static scenes, making them ideal for rendering moving objects against a relatively stable backdrop. However, they are less effective for static objects with pronounced reflective properties, such as mirrors, where traditional reflection probes are more appropriate. Our approaches shine when rendering scenes where moving objects significantly enhance visual realism, such as characters or vehicles traversing a static environment, contributing to a lifelike visual perception of the surroundings.

5.3 Possible improvements

5.3.1 *Enhancing robustness through incorporating varied object heights*

Currently, both approaches are trained solely on scenes where the main activity occurs within a single plane, and the reflective objects move only at a single height (varying only in x and z coordinates). Consequently, both networks are trained with constant y coordinates. It would be worthwhile to further generalize the approaches and make them more robust by incorporating different y coordinates in the training data, thus teaching them to predict reflections for objects at varying heights. In such a scenario, the first CNN would not require additional upgrades, while the second approach would need some adjustments, primarily considering a different spatial decomposition.

5.3.2 *Enhanced prediction quality*

To elevate prediction quality, augmenting network complexity by integrating additional fully connected or convolutional layers could enhance adaptability to training data. However, caution is warranted as increased complexity may elongate prediction times. Advanced CNN layers like capsule and dynamic layers offer potential replacements for traditional convolutional layers, enabling better adaptability to training data without significant model slowdown. Moreover, employing sophisticated loss functions such as perceptual or contrastive loss could further refine predictions by preserving content and style fidelity. Also, post-processing

techniques like noise reduction, sharpening, and contrast enhancement could refine predicted images, augmenting overall quality.

5.3.3 Model speed optimization

Implementing model compression techniques like pruning and quantization reduces model size and prediction times, albeit with potential prediction quality trade-offs. Model caching, wherein predicted images for known inputs are stored to circumvent redundant predictions, offers an efficient strategy for recurrent input scenarios, albeit with increased memory overhead.

5.3.4 Realistic reflections

Exploring methods to prefilter environment maps for more realistic reflections akin to mirror reflections could enhance scene realism.

The absence of self-shadows in object reflections could be addressed via prerendering shadows in training data, possibly using shadow mapping techniques.

6 CONCLUSION

In this paper, we endeavor to enhance existing methods for rendering reflections, particularly those reliant on reflection probes. One of the foremost challenges with these methods lies in rendering reflections as reflective objects move within a scene, and in mitigating the jarring transitions between disparate reflection probes.

To address this issue, we introduce two novel methods leveraging convolutional neural networks (CNNs) to generate environment maps at specific points within a scene. Both methods demonstrate success in predicting scene surroundings, seamlessly blending reflections as objects move through the scene. Particularly in complex environments, the first method outperforms real-time reflection probes provided by the Unity game engine, offering smoother and more natural transitions between reflections. However, due to its slower processing speed, the second method's comparative evaluation remains unfeasible for real-time applications. Notably, the resolution of predicted environment maps varies with scene complexity, rendering the current methods more suitable for rendering blurred reflections.

Future research avenues may explore enhancing image resolution in complex scenes and optimizing method speed. Furthermore, exploring pre-filtering techniques for environment maps could enhance the realism of rendered reflections.

7 REFERENCES

- [1] Sameer Agarwal, Ravi Ramamoorthi, Serge Belongie, and Henrik Wann Jensen. Structured importance sampling of environment maps. Association for Computing Machinery, 2003.
- [2] Michael Ashikhmin and Abhijeet Ghosh. Simple blurry reflections with environment maps. *Journal of Graphics Tools*, 2002.
- [3] James F. Blinn and Martin E. Newell. Texture and reflection in computer generated images. 1976.
- [4] Jan Kautz, Pere-Pau Vázquez, Wolfgang Heidrich, and Hans-Peter Seidel. A unified approach to prefiltered environment maps. In Bernard Péroche and Holly Rushmeier, editors, *Rendering Techniques 2000*, pages 185–196, Vienna, 2000. Springer Vienna.
- [5] Georgios Kopanas, Thomas Leimkühler, Gilles Rainer, Clément Jambon, and George Drettakis. Neural point catacaustics for novel-view synthesis of reflections. *ACM Transactions on Graphics (TOG)*, 41(6):1–15, 2022.
- [6] Jaroslav Křivánek and Mark Colbert. Real-time shading with filtered importance sampling. *Computer Graphics Forum*, pages 1147–1154, 2008.
- [7] Josiah Manson and Peter-Pike Sloan. Fast filtering of reflection probes. *Computer Graphics Forum*, pages 119–127, 2016.
- [8] Morgan McGuire, Michael Mara, Derek Nowrouzezahrai, and David Luebke. Real-time global illumination using precomputed light field probes. In *ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, page 11, February 2017. I3D 2017. URL: <https://casual-effects.com/research/McGuire2017LightField/index.html>.
- [9] Bui Tuong Phong. Illumination for computer generated pictures. In *Seminal graphics: pioneering efforts that shaped the field*, pages 95–101. 1998.
- [10] Ravi Ramamoorthi and Pat Hanrahan. Frequency space environment map rendering. In *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques*, page 517–526. Association for Computing Machinery, 2002.
- [11] Simon Rodriguez, Thomas Leimkühler, Siddhant Prakash, Chris Wyman, Peter Shirley, and George Shirley. Glossy probe reprojection for interactive global illumination. *ACM Transactions on Graphics (TOG)*, pages 1–16, 2020.
- [12] Bo Xia and Fen Kuang. Non-uniform illumination-guided probe placement. In *ICETIS 2022; 7th International Conference on Electronic Technology and Information Science*, pages 1–4, 2022.

GPU Cache Flush Minimization In Render Graph Systems

Roman Sandu
Phystech School of Applied
Mathematics and
Computer Science
Moscow Institute of Physics
and Technology
Institutskiy Pereulok, 9
Dolgoprudny, Moscow
Oblast, 141701, Russia
sandu.ra@phystech.edu

Alexandr Shcherbakov
Faculty of Computational
Mathematics and
Cybernetics
Lomonosov Moscow State
University
Moscow, 119991, Russia
alex.shcherbakov@
graphics.cs.msu.ru

ABSTRACT

Modern graphics APIs expose control over the infamously non-coherent GPU caches to application programmers through the mechanisms of pipeline barriers and render passes. A developer is then asked to group together their GPU computations based on memory access patterns such that cache flushes and invalidations are minimized, but render graph systems enable automation of this process. In this paper, we study the problem of finding an optimal execution order for a frame graph to minimize the amount of render pass breaks, which in turn minimizes cache control operations. We formulate and analyze a novel NP-complete problem MLGP and use it to propose an approach to render pass merging that results in 30% less render pass breaks when compared to previous works.

Keywords

frame graph, render graph, gpu, barrier, render pass, vulkan, dx12, tile based deferred renderer

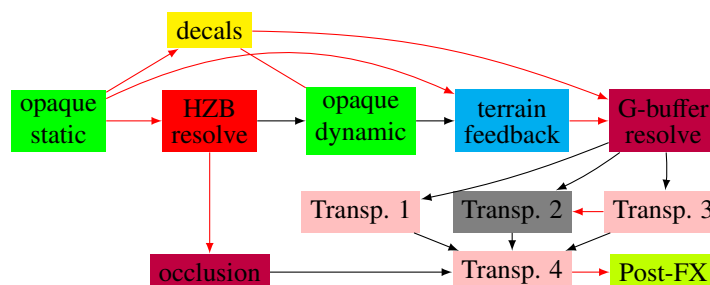


Figure 1: A contrived sample frame graph represented through a Λ graph. Nodes are computations, arrows represent causality or precedence, red edges represent barriers. Some precedence edges also require a barrier. Colors represent unique frame buffers requested by nodes. We are interested in finding a node execution order that enables optimal barrier batching and minimal render target changes.

1 INTRODUCTION

Frame Graphs and Cache Management

Throughout the history of computing, graphs have proven themselves to be an exceptionally effective tool for describing and working with computation. Most compilers use control flow graphs as one of the interme-

diat representations of a program; several approaches to concurrency use graphs explicitly to represent computations; various visual programming systems use graphs as their representation of choice, including graph-based shader authoring and gameplay programming systems as seen in industry-standard engines like the Unreal Engine [Epi]. Low-level graphics programming is no exception, most engines are either in the process of migrating from an immediate-style command dispatching architecture to a *frame graph* [ODo17] (also known as a *render graph* [Wih19]) based architecture, or have done so several years ago. A frame graph runtime library provides its user with an explicit node abstraction, which declaratively describes a computational atom. Different systems chose different granularity for the computations

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

abstracted inside nodes, from single dispatches of compute shaders or a fixed rendering algorithm, to an entire pass of rendering to a fixed set of render targets. The nodes communicate with each other through a resource system provided by the runtime, and the set of all user-created nodes constitute a computational graph that describes the process of computing a single application frame which is to be presented on the screen.

It is noteworthy that a principally different graph-based approach has recently appeared in the Godot engine [Ban24]. Godot uses an immediate-style user-facing API, but does not translate user-dispatched GPU commands directly into low-level graphics API command lists, instead constructing a data dependency graph from them, which is then processed analogously to traditional frame graphs. This approach is close in spirit to control flow graphs used by various compilers and allows to reap most benefits of a frame graph like system without breaking compatibility with existing application code.

The benefits of a graph-based approach are manifold. User-facing frame graph systems provide a robust architectural framework for organizing graphics application code, and both user-facing and Godot-style graph systems enable automatic optimization of various aspects of an application: memory allocation, work scheduling, resource management, cache management, and several others. For this paper, we are interested in cache management and work scheduling in particular.

While the previous generation of graphics APIs have hidden all of cache management infrastructure away from an application programmer inside the device-level driver, such an approach has proven itself to be quite inefficient at times, especially on mobile tile-based deferred renderer architectures, and so the new generation of graphics APIs, including Vulkan and Direct3D 12, delegates cache management to the application developer. Focusing on the Vulkan API, this is manifested in two aspects: pipeline barriers and render passes. Both of these are not exclusively cache management tools, however. A pipeline barrier is a tool that controls 3 aspects of an application:

- 1) cache invalidation and flushing,
- 2) GPU pipeline synchronization,
- 3) texture layout transitions.

Render passes, on the other hand, are a tool for controlling the special rendering output merging hardware available in most GPUs. Several parts of the rasterization pipeline can be implemented in hardware more optimally than in software via general-purpose memory operations, including Z-testing, alpha-testing and simple rendering writes. In particular, tile-based deferred renderer GPU architectures store the textures which participate in the output merging stage, usually referred to as *attachments*, in a special tile cache, which is never flushed or invalidated throughout a single render pass.

Obviously, any performance-sensitive program should strive to maximize cache locality, or in other words, minimize cache flushes and invalidations. For programs that execute on the GPU, this problem is two-fold: cache access must be independently optimized inside shader code and inside the high-level command stream executed by the GPU. The former is similar to CPU cache access optimizations and is covered extensively in literature (e.g. [Bav14]), while the latter is a problem that is novel for application programmers and has been previously solved by proprietary means inside drivers for OpenGL, DirectX11 and similar.

Being interested in frame graphs, we focus on the latter. In terms of the Vulkan graphics API, this means minimizing the amount of render passes and barriers that are executed throughout a frame while preserving the functionality and correctness of a program. Various work items with compatible *frame buffers*, the sets of active attachments, should be grouped together into render passes. Barriers should be grouped together too, as testing shows that fine-grained cache control exposed by the Vulkan API is usually not leveraged by device drivers, and a flush of any memory region from a certain cache usually leads to a flush of the entire cache. Moreover, as barriers also synchronize various GPU pipeline stages, generally by preventing new work from being scheduled onto a processing unit before previous work was completed, grouping them also minimizes the amount of stalling throughout the frame. Finally, it must be noted that the Vulkan API prohibits barriers from being executed within render passes, so minimizing the amount of render passes by grouping work together also automatically minimizes the amount of *grace points* throughout the frame at which a barrier may be executed, and so naturally leads to grouping of barriers.

2 RELATED WORKS

One of the earlier public works on merging render passes inside a frame graph is Hans-Kristian Arntzen's *Granite* engine [Arn17]. In *Granite*, each node represents an entire logical render pass that is then translated into a Vulkan subpass. The algorithm employed for render pass merging is executed after the graph has been sorted into a linear execution order and tries to produce a new ordering that greedily maximizes reasonable scoring heuristics of subpass attachment reads being forcibly merged with the writer pass, dependent subpasses being as distant from each other as possible, and not ending a pass for as long as possible. This engineering approach to the problem fits mobile-oriented development quite naturally, as it focuses on various subpass interactions that often occur in mobile Vulkan-based applications, but lacks rigorous analysis of effectiveness of the algorithm.

The approach *Granite* takes for its user-facing API is what may be called a *coarse-grained* approach, meaning

that each node represents an entire Vulkan-style subpass, or even an entire render pass for engines that do not target mobile devices. As far as the authors are aware, most commercial engines take a similar approach to their frame graph API: Unity [Tec; TAC21], Unreal [Epi], Frostbite [ODO17], one of Activision's engines and AMD's RPS [Adv] all use coarse-grained nodes. Among these, it is publicly known that render pass merging is performed only for Activision's Task Graph Renderer, according to their talk at Rendering Engine Architecture Conference 2023 [Cha23], although algorithmic details are not available publicly. This is to be expected, as for a render graph runtime with coarse-grained nodes rendering code is already grouped together based on what render pass it belongs into, except for the case of mobile rendering, where still manual pass management is usually employed.

An alternative approach is what can be referred to as *fine-grained* nodes. With this approach, each node represents a computation that may be smaller than a single pass or subpass, and hence each pass in the resulting GPU command stream consists of commands recorded by multiple nodes. Consider an opaque G-buffer render pass. Coarse-grained approach suggests that there should exist a single node responsible for this pass that dispatches rendering of multiple systems, plugins or modules, while a fine-grained approach suggests that any system that needs to render opaque geometry to the G-buffer should create its own node with its own «virtual» render pass and let the frame graph runtime merge these nodes into a single pass. This approach provides architectural benefits to the developer:

- no additional event-like system needs to be employed to make passes present in the engine by default customizable by optional subsystems;
- if desired, it is possible for an optional subsystem to forcibly break a render pass mid-way to read an intermediate result of rendering without modifying other code, for example, to create a hierarchical culling Z-buffer [GKM93] that contains only static geometry.

Note that in the case of fine-grained nodes, it is crucial for the runtime to have a robust render pass merging algorithm, as a single logical pass might consist of dozens of subsystem nodes. On the other hand, while coarse-grained nodes do not technically require such an algorithm, it is still desirable to have one, as extensibility of passes being handled outside of the frame graph runtime can lead to subsystems creating passes that are identical to other existing passes throughout a project's long lifetime or when the code that creates the initial pass is inaccessible for the developer.

As for the algorithmic problems described in this paper, to the best of authors knowledge, they are completely novel and have not been tackled previously.

Our Contribution

We focus on fine-grained frame graph runtimes and attempt to solve the render pass merging problem, although our results can also be applied to coarse-grained runtimes, or even Godot-like render graphs. We consider only the simplest case of each resulting render pass being constrained to have a single subpass, leaving multi-passes for future work. We formulate a rigorous statement of the *minimal lambda graph partition* problem that we consider to be general enough to be applicable to any frame graph system, the solution to which is then used to produce a node execution order that is optimal in its amount of render pass breaks. This problem is then analyzed and shown to be NP-complete even under some sensible regularity conditions. Finally, we present a greedy approach for solving this problem that results in graph execution orders that have about 30% less render passes than a baseline approach that closely matches that of the Granite engine.

3 EARLY STREAK SEGMENTATION

Mathematical model

For the purposes of this paper, the following mathematical model of a frame graph will be used.

Definition 1. A Λ -graph is a tuple (V, P, C, L) , where (V, P) is a directed acyclic graph, (V, C) is an undirected graph, and $L : V \rightarrow \mathbb{N}$.

The set V represents nodes present in a frame graph; the set of directed edges P represents precedence order between nodes, constraining possible execution orders; the set of undirected edges C models pairs of nodes that are in conflict with each other and require a barrier to be executed between them; and finally the label function L represents unique frame buffers used by each node, or more generally, any mutually exclusive global state that is expensive to change on the target platform. A possible frame graph of an application represented through this definition can be seen on figure 1.

Informally, the problem we are trying to solve is as follows. Find a topological sort of (V, P) and a partition of it into as few contiguous sequences as possible, such that within each sequence there are no conflicting nodes and all labels are the same.

When faced with this problem, the first approach that comes to mind is choosing the next node to be executed to have the same frame buffer as the previous one, which is close to Granite's approach. It must however be noted that depending on the algorithm chosen for topologically sorting, the optimal solution may not be achievable at all. Both a depth-first and breadth-first traversal based topological sorting algorithms are restricted in the set of possible orders of traversals, and the optimal order may lie outside this restricted set. This motivates us to

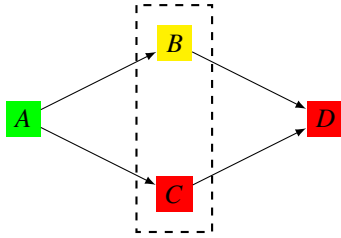


Figure 2: Simplest example where baseline Granite-like approach fails. Colors represent different frame buffers (including lack thereof), dashed outline represents the set of nodes among the which the algorithm must make a choice, node A already being scheduled as the first one. If C is selected as the next node to be executed, the next one has to be B, and so the resulting order is A, C, B, D and the algorithm has failed to merge C and D into a single pass.

always prefer Kahns algorithm [Kah62], which enables achieving any valid topological sort order from a graph by varying the strategy of selecting the next node to be processed. And so the first naive algorithm in its fullness would be using Khans topological sorting algorithm and prioritizing nodes with the same frame buffer as the last processed node when choosing the next node to process. This, however, does not yield satisfactory results on relatively straightforward examples, e.g. figure 2, which motivates the following formalization of the problem.

Definition 2. A *condensation* of a directed graph with respect to a partition of the node set $V = V_1 \sqcup \dots \sqcup V_s$ is a directed graph (W, Q) , where $W = (1, \dots, s)$ and $Q = \{(i, j) \mid \exists v \in V_i, \exists w \in V_j, (v, w) \in P \wedge i \neq j\}$.

Definition 3. Call a partition of the node set *correct* for a Λ -graph if it satisfies the following:

- 1) $\forall i, \forall v, w \in V_i, \lambda(v) = \lambda(w)$;
- 2) $\forall i, \forall v, w \in V_i, \{v, w\} \notin C$;
- 3) and the condensation w.r.t. to it is acyclic.

The subsets V_i constituting the partition will be referred to as *buckets*.

Definition 4. The problem of finding a correct partition of minimal size s is called *minimal lambda graph partition* and denoted as MLGP.

The statement of MLGP can informally be understood as partitioning a frame graph into render passes before it is sorted, and correctness conditions correspond to the fact that it should be possible to produce a node execution order with render passes corresponding precisely to partition buckets.

In presence of a minimal partition for the lambda graph, a corresponding execution order can be achieved as shown in algorithm 1. This algorithm is de-facto equivalent to sorting the condensation graph first and then sorting nodes within each partition set according to edges from P afterwards.

Next, note that when an optimal solution to the initially stated informal problem of sorting the graph while minimizing the amount of render pass breaks is available, a correct partition that will yield the same amount of passes when sorted with algorithm 1 can clearly be constructed. Hence constructing a minimal partition and then sorting the graph while using it will indeed solve the initial informal problem.

This mathematical statement is not exclusive to render-pass related problems, as each set of the partition simply represents a contiguous streak of nodes that can be executed with no barriers or global state changes (e.g. frame buffer changes) as represented by L . Following this line of thought, we call this approach *early streak segmentation*, as opposed to segmenting the graph into streaks mid-flight while sorting it.

Analysis of MLGP

First of all, $MLGP \in NP$, as checking the correctness of a partition can be done in polynomial time. Moreover, clearly,

Claim 1. MLGP is NP-complete.

Proof. Observe that a correct partition of a lambda graph (V, \emptyset, C, id) is equivalent to finding the chromatic number of (V, C) . \square

Algorithm 1 Modification of Khan's algorithm that builds an execution order based on a correct partition of a Λ -graph.

```

for  $v \in V$  do
     $D_v \leftarrow$  out-degree of  $v$  in  $(V, P)$ 
end for
for  $i \leftarrow 1$  to  $s$  do
     $E_i \leftarrow$  num. edges  $(v, w) \in P$  s.t.  $v \notin V_i$  and  $w \in V_i$ 
end for
 $F \leftarrow$  nodes with out-degree 0
 $R \leftarrow$  empty list
Denote  $V_v^{-1} = i$  when  $v \in V_i$ 
repeat
     $v \leftarrow$  any element  $w$  of  $F$  s.t.  $E_{V_v^{-1}} = 0$ 
if  $R$  not empty then
         $p \leftarrow$  last element of  $R$ 
         $v \leftarrow$  any element of  $F \cap V_p^{-1}$ 
end if
Append  $v$  to  $R$ 
for  $w \rightarrow v$  do
    Decrement  $D_w$ 
    Decrement  $E_{V_v^{-1}}$ 
if  $D_w = 0$  then
        Add  $w$  to  $F$ 
end if
end for
until no elements remain in  $F$ 
return  $R$ 

```

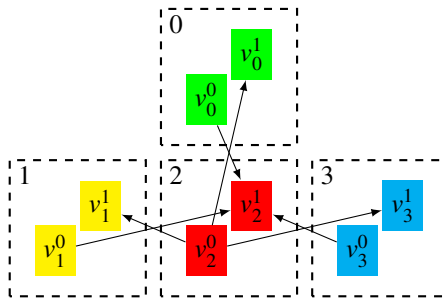


Figure 3: The construction described in claim 2. The initial graph is $V' = \{0, 1, 2, 3\}$, $E' = \{\{0, 2\}, \{1, 2\}, \{2, 3\}\}$. Nodes in V' are depicted through dashed rectangles, and the corresponding Λ graph nodes are located inside the rectangles, where colors represent different labels. The maximal anti-clique is $\{0, 1, 3\}$, and merging the corresponding nodes pairwise minimizes the partition.

Calculating chromatic numbers for arbitrary graph is known to be a notoriously hard problem, and so some regularity conditions must be enforced on Λ -graphs that we are to consider. Inspired by other fields of study, as well as our observation of graphs of various applications, we formulate a requirement analogous to lack of data races inside a multi-threaded program.

Regularity condition 1. $\forall \{v, w\} \in C, v \rightsquigarrow w \vee w \rightsquigarrow v$, where \rightsquigarrow denotes reachability. In other words, any conflict is ordered by precedence edges.

We have observed this restriction to be sensible for actual applications, and so have enforced it in the design of the user API for our frame graph runtime. One case where enforcing it has proven to be a problematic is careless superfluous usage of decompressed read-only depth attachments in legacy code, but such problems can and should be resolved on the application programmer side and not inside a frame graph runtime, as they lead to performance issues in any case due to decompressing and recompressing the depth buffer.

This condition is not enough to make the problem tractable, although the proof is less trivial.

Claim 2. Even under the regularity condition 1, MLGP is NP-complete.

Proof. We shall present a polynomial time reduction of the maximal anti-clique problem to MLGP. Consider an arbitrary graph (V', E') . The construction is then as follows:

- 1) $V = \bigcup_{i \in V'} \{v_i^0, v_i^1\}$;
- 2) $E = \bigcup_{(i,j) \in E'} \{(v_i^0, v_j^1), (v_j^0, v_i^1)\}$;
- 3) $C = \emptyset$;
- 4) $L(v_i^0) = L(v_i^1) = i$.

A visual representation of this construction in a simple case is shown in figure 3.

Observe that any partition that satisfies the first 2 correctness conditions essentially selects a subset of V' by

either merging the two nodes corresponding to some initial vertex or leaving them in separate partition buckets. Moreover, the set of vertices in V' whose pairs of nodes were merged form an anti-clique as long as the third correctness condition is satisfied, as otherwise a cycle would form in the condensation precisely along the edges generated for an initial edge in E' that prevented the set from being an anti-clique. On the other hand, for any anti-clique in (V', E') , the corresponding partition is correct by construction. Finally, note that the size of an anti-clique S and the size of the partition s are related as $|S| = |V| - s$. Hence, finding a minimal correct partition would indeed yield an anti-clique of maximal size in the initial graph. \square

This fact, however, should not discourage one from trying to solve MLGP in practice, as the construction presented above, per our informal observations, is a characteristic representation of cases where a greedy approach presented below fails to construct a minimal partition. On the other hand, the construction is contrived, as such configurations rarely occur in practice and make little sense from the standpoint of computational graphs.

Greedy Algorithm

We propose a greedy algorithm that is based on the idea of *stacks*, but to justify it we require some additional facts about optimal solutions. Below, the notation \rightsquigarrow is used to represent reachability along edges P or along edges of the condensation graph, and V_i^l is used to denote buckets whose nodes all have $L(v) = l$. Note also that the contents of this subsection all assume regularity condition 1 to hold.

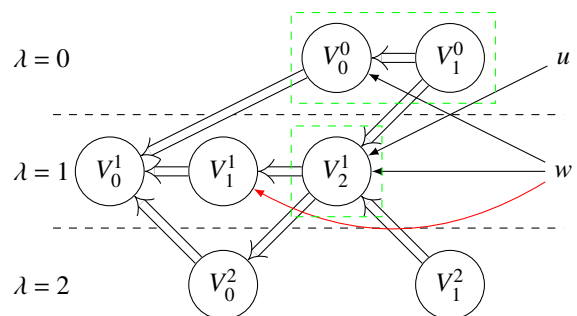


Figure 4: A visualization of the proposed approach. Circled nodes and thick arrows represent the condensation graph, u and w are nodes which are yet to be added into one of the sets, possibly a new one. Thin arrows represent precedence, red ones among them correspond to conflicts. Nodes and sets are laid out according to their labels as 3 stacks. Admissible sets for u and w are depicted using a green dashed outline. If the «deepest is best» greedy bucket selection strategy is used, w must be placed before u to avoid creating a new bucket.

Claim 3. In a minimal correct partition of a Λ graph, for any two buckets V_i^l and V_j^l with nodes in both having the same labels, either $V_i^l \rightsquigarrow V_j^l$ or $V_j^l \rightsquigarrow V_i^l$.

Proof. If two such buckets are not ordered, we are able to merge them together into a single bucket, which would contradict minimality. The resulting partition will still satisfy the first two correctness conditions, as the labels match, and any conflict between their nodes would have resulted in a path between them due to regularity condition 1. Finally, a cycle occurring in the condensation after the merge would imply a path between the two buckets, which is a contradiction. \square

For this reason, we only try to construct partitions that have this property of buckets corresponding to a certain label being linearly ordered. This fact will also henceforth be abuse to assume that the notation V_i^l indexes buckets in this exact linear order, and so $i < j \Leftrightarrow V_i^l \ll V_j^l$. For each label l , the sequence of V_i^l is what we refer to as a *stack*. The proposed algorithm will then try to iteratively take nodes in a topological order of $v < w \Leftrightarrow v \ll w$ and attempt to place each node into one of the buckets in the stack without violating correctness conditions of the partially built condensation graph. Figure 4 should be referred to for visual intuition of these definitions and claims, as well as the algorithm itself.

Definition 5. When adding a new node v with $L(v) = l$, having only outgoing precedence edges and possibly some conflict edges, into an existing Λ -graph and its partition, we call a bucket V_i^l admissible if:

- 1) no nodes in V_i^l conflict with v ;
- 2) for each previously existing node $w \in V_j^l$ such that $w \leftarrow v$, it holds that $j \leq i$;
- 3) for each previously existing node $w \in V_{j'}^{l'}$ such that $l \neq l'$ and $w \leftarrow v$, it holds that $V_i^l \not\ll V_{j'}^{l'}$.

Claim 4. Admissibility is monotonic along the stacks. In other words, if V_i^l is not admissible for v , then $\forall j < i$, V_j^l is not admissible too.

Proof. Observe that thanks to regularity condition 1, a conflict implies that there is an edge from v to some node in V_i^l . Inadmissibility due to violating the first condition then makes V_j^l inadmissible per the second condition. If V_i^l failed the second condition, clearly, V_j^l will too due to claim 3, and same reasoning holds for violating the third condition. \square

This fact greatly aids in efficient implementation of algorithms, as a lot of bucket candidates can be immediately eliminated.

Finally, observe that if a minimal partition of the graph is already known, it is possible to reconstruct it by growing

bucket stacks, iterating nodes in topological order and placing them only into admissible buckets, as a node being placed in an inadmissible bucket will definitely result in an incorrect partition. The only questions that remain is what exact order should nodes be considered in and what admissible bucket should be chosen. As we have shown the problem to be NP-complete, only heuristic approaches make sense, and the heuristics we have chosen is to use an arbitrary order and always prefer the «deepest» admissible bucket, i.e. the one with the smallest index. This is motivated by the fact that a sequence of buckets can accommodate an entire sequence of nodes which are not ordered with anything else but each require a barrier between them, which is a case that does indeed occur in practice for nodes that do compute dispatches. However, in our synthetic tests other «static» strategies like choosing the topmost or the midway admissible bucket did not have a significant influence on the results. Finally, the pseudo-code for the algorithm is shown in figure 2.

Algorithm 2 Proposed greedy solution to MLGP.

```

1: for  $l$  in the image of  $L$  do
2:    $S_l \leftarrow$  empty list
3: end for
4:  $F \leftarrow$  nodes with in-degree 0
5: Chose any topsort order s.t.  $v < w \Leftrightarrow v \ll w$ 
6: for  $v \in V$  in chosen order do
7:    $M \leftarrow \emptyset$ 
8:   for  $w \leftarrow v$  do
9:      $W \leftarrow$  bucket previously chosen for  $w$ 
10:    Add all buckets reachable from  $W$  to  $M$  excluding  $W$  itself
11:   end for
12:    $V_g \leftarrow$  deepest bucket in  $S_{L(v)} \setminus M$ 
13:   if  $\exists w \in V_g$  s.t.  $(v, w) \in C$  then
14:      $V_g \leftarrow$  next bucket in  $S_{L(v)}$ 
15:   end if
16:   Put  $v$  into  $V_g$ 
17:   if  $V_g$  ill-defined then
18:     Create new bucket and append it to  $S_{L(v)}$ 
19:   end if
20: end for
21: return All buckets in  $S$ 

```

4 EXPERIMENTAL RESULTS

The proposed algorithm was implemented in python with $O(|V|^3)$ time complexity and $O(|V|^2)$ space complexity. Random graphs were then used for comparing the proposed approach with a baseline and random sorting. The graph of one of released games that we have access to was observed to have a 0.025 probability of having an edge between two nodes, a 0.013 probability of having a conflict between nodes, and 0.2 proportion

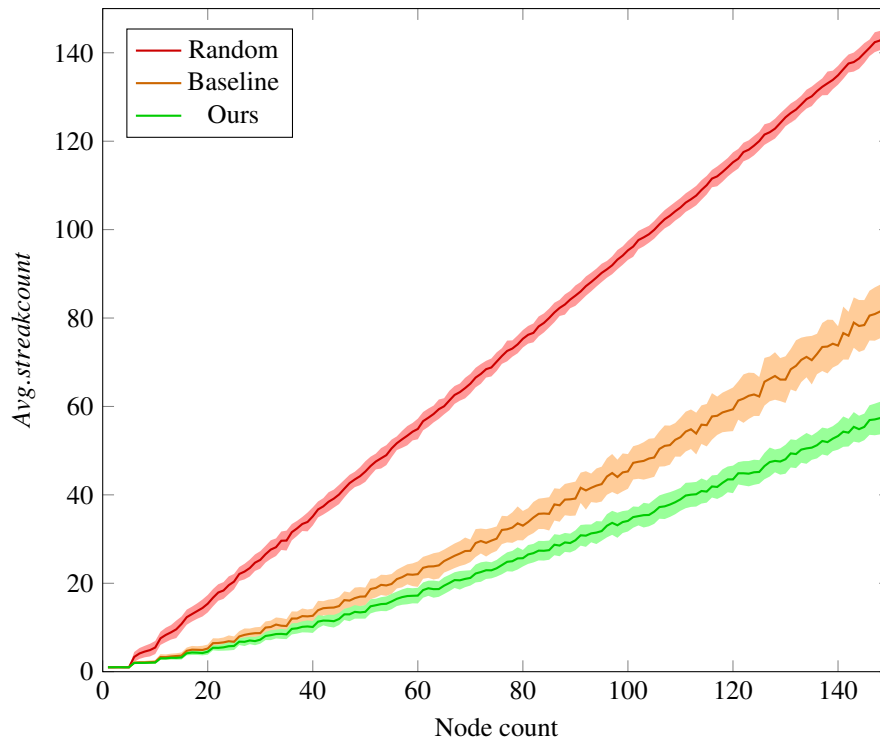


Figure 5: Comparison of proposed streak segmentation approach to baseline and random choice, less is better. For each graph size a sample set of 100 random graphs was generated and test results were averaged, confidence intervals shown as painted regions around plots.

of nodes having a unique frame buffer. The node count of this graph exceeds 100, which is characteristic of fine-grained systems. A sequence of sets of random graphs was then generated, distributed according to these parameters and three node sorting approaches were tested, see figure 5. As one might expect, randomized sorting results in render pass count being proportional to node count. For the baseline approach, Granite-like eager prioritization of continuing the current pass while sorting nodes using Khan’s algorithm was chosen. The plots demonstrate that the suggested approach gives a significant improvement over the baseline of 30% on average. It is also surprising that the proposed algorithm results in lower standard deviation, which implies more predictable performance with respect to changes in the graph when used in practice.

Currently, two live-service games with principally different fine-grained frame graphs are available to us, one targeted at desktop computers, one at mobile devices. Both games currently use manual barrier placement to achieve best possible performance, and furthermore, the latter uses manual Vulkan render pass placement. To move away from this manual approach, a robust render pass merging algorithm was required, and when we attempted to do so using Granite’s approach, we have observed the result to be less optimal than the manual approach, which made it impossible for us to gather practical performance data from mobile devices before

a robust pass merging algorithm was devised and tested in isolation.

However, we were able to preliminarily integrate the algorithm into the desktop title and observe that it likely constructs the optimal render pass segmentation for the current frame graph of the game. The graph consists of 171 nodes and uses 36 unique frame buffers on ultra graphics presets. By randomizing the node sorting order and recompiling the graph 10000 times, we observed the streak (render pass) count to vary from 54 to 58 with mean $\mu = 56.22$ and variance $\sigma^2 = 0.6$ when no render pass merging approach is used. As nodes are declared in an unspecified order due to being distributed between various modules, plugins and subsystems of an application, any of these results might occur in practice, which leads to undesirable non-deterministic performance. When the baseline approach is used, the graph is sorted to have 54 streaks independent of further order randomization, which alleviates non-deterministic performance concerns. But with the proposed approach, the streak count decreases to exactly 52 independent of randomization, which is explained by situations as shown on figure 2 being present in the graph. The fact that randomization could not lower the streak count to 52 throughout the 10000 iterations can be explained by the fact that the amount of correct topological orders for a graph behaves asymptotically as $O(n!)$ in general, mak-

ing the probability of the optimal order being selected at random increasingly small.

It must be noted that this frame graph has a reasonable amount of legacy code which is yet to be integrated with the frame graph runtime, and so results are expected to improve in the future as more knowledge about the renderer's structure and more node reordering opportunities are made available to the runtime.

The preliminary C++ implementation of the algorithm stores reachability info of the condensation graph in a bit matrix and heavily uses SIMD instructions to alleviate the high asymptotic complexity of the algorithm, which results in partitioning being executed in under 200 μ s on desktop PCs.

5 CONCLUSION

A seemingly simple problem of merging render passes appears to have much more depth to it than we initially anticipated. Implementing a two-pass approach of first partitioning the graph and only then sorting it enables saving almost a third of render pass breaks on synthetic tests, and hence tile cache flushes in practice. Moreover, although we initially started with render pass merging, we quickly found out that the rigorous problem statement through Λ -graphs naturally allows for grouping of barriers too, and is applicable not only to nodes that are parts of render passes, but to compute nodes too, as a contiguous streak of nodes without is a general enough to handle both.

Future Work

It is reasonable to think that the current version of the algorithm can be improved further. Time complexity can likely be improved without loss of quality, as well as the C++ implementation speed. Furthermore, it is reasonable to think that there are render pass breaks to be saved by using more advanced heuristics for choice of next node to be bucketed and choice of the bucket.

More work in the direction of handling multi-passes is to come, i.e. render passes that consist of multiple sub-passes. Technically, any two nodes that do rendering and do not require a barrier on any paths between them in P can be merged together into a single render pass with two subpasses, even if their frame buffers completely different, but it is not clear how such an approach would affect performance, so we suspect that heuristics represented mathematically as a label «closeness» metric will be required. Subpass attachment reads is another optimization that is crucial for mobile performance but is yet to be incorporated into our approach.

Finally, we are interested in applying profile guided optimization to render pass merging, as well as to render graph runtimes in general. Different execution orders may be optimal for different platforms, and so measuring GPU timings of node execution and then using

them as guides for a hybrid merging algorithm may lead to complete elimination of manual command list level cache optimizations from daily lives of rendering engineers, enabling development of applications at a higher abstraction level and increasing productivity.

6 REFERENCES

- [Adv] Inc. Advanced Micro Devices. *AMD Render Pipeline Shaders source code*. URL: <https://github.com/GPUOpen-LibrariesAndSDKs/RenderPipelineShaders>.
- [Arn17] Hans-Kristian Arntzen. *Render graphs and Vulkan — a deep dive*. 2017. URL: <https://themaister.net/blog/2017/08/15/render-graphs-and-vulkan-a-deep-dive/>.
- [Ban24] Darío Banini. *GPU synchronization in Godot 4.3 is getting a major upgrade*. 2024. URL: <https://godotengine.org/article/rendering-acyclic-graph/>.
- [Bav14] Louis Bavoil. In: (2014).
- [Cha23] Francois Durand Charlie Birtwistle. “Task Graph Renderer at Activision talk at REAC conference”. Rendering Engine Architecture Conference. 2023.
- [Epi] Inc. Epic Games. *Unreal Engine Render Dependency Graph*. URL: <https://docs.unrealengine.com/5.0/en-US/render-dependency-graph-in-unreal-engine/>.
- [GKM93] Ned Greene, Michael Kass, and Gavin Miller. “Hierarchical Z-buffer visibility”. In: *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*. 1993, pp. 231–238.
- [Kah62] A. B. Kahn. “Topological sorting of large networks”. In: *Commun. ACM* 5.11 (Nov. 1962), pp. 558–562.
- [ODo17] Yuriy O’Donnell. “FrameGraph: Extensible Rendering Architecture in Frostbite”. Game Developers Conference. 2017.
- [TAC21] Natalya Tatarchuk, Sebastian Aaltonen, and Timothy Cooper. “Unity Rendering Architecture”. SIGGRAPH 2021 REAC. 2021.
- [Tec] Unity Technologies. *Unity render graph system*. URL: <https://docs.unity3d.com/Packages/com.unity.render-pipelines.core@10.2/manual/render-graph-writing-a-render-pipeline.html>.
- [Wih19] Graham Wihlidal. “Halcyon: Rapid innovation using modern graphics”. Reboot Develop. 2019.

Exploitation of local adjacencies for parallel construction of a Reeb graph variant: cerebral vascular tree case

Charles LEPAIRE, Hakim BELHAOUARI

University of Poitiers
CNRS, XLIM,
I3M common lab,
Poitiers, France

charles.lepaire@univ-poitiers.fr
hakim.belhaouari@univ-poitiers.fr

Romain PASCUAL
Karlsruhe Institute of
Technology,
Karlsruhe, Germany

romain.pascual@kit.edu

Philippe MESEURE
University of Poitiers,
CNRS, XLIM,
Poitiers, France

philippe.meseure@xlim.fr

ABSTRACT

Strokes concerned more than 795,000 individuals annually in the United States as of 2021¹. Detecting thrombus (blood clot) is crucial for aiding surgeons in diagnosis, a process heavily reliant on 3D models reconstructed from medical imaging. While these models are very dense with information (many vertices, edges, faces in the mesh, and noise), extracting the critical data is essential to produce an accurate analysis to support the work of practitioners. Our research, conducted in collaboration with a consortium of surgeons, leverages generalized maps (g-maps) to compute quality criteria on the cerebral vascular tree. According to medical professionals, artifacts due to noise and thin topological changes are significant parameters among these criteria. These parameters can be determined via the Reeb graph, a topological descriptor commonly used in topological data analysis (TDA). In this article, we introduce a novel classification of saddle points, and a Reeb graph variant called the Local to Global Reeb graph (LGRG). We present parallel computation methods for critical points and LGRG, relying only on local information thanks to the homogeneity of the g-map formalism. We show that LGRG preserves the most subtle topological changes while simplifying the input into a graph formalism that respects the global structure of the mesh, allowing its use in future analyses.

Keywords

Topological data analysis ; Generalized maps ; Parallel computing ; Topology-based geometric modeling ; Reeb graph variant ; Critical points ; Cerebral vascular tree

1 INTRODUCTION

Topological data analysis (TDA) comes from mathematics, applying techniques from algebraic topology to analyze the shape and structure of data. It focuses on understanding the underlying geometric properties of data sets, such as their connectivity, holes, and loops, regardless of the specific metric used to represent the data. TDA can reveal essential features and patterns that may not be obvious through traditional statistical or geometric methods, proving particularly useful for analyzing complex, high-dimensional, or noisy data sets.

TDA leverages topological descriptors, retrieving specific information from the data. Most of these descriptors rely on the critical points of a function of interest over the data, i.e., the points where the function's derivative is null or undefined. Therefore, computing and classifying the critical points of a function is a crucial subroutine for many TDA tasks. For instance, a Reeb graph consists of nodes corresponding to the crit-

ical points and arcs describing the topological modifications between these points.

TDA being agnostic to the application domain, it has been successfully applied across various areas, ranging from medicine [16, 19, 26] to musicology [2] or chemistry [20]. Recently, TDA has been used to analyze 3D medical imaging for assisting practitioners [29, 27], e.g., for discovery in preclinical spinal cord injury and traumatic brain injury [19]. In particular, the Reeb graph has been used for 3D micro-vascular modeling [39] with non-invasive imaging modality to analyze important retina-associated diseases.

As an answer to the increasing data size, recent approaches propose parallel and massively parallel implementations of algorithms to build topological descriptors [28, 30]. Moreover, the complexity of data also presents challenges related to dimension and noise.

Objectives. We aim for a massively parallel implementation of a Reeb graph construction algorithm that preserves the fine-grained topological information of the data.

¹ cf. CDC facts: <https://www.cdc.gov/stroke/facts.htm>

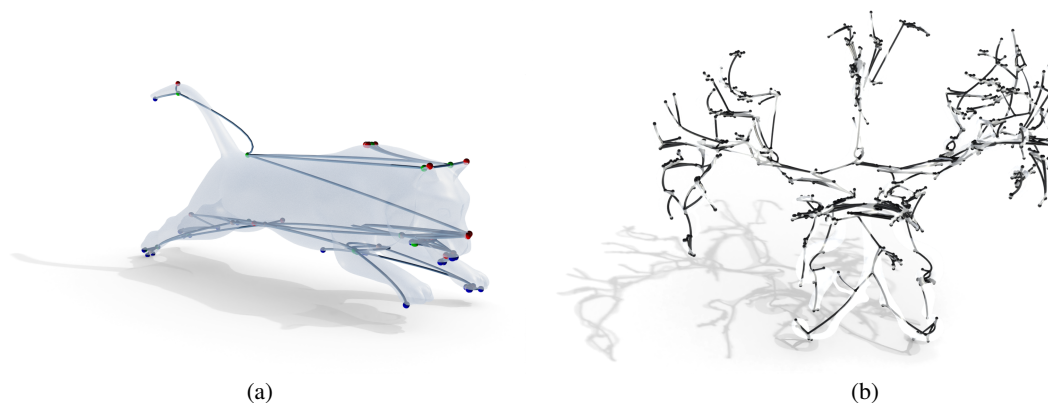


Figure 1: Local to Global Reeb Graph of a tiger (a), and a cerebral vascular tree (b).

Contributions. We propose to compute a Reeb graph variant called the Local to Global Reeb Graph (LGRG) via a local information diffusion algorithm (examples are given in Figure 1). Our LGRG construction algorithm results in three main contributions.

- We present a refined classification of critical points.
- We offer an extended Reeb graph, which can be computed using only local information provided by a topological description of the structure, without requiring a simplicial complex but only a mesh.
- We propose a massively parallel implementation of an algorithm computing LGRG relying on the generalized maps data structure [10].

A key idea of our approach is to exploit topological proximity rather than geometric one when building the graph's edge. We obtain a Reeb graph variant that also considers degenerate critical points (more precisely, degenerate saddle points), offering a robust and comprehensive solution for handling complex structures. To run in parallel, our algorithm for computing LGRG requires non-concurrent access to local topological information, namely, the edges and face corners around a vertex, which are natively available within generalized maps.

Paper organization. The paper is organized as follows. Section 2 reviews the various approaches to build (variants of) Reeb graphs and the existing parallelized algorithms. Section 3 recalls the generalized map data structure and the islet algorithm used for parallelization. Section 4 presents our refined classification of the critical points, while Section 5 describes our Local to Global Reeb graph. Section 6 is dedicated to our experimental results and comparison with other tools, with emphasis on cerebral vascular trees. Finally, Section 7 gives concluding remarks and discusses possible extensions of our work.

2 RELATED WORK

Given a Morse function [18] $f : M \rightarrow \mathbb{R}$ on a manifold M with distinct critical points, the Reeb graph [25]

is made of nodes associated with the critical points of the function and arcs where the level sets ($f^{-1}(c)$, for some c in \mathbb{R}) retain the same connectivity. Using discrete Morse function in [5], Reeb graphs can be studied for discrete structures [23, 8, 33, 12, 24, 35].

Most approaches that compute Reeb graphs assume that data are represented as a simplicial complex [12, 33] since it simplifies the algorithmic formulation of several standard computations in algebraic topology, such as homology groups. To this end, standard approaches usually triangulate manifolds before computations. For instance, a polyhedron would be decomposed into tetrahedra as a preprocessing step. We propose to work directly on a manifold mesh representation without this preprocessing step.

Discrete Reeb graphs proved fruitful across several areas of scientific visualization [13], such as shape understanding [4], segmentation [38, 15], feature detection [32], or data surface simplification [8]. Within medical applications, Makram et al. [17] used Reeb graphs to automatically locate cephalometric landmarks, Sun et al. [31] applied Reeb graphs to capture the clustering structure of brain white matter nerve fibers. Here, we leverage the topological information provided by Reeb graphs to analyze surface meshes of cerebral vascular trees generated by MRI image reconstruction.

Various generalizations of Reeb graphs have been proposed. The Extended Reeb Graph (ERG) was introduced in [7] as a conceptual model for surface representation. The ERG generalizes Reeb graphs by also considering degenerate critical points, a recurrent problem in discrete surface models while respecting the semantic representation of the Reeb graph. The Augmented Reeb Graph (ARG) adds additional information to the basic structure, such as shape, size, color, or texture. It improves the Reeb graph's accuracy and relevance in understanding complex data [34, 14]. Our Reeb graph variant is similar to the ERG but only accepts degenerated saddle points. Compared to the ARG, we do not

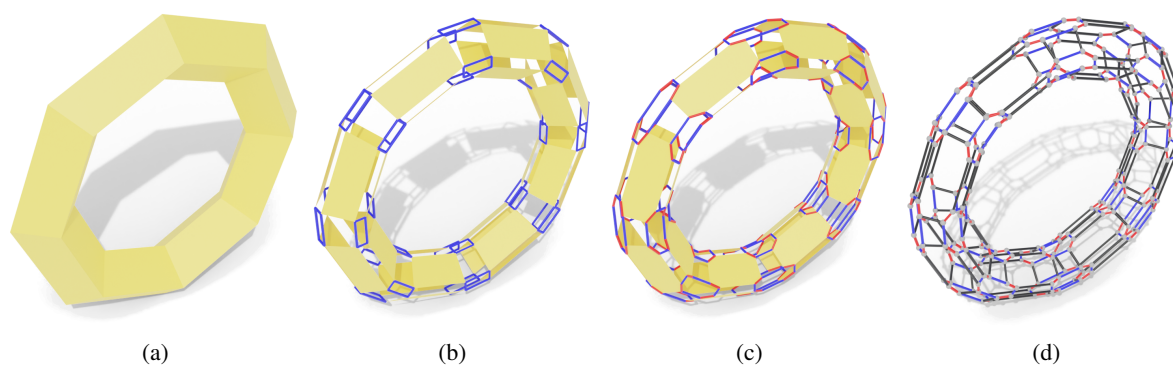


Figure 2: Recursive decomposition of an object to obtain its g-map representation: (a) geometric object, (b) face split (α_2), (c) edge split (α_1), (d) vertex split (α_0) and final graph.

store information on the Reeb graph but consider it built embedded into the mesh. Thus, information can be directly retrieved from the object.

Advances in medical imaging enabled the acquisition of higher quality, multimodal MRI images, resulting in larger, more complex yet more detailed and fine-grained images. A direct consequence is an increase in computation times needed to build Reeb graphs which resulted from parallel implementations for algorithms computing Reeb graphs. Hajij and Rosen [15] proposed one of the first parallel algorithms for Reeb graphs based on Doraiswamy and Natarajan's algorithm [11]. Their method partitions the manifold into submanifolds, computes Reeb graphs on the submanifolds, and glues the different Reeb graphs to obtain the complete graph. In particular, the partition stage assumes that the entire object is split into connected parts with approximately as many vertices in each part, without further consideration of the partitioning. Gueunet et al. [14] proposed a parallel algorithm for computing an ERG based on the optimal algorithm of Parsa [21], i.e., scanning the mesh along a given direction while maintaining the subcomponents of the level sets, incrementally building the Reeb graph. Guenet et al.'s algorithm scans the mesh in two directions but requires a post-processing step. These (variants of) Reeb graph constructions highly filter the input mesh, resulting in a graph where details are lost. This lost information can be essential for determining the object's structure with a more accurate representation. Our approach exploits local information of the vertices to categorize the critical points and then construct a graph describing the data's topology. Both parts run in parallel without filtering the input.

3 GENERALIZED MAP

This section presents the formalism of generalized maps (g-maps). In brief, a g-map is a compact view of the simplicial set formalism [1].

3.1 Topological model

Generalized maps [10] correspond to a generalization of edge-based models [37] in any dimension and belong to the class of combinatorial models used in topology-based geometric modeling. In this approach, the topology of an object is described as a cellular subdivision. Intuitively, an object's representation can be recovered by recursively splitting its topological cells by decreasing dimension. For instance, Figure 2 shows the decomposition of a low-poly torus in 2D. From the initial object (Fig. 2a), we separate the adjacent faces, which are linked by blue arcs depicting the α_2 -links, i.e., adjacency relations along dimension 2 (Fig. 2b). This process is iterated by splitting edges within faces via α_1 -links for dimension 1 (Fig. 2c) and finally the vertices via α_0 -links for dimension 0 (Fig. 2d). This last subdivision yields the darts of the g-map which we view as nodes of a graph whose arcs are the links obtained in the decomposition.

Formally, g-maps can be defined as undirected graphs labeled on arcs by dimensions and subject to additional constraints [22]. In this article, we consider g-maps of dimension 2, or 2-g-maps that we will simply call g-maps. We summarize here the properties that suffice to appreciate the content of our contributions. (1) Any dart admits a unique incident α_i -link for each $i \in \{0, 1, 2\}$. (2) When two darts share an α_i -link (for $i \in \{0, 1, 2\}$), they belong to the same k -cells for $k \neq i$, but to distinct i -cells. (3) G-maps represent orientable and non-orientable quasi-manifolds.

G-maps provide a combinatorial description of the object's decomposition into topological cells. These cells correspond to subgraphs induced by dimensions, called orbits. An orbit consists of all the darts reachable from an initial dart through a subset of all possible dimensions. For instance, the orbit on the dimensions 1 and 2, called a $\langle \alpha_1, \alpha_2 \rangle$ -orbit, retrieves darts by following links between cells of dimensions 1 and 2, thus always within the same cell of dimension 0. In other words, an $\langle \alpha_1, \alpha_2 \rangle$ -orbit encodes a vertex. Adding geometric positions to vertices then means that all darts within an

$\langle \alpha_1, \alpha_2 \rangle$ -orbit share the same position value. Such information is called embedding [3], which provides geometric information to the topological cells. This information is stored on the vertices of the graph, extending the topological representation of g-maps. In this article, we will manipulate 3D positions attached to the vertices and orientation boolean attached to the empty orbit, i.e., directly to the darts.

Working with an orientation boolean attached to each dart means that the represented object is necessarily orientable. In this case, whenever a dart has the value `true`, its α_i -neighbors (for $i \in \{0, 1, 2\}$) have `false`, and vice-versa. This property can be seen in Figure 3a, where the `true` darts are drawn in pink and the `false` darts in black. This property can be used to prevent concurrent writing in parallel algorithms. Indeed, if only `true` darts write information, no dart can write at the same time as any of its neighbors. Additionally, we use the orientation boolean to orient vertices (see Section 4.1).

3.2 Refresher on the islet algorithm

The islet algorithm [9] looks like the leader election algorithm of distributed systems over networks by choosing a dart within an $\langle o \rangle$ -orbit. It spreads information on all α_i -links given by $\langle o \rangle$, iterating once over each dimension before starting another step. It was used to partition orbits to speed up the computation time of specific topological transformations, namely the augment and update operations. As shown in [9], the algorithm can efficiently be parallelized.

In Figure 3, we illustrate the islet algorithm to find a representative dart within each vertex. We choose the representative dart as the one with the minimal ID (which is an integer). Since vertices correspond to $\langle \alpha_1, \alpha_2 \rangle$ -orbits, the values are alternately propagated on α_1 -links and α_2 -links. The object used is a subdivided square, whose g-map representation is given in Figure 3a, along with the darts' ID. Figure 3b shows the propagation along the α_1 -links and Figure 3c along the α_2 -links. For example, the ID 4 in Figure 3a has been propagated to three new darts in Figure 3c, now covering the two adjacent faces. Some darts received new values at each propagation step, meaning that the process should still be iterated. The algorithm stops when no dart changes its value, which necessarily happens as each orbit contains only finitely many darts. The result of the algorithm is given in Figure 3d, where darts in each $\langle \alpha_1, \alpha_2 \rangle$ -orbit share the minimum ID of a dart within the vertex.

4 CLASSIFICATION OF THE CRITICAL POINTS

This section presents our refined categorization of critical points, how to compute them, and how to exploit the

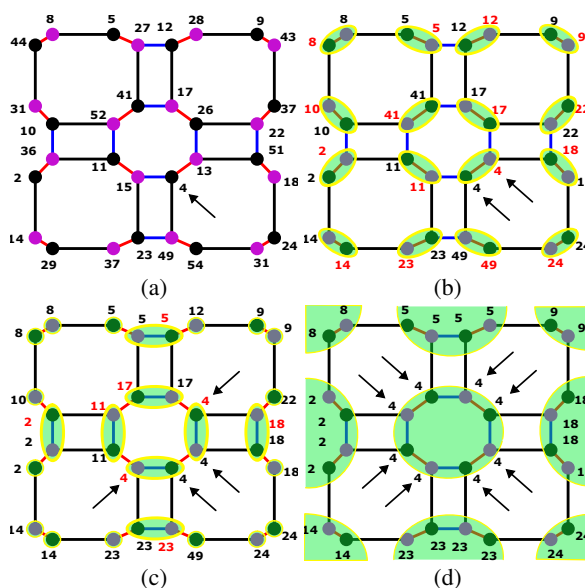


Figure 3: Steps of the islet algorithm 3.2 on the orbit $\langle \alpha_1, \alpha_2 \rangle$. (a) The initial g-map with values associated with the darts. (b) diffusion of minimal value inside the islet produced by α_1 -links. (c) diffusion inside the islet from α_2 -links. (d) Final state where the minimal value has been diffused for the whole orbit.

topological structure to obtain a parallel computation. Note that our computation method is axis-dependent (like all standard methods).

4.1 Local elevations

Assuming an oriented data structure to encode a mesh, we propose to classify the critical points of the mesh based on *elevation configurations* of the topological neighborhood of vertices. To compute the vertices elevation configurations, we first compute edge elevations, which yield face corner elevations along consecutive edges within faces, from which we deduce the vertex elevation configuration by aggregating the elevations of all its incident face corners. We now detail each step of the computation, assumed to be computed along a given axis \mathbf{u} for a given vertex v .

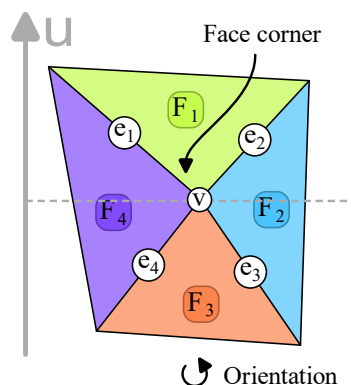


Figure 4: Computing the elevations.

Edge elevation Each edge e incident to v computes an *edge elevation* relative to v by comparing the position of v and that of its other endpoint v' , depending on \mathbf{u} . Formally, the edge elevation is the sign of the dot product $\mathbf{vv}' \cdot \mathbf{u}$. Intuitively, a positive edge elevation describes an edge in the same direction as the selected axis, i.e., v' is 'above' v along \mathbf{u} . In Figure 4, the edge e_1 has a positive edge relative to v . Note that the edge elevation might be null. Besides, the edge elevation is relative to the vertex v , and the edge elevation of e relative to v' is the opposite of the one relative to v .

Face corner elevation A face corner is a region delimited by two consecutive edges incident to the same vertex in a given face. Note that the orientation of the object naturally orients the face corners. In Figure 4, the face F_1 together with the edges e_1 and e_2 define one of the four face corners of the vertex v . The *face corner elevation* is the oriented pair of edge elevations for the two edges defining the face corner. If both edges have a positive or null edge elevation, the face corner has a HIGH elevation. Similarly, if both edges have a negative or null edge elevation, the face corner has a LOW elevation. Note that, if both edges have a null edge elevation, the face corner is FLAT. For instance, (F_3, e_3, e_4) has a LOW face corner elevation in Figure 4, while (F_1, e_1, e_2) has a HIGH one. The two remaining cases correspond to non-null edge elevations with opposite signs. The face corner elevation then depends on the orientation. If the first edge has a negative elevation and the second a positive one, the face corner elevation is UP. Reversely, a positive elevation followed by a negative one yields a DOWN face corner elevation. In Figure 4, (F_4, e_4, e_1) has a UP face corner elevation and (F_2, e_2, e_3) a DOWN one.

Vertex elevation sequence Finally, we derive a *vertex elevation sequence* for v as an ordered sequence of face corner elevations around a vertex. The sequence is obtained from the natural ordering induced by the object's orientation. This sequence is defined as up to one circular permutation. For instance, one vertex elevation sequence for v in Figure 4 is given by the sequence of face corner elevations of (F_1, e_1, e_2) , (F_2, e_2, e_3) , (F_3, e_3, e_4) , (F_4, e_4, e_1) , i.e., HIGH, DOWN, LOW, UP. The vertex elevation sequence is then used to classify the vertex and find the critical points. Note that, if one of the face corner elevations is FLAT, it is not considered when calculating the elevation sequence.

4.2 Classification

We propose to retrieve and classify a mesh's critical points based on the elevation sequence of its vertices. We distinguish four types of critical points. In contrast with the usual maxima, minima, and saddle points, we further differentiate between saddles of birth and death.

A (local) *minimum* is a vertex with only HIGHS in its elevation sequence, e.g., Figure 5a. A (local) *maximum* is a vertex with only LOWs in its elevation sequence, e.g., Figure 5b. A *saddle* is a vertex with at least two, by definition non-consecutive, UPs or DOWNs. The difference between saddles of birth and saddles of death comes from the direction of the vertex normal. The saddle is a *saddle of birth* if its normal aligns with \mathbf{u} (positive dot product) and a *saddle of death* otherwise. Concretely, a saddle of birth separates the inside of an object below and the outside above along the axis \mathbf{u} . Examples of saddles of birth and death are respectively given in Figure 5c and in Figure 5d. Any other elevation sequence corresponds to a *regular*, i.e., non-critical, point.

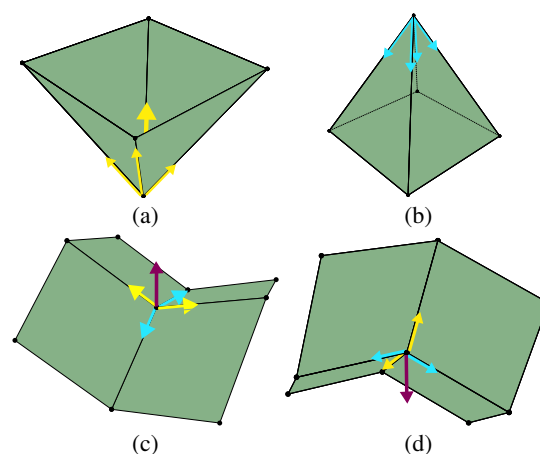


Figure 5: Classification of critical points (a) Local minimum. (b) Local maximum. (c) Saddle of birth. (d) Saddle of death.

Our classification represents two minor contributions. First, the classification only relies on topologically local information around a vertex and can be computed without first triangulating the manifold. Second, we further distinguish between the saddle points using only local information. A watchful reader will notice that standard approaches to Reeb graph computation build the classification of critical points essentially from all edge elevations around a vertex. We introduced the intermediate step of face corner elevations as an additional speed-up when extracting critical points in parallel. We now detail this parallel computation.

4.3 Parallelization

The three steps from Section 4.1 for the detection and classification of critical points only require local information around the vertices and can therefore easily be parallelized.

Edge elevation Within the g-map data structure, an edge e and a reference vertex v are both encoded within a single dart, while the α_0 -link allows accessing the other endpoint of the edge (in constant time). While two α_2 -linked darts encode a given edge e and a given vertex

v (they correspond to the two faces sharing the edge), only one dart has `true` as the orientation boolean. Since its α_0 -neighbor also has `false` for orientation boolean, the edge elevation can be computed by comparing the position of each `true`-oriented dart with its α_0 -neighbor. The subroutine can be realized simultaneously on all `true`-oriented darts without concurrent read or write, i.e., in parallel.

Face corner elevation A face corner corresponds to a $\langle \alpha_1 \rangle$ -orbit in a g-map and thus to two darts with different orientation boolean values. Thus, the face corner elevation is obtained by running a parallel for loop on all darts oriented `true` and comparing the dart's edge elevation with that of its α_1 -neighbor. The four cases, HIGH, LOW, UP, and DOWN are respectively illustrated in Figures 6a, 6b, 6c, and 6d.

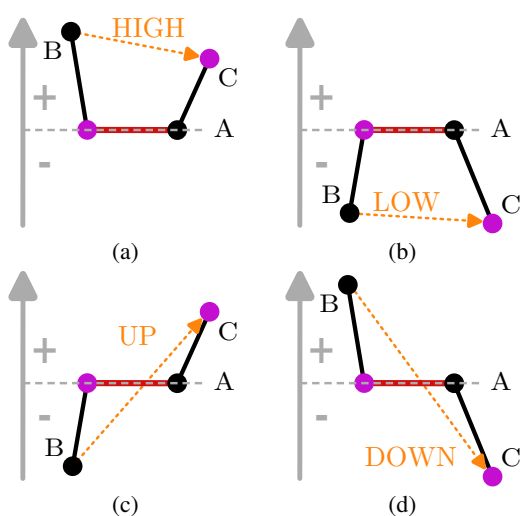


Figure 6: The different face corner elevations. (a) HIGH: Both edges have a positive elevation (B and C are “higher” than A). (b) LOW: Both edges have a negative elevation (B and C are “lower” than A). (c) UP: The dart from the orbit of A with a `true` orientation belongs to an edge with a negative elevation, and the other edge has a positive elevation. B is lower than A , which is lower than C . (d) DOWN: The dart from the orbit of A with a `true` orientation value belongs to an edge with a positive elevation, and the other edge has a negative elevation. (B is higher than A , which is higher than C).

Vertex elevation sequence Retrieving the vertex elevation sequence requires to elect one representative dart per vertex, i.e., per $\langle \alpha_1, \alpha_2 \rangle$ -orbit. Choosing such a dart can be performed in parallel via the islet algorithm. By starting with the representative dart of a vertex, its elevation sequence is obtained by cycling through the $\alpha_2 \circ \alpha_1$ -links around it and retrieving the associated face corner elevation. This task can again be performed in parallel within each vertex. In Figure 7, starting from any pink dart, we cycle through the

other three pink darts via $\alpha_2 \circ \alpha_1$ -links to obtain either UP, DOWN, UP, DOWN or DOWN, UP, DOWN, UP. Finally, each aggregated sequence is analyzed to classify the vertex.

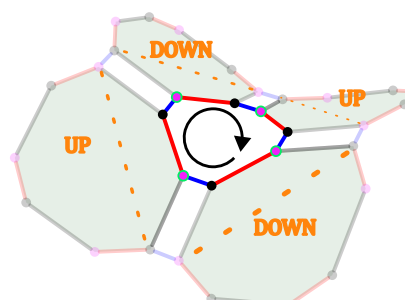


Figure 7: Elevation sequence around a vertex.

5 LOCAL TO GLOBAL REEB GRAPH (LGRG)

Once the critical points have been identified, we use a propagation algorithm that spreads information, similar to a gossip algorithm for distributed systems. The motivation is to obtain a partition of the mesh into sub-components created and ended by critical points. We then build our LGRG by connecting critical points (which are the nodes of the graph) with arcs encoding how the sub-components link these critical points. Note that a sub-component may correspond to several arcs in our Reeb graph variant.

Before detailing the propagation algorithm and the construction of the LGRG, we provide some insights via the example of Figure 8. The shown torus contains four critical points: a maximum in red, a minimum in blue, a saddle of birth in light pink, and a saddle of death in green (the elevations being computed along the vertical axis). These critical points split the torus vertices into four sub-components: green, red, purple, and yellow. The yellow sub-component corresponds to the propagation area of the blue minimum, ended by the pink saddle of birth, giving rise to the arc between the bottom two nodes in the graph of Figure 8b. For this specific object, each sub-component in Figure 8a results in an arc in Figure 8b. Note that each node is associated with a critical point drawn with the same color for ease of visualization.

In the sequel, we use the terms “above”, “below”, and “topmost” to be understood along \mathbf{u} , i.e., as the sign of the dot product with \mathbf{u} .

5.1 Propagation and sub-components

The motivation for a propagation algorithm is to obtain a construction fully given by the description of a behavior for the mesh vertices. Each vertex propagates some information related to a given critical point. Intuitively, the vertex spreads information describing where the

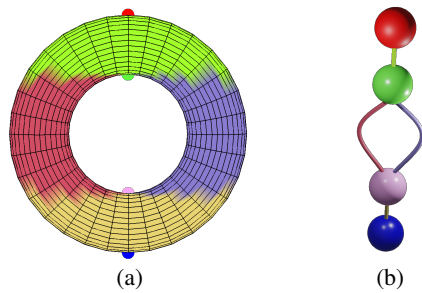


Figure 8: Area propagation on a mesh of a torus (a) the mesh and (b) the resulting graph, our Reeb graph variant, based on area adjacency.

last change in the connectivity of the level sets comes from. In practice, this corresponds to the topmost critical points located below the vertex propagating this information. Since the computation on vertices exploits locally retrieved information, it runs in parallel.

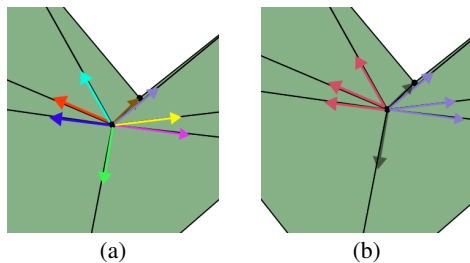


Figure 9: On the left (a), a saddle of birth and its incident edges elevation generates two sub-components. On the right (b), the same saddle of birth with the elevation of its incident edges grouped by equivalent edges (using the same color) shows the beginning of red and purple sub-components.

The information propagated and the propagation used naturally follows from the semantics associated with the (non-)critical vertices of the mesh. Let us recall the meaning of vertex classes concerning the changes to the local connectivity of the level sets to clarify the propagated labels. A regular point does not change the local connectivity; therefore, it initially does not own any label. A maximum ends a level set, meaning it initially owns no label and does not propagate any. A minimum creates a new level set, thus initially owning a unique label spread upwards. Similarly, a saddle of death merges several level sets into one, so it initially owns a unique label spread upwards. The more complex case concerns the saddles of birth. Such critical points split a level set into several. Therefore, it initially needs to own one label per created level set. For instance, the saddle of birth given in Figure 9a is incident to several edges with positive elevation. We cluster them into connected sequences of identical edge elevation around the saddle, as shown in Figure 9b. Thus, we create one label per

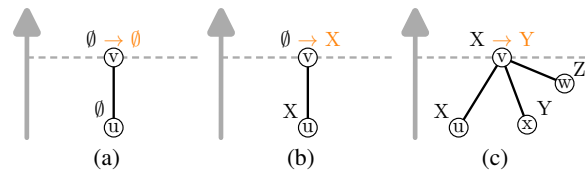


Figure 10: Diffusion pattern of a vertex v . (a) v is unlabelled and does not receive any label: nothing occurs. (b) v is unlabelled and receives some label X : v updates its label to X . (c) v is already labeled by X and receives the labels Y and Z from neighbors: it compares X , Y and Z to find that Y is the highest, so v updates its label to Y .

group of edges with positive elevation. All edges within a group initially propagate its label.

The partitioning algorithm alternates the propagation of labels along the edges with positive elevation and the update of vertex labels. When receiving labels, a vertex always keeps the topmost received label (note that only regular vertices follow this process). The various possible patterns are given in Figure 10. Note that the values of interest correspond to critical points. The algorithm stops when no vertex updates its label. A *sub-component* corresponds to the set of vertices sharing the same label when the algorithm stops. It corresponds to all regular vertices immediately higher than the critical point that gave them its label. By construction, a sub-component is delimited by one critical point below and at least one but possibly several critical points above.

The propagation in denser parts of the mesh is naturally slower than in sparser parts. Thus, some vertices may be first labeled by a critical point, which does not correspond to their final sub-component. Indeed, the algorithm later overwrites the label of these vertices (and thus of the whole area it propagated to) with one from a higher critical point, maybe even several times, until the highest one is found. In practice, an area propagates from a critical point upwards over a mesh until it can no longer propagate because the vertices of its boundary are either connected by edges with a negative elevation or connected by edges with a positive elevation to vertices labeled by a higher critical point (including critical points themselves).

The g-map data structure simplifies the algorithm as labels can be stored on darts and propagated along α_0 -arcs. Reasoning on darts essentially means that (a) vertices corresponding to saddles of birth store several labels among the darts of their $\langle \alpha_1, \alpha_2 \rangle$ -orbit, and (b) vertex update means choosing the topmost label within the $\langle \alpha_1, \alpha_2 \rangle$ -orbit of regular vertices. The vertex label is updated via the islet algorithm (see Section 3.2), propagating labels instead of dart IDs.

5.2 Local to Global Reeb graph construction

The propagation algorithm terminates when no vertex updates its label, constructing a mesh partition into sub-components. We then build the LGRG from the adjacency relationships between the sub-components. If an edge links two vertices in the mesh belonging to distinct sub-components, we add an arc to the LGRG between the associated critical points. Note that only one arc is added to the LGRG even if several edges in the mesh link vertices between the same two sub-components. In the propagation algorithm, saddles of birth create several sub-components (identified by a group of consecutive edges with positive elevation). Since these sub-components arise from the same critical point, the induced arcs in the LGRG are all linked to the same node corresponding to the saddle of birth. This construction also justifies propagating several identifiers from a saddle of birth. Indeed, had we propagated only one identifier, some arcs would have been missing in the LGRG. For instance, the two arcs between the saddle of birth and the saddle of death in the torus of Figure 8b would have been merged whereas they should remain distinct.

Once again, this step can be parallelized by checking all edges simultaneously and eliminating redundant pairs of sub-components. The islet algorithm directly enables this computation by identifying a dart within each edge $(\langle \alpha_0, \alpha_2 \rangle)$ -orbit and comparing its label with the one of its α_0 -neighbor.

6 RESULTS

Experiments were conducted on an 11th Gen Intel(R) Core(TM) i7-11850H @2.50GHz with 8 hyper-threaded cores (16 simultaneous threads) and 32GB of RAM. We used the implementation of g-maps provided by Jerboa [6] and Java streams (with JDK 17) for parallelization. We tested our method using the height function as a scalar field on standard geometric processing objects, e.g., a tiger, a dragon, etc. (see Figure 12), and medical vascular trees (see Figure 13) obtained via the marching cubes algorithm on the MRI images provided by the MICCAI CROWN challenge [36].

Table 1 provides statistics about the objects in our dataset and the critical points found by the three methods. Our approach finds a distribution of critical points on the standard objects similar to TTK, while ReCon produces many additional critical points. The example of the pegasus highlights that TTK filters the object and misses a lot of details. We find nearly 5 times more critical points on this object. These additional critical points come from the base and the wings, as shown in Figure 12e. Although it may not be relevant for encoding the topology of the pegasus, such a fine-grained description of the topological changes

corresponds to the data of interest for the medical staff in our consortium. In the second half of the table, statistics are given on twelve cerebrovascular trees taken from various patients. Recon still finds many more saddle points than our method and TTK. However, we detect more minima and maxima, highlighting that we better capture subtle topological variations filtered out by the other tools.

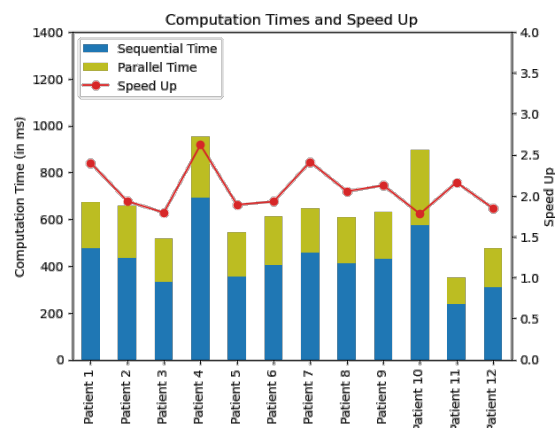


Figure 11: Comparison between sequential and parallel runtimes for LGRG on cerebral trees.

We also compared sequential and parallel executions of our method on the cerebral trees, i.e., where both the critical points classification and LGRG construction are performed as discussed in Sections 4 and 5. Computation times are given in Figure 11 and show an average speed up by a factor 2.08.

7 CONCLUSION

We proposed a novel approach for categorizing critical points by exploiting local topological information at each vertex and a variant of the Reeb graph built from a topology-induced information propagation from the critical points over the mesh. We then leveraged this local approach to provide a parallel algorithm for characterizing critical points and the computation of LGRG on the g-map data structure, which we implemented in Jerboa. In particular, using topological information reduces the dependency on geometric features, thus avoiding excessive filtering and maintaining data integrity. We plan to apply our LGRG construction for medical analysis to assist in the diagnosis of strokes. As of now, we also hope to understand better the topological features that need to be detected for the diagnosis with the help of the medical staff on the one hand and by exploiting machine learning approaches on our LGRG. For instance, apart from the circle of Willis, the existence of a topological cycle in the cerebral vascular tree is an indicator of a stroke. Still, whether this criterion is sufficient or even necessary remains unclear. Out of the medical field, our new LGRG also opens the way for

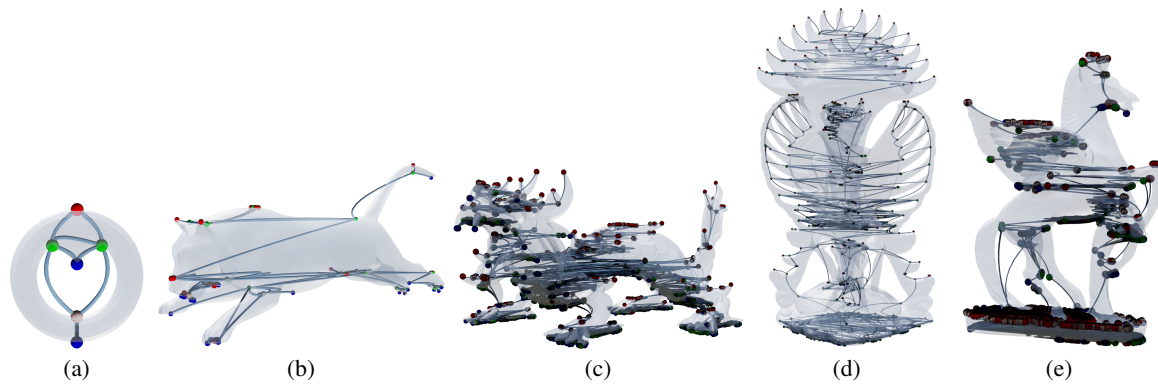


Figure 12: LGRG computed on a torus ((a), a tiger (b), a dragon called "XYZ RGB" (c), Garuda and Vishnu (d), and a pegasus (e).

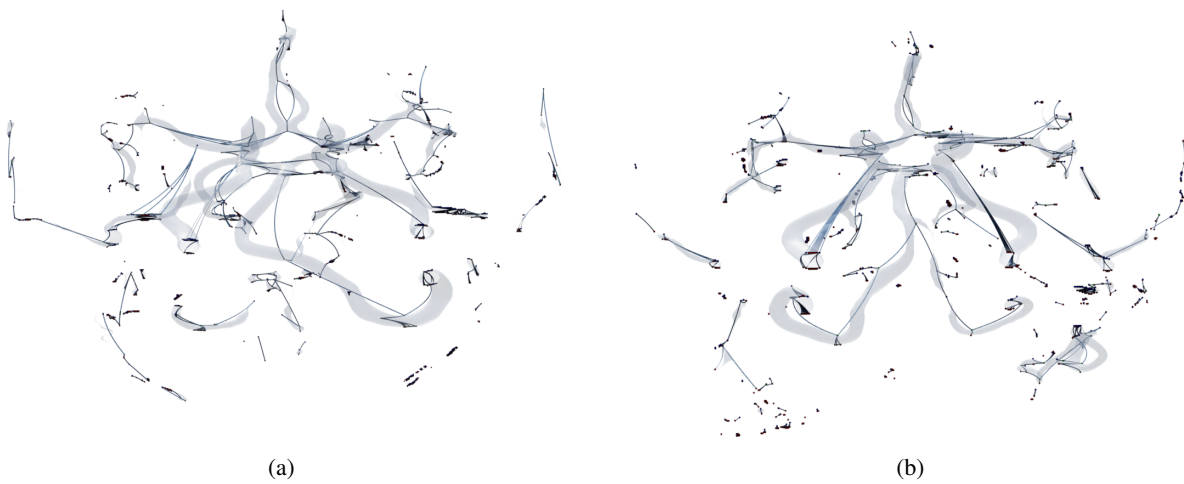


Figure 13: LGRG computed on the cerebral vascular trees of patient 1 (a), and patient 2 (b).

Object	#faces (*1000)	#vert (*1000)	Minimum			Maximum			Saddles		
			TTK	ReCon	Ours	TTK	ReCon	Ours	TTK	ReCon	Ours
Torus	1.1	0.6	2	2	2	1	1	1	3	3	3
Tiger	116.7	58.4	26	191	29	16	230	23	40	419	30
XYZ RGB	199.8	99.9	648	7,768	659	519	14,330	508	1,102	20,875	1,168
G. & V.	222.9	111.4	120	3,903	120	120	7,814	121	307	11,576	308
Pegasus	667.5	333.7	287	3,169	1,164	229	6,605	3,556	517	9,723	584
Patient 1	129.4	65.7	198	385	509	210	685	1246	261	926	167
Patient 2	123.9	62.7	239	-	367	249	-	1,086	263	-	166
Patient 3	100.8	50.9	122	307	272	126	475	624	154	695	114
Patient 4	152.3	77.1	308	469	653	307	807	1,667	458	1127	310
Patient 5	106.3	54.0	155	418	430	163	733	990	194	1023	133
Patient 6	125.7	63.7	166	400	396	192	609	949	238	895	171
Patient 7	137.2	69.5	271	470	448	298	722	1,367	329	971	230
Patient 8	117.0	59.5	235	376	500	256	778	1,386	272	927	170
Patient 9	133.8	67.8	204	477	511	227	767	1,089	280	1097	189
Patient 10	160.0	80.9	212	490	462	231	778	908	279	1118	200
Patient 11	82.1	41.5	130	264	273	127	439	692	159	604	101
Patient 12	101.8	51.6	177	340	419	190	582	1,023	241	803	173

Table 1: Comparison of the critical points computed with TTK, ReCon, and our method.

locally induced mesh segmentation and data compression. Furthermore, exploring a C++/CUDA implementation holds promise for accelerating our parallelization efforts, potentially leading to significant runtime improvements.

8 ACKNOWLEDGMENT

This research work is funded by Region Nouvelle-Aquitaine, France (AAP ESR 2021 Program - ARTERIA Project, AAPR2021A-2021-12189710) and SIEMENS Healthineers.

REFERENCES

- [1] S. Alayrangues, S. Peltier, G. Damiand, and P. Lienhardt. "Border operator for generalized maps". In: *DGCI*. Vol. 5810. Montreal, Canada: Springer, 2009, pp. 300–312. DOI: 10.1007/978-3-642-04397-0_26.
- [2] M. Andreatta. "Méthodes algébriques en musique et musicologie du XXe siècle : aspects théoriques, analytiques et compositionnels". PhD Dissertation. Paris, EHES, 2003.
- [3] A. Arnould, H. Belhaouari, T. Bellet, P. Le Gall, and R. Pascual. "Preserving consistency in geometric modeling with graph transformations". In: *MSCS* 32.3 (2022), pp. 300–347. DOI: 10.1017/S0960129522000226.
- [4] M. Attene, S. Biasotti, and M. Spagnuolo. "Shape understanding by contour-driven retiling". In: *Vis. Comput.* 19.2 (2003), pp. 127–138. DOI: 10.1007/s00371-002-0182-y.
- [5] T. F. Banchoff. "Critical Points and Curvature for Embedded Polyhedral Surfaces". In: *Am. Math. Mon.* 77.5 (1970), p. 475. DOI: 10.2307/2317380.
- [6] H. Belhaouari, A. Arnould, P. Le Gall, and T. Bellet. "Jerboa: A Graph Transformation Library for Topology-Based Geometric Modeling". In: *ICGT*. LNCS. 2014, pp. 269–284. DOI: 10.1007/978-3-319-09108-2_18.
- [7] S. Biasotti, B. Falcidieno, and M. Spagnuolo. "Extended Reeb Graphs for Surface Understanding and Description". In: *DGCI*. LNCS. 2000, pp. 185–197. DOI: 10.1007/3-540-44438-6_16.
- [8] S. Biasotti, D. Giorgi, M. Spagnuolo, and B. Falcidieno. "Reeb graphs for shape analysis and applications". In: *TCS. Computational Algebraic Geometry and Applications* 392.1 (2008), pp. 5–22. DOI: 10.1016/j.tcs.2007.10.018.
- [9] P. Bourquat, H. Belhaouari, P. Meseure, V. Gauthier, and A. Arnould. "Transparent Parallelization of Enrichment Operations in Geometric Modeling". In: *VISI-GRAPP/GRAPP*. 2020, pp. 125–136. DOI: 10.5220/0008965701250136.
- [10] G. Damiand and P. Lienhardt. *Combinatorial Maps: Efficient Data Structures for Computer Graphics and Image Processing*. CRC Press, 2014. ISBN: 978-1-4822-0652-4.
- [11] H. Doraiswamy and V. Natarajan. "Efficient algorithms for computing Reeb graphs". In: *Comput. Geom.* 42.6 (2009), pp. 606–616. DOI: 10.1016/j.comgeo.2008.12.003.
- [12] H. Doraiswamy and V. Natarajan. "Computing Reeb Graphs as a Union of Contour Trees". In: *TVCG* 19.2 (2013), pp. 249–262. DOI: 10.1109/TVCG.2012.115.
- [13] F. Escolano, E. R. Hancock, and S. Biasotti. "Complexity Fusion for Indexing Reeb Diagrams". In: *Computer Analysis of Images and Patterns*. LNCS. 2013, pp. 120–127. DOI: 10.1007/978-3-642-40261-6_14.
- [14] C. Gueunet, P. Fortin, J. Jomier, and J. Tierny. "Task-based Augmented Reeb Graphs with Dynamic ST-Trees". In: *EGPGV19*. 2019. DOI: 10.2312/pgv.20191107.
- [15] M. Hajij and P. Rosen. *An Efficient Data Retrieval Parallel Reeb Graph Algorithm*. 2020. DOI: 10.48550/arXiv.1810.08310.
- [16] C. F. Loughrey, P. Fitzpatrick, N. Orr, and A. Jurek-Loughrey. "The topology of data: opportunities for cancer research". In: *Bioinformatics* 37.19 (2021), pp. 3091–3098. DOI: 10.1093/bioinformatics/btab553.
- [17] M. Makram and H. Kamel. "Reeb Graph for Automatic 3 D Cephalometry". In: *Int. J. of Image Processing* 8.2 (2014).
- [18] J. W. Milnor, M. Spivak, and R. Wells. *Morse theory*. Ann. Math. Stud. 51. PUP, 1963.
- [19] J. L. Nielson et al. "Topological data analysis for discovery in preclinical spinal cord injury and traumatic brain injury". In: *Nat. Commun.* 6.1 (2015), p. 8581. DOI: 10.1038/ncomms9581.
- [20] M. Olejniczak, P. Severo, A. Gomes, and J. Tierny. "A Topological Data Analysis Perspective on Non-Covalent Interactions in Relativistic Calculations". In: *Int. J. of Quantum Chemistry* e26133 (2019). DOI: 10.1002/qua.26133.

- [21] S. Parsa. “A deterministic $O(m \log m)$ time algorithm for the Reeb graph”. In: *Discrete Comput. Geom.* 49 (2013), pp. 864–878. DOI: 10.1145/2261250.2261289.
- [22] R. Pascual, P. Le Gall, A. Arnould, and H. Belhaouari. “Topological consistency preservation with graph transformation schemes”. In: *SCP* 214 (2022), p. 102728. DOI: 10.1016/j.scico.2021.102728.
- [23] V. Pascucci, G. Scorzelli, P.-T. Bremer, and A. Mascarenhas. “Robust on-line computation of Reeb graphs”. In: *TOG* 26 (2007), p. 58. DOI: 10.1145/1276377.1276449.
- [24] A. Polette, J. Meunier, and J.-L. Mari. “Feature extraction using a shape descriptor graph based on discrete curvature patches”. In: *CGI*. 2015.
- [25] G. Reeb. “Sur les points singuliers d’une forme de Pfaff complètement intégrable ou d’une fonction numérique”. In: *Comptes Rendus Acad. Sciences Paris* 222 (1946), pp. 847–849.
- [26] M. Rucco, E. Merelli, D. Herman, D. Ramanan, T. Petrossian, L. Falsetti, C. Nitti, and A. Salvi. “Using topological data analysis for diagnosis pulmonary embolism”. In: *J. of Theo. and Appl. Comp. Sci.* 9.1 (2015), pp. 41–55.
- [27] A. Salch, A. Regalski, H. Abdallah, R. Suryadevara, M. J. Catanzaro, and V. A. Diwadkar. “From mathematics to medicine: A practical primer on topological data analysis (TDA) and the development of related analytic tools for the functional discovery of latent structure in fMRI data”. In: *PLOS One* 16.8 (2021). DOI: 10.1371/journal.pone.0255859.
- [28] N. Shivashankar and V. Natarajan. “Parallel Computation of 3D Morse-Smale Complexes”. In: *CGF* 31 (2012), pp. 965–974. DOI: 10.1111/j.1467-8659.2012.03089.x.
- [29] Y. Skaf and R. Laubenbacher. “Topological data analysis in biomedicine: A review”. In: *J. Biomed. Inform* 130 (2022), p. 104082. DOI: 10.1016/j.jbi.2022.104082.
- [30] V. Subhash, K. Pandey, and V. Natarajan. “A GPU Parallel Algorithm for Computing Morse-Smale Complexes”. In: *TVCG* 29.9 (2023), pp. 3873–3887. DOI: 10.1109/TVCG.2022.3174769.
- [31] J. Sun, M. Cieslak, S. T. Grafton, and S. Suri. “A Reeb graph approach to tractography”. In: *SIGSPATIAL* (2015).
- [32] S. Takahashi, Y. Takeshima, and I. Fujishiro. “Topological volume skeletonization and its application to transfer function design”. In: *GMOD* 66.1 (2004), pp. 24–49. DOI: 10.1016/j.gmod.2003.08.002.
- [33] J. Tierny, J.-P. Vandeborre, and M. Daoudi. “Partial 3D Shape Retrieval by Reeb Pattern Unfolding”. In: *CGF* 28 (2009), pp. 41–55. DOI: 10.1111/j.1467-8659.2008.01190.x.
- [34] T. Tung and F. Schmitt. “Augmented Reeb graphs for content-based retrieval of 3D mesh models”. In: *SMI*. 2004, pp. 157–166. DOI: 10.1109/SMI.2004.1314503.
- [35] J. Vidal, P. Guillou, and J. Tierny. “A Progressive Approach to Scalar Field Topology”. In: *TVCG* 27.6 (2021), pp. 2833–2850. DOI: 10.1109/TVCG.2021.3060500.
- [36] I. Vos, Y. Ruigrok, E. Bennink, M. Buser, B. Velthuis, and H. Kuijff. *Data of the Circle of Willis Intracranial Artery Classification and Quantification (CROWN) Challenge*. 2023. DOI: 10.34894/R05G1L.
- [37] K. Weiler. “Edge-Based Data Structures for Solid Modeling in Curved-Surface Environments”. In: *IEEE CG&A* 5.1 (1985), pp. 21–40. DOI: 10.1109/MCG.1985.276271.
- [38] N. Werghi. “Segmentation and modelling of full human body shape from 3D scan data: A survey”. In: *VISAPP*. 2006, pp. 189–197.
- [39] J. Zhang, A. H. Kashani, and Y. Shi. “3D Surface-Based Geometric and Topological Quantification of Retinal Microvasculature in OCT-Angiography via Reeb Analysis”. In: *MICCAI*. Vol. 11764. LNCS. 2019, pp. 57–65. DOI: 10.1007/978-3-030-32239-7_7.

A proposal of anomaly detection method based on natural data augmentation in the Eigenspace

Naoki Murakami
Chukyo University
101-2 Yagoto-honmachi
466-8666, Nagoya, Japan
murakami@asmi.sist.chukyo-u.ac.jp

Naoto Hiramatsu
Chukyo University
101-2 Yagoto-honmachi
466-8666, Nagoya, Japan
hiramatsu@asmi.sist.chukyo-u.ac.jp

Kobayashi Hiroki
Chukyo University
101-2 Yagoto-honmachi
466-8666, Nagoya, Japan
kobayashi@isl.sist.chukyo-u.ac.jp

Shuichi Akizuki
Chukyo University
101-2 Yagoto-honmachi
466-8666, Nagoya, Japan
s-akizuki@sist.chukyo-u.ac.jp

Manabu Hashimoto
Chukyo University
101-2 Yagoto-honmachi
466-8666, Nagoya, Japan
mana@isl.sist.chukyo-u.ac.jp

ABSTRACT

This paper proposes a natural data augmentation method and an anomaly removal artificial neural network for accurate anomaly detection. Anomaly detection is important because the provision of high-quality products is vital in the manufacturing industry. However, it is difficult to obtain a sufficient number of anomaly samples for the detection, which represents a significant challenge when it comes to achieving accurate anomaly detection by machine learning. General data augmentation methods generate new anomaly images by combining normal images and anomaly images. As an alternative, this paper describes a method that generates new anomaly images by using the Eigenspace. More natural anomaly images are generated than with general data augmentation methods. This paper also proposes an anomaly removal neural network that utilizes this natural data augmentation. The results of an anomaly detection experiment showed that the AUC of 94.7% was achieved for the capsule dataset when using anomaly images generated by the proposed data augmentation for training the anomaly removal neural network. This is 1.3% higher than the state-of-the-art data augmentation method that has been utilized for training the neural network. In the case of the pill dataset, AUC of 99.4% was achieved by proposed method. This is 3.0% higher than the state-of-the-art data augmentation method that has been utilized for training the neural network. The results of a series of experiments demonstrated that anomaly images generated by the proposed data augmentation are effective for training the neural network.

Keywords

Anomaly detection, Machine learning, Image generation, Data augmentation, Principal component analysis, Eigenspace

1. INTRODUCTION

Recently, machine learning has shown promise for accurate anomaly detection. However, there is a shortage of anomaly images for learning due to the difficulty to obtaining such images in the manufacturing industry. This is a critical issue when it comes to achieving accurate anomaly detection.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

Conventionally, three approaches to address this issue have been taken. In the first approach, methods use reconstruction models based on Generative adversarial networks (GANs) [Sch17] [Zen18] [Akc19]. These methods aim to successfully reconstruct normal images, while unsuccessfully reconstructing anomalies. However, they may be able to successfully reconstruct an anomaly image because the models learn to reconstruct the input image.

In the second approach, methods utilize pre-trained Convolutional neural networks (CNNs) as feature extractors [Coh20] [Rip21] [Def21] [Rot22]. These methods aim to model the normal features for detecting anomaly features. However, since they do

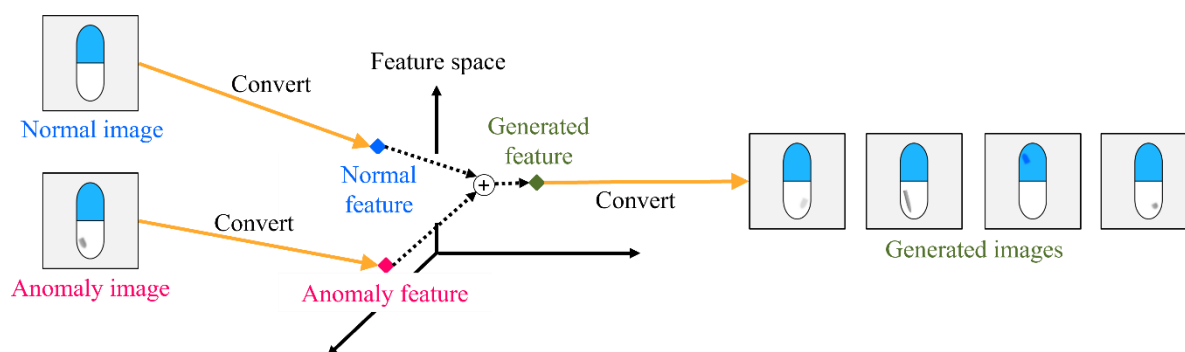


Figure 1. Basic concept of proposed method.

not learn anomaly images, they may miss very small anomalies.

In the third approach, methods generate artificial anomaly images by data augmentation. For example, CutPaste [Li21] trains the one-class discrimination of CNNs by using normal images and generated anomaly images. StainNoise [Col21], DRAEM [Zav21], and NSA [Sch22] train image reconstruction artificial neural networks using normal images and generated anomaly images and then determine anomalies from the input and output of the neural network. These methods can generate a lot of anomaly images by combining artificial anomalies and normal images. However, when the artificial anomalies are unnatural, artifacts may occur in the generated anomaly images, which affect the learning of the neural networks. In other words, if low-quality anomaly images are generated by the previous data augmentations, the accuracy of the subsequent anomaly detection may decrease.

Thus, Conventional data augmentation methods in the image space generate unnatural anomaly images. This is problematic for the quality of the training data. In this study, we propose a data augmentation method in the feature space instead of the conventional approach in the image space. Then, we propose a neural network for anomaly removal based on the data augmentation.

2. PREVIOUS DATA AUGMENTATION AND BASIC CONCEPT OF PROPOSED IDEA

In this section, we first describe the general approach to data augmentation and its issues. We then present our basic idea to address the problems of the previous methods.

2-1. Previous data augmentation methods and related problems

When applying machine learning to anomaly detection, it is necessary for the discriminator to learn various anomalies. However, there is a shortage of anomaly images for machine learning due to the

difficulty of obtaining a sufficient number of such images in the manufacturing industry.

In the field of anomaly detection, there are methods that generate anomaly images by combining artificial anomalies with normal images [Li21] [Col21] [Zav21] [Sch22]. These methods have the advantage of being able to generate the many anomaly images required for the training of discriminators. However, when the artificial anomalies are unnatural, artifacts may occur in the generated anomaly images, which in turn affects the learning of the discriminators. In other words, the problem here is that discriminators cannot acquire a generalizable performance with unnatural data augmentation. Our aim in this study is, therefore, to improve the quality of anomaly images generated by data augmentation.

2-2. Basic concept

Real images contain various information, such as the position of the object and the size of the anomalies. If we can extract and fuse these pieces of information from real images, we should be able to generate highly realistic anomaly images.

The basic concept is shown in Fig. 1. First, the images are converted into feature values, and next the normal feature values and anomaly feature values are fused in the feature space. Finally, high-reality anomaly images are generated by converting these feature values into images. Additionally, various anomaly images can be generated by applying transform processing to the anomaly feature values in the feature space.

3. PROPOSED METHOD

In this section, first, we explain our method of converting images into feature values using principal component analysis. Next, we present our method for combining normal feature values and anomaly feature values in the Eigenspace, along with our method for diversifying the anomaly information in the generated images. Finally, we show our method for training an anomaly removal neural network using normal images and generated anomaly images.

3-1. Feature transformation based on principal component analysis

In our approach, images are transformed into feature values by means of principal component analysis. There are two advantages to using principal component analysis for feature transformation: first, the feature values in the Eigenspace obtained in this way can be converted into images, and second, the feature values in the Eigenspace can be interpreted on the basis of the eigenvalues. Image generation methods utilizing the feature space include those using Variational Auto-Encoder (VAE) [Gar19] and GANs [Ant17] [Bow18]. However, since these methods use deep learning, it is difficult to interpret the features in the feature space.

Here, we transform images into features as follows. Principal component analysis is applied to N training images comprising many normal images and a small number of anomaly images. When the dimensionality of the image vectors is D dimensions ($N < D$), the N image vectors \mathbf{x}_n ($n = 1, 2, \dots, N$) are defined by

$$\mathbf{x}_n = (x_{n1}, x_{n2}, \dots, x_{nD})^T. \quad (1)$$

Next, the data is centered by taking the difference between the N image vectors \mathbf{x}_n and the mean vector $\bar{\mathbf{x}}$. The data matrix of the centered data $\bar{\mathbf{x}}_n$ ($n = 1, 2, \dots, N$) is defined by

$$\bar{\mathbf{X}} = (\bar{\mathbf{x}}_1, \dots, \bar{\mathbf{x}}_n)^T = (\mathbf{x}_1 - \bar{\mathbf{x}}, \dots, \mathbf{x}_n - \bar{\mathbf{x}})^T. \quad (2)$$

Using this data matrix $\bar{\mathbf{X}}$, the covariance matrix \mathbf{S} can be obtained by

$$\mathbf{S} = \frac{1}{N} \bar{\mathbf{X}}^T \bar{\mathbf{X}}. \quad (3)$$

In the principal component analysis, the eigenvalue problem for this covariance matrix \mathbf{S} is solved. As a result, the eigenvalues λ_j and the corresponding D dimensional eigenvectors \mathbf{a}_j are derived. When the dimensionality D is larger than the number of training images N , the number of non-zero eigenvalues and their corresponding eigenvectors obtained are $N - 1$.

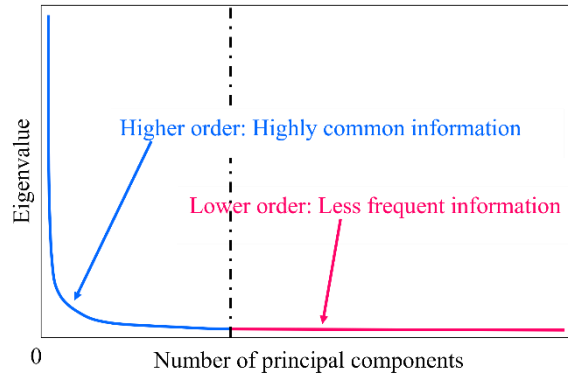


Figure 2. Relationship between eigenvalues and principal component numbers.

By selecting the eigenvectors \mathbf{a}_j corresponding to the eigenvalues λ_j in descending order, the Eigenspace \mathbf{A} is obtained. By projecting the centered image vectors $\bar{\mathbf{x}}_n$ onto Eigenspace \mathbf{A} , the image is transformed into the feature vectors \mathbf{s}_n , which is defined by

$$\mathbf{s}_n = \mathbf{A}^T \bar{\mathbf{x}}_n = (s_{n1}, s_{n2}, \dots, s_{nN-1})^T. \quad (4)$$

3-2. Fusion of anomaly features in Eigenspace

Next, using pairs of centered normal image vectors $\bar{\mathbf{x}}_{nor}$ and anomaly image vectors $\bar{\mathbf{x}}_{ano}$, new anomaly images $\tilde{\mathbf{x}}_{ano}$ are generated.

First, using Eq. (4) from Subsection 3-1, we project the normal image vectors $\bar{\mathbf{x}}_{nor}$ and the anomaly image vectors $\bar{\mathbf{x}}_{ano}$ onto Eigenspace \mathbf{A} . As a result, the feature vectors \mathbf{s}_{nor} and \mathbf{s}_{ano} are obtained.

The relationship between the eigenvalues λ_j and the principal component numbers is shown in Fig. 2. Here, we expect the top components with large eigenvalues to aggregate normal information correlated with all the data, while in contrast, the lower components with small eigenvalues are expected to aggregate anomaly information, especially for anomaly images.

Therefore, for the feature vectors \mathbf{s}_{nor} obtained from the normal images, the top components of the feature

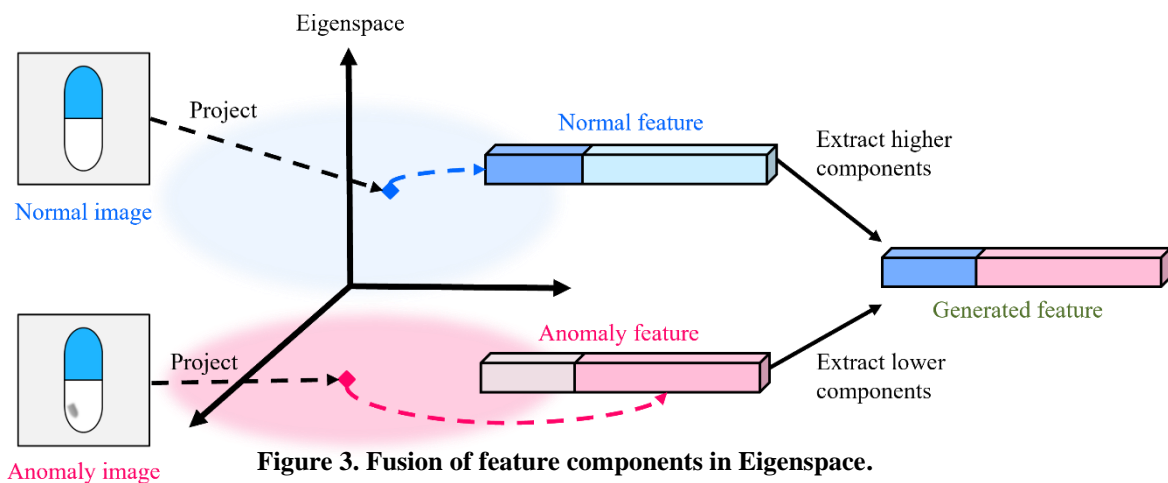


Figure 3. Fusion of feature components in Eigenspace.

vectors are extracted, and for the feature vectors \mathbf{s}_{ano} obtained from the anomaly images, the lower components are extracted. In this case, the boundary between the top and bottom features is qualitatively determined from the cumulative contribution rate graph, defining the components up to the M -th component as the top and the rest as the bottom.

Finally, new features are generated by combining the extracted feature components in the Eigenspace. Fig. 3 shows the generation of new features by fusing the normal and anomaly feature vectors in the Eigenspace. These features are transformed into images by adding the mean vector $\bar{\mathbf{x}}$ to the linear combination of their components and each eigenvector, as

$$\tilde{\mathbf{x}}_{ano} = \bar{\mathbf{x}} + \sum_{i=1}^M s_{nor(i)} \mathbf{a}_i + \sum_{i=M+1}^{N-1} s_{ano(i)} \mathbf{a}_i \quad (5)$$

By this operation, a new anomaly image $\tilde{\mathbf{x}}_{ano}$ is generated by fusing the defect features of defective image \mathbf{x}_{ano} into the normal image \mathbf{x}_{nor} .

3-3. Diversification processing of anomaly information

Using Eq. (5) from Subsection 3-2, new anomaly images are generated by fusing anomaly information from anomaly images into normal images. However, the position and size of anomalies in the generated images depend on the anomaly images used for the image generation. Therefore, variation in the generated images is limited to the number of combinations of normal and anomaly images. In this subsection, we modify Eq. (5) to variate the generated anomalies. There are two targets for diversifying anomalies: anomaly intensity and geometric information.

First, we diversify the anomaly intensity in the generated images by multiplying the weight coefficient w , as

$$\tilde{\mathbf{x}}_{ano} = \bar{\mathbf{x}} + \sum_{i=1}^M s_{nor(i)} \mathbf{a}_i + w \sum_{i=M+1}^{N-1} s_{ano(i)} \mathbf{a}_i \quad (6)$$

Next, we describe the diversification of geometric information about anomalies. In Eq. (5), the linear combination of anomaly features and lower eigenvectors represents anomaly information. Therefore, this anomaly information vector is transformed into the image-size matrix. Then, geometric transformations such as mirroring and reduction are applied to this matrix. Then, this matrix is transformed into a vector and added to the normal information vector. As a result, the geometric information of the anomaly in the generated image changes.

3-4. Anomaly removal neural network

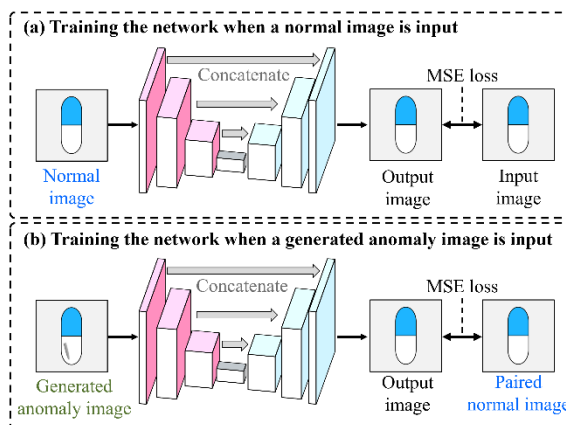


Figure 4. Training of anomaly removal neural network.

	Normal image	Anomaly images		
		Sample 1	Sample 2	Sample 3
Capsule				
Pill				

Figure 5. Example images of each dataset.

By using the data augmentation method described in Subsection 3-1 to Subsection 3-3, pairs of normal images and artificially generated anomaly images can be obtained. The artificially generated anomaly images in this pair are identical to the normal images except for the anomaly parts.

We then use this pair of images to train the anomaly removal neural network, as shown in Fig. 4. The neural network architecture utilizes U-Net [Ron15], which is a type of autoencoder. During the training, when a normal image is input, the Mean Squared Error (MSE) loss is calculated between the output \mathbf{y} of the neural network and the normal image \mathbf{x}_{nor} (Fig. 4(a)). When an artificial anomaly image generated using the proposed data augmentation is input, the MSE loss is calculated between the output \mathbf{y} of the neural network and the normal image \mathbf{x}_{nor} paired with this artificial anomaly image $\tilde{\mathbf{x}}_{ano}$ (Fig. 4(b)). The equation is shown below. As a result of this training, the anomaly removal neural network is obtained.

$$MSE = \frac{1}{nm} \sum_{j=0}^{n-1} \sum_{i=0}^{m-1} [\mathbf{y}(j, i) - \mathbf{x}_{nor}(j, i)]^2 \quad (7)$$

During the testing of the neural network, the anomaly score is defined as the sum of the absolute differences

between the input image x and the output image y of the neural network. The equation is shown below.

$$anomaly\ score = \sum_{j=0}^{n-1} \sum_{i=0}^{m-1} |y(j, i) - x(j, i)| \quad (8)$$

4. IMAGE GENERATION EXPERIMENT

We performed an image generation experiment in which new anomaly images were generated by the proposed data augmentation method using many normal images and a few anomaly images. Our objectives are to determine whether (1) various anomaly images can be generated by the fusion of normal and anomaly features in the Eigenspace and (2) the anomaly areas in the generated images appear natural.

4-1. Settings

We utilized the capsule dataset and the pill dataset comprising grayscale images sized 128×128 . Example images are shown in Fig 5, depicted in color for display purpose. Note that, since principal

component analysis is used for image generation, the position of the objects has been aligned in advance.

In this experiment, we utilized 300 normal images and 15 anomaly images from the capsule dataset. We utilized 1200 normal images and 15 anomaly images from the pill dataset. New anomaly images were generated using the proposed data augmentation method. We checked a graph of the cumulative contribution rate obtained by principal component analysis. We determined to be the top component up to a cumulative contribution rate of 99.0% for the capsule dataset and determined to be the top component up to a cumulative contribution rate of 90.0% for the pill dataset. Regarding the diversification process applied to anomaly information, a combination of scalar transformation was utilized. The value of the scalar multiplication was determined randomly within the range of 0.8 to 1.0. The mirror transformation was randomly selected from among three types (vertical, horizontal, and combined vertical and horizontal). The reduction transformation ratio was set randomly within the range of 0.8 to 1.0.













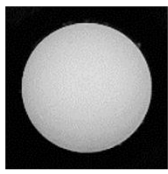
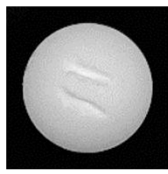
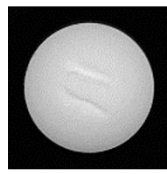
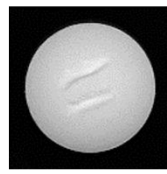
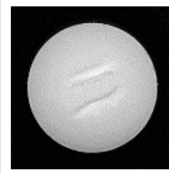
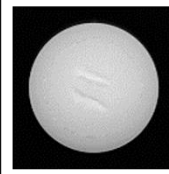
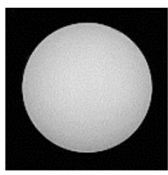
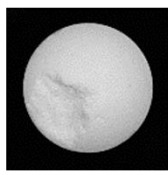
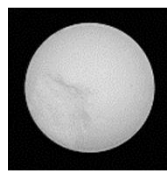
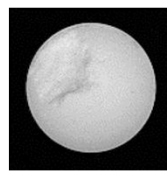
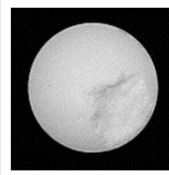
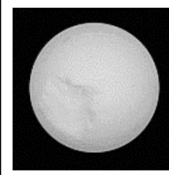
	Pair of real images		Examples of generated variations			
	Normal images	Anomaly images	Intensity	Geometry		
			Weight	Mirroring		Reduction
Capsule (1)						
Capsule (2)						
Pill (1)						
Pill (2)						

Figure 6. Results of image generation.

4-2. Results

Fig. 6 shows examples of pairs of normal images and anomaly images used for image generation, along with the generated images from those pairs. Note that, as in Fig.5, the images here are actually grayscale, but are depicted in color for display purpose. In terms of computation time (CPU: Intel Core i7-12700KF, Memory: 32 GB), Principal component analysis took 1566 sec, and generating 300 images took 442 sec for the capsule dataset.

The results of the capsule dataset in Fig. 6 show that new images have been generated by fusing the anomaly information from anomaly images onto normal images. Specifically, when the anomaly intensity is diversified, the anomaly intensity of the generated images changes compared to that of the anomaly images used for the data augmentation. Additionally, when the position of the anomaly is diversified by mirror transformation, the position of the anomaly in the anomaly images used for the data augmentation differs from in the generated images. At this time, only the position of the anomaly has changed, while the position of the capsule remains unchanged. When the size of the anomaly is diversified by reduction transformation, the anomalies in the generated images are smaller compared to those in the anomaly images used for the data augmentation. At this time, only the size of the anomaly has changed, while the size of the capsule itself remains unchanged. This is because the normal information and anomaly information have been appropriately extracted in the Eigenspace. Further, focusing on the anomaly area of the generated images, we can see that the boundary between the anomaly and the object is natural and that highly realistic anomaly images have been generated. Similar results were confirmed from the pill dataset.

These results demonstrate natural data augmentation has been achieved by the fusion of normal and anomaly information in the Eigenspace.

5. ANOMALY DETECTION EXPERIMENT

In the next experiment, we investigated whether the anomaly images generated by the proposed data augmentation are effective as training data for the anomaly removal neural network.

5-1. Settings

We utilized the same capsule and pill datasets here as in Section 4 and compared the following four cases.

- 1) The discriminator has been trained using anomaly images generated by the proposed data augmentation.
- 2) The anomaly removal neural network has been trained with only normal images.

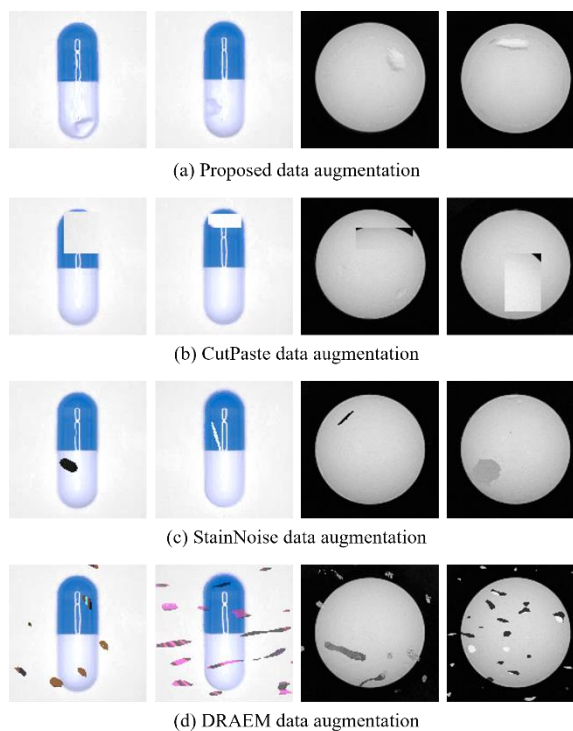


Figure 7. Qualitative comparison of proposed and previous methods.

	Capsule dataset			Pill dataset		
	Normal	Anomaly	Generated anomaly	Normal	Anomaly	Generated anomaly
Training ResNet using proposed method	300	0	300	1200	0	1200
Training U-Net using only normal image	300	0	0	1200	0	0
Training U-Net using previous method	300	0	300	1200	0	1200
Training U-Net using proposed method	300	0	300	1200	0	1200
Test	300	300	0	300	300	0

Table 1. Number of images in anomaly detection experiment.

- 3) The neural network has been trained using anomaly images generated by previous methods.
- 4) The neural network has been trained using anomaly images generated by the proposed data augmentation.

The previous methods we compared are CutPaste [Li21], StainNoise [Col21], and DRAEM [Zav21].

DRAEM is a state-of-the-art anomaly detection method using data augmentation.

Fig. 7 shows a comparison of the images generated by each method. CutPaste generates anomaly images by cutting out any rectangular region of a normal image and pasting it onto another area. StainNoise generates elliptical anomalies and synthesizes them into normal images to generate anomaly images. DRAEM generates anomalies from texture images of other

domains and synthesizes them into normal images to generate anomaly images. In this experiment, the previous methods used only the data augmentation part and used the same model as the proposed method for the training.

Table 1 breaks down the number of images used for training and testing the discriminator and the proposed method used 300 normal images and 300 artificial

anomaly images for training the neural network in the case of the capsule dataset. The 300 artificial anomaly images were generated using 15 real anomaly images and 300 normal images. In the case of the pill dataset, the proposed method used 1200 normal images and 1200 artificial anomaly images for training the neural network. The 1200 artificial anomaly images were generated using 15 real anomaly images and 1200

Data augmentation	Ours	Only normal	CutPaste	StainNoise	DRAEM	Ours
Model	ResNet	U-Net	U-Net	U-Net	U-Net	U-Net
AUC	85.2%	87.3%	90.3%	86.8%	93.4%	94.7%

Table 2. Results of the anomaly detection experiment for the capsule dataset.

Data augmentation	Ours	Only normal	CutPaste	StainNoise	DRAEM	Ours
Model	ResNet	U-Net	U-Net	U-Net	U-Net	U-Net
AUC	97.0%	63.7%	97.6%	99.4%	96.4%	99.4%

Table 3. Results of the anomaly detection experiment for the pill dataset.

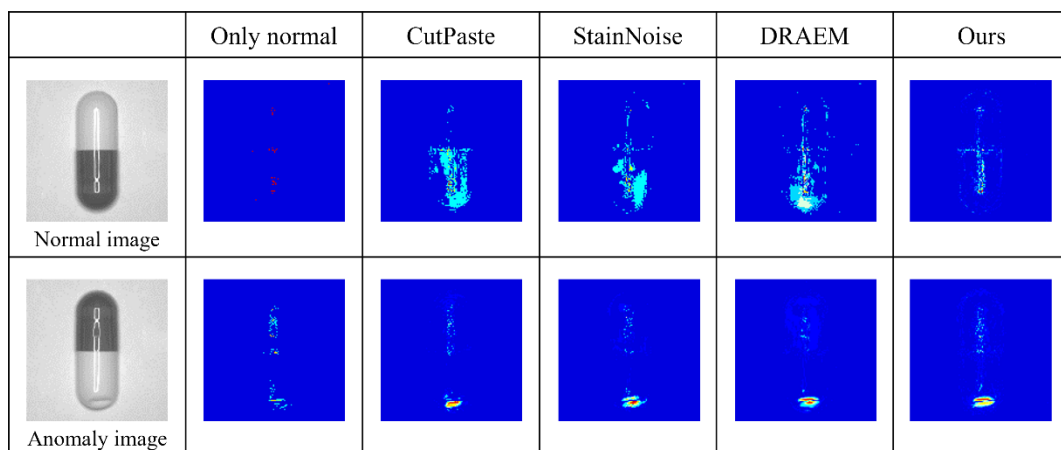


Figure 8. Anomaly score maps output from the anomaly removal neural network for capsule dataset.

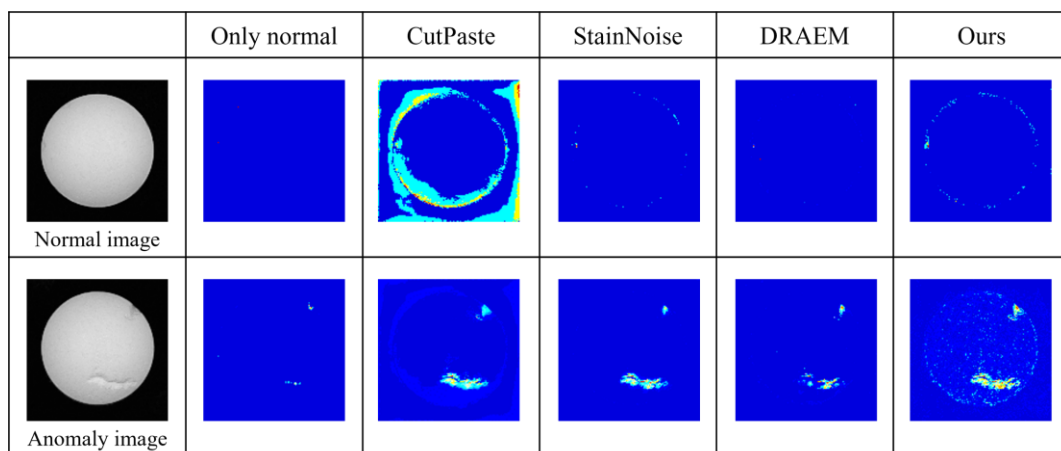


Figure 9. Anomaly score maps output from the anomaly removal neural network for pill dataset.

normal images. During testing, 300 normal images and 300 anomaly images were used.

Regarding the learning conditions, ResNet18 [He16] was used as the discriminator, and U-Net [Ron15] as the anomaly removal neural network. The batch size was set to 32 and the number of epochs was 300. The loss function for the discriminator was Binary Cross Entropy (BCE) loss. The loss function for the anomaly removal neural network was MSE loss. The optimization method was Adam [kin14] with the learning rate set to 0.0001. The anomaly detection performance was evaluated by the Area Under the Curve (AUC) during testing. AUC is the area under the ROC (Receiver Operating Characteristic) curve, which is created by calculating the true positive and false positive rates from the anomaly scores output from the neural network. Although the accuracy is also available as an evaluation indicator, AUC is more suitable from the viewpoint of threshold determination.

5-2. Results

The results of the anomaly detection experiments are listed in Table 2 and Table 3. Table 2 shows the experimental results for the capsule data set. Table 3 shows the experimental results for the pill dataset. From Table 2 (the capsule dataset), we can see the highest AUC of 94.7% was achieved when U-Net was trained using the proposed data augmentation. In contrast, the lowest AUC of 85.2% was achieved when ResNet18 was trained for class classification using the proposed data augmentation. From Table 3 (the pill dataset), we can see the highest AUC of 99.4% was achieved when U-Net was trained using the proposed data augmentation. In contrast, the lower AUC of 97.0% was achieved when ResNet18 was trained for class classification using the proposed data augmentation. The reason for the low result with ResNet is that, since its learning task is class discrimination, it was influenced by the number of real anomaly images used for the proposed data augmentation. Conversely, the anomaly removal neural network utilizing U-Net learns the task of removing anomalies from input images based on pairs of normal images and artificial anomaly images, so the influence of the number of real anomaly images used for the proposed data augmentation was minimal. These results indicate that when there are only a few real anomaly samples, training the proposed anomaly removal neural network can detect anomalies more accurately than class discrimination.

Furthermore, the proposed method had the highest AUC when compared with the case where the anomaly removal neural network was trained using artificially generated anomaly images by previous data augmentations. In the case of the capsule dataset, the proposed method is 1.3% higher than the result of DRAEM which is the state-of-the-art data

augmentation method. For the pill dataset, the proposed method is 3.0% higher than the result of DRAEM. Fig. 8 and Fig. 9 show the anomaly score map for input images when the anomaly removal neural network was trained using only normal images and when it was trained using each data augmentation method. First, when the anomaly removal neural network was trained only with normal images, the anomaly score map does not show any heat even when an anomaly image is input. This is because training the neural network only with normal images has given it the ability to output the same image as the input image. Next, when the anomaly removal neural network was trained using previous data augmentation methods, anomalies can be detected, but the anomaly score map also shows heat even when a normal image is input. In contrast, when the anomaly removal neural network was trained using the proposed data augmentation, anomalies can be detected and the anomaly score map does not show heat for the normal image. This is because the anomaly images generated by the proposed data augmentation are more natural than those generated by the previous data augmentation methods.

CONCLUSION

In this study, we proposed a natural data augmentation method that generates natural anomaly images by fusing normal and anomaly features in the Eigenspace, along with an anomaly removal neural network based on this natural data augmentation. The results of an image generation experiment, demonstrate that natural anomaly images can be generated from pairs of many normal images and a few anomaly images. In addition, by applying transform processing to the anomaly features in the Eigenspace, various anomaly images can be generated. The results of an anomaly detection experiment show that the AUC of 94.7% was achieved for the capsule dataset when utilizing anomaly images generated by the proposed data augmentation for training the anomaly removal neural network, which is 1.3% higher than the state-of-the-art data augmentation method that was utilized for training the neural network. In the case of the pill dataset, the results of an anomaly detection experiment show that the AUC of 99.4% was achieved when utilizing anomaly images generated by the proposed data augmentation for training the anomaly removal neural network, which is 3.0% higher than the state-of-the-art data augmentation method that was utilized for training the neural network. These findings demonstrate that anomaly images generated by the proposed data augmentation are effective for training the anomaly removal neural network.

6. REFERENCES

- [Sch17] Schlegl, Thomas, et al. "Unsupervised anomaly detection with generative adversarial networks to guide marker discovery", International conference on information processing in medical imaging, pp.146-157, 2017.
- [Zen18] Zenati, Houssam, et al. "Efficient gan-based anomaly detection" arXiv preprint arXiv:1802.06222, 2018.
- [Akc19] Akcay, Samet, et al. "Ganomaly: Semi-supervised anomaly detection via adversarial training", Asian Conference on Computer Vision, pp. 622-637, 2019.
- [Coh20] Cohen, Niv, and Yedid Hoshen. "Sub-image anomaly detection with deep pyramid correspondences", arXiv preprint arXiv:2005.02357, 2020.
- [Rip21] Rippel, Oliver, et al. "Modeling the distribution of normal data in pre-trained deep features for anomaly detection", International Conference on Pattern Recognition, pp. 6726-6733, 2021.
- [Def21] Defard, Thomas, et al. "Padim: a patch distribution modeling framework for anomaly detection and localization" International Conference on Pattern Recognition, pp. 475-489, 2021.
- [Rot22] Roth, Karsten, et al. "Towards total recall in industrial anomaly detection" Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 14318-14328, 2022.
- [Li21] Li, Chun-Liang, et al. "Cutpaste: Self-supervised learning for anomaly detection and localization" Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 9664-9674, 2021.
- [Col21] Collin, Anne-Sophie, et al. "Improved anomaly detection by training an autoencoder with skip connections on images corrupted with stain-shaped noise", International Conference on Pattern Recognition, pp. 7915-7922, 2021.
- [Zav21] Zavrtanik, Vitjan, et al. "Draem-a discriminatively trained reconstruction embedding for surface anomaly detection", Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 8330-8339, 2021.
- [Sch22] Schlüter, Hannah M., et al. "Natural synthetic anomalies for self-supervised anomaly detection and localization" European Conference on Computer Vision, pp. 474-489, 2022.
- [Gar19] Garay-Maestre, Unai, et al. "Data augmentation via variational auto-encoders", In Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications, pp. 29-37, 2019.
- [Ant17] Antoniou, Antreas, et al. "Data augmentation generative adversarial networks", arXiv preprint arXiv:1711.04340, 2017.
- [Bow18] Bowles, Christopher, et al. "Gan augmentation: Augmenting training data using generative adversarial networks" arXiv preprint arXiv:1810.10863, 2018.
- [Ron15] Ronneberger, Olaf, et al. "U-net: Convolutional networks for biomedical image segmentation", Medical image computing and computer-assisted intervention, pp. 234-241, 2015.
- [He16] He, Kaiming, et al. "Deep residual learning for image recognition", IEEE Conference on Computer Vision and Pattern Recognition, pp. 770-778, 2016.
- [Kin14] Kingma, Diederik P., et al. "Adam: A Method for Stochastic Optimization", arXiv preprint arXiv:1412.6980, 2014.

Fast Training Data Acquisition for Object Detection and Segmentation using Black Screen Luminance Keying

Thomas Pöllabauer

Volker Knauth

André Boller

Arjan Kuijper
Fraunhofer IGD, Fraunhoferstrasse 5, 64283, Darmstadt, Germany
TU Darmstadt, Karolinenplatz 5, 64289, Darmstadt, Germany
thomas.poellabauer@igd.fraunhofer.de

Dieter W. Fellner

ABSTRACT

Deep Neural Networks (DNNs) require large amounts of annotated training data for a good performance. Often this data is generated using manual labeling (error-prone and time-consuming) or rendering (requiring geometry and material information). Both approaches make it difficult or uneconomic to apply them to many small-scale applications. A fast and straightforward approach of acquiring the necessary training data would allow the adoption of deep learning to even the smallest of applications. Chroma keying is the process of replacing a color (usually blue or green) with another background. Instead of chroma keying, we propose luminance keying for fast and straightforward training image acquisition. We deploy a black screen with high light absorption (99.99%) to record roughly 1-minute long videos of our target objects, circumventing typical problems of chroma keying, such as color bleeding or color overlap between background color and object color. Next we automatically mask our objects using simple brightness thresholding, saving the need for manual annotation. Finally, we automatically place the objects on random backgrounds and train a 2D object detector. We do extensive evaluation of the performance on the widely-used YCB-V object set and compare favourably to other conventional techniques such as rendering, without needing 3D meshes, materials or any other information of our target objects and in a fraction of the time needed for other approaches. Our work demonstrates highly accurate training data acquisition allowing to start training state-of-the-art networks within minutes.

Keywords

Machine Learning, Object Detection, Object Segmentation, Deep Neural Networks.

1 INTRODUCTION

Modern machine learning (ML) is dominated by deep neural networks (DNNs). Training DNNs to state-of-the-art performance levels tends to require large amounts of training data to perform well. In many cases the lack of annotated data prevents the use of these networks. In the case of object segmentation and object detection a common approach - aside of costly manual labeling - is to resort to rendering to generate the required training images. To do so 3D meshes, textures, and material properties of the target objects are required, which is another barrier for many applications. In contrast, we only require the availability of the objects in question, a camera, a sufficiently large piece of special black cloth, and some lights. We completely circumvent many of the problems with traditional chroma keying such as color bleeding, same foreground/background color and similar, by utilizing a very low reflectance cloth and demonstrate its applicability in a fast, low-cost, high-quality setup, comparing favorably to much more complex data generation regimen.

Our contributions are the following:

- We propose a straightforward, easy to use setup to record high-quality training datasets for object segmentation and object detection.
- We present extensive evaluation and show the performance of our approach in comparison with other conventional data generation methods that require more information, such as meshes, textures, and materials, and/or much more processing time. To make our results more meaningful for the research community, we do all our evaluation on the common YCB-V dataset.
- We provide code, which automatically converts the recordings to datasets in COCO format for use with segmentation and 2D object detection algorithms, as well as our black screen recordings of the YCB-V objects.



Figure 1: A qualitative sample of all 21 YCB-V objects, that were recorded with a handheld smartphone and our proposed black background. It can be seen that the objects are well silhouetted against the background and can therefore be segmented in an easy way and many typical chroma key-associated problems are circumvented.

2 RELATED WORK

2.1 Data Generation for ML

Insufficient training data is a well documented problem in Machine Learning. In the domain of Computer Vision practitioners soon adopted rendering for data generation. Rendering has many suitable attributes, such as perfect ground truth, potentially unlimited amounts of data, and total control of scene composition, such as object pose and scene lighting. [TTS⁺18] demonstrates the combination of rendering on random backgrounds, as well as realistically placed objects in 3D scenes, while [HPH⁺19] shows another purely rendering-based approach. Another very common tool for data generation is BlenderProc [DSW⁺20], a pipeline extending Blender that allows for physically-based rendering. For a long time, the gap in feature representations between synthetic and real-world images was a problem [TFR⁺17, TPA⁺18, WGS⁺21]. With the use of physically-based rendering and the emergence of large foundation models such as CLIP [LLSH23, ODM⁺23, RKH⁺21], this problem is greatly reduced [SHL⁺23]. Another problem is not to be solved that easily however: In order to render you need 3D meshes of the objects in questions. There are approaches to estimate 3D shape [LGL⁺23] or additional views [VYB⁺24] based on a single image, but their quality is not on par with reality and their reconstruction is often purely probabilistic. Reconstructing 3D meshes with traditional methods such as photogrammetry [Sch05] is a very time consuming process. There is a move towards zero-shot ap-

proaches such as [NGP⁺23], which does not need images of the target objects for training, though still requires 3D meshes.

2.2 Significance of Reliable Data

The use of modern DNNs in computer vision improved solutions for a large variety of challenging tasks, provided sufficient training data. A prominent modern example would be Segment Anything (SAM), which is able to segment a very large amount of different objects in the wild [KMR⁺23]. However, ML processes come with significant risks and difficult to detect silent failures, if not handled correctly [HJS⁺20]. This is especially crucial in settings, where reliability and robustness is mandatory, such as in an industrial, clinical or dangerous contexts, which in turn can task-invalidate large unsupervised networks like SAM, due to its own limitations. One predominant challenge is, that trustworthy ML models need high-quality base data [LTH⁺22]. Many datasets are not task suited due to their size, inherent bias, dirtiness, or even just partial unfairness\incorrectness [WRSL23]. While these issues can be partly mitigated or worked around [RHW19, WRSL23], the relevant techniques introduce new layers of complexity and potential error sources. Albeit, all of these challenges can be overcome with manual labour and diligence, this can lead to the preemptive end for startups [BIRS22] and drive up cost for large companies.

2.3 Chroma Keying

Chroma Keying is a well established movie production technique from the 1920's, to segment objects in front of fixed mono-color backgrounds. In the beginning, a variety of different colors and shades like black/white, yellow, blue and green were popular [Fos10, Ram15]. Over time, the color green became the predominant background color for a variety of pragmatic reasons. It is easily distinguishable from human skin, rarely occurs outside of nature, does not require much lighting and is favourable for modern digital camera sensors because of higher sensitivity. However, some significant challenges remain: Green objects lead to falsely segmented foreground, which can also happen as a byproduct of color-bleeding or reflection. These effects strongly reduce segmentation quality for reflective materials, such as in metallic, transparent, and bright objects. Furthermore, object borders can become fuzzy due to lighting and merging effects with the background. While there are techniques to remedy drawbacks of conventional green screen, like color-unmixing [AAPS16], specialised capturing processes [BRS⁺22], color spill neutralization [GKTB10], threshold optimization [PJS17] and multiple background colors [SB96], they open up new problems and do not yet completely solve the inherent challenges while introducing additional capturing and/or post-processing effort.

2.4 Differentiation from similar works

LeCun et al. [LHB04] use a gray turn table to record objects and rely on chroma keying to replace foreground and background, while Dirr et al. use a white background [DBGD24]. Though a brighter background, like grey or white, would also satisfy the idea of luminance keying, they have major drawbacks in comparison to 99,99% light absorption black. First of all, they introduce far greater challenges for correct lighting conditions, arising from background shadows, background reflections and possible fuzzy edges, due to light scattering. Furthermore, it is harder to differentiate brighter object colors with bright backgrounds. In some cases, e.g. metallic or transparent objects, a non-black background is also prone to light shining partly through an object, reflections and refractions. These challenges are all naturally solved by our proposed solution, as no light is reflected or visible on the background by any relevant measure. Knauthe et al. use a capturing system, which foregoes a background for a white light source in a turn-table setup, which diminishes some of the challenges of a normal white background [KKvB⁺22]. However, this setup is very constrained in its rotation invariant use-case, due to the difficulties of building suitable shaped large light backgrounds. Agata et al. introduce dual color checker pattern backgrounds [AYK07, YAK08]. While this approach solves

foreground/background color confusion, the other issues presented earlier still persist. Additionally, the method requires more specialized backgrounds and introduces further complexity in the processing step, such as parameter tuning. Jin et al. developed a deep learning method for automatic real-time green screen keying [JLZ⁺22]. However, it is still limited by shadows and green spilling and requires a deep learning method with all inherent benefits and challenges.

3 APPROACH

3.1 Chroma and Luminance Key

We propose a simple yet effective data recording process: we place the objects on a very low-reflectance cloth, and record around 1-minute long video clips with a smartphone. These clips are processed via a cut and paste process, that uses the easily masked objects and places them on random backgrounds. Details on the processing are discussed in section 3.3.

3.1.1 Chroma Key with Green Screen

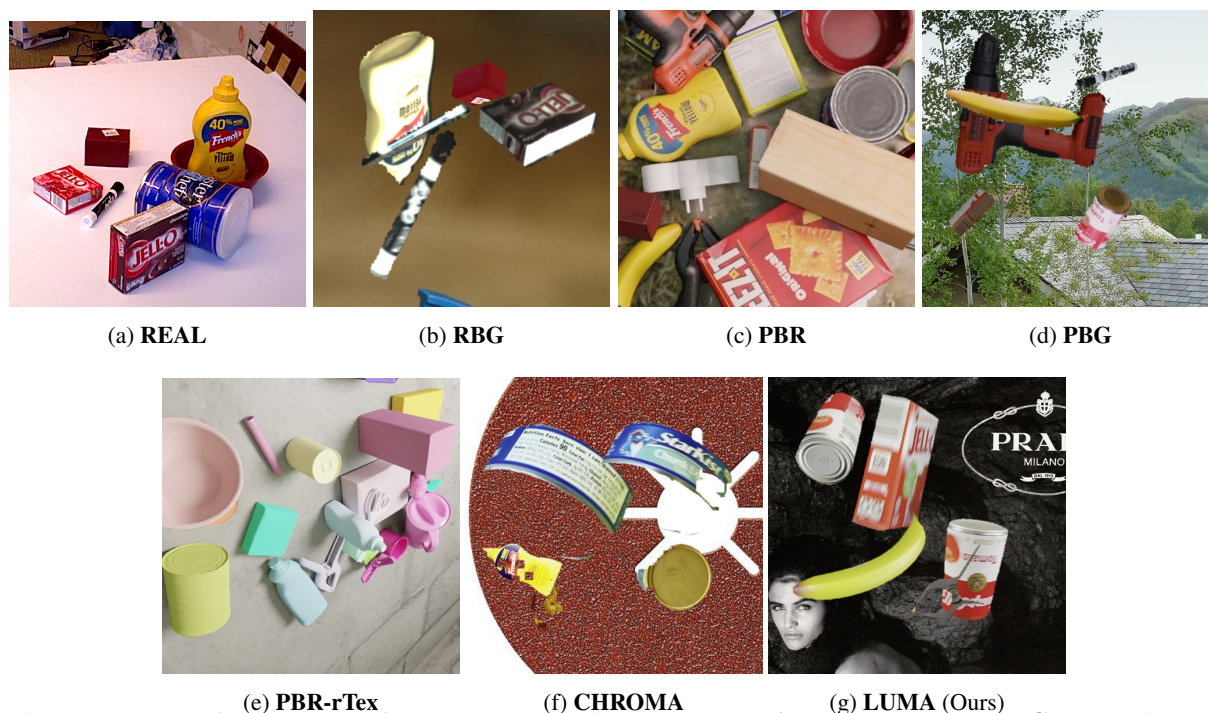
For comparison, we include recordings and experiments with the more common chroma keying approach. Arguably the most common colors are blue and green, with green being the most widely used color. Therefore we include a green screen in our evaluation.

3.1.2 Luminance Key with Black Screen

We propose Luminance Keying for fast and straightforward data acquisition for machine learning. This technique utilizes the brightness difference between an object and the background, instead of a designated color as with chroma keying. In practice, this is possible due to a textile background, which absorbs 99,99% of visible light. This leads to high contrast between object and background, even for very dark objects, allowing for high quality masking as illustrated in Figure 1. The recording process is exactly the same as with the conventional green screen described in section 3.1.1.

3.2 Baselines and Experiments

For our evaluation we include different data sources, all visualized in Figure 2: First, we include the real-world recordings of the YCB-V dataset as used in the prominent BOP Challenge [SHL⁺23]. However, there is a problem with the YCB-V test images: one could argue, that the limited number of scenes and camera poses reduces the expressiveness in terms of generalization. Therefore we also evaluate on the physically-based rendering (pbr) dataset provided by BOP. This dataset uses the reconstructed textures, leading to a high similarity in appearance compared to the real images, but has a much wider range of camera poses, object configurations, and lighting situations. These two



(a) **REAL** (b) **RBG** (c) **PBR** (d) **PBG**
(e) **PBR-rTex** (f) **CHROMA** (g) **LUMA (Ours)**
Figure 2: Samples from a subset of our evaluated training datasets. **REAL** are real images, **RBG** are real images with replaced backgrounds, **PBR** are physical based renderings, **PBG** are physical based renderings with background replacement, **PBR-rTex** are PBRs with randomized textures and **CHROMA**, as well as **LUMA (Ours)** stand for different capturing methods.

datasets, namely **REAL** and **PBR**, are used as a baseline on which all other data sources have to be tested. Also, we test all of our approaches on in-distribution samples, that is, on disjoint splits from the dataset, i.e. when training on our luminance images, we test on **REAL**, **PBR**, as well as on a disjoint set of luminance images. This should serve as a measure on how well an experiment generalizes to out-of-distribution samples. In more detail we use the following data representations:

Real-world images (REAL). Real-world recordings as described above.

Real-world images with background replacement (RBG). Real-world recordings. To test the influence of our background replacement script, we apply it to the real images as well. This gives us an idea on how much performance we loose because of our simple crop and paste approach of data generation.

Physically-based renderings (PBR). PBR renderings with reconstructed textures, realistic object placement, and light transport, as well as a high degree of camera and object pose variation.

Physically-based renderings with background replacement (PBG). Same as above, but again using background replacement for measuring its influence on performance.

Physically-based renderings with randomized textures (PBR-rTex). To simulate cases in which geometry is available, though we lack realistic textures

and materials, we include a set of pbr images with randomized surface attributes. Namely we randomize texture and surface details such as reflectance behavior.

Chroma key images using green screen (CHROMA). A set of green screen recordings to illustrate what results one can expect from using run-of-the-mill chroma keying methods. We use the green screen for segmentation and paste the crops on random photographs.

Luminance key images using black screen (LUMA). Our proposed method. A set of recordings utilizing the low-reflection black screen. Again, we have to crop and paste the objects on random backgrounds using our replacement script.

3.3 Training

We demonstrate the applicability of our approach on the 2D object detection use case and train YOLOX [GLW+21] networks.

Data Generation using cut and paste background replacement. To achieve technically good replacement results, we use a variety of probabilistic mechanisms that are applied during the replacement process inspired by [DMH17]. First, based on the masks provided, we filter out all objects in an image that are overlapped by others. This is trivial for luma, because there is no overlap during the acquisition, giving us near-perfect masks. In addition, we also make sure that the objects are not cut off at the edge of the image and finally do

a crop, centered around the object. Next we apply random affine transformations such as scaling, rotation and translation to the objects where the variable scaling, rotation and translation ensures variety in the appearance of the objects. After this process, the objects are placed on the background image based on randomly generated positions. We allow an overlap among objects of up to 20% in order to come closer to the real scenarios of YCB-V and make the network learn to deal with occlusion. For bleeding and to remove jaggies, the object masks are eroded and Gaussian noise is applied. As for the backgrounds we used random crops of images from the 50k HQ-data set [YCT⁺23]. Random cropping prevents the same background from being used more than once. **Parameterization.** To evaluate the practicability of our proposed method, we do an extensive comparison of different training sets. As shown in [HLWK18] freezing the backbone layers is useful for reducing the impact of domain shift. Since we introduce additional domain discrepancy with our background replacement, we add freezing to our evaluation of training from scratch, and using COCO-pretrained weights for initialization. As for the parameterization of YOLOX, we stick closely to the values and process as presented in their paper [GLW⁺21]. Most importantly for our comprehensive quantitative analysis we train model size "tiny" with image size 512x512, batch sizes of 64, half precision, multiscale range of 13, and for 300 epochs.

4 RESULTS

We report quantitative results on the common YCB-V object set and show samples of the achievable visual quality of the approach. In addition we show some problems of chroma keying, which luminance keying does not have in section 4.2.

4.1 Quantitative Results

In our first experiments we looked into answering the question, whether we want to train from scratch or load pre-trained weights, as well as whether to freeze the backbone. For a comprehensive list of all results, especially for unfrozen backbones, please refer to our tables presented in the appendix. Here we report results when freezing the backbone, as these led to best performance with our cut and paste background replacement. For evaluation we report both the Average Precision (AP) and the Average Recall (AR) metric in Tables 1 and 2. Since the currently available set of YCB-V objects has three modified objects (002_master_chef_can has a modified texture, for 019_pitcher_base the color changed from blue to a transparent color with a red lid, and for 035_power_drill there are some slight changes to the model), we report results on the subset of unchanged objects and call it YCB-V18, as well as

the results of the complete set.

As expected training and evaluating on REAL lead to best performance overall. This is no surprise since, among all data representations tested, features in the REAL training set are closest to the features found in the REAL test set. Physically-based rendering (PBR) comes in second place. The good performance of PBR can be explained with the photo-realistic depiction, wide range of physically-correct object placements, and lighting variations. Surprisingly the PBR set with randomized textures has very poor performance, far worse than expected. While we assumed a big drop, the extent might have to do with model size and might be less pronounced with more trainable parameters. As the 4 remaining approaches all build upon the cut and paste approach, we can directly compare the quality of the segmentation, as well as image fidelity. Here we find an outlier in the PBG (real photographs with background replacement) set: it under-performs the other approaches by a wide margin. Our assumption is that the reason is the comparatively poor masking quality. Masks are not pixel perfect and sometimes include background pixels or pixels belonging to an occluding object. Mask quality is no problem for RBG (pbr images with background replacement), having perfect masks by virtue of complete scene knowledge, but we argue RBG is held back by other problems, some similar to the ones found with chroma keying with color bleeding being the most obvious. Realistic light transport is a feature in fully rendered scenes, but becomes a detriment with cutting and pasting. Also, the textures, while being high fidelity, are not as good as real photographs. Finally, comparing LUMA with CHROMA we see an outperformance of more than 11% on both YCB-V18, as well as on the complete YCB-V set.

4.2 Qualitative Results

We depict some of the problems of chroma keying in Figure 3 and show the quality of our approach in Figure 1. In direct comparison with chroma keying using the green screen we note a significant improvement in mask quality. At the same time the process becomes noticeably faster and less error-prone since we do not have to match a specific color when thresholding to generate our masks. This problem is illustrated in Figure 4.

4.3 Discussion

We see a noticeable drop in performance related to our background replacement. This is to be expected since we remove relevant cues for the 3D scene understanding, such as global illumination, shadows, color bleeding, and realistic occlusion. However, when considering the subset of data representations utilizing the background replacement, namely REAL, PBR, CHROMA, and LUMA, we see the overperformance of LUMA.

TRAIN SET:	PBR-rTex	PBR	PBG	REAL	RBG	CHROMA	LUMA (Ours)
002_master_chef_can*	0.19/ 1.47 /-	44.65/ 62.13 /44.65	21.74/ 53.56 /17.35	6.29/ 73.76 /73.76	2.2/ 2.84 /0.0	9.54/ 46.75 /1.56	11.74/ 46.17 /3.64
003_cracker_box	0.09/ 0.0 /-	46.86/ 20.65 /46.86	16.33/ 5.15 /14.3	12.54/ 43.34 /43.34	10.06/ 9.65 /0.0	5.31/ 10.2 /0.02	13.68/ 20.9 /0.51
004_sugar_box	0.23/ 0.02 /-	32.95/ 48.36 /32.95	11.77/ 44.75 /13.03	5.62/ 49.54 /49.54	9.37/ 9.26 /66.26	9.42/ 38.23 /0.13	14.97/ 52.64 /0.48
005_tomato_soup_can	0.08/ 0.3 /-	38.04/ 55.53 /38.04	9.59/ 59.13 /10.47	5.75/ 70.21 /70.21	9.44/ 9.43 /78.67	4.73/ 36.42 /0.46	10.27/ 29.17 /1.42
006_mustard_bottle	0.65/ 0.14 /-	38.86/ 78.2 /38.86	14.19/ 76.07 /7.38	10.68/ 88.82 /88.82	12.7/ 12.45 /9.72	16.75/ 73.42 /2.48	17.63/ 79.9 /5.3
007_tuna_fish_can	0.24/ 0.2 /-	45.98/ 70.77 /45.98	11.04/ 65.36 /19.02	12.55/ 79.35 /79.35	10.69/ 10.74 /72.85	5.8/ 53.93 /0.23	11.34/ 60.97 /1.15
008_pudding_box	0.11/ 0.0 /-	28.54/ 4.48 /28.54	17.19/ 1.2 /22.23	1.77/ 6.64 /6.64	9.88/ 9.84 /0.0	4.05/ 2.43 /1.16	5.46/ 0.92 /2.5
009_gelatin_box	0.11/ 0.0 /-	36.82/ 71.21 /36.82	15.61/ 5.73 /22.94	6.11/ 65.41 /65.41	11.61/ 11.12 /66.94	2.26/ 0.52 /0.6	1.33/ 0.55 /2.41
010_potted_meat_can	0.31/ 0.27 /-	36.28/ 37.64 /36.28	11.4/ 18.3 /11.33	4.04/ 54.23 /54.23	9.06/ 9.04 /40.73	1.29/ 0.27 /2.83	11.52/ 26.22 /0.88
011_banana	5.79/ 6.58 /-	34.84/ 54.24 /34.84	14.91/ 40.67 /20.34	15.62/ 50.18 /50.18	14.18/ 14.28 /88.81	17.17/ 51.06 /3.16	14.95/ 48.64 /0.63
019_pitcher_base*	4.27/ 3.12 /-	54.15/ 55.57 /54.15	1.08/ 29.29 /0.55	13.56/ 80.89 /80.89	0.0/ 0.0 /0.0	6.68/ 0.38 /0.67	6.28/ 0.32 /0.09
021_bleach_cleanser	0.92/ 2.56 /-	33.82/ 43.58 /33.82	8.23/ 36.19 /6.22	2.42/ 61.31 /61.31	2.09/ 2.15 /8.35	8.03/ 47.44 /1.56	10.53/ 54.61 /5.26
024_bowl	2.14/ 0.22 /-	54.84/ 46.75 /54.84	4.96/ 0.01 /5.24	9.04/ 56.9 /56.9	2.47/ 1.93 /0.0	0.06/ 0.21 /0.08	6.99/ 14.44 /1.3
025_mug	0.18/ 3.3 /-	46.37/ 58.14 /46.37	10.89/ 1.87 /1.72	4.0/ 67.35 /67.35	3.01/ 3.01 /0.0	5.45/ 31.17 /1.02	2.04/ 14.37 /0.05
035_power_drill*	0.3/ 0.0 /-	23.28/ 13.26 /23.28	3.24/ 0.76 /1.39	2.28/ 43.72 /43.72	0.0/ 0.0 /0.0	3.87/ 5.71 /0.03	4.46/ 13.23 /0.04
036_wood_block	2.1/ 0.37 /-	45.99/ 24.57 /45.99	15.57/ 8.22 /14.56	8.42/ 34.06 /34.06	5.85/ 5.8 /0.0	10.28/ 12.75 /0.64	14.1/ 17.32 /2.53
037_scissors	0.67/ 0.0 /-	19.32/ 5.48 /19.32	0.67/ 0.16 /3.42	5.98/ 1.96 /1.96	1.36/ 1.32 /0.0	1.4/ 0.33 /0.69	1.35/ 0.52 /0.01
040_large_marker	0.04/ 0.0 /-	16.1/ 47.65 /16.1	2.54/ 22.29 /5.2	1.83/ 56.73 /56.73	0.47/ 0.41 /8.58	0.87/ 31.87 /0.23	0.16/ 5.42 /0.4
051_large_clamp	0.85/ 0.0 /-	18.2/ 47.43 /18.2	0.07/ 0.25 /2.5	2.96/ 6.4 /6.4	0.07/ 0.09 /0.01	1.85/ 0.28 /2.01	2.93/ 17.75 /0.07
052_extra_large_clamp	0.38/ 0.17 /-	19.27/ 7.21 /19.27	0.01/ 0.0 /2.45	2.38/ 9.8 /9.8	0.5/ 0.08 /0.0	8.79/ 11.31 /1.13	2.95/ 0.29 /1.82
061_foam_brick	0.71/ 0.01 /-	36.94/ 55.61 /36.94	15.86/ 3.38 /24.54	8.09/ 67.82 /67.82	6.99/ 6.63 /0.0	6.43/ 1.64 /1.11	6.01/ 5.22 /0.91
AVERAGE YCB-V	0.97/ 0.89 /-	35.81/ 43.26 /35.81	9.85/ 22.49 /10.98	6.76/ 50.88 /50.88	5.81/ 5.72 /21.0	6.19/ 21.73 /1.04	8.13/ 24.27 /1.49
AVERAGE YCB-V18	0.87/ 0.78 /-	35.0/ 43.19 /35.0	10.05/ 21.59 /11.73	6.66/ 48.34 /48.34	6.65/ 6.51 /24.5	6.11/ 22.42 /1.09	8.23/ 24.99 /1.54

Table 1: AP (higher is better) of all evaluated data representations, each tested on PBR, REAL, and in-distribution. In-distribution means the test set stems from the same data source (e.g. photograph, rendering, crop and paste) as the training set. Results on REAL in **bold font**. As expected, we see best results when training on the train split of the real data (REAL), followed by the physically simulated 3D scenes (PBR). Most importantly, among all 4 methods using the cut and paste approach for background replacement (PBG, RBG, CHROMA, LUMA), luminance (our proposed method) outperforms all others when tested on the real test data (REAL).

TRAIN SET:	PBR-rTex	PBR	PBG	REAL	RBG	CHROMA	LUMA (Ours)
002_master_chef_can*	8.05/ 18.8 /-	63.67/ 77.07 /63.67	39.66/ 71.67 /54.88	14.9/ 79.0 /79.0	4.59/ 4.63 /0.0	34.94/ 74.8 /9.17	33.24/ 71.97 /16.74
003_cracker_box	7.51/ 0.22 /-	62.82/ 66.44 /62.82	34.83/ 49.02 /56.82	22.93/ 62.89 /62.89	15.5/ 21.64 /0.0	13.62/ 36.22 /1.14	26.64/ 48.53 /4.63
004_sugar_box	6.75/ 2.61 /-	53.39/ 66.8 /53.39	31.07/ 74.37 /48.63	18.53/ 61.63 /61.63	27.76/ 75.68 /77.14	22.13/ 52.69 /5.68	32.84/ 69.07 /7.23
005_tomato_soup_can	4.2/ 5.96 /-	54.06/ 69.71 /54.06	22.89/ 66.91 /35.95	20.81/ 75.09 /75.09	26.39/ 68.96 /84.03	25.11/ 54.96 /7.86	30.4/ 61.8 /14.32
006_mustard_bottle	11.46/ 8.27 /-	57.22/ 87.6 /57.22	27.01/ 83.87 /34.08	23.68/ 91.27 /91.27	28.38/ 82.33 /55.56	30.54/ 83.87 /13.41	32.17/ 84.13 /21.22
007_tuna_fish_can	7.29/ 4.37 /-	57.74/ 78.83 /57.74	26.17/ 72.1 /51.98	30.23/ 83.77 /83.77	30.72/ 75.17 /84.8	24.34/ 72.77 /7.5	24.88/ 73.3 /11.4
008_pudding_box	3.31/ 0.27 /-	48.6/ 56.4 /48.6	33.25/ 23.73 /65.56	14.17/ 48.93 /48.93	24.87/ 54.27 /0.0	26.21/ 43.6 /8.24	27.76/ 27.07 /10.97
009_gelatin_box	2.96/ 0.0 /-	49.85/ 78.4 /49.85	26.41/ 50.8 /60.06	21.31/ 74.27 /74.27	28.77/ 72.4 /85.1	10.21/ 23.87 /4.89	8.33/ 16.27 /9.05
010_potted_meat_can	6.69/ 0.84 /-	53.21/ 64.98 /53.21	28.8/ 48.09 /38.76	19.73/ 59.6 /59.6	28.64/ 54.31 /78.69	7.47/ 10.4 /7.67	26.99/ 52.93 /5.43
011_banana	26.99/ 31.8 /-	49.23/ 70.87 /49.23	24.18/ 55.27 /46.25	28.16/ 61.4 /61.4	23.91/ 54.07 /90.0	25.33/ 58.27 /15.58	22.91/ 55.33 /10.21
019_pitcher_base*	29.3/ 28.53 /-	71.89/ 77.33 /71.89	4.47/ 34.76 /9.04	26.38/ 84.22 /84.22	0.0/ 0.0 /0.0	27.79/ 8.18 /5.28	25.67/ 6.44 /3.86
021_bleach_cleanser	16.59/ 17.13 /-	54.95/ 60.43 /54.95	22.5/ 58.4 /33.01	13.72/ 71.0 /71.0	2.78/ 17.27 /24.38	20.38/ 59.2 /11.14	20.94/ 63.1 /13.81
024_bowl	25.35/ 9.47 /-	66.4/ 73.07 /66.4	16.25/ 1.13 /29.93	14.96/ 63.13 /63.13	2.67/ 18.47 /0.0	0.58/ 5.0 /4.0	12.03/ 55.6 /11.33
025_mug	7.74/ 29.13 /-	60.39/ 74.13 /60.39	17.04/ 11.47 /22.32	18.81/ 71.93 /71.93	15.77/ 72.2 /0.0	19.77/ 72.53 /10.26	6.37/ 45.8 /2.7
035_power_drill*	7.57/ 0.07 /-	47.82/ 44.43 /47.82	11.34/ 7.3 /25.11	14.38/ 54.23 /54.23	0.0/ 0.0 /0.0	24.05/ 41.73 /1.92	20.71/ 54.23 /0.96
036_wood_block	15.6/ 19.87 /-	62.63/ 65.73 /62.63	32.68/ 32.93 /46.97	20.44/ 50.8 /50.8	12.61/ 28.67 /0.0	26.27/ 56.67 /8.75	28.67/ 57.07 /9.82
037_scissors	7.87/ 0.27 /-	34.22/ 20.8 /34.22	4.53/ 1.73 /35.71	12.38/ 6.4 /6.4	9.77/ 8.27 /0.0	9.97/ 6.27 /3.09	7.06/ 3.73 /1.22
040_large_marker	2.72/ 0.13 /-	30.6/ 62.0 /30.6	8.85/ 39.67 /38.86	7.12/ 63.33 /63.33	8.13/ 60.53 /33.79	6.54/ 59.8 /5.68	3.54/ 31.47 /3.1
051_large_clamp	6.62/ 0.0 /-	43.2/ 69.8 /43.2	3.06/ 12.47 /28.43	10.62/ 32.8 /32.8	0.35/ 11.47 /3.33	12.01/ 9.8 /10.44	13.17/ 36.53 /2.86
052_extra_large_clamp	8.39/ 4.2 /-	47.61/ 45.87 /47.61	0.77/ 0.0 /30.26	9.41/ 35.87 /35.87	0.07/ 0.07 /0.0	21.33/ 26.93 /6.18	12.54/ 9.07 /4.32
061_foam_brick	6.92/ 3.47 /-	52.27/ 72.93 /52.27	25.99/ 59.87 /56.78	23.88/ 76.27 /76.27	19.89/ 71.73 /0.0	22.89/ 25.47 /4.77	22.8/ 54.4 /6.92
AVERAGE YCB-V	10.47/ 8.83 /-	53.42/ 65.89 /53.42	21.04/ 40.74 /40.45	18.41/ 62.28 /62.28	14.84/ 40.58 /29.37	19.6/ 42.05 /7.27	20.93/ 46.56 /8.2
AVERAGE YCB-V18	9.72/ 7.67 /-	52.13/ 65.82 /52.13	21.46/ 41.21 /42.24	18.38/ 60.58 /60.58	17.05/ 47.08 /34.27	18.04/ 42.13 /7.57	20.0/ 46.96 /8.36

Table 2: AR (higher is better) of all evaluated data representation, corresponding to Table 1. Results on REAL in **bold font**. LUMA (Ours) compares favourably, outperforming PBG, RBG, and CHROMA.



Figure 3: Some of the problems with chroma keying. Color bleeding leads to part of the object appearing greenish, which leads to imperfect masking (top left). Luminance keying (top right) in contrast gives much improved masking. Other problems with chroma are the high reflectivity of conventional backgrounds, that lead to a "halo" effect at the edges (bottom left), and the cutting out of object parts close to the background color (bottom right).



Figure 4: Another problem of chroma keying. While the lighting did not change, a slight change in camera settings between two video clips lead to very different tones of green, making the thresholding much harder compared to luminance keying.

At the same time this relative gain in performance of REAL and PBR comes at a cost: creating a suitable big dataset to be able to train on REAL requires (usually manual) labeling, while the creation of a PBR dataset presumes the existence of 3D meshes or requires the scanning of the objects. Also, we have a throughput of roughly 50 images rendered per minute on a single A100 GPU, making the PBR dataset costly in terms of resources and time.

5 CONCLUSION

In conclusion, we propose a luminance keying method using a black screen with 99.99% light absorption

for efficient training data acquisition, significantly simplifying the process of training deep neural networks for object segmentation and detection. Our technique overcomes the limitations of manual annotation and rendering, traditionally required for creating annotated datasets, by employing a high-absorption black background to facilitate quick video recording and brightness-based automatic masking of objects. This approach not only expedites the data preparation process but also eliminates common issues associated with chroma keying, such as color bleeding and color overlap. We did extensive evaluation and find that our method compares favourably to much more involved data generation approaches on the YCB-V object set. Overall, this work enables the rapid deployment of deep learning applications across various scales, democratizing access to state-of-the-art object detection and segmentation technologies. For reproducibility and further research we publish our processing code, as well as our black screen YCB-V recordings at <https://huggingface.co/datasets/tpoellabauer/YCB-V-LUMA>.

6 REFERENCES

- [AAPS16] Yağiz Aksoy, Tunç Ozan Aydin, Marc Pollefeys, and Aljoša Smolić. Interactive high-quality green-screen keying via color unmixing. *ACM Transactions on Graphics (TOG)*, 36(4):1, 2016.
- [AYK07] Hiroki Agata, Atsushi Yamashita, and Toru Kaneko. Chroma key using a checker pattern background. *IEICE TRANSACTIONS on Information and Systems*, 90(1):242–249, 2007.
- [BIRS22] James Bessen, Stephen Michael Impink, Lydia Reichensperger, and Robert Seamans. The role of data for ai startup growth. *Research Policy*, 51(5):104513, 2022.
- [BRS⁺22] Lukas Block, Adrian Raiser, Lena Schön, Franziska Braun, and Oliver Riedel. Image-bot: generating synthetic object detection datasets for small and medium-sized manufacturing companies. *Procedia CIRP*, 107:434–439, 2022.
- [DBGD24] Jonas Dirr, Johannes C Bauer, Daniel Gebauer, and Rüdiger Daub. Cut-paste image generation for instance segmentation for robotic picking of industrial parts. *The International Journal of Advanced Manufacturing Technology*, 130(1):191–201, 2024.
- [DMH17] Debidatta Dwibedi, Ishan Misra, and Martial Hebert. Cut, paste and learn: Surprisingly easy synthesis for instance

- detection. In *Proceedings of the IEEE international conference on computer vision*, pages 1301–1310, 2017.
- [DSW⁺20] Maximilian Denninger, Martin Sundermeyer, Dominik Winkelbauer, Dmitry Olefir, Tomas Hodan, Youssef Zidan, Mohamad Elbadrawy, Markus Knauer, Harinandan Katam, and Ahsan Lodhi. Blenderproc: Reducing the reality gap with photorealistic rendering. In *International Conference on Robotics: Science and Systems, RSS 2020*, 2020.
- [Fos10] Jeff Foster. *The Green Screen Handbook*. Sybex (Wiley), 2010.
- [GKTB10] Anselm Grundhöfer, Daniel Kurz, Sebastian Thiele, and Oliver Bimber. Color invariant chroma keying and color spill neutralization for dynamic scenes and cameras. *The Visual Computer*, 26:1167–1176, 2010.
- [GLW⁺21] Zheng Ge, Songtao Liu, Feng Wang, Zeming Li, and Jian Sun. YOLO: Exceeding yolo series in 2021. *arXiv preprint arXiv:2107.08430*, 2021.
- [HJS⁺20] Ronan Hamon, Henrik Junklewitz, Ignacio Sanchez, et al. Robustness and explainability of artificial intelligence. *Publications Office of the European Union*, 207, 2020.
- [HLWK18] Stefan Hinterstoisser, Vincent Lepetit, Paul Wohlhart, and Kurt Konolige. On pre-trained image features and synthetic images for deep learning. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, pages 0–0, 2018.
- [HPH⁺19] Stefan Hinterstoisser, Olivier Pauly, Hauke Heibel, Marek Martina, and Martin Bokeloh. An annotation saved is an annotation earned: Using fully synthetic training for object detection. In *Proceedings of the IEEE/CVF international conference on computer vision workshops*, pages 0–0, 2019.
- [JLZ⁺22] Yue Jin, Zhaoxin Li, Dengming Zhu, Min Shi, and Zhaoqi Wang. Automatic and real-time green screen keying. *The Visual Computer*, 38(9):3135–3147, 2022.
- [KKvB⁺22] Volker Knauth, Maurice Kraus, Max von Buelow, Tristan Wirth, Arne Rak, Laurenz Merth, Alexander Erbe, Christian Kontermann, Stefan Guthe, Arjan Kuijper, et al. Alignment and reassembly of broken specimens for creep ductility measurements. In *VMV*, pages 33–40, 2022.
- [KMR⁺23] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4015–4026, 2023.
- [LGL⁺23] Xiaoxiao Long, Yuan-Chen Guo, Cheng Lin, Yuan Liu, Zhiyang Dou, Lingjie Liu, Yuexin Ma, Song-Hai Zhang, Marc Habermann, Christian Theobalt, et al. Wonder3d: Single image to 3d using cross-domain diffusion. *arXiv preprint arXiv:2310.15008*, 2023.
- [LHB04] Yann LeCun, Fu Jie Huang, and Leon Bottou. Learning methods for generic object recognition with invariance to pose and lighting. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, volume 2, pages II–104. IEEE, 2004.
- [LLSH23] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *International conference on machine learning*, pages 19730–19742. PMLR, 2023.
- [LTH⁺22] Weixin Liang, Girmaw Abebe Tadesse, Daniel Ho, Li Fei-Fei, Matei Zaharia, Ce Zhang, and James Zou. Advances, challenges and opportunities in creating data for trustworthy ai. *Nature Machine Intelligence*, 4(8):669–677, 2022.
- [NGP⁺23] Van Nguyen Nguyen, Thibault Groueix, Georgy Ponimatkin, Vincent Lepetit, and Tomas Hodan. Cnos: A strong baseline for cad-based novel object segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2134–2140, 2023.
- [ODM⁺23] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.
- [PJS17] Thanathorn Phoka, Warayu Jariyawattanarat, and Attawith Sudsang. Fine tuning for green screen matting. In *2017 9th*

- International Conference on Knowledge and Smart Technology (KST)*, pages 317–322. IEEE, 2017.
- [Ram15] Kathryn Ramey. *Experimental Filmmaking : Break the Machine*. Taylor Francis Ltd, 2015.
- [RHW19] Yuji Roh, Geon Heo, and Steven Euijong Whang. A survey on data collection for machine learning: a big data-ai integration perspective. *IEEE Transactions on Knowledge and Data Engineering*, 33(4):1328–1347, 2019.
- [RKH⁺21] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.
- [SB96] Alvy Ray Smith and James F Blinn. Blue screen matting. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 259–268, 1996.
- [Sch05] Toni Schenk. Introduction to photogrammetry. *The Ohio State University, Columbus*, 106(1), 2005.
- [SHL⁺23] Martin Sundermeyer, Tomáš Hodaň, Yann Labbe, Gu Wang, Eric Brachmann, Bertram Drost, Carsten Rother, and Jiří Matas. Bop challenge 2022 on detection, segmentation and pose estimation of specific rigid objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2784–2793, 2023.
- [TFR⁺17] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 23–30, 2017.
- [TPA⁺18] Jonathan Tremblay, Aayush Prakash, David Acuna, Mark Brophy, Varun Jampani, Cem Anil, Thang To, Eric Cameracci, Shaad Boochoon, and Stan Birchfield. Training deep networks with synthetic data: Bridging the reality gap by domain randomization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2018.
- [TTS⁺18] Jonathan Tremblay, Thang To, Balakumar Sundaralingam, Yu Xiang, Dieter Fox, and Stan Birchfield. Deep object pose estimation for semantic robotic grasping of household objects. *arXiv preprint arXiv:1809.10790*, 2018.
- [VYB⁺24] Vikram Voleti, Chun-Han Yao, Mark Boss, Adam Letts, David Pankratz, Dmitry Tochilkin, Christian Laforte, Robin Rombach, and Varun Jampani. Sv3d: Novel multi-view synthesis and 3d generation from a single image using latent video diffusion. *arXiv preprint arXiv:2403.12008*, 2024.
- [WGS⁺21] Olivia Wiles, Sven Goyal, Florian Stimberg, Sylvestre-Alvise Rebuffi, Ira Ktena, Krishnamurthy Dj Dvijotham, and Ali Taylan Cemgil. A fine-grained analysis on distribution shift. In *International Conference on Learning Representations*, 2021.
- [WRSL23] Steven Euijong Whang, Yuji Roh, Hwanjun Song, and Jae-Gil Lee. Data collection and quality challenges in deep learning: A data-centric ai perspective. *The VLDB Journal*, 32(4):791–813, 2023.
- [YAK08] Atsushi Yamashita, Hiroki Agata, and Toru Kaneko. Every color chromakey. In *2008 19th International Conference on Pattern Recognition*, pages 1–4. IEEE, 2008.
- [YCT⁺23] Qinhong Yang, Dongdong Chen, Zhentao Tan, Qiankun Liu, Qi Chu, Jianmin Bao, Lu Yuan, Gang Hua, and Nenghai Yu. Hq-50k: A large-scale, high-quality dataset for image restoration. *arXiv preprint arXiv:2306.05390*, 2023.

Appendix

Table 3: In our extensive evaluation we address the problem of domain gap / distribution shift between image sources. First we analyse the effect of freezing the backbone (first table), training from scratch (second table), and continuing training from weights pre-trained on COCO (third table). Next we train on our 7 proposed data representations. To emphasize the influence of data similarity, we next use a range of different validation sets, corresponding to different data representations and sources and determine the best checkpoint for each. Finally, we test the best checkpoint against relevant test sets. All sets and splits are disjoint. NoBGC .. Chroma data without background replacement, NoBGL .. Luma w/o bg. repl., 50 .. a set consisting of 50 percent images with and 50% images w/o bg. repl.

		BB Freezing, Pretrained on COCO																																	
		PBR rand. Tex		PBR		PBG		Real		RBG		Chroma		Luma																					
Train	Val	PBR	PBG	Real	RBG	Chroma	Luma	PBR	PBG	Real	RBG	Chroma	Luma	PBR	PBG	Real	RBG	Chroma	Luma	PBR	PBG	Real	RBG	Chroma	Luma										
Test	PBR	1.0	0.6	1.0	1.0	35.8	35.8	35.7	35.7	9.9	9.8	9.2	9.8	6.8	6.4	6.3	6.4	5.8	5.7	5.7	5.7	6.2	6.1	6.2	6.1	6.1	6.1	6.1	6.1	8.1	8.1	8.1	8.1	8.1	7.6
	PBG	0.2	0.2	0.2	0.2	4.6	4.6	4.6	4.6	10.8	11.0	10.2	10.9	2.1	2.1	2.1	2.1	6.5	6.6	6.6	6.6	4.9	4.9	4.9	5.0	5.0	5.0	5.0	5.0	5.5	5.5	5.6	5.5	5.6	5.3
	Real	0.8	0.2	0.9	0.8	43.3	43.3	43.2	43.1	22.1	21.1	22.5	21.1	48.1	50.7	50.9	50.7	26.2	25.6	25.6	25.3	21.7	21.4	21.7	21.3	21.3	21.3	21.3	24.6	24.2	24.3	24.3	24.3	24.1	23.9
	RBG	0.0	0.0	0.0	0.0	7.1	7.1	7.3	7.0	15.5	15.3	15.6	16.2	4.4	6.3	6.1	6.3	20.7	21.2	21.1	21.0	13.4	12.9	13.6	13.7	13.7	13.7	12.3	16.0	16.3	16.3	16.3	16.3	16.3	14.5
	NoBGC	0.0	0.0	0.0	0.0	3.7	3.6	3.6	3.5	0.1	0.0	0.1	0.1	0.6	0.9	1.0	0.8	0.5	0.6	0.5	0.6	0.7	1.0	0.6	1.0	1.0	1.0	1.4	0.5	0.9	0.9	0.9	0.9	1.0	1.7
NoBGL	0.0	0.0	0.0	0.0	4.9	5.0	4.9	5.1	0.1	0.1	0.1	0.1	1.0	1.1	1.2	1.1	0.3	0.4	0.4	0.4	1.0	1.3	0.7	1.2	1.2	1.2	2.1	1.0	1.5	1.5	1.5	1.5	1.5	1.7	3.4
Ckpt		246	16	166	288	288	249	263	246	190	289	95	243	30	173	249	179	214	267	277	285	198	170	270	288	288	288	94	200	248	246	246	288	249	45
		Without BB Freezing, From Scratch																																	
		PBR rand. Tex		PBR		PBG		Real		RBG		Chroma		Luma																					
Train	Val	PBR	PBG	Real	RBG	Chroma	Luma	PBR	PBG	Real	RBG	Chroma	Luma	PBR	PBG	Real	RBG	Chroma	Luma	PBR	PBG	Real	RBG	Chroma	Luma										
Test	PBR	1.8	0.8	0.1	1.6	73.7	72.9	68.7	72.3	10.4	2.4	7.6	6.8	1.6	1.6	1.6	1.6	0.7	0.2	0.5	0.1	3.5	3.1	2.9	3.1	1.6	1.6	3.1	6.0	5.1	5.5	5.1	2.8	4.2	5.9
	PBG	0.2	0.2	0.1	0.1	6.1	6.5	6.4	6.6	15.0	9.3	11.4	25.0	0.9	0.8	0.8	0.9	5.6	7.4	5.8	6.3	6.3	6.4	6.3	6.5	6.0	6.0	4.9	7.2	7.8	7.4	7.8	6.5	7.4	7.4
	Real	0.0	0.0	0.0	0.0	8.7	15.0	19.5	15.6	0.4	0.0	0.6	0.2	44.3	46.4	46.2	46.3	24.6	18.4	22.8	15.1	2.7	2.3	2.0	2.3	0.5	0.5	3.3	0.8	0.9	1.7	0.6	0.1	0.4	0.7
	RBG	0.0	0.0	0.0	0.0	0.9	1.0	2.2	1.2	4.8	1.3	1.2	4.7	2.7	2.6	1.8	2.1	35.1	43.3	36.5	39.0	8.5	9.6	7.6	9.2	4.2	4.2	6.9	10.0	10.6	11.0	10.8	4.0	9.9	9.6
	NoBGC	0.0	0.0	0.0	0.0	3.5	2.8	4.2	3.1	0.0	0.0	0.1	0.0	0.5	0.6	0.5	0.6	0.0	0.0	0.0	0.0	14.2	13.7	10.3	12.7	11.0	11.0	24.2	3.9	1.0	5.0	1.2	1.8	2.2	5.3
NoBGL	0.0	0.0	0.0	0.0	7.2	7.4	8.9	9.8	0.0	0.0	0.2	0.0	0.7	0.9	1.0	1.2	0.0	0.0	0.0	0.0	4.6	5.1	4.0	4.5	6.3	6.3	6.1	21.8	16.7	27.5	23.2	23.8	28.4	31.5	
Ckpt		48	6	2	101	169	80	43	63	18	289	6	39	103	161	182	165	29	165	32	271	48	63	85	64	250	250	22	54	188	75	172	288	219	60
		Without BB Freezing, Pretrained on COCO																																	
		PBR rand. Tex		PBR		PBG		Real		RBG		Chroma		Luma																					
Train	Val	PBR	PBG	Real	RBG	Chroma	Luma	PBR	PBG	Real	RBG	Chroma	Luma	PBR	PBG	Real	RBG	Chroma	Luma	PBR	PBG	Real	RBG	Chroma	Luma										
Test	PBR	3.2	3.2	3.2	3.2	74.7	52.8	52.8	52.8	9.6	2.2	8.7	9.6	10.5	10.5	10.1	10.1	4.9	4.8	4.8	0.0	5.8	5.8	5.3	5.8	2.0	2.8	2.8	8.3	6.3	7.1	7.7	3.4	4.3	7.1
	PBG	0.6	0.6	0.6	0.6	6.2	6.5	6.5	6.5	19.8	9.4	16.6	19.8	3.3	3.3	3.4	3.4	8.9	9.5	9.5	7.5	6.6	6.6	5.4	6.6	6.3	6.7	6.7	7.2	5.6	6.4	6.3	6.7	5.6	
	Real	0.5	0.5	0.5	0.5	11.4	26.2	26.2	26.2	0.0	0.0	0.1	0.0	64.2	64.2	64.3	64.3	21.8	24.8	24.8	14.7	2.6	2.6	10.0	2.6	0.5	1.0	1.0	2.3	0.5	7.2	1.7	0.1	0.1	7.2
	RBG	0.0	0.0	0.0	0.0	2.8	6.7	6.7	6.7	15.2	0.6	13.0	15.2	11.3	11.3	10.6	10.6	31.9	34.1	34.1	41.4	9.6	9.6	14.9	9.6	6.4	6.9	6.9	10.2	7.5	12.6	13.0	4.3	6.1	12.6
	NoBGC	0.0	0.0	0.0	0.0	8.5	6.6	6.6	6.6	0.0	0.0	0.0	0.0	5.9	5.9	5.9	5.9	0.3	0.0	0.0	0.0	11.9	11.9	11.5	11.9	11.6	19.0	19.0	4.2	0.5	12.4	2.5	1.1	1.4	12.4
NoBGL	0.0	0.0	0.0	0.0	11.0	8.5	8.5	8.5	0.1	0.0	0.0	0.1	3.2	3.2	3.3	3.3	0.1	0.1	0.1	0.0	4.1	4.1	4.7	4.1	2.8	6.3	6.3	20.0	6.3	24.9	16.8	12.5	14.2	24.9	
Ckpt		1	1	1	1	140	3	3	3	13	279	7	13	9	9	6	6	5	7	7	261	8	8	2	8	246	166	166	11	158	2	20	288	249	2

First Results on Using Transformer Architecture for Extroversion Personality Trait Recognition

Alan Goncalves University of Campinas R. Paschoal Marmo, 1888 Jd Nova Italia CEP 13484-332, Limeira, SP a122507@dac.unicamp.br	Marco A. G. Carvalho University of Campinas R. Paschoal Marmo, 1888 Jd Nova Italia CEP 13484-332, Limeira, SP magic@unicamp.br	Josue J. G. Ramos Renato Archer IT Center Rod. Dom Pedro I, SP-065 Km 143,6 - Amarais CEP 13069-901, Campinas, SP jgramos@cti.gov.br	Pedro V. V. Paiva University of Campinas R. Paschoal Marmo, 1888 Jd Nova Italia CEP 13484-332, Limeira, SP p193016@dac.unicamp.br
--	--	--	---

ABSTRACT

Personality traits are characteristics that can describe a person's behavior, also reflecting their thoughts and feelings. There are those who support the idea that traits can be strong predictors of leadership, implying emotional stability of the individual. Knowing the importance of the subject, areas such as psychology and neuropsychology have been studying and analyzing personality, aiming to better understand such patterns that guide behavior. A model widely accepted to categorize personality traits is known as *Big Five* and uses the acronym OCEAN: Openness, Conscientiousness, Extroversion, Agreeableness and Neuroticism. On the other hand, new approaches that emerged from the field of computer vision allow to analyzing personality from visual data, making this new area of research quite attractive for researchers. This work presents an initial study of the use of the Transformer architecture to analyze personality traits, with a specific focus on extroversion, using digital videos of human faces. A literature review was carried out focusing on the application of computational techniques in this issue involving deep learning and Transformers. We also accomplished an experiment analysing Extroversion personality trait, as a starting point for our studies, using the ChaLearn dataset. An AUC (Area under the ROC Curve) value of 71.04% was obtained, with fine adjustment of parameters in the transformer, demonstrating the robustness of the proposed architecture.

Keywords

Personality trait Recognition, OCEAN model, Affect Computing, Video Processing.

1 INTRODUCTION

To understand what Personality Traits are, we first need to understand what are the dimensions that define them. The theory of personality traits is an approach based on the definition and evaluation, and it is related to habitual patterns of behavior, thoughts and emotions that are relatively stable over time [costa1998trait]. They are important for psychology because they help to describe and explain how people behave and react to different situations [thoresen2004big]. They also provide a structure to understand the consistency and stability of an individual's psychological characteristics

in different contexts [chioqueta2005personality]. To understand human personality, theorists and psychologists have developed frameworks for organizing this vast amount of information, often using personality scales. Different feature models have been proposed in order to represent personality traits such as 16PF [cattell1986number], Big-Two [abele2007agency], HEXACO [ashton2007empirical], NEO-PI-R and NEO-FFI [sharpe2001revised]. Nevertheless, in 1961, Tupes and Christal [tupes1992recurrent] found five recurring factors in analysis of personality classifications. This model became known as the Five Great Traits, or simply *Big Five*, represented by OCEAN Acronym: Openness, Conscientiousness, Extroversion, Agreeableness and Neuroticism. The Big Five became the most adopted and influential model in the field of psychology. Among the various application areas of using a personality trait representation model, we have: **Psychology**: used to better understand human behavior and help in the diagnosis and treatment of mental disorders; **Human Resources**: evaluation of job

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

candidates, helping in the selection of those who best fit the needs of the company; **Marketing:** Identifying consumer profile and personalizing of strategies, and **Artificial Intelligence Systems:** Considered one of the most promising areas of the moment, helping to improve the interaction between humans and machines.

Currently, the assessment of personality traits is performed mainly through questionnaires and structured interviews conducted by psychologists or other mental health professionals. Psychological studies have shown that the face can also play a central role in everyday interpersonal assessments, being of great importance for the identification of personality traits [todorov2008understanding]. Several papers have established a correlation between facial appearance and personality traits [kramer2010internal] [penton2006personality] [little2007using] [sutherland2015personality] [junior2019first] [gucluturk2017visualizing]. In this context, computer vision techniques can play an important role in supporting the assessment of personality traits.

In recent years, advances in the area of natural language processing and, in particular, the use of Transformer architecture achieved high effectiveness in textual analysis tasks. The most popular Transformer architectures include Bidirectional Encoder representations of transformers BERT [devlin2018bert] and Generative Pretrained Transformer (GPT) v1-3 [radford2018improving] [radford2019language], for instance. These models have the ability to understand and process large volumes of text, identifying complex semantic patterns and relationships. For this reason, many researchers directed their efforts to the application of this new approach in their computer vision problems, in order to obtain the first impressions about an individual's personality trait.

Transformer is based on two components, where the first encodes a source sentence x and the second decodes a target sentence y . The Encoder-Decoder structure is used in most neural sequence transduction models. The Transformer architecture proposed by Vaswani [vaswani2017attention], is an innovative approach to sequence processing, which overcame many limitations of previous approaches in Natural Language Processing (NLP). Its ability to model long-range relationships and complex contexts revolutionized the field and led to the development of powerful models such as BERT, GPT and others that achieved exceptional results. Multi-head attention is the backbone of the Transformer architecture.

This paper presents an initial study of the application of Transformers Architecture to identify extroversion personality trait through face analysis in digital videos. We intend to answer the following research question:

Do Transformers offer significant benefits in the context of identifying personality traits due to their contextual learning capability, long-term representation, generalization, and interpretability? We also describe and present a systematic review of the literature in the area, summarizing gaps and opportunities, citing existing datasets and the approaches used to treat the problem of identifying personality traits. The rest of this paper is organized as follows: Section 2 presents a brief review of related works and the systematic review carried out. The approach adopted is presented in Section 3. Section 4 shows initial results and presents the performance evaluation by means of accuracy metric. Finally, in Section 5, we present the main conclusions and future work.

2 RELATED WORKS

We obtained the state-of-the-art through a systematic review of the literature (SRL) and were used these key-words: personality traits, computational Techniques, computer vision, images, videos, machine learning and deep learning, following the procedures established in the methodology proposed by [kitchenham2009systematic]. The initial survey of this systematic review resulted in 281 papers, of which 219 works analyzed were identified as not being duplicated, from these, 39 were selected. However, to cover the most recent work, an update to the SRL was carried out in May 2024, resulting in an increase of 17 new articles, bringing the total to 56. The papers were collected from 7 different databases, as can be seen in Table 1. We can see that most of the papers were selected from the IEEE database.

The analysis step of each of the selected articles was guided by three research questions:

- What computational approaches and data modalities were used to assess personality traits?
- What classification or regression models were used?
- Which dataset was used for Training and Testing?

Searched databases	Number of Papers
Web of Science	7
Science Direct	5
Springer	3
Engineering Village	3
IEEE	23
ACM	4
Scopus	10
Wiley	1
Total	56

Table 1: Number of articles extracted per search base.

The SRL was carried out at the end of 2022 and updated in May 2024, thus considering works published

between 2017 and 2024. Figure 1 shows the distribution of these articles over the years researched.

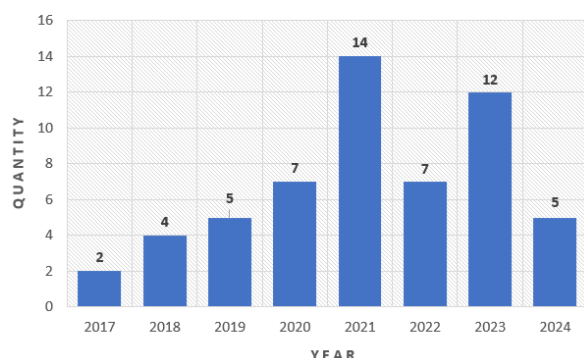


Figure 1: Papers published per year. Source: Author.

Table 2 shows some of the most relevant papers that are related to our work. In addition to highlighting the adopted approach, it also shows the modalities concerning the type of data used to assess personality traits.

Paper	Year	Approach	Modalities
Zhang et al.	2017	CNN	I
Rai, Nishant.	2016	CNN	A+I
Wei et al.	2017	CNN	A+I
Agastya et al.	2019	CNN	V+I
Yesu et al.	2021	CNN	V+I
Moreno et al.	2020	DCNN	I
Xu et al.	2021	DNN	I
Ventura et al.	2017	DAN	I
Ibrahim et al.	2023	CNN	I
Gucluturk et al.	2017	Resnet	V+I
Bounab et al.	2024	LSTM	V

Table 2: Main Features Paper | A-Audio, I-Image, V-Video

Within the established period of the SRL, we noticed a large presence in the approach to the modalities of Images and Videos. Both Ventura [Ventura2017CVPRWorkshops] and Gucluturk [gucluturk2017visualizing] in 2017 used the Regression technique for their works. While Ventura's work used the interpretability of the video modality and worked with the architecture called DAN+, which is an extension of the Descriptor Aggregation Networks (DAN), Gucluturk's model [gucluturk2017visualizing] used a ResNet Deep learning methodology, which took advantage of the visual and auditory information available in the dataset. Still in 2017, Zhang [zhang2017physiognomy] published a paper that aimed to evaluate the personality traits and intelligence of individuals from their faces, using images. They explained that the face can play a crucial role in interpersonal relationships, thus being able to impact important social events, such as elections and court decisions. The authors study uses Convolutional Neural Networks (CNN) to recognize personality traits.

The work presented by Rai Nishant in 2017[rai2016bi] used Regression Assignments and CNN. Their approach aimed to predict features by combining several models including Deep Neural Networks that focused on leveraging visual information on faces. In addition to observing face information through images, this work also combine the use of audio in order to provide a more precise accuracy. In 2019, Agastya[agastya2019systematic] presented a significant systematic review of the literature, analyzing 25 papers. This work sought to understand the approaches in the recognition of personality traits using deep learning algorithms. Yesu's work in 2021[yesu2021big] has focused on image observation using Convolutional Neural Networks. His work aimed to explore the possibility of mapping the Big Five personality traits, generating a score for each of the personalities. The goal is training computers to understand the personality traits of human beings based on their facial shapes.

At the end of reading the works, we identified the lack of datasets with videos labeled with personality traits to be used in the assessments. Another point is that several Deep Learning techniques such as CNN, SVM, DAN, DNN, etc are being used, however, no work using the transformers architecture was found. This was the main motivation for pursuing this work. The Figure 2 illustrates the distribution of the most used datasets in the articles, reflecting the papers that were selected during the SLR.

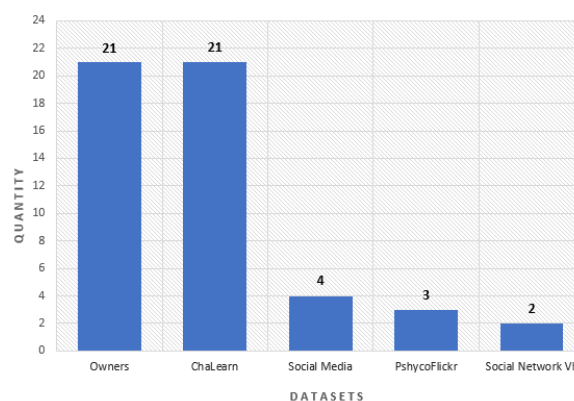


Figure 2: Dataset used by Papers selected during the SLR. Source: Author.

3 PROPOSED APPROACH

In this section we present our proposed approach to classify the extroversion personality trait. It consists of three main steps: Dataset preparation, Pre-processing and Classification, according to the pipeline shown in Figure 3. Each step is described through the remainder of this section.

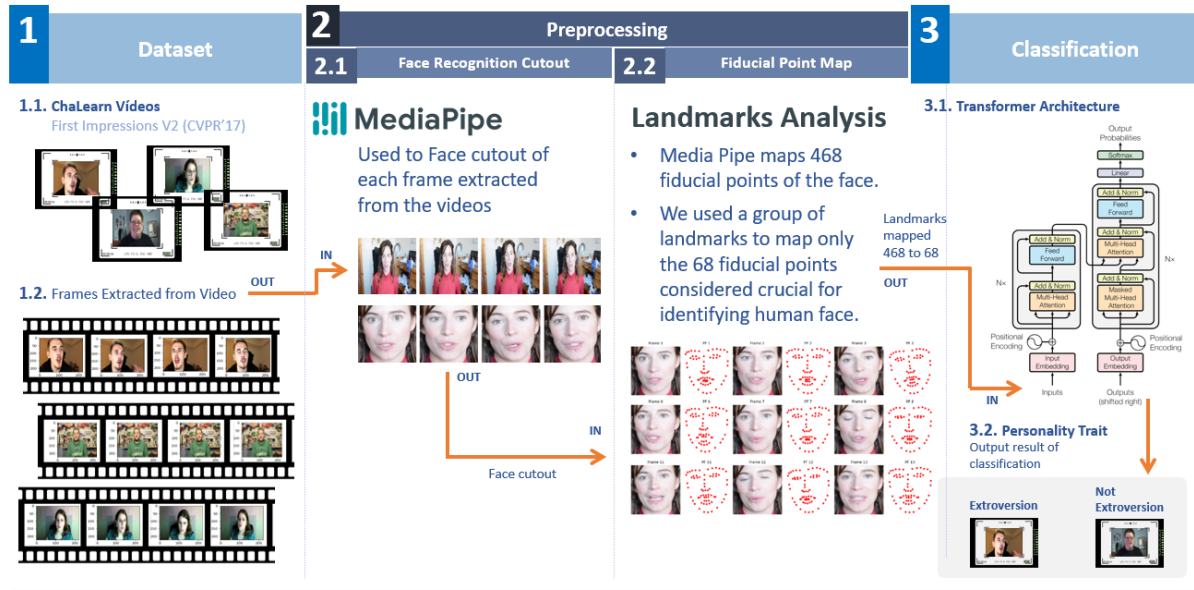


Figure 3: Proposed Methodology approach using Transformer for extroversion personality trait recognition. Source: Author

3.1 Dataset preparation

This step is responsible for presenting dataset characteristics and how its preparation was implemented.

3.1.1 First Impressions dataset

First Impressions is a personality trait dataset comprising 10,000 clips (average length 15 seconds) extracted from over 3,000 different high definition (HD) YouTube videos of people looking and speaking in English into a camera. Videos are split into training, validation, and testing sets with an aspect ratio of 3:1:1. People in the video have different gender, age, nationality and ethnicity[escalante2020modeling]. Videos were labeled using Amazon MechanicalTurk (AMT) service; Its post processing generate the values for the different personality traits. Some principles were adopted to guarantee the reliability of the labels[escalante2020modeling]. The personality traits considered were those of the Big Five model.

3.1.2 Frame Extraction

In this step we extracted the ChaLearn dataset package *First Impressions V2 (CVPR'17)*¹. Also, we selected frames taken from the videos at regular intervals. For our analyses, we developed a Frame extractor from the labeled videos of the ChaLearn dataset and we set the Number of Frames as 8 and the frame format as (224, 224, 3). Figure 4 illustrates a frame extraction for the "zEyRyTnIw5l.005.mp4" video, classified as Extroverted.

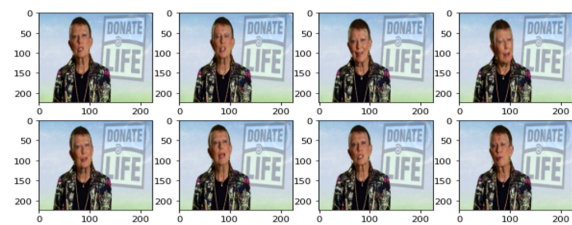


Figure 4: Resulting frames obtained from video extraction step. Source: Author.

3.2 Preprocessing

This step consists of two data preparation tasks for the classification model: Face Recognition CutOut, and Fiducial Point Map. In our proposed work, **Face Recognition CutOut** from video frames is performed by Google Media Pipe Face Mesh Library. Google MediaPipe is able to predict and return landmarks of a human face based on only one perspective with high accuracy [ali2021classical]. Human face identification through facial alignment is one of the main challenges of the pre-processing stage in the general facial recognition process. So, Google's MediaPipe library uses a deep learning model trained on a large amount of data to perform facial mapping using landmarks. Specifically, MediaPipe Face Detection and MediaPipe Face Mesh are two CNN models used to detect faces and estimate facial landmarks, respectively. In the second preprocessing task, to perform the **Fiducial Point Map**, we defined a list with just 68 landmarks that contains the indices of the facial landmarks that will be extracted from the detected face [zhang2022applications]. These indices correspond to the fiducial points identified by MediaPipe's Face Mesh

¹ <https://chalearnlap.cvc.uab.cat/>

model. The Face Mesh model is capable of detecting up to 468 facial landmarks in an image. Thus, only 68 of the 468 points are selected for extraction. These points were chosen because they are the most relevant points for face recognition analysis. This list of fiducial points is essential for extracting relevant facial features from an image, such as the contours of the face, the position of the eyes, mouth, etc [samaan2022mediapipe]. These features can be used for a variety of purposes, such as facial recognition, facial expression analysis, blink detection, among other computer vision applications.

3.3 Classification

The classification of personality traits is performed using the Transformers architecture. The Transformers architecture was proposed by Vaswani and his team, in the article Attention is All You Need [vaswani2017attention], and it is the approach of a new simple network architecture, based only on attention mechanisms, not requiring recurrence and convolutions. Attention mechanisms in neural networks, such as Transformer, allow the network to focus on specific parts of the input during processing. Instead of treating all input at once, these mechanisms prioritize certain parts, giving them more "attention" during calculation. Thus, in the context of images or videos, attention mechanisms allow the network to focus on specific regions of the image or specific frames of the video. Instead of processing the image as a whole at once, the network with attention mechanisms can direct its attention to specific parts of the image that are most relevant to the task at hand. This allows for a more focused and adaptive approach to processing visual information, which can lead to better performance in tasks such as object recognition, image segmentation or video analysis. This third block is divided into two stages. The first one, is the **Transformer Model**: Step where we will apply the Transformer architecture in Image Classification to identify personality traits and the second, is the **Personality Traits Classification**: Results of image classification, according to their respective personality traits.

4 INITIAL EXPERIMENT & RESULTS

In this section we show our initial results obtained for the extroversion personality trait through the proposed approach described in the previous section. Table 3 shows the Hyperparameters that were used in this first experiment.

- **N HEADS** (Number of Heads): This parameter refers to the number of attention heads in a Transformer model.
- **N LAYERS** (Number of layers): Indicates the number of layers stacked in the Transformer model.

Hyper Parameter	Value
N_HEADS	2
N_LAYERS	5
DROPOUT	0.3
MLP	256
BATCH SIZE	1280
ACTIVATION	gelu
D_MODEL	128
D_FF	512
EPOCHS	5000

Table 3: Main Feature parameters

- **DROPOUT**: Dropout is a regularization technique commonly used in neural networks to prevent overfitting. This value represents the dropout rate, that is, the proportion of units that will be temporarily deactivated randomly during training to prevent the network from becoming too dependent on certain neurons.
- **MLP (Multilayer Perceptron)**: This is the size of the hidden layer of the feedforward neural network inside the Transformer.
- **BATCH SIZE**: The batch size indicates the number of training examples used in one iteration.
- **ACTIVATION**: Refers to the activation function used in the hidden layers of the neural network.
- **D MODEL (Model dimension)**: It is the dimension of representation of the input and output vectors in each layer of the model.
- **D FF (Feedforward Dimension)**: It is the dimension of the hidden layer in the feedforward neural network inside the Transformer.
- **EPOCHS (Epochs)**: Represents the number of times the entire dataset is passed through the model during training.

We employ the TensorFlow framework to train the proposed network on a server with 64-GB RAM, Xeon 12x 2.4GHz (computer equipped with a Xeon processor with 12 processing cores), and 1x TITAN X (computer capable of handling advanced graphics and computationally intensive tasks). Each video in First Impressions Dataset receives a score associated with each of the personality traits: extroversion, neuroticism, agreeableness, conscientiousness and openness. In this way, and taking into account the 6000 videos destined for the training stage, we create a new dataset designed for a binary classification, in which these videos were separated into two distinct classes: Extroversion and non-Extroversion. In our case, we used a threshold of 0.5 for the extroversion attribute, as recommended by [escalante2020modeling].

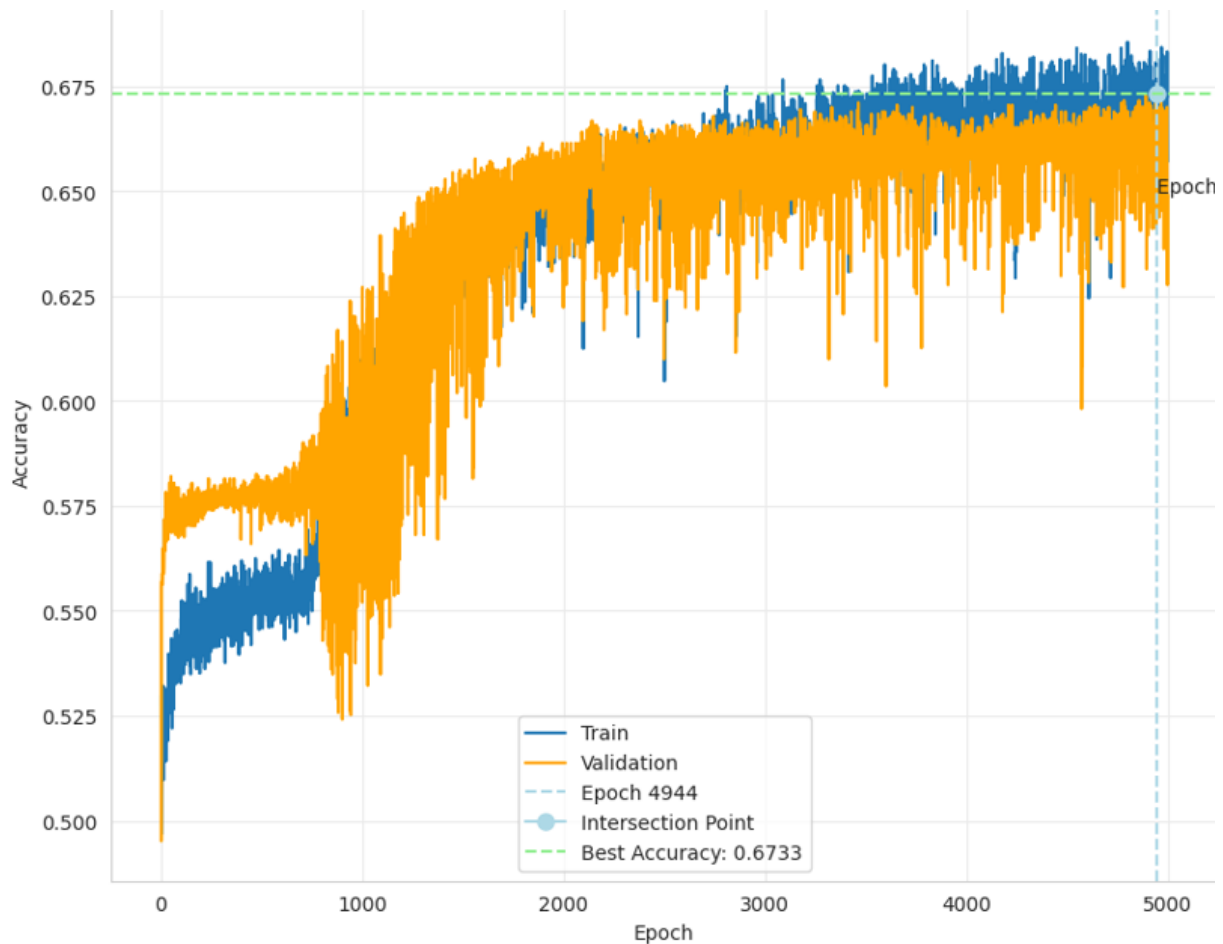


Figure 5: Accuracy obtained in the proposed approach. Source: Author.

Table 4 shows the number of videos per class in this initial experiment.

Personality Trait	Quantity	%
Extraversion	2.688	44.8
Non-Extraversion	3.312	55.2
Total	6.000	100.0

Table 4: Number of Videos for Extraversion class

Finally, following the training stage, we implemented the classification stage for a total of 2000 videos from the validation set. Figure 5 presents the Accuracy metric obtained from the both training and validation sets according to the number of epochs. We observe that the training convergence in 750 epochs demonstrates that the model is learning the patterns in the training data, from this moment on. This is positive, as it implies that the model can generalize and adjust its weights to achieve stable performance for unknown data. A validation AUC accuracy of 71,04% is very positive, demonstrating satisfactory performance, indicating that the model is generalizing well to data not seen during training. It is also possible to verify that validation accuracy followed training accuracy. When this happens

in a machine learning model, it is generally a positive sign as it suggests that the model is generalizing well, which is one of the main goals in model training.

Table 5 shows a comparison of the AUC accuracy results for related works, exclusively to the Extroversion personality trait, comparing to the accuracy obtained in the first experiment carried out in our proposed approach.

Paper	Approach	AUC
Proposed Approach	Transformer	71.04%
NJU-LAMDA	CNN	83.91%
Evolgen	LSTM	82.50%
DCC	ResNet	81.78%
Ucas	CNN	84.21%
BU-KNU	CNN	84.38%
Pandora	CNN	80.97%
Pilab	CNN+DAN	71.39%
Kaizoku	CNN	72.86%
ITU-SiMiT	CNN	44.10%

Table 5: Performance comparison: AUC results for the Extroversion personality Trait [ponce2016chlearn]

5 CONCLUSIONS

This work carried out extroversion personality trait classification using a Transformer architecture. A dataset widely adopted in the literature was used in the experiments, obtaining an initial AUC of 71,04%, with few adjustments in the hyperparameters of the classification model. For now, the goal was to validate the proposed pipeline. In a scenario of binary classification of personality traits, extroversion or not, Transformers emerge as a powerful and innovative tool. Its ability to capture complex contexts, learn meaningful representations and deal with image sequences opens opportunities to a deeper understanding of the relationship between video and personality. Therefore, in this context, the **main contributions** of our paper was providing a systematic literature review of the use of computational techniques in order to classify personality traits. In addition, we present a pipeline in a novel approach able to classify personality traits. As future works, the challenges we hope to face are related to three tasks: **1. Descriptors** - Use of new descriptors that combine facial landmarks; **2. Personality Traits** - Obtain satisfactory accuracy results involving the five great personality traits that make up the Big Five model; and, **3. Data Analysis** - Analyze distinct datasets, allowing personality traits to be identified on a large scale.

6 ACKNOWLEDGMENTS

"This study was financed in part by the Coordination of Improvement of Higher Education Personnel - Brazil (CAPES) - Finance Code 001".

The authors are grateful to the Renato Archer IT Center (CTI) Campinas, for its infrastructure support.

7 REFERENCES

- [costa1998trait] Costa, Paul T., and Robert R. McCrae. "Trait theories of personality." *Advanced personality*. Boston, MA: Springer US, pp.138-145, 1998.
- [thoresen2004big] Thoresen, Carl J., et al. "The big five personality traits and individual job performance growth trajectories in maintenance and transitional job stages." *Journal of applied psychology* 89.5, pp.835, 2004.
- [chioqueta2005personality] Chioqueta, Andrea P., and Tore C. Stiles. "Personality traits and the development of depression, hopelessness, and suicide ideation." *Personality and individual differences* 38.6, pp.1283-1291, 2005.
- [cattell1986number] Cattell, Raymond B., and Samuel E. Krug. "The number of factors in the 16PF: A review of the evidence with special emphasis on methodological problems." *Educational and Psychological Measurement* 46.3, pp.509-522, 1986.
- [abele2007agency] Abele, Andrea E., and Bogdan Wojciszke. "Agency and communion from the perspective of self versus others." *Journal of personality and social psychology* 93.5, pp.751, 2007.
- [ashton2007empirical] Ashton, Michael C., and Kibeom Lee. "Empirical, theoretical, and practical advantages of the HEXACO model of personality structure." *Personality and social psychology review* 11.2, pp.150-166, 2007.
- [sharpe2001revised] Sharpe, J. P., and S. Desai. "The revised Neo Personality Inventory and the MMPI-2 Psychopathology Five in the prediction of aggression." *Personality and Individual Differences* 31.4, pp.505-518, 2001.
- [tupes1992recurrent] Tupes, Ernest C., and Raymond E. Christal. "Recurrent personality factors based on trait ratings." *Journal of personality* 60.2, pp.225-251, 1992.
- [todorov2008understanding] Todorov, Alexander, et al. "Understanding evaluation of faces on social dimensions." *Trends in cognitive sciences* 12.12, pp.455-460, 2008.
- [kramer2010internal] Kramer, Robin SS, and Robert Ward. "Internal facial features are signals of personality and health." *Quarterly Journal of Experimental Psychology* 63.11, pp.2273-2287, 2010.
- [penton2006personality] Penton-Voak, Ian S., et al. "Personality judgments from natural and composite facial images: More evidence for a kernel of truth in social perception." *Social cognition* 24.5, pp.607-640, 2006.
- [little2007using] Little, Anthony C., and David I. Perrett. "Using composite images to assess accuracy in personality attribution to faces." *British Journal of Psychology* 98.1, pp.111-126, 2007.
- [sutherland2015personality] Sutherland, Clare AM, et al. "Personality judgments from everyday images of faces." *Frontiers in psychology* 6, pp.154136, 2015.
- [junior2019first] Junior, Julio CS Jacques, et al. "First impressions: A survey on vision-based apparent personality trait analysis." *IEEE Transactions on Affective Computing* 13.1, pp.75-95, 2019.
- [Ventura2017CVPRWorkshops] Ventura, Carles, David Masip, and Agata Lapedriza. "Interpreting cnn models for apparent personality trait regression." *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*. 2017.
- [gucluturk2017visualizing] Gucluturk, Yagmur, et al. "Visualizing apparent personality analysis with deep residual networks." *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2017.

- [devlin2018bert] Devlin, Jacob, et al. "Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805, 2018.
- [radford2018improving] Radford, Alec, et al. "Improving language understanding by generative pre-training.", 2018.
- [radford2019language] Radford, Alec, et al. "Language models are unsupervised multitask learners." OpenAI blog 1.8, pp.9, 2019.
- [vaswani2017attention] Vaswani, Ashish, et al. "Attention is all you need." *Advances in neural information processing systems* 30, 2017.
- [kitchenham2009systematic] Kitchenham, Barbara, et al. Systematic literature reviews in software engineering. A systematic literature review. *Information and software technology* 51.1, pp. 7-15, 2009.
- [zhang2017physiognomy] Zhang, Ting, et al. "Physiognomy: Personality traits prediction by learning." *International Journal of Automation and Computing* 14.4, 386-395, 2017.
- [rai2016bi] Rai, Nishant. "Bi-modal regression for apparent personality trait recognition." 2016 23rd International Conference on Pattern Recognition (ICPR). IEEE, 2016.
- [agastya2019systematic] Agastya, I. Made Artha, Dini Oktarina Dwi Handayani, and Teddy Mantoro. "A systematic literature review of deep learning algorithms for personality trait recognition." 2019 5th International conference on computing engineering and design (ICCED). IEEE, 2019.
- [yesu2021big] Yesu, Kolhandai, et al. "Big five personality traits inference from five facial shapes using CNN." 2021 IEEE 4th International Conference on Computing, Power and Communication Technologies (GUCON). IEEE, 2021.
- [escalante2020modeling] Escalante, Hugo Jair, et al. "Modeling, recognizing, and explaining apparent personality from videos." *IEEE Transactions on Affective Computing* 13.2, pp.894-911, 2020.
- [tan2019efficientnet] Tan, Mingxing, and Quoc Le. "Efficientnet: Rethinking model scaling for convolutional neural networks." *International conference on machine learning*. PMLR, 2019.
- [charlton2010principal] Charlton, Martin, et al. "Principal components analysis: From global to local.", 2010.
- [wei2017deep] Wei, Xiu-Shen, et al. "Deep bimodal regression of apparent personality traits from short video sequences." *IEEE Transactions on Affective Computing* 9.3, pp.303-315, 2017.
- [xu2021prediction] Xu, Jia, et al. "Prediction of the big five personality traits using static facial images of college students with different academic backgrounds." *Ieee Access* 9, pp.76822-76832, 2021.
- [helm2020single] Helm, Daniel, and Martin Kampel. "Single-modal video analysis of personality traits using low-level visual features." 2020 Tenth International Conference on Image Processing Theory, Tools and Applications (IPTA). IEEE, 2020.
- [ali2021classical] Ali, Waqar, et al. "Classical and modern face recognition approaches: a complete review." *Multimedia tools and applications* 80, pp.4825-4880, 2021.
- [zhang2022applications] Zhang, Yongyuan. "Applications of Google MediaPipe Pose Estimation Using a Single Camera." (2022).
- [samaa2022mediapipe] Samaan, Gerges H., et al. "Mediapipe landmarks with rnn for dynamic sign language recognition." *Electronics* 11.19, pp.3228, 2022.
- [ibrahim2023hybrid] Ibrahim, Ruba Talal, et al. "Hybrid intelligent technique with deep learning to classify personality traits". *International Journal of Computing and Digital Systems* 13.1, pp.231-244, 2023.
- [bounab2024towards] Bounab, Yazid, et al. "Towards job screening and personality traits estimation from video transcriptions". *Expert Systems with Applications* 238, pp.122016, 2024.
- [ponce2016chalearn] Ponce-Lopez, Victor, et al. "Chalearn lap 2016: First round challenge on first impressions-dataset and results". *Computer Vision - ECCV 2016 Workshops: Amsterdam, The Netherlands, October 8-10 and 15-16, Proceedings, Part III* 14, pp.400-418, 2016.