ISSN 1213-6972(print) ISSN 1213-6964 (online) *Volume 29, Number 1-2, 2021*

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.

Vaclav Skala – Union Agency

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***

ISSN 1213-6972(print) ISSN 1213-6964 (online) *Volume 29, Number 1-2, 2021*

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.

Vaclav Skala – Union Agency

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

Journal of WSCG

Board of Reviewers
2021

WSCG 2021

Board of Reviewers

Aguirre-Lopez,M, (Mexico)
Al-Ameen,Z, (Iraq)
Al-Darraji,S, (Iraq)
Baranoski,G, (Canada)
Barton,M, (Spain)
Benes,B, (United States)
Benger,W, (Austria)
Benziane,S, (Algeria)
Bouatouch,K, (France)
Bourke,P, (Australia)
Burova,I, (Russia)
Cabiddu,D, (Italy)
Cakmak,H, (Germany)
Cannavo,A, (Italy)
Carmo,M, (Portugal)
Dachsbacher,C, (Germany)
De Martino,J, (Brazil)
Debelov,V, (Russia)
Delibasoglu,I, (Turkey)
Drakopoulos,V, (Greece)
Drechsler,K, (Germany)
Durikovic,R, (Slovakia)
Dziembowski,A, (Poland)
Eid,A, (Egypt)
Eisemann,M, (Germany)
Etemad,K, (Canada)
Feito,F, (Spain)
Feld,S, (Netherlands)
Ferguson,S, (United Kingdom)
Fortunato Costa,B, (Brazil)
Fuenfzig,C, (Germany)
Gallucci,A, (Netherlands)
Galo,M, (Brazil)
Gangisetty,S, (India)
Gavrilova,M, (Canada)
Gdawiec,K, (Poland)
Gerhards,J, (France)
Giannini,F, (Italy)
Giraldo,J, (France)
Gobbetti,E, (Italy)
Goebel,M, (Germany)
Goncalves,A, (Portugal)
Grajek,T, (Poland)
Gudukbay,U, (Turkey)
Gunther,T, (Germany)
Ha,N, (Viet Nam)
Hast,A, (Sweden)
Hauenstein,J, (United States)
Hitzer,E, (Japan)
Horain,P, (France)
Hu,C, (Taiwan)
Hu,S, (China)
Hu,P, (Belgium)
Chaudhuri,D, (India)
Chmielewski,L, (Poland)
Jacek,K, (Poland)
Juan,C, (Spain)
Jurado,J, (Spain)

Justine,B, (France)
Karim,A.A.,S, (Malaysia)
Klosowski,J, (United States)
Komati,K, (Brazil)
Kumar,S, (India)
Kumar,S, (India)
Kurasova,O, (Lithuania)
Kurt,M, (Turkey)
Last,P, (Germany)
Lee,J, (United States)
Liu,S, (China)
Liu,B, (China)
Lobachev,O, (Germany)
Manzke,M, (Ireland)
Manzke,M, (Ireland)
Marco,C, (Brazil)
Marques,R, (Spain)
Mastmeyer,A, (Germany)
Matey,L, (Spain)
Max,N, (United States)
Meyer,A, (France)
Mieloch,D, (Poland)
MOHD SUAIB,N, (Malaysia)
Montrucchio,B, (Italy)
Muller,H, (Germany)
Muraleedharan,L, (India)
Nguyen,V, (Viet Nam)
Oliveira,J, (Portugal)
Pagnutti,G, (Italy)
Papagiannakis,G, ()
Papaioannou,G, (Greece)
Paquette,E, (Canada)
Parakkat,A, (Netherlands)
Pedrini,H, (Brazil)
Pereira,J, (Portugal)
Pérez-Díaz,S, (Spain)
Pintus,R, (Italy)
Platis,N, (Greece)
Radouane,K, (France)
Raffin,R, (France)
Ray,B, (India)
Reshetov,A, (United States)
Richardson,J, (United States)
Rodrigues,N, (Portugal)
Rodrigues,J, (Portugal)
Rojas-Sola,J, (Spain)
Rushmeier,H, (United States)
S P,R, (India)
Savchenko,V, (Japan)
Segura,R, (Spain)
Scheuermann,G, (Germany)
Sik-Lanyi,C, (Hungary)
Sirakov,N, (United States)
Skala,V, (Czech Republic)
Skopin,I, (Russia)
Szecsi,L, (Hungary)
Tas,F, (Turkey)
Tavares,J, (Portugal)
Thalmann,D, (Switzerland)
Todt,E, (Brazil)
Toler-Franklin,C, (United States)
Tourre,V, (France)
Trapp,M, (Germany)
Tuan,N, (United Kingdom)
Tytkowski,K, (Poland)
Wiegreffe,D, (Germany)
Wu,S, (Brazil)
Wuethrich,C, (Germany)
Yoshizawa,S, (Japan)
Zwettler,G, (Austria)

Journal of WSCG

Vol.29, No.1-2, 2021

Contents

ISSN 1213-6972 (print)
ISSN 1213-6964 (online)
Journal of WSCG
http://www.wscg.eu
Vol.29, No-1-2, 2021

Drone interactions within the field of Augmented Reality

Damia Fuentes Escote
Department of Computer Sciene
University of Colorado Colorado Springs
Colorado Springs, CO, 80918, USA
dfunetes@uccs.edu

Sudhanshu Kumar Semwal
Department of Computer Sciene
University of Colorado Colorado Springs
Colorado Springs, CO, 80918, USA
ssemwal@uccs.edu

ABSTRACT

Using drones and augmented reality paradigm, new forms of interactive algorithms has been created and proposed. We start with a first person view interaction where the drone mimics the movement of one person's head wearing a HMD so that movements of the head can be mapped to actions by the drones. We then provide two novel AR/VR applications of drones to create something similar to third person view in 2D and 3D. To get started, our first idea is to control a drone using head movements. The second application which we implemented is to provide an implementation where tangible platforms are used by the drone to react to the movements of the character. Finally our third implementaton if to create and AR world using real outdoor scenery and asking a drone to mimic a third person view combining the real scenery with a synthetic actor so that based on the synthetic actor movement the drone changes its behavior correctly in the real-word trying to provide a synchronized view of the real and synthetic word. There are three novel ideas providing a new form of interactions which will improve with drones functionality in future. Our implementation shows the feasibility of our idea as discussed in the paper.

Keywords

Augmented and Virtual Realty, New Paradigm using Drones.

1 INTRODUCTION

While present focus for drones is mostly for disasters, product delivery and public safety [1-6,10-15], we think there are other major applications that can be useful too. For example, to provide new camera angles, 3D AR games, web-contents generation, individual recreational activities and computationally generated social interactions, instead of drones just taking photos/videos and racing.

A drone is a relatively new technology that could also be used for video games as well. This paper proposes a new paradigm between drones, AR and user interaction. The drone camera could be used as the camera view of an AR, 3D or 2D video game. This way, video games could be based in real life environments that are visible to the drone, not to the phone or the user. This newer paradigm provides augmentation of a different kind, bringing the immediately reality surrounding the drones to our AR worlds. This idea is novel and we provide a successful implementation of our idea with video-sequences of our work. We focused on feasibility of controlling drones, and drones being able to play the game and show the user how the game can be played, these are simple games, still the feasibility of our proposal is shown specially that using drones Augmented Reality applications provide new experiences. In future, we are planning IRB studies by extending our drone based ideas, including our other works [7-9], to include drone generated scenes from far away and project them for interaction on large screens [7-9].

The environment would be a real park field seen from the drone camera, as shown in Figure 1. The character, Figure 2, would be superimposed to the real environment and it will be controlled by the user. As an example, imagine being a video game of this type as shown in Figure 3. The user may be situated anywhere in the surrounding environment, at his home, or at any place of the world. Notice that in an interaction of this type the user moves the character, not the camera view which is drone's view. Our work is different that all the drone interactions and games released up until now where the user controls the drone. To the best of our knowledge, no implementation exist where a drone is controlled automatically while the user controls an AR character, and that is precisely our goal in this paper.

The main idea of our paper is new form of interactions emerge where the view point is the drone's camera and the user controls an AR character that is graphically imposed over the video feed. Then, the drone *moves* automatically depending on how the user moves the character.

With this idea in mind, we now could create games and interactions in the streets, in a park, in our garden or inside our house. We could create them in 2D or 3D, online or offline, multiplayer or single player. We could have games where the game's behavior depends on how the real world is behaving in that exact moment. For example, if there is a physical car, even though not implemented by us, we imagine that the game could simulate an enemy behind it. If there is a tree, the enemy could appear from behind a tree. In other words, we are providing a new way of designing AR 2D/3D games based on what the interests of the game creator. Only a proof of concept is provided in this paper. A feasibility study of this new way of engaging AR-Drone Games is presented.

2 FPV WITH HEAD TRACKING CONTROL INTERACTION

We started by controlling drone movements with the head mounted display (**?**). The user wears VR goggles and sees what the drone sees through its' camera. Also called First Person View, or FPV. This implementation included a smartphone inside the VR goggles which using an own Android app will send orders to the drone via WiFi. For this idea, we used the DJI Spark drone. As seen in Figure 7-11 any movement of the users head, is executed by the drone.

2.1 Mimicking Drone Algorithm Implementation

In order to have roll, pitch and yaw of the android phone mimicked, two sensors are needed: accelerometer and magnetic field. A rotation matrix is created from the readings of these sensors and is then expressed as three orientation angles in degrees: roll, pitch and yaw. In another thread, these variables are retrieved and are sent to the drone every short period of time together with the throttle value. We send the change in degrees of the phone to the drone. In order to do this we first get the orientation of the drone and after that we sum the change difference of the phone yaw. Roll and pitch are treated a little bit differently as they are received in degrees and then converted to be sent as velocities. We have them first normalized from -1 to 1, and then finally we multiply by a pre-set maximum speed, see Algorithm 1.

2.2 VR Goggles feedback

Previous section explained how to control the drone using head movements. But the user also needs to see the live video feed of the drone to actually move the drone to where he wants to go. And he needs to see it in the VR goggles. Since we are using one live stream from the drone, the same image is displayed in both eyes. Therefore, depth of field can not be perceived still that will be something which is feasible in future as that will require two coordinated drones to keep same distance from each other equal to the inter-eye distance. See Algorithm 2.

Algorithm 1 Pseudocode for Mimicking drone control

1: $roll$, $pitch$, yaw, $lastpyaw \leftarrow$ 0.0,0.0,0.0,0.0
2: **loop**
3: $pacce \leftarrow$ get phone accelerometer reading
4: $pmagn \leftarrow$ get phone magnetometer reading
5: $proll$, $ppitch$, $pyaw \leftarrow$ calculate rotation matrix using $pacce$ and $pmagn$ readings and express it as three orientation angles: roll, pitch and yaw
6: **if** yaw is 0.0 **then**
7: $yaw \leftarrow$ get drone compass orientation (between -180 Â° and 180)
8: **else**
9: $yaw \leftarrow yaw - (lastpyaw - pyaw)$
10: **end if**
11: $lastpyaw \leftarrow pyaw$
12: $yaw \leftarrow ((yaw + 180)\%360) - 180$
13: $nroll$, $npitch \leftarrow$ normalize $roll$ and $pitch$ to fall between -1 and 1
14: $sroll$, $spitch \leftarrow nroll * maximumspeed$, $npitch * maximumspeed$
15: move drone according to yaw, $sroll$ and $spitch$
16: **end loop**

Algorithm 2 Pseudocode for user feedback

1: $screenx \leftarrow$ get mobile phone screen size height
2: $screeny \leftarrow$ get mobile phone screen size width
3: **loop**
4: $frame \leftarrow$ get actual frame from drone
5: $frameeye \leftarrow$ center fill crop $frame$ with height $screenx$ and width $screeny/2$
6: Update the UI by placing $frameeye$ both in the right and the left of the screen
7: **end loop**

2.3 Results: Mimiking Drones

The experience was a high-fidelity implementation, and to us, it felt like that we are actually the drone. This interaction could be fully utilized as FPV (First-Person-View) games like shooters. We also have some drawbacks as follows:

- Noticeable delay. Around 1 second between user turn and user sees the turn.
- Camera field of view of the DJI Spark is too narrow.

There are following improvements which could provide better interaction in future:

- Have a wide camera field of view. The one used in the DJI Spark is too narrow and it was difficult

to see what the drone has around it. With a wider field of view the experience would be more real and attractive.

- Make the drone have two cameras separated 2.3 inches simulating two eyes. Then put the live video feed of each camera to the right and the left of the phone screen. This way, the user will be able to perceive in 3D.
- Having real time information of the drone like the IMU in order to be able to add AR to the real time video feed.

3 IMPLEMENTING PLATFORM INTERACTION USING DRONE

In a platform game, the controlled character must jump and climb between suspended platforms while avoiding obstacles. In order to make this type of game to adapt to new AR and drones paradigm we thought that the platforms could be drawn physically in the real world. The virtual character would jump between those real platforms based on the drone's camera, while drone will try to have the character always in the center of the screen. For implementation, we used the DJI Tello and DJI Tello EDU drones, while simulating the output and controls from the computer.

3.1 Drone detecting the platforms

Our algorithm detects the tangible (drawn) physical platforms using the live camera video sequence of a drone. Platforms are drawn in black in a big white paper as seen in Figure 13-18, we have used black tape to indicate platforms.

We implemented a simple algorithm to detect the platforms in real-time so that we can determine if the character is in a platform or not at every frame. For every frame, we check if the foot of the character is over black or white pixels. If the foot is falling on black pixels this means that the character is over a platform. Also, if it is falling over white pixels this means that it is not on a platform. Pixels in black and white have a value from 0 to 255. 0 is the complete black while 255 is the complete white. In order to differentiate if a pixel is over blacks or whites we used a $THRESHOLD$ of 50. Meaning that any pixel with a value lower than 50 will be considered black. And therefore that pixel will be understood as a platform. The pseudocode for this can be found in Algorithm 3.

Algorithm 3 Pseudocode for detecting if character is in a platform

```
x0 ← get horizontal center position of the character
x1 ← get horizontal left position of the character
x2 ← get horizontal right position of the character
y ← get vertical bottom position of the character
if (x0,y), (x1,y), (x2,y) fall inside the frame and the pixel color in black and white is lower than THRESHOLD then
    return True
else
    return False
end if
```

3.2 Moving the character on Platforms

Once we know if the character is on a platform, we want the character to start moving and jumping. For that, we use inputs from the user indicating move left or move right and jump key-strokes. Move left and right will be always directly applied to the character. The jump input will only by applied if the character is on top of a platform. Finally, a gravity effect will be applied when the character is not on top of a platform. Also, the character stops to fall once it hits a platform. At that moment, its vertical velocity will be set up to 0. The pseudocode for this can be found in Algorithm 4.

Algorithm 4 Pseudocode for moving the character

```
movey ← 0
loop
    frame ← get actual frame from drone
    if user is pressing right then
        movex ← STEPS
    else if user is pressing left then
        movex ← −STEPS
    else
        movex ← 0
    end if
    isinplatform ← check if character is in platform
    if user pressed jump and isinplatform then
        movey ← −STEPS × JUMPACCELERATION
    else if movey greater or equal than 0 and isinplatform then
        movey ← 0
    else
        movey ← movey + gravity
    end if
    move character movex vertically and movey horizontally
end loop
```

3.3 Applying AR to a simple character

Right now the character is just on the screen, and its movement is 100% relative to the computer screen. When the drone moves, the camera and the background will also move; the character will stay at the same place relatively to the screen. The expected behavior would be that the character should be pinned relatively to the

game map. And the game map is the live video feed of the drone's camera. We have simulated this behavior as follows:

1. Track key-points and descriptors of every frame using an ORB detector (explained below).
2. Find the same key-points from the past and actual frame using a BF matching algorithm (explained below).
3. For each match get their movement in the x and y directions from the past frame to the actual one.
4. Order the matches in order of relevance so that very match has a distance that tells objectively how accurate it is.
5. Find outlier matches and remove them (how to find outliers is explained below).
6. Calculate the weighted average x and y movement of the resulting matches. The weights give more importance to the first matches and less to the last ones.
7. Move the character the same amount in x and y as the average calculated in the step before.

With this strategy, the character stays in the same game map place no matter how the drone moves. The pseudocode for this can be found in "Algorithm 5".

3.3.1 ORB detector

We used the FAST (Features from Accelerated Segment Test) [1] detection test for a corner detection method, which is used to extract feature points and to track and map objects in many computer vision tasks. BRIEF (Binary Robust Independent Elementary Features) use binary strings as an efficient feature point descriptor. It is very fast both to build and to match. ORB (Oriented FAST and Rotated BRIEF) [1] is a fusion of FAST keypoint detector and BRIEF descriptor with many modifications which enhances the performance.

3.3.2 Brute Force (BF) matcher and Outliers

BF matcher finds the closest descriptor in the second set by trying all possible combinations. An outlier is a data point that significantly differs from the other data points.

3.4 Moving the drone

In our algorithm, as the character moves through all the platforms, the drone follows the character so that the character is always at the center of the screen. Naturally the camera of the drone will not include all the game map, instead, just a part of it. Results of the drone hovering over the map with platforms and keeping the character at the center is provided as successful implementation of our algorithm. In order to achieve this, we check where the character is at every frame and depending on that the drone is moved up, down, left or/and right. For instance, if the character is in the left side, the drone is moved to the left. If the character is in the right-up corner, the drone is moved up and right. The drone is moved until the character falls in the center of the screen again. The pseudocode for this can be found in Algorithm 5-6.

Algorithm 5 Pseudocode for applying AR to the character

```
Initialize newkeypoints as newdescriptors as None
loop
    frame ← get actual frame from drone
    previouskeypoints ← newkeypoints
    previousdescriptors ← newdescriptors
    newkeypoints, newdescriptors ← run ORB detector on frame
    if previousdescriptors is not None then
        matches ← run BF matcher with newdescriptors and previousdescriptors
        matches ← sort matches in the order of their distance
        Initialize movementsx and movementsy as emtpy lists
        for match in first 20 matches do
            listmovx.insert(using match, previouskeypoints and newkeypoints calculate the vertical movement of that match)
            listmovy.insert(using match, previouskeypoints and newkeypoints calculate the horizontal movement of that match)
        end for
        Remove outliers of listmovx and listmovy
        Move character the weighted average of movementsx vertically and movementsy horizontally.
    end if
end loop
```

3.5 Discussion

With this approach, we have proved that this new paradigm is possible and feasible. See the video-sequences submitted with this paper. The user is able to control an AR character through the real time video feed of the drone. And the drone is moving through the real world in order to maintain the AR character in the center of the video feed all the time. We can see this in 13-16. The accuracy of the method is high. When the AR character goes too much towards the borders, the drone automatically moves in order to maintain it

Algorithm 6 Pseudocode for moving the drone

```
1:  loop
2:      leftrightvelocity ← 0
3:      updownvelocity ← 0
4:      characterx ← get character horizontal center position
5:      charactery ← get character vertical center position
6:      hhss ← half of the horizontal screen size
7:      hvss ← half of the vertical screen size
8:      if characterx is less than hhss − CENTEROFFSET then
9:          leftrightvelocity ← −DRONEVELOCITY
10:     else if characterx is bigger than hhss + CENTEROFFSET then
11:         leftrightvelocity ← DRONEVELOCITY
12:     end if
13:     if charactery is less than hvss − CENTEROFFSET then
14:         updownvelocity ← DRONEVELOCITY
15:     else if charactery is bigger than hvss + CENTEROFFSET then
16:         updownvelocity ← −DRONEVELOCITY
17:     end if
18:     move drone according to leftrightvelocity and updownvelocity
19: end loop
```

inside the screen. We can see this in 17-18. Hardly ever the character goes off the screen, even intentionally. So, our algorithm has been successful in that sense. For this algorithm we set up the velocities so that the drone is faster than the character. Even so, if that is the case and the character moves really fast and gets out of the screen, the drone will keep trying to move till the character falls inside the center of the screen. But while the character is outside the screen, it will not be able to stop on a platform because the platform to stop will be inside the field of view of the camera.

The delay is pretty immediate although there is some, but it is almost exactly the same delay as the delay between sending commands to the drone and see the results in the real time video feed. Which is around 0.5s. Our is interactive as we can see the drone reacting to the scene and moving accordingly, thus the implementation shows feasibility of our proposed AR paradigm. In our opinion, set-up time for this novel interaction is time consuming. Set-up time include: having to charge the drone, set up the drone, connecting to it's wifi and taking off. There are other parts like creating the map with papers and black tape, and the actual playing that were comfortable and fun. Playing the game felt like something novel and creative. We draw the platforms. We knew that we could change them, we could add/remove/change platforms while we were playing which provides customizability to our taste as to where we will have platforms. In some sense, we could design a new level of the game at our pleasure by simply moving the platform-tapes and have slightly different interaction very quickly. This is a novel AR paradigm for Games in our opinion.

This implementation is just the beginning of a very broad topic in VR-Drones and Augmented Computer Games area. There are so many different directions for research to further develop. One drawback of our implementation was that when the character goes out of the screen and lands on a platform, it wont stop at it because the game would not know that there is a platform there. A more sophisticated platform recognition algorithm would be needed instead of the one we use to show the feasibility.

4 THIRD PERSON VIEW INTERACTION USING DRONES AND AR

The third novel idea which we implemented tries to simulate a 3D third person view interaction. Third-person is a perspective view where the player can see the body of the controlled character. In this novel interaction, the users sees a 3D character on top of the real ground and controls it. Then, the drone moves according to how the character is moved. For implementing this idea, we used the DJI Tello and DJI Tello EDU drones and the output and controls are from the computer.

The input for this type of interaction is how the user moves the character (forward, backward, right, left, turning right or left, or any combination) and the output is how the drone should move in terms of yaw, pitch and roll velocities, i.e. how the selection of moving buttons change the drone view which in turn will change the camera views which the user sees and interactively effect the choices by the user. This is shown in our third demo submitted with this paper for review.

As seen in Figure 19 if the user moves the character forward, the drone moves forward, and the same for moving backwards. As seen in Figure 20 if the user moves the character to the left, the drone moves to the left, and the same for moving to the right. But the things get a little bit more tricky when rotating the character, because the drone has to now rotate about that character correctly. In that case, as seen in Figure 21 if the user rotates the character to the left, the drone has to orbit around the character. The drone has to rotate slight to the left (negative yaw) and at the same time move fast to the right (positive roll). So, both yaw and roll are applied at the same time. The workflow of this algorithm can be seen in Algorithm 7.

The implementation is an open-loop which means that no feedback is created. The character is pinned to the screen. Key-points and descriptors are not used. The reason why this idea has been implemented this way is

Algorithm 7 Pseudocode for 3rd person view drone control

```
loop
    cLeftRight, cForwBack, cRotLeftRight ← get character movement from the user (1, -1 or 0, which express no movement)
    roll, pitch, yaw ← 0.0,0.0,0.0,0.0
    if cForwBack is not 0 then
        pitch ← cForwBack ∗ SPEED
    end if
    if cLeftRight is not 0 then
        roll ← cLeftRight ∗ SPEED
    end if
    if cRotLeftRight is not 0 then
        yaw ← −cRotLeftRight ∗ ROTATIONSPEED
        roll ← roll + cRotLeftRight ∗ SPEED ∗ 2
    end if
    move drone according to yaw, roll and pitch
end loop
```

because in order to give feedback to the algorithm we would need to detect the ground plane, and that can not be done at this time as we have limited access to sensors of the DJI Tello or DJI Tello EDU drones.

4.1 Discussion

We have submitted our results the third video-sequence. Results show that this idea could benefit from having some feedback loop. Nevertheless, we showed that an algorithm of this type can be implemented and showed the feasibility of this idea where the user controls the AR character through the real time video feed of the drone; and the drone moves through the real world in order to simulate the camera to provide 3rd person view.

Moving the character forward was straight forward. One of the things we tested was to set up a real person next to the AR character and moved the character at the same speed as the person. We can see this in 22-25. It was possible to simulate this effect as if it was an AR game. We also tested rotating the character, as seen in 26-30. While the character is rotating, it is moving a little bit from its own piece of ground like if it was moving with a little circumference. With perfect parameters of drone rotating velocity and left/right velocity we could have the AR character all the time in the same piece of ground. But then, this parameters would be different for a different drone height which makes it cumbersome/impractical to implement at this time. A lot of fine-tuning of parameters were required in these experiements. Finally, we also tested moving the AR character sideways. We can see this in Figures 31-35. The results for this movement were very similar as moving the character forward: it was possible to simulate this effect as if it was an AR game. Some time lapse exists when the user moved the character move and the drone catches up with that event. There is a noticeable 1s of lag-time in this case. Interactions in which the character needs to rotate quickly can be improved in future. The velocity of the drone is limited. As the drone orbits around the character, bigger is the orbit faster the drone needs to move. To orbit around the character, the drone uses two velocities: rotating and moving sideways. Drones provide limited functionality and may need to improve in future.

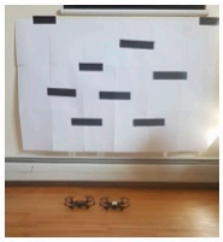


Figure 1: The game platforms/background

Figure 2: The environment

Figure 3: The character graphics

Figure 4: The AR game with the drone camera as the camera view

4.2 Future work

As we have seen, the main drawback of our open-loop implementation is when we want to rotate the character. One thing that we could do is to detect the key-points and descriptor of the piece of ground where the character is. Then, move the drone trying to maintain the character over those key-points and descriptors all the time. Also, detecting the ground-plane could be used in this type of interaction in the future. Future drones will need sensor data, together with the live video feed for that to be feasible.

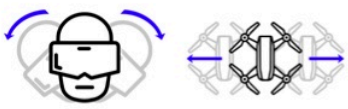


Figure 5: Tilting the head to the right, the drone should move to the right

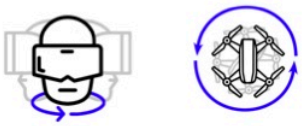


Figure 6: Rotating the head, the drone should rotate

Figure 7: Tilting the head to the front/back, the drone moves to the front/back

Figure 8: Mobile phone screen

Figure 9: Start of the program

Figure 10: Tilting the head to the right, the drone moves to the right

Figure 11: Rotating 90 degrees to the right, the drone rotates 90Â° to the right

Figure 12: Tilting the head to the front, the drone moves to the front

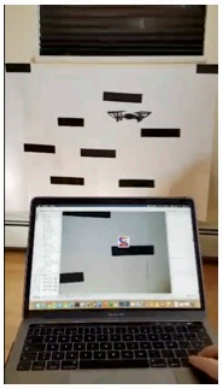


Figure 13: Platform results 1

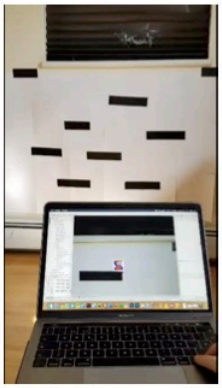


Figure 14: Platform results 2

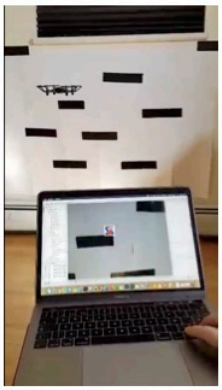


Figure 15: Platform results 3

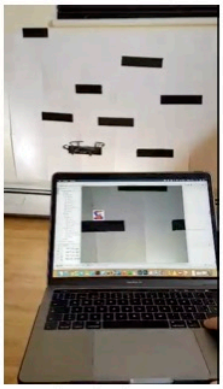


Figure 16: Platform results 4

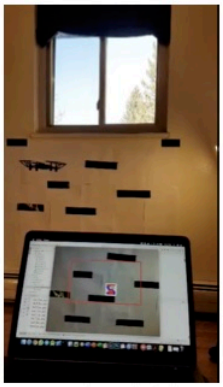


Figure 17: Platform results with drone moving bounds 1: Before jumping

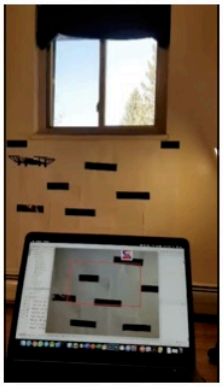


Figure 18: Platform results with drone moving bounds 2: While jumping going out of the rectangle

Figure 19: 3rd person view interaction: Moving character forward

Figure 20: 3rd person view interaction: Moving character right and left

Figure 21: 3rd person view interaction: Rotating the character to the left

Figure 22: 3rd person view interaction results forward 1

Figure 23: 3rd person view interaction results forward 2

Figure 24: 3rd person view interaction results forward 3

Figure 25: 3rd person view interaction results forward 4

Figure 26: 3rd person view interaction results rotating example 1

Figure 27: 3rd person view interaction results rotating example 2

Figure 28: 3rd person view interaction results rotating example 3

Figure 29: 3rd person view interaction results rotating example 4

Figure 30: 3rd person view interaction results rotating example 5

Figure 31: 3rd person view interaction results facing side 1

Figure 32: 3rd person view interaction results facing side 2

Figure 33: 3rd person view interaction results facing side 3

Figure 34: 3rd person view interaction results facing side 4

Figure 35: 3rd person view interaction results facing side 5

5 CONCLUSION

In this paper, we have explored the drone interactive applications within the field of Augmented Reality. Furthermore, new forms of interactive algorithms has been created and proposed, mainly focused in first and third view interactions. We have been successful in implementing three different basic introductory types of interactions. Our contributions is in the area of novel interactions paradigm for future drone applications in AR world/games. The first type of interaction was the drone VR FPV with head tracking control, which demonstrated how easy is to make the user feel as if they were the drone. The second one was the platforms interaction, which demonstrated that AR games where the user controls the AR character are possible, feasible, curious and fun. Finally, the third idea which we implemented was the third person view interaction, where we argue that this interaction can be used for many AR games in future. We are planning future IRB studies and detailed evaluation of our work in future research. Detailed evaluations strategies are being planned with several drones collaborating together to bring video-scenes to far-away audiences, and allowing interaction. Finally, we believe that much more would be done in this new and exciting novel area of AR with drones as the drones improve in future.

REFERENCES

1. E. Rublee, V. Rabaud, K. Konolige and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," 2011 International Conference on Computer Vision, 2011, pp. 2564-2571.

2. Calonder M., Lepetit V., Strecha C., Fua P. (2010) BRIEF: Binary Robust Independent Elementary Features. In: Daniilidis K., Maragos P., Paragios N. (eds) Computer Vision ? ECCV 2010. ECCV 2010. Lecture Notes in Computer Science, vol 6314. Springer, Berlin, Heidelberg.

3. Viswanathan, N., Kelty-Stephen, D.G. Comparing speech and nonspeech context effects across timescales in coarticulatory contexts. Atten Percept Psychophys 80, 316-324 (2018).

4. Yunchao Wei, Wei Xia, Min Lin, Junshi Huang, Bingbing Ni, Jian Dong, Yao Zhao and Shuicheng Yan, IEEE Transactions on Pattern Analysis and Machine Intelligence, HCP: A Flexible CNN Framework for Multi-Label Image Classification, 2016

5. Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun, Microsoft Research, Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks, 2016.

6. Bo Li, Junjie Yan, Wei Wu, Zheng Zhu, Xiaolin Hu, The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), High Performance Visual Tracking With Siamese Region Proposal Network, 2018.

7. Guilleum Budia Tirado, Sudhanshu Kumar Semwal, Automatic Parking Spot Detection System for Mobile Phnes using Drones and Deep Learning, Computer Science CSRN Notes, pp. 1-13, WSCG 2021 Proceedings, Plzen, CZ.

8. Guilleum Budia Tirado, Sudhanshu Kumar Semwal, Automatic Parking Spot Detection System with Image-Based machine learning, drones, and mobile platforms, MS thesis, UCCS, CO, USA. Advisor: SK Semwal, pp. 1-104, 2020.

9. Eric Marin Velazquez, Sudhanshu Kumar Semwal, Using Autonomous Drone Interactions towards Mobile Personal Spaced for Indoor Environments, Computer Science CSRN Notes, pp. 1-10, WSCG 2021 Proceedings, Plzen, CZ.

10. Punarjay Chakravarty, Klaas Kelchtermans, Tom Roussel, Stijn Wellens, Tinne Tuytelaars and Luc Van Eycken,IEEE International Conference on Robotics and Automation (ICRA), CNN-based Single Image Obstacle Avoidance on a Quadrotor, , pp.6369-6374, 2017.

11. Widodo Budiharto , Alexander A S Gunawan , Jarot S. Suroso , Andry Chowanda , Aurello Patrik and Gaudi Utama, International Conference on Computer and Communication Systems, Fast Object Detection for Quadcopter Drone using Deep Learning, pp. 192-195, 2018.

12. Z. Kaleem and M. H. Rehmani, "Amateur Drone Monitoring: State-of-the-Art Architectures, Key Enabling Technologies, and Future Research Directions," in IEEE Wireless Communications, vol. 25, no. 2, pp. 150-159, April 2018.

13. L. V. Santana, A. S. Brandão, M. Sarcinelli-Filho and R. Carelli, "A trajectory tracking and 3D positioning controller for the AR.Drone quadrotor," 2014 International Conference on Unmanned Aircraft Systems (ICUAS), 2014, pp. 756-767.

14. J. D. Renwick, L. J. Klein and H. F. Hamann, "Drone-based reconstruction for 3D geospatial data processing," 2016 IEEE 3rd World Forum on Internet of Things (WF-IoT), 2016, pp. 729-734.

15. Kang, H., Li, H., Zhang, J., Lu, X., Benes, B. (2018). FlyCam: Multitouch Gesture Controlled Drone Gimbal Photography. IEEE Robotics and Automation Letters, 3(4), 3717-3724.

ISSN 1213-6972 (print)
ISSN 1213-6964 (online)
Journal of WSCG
http://www.wscg.eu
Vol.29, No-1-2, 2021

New Methods and Novel Framework for Hypersurface Curvature Determination and Analysis

Jacob D. Hauenstein
The University of Alabama in Huntsville
301 Sparkman Drive
Huntsville, AL 35899
jacob.hauenstein@uah.edu

Timothy S. Newman
The University of Alabama in Huntsville
301 Sparkman Drive
Huntsville, AL 35899
newmant@uah.edu

ABSTRACT

New methods for hypersurface (that is, 3-dimensional manifold) curvature determination in volumetric data are introduced. One method is convolution-based. Another method is spline-based. Method accuracy is also analyzed, with that analysis involving comparison of the methods with each other as well as against two existing convolution-based methods. The accuracy analysis utilizes a novel framework that enables curvature determination method accuracy analysis via dynamically generated synthetic test datasets formed from continuous trivariate functions. Such functions enable accuracy analysis versus ground truth. The framework is also described here.

Keywords

Curvature, Volumetric Data, 3D Manifold, Hypersurface, Imaging

1 INTRODUCTION

Curvature, a translation- and rotation-invariant descriptor, describes the shape of the underlying data (e.g., a continuous function or discrete dataset) at a given point within the data. It is commonly used for many tasks in areas including quality control, healthcare, visualization, computer graphics, etc. Because curvature is a shape descriptor, uses in such areas often exploit curvature for model analysis [Hul12], object recognition, and general shape-based analysis tasks or tasks that benefit from its translation- and rotation-invariant shape descriptive capabilities, e.g., [Wan16, Lef18, Bib16, Bes86, Bel12, Sou16]. Curvature's use also extends to less obvious shape-based tasks, such as in initializing streamlines in vector field visualization [Wei02] and image denoising [Myl15], as well as in improving the accuracy of neural networks used for object detection [Wan16], surface reconstruction [Lef17], biometrics [Sya17], etc.

While it is possible to compute and utilize curvature within 3-dimensional (3D) manifolds, many tasks, including the ones just described, instead compute curvature within surfaces (2-dimensional manifolds), even when the underlying data from which curvature is measured is 3D (e.g., volumetric data). Methods to find curvatures on surface intersection curves also exist, with new methods recently reported for intersection curves of surfaces of high dimension [Özc21]. Other recent work considering curvature of 3D manifolds (also known as *hypersurfaces* [Mon92]) has shown promise for shape-descriptive tasks within volumetric data, especially in healthcare [Suz18, Hir18] and seismological applications [Ald14].

Varied measures of surface and hypersurface curvatures exist. The *principal curvatures* are among the most commonly used, including in aforementioned tasks. In the case of surfaces, there are two principal curvatures (denoted κ_1 and κ_2), and, for each point on the surface, these curvatures represent the minimum and maximum curvature at the point (ordered such that $\kappa_1 > \kappa_2$). In contrast, for hypersurfaces, there are three principal curvatures (denoted κ_1, κ_2, and κ_3) for each point on the hypersurface (ordered such that $\kappa_1 > \kappa_2 > \kappa_3$).

While many curvature tasks utilize the principal curvatures, these curvatures may not be computed exactly, because the data on which curvature is to be computed is real, sensed data (such as from a medical scanner). As a second derivative quantity, exact computation of the principal curvatures requires knowledge of the underlying continuous form of the surface or hypersurface to compute the necessary derivatives. In the case of real, sensed data, the continuous form is typically unknown and only discrete data is available; the principal curvatures must then be determined by other means (e.g., by estimating the necessary derivative quantities from the discrete data). Determination of curvature from discrete data is often inexact. Different determination methods may produce differing levels of accuracy on a given data

set, as recently noted in two studies of surface curvature determination methods [Hau20, Kro19].

Our focus in this work is to determine hypersurface principal curvatures from discrete volumetric datasets (henceforth simply *volumes*). We describe two new methods for determination of such principal curvatures. We also describe a framework that can be used to dynamically generate volumes, along with ground truth curvature values, to evaluate and compare the accuracy of each method's curvature determination. Furthermore, we use this framework to evaluate the accuracy of our two new methods for principal curvature determination (as well as two existing methods).

In the paper, Section 2 provides background on the mathematics of hypersurface curvature and related methods for determining hypersurface and surface curvature in volumes. Section 3 discusses our new methods for determining hypersurface curvature in volumes. Section 4 describes our testing and evaluation framework. Section 5 describes the volumes we generated using the framework. Section 6 details results obtained from these volumes. Section 7 concludes the work.

2 BACKGROUND / PREVIOUS WORK

First, we provide a brief overview of the mathematics for determining curvature of hypersurfaces within continuous volumetric data. The presentation here closely follows the formulations in both [Mon92] and [Ham94]. Next, we briefly describe the two existing methods for determining hypersurface curvature in volumes considered in our studies presented later as well as an existing study of the two methods. Finally, we present some details on methods for determining conventional surface curvature in volumes, as our two new hypersurface curvature determination methods, presented later, are inspired by these methods.

The mathematics notation we use is as follows. A grid (or sample) point within the volume is denoted as (x,y,z), where $0 \leq x < N_x$, $0 \leq y < N_y$, $0 \leq z < N_z$ for a volume of size $N_x \times N_y \times N_z$. The value at point (x,y,z) is denoted $f(x,y,z)$; f represents the underlying continuous function that generates the discrete volume. f_x represents the partial derivative of f in the x direction.

2.1 Hypersurface Curvature Mathematics

When the continuous form of the function f that generates the volume is known, the three principal curvatures of the hypersurface can be computed at each point within the volume fairly easily. They are simply, at each point in the volume, the eigenvalues of the matrix:

$$\frac{1}{l}\begin{bmatrix} f_{xx} & f_{xy} & f_{xz} \\ f_{xy} & f_{yy} & f_{yz} \\ f_{xz} & f_{yz} & f_{zz} \end{bmatrix} \begin{bmatrix} 1+f_x^2 & f_x f_y & f_x f_z \\ f_x f_y & 1+f_y^2 & f_y f_z \\ f_x f_z & f_y f_z & 1+f_z^2 \end{bmatrix}^{-1}, \tag{1}$$

where

$$l = \sqrt{1+f_x^2+f_y^2+f_z^2}. \tag{2}$$

We denote these eigenvalues λ_1, λ_2, and λ_3. As mentioned previously, computation of κ_1, κ_2, and κ_3 at any point requires knowledge of the first and second derivatives of f at the point, and, for many curvature-based tasks, the continuous form of f is unknown because the data is not continuous. Consequently, the first and second derivatives often must be estimated to determine κ_1, κ_2, and κ_3. All of the hypersurface curvature determination methods discussed in this work operate by estimating the necessary derivatives using varying approaches and then using those estimated derivatives to evaluate Eq. (1) (i.e., compute λ_1, λ_2, and λ_3).

2.2 Existing Hypersurface Curvature Methods and Comparative Studies

In our studies we compare our new methods with two existing methods for determining hypersurface curvature in volumes. First, we describe these two existing methods. The first, denoted **TEF**, uses convolution with kernels based on the Taylor Expansion in order to estimate derivatives (from which curvature is then computed). The second, denoted as **OPF**, uses convolution with a set of orthogonal polynomials in order to estimate derivatives.

2.2.1 TEF

The **TEF** method uses convolution filters derived from the Taylor Expansion. It determines hypersurface curvatures by first estimating all derivatives necessary for computing hypersurface curvature. These estimates are made through doing a series of convolutions along each axis of the volume. Once these convolutions are completed, the three principal curvatures are computed as λ_1, λ_2, and λ_3 using the estimated derivatives.

The convolution filters used by **TEF** were developed using a framework described by Möller et al. [Möl98]. The same framework was also previously used in a surface curvature method for volumes [Kin03].

For the **TEF** method, convolution filters are sampled from the Taylor Expansion using specified accuracy and continuity parameters. Previous studies of both hypersurface curvature and surface curvature methods that utilize the Möller et al. framework have used filters with C^3 continuity and fourth order accuracy [Hau19, Hau20], and we follow that usage in our own studies. When no noise or other errors are present, such filters allow for exact reconstruction of functions of degree 3 or lower.

When configured as described, the **TEF** first derivative filter is:

$$\{-\frac{1}{12}, \frac{2}{3}, 0, -\frac{2}{3}, \frac{1}{12}\}, \tag{3}$$

and the second derivative filter is:

$$\{-\frac{1}{24},\frac{1}{6},\frac{17}{24},-\frac{5}{3},\frac{17}{24},\frac{1}{6},-\frac{1}{24}\}. \quad (4)$$

2.2.2 OPF

The **OPF** method uses convolution filters derived from orthogonal polynomials to estimate all the necessary derivatives for determination of hypersurface curvature. Convolution with such filters is equivalent to a least squares fitting, and it thus gives identical results to that of a linear regression-based approach. Like the **TEF** approach, the **OPF** approach uses these derivative estimates to compute the three principal curvatures at each point in the volume as λ_1, λ_2, and λ_3 of Eq. (1).

OPF uses convolution filters of size N, where N is an odd number. The filters are generated by sampling orthogonal polynomials, denoted b_0, b_1, b_2, at N sample locations. b_0, b_1, and b_2 are given by:

$$b_0(\theta) = \frac{1}{N}, \quad (5)$$

$$b_1(\theta) = \frac{3}{M(M+1)(2M+1)}\theta, \quad (6)$$

$$b_2(\theta) = \frac{1}{P(M)}(\theta^2 - \frac{M(M+1)}{3}), \quad (7)$$

where $M = \frac{N-1}{2}$ and $P(M)$ is given by:

$$P(M) = \frac{8}{45}M^5 + \frac{4}{9}M^4 + \frac{2}{9}M^3 - \frac{1}{9}M^2 - \frac{1}{15}M. \quad (8)$$

θ denotes the locations at which each polynomial is sampled, where $\theta \in \{-\frac{N-1}{2},...,-1,0,1,...,\frac{N-1}{2}\}$. The b_0 filter smooths; b_1 estimates the first derivative; and b_2 estimates the second derivative.

Previous works have recommended $N = 7$ [Hau20], but, in our studies, we test varying values of N including $N = 3, N = 5, N = 7$, and $N = 9$. When considering **OPF** at a specific N value, we will use the notation **OPF***N*, where N is one of 3, 5, 7, or 9.

2.2.3 Comparative Studies

We are aware of only one existing study that comparatively evaluated methods for determination of hypersurface curvature in volumes. In that study [Hau19], two hypersurface curvature methods were compared. That study compared performance of the **TEF** and **OPF** methods and concluded that (1) **TEF** exhibits relatively low error (i.e., high accuracy) in curvature when no noise is present in the data, and (2) **OPF** generally is more accurate in the presence of noise.

2.3 Related Methods for Surface Curvature in Volumes

The two new methods for determining hypersurface curvature later described in this paper are analogous to two existing methods for determining surface curvature in volumes. Here, we describe those two existing surface curvature methods. Both of these methods determine surface curvature by first estimating derivatives at every point in the volume and then computing the curvature at every point in the volume using the estimated derivatives. The first described existing method uses B-Spline fitting to estimate surface curvatures. The second described method uses convolution with Deriche filters.

2.3.1 B-Spline

Soldea et al. [Sol06] described a method for determining surface curvature in volumes using B-Spline fitting. The approach used by Soldea et al. directly uses the volume data as the coefficients for a trivariate B-Spline. Since the trivariate B-Spline formed from the data is a continuous function that approximates the volume, the necessary derivatives of the B-Spline can be computed at each point, and these derivatives are used to compute curvature.

2.3.2 Deriche Filters

Monga et al. [Mon92] described a method for determining surface curvature in volumes using convolution with 3D Deriche filters. Such filters are of special interest because they are formulated such that they can be implemented as infinite impulse response (IIR) filters, making them conducive to implementation on some hardware [Der90]. The Monga et al. strategy uses three filters. One filter is used to smooth, one filter is used to estimate the first derivative, and one filter is used to to estimate the second derivative. From these estimated derivatives, curvature is determined.

3 NOVEL METHODS FOR HYPERSURFACE CURVATURE DETERMINATION

Here, we describe the two new methods we introduce here for determining hypersurface curvature in volumes.

3.1 B-Splines (BS)

Our first new method for determining hypersurface curvature in volumes, denoted **BS**, forms a B-Spline that provides a continuous form approximation of the input volume f. It is analogous to the approach used by Soldea et al. [Sol06] for determining surface curvature in volumes. The method determines curvature by first forming a continuous trivariate B-Spline, denoted $\hat{f}$, from the discrete volume. $\hat{f}$ is then used to compute derivatives from which curvature is calculated, and $\hat{f}$ uses a volume of form:

$$\hat{f}(x,y,z) = \quad (9)$$
$$\sum_{i=0}^{N_x}\sum_{j=0}^{N_y}\sum_{k=0}^{N_z} f(i,j,k)B_{i,s_x,\tau_x}(x)B_{j,s_y,\tau_y}(y)B_{k,s_z,\tau_z}(z),$$

where each $B_{l,s,\tau}$ term denotes the l-th B-Spline blending function of degree s over a knot sequence τ. Since $\hat{f}$ is continuous, its first and second derivatives can be calculated at each point, and they are used to determine curvature (as λ_1, λ_2, and λ_3) at each point.

In our experiments on the **BS** method, reported later, τ is a uniformly spaced knot vector, and we test several values of s, including $s = 2$, $s = 3$, $s = 4$, and $s = 5$. To our knowledge, these experiments are the first reports on the use of splines of higher than cubic degree in any variety of curvature determination. When considering **BS** at a specific s value, we will use the notation **BS***s*, where s is one of 2, 3, 4, or 5.

3.2 Deriche Filters (DF)

Our second new method for determining hypersurface curvature in volumes, denoted **DF**, uses convolution with the Deriche filters. It is analogous to the approach used by Monga et al. [Mon92] for determining surface curvature in volumes. The method determines curvature by first convolving with the three Deriche filters to estimate derivatives from which curvature is calculated. We denote the three filters as $\hat{f}_0$, $\hat{f}_1$, and $\hat{f}_2$. The filters are defined as continuous functions, and they are sampled to produce the parameters for the convolution with the volume. Convolution with the sampling of $\hat{f}_0$ provides smoothing, with the sampling of $\hat{f}_1$ provides estimates for the first derivative, and with the sampling of $\hat{f}_2$ provides estimates for the second derivative. Convolution is always applied along all three axes at each point, with the directional derivatives estimated using the relevant derivative estimation filters along any axis on which derivatives are to be estimated. In this convolution, the smoothing filter is applied along all other axes (i.e., other than the axis for the directional derivative). $\hat{f}_0$, $\hat{f}_1$, and $\hat{f}_2$ are defined as [Mon92]:

$$\hat{f}_0(t) = c_0(1+\alpha|t|)e^{-\alpha|t|}, \tag{10}$$

$$\hat{f}_1(t) = -c_1 t\alpha^2 e^{-\alpha|t|}, \tag{11}$$

$$\hat{f}_2(t) = c_2(1-c_3\alpha|t|)e^{-\alpha|t|}, \tag{12}$$

where c_0, c_1, c_2, and c_3 are defined by

$$c_0 = \frac{(1-e^{-\alpha})^2}{1+2e^{-\alpha}\alpha-e^{-2\alpha}}, \tag{13}$$

$$c_1 = \frac{-(1-e^{-\alpha^3})}{2\alpha^2 e^{-\alpha}(1+e^{-\alpha})}, \tag{14}$$

$$c_2 = \frac{-2(1-e^{-\alpha 4})}{1+2e^{-\alpha}-2e^{-3\alpha}-e^{-4\alpha}}, \text{ and} \tag{15}$$

$$c_3 = \frac{(1-e^{-2\alpha})}{2\alpha e^{-\alpha}}. \tag{16}$$

The α term controls the level of smoothing when estimating derivatives, with smaller values providing more smoothing. The optimal value for it depends on a number of factors, including the amount and type of noise in the volume. In our experiments on the **DF** method, reported later, we test a variety of α values, including $\alpha = 1.0$, $\alpha = 3.0$, $\alpha = 5.0$, and $\alpha = 10.0$. When considering **DF** at a specific α value, we will use the notation **DF**α, where α is one of 1.0, 3.0, 5.0, or 10.0.

4 VOLUME GENERATION AND EVALUATION FRAMEWORK

This section describes our framework for generating volumes and evaluating curvature determination method accuracy. This framework is used to generate volumes (including application of rotations, shifts, and scalings) from continuous form trivariate expressions, compute the ground truth curvatures in such volumes, and evaluate the accuracy of the hypersurface curvatures determined by each determination method on the generated volumes.

4.1 Overview

Our testing and evaluation framework enables rapid evaluation of hypersurface curvature determination method accuracy. It is capable of dynamically generating volumes of a user-specified number of samples directly from continuous form trivariate expressions. These volumes can be generated at arbitrary, user specified rotations, shifts, and scales. Because these volumes are dynamically generated at run-time, the framework allows for rapid testing of accuracy on many volumes. This characteristic makes the framework particularly useful for generation of many closely related volumes, such as generation of many different rotations of volumes from the same continuous expression, e.g., in order to evaluate the sensitivity of hypersurface curvature determination methods to changes in rotation. Because all volumes, regardless of rotation, shift, or scale, are generated directly from the continuous form expression, ground truth values are always available to use for accuracy evaluation of all determination methods.

The framework maintains a list of all curvature determination methods for which accuracy is to be evaluated, including any relevant parameter settings for each method. Each combination of determination method and parameters defines a *determination method set*. Because a single determination method can be included on this list at differing parameter settings (i.e., in different determination method sets), the framework makes it possible to rapidly test curvature determination methods at a variety of possible parameter settings.

Upon generation of a volume, the framework runs each hypersurface curvature determination method set in the list on the volume, collects the determined curvatures from each method, and computes the absolute and relative error for each point of interest within the volume.

The framework thus enables rapid generation of volumes and rapid testing of curvature determination methods.

4.2 Generation of Test Volumes

To generate a volume used for testing a list of determination method sets via our testing framework, a number of characteristics that describe the volume to be generated must be specified (i.e., input). We describe here those characteristics and how the volume is generated.

The number of samples in each of the x, y, and z directions for the generated volume, N_x, N_y, and N_z, must be specified. For our experiments, described later, we have set $N_x = N_y = N_z = 32$.

The scale and shift parameters must be specified for each axis. At a shift of 0.0 and scale of 1.0 on each axis, the volume will be generated by sampling the continuous form expression f at unit spacing along each axis, resulting in samples at locations ranging from 0 to $N_x - 1$ along the x-axis, from 0 to $N_y - 1$ along the y-axis, and from 0 to $N_z - 1$ along the z-axis. That is, the sample at location (x_s,y_s,z_s) within the discrete volume will contain the value $f(x_s, y_s, z_s)$. In contrast, using the same shift but a scaling of 0.5 on each axis, the volume will be generated by sampling the continuous form expression at a spacing of 0.5 along each axis starting from 0. That is, the sample at location (x_s,y_s,z_s) within the discrete volume will contain the value $f(0.5x_s, 0.5y_s, 0.5z_s)$. If a shift of -1 is also applied on each axis, the volume will be generated by sampling the continuous form starting at -1 on each axis. That is, the sample at location (x_s,y_s,z_s) within the discrete volume will contain the value $f(0.5x_s - 1.0, 0.5y_s - 1.0, 0.5z_s - 1.0)$. Thus, the scaling parameter selects the spacing used for sampling, and the shift parameter selects the region where the sampling takes place.

The continuous form expression f must be specified. It is specified using a standard C-style mathematical notation understood by the *libmatheval* library [Fre14], but with placeholder variables *XFIX*, *YFIX*, and *ZFIX* used in place of the x, y, and z variables in the expression. These placeholder variables are replaced by the framework with appropriately scaled versions of the x, y, and z variables using a string find-and-replace within the expression. For example, if all scales along each axis are set at 0.5, the continuous form expression specified as $XFIX + YFIX + ZFIX$ would be replaced with $(0.5{*}x) + (0.5{*}y) + (0.5{*}z)$. In general, *XFIX* is replaced with $(\rho_x{*}x)$, where ρ_x is the scaling along the x-axis. *YFIX* and *ZFIX* are replaced analogously.

The rotation parameters must also be specified. By default, no rotation is applied, and sampling of the continuous expression is done exactly along the x-, y-, and z-axis as described in the prior two paragraphs. However, the framework allows for arbitrary rotation through the origin by specifying the angle of rotation and the axis of rotation. When a rotation is specified, the framework generates a rotation matrix describing the requested rotation, and the rotation is applied by multiplying each sample point's coordinates by the rotation matrix prior to sampling the continuous expression. For example, if a rotation angle of 90 degrees and a rotation axis of $(0,0,1)$ is specified, the continuous expression's x-axis will be mapped to the resulting volume's y-axis (e.g., location (0,1,0) in the volume will contain the value sampled via $f(1,0,0)$). When generating the volume, rotation is applied prior to shifting or scaling.

Additionally, the level of Gaussian noise must be configured. It is specified by standard deviation, and the standard deviation value is used to generate Gaussian noise of mean zero and specified standard deviation.

Finally, the volume is generated by sampling the replaced expression, resulting in a volume generated from the continuous expression at the appropriate scale and shift. Because the continuous form expression is known for the underlying volume, the framework is able to compute the ground truth curvature at each point in the volume (utilizing *libmatheval*'s symbolic derivative capabilities [Fre14]).

4.3 Evaluation of Accuracy of Test Volumes

For evaluation of accuracy, a list of *points of interest* (POI) within each test volume is also specified for the framework. These POIs specify which sample locations within the volume that the framework will include when calculating the accuracy of each determination method set. The use of POIs, as opposed to calculating accuracy using every sample point in the volume, allows for finer control over the accuracy evaluation process. For example, determined curvatures are often very inaccurate at volume edges due to insufficient support for convolution kernels. But, POIs can be selected such that the edges of the volume are not considered in accuracy evaluation. Each POI may optionally be used as the center of rotation when a rotation is applied. This results in the rotated volumes and POIs being equal in number, where each volume is rotated about a different POI. These volumes can then be used to evaluate the sensitivity of a method set to rotation, because, for each rotation, the value of curvature at the POI about which the volume is rotated should be unchanged. If the determined curvature value, or the error in determined curvature value, changes at the POI about which the volume is rotated, then the method set is sensitive to rotation.

Once the POIs are specified, the framework computes the absolute difference between the expected and actual curvature results at each POI. Using the results from each POI within the volume, the framework produces a number of aggregate results across all POIs in the volume for each determination method set. These aggregate

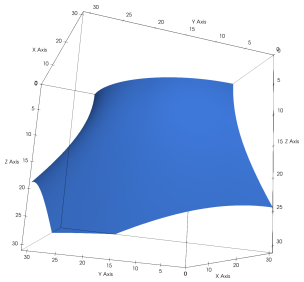


Figure 1: A rendering of a (2-D) isosurface of Genus3.

results include the arithmetic average of both the absolute errors and relative errors in each of κ_1, κ_2, and κ_3 for each determination method set on the volume.

5 GENERATED VOLUMES

Here, we describe the volumes generated by the framework and considered in our accuracy experiments. For all volumes, the framework is configured to generate volume datasets of size $32 \times 32 \times 32$, and the POIs selected are all the points within a $3 \times 3 \times 3$ neighborhood centered at the sample location $(15, 15, 15)$. This set of POIs ensures that none of the points used in accuracy evaluation are near the edge of the dataset.

The first volume is one generated from the quadratic polynomial expression:

$$x^2 + y^2 + z^2. \tag{17}$$

This expression has been used previously to generate volumes in other curvature studies (e.g., [Hau19, Hau20]). Because the 2D surfaces contained within the volume are spherical in shape, we will denote this expression (and the volumes it generates) *Spheres*. For volumes generated from this expression, we use a scaling of 1.0 and a shift of 0.0 on each axis. We consider the Spheres volumes at a variety of rotations (from -45 degrees to +45 degrees through the axis $(0, 0, 1)$), and both with and without noise. For each of these rotations, one rotated volume per POI is generated as described in Section 4.3.

The second volume is one generated from a higher degree polynomial expression:

$$(1 - (\frac{x}{6})^2 - (\frac{y}{3.5})^2)((x-3.9)^2 + y^2 - 1.44)$$
$$(x^2 + y^2 - 1.44)[(x+3.9)^2 + y^2 - 1.44] - 256z^2. \tag{18}$$

Because this expression exhibits a genus three 2D surface [Woo10], we will denote this expression (and the volumes it generates) *Genus3*. A rendering of a 2-D isosurface of Genus3 is shown in Fig. 1. For volumes generated from this expression, we use a scaling of $\frac{1}{64}$ and a shift of 0.0 on each axis. We consider the Genus3 volumes at a variety of rotations (from -45 degrees to +45 degrees through the axis $(0, 0, 1)$) and both with and without noise. For each of these rotations, one rotated volume per POI is generated as described in Section 4.3.

6 EXPERIMENTS AND RESULTS

In this section, we describe our experiments and results comparing the accuracy of our new **BS** and **DF** methods as well as the existing **TEF** and **OPS** methods. These results were all obtained using the framework described in Section 4, and the volumes generated for testing are as described in Section 5. We start with experiments on noise-free volumes.

6.1 Noise-Free Accuracy Results

First, we present an evaluation of how each method performs on the volumes when no noise is present. For these, the volumes were considered at 91 rotations (angles) (as described in Section 5), allowing simultaneous evaluation of the relative performance of each method on noise-free data and evaluation of how sensitive each method is to rotation (which should not change the curvature values or amount of error).

6.1.1 Spheres

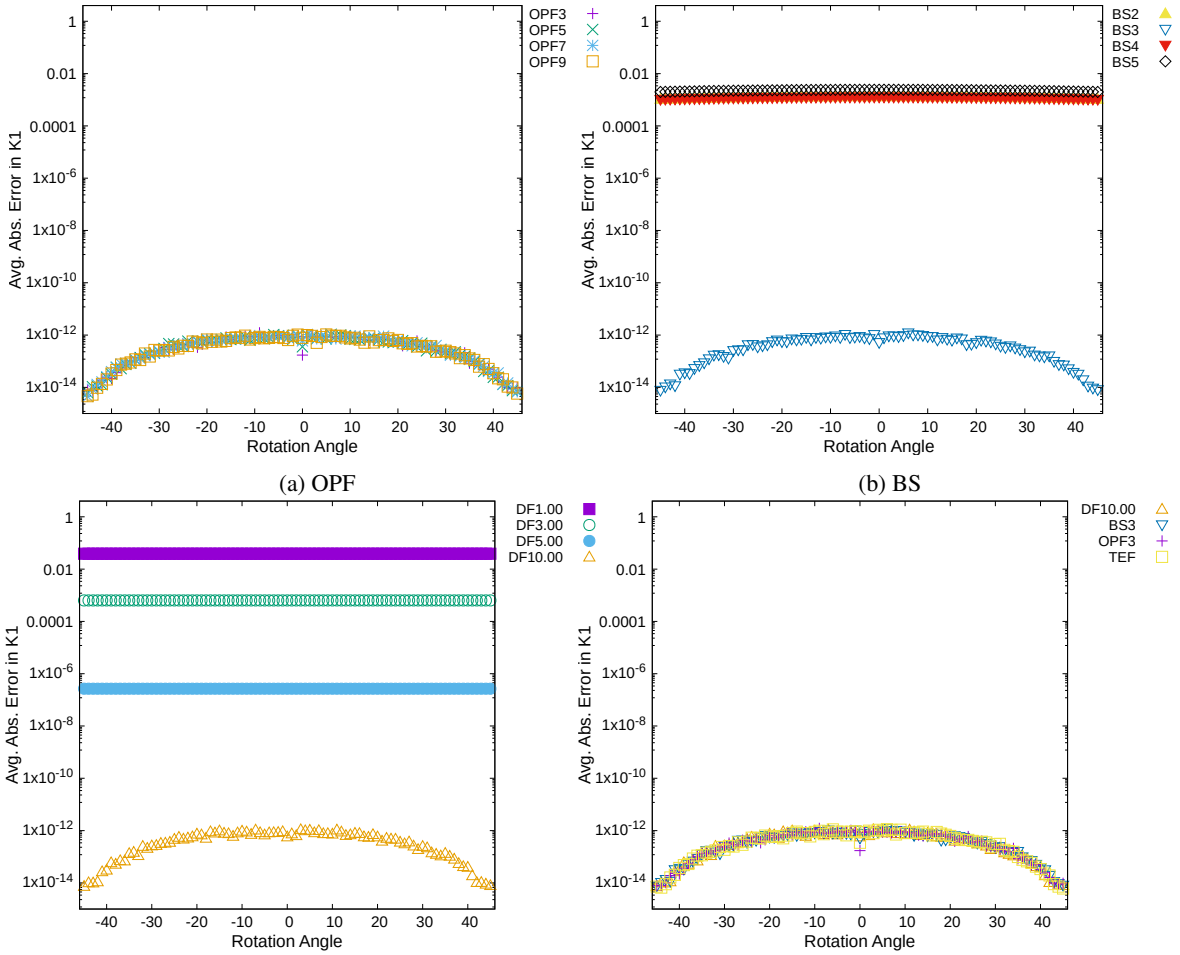
For the relatively simple Spheres volumes, most methods exhibit little error in determined curvatures. Fig. 2(a) shows the average absolute error in κ_1 for the **OPF** method at 91 rotations of Spheres volumes. While the amount of error in the determined curvatures does vary with the rotation, the amount of error at each rotation is extremely small and likely attributable to the limitations of floating point precision. Additionally, for these volumes, the filter size does not seem to have any appreciable impact on accuracy. This is likely because the hypersurface shapes within the dataset do not exhibit much variation over any of these filter sizes.

Fig. 2(b) shows the average absolute error in κ_1 for **BS** at the same 91 rotations. Here, **BS3** exhibits much lower error than the other degrees of **BS**. Like **OPF**, rotational sensitivity is low enough that it can be attributed to floating point precision.

Fig. 2(c) shows the average absolute error in κ_1 for the **DF** method at the same 91 rotations. Unlike **OPF**, here, accuracy strongly depends on the parameter settings. At lower α values, the error is much higher than at the higher α values, with $\alpha = 10.0$ exhibiting the smallest error. Only at $\alpha = 10.0$, where the errors are lowest, is it possible to see any change in error with rotation. Like in the **OPF** case, those changes are sufficiently small that they are likely explained by limitations of floating point precision.

Fig. 2(d) shows **OPF3**, **BS3**, **DF10.00** (the most accurate of each of **OPF**, **BS**, and **DF**) versus **TEF**. Here, the **BS3**, **OPF3**, **DF10.00**, and **TEF** methods all perform nearly identically and extremely accurately.

Due to space constraints, only plots of error in κ_1 are shown, but the results for κ_2 and κ_3 errors are very similar to these in relative accuracy of the methods and



(a) OPF (b) BS

(c) DF (d) OPF3, BS3, DF10.00, and TEF

Figure 2: Error levels in κ_1 on 91 different rotations of noise-free Spheres volumes

rotational sensitivity. For κ_2, the average absolute error across all the rotations is about 4.6×10^{-13} for **OPF** at all parameter settings, about 4.8×10^{-13} for **DF10.00**, about 4.6×10^{-13} for **BS3**, and about 4.8×10^{-13} for **TEF**. For κ_3, the average absolute error across all the rotations is nearly identical for **OPF** at all parameter settings, **DF10.00**, **BS3**, and **TEF**, with all of them having error of about 2.8×10^{-16}.

6.1.2 Genus3

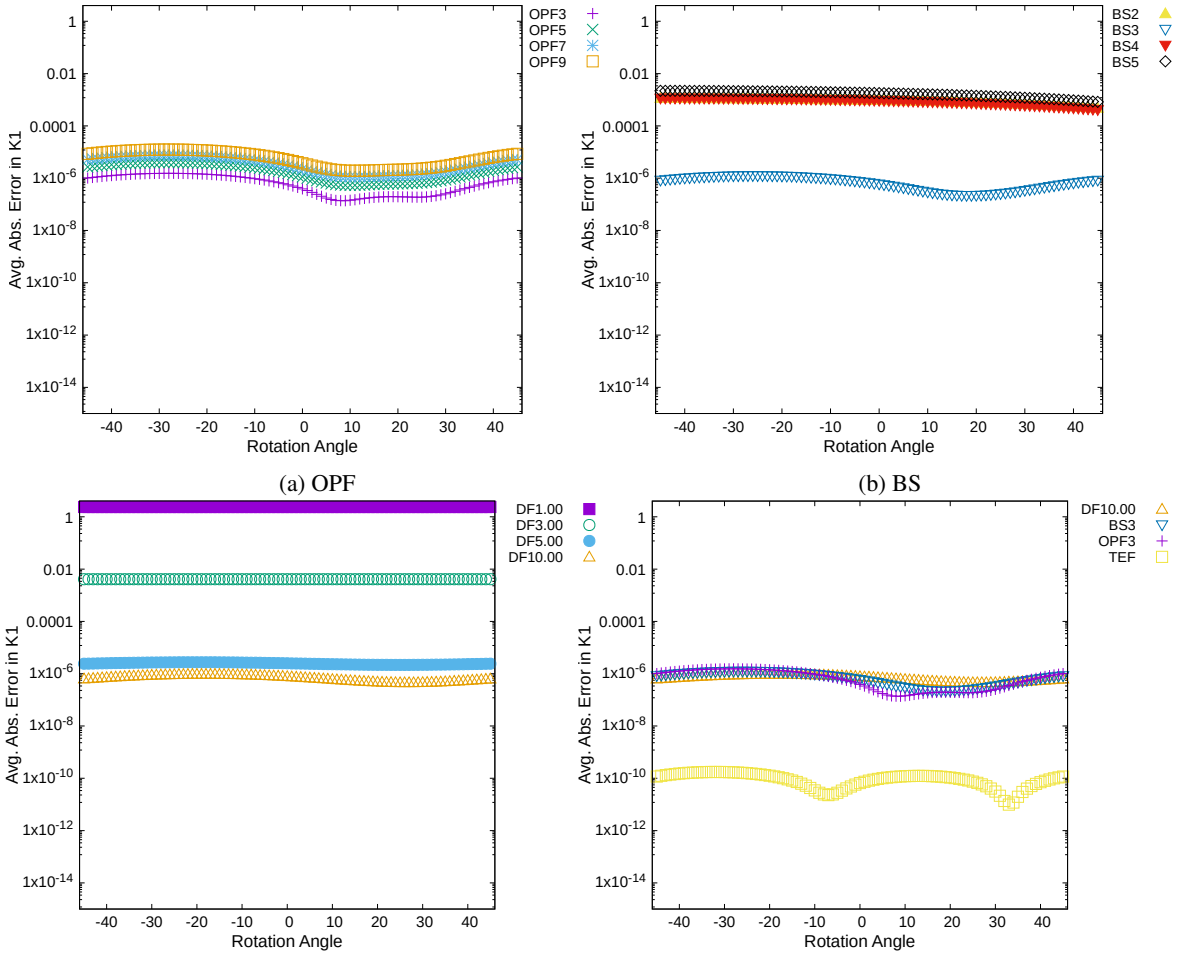
For Genus3 volumes, errors are notably higher than in the simpler Spheres volume. Fig. 3(a) shows the average absolute error in κ_1 for **OPF** at 91 rotations. Compared to Spheres, the change in error across rotation angles is much more pronounced. Whereas, in the case of Spheres, the changes in error with rotation were small enough that they could be explained by floating point limitations, the errors here are larger and likely indicate some amount of rotational sensitivity within **OPF**. Also unlike the Spheres, for these volumes, the filter size notably impacts the results, with **OPF3** exhibiting the least error.

Fig. 3(b) shows the average absolute error in κ_1 for **BS** at the same 91 rotations. Here, **BS3** again exhibits the lowest levels of error. **BS3** exhibits similar accuracy to **OPF3**, but it is more consistent in terms of exhibiting a similar error level across all rotations.

Fig. 3(c) shows the average absolute error in κ_1 for the **DF** method at the same 91 rotations. Again, accuracy strongly depends on the parameter settings for the **DF** method. $\alpha = 10.0$ again exhibits the smallest errors. **DF10.0** exhibits very small changes in error as rotation changes, suggesting that the **DF** methods are not particularly sensitive to rotation.

Fig. 3(d) shows **OPF3**, **BS3**, **DF10.00** (the most accurate of each of **OPF**, **BS**, and **DF**) versus **TEF**. Here, all methods exhibit low error, but **TEF** is much more accurate than the others.

Due to space constraints, only plots of error in κ_1 are shown, but the results for κ_2 and κ_3 errors are very similar to these in relative accuracy of the methods and rotational sensitivity. For κ_2, the average absolute error across all the rotations is about 7.5×10^{-7} for



(a) OPF (b) BS

(c) DF (d) OPF3, BS3, DF10.00, and TEF

Figure 3: Error levels in κ_1 on 91 different rotations of noise-free Genus3 volumes

OPF3, about 5.5×10^{-7} for **DF10.00**, about 6.5×10^{-7} for **BS3**, and about 1.9×10^{-11} for **TEF**. For κ_3, the average absolute error across all the rotations is about 1.2×10^{-7} for **OPF3**, about 1.0×10^{-7} for **DF10.00**, about 1.1×10^{-7} for **BS3**, and about 5.2×10^{-12} for **TEF**.

6.2 Noise-Added Accuracy Results

Next, we present an evaluation of how each method performs on the volumes when noise is added. For each of these studies, the same level of random (Gaussian) noise was added 30 times to both Spheres and Genus3, resulting in 30 volumes (or *trials*) of each with the same level of (different) random noise.

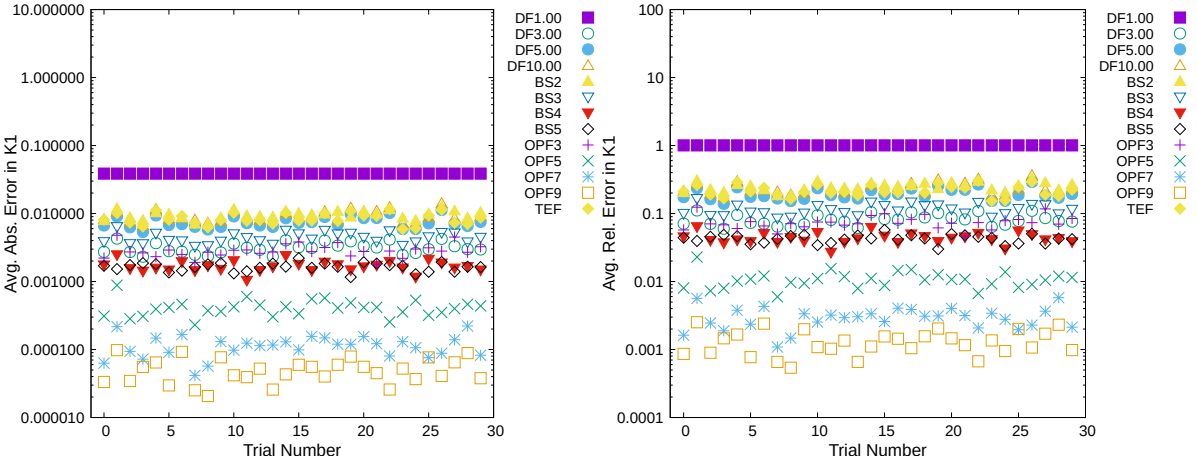
6.2.1 *Spheres*

Fig. 4(a) shows the average absolute error in κ_1 for the **OPF** method for 30 trials of noise with standard deviation of 0.25. In the presence of noise, **OPF** benefits from larger filter sizes, with **OPF9** exhibiting the lowest level of error in all trials, followed by **OPF7** and **OPF5**. The methods with the next lowest level of error are the B-Spline-based methods, with either **BS4** or **BS5** most accurate in most trials. This is a change from the noise-free cases, where **BS3** was consistently the most accurate, hinting that it may be beneficial to utilize higher degree splines in the presence of noise. Fig. 4(b) shows relative errors for the same volumes.

Again, due to space constraints, only results for κ_1 are shown, but relative performances are similar for κ_2 and κ_3. Because κ_3 is the smallest of the curvatures its relative error tends to be larger. For κ_2, the average absolute error across all the rotations is about 4.9×10^{-5} for **OPF9**, about 0.004 for **DF3.00**, about 0.003 for **BS4**, and about 0.008 for **TEF**. For κ_3, the average absolute error across all the rotations is about 1.3×10^{-8} for **OPF9**, about 1.2×10^{-6} for **DF3.00**, about 1.4×10^{-6} for **BS4**, and about 3.1×10^{-6} for **TEF**.

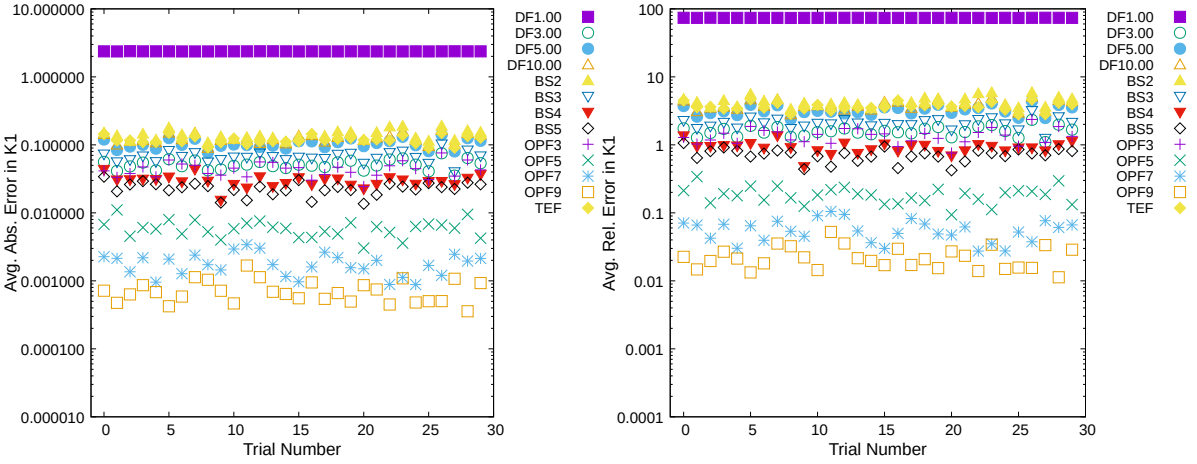
6.2.2 *Genus3*

Fig. 5(a) shows the average absolute error in κ_1 for the **OPF** method for 30 trials of noise with standard deviation of 0.25. Here, the results are similar to the Spheres



(a) Absolute Errors (b) Relative Errors

Figure 4: Error levels in κ_1 on noise-added Spheres



(a) Absolute Errors (b) Relative Errors

Figure 5: Error levels in κ_1 on noise-added Genus3

case. **OPF9** is best followed by **OPF7** and **OPF5**. Again, either **BS4** or **BS5** is next most accurate. Fig. 5(b) shows relative errors for the same volumes.

Again, due to space constraints, only results for κ_1 are shown, but relative performances are similar for κ_2 and κ_3. Due to the fact that κ_3 is the smallest of the curvatures, though, its relative error tends to be larger. For κ_2, the average absolute error across all the rotations is about 0.0002 for **OPF9**, about 0.1 for **DF2.00**, about 0.01 for **BS5**, and about 0.07 for **TEF**. For κ_3, the average absolute error across all the rotations is about 0.0005 for **OPF9**, about 0.03 for **DF2.00**, about 0.02 for **BS5**, and about 0.1 for **TEF**.

7 CONCLUSION

In this paper we have exhibited a framework for generating volumes and testing the accuracy of hypersurface curvature determination methods. In addition, we have introduced two new methods for determining such curvature. We have also compared these methods against two existing methods. One of these new methods, **BS**, uses B-Splines to estimate derivatives and then these estimated derivatives are used to determine the three hypersurface curvatures. The other method, **DF**, uses convolution with Deriche filters to estimate derivatives.

In summary, in our tests we found that **TEF** often exhibits the lowest error levels when no noise is present. When noise is present, **OPF** tends to exhibit much lower error levels than **TEF**, especially at larger filter sizes like **OPF9**. The new **BS3** method exhibits competitive accuracy on noise-free data.

In future works, we plan to introduce additional curvature determination methods and perform evaluations on a larger number of volumes and a larger number of parameter settings.

8 REFERENCES

[Ald14] G. Aldrich, A. Gimenez, M. Oskin, et al. Curvature-based crease surfaces for wave visualization. *Vision, Modeling & Vis.*, pp. 39–46, 2014.

[Bel12] A. Belyaev. Focal surfaces, skeletons, and ridges for shape analysis and processing. *Proc., 2012 Joint Int'l Conf. on Human-Centered Comput. Environ.*, HCCE '12, pp. 58–58, 2012.

[Bes86] P. J. Besl and R. C. Jain. Invariant surface characteristics for 3d object recognition in range images. *Comput. Vision, Graphics, Image Processing*, 33(1):33 – 80, 1986.

[Bib16] P. Bibiloni, M. González-Hidalgo, and S. Massanet. A survey on curvilinear object segmentation in multiple applications. *Pattern Recognition*, 60:949–970, Dec 2016.

[Der90] R. Deriche. Fast algorithms for low-level vision. *IEEE T-PAMI*, 12(1):78–87, Jan 1990.

[Fre14] Free Software Foundation. libmatheval (version 1.1.11), 2014. https://www.gnu.org/software/libmatheval/.

[Ham94] B. Hamann. Curvature approximation of 3D manifolds in 4D space. *Comput. Aided Geometric Design*, 11(6):621 – 632, 1994.

[Hau19] J. D. Hauenstein and T. S. Newman. Exhibition and evaluation of two schemes for determining hypersurface curvature in volumetric data. *J. of WSCG*, 27(2):121–129, 5 2019.

[Hau20] J. D. Hauenstein and T. S. Newman. Descriptions and evaluations of methods for determining surface curvature in volumetric data. *Computers & Graphics*, 86:52–70, 2 2020.

[Hir18] Y. Hirano. Categorization of lung tumors into benign/malignant, solid/GGO, and typical benign/others. K. Suzuki and Y. Chen, editors, *Artif. Intel. in Decision Support Systems for Diag. in Medical Imaging*, pp. 193–208. 2018.

[Hul12] R. Hulìk and P. Kršek. Local projections method and curvature approximation on 3D polygonal models. *Communic. Papers Proc., WSCG 2012*, pp. 223–230, 2012.

[Kin03] G. Kindlmann, R. Whitaker, T. Tasdizen, and T. Möller. Curvature-based transfer functions for direct volume rendering: methods and applications. *Proc., Vis. '03*, pp. 513 – 520, 2003.

[Kro19] M. Kronenberger, O. Wirjadi, and H. Hagen. Empirical comparison of curvature estimators on volume images and triangle meshes. *IEEE T-Vis. Comput. Graphics*, 25(10):3032–3041, 2019.

[Lef17] D. Lefloch, M. Kluge, H. Sarbolandi, et al. Comprehensive use of curvature for robust and accurate online surface reconstruction. *IEEE T-PAMI*, 39(12):2349–2365, Dec 2017.

[Lef18] L. Lefkovits and S. Lefkovits. Two-phase MRI brain tumor segmentation using random forests and level set methods. *Short Papers Proc., WSCG 2018*, pp. 152–159, 2018.

[Möl98] T. Möller, K. Mueller, Y. Kurzion, et al. Design of accurate and smooth filters for function and derivative reconstruction. *Proc., IEEE Symp. Volume Vis. 1998*, pp. 143 – 151, 1998.

[Mon92] O. Monga and S. Benayoun. Using partial derivatives of 3D images to extract typical surface features. Research Report RR-1599, INRIA, 1992.

[Myl15] M. Myllykoski, R. Glowinski, T. Karkkainen, and T. Rossi. A gpu-accelerated augmented lagrangian based l1-mean curvature image denoising algorithm implementation. *Full Papers Proc., WSCG 2015*, pp. 119–128, 2015.

[Özc21] B. Özcetin and M. Düldül. Curvature computations for the intersection curves of hypersurfaces in euclidean n-space. *Computer Aided Geometric Design*, 84(101954), 2021.

[Sol06] O. Soldea, G. Elber, and E. Rivlin. Global segmentation and curvature analysis of volumetric data sets using trivariate b-spline functions. *IEEE T-PAMI*, 28(2):265 – 278, 2006.

[Sou16] M. Soufi, H. Arimura, K. Nakamura, et al. Feasibility of differential geometry-based features in detection of anatomical feature points on patient surfaces in range image-guided radiation therapy. *Int'l J. of Comput. Assisted Radiology and Surgery*, 11(11):1993–2006, 2016.

[Suz18] H. Suzuki, Y. Kawata, N. Niki, et al. Automated assessment of aortic and main pulmonary arterial diameters using model-based blood vessel segmentation for predicting chronic thromboembolic pulmonary hypertension in low-dose ct lung screening. *Medical Imaging 2018: Comput.-Aided Diagnosis*, volume 10575, 2018.

[Sya17] M. A. Syarif, T. S. Ong, A. B. J. Teoh, and C. Tee. Enhanced maximum curvature descriptors for finger vein verification. *Multimedia Tools and Appl.*, 76(5):6859–6887, Mar 2017.

[Wan16] C. Wang and K. Siddiqi. Differential geometry boosts convolutional neural networks for object detection. *Proc., Diff. Geo. in Comp. Vision and Machine Learn. Work. (CVPRW Proc.)*, pp. 1006–1013, June 2016.

[Wei02] T. Weinkauf and H. Theisel. Curvature measures of 3d vector fields and their applications. *J. of WSCG*, 10(2):507–517, 2002.

[Woo10] B. A. Wood, J. K. Lee, M. Maskey, and T. S. Newman. Higher order approximating normals and their impact on isosurface shading accuracy. *Machine Graphics & Vision Intl. J.*, 19(2):201–221, 2010.

ISSN 1213-6972 (print)
ISSN 1213-6964 (online)
Journal of WSCG
http://www.wscg.eu
Vol.29, No-1-2, 2021

FastRIFE: Optimization of Real-Time Intermediate Flow Estimation for Video Frame Interpolation

Malwina Kubas
Warsaw University of Technology
Faculty of Electrical Engineering
ul. Koszykowa 75
00-662 Warszawa, Poland
malwina.kubas.stud@pw.edu.pl

Grzegorz Sarwas
Warsaw University of Technology
Faculty of Electrical Engineering
ul. Koszykowa 75
00-662 Warszawa, Poland
grzegorz.sarwas@pw.edu.pl

Abstract

The problem of video inter-frame interpolation is an essential task in the field of image processing. Correctly increasing the number of frames in the recording while maintaining smooth movement allows to improve the quality of played video sequence, enables more effective compression and creating a slow-motion recording. This paper proposes the FastRIFE algorithm, which is some speed improvement of the RIFE (Real-Time Intermediate Flow Estimation) model. The novel method was examined and compared with other recently published algorithms. All source codes are available at:
https://gitlab.com/malwinq/interpolation-of-images-for-slow-motion-videos.

Keywords

Video Interpolation, Flow Estimation, Slow-Motion

1 INTRODUCTION

Video frame interpolation is one of the most important issues in the field of image processing. Correctly reproduced or multiplied inter-frame information allows its use in a whole range of problems, from video compression [Wu18], through improving the quality of records, to generating slow-motion videos [Men20] or even view synthesis [Fly16]. Mid-frame interpolation performed in real-time on high-resolution images also increases the image's smoothness in video games or live broadcasts. With fast and accurate algorithms, we can reduce the costs associated with the construction of high-speed video cameras or provide services to users with limited hardware or transmission resources.

Algorithms of this class should deal with complex, nonlinear motion models, as well as with changing parameters of the real vision scene, such as shadows, changing the color temperature, or brightness. Conventional approaches are based on motion prediction and compensation [Haa93; Bao18], which are used in various display devices [Wu15].

Motion estimation is used to determine motion vectors in a block or pixel pattern between two frames. Block-based methods [Haa93] assume that pixels in a block have the same inter-frame shift and use search strategies [Zhu00; Gao00], and selection criteria [Haa93; Wan10] to obtain the optimal motion vector. Other methods of movement estimation are optical flow algorithms that estimate the motion vector for each pixel separately, which makes them more computationally expensive. In recent years there has been noticed significant progress in this area. New algorithms have been developed based on the optimization of variance [Bro04], searching for the nearest neighbors [Che13], filtering the size of costs [Xu17] and deep convolutional neural networks (CNN) [Ran17; Dos15]. However, very often, the quality of the obtained interpolations is burdened with an increase in the required computing resources and often limits their use in the case of users working on standard equipment or mobile phones.

Flow-based video frame interpolation algorithms achieve very good results [Jia18; Nik18; Bao19; Xue19; Hua21]. These solutions include two steps: warping the input frames according to approximated optical flows and fusing and refining the warped frames using CNN. Flow-based algorithms can be divided into forward warping methods and more often used back-warping methods. A common operation of these methods is to calculate bidirectional optical flow to generate an intermediate flow. These operations require a large number of computing resources and are not suitable for real-time scenarios.

DOI:10.24132/JWSCG.2021.29.3

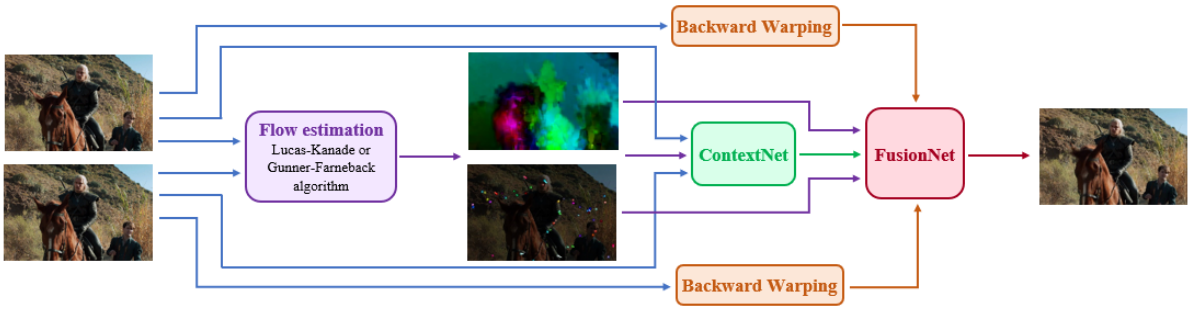


Figure 1: Architecture of the proposed video frame interpolation algorithm. We estimate the optical flow with analytical methods and use fine-tuned neural network models from RIFE: ContextNet and FusionNet.

This work aims to create an efficient video frame interpolation model, which can be implemented on many devices and run in real-time. Authors will compare state-of-the-art techniques, best in image interpolation field at the moment, and propose an improvement in terms of execution time in one of them. Algorithms which will be analyzed and compared are: MEMC-Net [Bao21], SepConv [Nik17], Softmax Splatting [Nik20], DAIN [Bao19] and RIFE [Hua21], all using different approaches.

2 RELATED WORK

The topic of video frame interpolation has been widely discussed in the research papers [Kim14; Dos15; Nik17; Bao21; Jia18]. The first groundbreaking method was proposed in [Lon16]. Long *et al.* were the first to use the convolutional neural network, which aimed to estimate the result frame directly. Next generations of image interpolation algorithms brought significantly better results, each using more extensive neural network structures.

Some algorithms, like SepConv [Nik17], were built and trained end-to-end using only one pass through the encoder-decoder structure. However the majority of methods use an additional sub-networks, of which the most important is the optical flow estimation. Having the information about pixels transition allows generating more accurate results and boost benchmark metrics. This approach was used e.g. in the MEMC-Net method, which combines both motion estimation, motion compensation and post-processing by four sub-networks [Bao21].

Bao *et al.* composed an improvement of MEMC-Net model called DAIN (Depth-Aware INterpolation) [Bao19], where the information of pixels depth was used, improving dealing with the problems related to occlusion and large object motion by sampling closer objects better. Depth estimation is one of the most challenging tasks in the image processing field of study, so the whole process takes a noticeable amount of time. Authors used a fine-tuned model called hourglass [Che16] network learned on the MegaDepth dataset [Li18]. DAIN method estimates the bi-directional optical flow using an off-the-shelf solution called PWC-Net [Xue19].

Another excellent flow-based method is Softmax Splatting [Nik20]. Niklaus and Liu proposed a solution of forward warping operator, which works conversely to backward warping and is used in most methods. Softmax Splatting uses an importance mask and weights all pixels in the input frame. Authors used fine-tuned PWC-Net for estimating optical flow, similarly to DAIN.

One of the newest algorithm is RIFE - Real-Time Intermediate Flow Estimation [Hua21], which is able to achieve comparable results on standard benchmarks when compared to previous methods but also works significantly faster. More details about this solution are presented in the next section.

3 PROPOSED SOLUTION

Most state-of-the-art algorithms generate intermediate frames by combining bi-directional optical flows and additional networks, e.g. estimating depth, which makes these methods unable to real-time work. Huang *et al.* proposed in RIFE [Hua21] a solution of simple neural network for estimating optical flows called IFNet and a network for generating interpolated frames called FusionNet.

The IFNet comprises three IFBlocks, each built with six ResNet modules and operating on increasing image resolutions. The output flow is very good quality and runs six times faster than the PWCNet and 20 times faster than the LiteFlowNet [Hui18]. After estimating the flow, coarse reconstructions are generated by backward warping and then the fusion process is performed. The process includes context extraction and the FusionNet with an architecture similar to U-Net, both consisting four ResNet blocks.

In this paper authors will analyze the RIFE model with different module for estimating optical flow. Although the IFNet gives excellent results and runs very fast, authors wanted to test analytical methods and compare the

Overlayed inputs | Optical flow by GF method | Optical flow by LK method | Interpolated frame with GF method | Interpolated frame with LK method | Ground-truth frame

Figure 2: Example of video frame interpolation results. Authors propose a simplified RIFE model with optical flow estimated using analytical methods: Gunnar-Farnebäck and Lucas-Kanade.

results of such a simplified model in terms of runtime and quality. This change can speed up the algorithm even more, making the whole process capable of running real-time on many more devices.

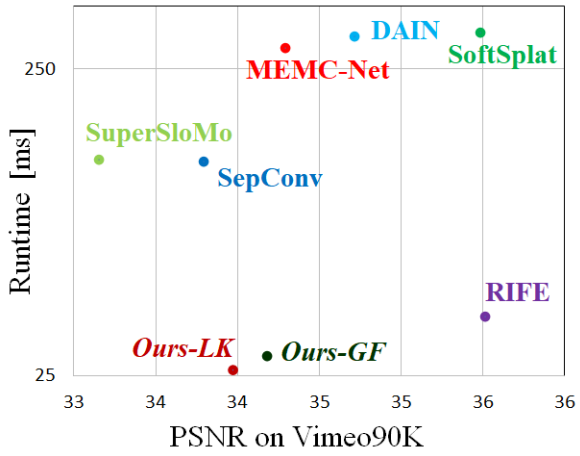


Figure 3: Speed and accuracy trade-off measured on RTX 2080 Ti GPU. Our models compared with previous frame interpolation methods: RIFE [Hua21], Softmax Splatting [Nik20], DAIN [Bao19], MEMC-Net [Bao21], SuperSloMo [Jia18] and SepConv [Nik17]. Comparison done on Vimeo90K test dataset. Please note the logarithmic runtime scale.

Authors replaced the IFNet part with analytical methods described below. Then, new FusionNet and ContextNet models were fine-tuned on the proposed solution. The full architecture is shown in Figure 1. Authors decided to test two of the most widely used algorithms, one generating dense flow and one which estimates sparse results. The example of optical flows and frame interpolation results are shown in Figure 2 and the speed/accuracy trade-off is presented in Figure 3.

3.1 Gunnar-Farnebäck algorithm

The Gunnar-Farnebäck method [Far03] gives dense results, which means that flow values are generated for every pixel. The algorithm detects changes in pixel intensity between two images using polynomial expansions and highlights pixels with the most significant changes.

The idea depends on an approximation of neighbourhood pixels in both frames by quadratic polynomials. Pixels' displacements are estimated from the differentiation of transforms of polynomials under translations. The algorithm is implemented by hierarchical convolutions what makes the process very efficient.

After the tests authors set the parameters of the Gunnar-Farnebäck (GF) method as follows:

- scale of image pyramids: 0.2,
- number of pyramid layers: 3,
- size of the pixel neighborhood: 5,
- size of averaging windows: 15×15.

The above sets give the best runtime to accuracy trade-off.

3.2 Lucas-Kanade algorithm

Dense methods can give very accurate results. However, sometimes the speed to accurate ratio might be more favorable for sparse methods. These algorithms calculate optical flow only for some number for pixels, extracted by the detector as feature points. To feed the results to the FusionNet it was needed to change the values from sparse to dense form.

Authors used Lucas-Kanade (LK) algorithm, which is the most commonly used analytical method for generating optical flow. It takes a 3×3 patch around the point and assumes all nine pixels will have the same flow. The solution is calculated with the least squares method. It is proven that the Lucas-Kanade works better with corner points, so for feature points detections authors used the Shi-Tomasi algorithm. The function calculates corner quality measure in each pixel's region using minimum eigenvalues and Harris Corner Detection.

Parameters which were used in the Lucas-Kanade algorithm are:

- size of search window: 15×15,
- number of pyramids: 1 (single level),

- termination criteria: epsilon and criteria count,

and for the Shi-Tomasi detector:

- maximum number of corners: 100,
- minimum accepted quality of corners: 0.1,
- minimum possible distance between the corners: 10,
- size of computing block: 7.

3.3 Runtime

Runtime comparison can be found in Table 1. IFNet works much faster than any other state-of-the-art method, but anyway optical flow can be obtained by analytical methods in less time. One of the important issues is that most of the standard flow methods have to run the estimation process twice if a bi-directional flow is desired. The results were measured again on RTX 2080 Ti GPU.

Method	Runtime [ms]
PWCNet	125
LiteFlowNet	370
IFNet	34
Ours - GF	**9**
Ours - LK	**7**

Table 1: Runtime comparison of optical flow estimation algorithms

4 EXPERIMENTS

This section will provide evaluation results, comparing the last and best methods with our proposal. Comparison is made based on quantitative and visual results using common metrics and datasets as well as algorithms runtime.

4.1 Learning strategy

Our solution was trained on Vimeo90K dataset with optical flow labels generated by the LiteFlowNet [Hui18] in PyTorch version [Nik19]. Authors used the AdamW optimizer, the same as used in RIFE [Hua21].

4.1.1 Loss function

To address the problem of training the IFNet module, Huang *et al.* proposed a solution of leakage distillation schema, which compares the network result with the output of the pre-trained optical flow estimation method. The training loss is a linear combination of reconstruction loss L_{rec}, census loss L_{cen} and leakage distillation loss L_{dis}. Our solution was trained using the same loss function:

$$L = L_{rec} + L_{cen} + \lambda L_{dis}, \tag{1}$$

with the weight of leakage distillation loss 10 times smaller than the two others ($\lambda = 0.1$).

4.1.2 Training dataset

Vimeo90K is a video dataset containing 73,171 frame triplets with 448×256 pixels image resolution [Xue19]. This dataset was used as a training set in all compared methods.

4.1.3 Computational efficiency

All analyzed algorithms are implemented in PyTorch [Pas17] using CUDA, cuDNN, and CuPy [Che14]. The proposed solutions was implemented using OpenCV functions, and RIFE model was fine-tuned on Nvidia RTX 2080 Ti GPU with 11GB of RAM for 150 epochs which took 15 days to converge.

4.2 Evaluation datasets

For evaluation, the following publicly available datasets have been used:

4.2.1 Middlebury

Middlebury contains two subsets: Other with ground truth interpolation results and Evaluation, which output frames are not publicly available [Bak07]. It is the most widely used dataset for testing image interpolation algorithms. The resolution of frames is 640×480 pixels.

4.2.2 UCF101

UCF101 provides 379 frame triplets from action videos with 101 categories [Soo12] [Liu17]. Images resolution is 256×256 pixels.

4.2.3 Vimeo90K

Vimeo90K was used for training, but it contains also the test subset - 3,782 triplets with 448×256 pixels image resolution [Xue19].

4.3 Metrics

The following metrics have evaluated each algorithm. PSNR (Peak Signal-to-Noise Ratio):

$$PSNR(i,j) = 10 \cdot \log_{10} \frac{255^2}{MSE(i,j)}. \tag{2}$$

SSIM (Structural Similarity, l - luminance, c - contrast and s - structure):

$$SSIM(i,j) = l(i,j) \cdot c(i,j) \cdot s(i,j). \tag{3}$$

IE (Interpolation Error) is the arithmetic average of a difference between the interpolated image and the ground truth frame:

$$IE(i,j) = \overline{|i-j|}, \tag{4}$$

where i is interpolated frame and j is the ground truth image [Wan04]. On UCF101 and Vimeo90K datasets we used PSNR and SSIM (higher values mean better results), on Middlebury Evaluation and Other we evaluated IE (lower values mean better results). This set of benchmarks is used in the majority of frame interpolation articles and datasets websites.

Figure 4: Qualitative comparison on UCF101 dataset

Method	Middlebury IE	UCF101 PSNR	UCF101 SSIM	Vimeo90K PSNR	Vimeo90K SSIM	Parameters (million)
SepConv	2.27	34.78	0.967	33.79	0.970	21.6
MEMC-Net	2.12	35.01	0.968	34.29	0.970	70.3
DAIN	2.04	34.99	0.968	34.71	0.976	24.0
SoftSplat	1.81	35.10	0.948	35.48	0.964	7.7
RIFE	1.96	35.25	0.969	35.51	0.978	9.8
Ours: FastRIFE - GF	**2.89**	**34.84**	**0.968**	**34.18**	**0.968**	**4.1**
Ours: FastRIFE - LK	**2.91**	**34.26**	**0.967**	**33.97**	**0.967**	**4.1**

Table 2: Quantitative comparison of described methods on Middlebury Other, UCF101 and Vimeo90K test

4.4 Quantitative comparison

Table 2 shows a comparison of the results obtained on datasets: Middlebury Other, UCF101 and Vimeo90K, red marks the best performance, blue marks second best score. Results of evaluation on Middlebury benchmark are not available yet.

The proposed model performs comparably to other methods, scoring very similar results in both PSNR and SSIM. The most significant gap is between the results of Interpolation Error in Middlebury Other dataset. FastRIFE generates better values than the SepConv model in UCF101 and Vimeo90K benchmarks and worse results than other algorithms, but a significant reduction is shown on the parameters column.

Evaluation of both optical flow methods gives similar results. However, the model which uses the Gunnar-Farnebäck algorithm performs slightly better in every benchmark, which means that the FusionNet behaves better with dense flow estimation.

4.5 Visual Comparison

Figure 4 shows examples of frame estimation with comparison made on the DAIN and RIFE methods. These images show dynamic scenes, which are more challenging to analyze for frames interpolation methods. The resolution of images is small (256×256), but both GF and LK methods generate acceptable results, similar to DAIN. The worst smudging is shown on RIFE results, which works better with high-resolution videos.

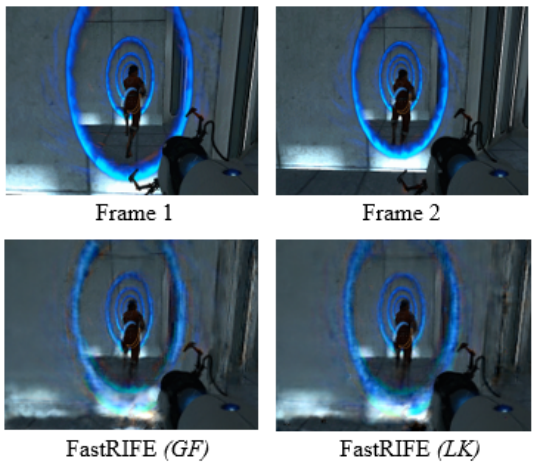


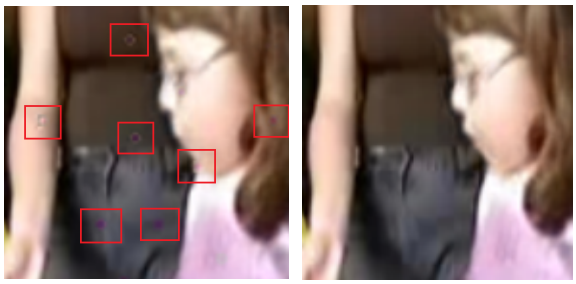
Figure 5: Visual comparison on HD frames

Method	Inference time on 448×256 image [ms]	Time of learning one epoch [h]
SepConv	125	-
MEMC-Net	293	-
SoftSplat	329	-
DAIN	320	9
RIFE	39	4.5
Ours: FastRIFE - GF	**29**	**2.4**
Ours: FastRIFE - LK	**26**	**2.4**

Table 3: Runtime comparison of described methods tested on 448×256 image (device: RTX 2080 Ti GPU)

The results of high-resolution image interpolation are shown in Figure 5. Both FastRIFE models generate very blurry frames. It is probably caused by including only small resolution videos in the training process and setting constant parameters when calculating optical flow. The issue could be repaired as future work. Other methods: DAIN and RIFE work well with HD resolution frames.

The advantage of the proposed models is the size of occupied memory. For most methods, analyzing HD images results in an out-of-memory error on our GPU, while RIFE allocates only 3 GB and FastRIFE only 1.5 GB of RAM.



FastRIFE - LK FastRIFE - GF

Figure 6: Sparse optical flow issue. Sometimes output frames suffers from spots visible in zoom

One important issue related to using the sparse optical flow method (Lucas-Kanade) is overlapping feature points as visible spots, as shown in Figure 6. The spots are noticeable after zooming small resolution images. Such a problem can be reported only for a model with the LK method. FastRIFE which uses the Gunnar-Farnebäck algorithm is this issue free.

4.6 Runtime Comparison

The usage of analytical methods and simplified RIFE model caused that FastRIFE performs favourably faster than any other state-of-the-art algorithm. Our models have 4 million parameters, which is the smallest neural network structure from all video frame interpolation methods. The runtime comparison is shown in Table 3.

Compared to any other model, except for RIFE, our solution runs up to 10 times faster. FastRIFE has slightly better benchmark results from SepConv, but runtime has been compressed from 125 to less than 30 milliseconds. The replacement of the optical flow module in RIFE saved 25% of the time needed to interpolate frames. As shown in Table 3, also the time of learning one epoch was significantly reduced.

5 CONCLUSION

In this paper, we proposed a solution to RIFE model simplification in terms of optical flow estimation. FastRIFE uses well-known analytical methods instead of additional neural network modules, which results in excellent runtime improvement with an acceptable slight drop in the quality of output frames.

Another advantage of model compression is the reduction of memory usage. Thanks to the small number of parameters, FastRIFE is able to generate output frames with the allocation of max 1.5 GB of RAM in the case of HD videos. Low memory consumption with short execution time makes the model possible to implement on many devices, including mobile phones.

The FastRIFE algorithm was analyzed with two method types for estimating optical flow: sparse (Lucas-Kanade) and dense (Gunnar-Farnebäck). Sparse flow performs faster, but better-quality results were always generated with the GF method. FastRIFE works appropriately in small resolution images but introduces prominently blurry regions when interpolating HD videos.

Future work that may be performed includes the improvement of the FastRIFE model in order to obtain better results on high resolution videos. It can probably be achieved by including HD images in the training dataset or by adjusting optical flow algorithm parameters on the fly based on video resolution or other criteria. Our proposed method can result in future research on increasing the efficiency of video frame interpolation algorithms.

REFERENCES

[Bak07] S. Baker et al. “A Database and Evaluation Methodology for Optical Flow”. In: *2007 IEEE 11th International Conference on*

Computer Vision. 2007, pp. 1–8. DOI: 10.1109/ICCV.2007.4408903.

[Bao18] W. Bao et al. "High-Order Model and Dynamic Filtering for Frame Rate Up-Conversion". In: *IEEE Transactions on Image Processing* 27.8 (2018), pp. 3813–3826. DOI: 10.1109/TIP.2018.2825100.

[Bao19] W. Bao et al. "Depth-Aware Video Frame Interpolation". In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 3698–3707. DOI: 10.1109/CVPR.2019.00382.

[Bao21] W. Bao et al. "MEMC-Net: Motion Estimation and Motion Compensation Driven Neural Network for Video Interpolation and Enhancement". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 43.3 (2021), pp. 933–948. DOI: 10.1109/TPAMI.2019.2941941.

[Bro04] T. Brox et al. "High Accuracy Optical Flow Estimation Based on a Theory for Warping". In: *Computer Vision - ECCV 2004*. Ed. by T. Pajdla and J. Matas. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 25–36. ISBN: 978-3-540-24673-2.

[Che13] Z. Chen et al. "Large Displacement Optical Flow from Nearest Neighbor Fields". In: *2013 IEEE Conference on Computer Vision and Pattern Recognition*. 2013, pp. 2443–2450. DOI: 10.1109/CVPR.2013.316.

[Che14] S. Chetlur et al. "CuDNN: Efficient primitives for deep learning". In: *International Journal of Computer Vision*. 2014. DOI: arXiv:1410.0759.

[Che16] W. Chen et al. "Single-Image Depth Perception in the Wild". In: *Advances in Neural Information Processing Systems 29*. Ed. by D. D. Lee et al. Curran Associates, Inc., 2016, pp. 730–738.

[Dos15] A. Dosovitskiy et al. "FlowNet: Learning Optical Flow with Convolutional Networks". In: *2015 IEEE International Conference on Computer Vision (ICCV)*. 2015, pp. 2758–2766. DOI: 10.1109/ICCV.2015.316.

[Far03] G. Farnebäck. "Two-Frame Motion Estimation Based on Polynomial Expansion". In: *Scandinavian Conference on Image Analysis*. 2003. DOI: 10.1007/3-540-45103-X_50.

[Fly16] J. Flynn et al. "Deep Stereo: Learning to Predict New Views from the World's Imagery". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 5515–5524. DOI: 10.1109/CVPR.2016.595.

[Gao00] X. Q. Gao, C. J. Duanmu, and C. R. Zou. "A multilevel successive elimination algorithm for block matching motion estimation". In: *IEEE Transactions on Image Processing* 9.3 (2000), pp. 501–504. DOI: 10.1109/83.826786.

[Haa93] G. de Haan et al. "True-motion estimation with 3-D recursive search block matching". In: *IEEE Transactions on Circuits and Systems for Video Technology* 3.5 (1993), pp. 368–379. DOI: 10.1109/76.246088.

[Hua21] Z. Huang et al. *RIFE: Real-Time Intermediate Flow Estimation for Video Frame Interpolation*. 2021. arXiv: 2011.06294 [cs.CV].

[Hui18] T.-W. Hui, X. Tang, and C. C. Loy. "LiteFlowNet: A Lightweight Convolutional Neural Network for Optical Flow Estimation". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2018.

[Jia18] H. Jiang et al. "Super SloMo: High Quality Estimation of Multiple Intermediate Frames for Video Interpolation". In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2018, pp. 9000–9008. DOI: 10.1109/CVPR.2018.00938.

[Kim14] U. S. Kim and M. H. Sunwoo. "New Frame Rate Up-Conversion Algorithms With Low Computational Complexity". In: *IEEE Transactions on Circuits and Systems for Video Technology* 24.3 (2014), pp. 384–393. DOI: 10.1109/TCSVT.2013.2278142.

[Li18] Z. Li and N. Snavely. "MegaDepth: Learning Single-View Depth Prediction from Internet Photos". In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2018, pp. 2041–2050. DOI: 10.1109/CVPR.2018.00218.

[Liu17] Z. Liu et al. "Video Frame Synthesis Using Deep Voxel Flow". In: *2017 IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 4473–4481. DOI: 10.1109/ICCV.2017.478.

[Lon16] G. Long et al. “Learning Image Matching by Simply Watching Video”. In: *Computer Vision – ECCV 2016*. Ed. by B. Leibe et al. Cham: Springer International Publishing, 2016, pp. 434–450. ISBN: 978-3-319-46466-4.

[Men20] H. Men et al. “Visual Quality Assessment for Interpolated Slow-Motion Videos Based on a Novel Database”. In: *2020 Twelfth International Conference on Quality of Multimedia Experience (QoMEX)*. 2020, pp. 1–6. DOI: 10.1109/QoMEX48832.2020.9123096.

[Nik17] S. Niklaus, L. Mai, and F. Liu. “Video Frame Interpolation via Adaptive Separable Convolution”. In: *2017 IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 261–270. DOI: 10.1109/ICCV.2017.37.

[Nik18] S. Niklaus and F. Liu. “Context-Aware Synthesis for Video Frame Interpolation”. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2018, pp. 1701–1710. DOI: 10.1109/CVPR.2018.00183.

[Nik19] S. Niklaus. *A Reimplementation of LiteFlowNet Using PyTorch*. https://github.com/sniklaus/pytorch-liteflownet. 2019.

[Nik20] S. Niklaus and F. Liu. “Softmax Splatting for Video Frame Interpolation”. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, pp. 5436–5445. DOI: 10.1109/CVPR42600.2020.00548.

[Pas17] A. Paszke et al. “Automatic differentiation in PyTorch”. In: *NIPS 2017 Autodiff Workshop: The Future of Gradient-based Machine Learning Software and Techniques*. 2017.

[Ran17] A. Ranjan and M. J. Black. “Optical Flow Estimation Using a Spatial Pyramid Network”. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 2720–2729. DOI: 10.1109/CVPR.2017.291.

[Soo12] K. Soomro, A. R. Zamir, and M. Shah. “UCF101: A dataset of 101 human actions classes from videos in the wild”. In: *International Journal of Computer Vision*. 2012. DOI: arXiv:1212.0402.

[Wan04] Z. Wang et al. “Image quality assessment: from error visibility to structural similarity”. In: *IEEE Transactions on Image Processing* 13.4 (2004), pp. 600–612. DOI: 10.1109/TIP.2003.819861.

[Wan10] C. Wang et al. “Frame Rate Up-Conversion Using Trilateral Filtering”. In: *IEEE Transactions on Circuits and Systems for Video Technology* 20.6 (2010), pp. 886–893. DOI: 10.1109/TCSVT.2010.2046057.

[Wu15] J. Wu et al. “Enabling Adaptive High-Frame-Rate Video Streaming in Mobile Cloud Gaming Applications”. In: *IEEE Transactions on Circuits and Systems for Video Technology* 25.12 (2015), pp. 1988–2001. DOI: 10.1109/TCSVT.2015.2441412.

[Wu18] C.-Y. Wu, N. Singhal, and P. Krahenbuhl. “Video Compression through Image Interpolation”. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. Sept. 2018.

[Xu17] J. Xu, R. Ranftl, and V. Koltun. “Accurate Optical Flow via Direct Cost Volume Processing”. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 5807–5815. DOI: 10.1109/CVPR.2017.615.

[Xue19] T. Xue et al. “Video Enhancement with Task-Oriented Flow”. In: *International Journal of Computer Vision (IJCV)* 127.8 (2019), pp. 1106–1125.

[Zhu00] S. Zhu and K.-K. Ma. “A new diamond search algorithm for fast block-matching motion estimation”. In: *IEEE Transactions on Image Processing* 9.2 (2000), pp. 287–290. DOI: 10.1109/83.821744.

ISSN 1213-6972 (print)
ISSN 1213-6964 (online)
Journal of WSCG
http://www.wscg.eu
Vol.29, No-1-2, 2021

Interactive Individualized Neuroanatomy Labeling for Neuroanatomy Teaching

Felippe T. Angelo
Unicamp/FEEC/DCA, Campinas, SP, Brazil
trigueiro.angelo@outlook.com

Raphael Voltoline
Unicamp/FEEC/DCA, Campinas, SP, Brazil
raphavoltoline.rvr@gmail.com

Giuliano R. Gonçalves
Unicamp/FCM, Campinas, SP, Brazil
giulianoanatomia@gmail.com

Wu, Shin-Ting
Unicamp/FEEC/DCA, Campinas, SP, Brazil
ting@dca.fee.unicamp.br

ABSTRACT

As the imaging technology and the understanding of neurological disease improve, a solid understanding of neuroanatomy has become increasingly relevant. Neuroanatomy teaching includes the practice of cadaveric dissection and neuroanatomy atlases consisting of images of a brain with its labeled structures. However, the natural inter-individual neuroanatomical variability cannot be taken into account. This work addresses the individual gross neuroanatomy atlas that could enrich medical students' experiences with various individual variations in anatomical landmarks and their spatial relationships. We propose to deform the CerebrA cortical atlas into the individual anatomical magnetic resonance imaging data to increase students' opportunity to contact normal neuroanatomical variations in the early stages of studies. Besides, we include interactive queries on the labels/names of neuroanatomical structures from an individual neuroanatomical atlas in a 3D space. An implementation on top of SimpleITK library and VMTK-Neuro software is presented. We generated a series of surface and internal neuroanatomy maps from 16 test volumes to attest to the potential of the proposed technique in brain labeling. For the age group between 10 to 75, there is evidence that the superficial cortical labeling is accurate with the visual assessment of the degree of concordance between the neuroanatomical and label boundaries.

Keywords

Neuroanatomical Atlas, CerebrA, Mindboggle 101 atlas, MNI-ICBM2009c template, Image-Registration, Brain Labeling.

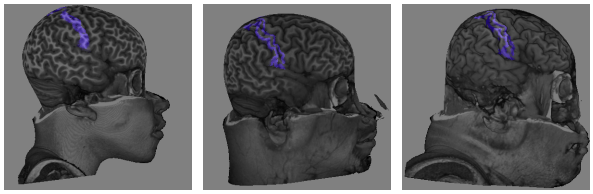
1 INTRODUCTION

Neuroanatomy is considered a problematic and unenviable topic by the medical community. In the early 90s, Jozefowicz [11] observed a great disinterest among the medical students and the development of the "necrophobia syndrome" in approximately half of them. Although most junior neurologists doubt the usefulness of neuroanatomy, its relevance to clinical practice increases as the understanding of neurological diseases, the imaging technology, and the patient-customized treatment improve [3].

Over the past years, teaching neuroanatomy has changed substantially. Several educational technologies [31] have been developed to allow students to experience clinical practices without resorting to traditional approaches, such as cadaveric dissection [17] and atlas of neuroanatomy [3]. Despite providing students with a detailed view of brain structure, cadaveric dissection faces the logistics and regulation related to cadaver purchase, storage, and transport [17]. The atlas with photographic images is still the most common way to study brain anatomy at the student's own pace. However, they cannot fully convey the structures' spatial relationships, which is recognized as fundamental to minimally invasive neurosurgery [30]. Besides, both techniques allow exploring only one or a few single brains.

There are many normal variations in a reference landmark's individual locations, either due to age [1] or due to variation in the cortical folds [13]. Fig. 1 illustrates the variations in the postcentral gyrus, highlighted in blue, across the brain of 3 healthy subjects. The Mindboggle individually manually labeled 101 human brain images were yet an attempt to establish normative morphometric variations in a healthy population [13]. Recently, Manera et al. introduced in [20] the Cerebrum Atlas (CerebrA), which is based on an accurate nonlinear registration of cortical (the outer layer) and subcortical (underneath the cerebral cortex) labeling from

Mindboggle 101 to the symmetric MNI-ICBM2009c template [8], followed by manual editing.



(a) 10-year-old male (b) 23-year-old male (c) 62-year-old male

Figure 1: Normal variations of a postcentral gyrus.

In this paper, we propose to evaluate the potential of CerebrA in automatic brain labeling of a wide age range for generating a variety of labeled brains of normal variations. We implemented a labeling module and integrated it into the VMTK-Neuro software. VMTK-Neuro provides functions for 3D interactive visualization of neuroanatomical data from the cortical view. Thereby, it helps teachers assess neuroanatomical boundaries in a cortical view and prepare instructional material. Besides, students can explore from these cortical views the normal variations across the available brains.

Our paper is organized as follows. In Section 2, we present an overview of some works that address the exploration and the labeling of neuroanatomical volumes. Next, in Section 3, we provide a detailed explanation of the proposed environment. Then, in Section 4, we present specific information about the software technologies we used for the implementation. After that, in Section 5, we show our results. Finally, in Section 6, we make some concluding remarks. Note that the anatomical slices throughout this paper are displayed in the radiological convention, i.e., the patient's left on the right of the image.

Contributions

The contributions of this paper are twofold. First, it presents an individual neuroanatomical labeling procedure using the CerebrA atlas to generate labeled brains that differ in normal neuroanatomical variations. Second, we show how to integrate an interactive environment to let a teacher and a student explore the labeled neuroanatomical structures from the cortical superficial views.

2 RELATED WORK

The aging population has significantly impacted the increase of neurodegenerative diseases, whose treatment requires a solid understanding of neuroanatomy. As already mentioned, several works have addressed neuroanatomy teaching approaches and tools to spark more interest among preclinical students. This section focuses on the results related to interactive visualization of neuroanatomical atlas and efforts towards brain labeling.

Johnson and Becker [10] led a group that developed a visualization tool for encephalic structures through neuroimages. Thus, it is possible to visualize both labeled structures of healthy and pathological brains. In turn, John Sundsten [27] devised an atlas using the photos of dissection and the drawings of labeled structures. A disadvantage of using these types of the atlas is that the images are limited to brain slices, requiring a spatial cognitive effort to understand the spatial relationships between the cortical surface structures.

Although Martin and Soliz [21] designed an atlas that includes the cortical surface anatomy, the maps are either brain photographs or drawings. Likewise is the teaching website on Pathology, Neuropathology, and Neuroimaging designed by Queiroz and Paes [5]. The informative data comprises only the photographs of brains and microscope slides.

Ding et al. [6] developed a visual exploration tool for a neuroanatomical atlas. Differently from other works, it addressed neuroanatomy at cellular resolution. The atlas was built from anatomical and diffusion-weighted magnetic resonance imaging. Thus, both the gross anatomy and the anatomy of neural pathways can be queried. However, as the previously mentioned works, it is impossible to visualize the labeled structures in three dimensions. Moreover, brain variability was not a concern.

Christensen et al. [4] presented a flow deformation model to transform a generic digital neuroanatomical atlas into the individual brain's shape. They claimed that the annotations of structure names could facilitate the interpretation and analysis of brain scans. In our work, we particularized the case for the CerebrA atlas. We also devised a way to create an annotated volume based on the MNI-ICBM2009c template and visualize the annotated regions in 3D.

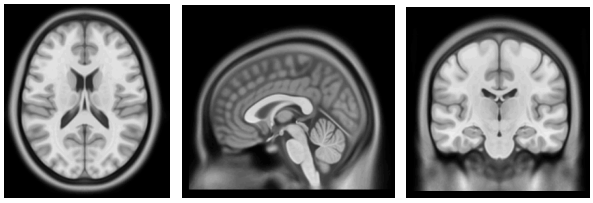
The Talairach atlas was created by the neurosurgeons Jean Talairach and Gabor Szikla [28] as a proposal to standardize a grid for the surgery of epilepsy. The grid was based on the conjecture that distances to lesions in the brain are proportional to overall brain size. In 1988 Talairach and Tournoux [29] presented the second version of the Talairach atlas based on post mortem dissection of a 60-year-old woman. Up to now, this atlas has served as the ad hoc standard for reporting locations of activation foci in functional brain mapping studies.

Lancaster et al. [18] further proposed an approach to creating a volume representing a discretization of the Talairach atlas. It allows the registration of an anatomical volume onto the Talairach maps. Consequently, it allows each anatomical volume voxel's association to a Talairach label that corresponds to the cerebral cortex region (Brodmann area).

Klein et al. [15] showed that the automatic and manual labeling agreement could be significantly improved when multiple atlases were used. In their experiments, they used 20 manually labeled brain images as atlases. Then, they applied the Mindboggle software to automatically label each of the brain scans based on multiple manually labeled scans. The drawback of a multi-atlas approach was its high computational cost.

In parallel, the Montreal Neurological Institute (MNI) average brains have been evolving [7]. The first template was the MNI305, an average of 305 T1-weighted magnetic resonance imaging (MRI) scans, linearly transformed into Talairach space. The MNI152, an improvement of the MNI305, was created from the average of 152 T1-weighted MRI scans of a normative young adult population with higher resolution and contrast [7]. It was adopted to define standard anatomy by the International Consortium of Brain Mapping (ICBM).

The linear version of the MNI152 data was evolved to a nonlinear one. A nonlinear MNI152 was built through a series of iterations starting from the MNI152 linear template [8]. Individual native MRIs are nonlinearly fitted to the average template computed in the previous iteration at each iteration. In the most recent phase of the MNI project, 6 versions of the MNI152 can be found and they are divided according to its symmetry and spatial resolution. Fig. 2 shows the axial, sagittal, and coronal slices of the MNI152 (symmetric with 1 mm of spatial resolution) volume.



(a) Axial (b) Sagittal (c) Coronal

Figure 2: MNI152 volume: an average of 152 scans non-linearly transformed to Talairach space.

Finally, in 2009, the MNI released an updated version of nonlinearly registered 152 acquisitions. To date, this template presents the best resolution and detail. Recently, Manera et al. [20] nonlinearly registered the symmetric MNI-ICBM2009 template to the Mindboggle-101 dataset [12], and, after manual editing, they created the Cerebrum Atlas (CerebrA).

3 OUR PROPOSAL

In this work, we propose to label a T1-weighted individual volume using CerebrA and allow a teacher to assess the labeling accuracy interactively. We also suggest that students use the same visual assessment environment to explore normal variations in neuroanatomical structures.

Fig. 3 presents an overview of our interactive individual neuroanatomy query tool's architecture comprising two significant modules: labeling and interactive visualization. The labeling module is responsible for estimating a non-linear transformation of the nonlinear symmetric CerebrA [20] into an individual anatomical MRI volume. And the interactive visualization module supports interactive queries of the names and visual feedback of the neuroanatomical structures of an individual MRI volume in both planar and curvilinear reformattings.

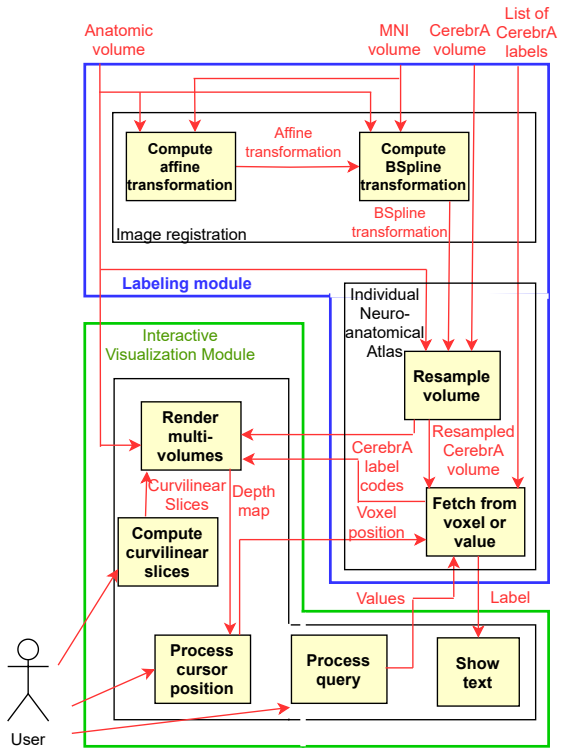


Figure 3: Proposed architecture for an individual interactive neuroanatomy atlas.

3.1 Labeling Module

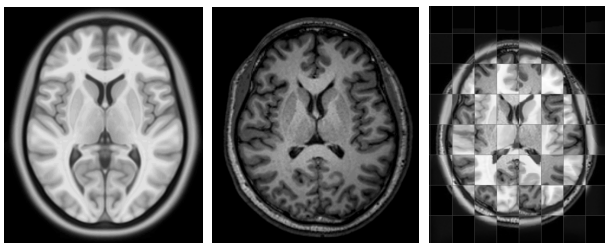
This module, outlined in blue in Fig. 3, is responsible for building an individual neuroanatomy atlas. We proposed to fit the CerebrA into the individual brain. For supporting the voxelwise query of a brain, the CerebrA is **resampled** in the resolution of an individual MRI volume, as depicted in Fig. 3. We applied the nearest-neighbor algorithm to interpolate the labels to ensure the uniqueness of names for each voxel. Then, for the sake of time performance, we performed the registration in two steps.

In the first step, the patient and the MNI152 volumes are coarsely aligned by an **affine transformation**, as sketched in Fig. 3. Then, the mutual information is the metric applied to evaluate the similarity between two images [22]. This metric is optimized using the Stochastic Gradient Descent algorithm [15]. To improve the algorithm convergence, we opted for a hier-

archical multiresolution approach [25] with two levels, a level where the images have a quarter of their original size and another level where they have half of their original size.

In the second step, the matching is improved using the Deformable Hierarchical BSpline algorithm [25]. We call it a **BSpline transformation**, as shown in Fig. 3 because the transformation is based on a grid of BSpline control points that supports the local deformation. We also used the hierarchical multiresolution approach [25] but in this case, with three levels, doubling the number of control points and the size of the images, starting from one-quarter of their original size, from one level to another. In this way, we could deal with coarse differences in lower resolution while adjusting subtle differences in higher resolution. The control points are also calculated by maximizing the mutual information metric [22] using the Stochastic Gradient Descent algorithm.

Fig. 4 illustrates a non-linear registration of the MNI152 symmetric volume (Fig. 4a) to a T1-weighted MRI volume (Fig. 4b) of healthy control. The checkerboard pattern (Fig. 4c) allows assessing the continuity of the neuroanatomical boundaries at the edges of the squares in which the two volumes are shown alternately.



(a) MNI152 (b) Control Volume (c) Checkerboard

Figure 4: Non-linear registration of (a) the MNI152 to (b) a T1-weighted volume (c) in a checkerboard pattern.

There is a 1:1 correspondence between the individual MRI voxels and the CerebrA voxels after registration. The voxels in the anatomical MRI volume carry the tissues' response to the applied magnetic field gradient. In contrast, each atlas voxel has one of the 102 label codes available on the NIST website [19]. In short, from a triple of coordinates (x,y,z), we could access two kinds of data necessary for rendering: the image of a neuroanatomical structure and its label code.

3.2 Interactive Visualization Module

This module, highlighted with the green outline in Fig. 3, is responsible for rendering the MRI volume and the CerebrA atlas into a single image and showing the region of interest's label at interactive rates. Because the registration and the sampling steps are time-consuming but carried out only once for each brain, we considered that they belong to the setup of a session of visual assessment or label queries when the interactions actually occur. For interactive label queries, we should address three issues: the rendering of a labeled anatomical volume at interactive rates, the views of the labeled anatomical MRI volume, and the effective fetch of the structure's name from a point on the cortical surface at which an input device points or the voxels' query from the input structure's name.

3.2.1 Rendering

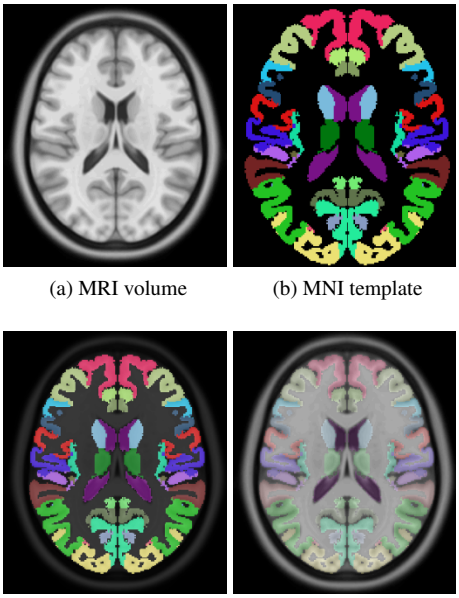
We mapped the structure labels of the CerebrA atlas onto distinct colors according to the FreeSurferColorLUT [14]. We implemented this colormap with a 1D texture lookup table, setting the visible voxel's opacity as 1 and of the other ones 0. Then, we applied the GPU-based rendering algorithm, proposed by Wu et al.[35], to generate a **multimodal volume** from the registered CerebrA atlas and a neuroanatomical MRI volume (Fig. 3). It consists of raycasting the anatomical MRI volume. For each pixel in the image, a ray is cast into the volume. The ray is resampled at equal intervals, and the corresponding position in the registered CerebrA atlas is also resampled. The weighted contributions of these two samples are accumulated to define the final color of the pixel. Two interpolation schemes were used in the resampling: the trilinear interpolation to the MRI volume (Fig. 5.(a)) and the nearest-neighbor interpolation to the MNI CerebrA labels (Fig. 5.(b)). Different weighting factors lead to different images, as illustrated in Figs. 5.(c) and (d). Interactive control of the weighting factors facilitates visually checking the alignment between neuroanatomical boundaries and the edges of labeled regions.

3.2.2 Views

Besides the classical views of a volume in 3 anatomical planes: axial, coronal, and sagittal (Fig. 6), we devised a way to view the cortical surface at different depths (Fig. 7). It can help one better understanding the spatial relationship between cortical folds and label boundaries. We applied the curvilinear reformatting algorithm presented by Wu et al. [34] to **compute curvilinear slices** (Fig. 3). The computation consists of four steps: (1) estimating the cortical surface of the brain of interest; (2) computing a mesh of the estimated cortical surface; (3) generating a series of meshes by offsetting the cortical mesh at different depths; and (4) applying the depthmaps of the meshes to control the range of the voxels to be accumulated along the casting ray. A curvilinearly reformatted view of a T1-weighted MRI volume is shown in Fig. 1.

3.2.3 Interaction

For querying the label (or name) of any region (**show text** in Fig. 3), the user interacts with the displayed image through a cursor. Fig. 6 shows a 2D crosshair cursor



(a) MRI volume (b) MNI template

(c) 0.25MRI+0.75MNI (d) 0.75MRI+0.25MNI

Figure 5: Blending of (a) an anatomical MRI volume and (b) the registered CerebrA with (c) 25% and (d) 75% of MRI data.

and Fig. 7 a 3D cursor over the right cerebellum on the displayed images as visual feedback of mouse motions. A click with the mouse on a 3D view of the labeled anatomical volume triggers the procedure that converts the 2D position of the clicked pixel into the spatial position of the corresponding voxel [33] (**process cursor position** in Fig. 3). The registered CerebrA label code is accessed from the coordinate (x,y,z) of the picked MRI voxel. We used a balanced binary search algorithm with the worst-case performance $O(logn)$ to access the CerebrA label code and get the structure's name.

The brain region's query from a structure's name is also supported (**process query** in Fig. 3). We organized the label codes and their corresponding structure names [14] in a lookup table and applied a linear search to get a label code from a structure name. Though its high cost of $O(n)$, it is affordable due to the small number of structure names. The size n is in the order of magnitude of a hundred. The result could be either in **text** format (**show text** in Fig. 3) or in colored regions whose rendering is presented in Section 3.2.1. The two query modes allow the user to explore intuitively and smoothly the neuroanatomical structures individually.

4 IMPLEMENTATION

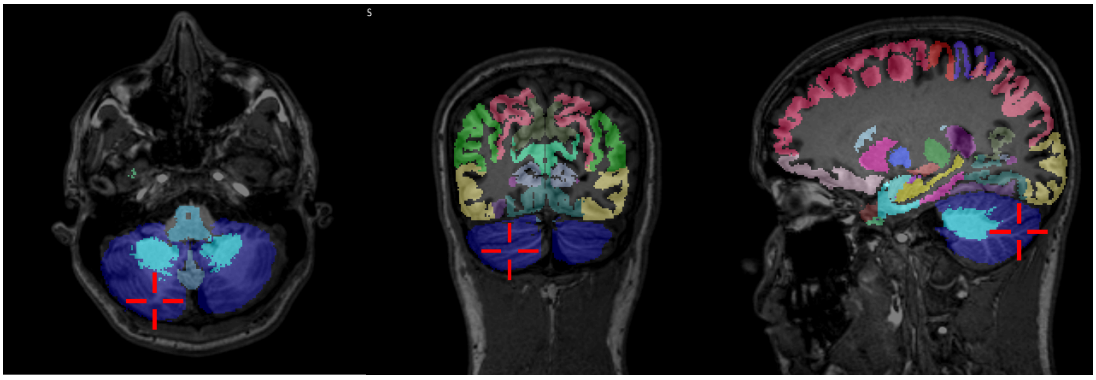
With maximum reuse in mind, we surveyed the applications that provide the functions that would enable the implementation of our proposal. We decided on the functions available in the SimpleITK library [2] to implement the labeling module and the Qt-based VMTK-Neuro [32] to program the interactive visualization module. SimpleITK is a wrapper for the well-known ITK [9] and VMTK-Neuro (*Visual Manipulation Toolkit for Neuroimages*) is a multi-platform (Windows, Mac, and Ubuntu) multi-modal exploratory visualization software. Both are programmed in C++. In addition, built-in functions provided by the algorithm library in C++ standard template libraries (STL) [24] were used to process the CerebrA label codes. With all these tools, we were left with the challenge of integrating them into an application that supports interactive queries of regions covered in the CerebrA atlas.

We distributed the functions between CPU and GPU as depicted in Fig. 8. There is a time-consuming overhead for processing on the CPU an individual CerebrA atlas, which could be saved in a file, as shown in the pink box, and reused in different query sessions. During a query session, the user events are processed by the Qt-based widgets (CPU), and, for the sake of performance, the images are concurrently rendered on the GPU.

We used the functions of the classes IMAGEREGISTRATIONMETHOD, AFFINETRANSFORM and BSPLINETRANFORM, from the SimpleITK image analysis library, to implement the Labeling Module components (Section 3.1). The IMAGEREGISTRATIONMETHOD provides a wide variety of functions that help implement an image registration algorithm. These functions include a series of optimization methods, objective functions, and interfaces to the available classes of transformation models. More specifically, as depicted in Fig. 8, we applied the AFFINETRANSFORM and BSPLINETRANFORM models with, respectively, the parameter values: Maximum number of iterations per level (200,200); Number of hierarchical levels (2,3); Sampling percentages (10%,10%); Learning rate (0.1,0.1); Number of histogram bins (40,40); Initial number of control points (–,2); Convergence value (1e-6,1e-6); Initial variance of the Gaussian transformation (4,4); Initial shrink factor (4,4); BSpline order (–,4); Method for rescaling and optimization parameters: (Jacobian, Physical Shift). These values were obtained empirically by registering the 16 control volumes individually.

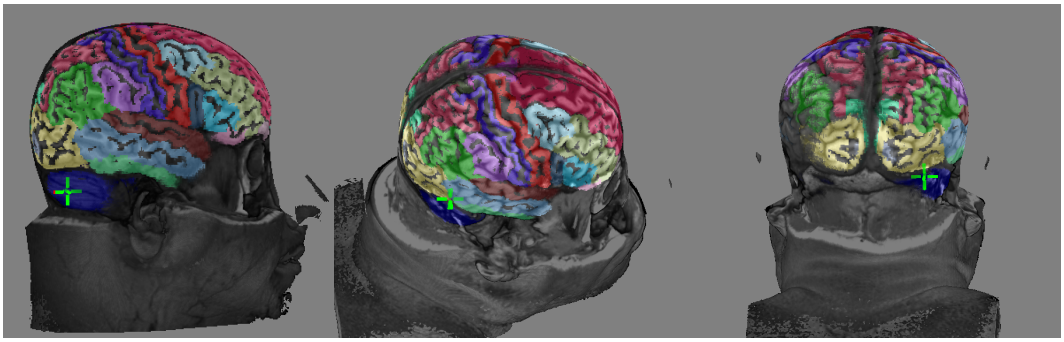
For benefiting from the interactive environment that the VMTK-Neuro application offers, we have implemented the Interactive Visualization Module on its top. The registered CerebrA atlas and MRI volume should be preloaded as filtered texture into the GPU at the setup of a query session. Two exploration methods are supported: either through 2D anatomical views or through a 3D view of curvilinearly reformatted volume.

A series of curvilinear slices is generated using the functions available in the CURVILINEARREFORMATTING module. The yellow box in Fig. 8 highlights it. We also reused the blending function available in the



(a) Axial (b) Coronal (c) Sagittal

Figure 6: Label querying with a red cursor of the internal structure of neuroanatomy (right cerebellum) from 2D views of 62-year-old male subject.



(a) Right (b) Top (c) Back

Figure 7: Label querying with a green cursor of surface neuroanatomy (right cerebellum) from 3D views of a 62-year-old male subject.

MULTIMODALRENDERING module for weighting the contributions of the anatomical MRI volume and the registered CerebrA atlas to a single rendered image. When not explicitly mentioned, the images presented in this work result from the blending of the label code colors into the grayscale anatomical MRI volume in 50%.

Fig. 6 shows the interface for label queries from the classical anatomical 2D views. One can retrieve the label by pointing a cursor (a red crosshair) on the colored internal structure in any of three planes (axial, sagittal, and coronal). Fig. 7 illustrates the interface for queries of the (cortical) surface anatomy from a 3D view of the same subject. The user can retrieve any label by pointing a cursor (a green crosshair) on a colored cortical region.

5 RESULTS

Fig. 9 presents the interface of the implemented prototype. It consists of 4 coordinated canvases: one canvas on the left side displays the 3D view of the customized neuroanatomy atlas, and three canvas on the right side displaying axial, sagittal, and coronal slice planes. The cursors in four views point coordinately at the same spatial position. Note that when one clicks a mouse on a point, the name query is automatically triggered. As a result, the value associated with the point's label code is promptly displayed on the widget, outlined by the orange line.

We designed three experiments to show how the proposed environment can visually assess CerebrA-based brain labeling's quality concerning neuroanatomy teaching at interactive rates. First, since accurate registration results in proper labeling, we checked the accuracy of CerebrA's registration to an MRI volume. Second, we visually assessed the concordance between neuroanatomical and label boundaries in all test volumes to rate how misregistration can impact the labeling's visual quality. Finally, in the third experiment, we evaluated the proposed algorithm's time performance for its interactivity.

The experiments were performed on the desktop provided with 8Gb of RAM Memory, an Intel Core i7 Processor 860 2.8GHz and an NVIDIA GeForce GT 630 GPU, and on the laptop provided with an Intel Core(TM) i7-6700HQ 2.60 GHz with 8GB RAM and an NVIDIA GeForce GTX 960 with 4GB VRAM. We chose randomly from our university hospital's database 16 MRI volumes of healthy subjects (8 males)

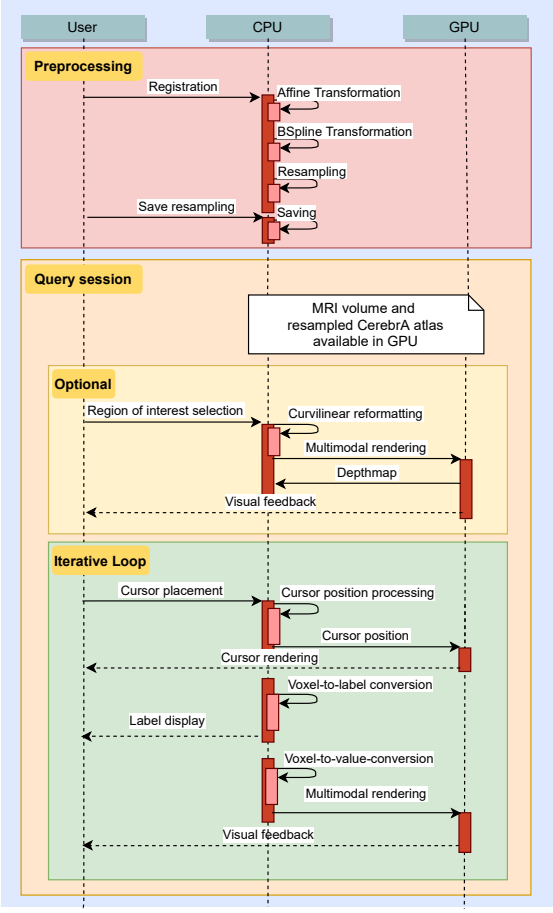


Figure 8: Sequence diagram in two stages: (1) Labeling module: preprocessing of an individual CerebrA atlas using SimpleITK, (2) Interactive Visualization module: query of neuroanatomical structures in the VMTK-Neuro environment.

ranged from 9 to 79 years. The T1-weighted spin-echo sequences were acquired on a Philips Achieva 3D Scanner using the acquisition parameters: voxel size =1x1x1 mm, no gap, TR=7 ms, TE=3.2 ms, flip angle=8, matrix=240x240, FOV=240x240, and resolution=180x240x240. All subjects enrolled in the present study signed an informed consent form approved by our university's Ethics Committee.

5.1 Nonlinear Registration

For assessing the registration accuracy visually, we used the SimpleITK features from Jupyter Notebooks [16]. We applied the HISTOGRAMMATCHING image filter to the MRI scans before displaying them on a checkerboard pattern, pairwise with the registered MNI152, as shown in Fig. 4. Since the filter standardizes the grayscale values of a control volume by matching the control volume histogram's shape to the MNI152 histogram, it ensures that similar intensities in two volumes have similar tissue meaning [23].

We scrolled the slices in axial, coronal, and sagittal views and checked the continuity of the neuroanatomical boundaries at the squares' edges in each slice. We found that nonlinear registration using the parameters in Section 4 failed for volumes under 30 and over 70 years. Fig. 10a and Fig. 10b illustrate misregistration where there are several mismatches of the neuroanatomical contours along the squares' edges in axial slices. In Fig. 10c, the anatomical boundaries are better matched.

Our finding corroborates the results obtained by Allen et al. [1]. They concluded that gray matter decreased linearly with age, resulting in a decline of about 9,1–9.8% between the ages of 30 and 70 and a decline of 11.3–12.3% by 80. On the other hand, white matter volume increased until the mid-50s; after then, it declined rapidly. At 70 years, white matter volume was only 5.6–6.4% less than at 30 years, but by age 80, a cubic regression model predicted that the decrease would be 21.6–25.0%.

5.2 Labeling

We carried out the second and the third experiments using the implemented interactive labeling system (Section 4). The visual assessment is based on the agreement of the neuroanatomical and label boundaries from 4 views: the 3D superficial cortical view, the coronal and axial plane views at the level of septum pellucidum, and the mid-sagittal plane view.

For the first view, we used as anatomical references the frontal, parietal, occipital, and temporal gyri, the lateral and central sulcus, and the longitudinal fissure (LF). For the second view, we chose the LF, the cingulate gyrus (CG), the lateral ventricles (LV), the third ventricle (3V), the septum pellucidum (SP), the basal ganglia (BG), the amygdala, the insular cortex (I), and the lateral sulcus. For the third view, the neuroanatomical landmarks were the LF, LV, 3V, SP, I, and the corpus callosum (CC). Finally, for the fourth view, the references were the CG, LV, CC, cerebellum, the fourth ventricle, the parieto-occipital sulcus, the calcarine sulcus, the brainstem, and the occipital, parietal, and frontal lobes.

As expected, the degree of concordance between the neuroanatomical and label boundaries was low for misregistered pairs of volumes. Fig. 11 illustrates the labeled anatomical structures in the axial slice at the height of the septum pellucidum of the volumes shown in Fig. 10. From the landmarks pointed by the red arrows, one can visually perceive the two boundaries' discrepancies. Our experiment lets us infer that, from the classical plane views, the less the MRI volume is aligned to the CerebrA (Figs. 10a and 10b), the more discrepant are the label boundaries to the anatomical boundaries (Figs. 11a and 11b).

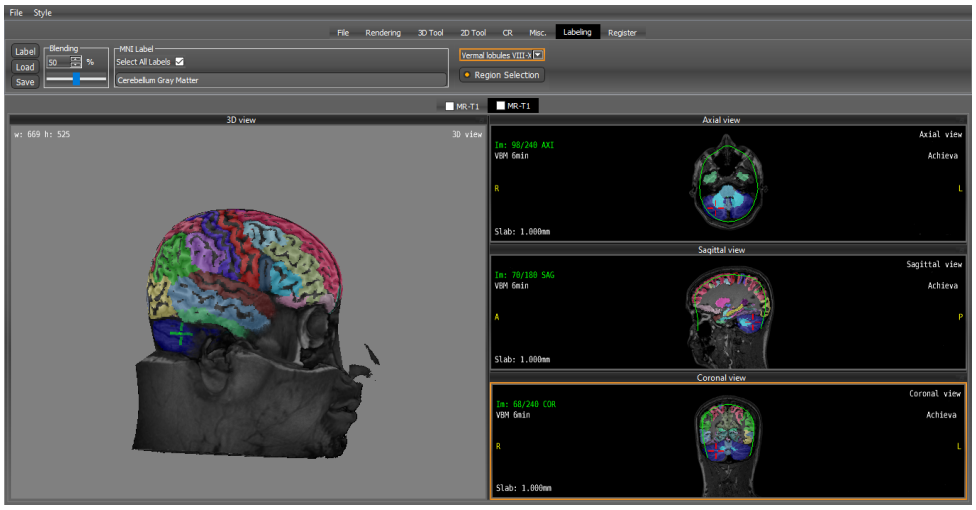
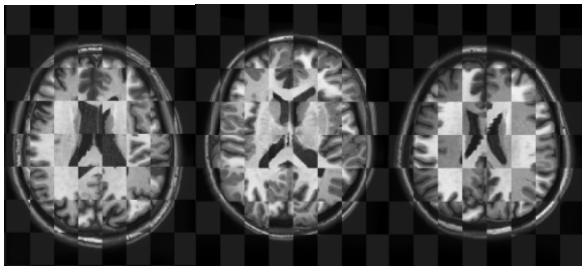
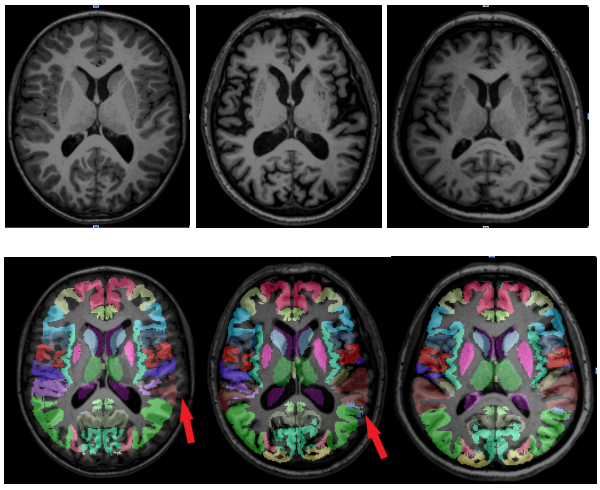


Figure 9: Interface of our proposed individual neuroanatomy atlas. The interface displays the name of the brainstem pointed by the red cursor.



(a) 11 year-old (b) 78 year-old (c) 60 year-old

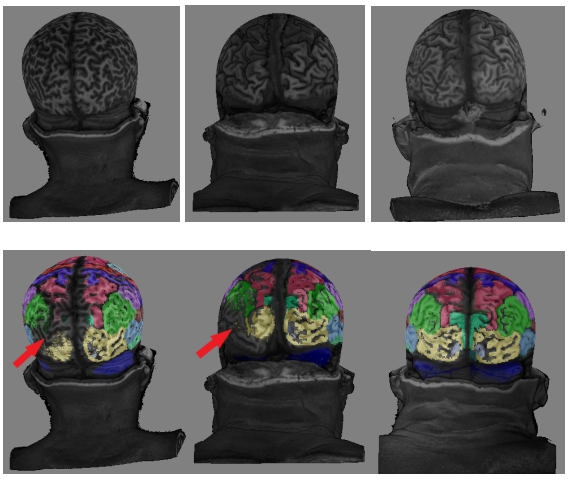
Figure 10: Visual assessment of registration accuracy of a volume: (a) under 30; (b) over 70; and (c) between 30 and 70 years of age.



(a) 11 year-old (b) 78 year-old (c) 60 year-old

Figure 11: Visual assessment of labeling accuracy of a volume from axial slices: (a) under 30; (b) over 70; and (c) between 30 and 70 years of age.

However, to our surprise, such discrepancies were not notable from 3D superficial cortical views as depicted in Fig. 12. It looked that the anatomical and label boundaries were perfectly matchable even when the volumes were misregistered (Figs. 12a and 12b). Only an attentive user would notice some minor flaws in the labeled regions highlighted by the red arrows. Further investigation led us to see that the multimodal rendering procedure we applied accommodates the differences between the superficial boundaries with some slack because it only retrieves the color codes of the control volume's visible voxels. The labels that leaked out the control volume are automatically discarded (Section 3.2.1). The perceived flaws are in the regions where the difference is greater than the cortical thickness, which is few regions.



(a) 11 year-old (b) 78 year-old (c) 60 year-old

Figure 12: Visual assessment of labeling accuracy of a volume from 3D superficial cortical view: (a) under 30; (b) over 70; and (c) between 30 and 70 years of age.

5.3 Time Performance

We measured 3D rendering time and the label query's processing time to evaluate our label querying envi-

ronment's interactivity. Both measurements were performed on an image of 900x900 pixels. The average times were 42.92 ms for rendering and 2.67 ms for query by the label code. Query by a structure's name has taken in average 0.0589 ms. All these times are below the recommended response time limits, suggesting that the environment is interactive. It is, however, worth remarking that the generation of an individual CerebrA atlas is too costly in time. It took less than 20 min in total on the desktop computer, with around 19 min to register and 7s to resample.

6 CONCLUSION

We presented an automatic individual CerebrA-based neuroanatomical labeling system to support gross neuroanatomy teaching. Because the CerebrA atlas is based on the symmetric MNI-ICBM2009c template, we conjectured that a nonlinear registration could register CerebrA to most healthy MRI volumes within a margin of tolerance that is acceptable for the teaching purpose. We implemented a prototype on top of available tools and libraries, namely the C++ Standard Template Library [24], SimpleITK [2], and VMTK-Neuro [32].

Our experiments with 16 test control volumes ranged from 9 to 79 years show that the proposed system provides interactive queries and visually accurate labeled cortical surfaces even if the CerebrA atlas could not fit perfectly into an MRI volume of interest. Because this cortical superficial view helps in preoperative planning of the entry points [26], we believe that the developed individual neuroanatomy atlas tool could be useful as complementary courseware to neuroanatomy. Furthermore, afforded curvilinear reformatting and interactive querying by values help students gain familiarity with spatial relationships of the cortical superficial structures of diverse sizes and shapes.

There is still a lot of implementation work to be done. In a short time, we plan to conclude the interface to the querying by values, from which the user can change the color codes and the level of brain hierarchy to be displayed. In the medium term, we also like to integrate the Talairach atlas, which is still widely applied for functional brain mapping. Another point that deserves special attention is the fitting of CerebrA to the age group's volumes outside the range of 30 to 70 years. We would like to know whether there is a procedure to perform this registration in the long run. Finally, we intend to conduct tests directly involving preclinical students to assess the proposed atlas's usability.

7 REFERENCES

[1] John S. Allen, Joel Bruss, C. Kice Brown, and Hanna Damasio. Normal neuroanatomical variation due to age: The major lobes and a parcellation of the temporal region. *Neurobiology of Aging*, 26(9):1245–1260, October 2005.

[2] Richard Beare, Bradley Lowekamp, and Ziv Yaniv. Image segmentation, registration and characterization in r with simpleitk. *Journal of statistical software*, 86, 2018.

[3] Bernard S. Chang and Zoltán Molnár. Practical neuroanatomy teaching in the 21st century. *Annals of Neurology*, 77(6):911–916, May 2015.

[4] G.E. Christensen, M.I. Miller, M.W. Vannier, and U. Grenander. Individualizing neuro-anatomical atlases using a massively parallel computer. *Computer*, 29(1):32–38, 1996.

[5] Luciano de Souza Queiroz and Rogério Augusto Paes. Teaching site for pathological anatomy, neuropathology and neurimaging, 2006. Accessed on: July, 2020.

[6] Song-Lin Ding, Joshua J Royall, Susan M Sunkin, Lydia Ng, Benjamin AC Facer, Phil Lesnar, Angie Guillozet-Bongaarts, Bergen McMurray, Aaron Szafer, Tim A Dolbeare, et al. Comprehensive cellular-resolution atlas of the adult human brain. *Journal of Comparative Neurology*, 524(16):3127–3481, 2016.

[7] Alan C Evans, Andrew L Janke, D Louis Collins, and Sylvain Baillet. Brain templates and atlases. *Neuroimage*, 62(2):911–922, 2012.

[8] Vladimir Fonov, Alan C. Evans, Kelly Botteron, C. Robert Almli, Robert C. McKinstry, and D. Louis Collins. Unbiased average age-appropriate atlases for pediatric studies. *NeuroImage*, 54(1):313–327, January 2011.

[9] Hans J. Johnson, M. McCormick, L. Ibáñez, and The Insight Software Consortium. *The ITK Software Guide*. Kitware, Inc., third edition, 2013. *In press*.

[10] Keith A. Johnson and J. Alex Becker. The whole brain atlas, 1999. Accessed on: July, 2019 (`http://www.med.harvard.edu/AANLIB/`).

[11] R. F. Jozefowicz. Neurophobia: The fear of neurology among medical students. *Archives of Neurology*, 51(4):328–329, April 1994.

[12] Arno Klein, Brett Mensh, Satrajit Ghosh, Jason Tourville, and Joy Hirsch. Mindboggle: automated brain labeling with multiple atlases. *BMC medical imaging*, 5(1):7, 2005.

[13] Arno Klein and Jason Tourville. 101 labeled brain images and a consistent human cortical labeling protocol. *Frontiers in Neuroscience*, 6, 2012.

[14] Arno Klein and Jason Tourville. Mindboggle-101 Labels, jul 2015. Accessed on: Feb 2021.

[15] Stefan Klein, Marius Staring, and Josien PW

Pluim. Evaluation of optimization methods for nonrigid medical image registration using mutual information and b-splines. *IEEE transactions on image processing*, 16(12):2879–2890, 2007.

[16] Thomas Kluyver, Benjamin Ragan-Kelley, Fernando Pérez, Brian Granger, Matthias Bussonnier, Jonathan Frederic, Kyle Kelley, Jessica Hamrick, Jason Grout, Sylvain Corlay, Paul Ivanov, Damián Avila, Safia Abdalla, and Carol Willing. Jupyter notebooks – a publishing format for reproducible computational workflows. In F. Loizides and B. Schmidt, editors, *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, pages 87 – 90. IOS Press, 2016.

[17] Sanjay Konakondla, Reginald Fong, and Clemens M Schirmer. Simulation training in neurosurgery: advances in education and practice. *Advances in medical education and practice*, 8:465, 2017.

[18] J.L. Lancaster, L.H. Rainey, J.L. Summerlin, C.S. Freitas, P.T. Fox, A.C. Evans, A.W. Toga, and J.C. Mazziotta. Automated labeling of the human brain: A preliminary report on the development and evaluation of a forward-transform method. *Human Brain Mapping*, 5(4):238–242, 1997.

[19] Ana L. Manera, Mahsa Dadar, Vladimir Fonov, and D. Louis Collins. *CerebrA atlas*. Kitware, Inc., first edition, 2020.

[20] Ana L. Manera, Mahsa Dadar, Vladimir Fonov, and D. Louis Collins. CerebrA, registration and manual label correction of mindboggle-101 atlas for MNI-ICBM152 template. *Scientific Data*, 7(1), July 2020.

[21] John H. Martin and Ewa Soliz. Interactive neuroanatomy atlas, 2003. Accessed on: July, 2020 (`http://www.columbia.edu/itc/hs/medical/neuroanatomy/neuroanat/`).

[22] David Mattes, David R Haynor, Hubert Vesselle, Thomas K Lewellen, and William Eubank. Pet-ct image registration in the chest using free-form deformations. *IEEE transactions on medical imaging*, 22(1):120–128, 2003.

[23] L.G. Nyul, J.K. Udupa, and Xuan Zhang. New variants of a method of MRI scale standardization. *IEEE Transactions on Medical Imaging*, 19(2):143–150, 2000.

[24] P.J. Plauger, Meng Lee, David Musser, and Alexander A. Stepanov. *C++ Standard Template Library*. Prentice Hall PTR, USA, 1st edition, 2000.

[25] D. Rueckert, L.I. Sonoda, C. Hayes, D.L.G. Hill, M.O. Leach, and D.J. Hawkes. Nonrigid registration using free-form deformations: application to breast MR images. *IEEE Transactions on Medical Imaging*, 18(8):712–721, 1999.

[26] Reuben R. Shamir, Leo Joskowicz, Luca Antiga, Roberto I. Foroni, and Yigal Shoshan. Trajectory planning method for reduced patient risk in image-guided neurosurgery: concept and preliminary results. In *Medical Imaging 2010: Visualization, Image-Guided Procedures, and Modeling*. SPIE, March 2010.

[27] John W. Sundsten. 2-d and 3-d views of the brain from cadaver sections, mri scans, and computer reconstructions., 1997. Accessed on: July, 2019 (`http://da.si.washington.edu/da.html`).

[28] J. Talairach and G. Szikla. *atlas d'anatomie stereotaxique du telencephale*. Masson & Cie, 1967.

[29] J. Talairach and P. Tournoux. *Co-planar Stereotaxic Atlas of the Human Brain.* Georg Thieme Verlag, 1988.

[30] Tina Vajsbaher, Holger Schultheis, and Nader K Francis. Spatial cognition in minimally invasive surgery: a systematic review. *BMC Surgery*, 18(1), November 2018.

[31] Matthew C. Welch, Jonathan Yu, M. Benjamin Larkin, Erin K. Graves, and David Mears. A multimedia educational module for teaching early medical neuroanatomy. *MedEdPORTAL*, 16(1), January 2020.

[32] S. T. Wu, A. C. Valente, L. d. S. Watanabe, C. L. Yasuda, A. C. Coan, and F. Cendes. Pre-alignment for Co-registration in Native Space. In *2014 27th SIBGRAPI Conference on Graphics, Patterns and Images*, pages 41–48, Aug 2014.

[33] S.-T. Wu, José Elías Yauri Vidalón, and Lionis de Souza Watanabe. Snapping a cursor on volume data. In *24th SIBGRAPI Conference on Graphics, Patterns and Images, Sibgrapi 2011, Alagoas, Maceió, Brazil, August 28-31, 2011*, pages 109–116, 2011.

[34] Shin-Ting Wu, Wallace Souza Loos, Dayvid Leonardo de Castro Oliveira, Fernando Cendes, Clarissa L. Yasuda, and Enrico Ghizoni. Interactive patient-customized curvilinear reformatting for improving neurosurgical planning. *International Journal of Computer Assisted Radiology and Surgery*, 14(5):851–859, October 2018.

[35] Shin Ting Wu, Raphael Voltoline, Wallace Souza Loos, Jose Angel Ivan Rubianes Silva, Lionis de Souza Watanabe, Barbara Amorim, Ana Carolina Coan, Fernando Cendes, and Clarissa L. Yasuda. Toward a multimodal diagnostic exploratory visualization of focal cortical dysplasia. *IEEE Computer Graphics and Applications*, 38(3):73–89, 2018.

ISSN 1213-6972 (print)
ISSN 1213-6964 (online)
Journal of WSCG
http://www.wscg.eu
Vol.29, No-1-2, 2021

Word Recognition using Embedded Prototype Subspace Classifiers on a new Imbalanced Dataset

Anders Hast and Ekta Vats
Department of Information Technology
Uppsala University
SE-751 05 Uppsala, Sweden
anders.hast@it.uu.se; ekta.vats@it.uu.se

ABSTRACT

This paper presents an approach towards word recognition based on embedded prototype subspace classification. The purpose of this paper is three-fold. Firstly, a new dataset for word recognition is presented, which is extracted from the Esposalles database consisting of the Barcelona cathedral marriage records. Secondly, different clustering techniques are evaluated for Embedded Prototype Subspace Classifiers. The dataset, containing 30 different classes of words is heavily imbalanced, and some word classes are very similar, which renders the classification task rather challenging. For ease of use, no stratified sampling is done in advance, and the impact of different data splits is evaluated for different clustering techniques. It will be demonstrated that the original clustering technique based on scaling the bandwidth has to be adjusted for this new dataset. Thirdly, an algorithm is therefore proposed that finds k clusters, striving to obtain a certain amount of feature points in each cluster, rather than finding some clusters based on scaling the Silverman's rule of thumb. Furthermore, Self Organising Maps are also evaluated as both a clustering and embedding technique.

Keywords

Subspaces, Embedded Prototypes, Clustering, Deep Learning, Self Organising Maps, t-SNE, Data splits.

1 INTRODUCTION

Recently, *Embedded Prototype Subspace Classification* (EPSC) [HLV19, HL20] has proven to be able to classify datasets containing single digits, characters and even objects, such as the MNIST dataset of handwritten digits [LCB10], E-MNIST containing letters [CATvS17], the Kuzushiji-MNIST dataset containing Japanese handwritten characters [CBK*18], and the Fashion MNIST (F-MNIST) [XRV17] containing small images of clothes and accessories.

The advantage of EPSC compared to deep learning based methods [Sha18] for handwritten text recognition [KDJ18, DKMJ18, SF16] is that EPSC do not require powerful GPU resources in the training process and have no hidden layers, which makes it compact and fast. In general, EPSC learns from the embedding of feature vectors and creates a so-called subspaces from each cluster [KLR*77]. Even though EPSC does not always outperform the state-of-the-art deep learning approaches, it performs significantly as an alternative, where the learning and classification processes are both easy to interpret [Kri19, CPC19], explain [ADRS*19, GSC*19, CPC19], and visualise.

The main contributions of this paper are as follows. First of all a new dataset based on the Esposalles dataset [RFS*13] is presented, where 30 different words were extracted from the given training set. This new dataset can be used for the purpose of evaluating word recognition methods, rather than performing character or digit level recognition. This dataset is by intention heavily imbalanced, which makes it more interesting for real-world problems. Secondly, we present and compare three different methods for computing clusters, aimed at handling imbalanced datasets. Thirdly, an algorithm that finds k seed points for K-means clustering [HW79] is proposed .

2 BACKGROUND

Subspaces have been used for classification in pattern recognition since it was first proposed by Watanabe et al. [WP73] in 1967, and later further developed by Kohonen and others [WLK*67, KLR*77, KO76, KRMV76, OK88]. In general, by computing the norm of the projected feature vector to be classified into each subspace, the process can be regarded as a two layer neural network [OK88, Laa07], where the weights are mathematically defined through Principal Component

DOI:10.24132/JWSCG.2021.29.5

Analysis (PCA) [Laa07]. Another important advantage is that the learning process can easily be visualised, which makes it easy to understand, interpret and explain, as compared to most of the state-of-the-art deep learning approaches.

2.1 Subspace Classification

Herein, we have used the same kind of subspaces that were presented in [HLV19, Laa07, OK88]. Hence, every image to be classified is represented by a feature vector $\mathbf{x}$ with m real-valued elements $\mathbf{x}_j = \{x_1, z_2 ... x_m\}, \in \mathbb{R}$, such that the operations take place in a m-dimensional vector space $\mathbb{R}^m$. Any set of n linearly independent basis vectors $\{\mathbf{u}_1, \mathbf{u}_2, ... \mathbf{u}_n\}$, where $\mathbf{u}_i = \{w_{1,j}, w_{2,j} ... w_{m,j}\}, w_{i,j} \in \mathbb{R}$, which can be combined into an $m \times n$ matrix $\mathbf{U} \in \mathbb{R}^{m \times n}$, span a subspace $\mathscr{L}_\mathbf{U}$

$$\mathscr{L}_\mathbf{U} = \{\mathbf{x} | \mathbf{x} = \sum_{i=1}^{n} \rho_i \mathbf{u}_i, \rho_i \in \mathbb{R}\} \tag{1}$$

where,

$$\rho_i = \mathbf{x}^T \mathbf{u}_i = \sum_{j=1}^{m} x_j w_{i,j} \tag{2}$$

Classification of a feature vector can be performed by projecting $\mathbf{x}$ onto each and every subspace $\mathscr{L}_{\mathbf{U}_k}$. The vector $\hat{\mathbf{x}}$ will in this way be a reconstruction of $\mathbf{x}$, using all vectors in the subspace through

$$\hat{\mathbf{x}} = \sum_{i=1}^{n} (\mathbf{x}^T \mathbf{u}_i) \mathbf{u}_i \tag{3}$$

$$= \sum_{i=1}^{n} \rho_i \mathbf{u}_i \tag{4}$$

$$= \mathbf{U}^T \mathbf{U} x^T \tag{5}$$

By normalising all the vectors in $\mathbf{U}$, the norm of the projected vector can be simplified as

$$||\hat{\mathbf{x}}||^2 = (\mathbf{U}x^T) \cdot (\mathbf{U}x^T) \tag{6}$$

$$= (\mathbf{U}x^T)^2 \tag{7}$$

$$= \sum_{i=1}^{n} {\rho_i}^2 \tag{8}$$

In this way, the feature vector $\mathbf{x}$, which is most similar to the feature vectors that were used to construct the subspace in question $\mathscr{L}_{\mathbf{U}_k}$, will subsequently also have the largest norm $||\hat{\mathbf{x}}||^2$.

2.2 Embedding and Clustering

In general, some group of prototypes are selected for the construction of each subspace by searching for the k nearest neighbors in the feature space, which is a rather time consuming process. The idea of EPSC [HLV19] is on the other hand to use t-distributed stochastic neighbour embedding (t-SNE) [MH08], which is both a visualisation technique as well as a machine learning technique, used to reduce the number of dimensions of high dimensional data (e.g. 2 dimensions in this case). In this process, clusters are formed since t-SNE strives to move similar features (represented by their projected points) closer to each other and dissimilar points further away from each other. Nevertheless, any embedding technique could be used for this purpose, such as Uniform Manifold Approximation and Projection (UMAP) [MH18].

Hast et al. [HLV19] used kernel density estimation (KDE) [CHTT96] and watershed transform on the inverse image to find clusters in a two-dimensional image space. Alternatively, Mean-Shift [CM02, FH75] could be used that also finds the number of clusters depending on the size of the Gaussian Kernel chosen. However, as will be investigated further herein, other algorithms that require specifying the exact number of clusters, such as K-means [HW79] could also be used. Nevertheless, depending upon the problem at hand, other similar algorithms such as DBSCAN [EKSX96] can also be employed.

It is also studied that the clustering techniques that work well for balanced dataset, such as MNIST [LCB10], does not perform well for imbalanced data, where there is an imbalance in the frequency of occurrence of labels in the dataset. Therefore, this work does not rely on automatic cluster selection based on Silverman's rule of thumb for computing the bandwidth h of the clustering

$$h = \left(\frac{4\sigma^5}{3n} \right)^{1/5} \tag{9}$$

where σ is the standard deviation of n samples.

3 THE PROPOSED APPROACH

Instead of using the bandwidth for clustering, better performance was achieved by computing k clusters, striving for these clusters to contain a certain predefined number of features n_f. An intuitive choice is therefore to use K-means clustering approach. Experimentally, it was found that $n_f = 40$ was by large an optimal choice for the number of clusters. However, typically this value depends on the data, and can therefore be evaluated in the learning process with respect to the type of recognition task at hand.

A drawback of K-means clustering is that it finds k clusters by initialising k random seed points, and is not confident if the same clusters will be found when the algorithm is executed multiple times. Since repeatability is important, a deterministic approach was devised that makes use of Kernel Density Estimation (KDE). In general, a certain value of σ is used to splat Gaussians on a

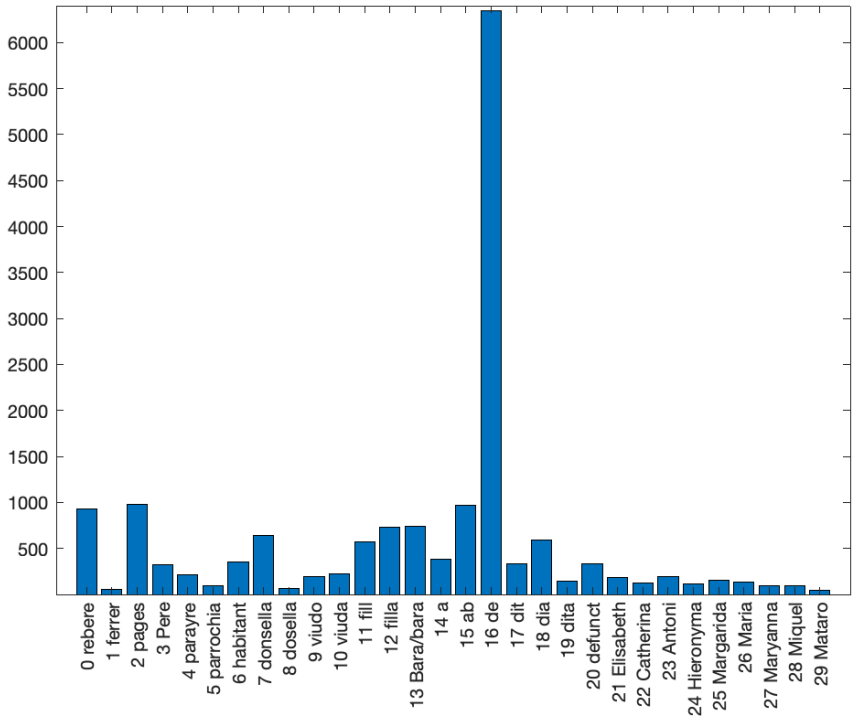


Figure 1: Distribution of words in the "imbalanced" dataset.

small image. The cluster centres are estimated by non-maximum suppression. Depending on the number of maximum (clusters) found, the σ is adjusted, and the process is repeated until k cluster centers are at hand. These are subsequently used to initialise the seed points for K-means, in order to make the algorithm find clusters centered around those maximum points.

The same adaptive procedure can also be used for the original idea of KDE and watershed to obtain k clusters, and are therefore referred to as "adaptive" in our experiments. The original approach was used in the experiments so that the improvement by the adaptive approaches could also be evaluated.

Furthermore, a self-organizing map (SOM) [Koh82] can be used, which is a dimensionality reduction technique based on unsupervised competitive learning of an artificial neural network (ANN). It generates a 2D map representation of an input space of the training samples, where the features are placed in buckets. They also demonstrate well for finding k clusters, with an advantage of having the embedding itself as part of the clustering process. However, the disadvantage is that SOM cannot be used to produce elegant scatter plots like t-SNE, as the points end up in the buckets. Nonetheless, an improved version of SOM can be used to visualise the scatter plots, called as EmbeddSOM [KKV19], which is based on FlowSOM [VGCVH*15].

SOM was configured in a way that it would strive to use a $n \times m$ map, where $n \times m = k$. However, since it is rather slow and impractical to be used for large classes,

Figure 2: The placement of each word in the 90×160 rectangular bounding box. The background is removed but the word is not binarised.

an upper limit of $n = m = 6$ was set. Similarly, an upper limit for K-means was set to a maximum of $k = 40$. All of these limits were set ad-hoc and can be changed depending on the dataset at hand. Nonetheless, all the three approaches are therefore referred to as "Adaptive" since they adaptively set the number of wanted clusters depending on the size of each class.

4 INTRODUCING THE NEW DATASET

To the best of authors knowledge and taking inspiration from the MNIST dataset, this is the first attempt at creating a dataset for handwritten word recognition for a public research domain, which is based on the Esposalles database [RFS*13]. In order to render the dataset more significant and challenging, 30 words were chosen, producing a total of 16354 word images, where some words are very similar in nature, and Figure 1

Figure 3: The 51 occurrences of the word "ferrer", with highlights on different handwriting styles. It can be noted how the character "f" is written in different styles.

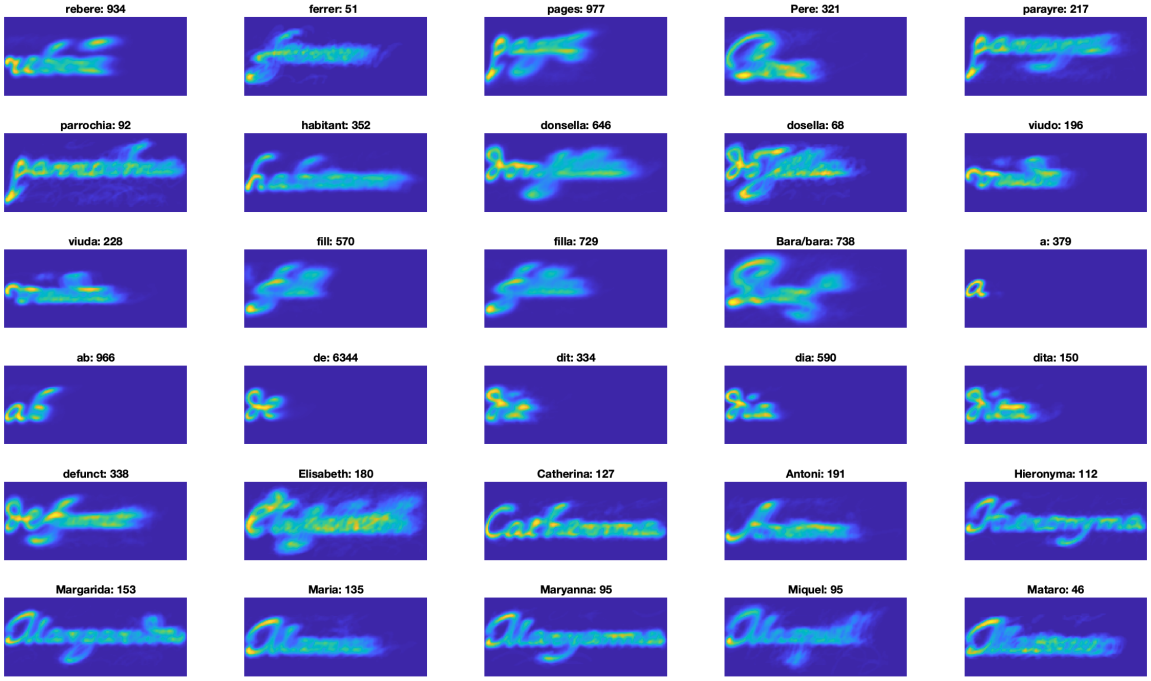


Figure 4: Heatmaps of all the words in the dataset, clustered by the respective class. The number of images per class are shown above each heatmap. Figure best viewed in color.

presents the distribution of words in the dataset. As can be seen, the dataset is heavily imbalanced, and the Shannon equitability index is only 0.7444, as compared to the balanced MNIST dataset with a Shannon equitability index of 0.9994.

Each word has been extracted using the bounding box coordinates provided as part of the Esposalles database. However, the input image is processed as follows. To begin with, background removal using [VHS17] is performed on each page, and the word is extracted.

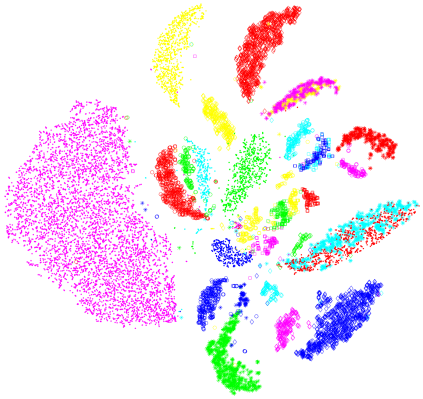
There exist some noise in the form of small blobs due to bleed through which is removed automatically, and the rectangular bounding boxes are adjusted so they perfectly encapsulate the word region [VH17]. Finally, the word is centered (or normalised), and placed to the left in a 90×160 rectangular box to accommodate words with larger length (e.g. as long as 9 characters), as shown in Figure 2. This also means that the word will be placed a little bit differently, with respect to its core, depending on whether the word has ascenders and/or descenders.

The variability within a certain class can be quite large depending on the number of writers, and whether there exists more noise in the image. The variation can be visualised as heatmaps of each class, as presented in Figure 4. These are created by adding all words belonging to a class into one single image, and the resulting values are colour coded. It can be noted that some words demonstrate a larger variation than others, as the heatmap is visually more blurry.

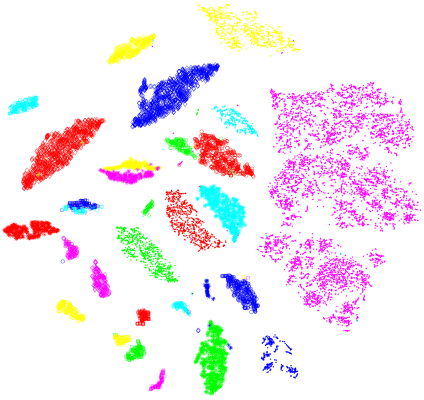
For ease of use, no stratified sampling is done in advance, or in other words, the word images are not split into predefined training and testing sets. Such splits are commonly provided for MNIST and many other popular datasets, where some also provide a validation set. However, this work allows the researchers with the flexibility to split the data in their preferred way and evaluate different properties of the dataset as well as the machine learning algorithm.

The dataset can be visualised using t-SNE or any other embedding technique, such as UMAP [MH18]. In Figure 5, t-SNE was used on the following: 5a: the word images, 5b: Histogram of Gradients (HOG) feature vectors [DT05], and 5c: mFFT, which are Fast Fourier Transform (FFT) based features with combinations of some of the most significant elements of the magnitude of the FFT.

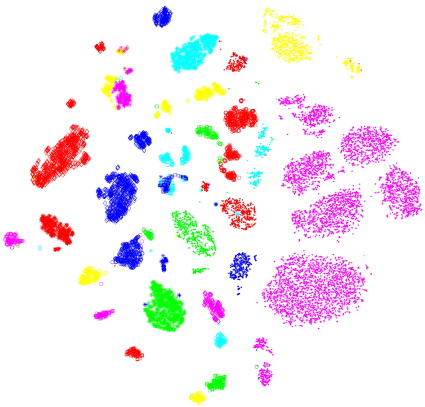
In Figure 5a, a big magenta coloured cluster can be observed where each point representing the word "de" is split into smaller clusters depending on its characteristic look in both Figure 5b and Figure 5c. Moreover, the cluster to the middle right containing the two similar words, "viudo" and "viuda", is mixed in Figure 5a, while in Figure 5b and Figure 5c it is automatically split into two distinct clusters. This suggests using efficient feature vectors instead of using simply the word images. For simplicity, hand crafted features are used, but depending upon the problem at hand and the availability of the resources, CNN based features can also be used.



(a) Word images used as features.



(b) HOG features.



(c) mFFT features.

Figure 5: Visualisation of how effectively different features can separate different classes. Figure best viewed in color.

5 RESULTS

Since the dataset is heavily imbalanced, the so-called Macro Average Arithmetic (MAA) [AAVPS13] is computed instead of the overall accuracy. The latter can be rather misleading for imbalanced datasets, when for instance a few classes with many occurrences and high accuracy can skew the overall accuracy. Figure 6 highlights that most classes particularly those with many occurrences perform well, while some with just a few occurrences yield a low accuracy. Hence, it is a better approach to compute the accuracy of each class indi-

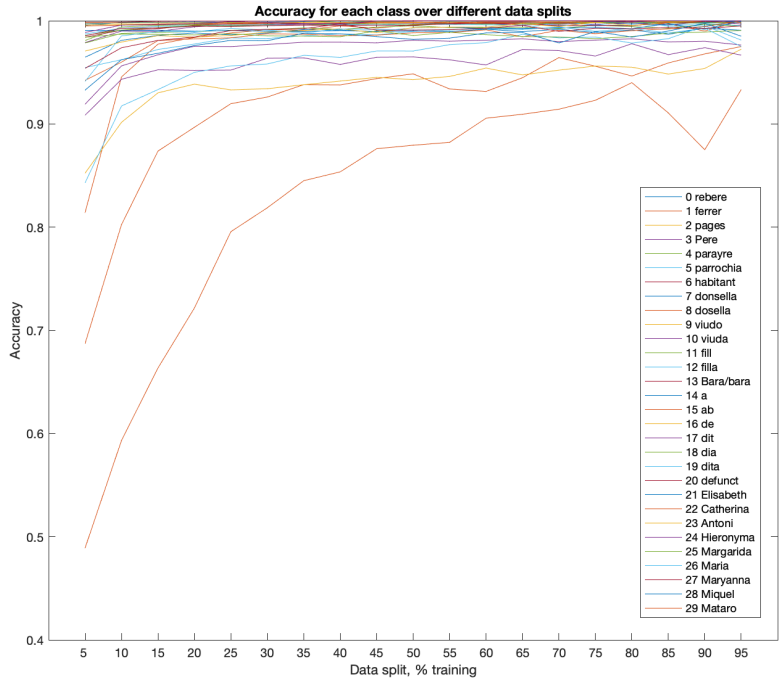


Figure 6: Accuracy computed for each class individually. Classes with many occurrences generally have higher accuracy than classes with fewer occurrences.

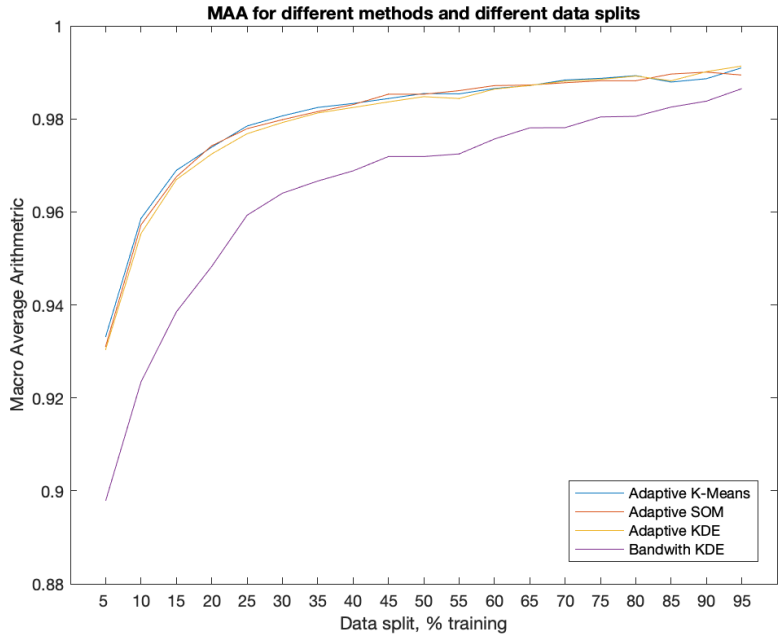


Figure 7: Macro Average Arithmetic (MAA) for different methods and different data splits.

vidually, and then compute the average. The accuracy of each class is

$$ACC_j = \frac{CC_j}{N_j} \tag{10}$$

where CC_j are the number of correctly classified words in class j and N_j is the number of samples (i.e. words) in the same class.

The MAA is defined as the arithmetic average of the partial accuracies of each class

$$MAA = \frac{\sum_{i=1}^{J} ACC_i}{J} \tag{11}$$

where J is the number of classes.

Figure 7 presents a comparison with the original method, referred to as "bandwidth KDE", since it is

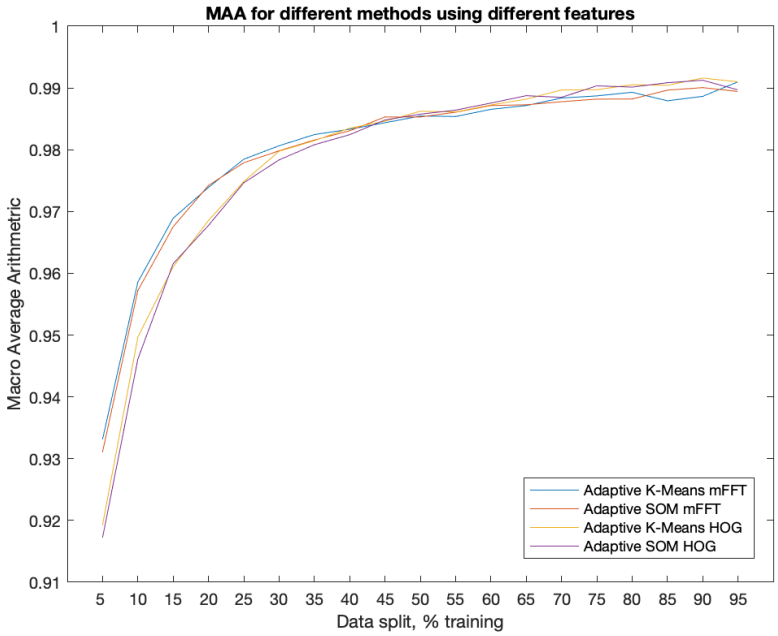


Figure 8: Macro Average Arithmetic (MAA) for different methods and different feature vectors and data splits.

based on scaling the Silverman's rule of thumb and generating clusters based on the distribution of points in 2D space. This will yield clusters with rather varying number of points. The other three approaches (SOM, K-Means and KDE) are based on the aforementioned idea of obtaining the clusters that have a similar number of points in them. As can be seen, using this idea on SOM, K-Means or KDE, all behave quite similar, and perform better than bandwidth KDE. For the proposed dataset and the setting of parameters, it is hard to choose a winner. However, it should be noted that the whole parameter space was not spanned to find the best settings, since it also depends on the dataset at hand. Hence, this is proposed for future research. All experiments were repeated 40 times for each data split and the average result is reported in the plots. This also highlights that the learning set and the testing set in each run contained different permutations of words. The same random seed was used for each group of experiments depicted by one curve, so that the curves could be compared individually on the same basis.

In Figure 7, the mFFT was used as feature vectors, which has a length of 1116, which is only 7.8% of the original size of the word images ($90 \times 160 = 14400$). It is interesting to see how the EPSC performs with different feature vectors e.g. the HOG, which is 6940 long, i.e. 47.5% of the original images. The results are presented in Figure 8, and it can be noted that the X-axis is different here in order to better observe the difference between the plots. Nonetheless, HOG seems to perform well after around a 50% split, while mFFT is more effective for less learning data, which is encouraging since they are much shorter.

6 DISCUSSION

It is observed that the EPSC handles imbalanced datasets very well, since high accuracy is obtained for most of the classes with just a few occurrences. Of course, just one dataset is not enough to provide an absolute answer. However, the proposed dataset is indeed useful and challenging, with some words classes very similar in nature. EPSC handles the imbalance in the following way. When there are several occurrences in one class, it creates subspaces that capture the variation within that class. Hence, it correctly classifies that class, and the images not belonging to the class will instead be correctly classified by subspaces belonging to other classes. That is, if it has enough learning examples to create the subspaces for those classes. Hence, having several occurrences for one class does not seem to pose big problem. Having too few, on the other hand, becomes challenging since the subspaces created do not capture the variation very well. For instance only one subspace is created for the word "ferrer" when 5% is used for learning, and that subspace is created from only 3 occurrences. Typically, the word classes that are very similar will suffer more from having a very small variation of word examples to create the subspaces from.

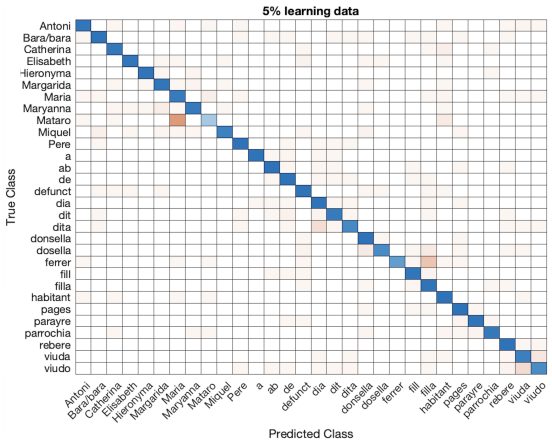
When dealing with imbalanced datasets, one can choose different strategies, such as to over-sample the minority class, under-sample the majority class or generate synthetic samples. When transcribing a document, one can use any of these strategies to make a learning model using the words already transcribed, and perform automatic transcription for the rest. How-

ever, in this paper the main focus was to investigate how well EPSC can handle imbalanced sets and which clustering approaches can be used efficiently. If under-sampling and over-sampling are not suitable strategies for EPSC (since it is desirable to keep as much variation as possible), then data augmentation of the minority classes can potentially be a solution to improve the performance. This is proposed for future research.

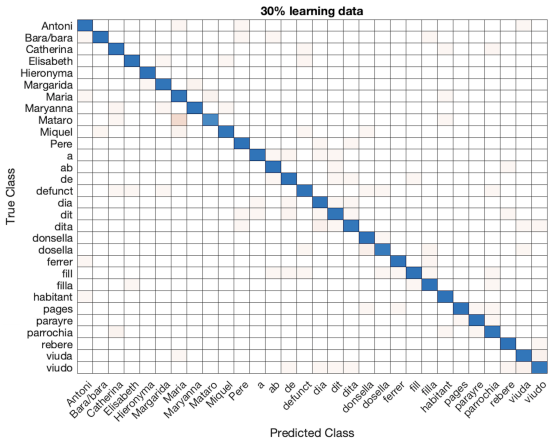
The MAA for all clustering techniques and feature vectors, reach about 98% for a data split of 30% used for training, which can be regarded as a good result for such a challenging dataset. Even for the human eye, it is at times difficult to differ handwritten words like "viudo" from "viuda". It is observed that 17 of the classes have an accuracy over 99.0% for the same data split, 14 lies over 99.5% and 7 are over 99.7%, when using K-means and mFFT. The Confusion matrices for 5%, 30%m and 60% learning data are shown in Figure 9. It can be noted that the word "Mataro" is often misclassified as "Maria", while the words "viudo" and "viuda" are interestingly much less confused.

7 CONCLUSION

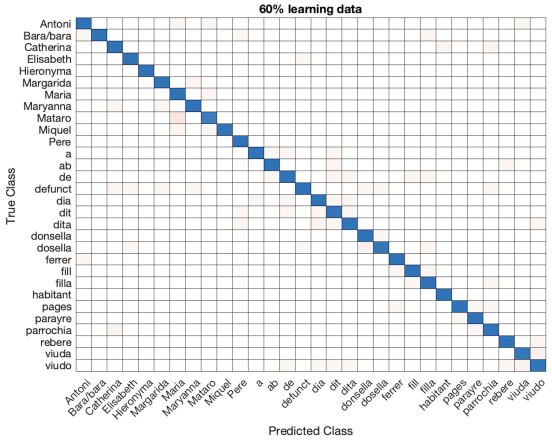
The original approach for clustering for EPSC works very well for balanced datasets. However, imbalanced datasets are generally harder to handle and three different clustering approaches for obtaining subspaces were presented and analysed in this work. Experimental results validate the performance, and all the three approaches performed significantly regardless of the feature vector being used. It did not make a big difference whether t-SNE or SOM were used as embeddings. Neither did, using the proposed adaptive versions of the original idea of clustering (using the watershed transform or K-means), add a significant difference. Hence, all three methods are good candidates for future experiments. However, SOM has the drawback of not being a visualisation technique by its own. All parameters in the EPSC were not systematically investigated for the new heavily imbalanced dataset in the current experiments. Nevertheless, a baseline was given for future experimenting using EPSC. Furthermore, handcrafted features were used as they are fast and simple, rendering the whole pipeline with EPSC fast and simple too. As future work, CNN based features would be evaluated for the given dataset. However, full pipelines of deep learning approaches also need to handle the imbalance in specific ways. Therefore, it will be an interesting area of research for further investigations on using the dataset for different data splits and machine learning algorithms.



(a) Confusion matrix with 5% learning data.



(b) Confusion matrix with 30% learning data.



(c) Confusion matrix with 60% learning data.

Figure 9: Confusion matrices for 5%, 30% and 60% learning data respectively.

8 ACKNOWLEDGMENTS

This work has been partially supported by the Riksbankens Jubileumsfond (Dnr NHS14-2068:1). The computations were performed on resources provided by SNIC through UPPMAX under project SNIC 2020/15-177. The authors wish to thank Raphaela Heil for fruitful discussions in the development of the ideas presented. The presented dataset is publicly available at `https://andershast.com/datasets/`

9 REFERENCES

[AAVPS13] Alejo R., Antonio J. A., Valdovinos R. M., Pacheco-Sánchez J. H.: Assessments metrics for multi-class imbalance learning: A preliminary study. In *Pattern Recognition* (Berlin, Heidelberg, 2013), Carrasco-Ochoa J. A., Martínez-Trinidad J. F., Rodríguez J. S., di Baja G. S., (Eds.), Springer Berlin Heidelberg, pp. 335–343.

[ADRS*19] Arrieta A. B., D'iaz-Rodr'iguez N., Ser J. D., Bennetot A., Tabik S., Barbado A., Garc'ia S., Gil-L'opez S., Molina D., Benjamins R., Chatila R., Herrera F.: Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. *ArXiv abs/1910.10045* (2019).

[CATvS17] Cohen G., Afshar S., Tapson J., van Schaik A.: EMNIST: an extension of MNIST to handwritten letters. *CoRR abs/1702.05373* (2017).

[CBK*18] Clanuwat T., Bober-Irizar M., Kitamoto A., Lamb A., Yamamoto K., Ha D.: Deep learning for classical japanese literature. *CoRR abs/1812.01718* (2018).

[CHTT96] Carbon M., Hallin M., Tat Tran L.: Kernel density estimation for random fields: the l 1 theory. *Journal of nonparametric Statistics 6*, 2-3 (1996), 157–170.

[CM02] Comaniciu D., Meer P.: Mean shift: a robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence 24*, 5 (May 2002), 603–619.

[CPC19] Carvalho D. V., Pereira E. M., Cardoso J. S.: Machine learning interpretability: A survey on methods and metrics. *Electronics 8*, 8 (Jul 2019), 832.

[DKMJ18] Dutta K., Krishnan P., Mathew M., Jawahar C.: Improving cnn-rnn hybrid networks for handwriting recognition. In *2018 16th International Conference on Frontiers in Handwriting Recognition (ICFHR)* (2018), IEEE, pp. 80–85.

[DT05] Dalal N., Triggs B.: Histograms of oriented gradients for human detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)* (June 2005), vol. 1, pp. 886–893 vol. 1.

[EKSX96] Ester M., Kriegel H.-P., Sander J., Xu X.: A density-based algorithm for discovering clusters a density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining* (1996), KDD96, AAAI Press, pp. 226–231.

[FH75] Fukunaga K., Hostetler L.: The estimation of the gradient of a density function, with applications in pattern recognition. *IEEE Transactions on Information Theory 21*, 1 (January 1975), 32–40.

[GSC*19] Gunning D., Stefik M., Choi J., Miller T., Stumpf S., Yang G.-Z.: Xai—explainable artificial intelligence. *Science Robotics 4*, 37 (2019).

[HL20] Hast A., Lind M.: Ensembles and cascading of embedded prototype subspace classifiers. *Journal of WSCG 28*, 1/2 (2020), 89–95.

[HLV19] Hast A., Lind M., Vats E.: Embedded prototype subspace classification : A subspace learning framework. In *The 18th International Conference on Computer Analysis of Images and Patterns (CAIP)* (2019), Lecture Notes in Computer Science, pp. 581–592.

[HW79] Hartigan J. A., Wong M. A.: Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics) 28*, 1 (1979), 100–108.

[KDJ18] Krishnan P., Dutta K., Jawahar C.: Word spotting and recognition using deep embedding. In *2018 13th IAPR International Workshop on Document Analysis Systems (DAS)* (2018), IEEE, pp. 1–6.

[KKV19] Kratochvíl M., Koladiya A., Vondrášek J.: Generalized embedsom on quadtree-structured self-organizing maps. *F1000Research* (2019).

[KLR*77] Kohonen T., Lehtiö P., Rovamo J., Hyvärinen J., Bry K., Vainio L.: A principle of neural associative memory. *Neuroscience 2*, 6 (1977), 1065 – 1076.

[KO76] Kohonen T., Oja E.: Fast adaptive formation of orthogonalizing filters and associative memory in recurrent networks of neuron-like elements. *Biological Cybernetics 21*, 2 (Jun 1976), 85–95.

[Koh82] Kohonen T.: Self-organized formation of topologically correct feature maps. *Biological Cybernetics 43*, 1 (Jan. 1982), 59–69.

[Kri19] Krishnan M.: Against interpretability: a critical examination of the interpretability problem in machine learning. *Philosophy & Technology* (2019).

[KRMV76] Kohonen T., Reuhkala E., Mäkisara K., Vainio L.: Associative recall of images. *Biological Cybernetics 22*, 3 (Sep 1976), 159–168.

[Laa07] Laaksonen J.: *Subspace classifiers in recognition of handwritten digits*. G4 monografiaväitöskirja, Helsinki University of Technology, 1997-05-07.

[LCB10] LeCun Y., Cortes C., Burges C.: Mnist handwritten digit database. *AT&T Labs [Online]. Available: http://yann. lecun. com/exdb/mnist 2* (2010).

[MH08] Maaten L. v. d., Hinton G.: Visualizing data using t-sne. *Journal of machine learning research 9*, Nov (2008), 2579–2605.

[MH18] McInnes L., Healy J.: UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. *ArXiv e-prints* (Feb. 2018).

[OK88] Oja E., Kohonen T.: The subspace learning algorithm as a formalism for pattern recognition and neural networks. In *IEEE 1988 International Conference on Neural Networks* (July 1988), vol. 1, pp. 277–284.

[RFS*13] Romero V., Fornés A., Serrano N., Sánchez J. A., Toselli A. H., Frinken V., Vidal E., Lladós J.: The ESPOSALLES database: An ancient marriage license corpus for off-line handwriting recognition. *Pattern Recognition 46*, 6 (2013), 1658–1669.

[SF16] Sudholt S., Fink G. A.: Phocnet: A deep convolutional neural network for word spotting in handwritten documents. In *ICFHR* (2016), IEEE Computer Society, pp. 277–282.

[Sha18] Shapshak P.: Artificial intelligence and brain. *Bioinformation 14*, 1 (2018), 38.

[VGCVH*15] Van Gassen S., Callebaut B., Van Helden M. J., Lambrecht B. N., Demeester P., Dhaene T., Saeys Y.: Flowsom: Using self-organizing maps for visualization and interpretation of cytometry data. *Cytometry Part A 87*, 7 (2015), 636–645.

[VH17] Vats E., Hast A.: On-the-fly historical handwritten text annotation. In *Document Analysis and Recognition (ICDAR), 2017 14th IAPR International Conference on* (2017), vol. 8, IEEE, pp. 10–14.

[VHS17] Vats E., Hast A., Singh P.: Automatic document image binarization using bayesian optimization. In *Proceedings of the 4th International Workshop on Historical Document Imaging and Processing* (2017), ACM, pp. 89–94.

[WLK*67] Watanabe W., Lambert P. F., Kulikowski C. A., Buxto J. L., Walker R.: Evaluation and selection of variables in pattern recognition. In *Computer and Information Sciences* (1967), Tou J., (Ed.), vol. 2, New York: Academic Press, pp. 91–122.

[WP73] Watanabe S., Pakvasa N.: Subspace method in pattern recognition. In *1st Int. J. Conference on Pattern Recognition, Washington DC* (1973), pp. 25–32.

[XRV17] Xiao H., Rasul K., Vollgraf R.: Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *CoRR abs/1708.07747* (2017).