ISSN 1213-6972 *Volume 26, Number 1, 2018*

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 *Volume 26, Number 1, 2018*

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
2018

Aburumman,N. (France)
Assarsson,U. (Sweden)
Ayala,D. (Spain)
Azari,B. (Germany)
Benes,B. (United States)
Benger,W. (United States)
Bouatouch,K. (France)
Bourke,P. (Australia)
Carmo,M. (Portugal)
Carvalho,M. (Brazil)
Daniel,M. (France)
de Geus,K. (Brazil)
De Martino,J. (Brazil)
de Souza Paiva,J. (Brazil)
Dingliana,J. (Ireland)
Durikovic,R. (Slovakia)
Feito,F. (Spain)
Feng,J. (China)
Ferguson,S. (United Kingdom)
Galo,M. (Brazil)
Galo,M. (Brazil)
Garcia Hernandez,R. (Germany)
Garcia-Alonso,A. (Spain)
Gavrilova,M. (Canada)
Gdawiec,K. (Poland)
Giannini,F. (Italy)
Goncalves,A. (Portugal)
Gudukbay,U. (Turkey)
Hernandez,B. (United States)
Horain,P. (France)
Charalambous,P. (Cyprus)
Juan,M. (Spain)
Kanai,T. (Japan)
Klosowski,J. (United States)
Kurt,M. (Turkey)
Lee,J. (United States)
Lisowska,A. (Poland)
Lobachev,O. (Germany)
Luo,S. (Ireland)
Marques,R. (Spain)
MASTMEYER,A. (Germany)
Metodiev,N. (United States)
Molla,R. (Spain)
Montrucchio,B. (Italy)
Muller,H. (Germany)
Oliveira,J. (Portugal)
Oyarzun Laura,C. (Germany)
Papaioannou,G. (Greece)
Patow,G. (Spain)
Pedrini,H. (Brazil)
Peytavie,A. (France)
Puig,A. (Spain)
Ramires Fernandes,A. (Portugal)

Renaud,c. (France)
Ribeiro,R. (Portugal)
Richardson,J. (United States)
Rodrigues,J. (Portugal)
Rojas-Sola,J. (Spain)
Sanna,A. (Italy)
Santos,L. (Portugal)
Segura,R. (Spain)
Skala,V. (Czech Republic)
Sousa,A. (Portugal)
Subsol,G. (France)
Szecsi,L. (Hungary)
Tavares,J. (Portugal)
Teschner,M. (Germany)
Thalmann,D. (Switzerland)
Todt,E. (Brazil)
Tokuta,A. (United States)
Trapp,M. (Germany)
Vanderhaeghe,D. (France)
Vidal,V. (France)
Vierjahn,T. (Germany)
Wuensche,B. (New Zealand)
Wuethrich,C. (Germany)
Xu,K. (China)
Yin,Y. (United States)
Yoshizawa,S. (Japan)
Zwettler,G. (Austria)

Journal of WSCG
Vol.20, No.1

Contents

ISSN 1213-6972 Journal of WSCG http://www.wscg.eu/ Vol.26, 2018, No.1

Analyzing Histone Modifications Using Tiled Binned Clustering and 3D Scatter Plots

Dirk Zeckzer[1]
zeckzer@informatik.uni-leipzig.de

Daniel Wiegreffe[1,2]
daniel@bioinf.uni-leipzig.de

Lydia Müller[2,3]
lydia@bioinf.uni-leipzig.de

[1]Image and Signal Processing Group, Leipzig University

[2]Bioinformatics, Leipzig University

[3]Natural Language Processing Group, Leipzig University

ABSTRACT

A major goal in epigenetics is understanding how cells differentiate into different cell types. Besides the increase of individual data sets, the amount of replicated experiments generating a tremendous amount of data is ever increasing. While biologists primarily analyze their data on the highest level using statistical correlations or on the lowest level analyzing nucleotide sequences, determining the fate of histone modifications during cell specification necessitates improved analysis capabilities on one or more intermediate levels. For this type of analysis, it proved to be very useful to use tiled binned scatter plot matrices showing binary relationships or to use tiled binned 3D scatter plots showing ternary relationships. Quarternary or general n-ary relationships are not easily analyzable using visualization techniques like scatter plots, only. Therefore, we augmented existing clustering methods with the tiling and binning idea enabling the analysis of n-ary relationships. Analyzing the changes of histone modifications comparing two cell lines using tiled binned clustering, we found new, unknown relations in the data.

Keywords

Clustering, Binning, Tiles, 3D Scatterplot, Biological Visualization, Histone Modifications

1 INTRODUCTION

Epigenetics, "The study of heritable changes in gene expression that are not mediated at the DNA sequence level" [5] is a very important research area. As part of epigenetics, researcher study histone modifications. Histone modifications are important for the development of the cells regulating the transcription states of genes and thereby their overall expression patterns. The evolution of certain histone modifications plays an important role during the differentiation of cells, e.g., from embryonic stem cells to embryonic fibroblasts or to neural progenitor cells [23, 27, 28].

In order to analyze this fate of histone modifications, one or more intermediate levels of data analysis are needed that go beyond the high-level statistical correlations and the low-level sequence-level analyses. Steiner et al. [23] proposed a visualization based on self-organizing maps (SOMs) on an intermediate level. Moreover, Zeckzer et al. proposed tiled binned scatter plot matrices (TiBi-SPLOM) [28] and tiled binned 3D scatter plots (TiBi-3D) [27] fostering analyzing the fate of epigenetic marks during differentiation. While our previous approaches [23, 27, 28] provide effective and efficient visualizations of histone modification data supporting these intermediate level analyses, there is still room for improvements.

To go beyond *binary* and *ternary* relationship analyses supported by these approaches and thus supporting general *n-ary* relationship analysis, we propose tiled binned clustering. The results of this processing step are then displayed using TiBi-3D-like visualizations. Using visualizations based on TiBi-SPLOM could be used as an alternative but are not yet implemented.

Concretely, the contributions of this paper are:

- *Visual Analysis*
  - A tiled binned clustering method that together with an adapted tiled binned 3D scatter plot fosters analyzing *n*-ary relations on histone modification data.
  - A complete work flow from the raw data to the final visualization and interaction implemented in TiBi-Cluster.

https://doi.org/10.24132/JWSCG.2018.26.1.1

- *Biology*
  - Comparison of 6 histone modifications between mesendodermal cells and mesenchymal stem cells based on data from the NIH Roadmap Epigenomics project [20].

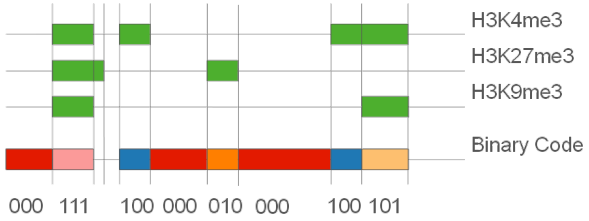
2 BIOLOGICAL BACKGROUND

2.1 Data Sets

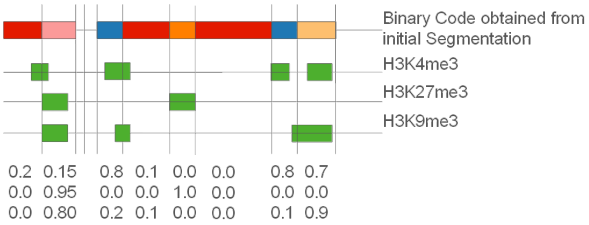
As data sets, we chose mono-, di-, and trimethylation of histone H3 at lysine K4 (H3K4me1, H3K4me2, H3K4me3), acetylation and trimethylation of histone H3 at lysine K27 (H3K27ac, H3K27me3), and trimethylation at histone H3 at lysine K36 (H3K36me3). Those data sets are available on the NIH Roadmap Epigenomics project's web portal (`http://egg2.wustl.edu/`, Release 9) [20]. The first three modifications are associated with enhancers and transcription. H3K4me1 is found at enhancers in the promoter regions. It is associated with silencing and even demarcates H3K4me3 [4]. H3K4me2 is a mark for transcription factor binding sites and increases the binding of transcription factors to their binding site. H3K4me3 is found at promotors and enhances the binding of the polymerase complex. Acetylation at H3K27 changes the charge of the histone, and thus, leads to an open chromatin formation enabling transcription. It is furthermore found with H3K4me2 at transcription factor binding sites. H3K27me3 is a mark indicating repressed transcription. H3K36me3 is known to enrich at splice sites. In this way, it indicates not only active transcription but also splicing events at specific splice sites.

2.2 Preprocessing

To measure those epigenetic marks between multiple cell types and cell lines, ChIP-seq (chromatin immunoprecipitation sequencing) is performed for every replicate. The results of this procedure consist of billions of short DNA sequences distributed over the whole genome marking the associated histone. We downloaded the raw data of these ChIP-seq experiments. During the next step, the data is mapped against the human reference genome version hg19 with the mapper segemehl [11] with 80% accuracy. Afterwards, we used the Picard Tools [1] to remove PCR duplicates. These steps result in the short DNA sequences being annotated to specific regions in the human genome. As ChIP-seq data contains small errors and some DNA sequences are falsely aligned to regions within the genome, only the significant peaks within the genome-wide distribution of the mapped DNA sequences are treated as a valid histone modification mark (the method being applied is called 'peak-calling'). We used the peak-caller Sierra Platinum [18] to extract those significant regions having support by multiple replicates.



(a) Example of the binary code resulting from segmentating the reference cell line (or reference cell type) based on 3 histone modifications (marks). The Binary Code represents the type of modification pattern present in the chromatin segment (here: 8 different codes).



(b) Example of the additional data annotation based on the segmentation of the reference data. The vectors represent the type of modification pattern present in this segment relative to the reference. The same three histone modifications for a different cell type are used here.

Figure 1: Overview of the two-stage segmentation procedure: Segmentation of the reference data and annotation of the additional data.

Based on these peak calls we calculated a whole genome segmentation. This method was described in detail by Steiner et al. [23]. During this process, all peaks are projected onto the genome (see Figure 1(a)). Nucleotide sequences with the same combination of peaks from different histone modifications as well as nucleotide sequences without any peaks are the *segments*. The approximated length of DNA wrapped around a histone complex together with the linker between two of them is approximately 200 nucleotides. As shorter segments are most likely noise, they are discarded during segmentation.

The generated segments are described by their location using a unique identifier. Besides this identifier, each segment stores a multi-dimensional data vector. The first data column contains the code from the reference cell type used during segmentation (see Figure 1(a)). In our case, the histone modification peaks for H3K4me1, H3K4me2, H3K4me3, H3K27ac, H3K27me3, and H3K36me3 of the mesendodermal stem cells were used. The subsequent columns contain the overlap of each histone modification for all other cell types (Figure 1(b)). In our case, six columns are created overlapping the histone modification peaks for H3K4me1, H3K4me2, H3K4me3, H3K27ac, H3K27me3, and H3K36me3 of the mesenchymal stem cells with the previously created segments. Furthermore, the CG (cytosine-guanine dinucleotide) density and the length of each segment are stored in each segment. This

reference-based segmentation is possible, since the mesenchymal stem cells arise from the mesendodermal cells and changes of histone marks indicate variances due to cell differentiation. The segmentation results in 1,419,149 data points.

3 RELATED WORK

Due to size of the available data set, the visualization of chromatin data is mainly based on two approaches: a coarse grained high level analysis and a low level analysis at single gene level. The high level analysis is supported by heatmaps that can visualize the correlations between multiple modifications over the whole genome (used by e.g., Kuang et al. [15]). It is possible to visualize correlations between functional groups of the genome either for multiple modifications at a single point in time or for a single cell type, or for a single modification at multiple points in time or for multiple cell types. The low level analysis can be supported by using a genome browser like WashU [29] or UCSC [14]. A genome browser annotates data tracks linearly to a specific genomic region based on the chromosome positions. With these tracks it is possible to analyze multiple modifications, time points, and cell types within a single visualization but only for a limited region. The comparison between these regions is left to the expert and is tedious. Both visualization techniques do not provide a semi-global trend analysis that supports both the comparison between multiple modifications, and multiple time points or multiple cell types.

Recently, Steiner et al. [23] and Zeckzer et al. [27, 28] presented three new approaches for the analysis of chromatin data supported by interactive visualizations. Moreover, Gerighausen et al. [9] applied a combination of k-means clustering and principal component analysis (PCA) to chromatin data. To our knowledge, these four approaches are the only ones available for intermediate level analysis of chromatin data and the fate of histone modifications.

Previous work related to the visualization of multivariate data in general was reviewed by them, too, including purely visual methods as well as dimensionality reduction and clustering approaches combined with the visualizations of their outcome [10]. The latter methods were tested by us before (unpublished, see also Section 5 for the clustering methods and the clustering result assessments considered) but without conclusive results. These and other unrelated techniques are also described in the recent state-of-the-art report by Liu et al. [16]. The comparison of different scatter plot techniques and dimension reduction methods were extensively reviewed by Sedlmair et al. [22]. They propose the usage of 2D scatter plots for most cases except for artificial or grid data sets for which 3D scatter plots outperform the alternative 2D scatter plot techniques and scatter plot matrices. The data obtained after tiled-binned clustering (and also after tiling and binning alone) falls exactly into the latter category, namely grid data, where—according to Sedlmair et al. [22]—3D scatter plots outperform their 2D counterparts.

4 TASK AND GOALS

4.1 Task

The task of the analyst—biologist or bioinformatician—comprises the analysis of the fate of histone modifications. The underlying data consists of $10^5 - 10^7$ data points (genome segments) with multiple dimensions per cell type (in our case: 6 different histone modifications for mesendodermal cells and mesenchymal stem cells, respectively) and additional dimensions characterizing the segments (here: CG density and segment length). Visualization and interaction facilities as well as the methodology for creating the visualization are developed such that the analyst is supported in creating and verifying hypotheses as well as in gaining additional insights into the changes in epigenetic marks during cell differentiation. Creating hypotheses and gaining additional information are exploratory tasks that are complemented by the verification task.

4.2 Goals

The major goal of this methodology including the visualization and the interaction facilities is supporting the analyst in performing the task. Additionally, visualization as well as interaction facilities are designed following the recommendations from information visualization literature [19,25,26]. The literature provides a set of guidelines any visualization should adhere to (see also Zeckzer et al. [27,28]):

1. The methodology and the visualization should be independent of data specific properties the only assumption being that the number of dimensions is fixed for all data points. This allows for a flexible work flow and flexible visualizations that in principle can also handle data sets from other domains.

2. The approach should use real screen estate efficiently.

3. The approach should provide a good overview of the data.

4. The approach should ease the identification of pattern in the data.

5. The visualizations should be fast to create and fast to interact with.

5 TIBI-CLUSTER—METHODOLOGY

Given tabular data consisting of data points (rows) and attributes (columns), the amount of data values (cells) is the product of the number of data points and the number of attributes. Frequently, the size of the data to be analyzed is too large for manual inspection. Visualization and interaction associated with the visualization can mitigate this problem and support analyses of small, medium, and large data. If the size of the data is too large, however, the amount of pixels on the screen is no longer sufficient to display all data points. To be able to show a summary or all of the data, three principal methods are often applied, among others: using tiled displays (i.e., 2, 3, or more displays), reducing the number of attributes (dimensionality reduction), or clustering the data points.

In the case of ChIP-Seq data preprocessed as described in the biological background (Section 2), clustering [13] failed to provide any useful insights. Testing the divisive clustering methods k-means++ [2, 17], k-median [13], and even consensus clustering [7, 8] in conjunction with clustering assessment strategies and indices like Davies Bouldin Index [6], Hubert Statistics [12], Dunn Index [3], and silhouettes [21] (see also Jain and Dubes [13]) lead to a partitioning of the data with limited expressiveness [9]. Applying the visualization strategies TiBi-SPLOM [28] and TiBi-3D [27] showed why: the characteristics of the data makes the direct application of clustering unsuitable. The data is very noisy. Even applying hierarchical clustering methods would not be beneficial in this context.

As a consequence of clustering methods not being suitable for the original data and the success of the idea of tiled-binning of the data for visualizing the data, a combination of both was conceived. The resulting methodology applies tiling and binning to the original data *before* applying clustering methods. The benefit is that the noise in the data is smoothed and that the inhomogeneity of the data is not a problem any more. Overall it shows, that tiling and binning the original data as a first step benefits both visualization and clustering methods.

The methodology for creating visualizations supporting the analysis of histone modification fate during cell differentiation is based on the following pipeline:

1. Loading the data (Section 6.1).
2. Assigning the individual data points to tiles of bins (Section 6.1).
3. Filtering bins (Section 6.2).
4. Clustering the filtered bins (Section 6.3).
5. Mapping the resulting d-dimensional data to the three dimension of a 3D tiled binned scatter plot (Section 6.4).

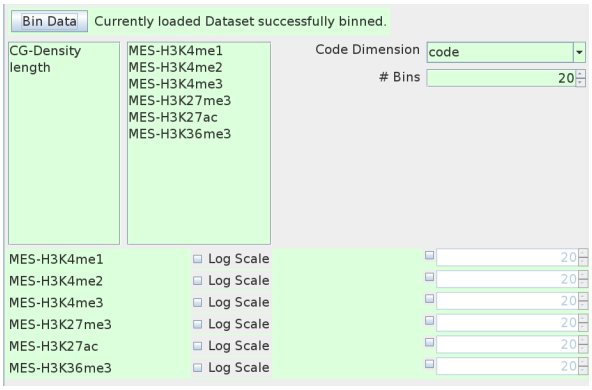


Figure 2: Selecting the dimensions for binning and the number of bins. The two list boxes show the available (left) and the selected (right) dimensions, respectively (middle left part). Clicking on one of the entries moves it to the opposite side, i.e., selected dimensions are deselected and deselected ones are selected. Moreover, the code dimension can be selected using the drop-down box and the number of bins for all selected dimensions can be chosen (middle right). Further, for each of the selected dimensions, the number of bins and whether the dimension should be scaled logarithmically can be selected individually (bottom). After selecting all items, binning can be started by pressing the "Bin Data" button (top). After binning, the background changes from red (changes pending) to green (changes applied), the latter state being shown in this screen shot.

The user interface for setting up and performing each step is described next (Sections 6.1–6.3). It is complemented by the visualization of the tiled binned clustering results (Section 6.4), interaction mechanisms allowing to change the setup of the visualization (Section 6.4), as well as tables showing detailed information about the tiles, the segments of a tile, and the centroids of the final cluster division (Section 6.5). Each step of the workflow and the tables are represented as consecutively ordered tabs within the GUI.

6 TIBI-CLUSTER—USER INTERFACE

All panels provided by TiBi-Cluster that were used to obtain the biological insights reported (Section 7) are also shown and explained in the accompanying video.

6.1 Creating Tiled-Bins of the Data

First, the data is loaded using the menu-item "File – Open". Each data point is a segment. Let n_s be the number of segments and let s_i, $i \in \{1,\ldots,n_s\}$ be a segment. Each segment has assigned n_a attributes a_j, $j \in \{1,\ldots,n_a\}$, one of them being its segmentation code $c(s_i)$ and one of them being its length. Altogether there are $n_s \cdot n_a$ values in the table. Subsequently, each attribute is referred to as a *dimension*.

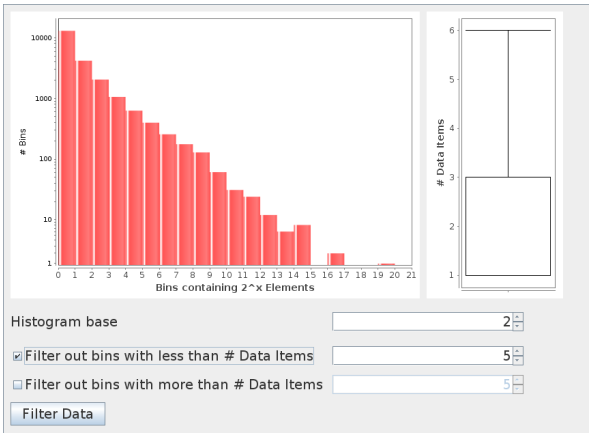


Figure 3: Filtering panel. Top-Left: histogram showing a double-logarithmic plot of the number of elements per bin (x-axis) versus the number of bins having that much elements (y-axis). Top-Right: box plot showing the distribution of elements per bin. Bottom: interaction items for changing the base of the logarithm of the number of elements per bin (x-axis), as well as the lower and the upper boundary of the filter range (number of elements per bin). Pressing the "Filter Data" button applies the filter to the data and the next step can be parameterized and performed.

Next, binning is performed. Figure 2 shows the 'Binning' tab that enables the user selecting the dimensions that should be analyzed. On the right hand side (middle), the dimension representing the segment codes can be chosen. On the left hand side (middle), the dimensions to be analyzed can be selected. In the lower part, the number of bins can be specified either uniformly for all selected dimensions or individually for each selected dimension separately. Additionally, logarithmic scaling can be chosen for each dimension selected. Binning is started by pressing the "Bin Data" button (top).

Binning itself divides the range of values of each dimension a_j into the specified number of intervals. The cross product of the intervals of all attributes gives the set of all possible bins. Then, each segment s_i is assigned to the bin b it belongs to.

Further, each bin is divided into several tiles. Each tile represents one of the n_c possible segment codes. Thus, each segment s_i is assigned to the tile $t_k, k \in \{1, \ldots n_c\}$ of its bin b according to its code $c(s_i) =: k$. In other words, each bin contains n_c tiles and thus can be written as an n_c-dimensional vector $\vec{b} = (t_1, \ldots, t_{n_c})^T$. Each $t_k, k = 1, \ldots, n_c$ is the k-th tile of bin b and represents the number of segments put into that tile.

The binning step can be repeated any time. Then, the subsequent steps have to be repeated, too.

6.2 Filtering the Bins

The second step after binning is filtering. Filtering is performed in the 'Filtering' tab before clustering the data to enable the analyst to remove bins that are not in the focus of the current analysis early. Moreover, filtering before clustering removes "noise" from the data that otherwise might influence the quality of the resulting clustering. The step needs to be under the control of the analyst as domain knowledge is necessary to select the interesting bins and to remove the remaining ones. The removed bins are marked as filtered. Thus, while they are not included in the clustering process, they can still be displayed in the visualization forming their own "filtered bins" cluster. Of course the analyst can choose to keep all bins, perform the subsequent steps, and come back to filtering after having a first idea of the data being analysed.

Selecting the bins to keep and those to remove is supported by two simple and effective visualizations (Figure 3, top). The first one (displayed on the left side) is a histogram with double logarithmic scales showing the size of the bins on the x-axis and the number of bins of the respective size on the y-axis. To improve the understanding of the bin-size distribution, a box plot is shown (on the right side). It gives the median, the lower and upper quartiles as well as the whiskers that represent the 1.5 IQR variability of the distribution of the number of segments per bin. At the bottom, the base of the logarithm used for the number of bin elements (x-axis) can be chosen. Moreover, the minimum and the maximum of segments per bin can be chosen. Bins with less than the minimum and more than the maximum of segments per bin are removed. The same can be achieved by graphically selecting an interval in the histogram. However, the spinners allow to provide exact values for the borders of the selected range.

Filtering is performed by pressing the "Filter Data" button. The filtering range can be changed and applied any time. Then, the clustering step has to be repeated, too.

6.3 Clustering the Filtered Bins

The most complex step in the methodology—algorithmically as well as with respect to the computational resources needed—is the clustering of the filtered bins. This is reflected by the interface in the 'Clustering' tab for setting the clustering parameters (Figure 4, description top-down). First the analyst can decide whether to use a single clustering method or to use consensus clustering based on the selected single clustering method. Next, the analyst chooses the parameters for the single clustering method. Currently, k-means and k-median can be selected as single clustering method. For both, the empty-cluster strategy—the strategy to apply if an empty cluster is produced during the process—determines the outcome of the clustering. The available empty-cluster strategies are to split the cluster with the largest variance or with the largest number of data points. Alternatively, a new

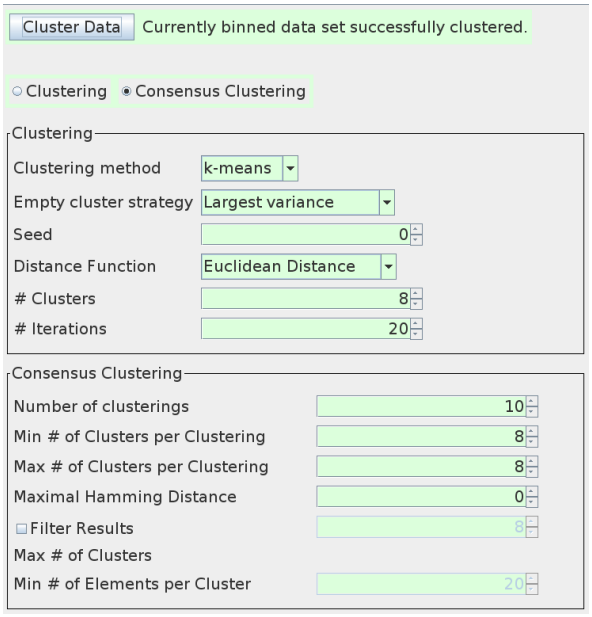


Figure 4: Interface for selecting the clustering method and its associated parameters. The clustering is started by pressing the "Cluster Data" button. Green backgrounds represent unchanged parameters, while red backgrounds represent parameters that changed. If the current parameter setting was used for the current clustering, button and label background are green (as shown here), and red otherwise.

centroid can also be determined by locating the farthest data point from the centroid of the empty cluster and then assigning it as a new centroid. As both clustering methods are based on random selection of the first point, the seed for the random number generator can be selected, This allows to obtain different results while all results can be reproduced.

Next, a distance function has to be selected. As defined in Section 6.1 we use vectors that represent the tiled and binned data for the clustering. The available functions for computing the distance d between two such vectors $\vec{b_1}$ and $\vec{b_2}$ are listed in Table 1. In addition, the value p can be specified when using the "LP-Norm".

K-means as well as k-median clustering require the number of clusters k to be specified before clustering. Moreover, it is useful to provide a maximum number of iterations.

If consensus clustering is selected, further parameters can be chosen by the analyst. Consensus clustering is based on the repeated clustering of a data set followed by computing the final cluster assignment based on a consensus function. The analyst can thus choose the number of times clustering is performed as well as the minimum and the maximum number of clusters created during each of the individual clusterings. The consensus function is based on computing the Hamming distance between the assignments of data points to clusters. The analyst can choose, what the maximum hamming distance for joining two clusters is. Finally, the consensus clustering results can be filtered according to two criteria. First of all, the maximum number of clusters generated by the consensus clustering can be specified. Moreover, selecting the minimum number of elements per cluster allows to remove outliers that are assigned to different clusters each time clustering is performed compared to the other members of the cluster.

Table 1: The different distance measures avaiable for clustering.

Name	$d(\vec{b_1},\vec{b_2}) =$
Angular distance	$cos^{-1}\left(\frac{\vec{b_1}\circ\vec{b_2}}{\lvert\vec{b_1}\rvert\cdot\lvert\vec{b_2}\rvert}\right)$
Euclidean distance	$\sqrt{\sum_{i=0}^{n}\left(\vec{b_{1,i}}-\vec{b_{2,i}}\right)^2}$
Manhattan distance	$\sum_{i=0}^{n}\left\lvert\vec{b_{1,i}}-\vec{b_{2,i}}\right\rvert$
LP-Norm	$\sqrt[p]{\sum_{i=0}^{n}\left\lvert\vec{b_{1,i}}-\vec{b_{2,i}}\right\rvert^p}$
Mahalanobis distance	$\sqrt{\left(\vec{b_1}-\vec{b_2}\right)^T S^{-1}\left(\vec{b_1}-\vec{b_2}\right)}$
Bin size difference	$\left\lvert\sum_{i=0}^{n}\vec{b_{1,i}}-\sum_{i=0}^{n}\vec{b_{2,i}}\right\rvert$

After selecting the clustering strategy and setting all parameters to the desired values, the analyst starts the clustering of the filtered bins by pressing the "Cluster Data" button. Upon termination, the background changes its color to green. Should any of the parameters be changed after clustering, its respective background color is set to red. At the same time, the background of the button for starting the clustering and the background of the message associated to the button are changed to red, too. The clustering parameters can be changed and applied any time.

The result of the clustering step is a set of clusters representing those bins whose tiles have a similar distribution with respect to the number of segments they contain. Thus, bins are not necessarily similar if they are close to each other with respect to their attribute values. However, if they are, this implies that bins that are close together share a similar distribution. The biological interpretation of clustering results and the biological insights gained are presented in Section 7.

6.4 Mapping the Tiled Binned Clusters to 3D Tiled Binned Scatter Plots

The visualization is based on the tiled binned 3D scatter plots introduced by Zeckzer et al. [27]. However, the mapping of information to visual elements is different. We redesignate the meaning of the spheres in the original 3d scatter plot and assign each cluster to one

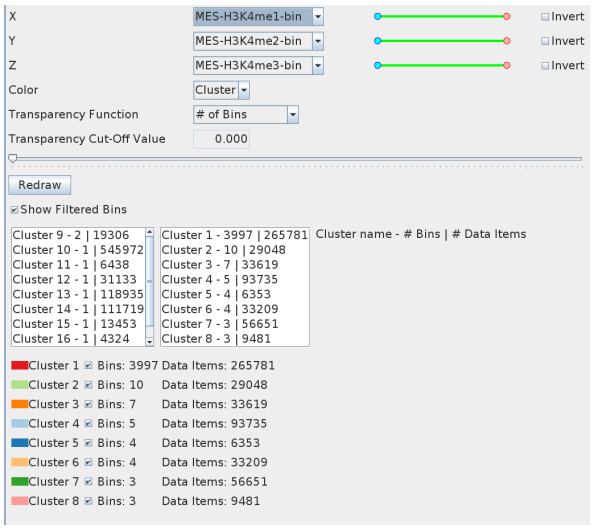


Figure 5: Mapping panel. Top: The analyst can choose which binning dimension is mapped onto *x*-, *y*-, and *z*-axis, respectively. Cutting planes and inverted cutting planes can be used to restrict the information shown and such eases the analysis of the data focused upon. Either the number of data items or the number of bins per cluster is mapped onto the transparency of each sphere. Bottom: The analyst selects up to eight clusters for the current investigation. Information about the selected clusters is displayed and each selected cluster can be disabled. Each cluster for each bin is represented by a sphere. Each cluster number is mapped onto color and position of the sphere.

sphere in our adopted visualization. Thus, eight clusters can be analyzed in parallel as shown in Figure 6. Those can be selected from the left list box in the lower part of the 'Mapping' tab (Figure 5) by clicking on an entry. Those clusters selected for display are shown in the right list box, which additionally can be used for deselecting clusters by clicking on the respective entry. Each cluster is assigned to a tile and thus will be represented by a sphere having the position corresponding to that tile in each bin, as well as the color corresponding to that tile. Cluster color, cluster number, and number of bins as well as number of data items per cluster are listed below those two list boxes. Moreover, a check box allows selecting and deselecting each cluster for display. If the analyst wishes, the filtered bins can be included into the display by marking the check box above the lists. Each filtered bin is represented by a *gray* sphere centered inside the area of its bin.

The number of bins per cluster or the number of data items per cluster can be mapped to the transparency of the spheres. Additionally, a transparency cut-off value can be selected showing only spheres having a higher transparency than the cut-off value selected.

The dimensions that are mapped onto the *x*-, *y*-, and *z*-axes are selected by the analyst using the respective drop-down boxes at the top of the interface. Moreover, cutting planes and inverted cutting planes can be activated by the sliders and the checkboxes to the right of the axes selection, respectively.

6.5 Tile Table, Segment Table, and Centroid Table

The 'Tile Table' tab contains an entry for each tile shown, i.e., for each sphere in the 3D scatter plot. Besides the cluster number, the number of segments of the cluster having a specific modification or a specific code are shown. The code was created by the segmentation process, whereas the modifications were chosen by the analyst while selecting the modifications used for binning (Section 6.1).

Selecting a sphere or selecting a row in the tile table results in showing all segments of that tile in the 'Segment Table' tab (linked views). Here, for each segment its long ID, its short ID as well as modification coverage or other information from the additional data selected during segmentation are shown. Moreover, CG densities computed during segmentation are displayed. Finally, the length of each segment is provided.

Selecting a row in the tile table highlights the respective sphere in the 3D scatter plot. Selecting a sphere in the 3D scatter plot leads to showing only the related tile entries in the tile table. Thus, tile table and 3D scatter plot are linked, too.

The 'Centroid Table' tab provides information about the centroids of each cluster. Hereby, each row represents one modification selected for binning, whereas each column represents a cluster. The centroid of each cluster is then given by the entries in the cells connecting the cluster to the respective modifications.

6.6 Visual Control Tab

It is possible to change the *point of view* to any angle. For exploring the visualization with any degree of detail, it is possible to *zoom* in and out. Thus, it is possible to navigate in any direction required to obtain the best views on the data. Additionally, an *auto-rotation* of the plot eases getting an overview of the data set facilitating the choice of suitable viewpoints. Any perspective can be *saved* and *loaded*, which supports comparing different data sets using the same perspective without effort. Most importantly, snapshots of the visualization can be saved to file.

7 BIOLOGICAL INSIGHTS

We demonstrate the functionality and benefits of TiBiCluster using a data set with 2 cell types and 6 histone modifications per cell type. It is compiled as described in Steiner et al. [23] but extends the code calculation to 6 dimensions yielding $2^6 = 64$ codes ranging from

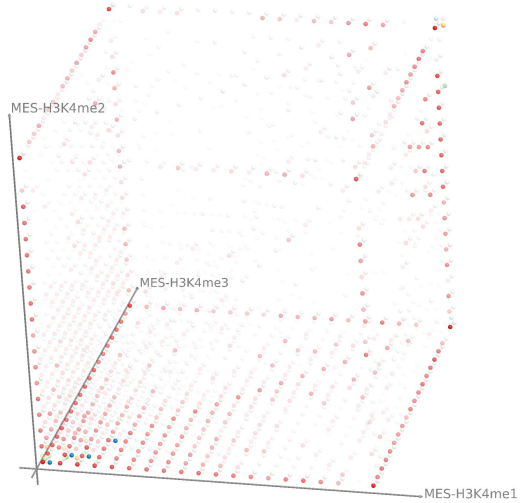
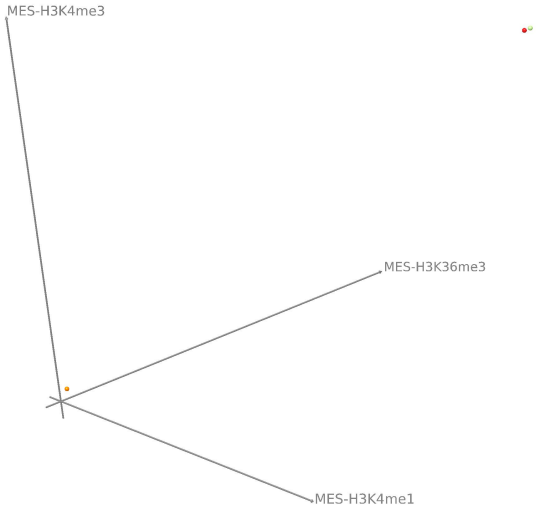


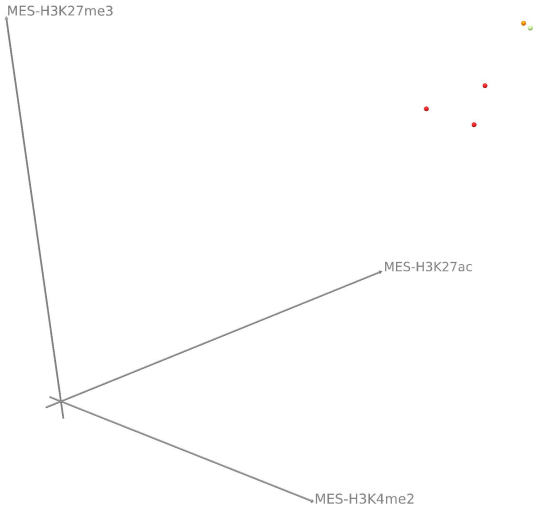
Figure 6: Results obtained after binning (Figure 2), filtering (Figure 3), and clustering (Figure 4). The parameter settings for the visual mapping are shown in Figure 5.

0 to 63. The reference cell type is the mesendodermal cell line. Data from a mesenchymal stem cell line is compared to it. Mesendodermal cells arise early in the embryogenesis, i.e., the development of an embryo out of a fertilized egg. They differentiate into mesodermal and endodermal cells, and finally into various organs. Mesenchymal stem cells differentiate out of successors of the mesendodermal cells. However, several differentiation steps lie in-between them.

The mesenchymal stem cell data is clustered using TiBi-Cluster. We choose the six modifications as the six dimensions for further analysis (Figure 2). Each dimension is binned with 20 bins. Most bins contain only a few elements, while only a few contain many (Figure 3). Bins with less than 5 segments are filtered out to focus more on coordinated changes. The data is clustered using consensus clustering (Figure 4). Ten instances of k-means clustering with euclidean distance and 8 clusters each is performed. Bins with Hamming distance 0 between the 10 cluster assignments obtained are summarized to one cluster. No clusters are filtered. The clustering results in 17 clusters. The parameters of the original mapping and selected clusters (Figure 5) yield a visualization showing the first eight clusters (Figure 6). Cluster 1 (red spheres) contains bins distributed over the whole hyper cube, i.e., segments with any epigenetic state in mesenchymal cells. All these segments are unmodified in mesendodermal cells and are located in bins containing only a few modification. These segments are thus genomic regions which become newly modified in mesenchymal cells but carry an unusual epigenetic state. Only few other clusters exist near the H3K4me1 – H3K4me3 plane (bottom: blue, green, and yellow spheres) and near



(a) Dimensions: H3K4me1, H3K4me3, H3K36me3



(b) Dimensions: H3K27ac, H3K27me3, and H3K4me2

Figure 7: 3D tiled bin scatter plots for cluster 8 (red spheres), cluster 14 (light green spheres), and cluster 17 (orange spheres).

the H3K4me1 – H3K4me3 – H3K4me2 corner (upper right: desaturated spheres).

Changing the selection of clusters displayed, the analyst finds clusters 8, 12, 13, 14, and 17 particularly interesting.

Two clusters, cluster 8 and cluster 14 are genes marked with all modifications in both cell types (see Figure 7). In total, about 120,000 segments belong to the two clusters. Please note, that this state, i.e., all marks studied, is theoretically possible but very unlikely. Mono-, di-, and trimethylation of H3K4 may only be observed in a single cell at the same position if distributed to the four copies of H3K4 in the histone complexes attached to the two alleles. Acetylation and trimethylation of H3K27 have an contrary effect, namely, formation of open and

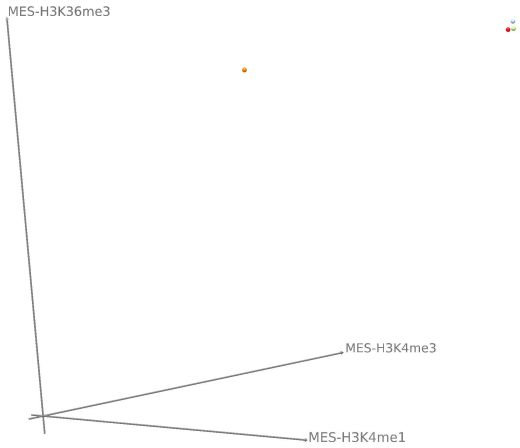


Figure 8: 3D tiled bin scatter plot for cluster 4 (red spheres), cluster 6 (light green spheres), cluster 12 (orange spheres), and cluster 13 (light blue spheres). Dimensions: H3K4me1, H3K4me3, H3K36me3

closed chromatin. They therefore do not occur at the same histone complex but only at the two different histone complexes attached to the two alleles. More likely, this is a mixed state in the population of cells. This indicates that the epigenetic state of these genomic regions is either not important for the cells identity and function or undergoes changes due to ongoing differentiation.

Segments in cluster 17 (almost 15,000) are in a mixed, undetermined state in mesendodermal cells, i.e., all or almost all marks. However, in mesenchymal stem cells, only three marks are present: H3K4me2, H3K27me3, and H3K27ac (see Figure 7) Again, it is unlikely that methylation and acetylation occur at the same histone complex due to their antagonistic effects. The combination of H3K4me2 and H3K27ac was shown to mark transcription factor binding sites and to recruit cell type specific transcription factors to them [24]. Since the third mark, H3K27me3, likely does not occur at the same allele or cell, the observed combination may indicate either an epigenome-driven recruitment of transcription factors in some cells or repression of the same loci in other cells. Alternatively, the recruitment is specific for one allele while the other allele is silenced by H3K27me3. Since ChIP-seq does not allow single cell measurements nor simultaneous measurement of combination of marks, we cannot distinguish between those alternatives.

In several clusters, we observe the combination of H3K4me1 and H3K36me3 (see Figure 8). These are cluster 12, cluster 13, as well as cluster 4 in combination with H3K4me3 and cluster 6 in combination with H3K4me3 and H3K27me3. Based on the current knowledge, this combination is unexpected. H3K4me1 is supposed to localize to promoters and represses transcription. H3K36me3 is associated with active splicing and thus with transcription and localized to splice sites. Even though one would expect those marks at different genomic loci and not in combination, we observe a strong co-occurrence of both marks.
We conclude therefore that their function is more diverse than known so far.

8 CONCLUSION

We introduced TiBi-Cluster, a methodology that combines tiling and binning with clustering. The results are then shown in tiled binned 3D scatter plots introduced before and adapted to the current methodology. Altogether, the methodology and the tool supporting the methodology allow analyzing the fate of chromatin modifications during cell differentiation. Our method proved to be vastly beneficial while analyzing and comparing six histone modifications in two cell lines. New, unknown relations were uncovered using TiBi-Cluster.

9 ACKNOWLEDGMENTS

We thank all the unknown reviewers of previous versions of our paper for their valuable comments. This work was partially funded by the German Federal Ministry of Education and Research (BMBF) within the project Competence Center for Scalable Data Services and Solutions (ScaDS) Dresden/Leipzig (BMBF grant 01IS14014B).

10 REFERENCES

[1] A set of tools (in Java) for working with next generation sequencing data in the BAM (`http://samtools.sourceforge.net`) format. `http://broadinstitute.github.io/picard/`.

[2] D. Arthur and S. Vassilvitskii. k-means++: The Advantages of Careful Seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pp. 1027–1035. Society for Industrial and Applied Mathematics Philadelphia, PA, USA, 2007.

[3] J. C. Bezdek and N. R. Pal. Some new indexes of cluster validity. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 28(3):301–315, Jun 1998. doi: 10.1109/3477.678624

[4] J. Cheng, R. Blum, C. Bowman, D. Hu, A. Shilatifard, S. Shen, and B. Dynlacht. A Role for H3K4 Monomethylation in Gene Repression and Partitioning of Chromatin Readers. *Molecular Cell*, 53(6):979–992, 2014. doi: 10.1016/j.molcel.2014.02.032

[5] P. Cheung and P. Lau. Epigenetic regulation by histone methylation and histone variants. *Mol Endocrinol*, 19(3):563–73, Mar 2005.

[6] D. L. Davies and D. W. Bouldin. A Cluster Separation Measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-1(2):224–227, April 1979. doi: 10.1109/TPAMI.1979.4766909

[7] A. L. N. Fred. *Finding Consistent Clusters in Data Partitions*, pp. 309–318. Springer Berlin Heidelberg, Berlin, Heidelberg, 2001. doi: 10.1007/3-540-48219-9_31

[8] A. L. N. Fred and A. K. Jain. Data clustering using evidence accumulation. In *Object recognition supported by user interaction for service robots*, vol. 4, pp. 276–280, 2002. doi: 10.1109/ICPR.2002.1047450

[9] D. Gerighausen, D. Zeckzer, L. Müller, and S. J. Prohaska. ChromatinVis: a tool for analyzing epigenetic data, 2014. Poster presented at 2nd EMBO Conf. on Visualizing Biological Data, Heidelberg, Germany.

[10] G. Grinstein, M. Trutschl, and U. Cvek. High-Dimensional Visualizations. In *Proceedings of the VII Data Mining Conference KDD Workshop 2001*, pp. 7–19. ACM Press, New York, San Francisco-CA, USA, 2001.

[11] S. Hoffmann, C. Otto, S. Kurtz, C. M. Sharma, P. Khaitovich, J. Vogel, P. F. Stadler, and J. Hackermüller. Fast mapping of short sequences with mismatches, insertions and deletions using index structures. *PLoS Computational Biology*, 5(9):e1000502, Sep 2009. doi: 10.1371/journal.pcbi.1000502

[12] L. Hubert and P. Arabie. Comparing partitions. *Journal of Classification*, 2(1):193–218, 1985. doi: 10.1007/BF01908075

[13] A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data*. Prentice-Hall, 1988.

[14] W. J. Kent, C. W. Sugnet, T. S. Furey, K. M. Roskin, T. H. Pringle, A. M. Zahler, and D. Haussler. The Human Genome Browser at UCSC. *Genome Research*, 12(6):996–1006, 2002.

[15] Z. Kuang, L. Cai, X. Zhang, H. Ji, B. P. Tu, and J. D. Boeke. High-temporal-resolution view of transcription and chromatin states across distinct metabolic states in budding yeast. *Nature structural & molecular biology*, 21(10):854–863, 2014.

[16] S. Liu, D. Maljovec, B. Wang, P.-T. Bremer, and V. Pascucci. Visualizing high-dimensional data: Advances in the past decade. In *Proc. Eurographics Conf. Visualization*, pp. 20151115–127, 2015.

[17] J. MacQueen. Some Methods for classification and Analysis of Multivariate Observations. In *Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability*, pp. 281–297. University of California Press, 1967.

[18] L. Müller, D. Gerighausen, M. Farman, and D. Zeckzer. Sierra Platinum: A Fast and Robust Multiple-Replicate Peak Caller With Visual Quality-Control and -Steering. *BMC Bioinformatics*, 17(1):1–13, 2016. doi: 10.1186/s12859-016-1248-6

[19] T. Munzner. *Visualization Analysis and Design: Principles, Techniques, and Practice*. A K Peters Visualization Series, 2014.

[20] Roadmap Epigenomics Consortium et al. Integrative analysis of 111 reference human epigenomes. *Nature*, 518(7539):317–30, Feb 2015. doi: 10.1038/nature14248

[21] P. Rousseeuw. Silhouettes: A Graphical Aid to the Interpretation and Validation of Cluster Analysis. *J. Comput. Appl. Math.*, 20(1):53–65, nov 1987. doi: 10.1016/0377-0427(87)90125-7

[22] M. Sedlmair, T. Munzner, and M. Tory. Empirical guidance on scatterplot and dimension reduction technique choices. *IEEE transactions on visualization and computer graphics*, 19(12):2634–2643, 2013.

[23] L. Steiner, L. Hopp, H. Wirth, J. Galle, H. Binder, S. J. Prohaska, and T. Rohlf. A Global Genome Segmentation Method for Exploration of Epigenetic Patterns. *PLOS ONE*, 7(10):e46811, 2012.

[24] Y. Wang, X. Li, and H. Hu. H3K4me2 reliably defines transcription factor binding regions in different cells. *Genomics*, 103(2-3):222–228, 2014. doi: 10.1016/j.ygeno.2014.02.002

[25] M. Ward, G. Grinstein, and D. Keim. *Interactive Data Visualization: Foundations, Techniques, and Applications*. A K Peters, 2010.

[26] C. Ware. *Information Visualization: Perception for Design*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2 ed., 2004.

[27] D. Zeckzer, D. Gerighausen, and L. Müller. Analyzing Histone Modifications in iPS Cells Using Tiled Binned 3D Scatter Plots. In *2016 Big Data Visual Analytics (BDVA)*, pp. 1–8, Nov 2016. doi: 10.1109/BDVA.2016.7787042

[28] D. Zeckzer, D. Gerighausen, L. Steiner, and S. J. Prohaska. Analyzing Chromatin Using Tiled Binned Scatterplot Matrices. *2014 IEEE Symposium on Biological Data Visualization (BioVis)*, abs/1407.2084, 2014.

[29] X. Zhou, R. F. Lowdon, D. Li, H. A. Lawson, P. A. Madden, J. F. Costello, and T. Wang. Exploring long-range genome interactions using the WashU Epigenome Browser. *Nature methods*, 10(5):375–376, 2013.

ISSN 1213-6972 Journal of WSCG http://www.wscg.eu/ Vol.26, 2018, No.1

Fast Indirect Lighting Approximations using the Representative Candidate Line Space

Kevin Keul Tilman Koß Stefan Müller
Department of Computer Graphics,
Institute for Computational Visualistics,
University of Koblenz-Landau,
Koblenz, Germany
{keul | tkoss | stefanm}@uni-koblenz.de

ABSTRACT

We propose a novel approach for using directional Line Space information in calculation of indirect illumination. Typically, the Line Space is build on top of regular recursive grids and contains visibility information which is used to perform an efficient empty space skipping during traversal. In our method we extend the stored information by precomputed representative candidates, which are based on the Line Space shafts and serve as an approximation of the actual scene geometry. By using these candidates it is not necessary to compute any intersection tests and therefore the traversal is accelerated. However, the candidate approximation leads to visible artifacts. We therefore present a technique that significantly reduces these artifacts by extrapolation of the actual surface and demonstrate that the artifacts are nearly not perceivable in the application of indirect illumination. Moreover we adapt the Line Space to other data structures like bounding volume hierarchies (BVHs) which further increases the performance in ray tracing. Compared to the pure data structures we achieve significantly better performance with nearly no drawback in quality of indirect lighting.

Keywords

Visualization, Computer Graphics, Ray Tracing, Data Structures, Visibility Algorithms

1 INTRODUCTION

Calculation of global illumination and indirect lighting is a non-trivial task which significantly improves realism of generated images and renders the possibility for photo realistic effects. The two main ways for computation of global illumination are depth-based rasterization techniques and ray tracing. The former is typically used in interactive and real-time rendering, due to the high performance that is achieved. The basic idea is to determine the visible scene primitives through projection to the screen in object order. Adding complex rendering effects like global illumination without ray tracing is a non-trivial task and suffers from different quality problems [Rit12]. In ray tracing the visible surfaces are calculated per screen pixel by computing intersections between rays and the scene primitives. By using additional rays per pixel it is possible to calculate complex rendering effects and indirect lighting. However, be-

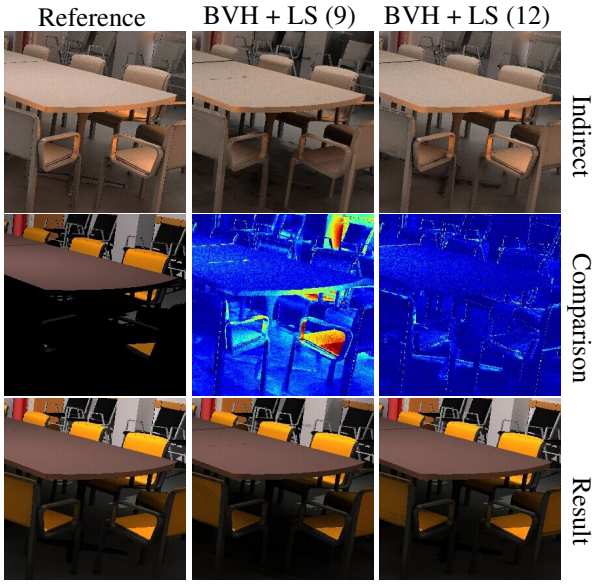


Figure 1: Example of our technique. The left column shows correct results as reference, the other columns show the utilization of precomputed scene primitives in the Line Space using a low and a high depth parameter. In the top row indirect illumination is presented. The middle row shows a comparison to ground truth, where the left image presents only direct illumination and shadows. The last row consists of the final images.

https://doi.org/10.24132/JWSCG.2018.26.1.2

cause of the huge number of intersection calculations, this process is quite slow and therefore a well defined acceleration data structure is needed.

Most data structures used for this purpose work in a spatial manner by grouping scene primitives within bounding volumes and thereby limiting the needed intersection tests to a minimum. During ray traversal the bounding volumes are tested for intersection first and only those scene primitives that are contained by intersected bounding volumes are tested for intersection with the ray. Most of the primitives are excluded within a short time. On a basic level spatial data structures are distinguished by the size and arrangement of these volumes [Hav00]. A common similarity of most of the used data structures is that axis aligned bounding boxes are used as bounding volumes because of their simplicity. Still, a lot of intersection tests need to be calculated.

A further approach is to precompute visibility in a data structure and therefore eliminating the need of intersection tests. More recently this technique received renewed interest and was used to accelerate the traversal of shadow rays and intersection finding. Recursive grids were extended by directional information of the Line Space, which uses ray clustering into predefined shafts. Binary visibility information based on the shafts was computed and during runtime applied to the contained rays. This allowed a direct access of visibility information instead of complex intersection tests.

While in previous work only binary visibility information was used, we further extend the Line Space by precomputing a representative candidate (i.e. a triangle) for each non-empty shaft. This leads to significantly faster but approximated results, which can be used for the acceleration of indirect lighting computations. While the results suffer from approximation artifacts, it was shown in [Yu09] that indirect illumination does not require correct results and therefore the artifacts can be disregarded in this context, as shown in Figure 1. Moreover we use a general Line Space description on basis of bounding boxes. With this our approach can be applied to almost every spatial data structure used as base structure. We demonstrate this applicability with the NTree, a regular recursive grid structure, as used in previous work, and BVHs and therefore show the general utility in terms of accelerating performance. The main contributions of our paper are:

- An approach for precomputation of possible intersection candidates based on the simplification of clustering rays into shafts in the Line Space with the application of indirect lighting calculation without the need of intersection tests.
- A generalization of the Line Space to bounding boxes and therefore the adaption to almost all commonly used spatial data structures.
- An evaluation in terms of performance, memory consumption, initialization speed and quality of the base data structure in combination with the Line Space and the comparison to the pure data structure.

2 RELATED WORK

Rendering of indirect lighting and global illumination is a well studied and complex topic. Therefore we refer to [Wal03] [Pha16] and [Rit12] for a broad overview of techniques and possibilities in ray tracing and rasterization based techniques. We focus specifically on the acceleration of approximated intersection point computation through the usage of sophisticated data structures.

Spatial data structures.

Nowadays most acceleration data structures for ray tracing work with a hierarchical spatial subdivision of the scene space [Hav00]. All geometrical objects of the scene are arranged depending on their spatial localization and clustered in separate bounding volumes of the data structure. During rendering the rays are traversed along those bounding volumes and only the scene objects within passed volumes are used for intersection tests. In this kind recursive grids [Jev89] are the combination of a grid like subdivision of volumes in a recursive hierarchical structure. It was shown, that ray traversal greatly benefits from those data structures. The bounding volume hierarchy (BVH) works by recursively grouping of adjacent scene primitives within axis aligned bounding boxes [Ail13]. It is currently the most used data structure, mainly because of the good performance even in dynamic scenes, the small memory footprint and fast build time in comparison to other data structures [Vin16]. It is important for BVHs to construct a high quality tree and many different initialization algorithms were proposed for this purpose. The surface area heuristic (SAH) used in combination with spatial splits is an example for a good tree construction [Sti09]. Moreover there exist build algorithms that work on already constructed BVHs, which are then optimized to gain better quality [Kar13]. The bonsai algorithm uses a two-level construction with optimization in so called mini trees resulting in high performance and good build times [Gan15]. By using agglomerative clustering and multi-threaded CPU approximations a good trade-off between quality and construction time can be found [Gu13]. By optimizing spatial splitting during construction this trade-off is further improved [Wod17] [Mei17]. While all previously mentioned approaches result in good tree quality, their construction takes quite a long time and dynamic scenes are not covered. Using linear sorting through morton codes leads to the linear BVH (LBVH) with fast build times due to parallelization on the GPU, however with inferior tree quality [Lau09]. An advancement to this is

the hierarchical LBVH (HLBVH), which is especially used in full dynamic scenes because of the fast creation [Pan10] [Gar11]. Further optimization of fast BVH construction can be made by refitting of splitting planes in dynamic scenes [Yin14]. Usually the traversal of a BVH works in a stack-based manner [Ail12]. More recently combinations of multiple data structures are explored [Wan16].

Directional data structures.

Using directional information instead of or in addition to spatial information provides the possibility for visibility precomputation. Most attempts aim to generalize rays on a higher level and then precompute information on an intermediate representation. Examples for this are the generalization of rays to cones [Ama84], beams [Hec84] [Res05] [Lai09] or more generally to higher dimensional generalizations [Arv87]. In the latter, rays are classified by their three dimensional origin and their two dimensional direction and for each resulting five dimensional generalization a sorted list of intersecting objects is stored. This was optimized by reducing the generalization to four dimensions [Kwo98]. The four dimensional visibility field could be projected onto a bounding sphere which was then used to speed up ambient occlusion and stochastic ray tracing calculations [Mor07] [Gai10]. In a similar manner visibility information can be projected on planes, leading to intersection fields which were used for fast computation of global illumination [Ren05]. The concept of generalizing rays to shafts was introduced, where each shaft is the volume that is constructed by connecting two patches and forming their convex hull [Hai94] [Dre97]. There, a candidate list per shaft is created and later on used for all rays that pass a given shaft, which was applied to ray tracing and radiosity calculations. Recently, this approach was combined with a spatial recursive grid structure in terms of empty space skipping [Keu16] and shadow calculation [Bil16] [Keu17]. In this context, shafts have binary visibility information and are created between all patches of the regular subdivided boundary of each branching node in the tree hierarchy resulting in the Line Space.

In our approach the Line Space can be adapted to all spatial data structures that use bounding boxes of any kind. In addition to the binary visibility information of previous approaches, we store actual geometry information in the Line Space.

3 LINE SPACE WITH REPRESENTATIVE CANDIDATE DATA

The Line Space, as proposed by previous work [Keu16], is a data structure providing directional information for a given bounding box. The six faces of the box are equally divided into N^2 rectangular patches. Pairs of those patches that are arranged on different box faces are defined as *shafts*. The volume of a shaft is the convex set of all line segments connecting any point in the start patch with any point in the end patch. The Line Space stores arbitrary data for each shaft. A Line Space where a substitution of the start and end patches leads to the same result as the original one is called *symmetric*. There are $30N^4$ shafts in a non-symmetric Line Space and $15N^4$ shafts in a symmetric Line Space, therefore potentially resulting in a big memory consumption, as shown by previous work.

3.1 Representative Candidate per shaft

Until now, the Line Space was only used to store binary visibility information, i.e. whether a given shaft contains any geometry at all. This approach was utilized for empty space skipping [Keu16] and accelerated shadow computation [Bil16] [Keu17]. We extend it by storing a representative triangle for each shaft, which serves as an approximation for the geometry inside of the shaft. The stored information can be used during the traversal step of ray tracing to get a possible intersection between a given ray and the scene geometry. Instead of testing all candidate triangles of the given bounding box for intersections, only the previously stored representative triangle is considered. The intersection between the ray and the bounding box geometry is approximated as shown in algorithm 1.

Algorithm 1 The accelerated intersection algorithm between a ray and a Line Space bounding box.

```
(t_start, t_end) ← points where ray intersects box
i ← CALCPATCH(t_start)                    ▷ start patch
j ← CALCPATCH(t_end)                      ▷ end patch
shaftID ← CALCSHAFT(i, j)
triangle ← GETCANDIDATETRIANGLE(shaftID)
if triangle exists then
    return ray triangle intersection
return 0
```

The representative candidate Line Space is non-symmetric. The candidate used as the shaft representative is the triangle that optimally approximates the object surface within the shaft. To find it we search for an intersection between the geometry and the ray defined by the centroids of the shaft's start and end patches. This is illustrated in Figure 2. If no intersection is found, the shaft is marked as empty. The percentage of empty Line Space shafts is typically between $30\% - 70\%$ for manifold meshes and therefore a lot of memory can be saved with an appropriate memory layout, which is explained later on.

The surface inside a shaft can be classified into three categories:

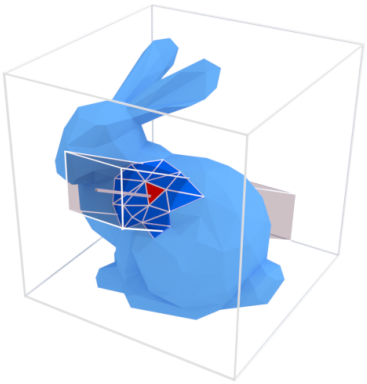


Figure 2: The representative candidate triangle (shown in red) is the triangle that is used to approximate the geometry in one shaft. This triangle is found by computing an intersection between the geometry and the centroid ray of the shaft.

1. The surface is closed and covers the whole shaft width.

2. The shaft lies at the boundary of a surface or contains multiple disconnected surfaces.

3. The shaft is empty.

Since we only store the single triangle per shaft, it is not inherently possible to distinguish the first two cases. The candidate triangle does not necessarily cover the whole shaft surface, as it is shown in Figure 2. To compensate this, the triangle is treated as infinite plane defined by its vertices. This approximation is used when searching for an intersection between a ray and the geometry contained in the shaft. Per-vertex normals can be interpolated by using the intersection parameters of the constructed plane. If the intersection point is outside of the triangle, it is computed by the extrapolation, therefore providing smooth normals for the whole shaft width. This is a rather big approximation, especially for highly curved surfaces, however it lowers discontinuity artifacts. To reduce artifacts for shafts that are not fully covered by a surface, edges can be found by calculating the angle between the extrapolated normal and the mean normal of the triangle. If the angle is bigger than a given threshold then the intersection is discarded.

The representative candidate Line Space stores a reference to a triangle for each shaft. In our case, this reference is a 32-bit index pointing to a buffer containing all triangles of the scene geometry. Depending on the storage layout of the scene geometry, more space efficient data types are possible. Since the Line Space stores data for every combination of start and end patch, it contains $M = 30N^4$ elements. While most of these shafts are empty and do not point to a valid triangle, we are able to only store the shaft information of filled shafts. This

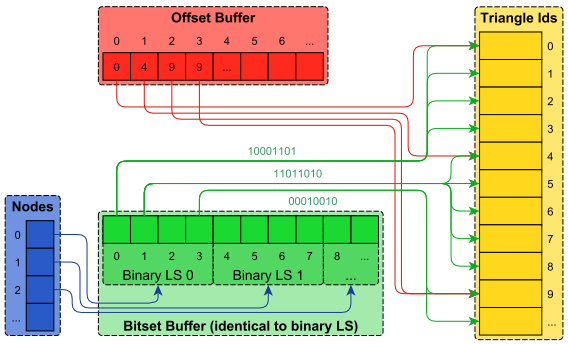


Figure 3: The sparse memory layout of our structure. Nodes have references to their bitsets (shown in green), which signal whether associated shafts are filled or empty. To access the triangle data, an additional offset per bitset is needed (shown in red).

is done by using a simple sparse scheme based on a bitset and offset buffer to skip empty shaft entries and only store filled shafts consecutively in memory. In addition we need to store one bit of information for each shaft, signaling whether the shaft is empty or filled. These bits are grouped in bitsets and are efficiently represented by 32-bit words. Moreover an offset for every bitset is stored, which specifies where the corresponding filled shaft entries withing the shaft buffer are located. This is shown in Figure 3. The memory overhead of this scheme therefore sums up to $\frac{M}{16}$ 32-bit words. Finally, the sparse storage is more memory efficient compared to dense storage if less than $\frac{15}{16} \approx 93\%$ of the shafts are filled, which is always the case. The offset computation is done by a parallel prefix sum, efficiently calculated on the GPU. It should be noted that the bitsets are identical with the binary Line Space, and therefore they are implicitly calculated. The data access with the sparse memory layout has a constant time complexity and is presented in algorithm 2. We use 32-bit words for all bitset and offset operations due to the better interaction with GPU computations. However, this can be generalized for arbitrary sizes like 64-bit words.

Algorithm 2 The algorithm presents the access of the shaft data in the sparse memory scheme.

```
procedure GETSPARSEDATA(ShaftIndex n)
    bitsetID ← ⌊n/32⌋
    bitset ← GETBITSET(bitsetID)
    bit ← n mod 32
    if (bitset & (1 ≪ bit)) ≠ 0 then
        offset ← GETOFFSET(bitsetID)
        id ← BITCOUNT(bitset ≪ (31 - bit)) - 1)
        return GETDATA(offset + id)
    else
        return 0
```

3.2 General Line Space for spatial datastructures

Because of the construction on top of bounding boxes, the Line Space can be adapted and integrated into each spatial data structure that consists of bounding boxes in any way. It provides directional information in addition to the spatial subdivision and therefore is able to minimize the traversal cost. Typical data structuces that can be used for this are Octrees, BVHs, k-d trees, uniform grids or recursive grids (such as the NTree in previous work).

We generate a representative candidate Line Space for selected tree nodes of the data structure to approximate the scene geometry. While the produced errors are too striking for direct illumination of primary rays, they are less perceivable when used for indirect illumination, which is in accordance to [Yu09], stating that accurate calculations are not required for global and indirect illumination. Therefore we use the representative candidate as approximation instead of the correct triangle data to calculate the intersection points in indirect illumination. In terms of the underlying base data structure it is necessary to consider which nodes in the tree need to store the Line Space information.

Adaptation to NTree

The data structure previously used for Line Space computations is the *NTree*, which is a regular recursive grid that repeatedly subdivides the scene and the already produced subdivisions into N^3 equally sized bounding boxes. In our case, we constrain the size of theses boxes to be cubes. This simplifies some computations when building and traversing the NTree while not having any detrimental effect for the data structure. The NTree provides increased traversal performance in comparison to a regular Octree or single layer uniform grid. This is because the tree width of an NTree with $N > 2$ can be larger than for Octrees, effectively lowering the tree depth. Since the bounding boxes of NTree nodes are equally sized, the NTree is a natural fit for Line Space computations, as shown in previous work. The Line Space patch size for a specific tree depth is equal for all nodes and correlates with the size of the node subdivisions.

To use the NTree with the representative candidate Line Space for indirect lighting approximations we first generate the NTree including the exact triangle data of the scene. This NTree can be used for all exact computations like primary or shadow rays. Moreover we utilize it for faster initialization of the representative candidates. We only compute the Line Space data on specific nodes in the NTree, which are determined by the depth D or a triangle count lower than T (i.e. $T = 8$). In our case the NTree and therefore also the representative candidate Line Space have a parameter values $N = 6$ or $N = 10$ and $D = 2$ for Line Space utilization, which is consistent with the results of previous work. The N value describes the branching factor of the NTree as well as the Line Space resolution. It was found to be accurate enough for the approximation while also granting sufficient performance. The depth parameter D also determines the performance, memory requirements and the approximation accuracy of the Line Space. Higher depth values lead to more Line Space nodes and therefore higher computation times and memory consumption. However, lower depth values decrease the accuracy of scene approximations but significantly increase the traversal performance. This is due to the fact that the number of nodes greatly increases with the depth of the underlying tree. The shown value of $D = 2$ for the usage of Line Spaces within the NTree is sufficient for quality, traversal speed and memory consumption.

When traversing indirect rays, these Line Space nodes are treated as leaf nodes in the NTree and are therefore able to terminate the traversal. Intersections with the scene are calculated by the procedure explained in subsection 3.1. The general traversal algorithm of the data structure does not need to be changed, only the handling of leaf nodes needs to be replaced by the appropriate Line Space calculations as shown in algorithm 1.

Adaptation to BVH

By using a BVH, the scene is recursively subdivided by axis aligned bounding boxes that tightly enclose the geometry. Besides the NTree, the BVH is also used as a base data structure for the Line Space in our work. The representative candidate in the Line Space nodes are used in the same way as described with the NTree. Due to the reason that every BVH node branches into only 2 subnodes, the tree depth is normally much higher than for the NTree. With this BVH nodes converge to the actual scene geometry and they are not constrained to be equal in size in every tree layer. Typically less nodes are needed in the BVH for scenes with a high amount of empty space. Again, the depth D and the triangle count T are used to determine whether a node is extended by a Line Space. Furthermore, it is possible to consider the box size as criterion for this determination, but this was not done in our work.

Nevertheless, the usage of a BVH with the representative candidate Line Space has two disadvantages. Since the bounding boxes are not cubical, the shaft patch size will slightly differ for each bounding box. The approximation artifacts in that case are not distributed in any predictive manner, and therefore have significant impact, depending on the used parameter set. This is visible in the results using low parameter sets for the BVH Line Space. Additionally, BVH nodes may overlap, especially in scenes where the triangle size is highly diverse. Therefore, multiple Line Space nodes and their shafts may overlap and approximate the same scene geometry using different representative candidates. These

effects are mostly visible in architectural scenes, as shown in the results. A spatial split BVH construction might reduce these effects significantly, however this was not used in our work.

4 RESULTS

For the evaluation we used two different base data structures: the NTree, a regular recursive grid as it was used in previous work on the binary Line Space [Keu16] [Bil16], and a state of the art BVH algorithm [Ail12], which is used as comparison in multiple related works. For the NTree we used a branching factor of 6 and 10 and a hierarchical depth of 3, as those are the proposed parameters by [Keu16]. We did not incorporate any further optimizations of the BVH tree quality, as recently proposed [Gan15] [Yin14]. The Line Space with precomputed representative shaft candidates is then used in a given hierarchical depth of the base data structure. Within the NTree this depth is set to the lowest branching nodes in the hierarchy, i.e. depth 2. The Line Space depth within the BVH is more versatile and can be set arbitrarily to achieve a good trade-off between performance and memory consumption as well as initialization time. The chosen depths and their impact are shown in the diagram and the visual results.

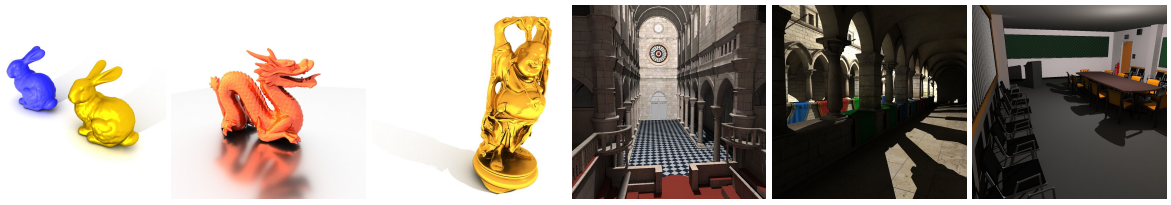
We measured the quality and the differences in performance, initialization time and memory consumption. For this purpose different widely used test scenes with special characteristics are evaluated. In principle they are dividable into two categories: scenes containing a single object (BUNNY, DRAGON and BUDDHA) and architectural scenes (SIBENIK, SPONZA and CONFERENCE), which are more suitable for usage in video games. Apart from this, the number of scene primitives varies significantly in the used scenes and ranges from ~70k triangles up to ~1 million triangles. The test results were produced on a GeForce GTX 1080, however the relative performance is the same on similar systems. All data structures are implemented in the same environment and supported by acceleration in agreement. Hence, a fair comparison of the used structures is guaranteed. The resolution of the renderings was in all cases 720p. Primary and shadow rays were rendered with fast rasterization accelerated by a binary Line Space techniques as proposed by [Bil16] and [Keu17] and are therefore out of our scope. Regarding this, our approach accelerates calculations of all indirect lighting effects, resulting for example in ambient occlusion, diffuse illumination and glossy reflections.

Table 1 shows the quantitative results using the two main data structures with and without the acceleration of the Line Space with the mentioned depth parameter. We evaluated the build time, the memory consumption and the performance in ray tracing for indirect rays. Obviously, the computation time and memory consumption of the Line Space need to be summed up on the values of the base data structure. The illustrated Line Space values in the table are already combined and therefore show the total sum. Moreover the build times and the memory consumption of the Line Space significantly scale with the total number of computed Line Spaces and not the number of scene triangles. For BVH initialization we use a binned SAH construction, resulting in good quality but non-interactive build times. The used build algorithm significantly affects the build time of the BVH, but as our work focuses on the relative comparison of the data structure with the usage of the Line Space, this does not affect our results.

The runtime performance was measured in frames per second only counting indirect illumination. Apart from absolute values, the relative differences between pure and Line Space supported data structures show the benefit of our approach. The BVH performance is near state-of-the-art in ray tracing performance. It is mainly regulated by the quality of the underlying tree and can be further optimized by recent techniques as proposed by [Gan15] and [Yin14]. Using our technique gives a significantly better performance, however with approximated results. This is due to the simplification of shaft data, where only a single candidate is stored and used for all rays passing a given shaft. Moreover it is noticeable that the usage of the Line Space accelerates the data structure to an acceptable level, even in those cases where the base data structure performs very poorly.

Figure 5 shows the qualitative differences of various depths used in Line Space accelerated BVH in comparison to ground truth data. It is observable that lower parameter sets result in visible artifacts due to shaft simplification. However, by using higher depths for the Line Space within the BVH hierarchy the artifacts become manageable. The artifacts are especially noticeable in bigger scenes when the camera is positioned close to an object. This is mostly observable in the architectural scenes as shown in figure 6 on the last page. There, also the NTree results with the Line Space are shown. As with the BVH, the quality of the Line Space accelerated NTree improves when the Line Space is used in a deeper level of the tree hierarchy. Because of its regular structure, the NTree Line Space is mostly better suited for big architectural scenes and produces less approximation artifacts for these cases in comparison to the BVH Line Space. However, as it was shown by [Yu09], correctness in indirect illumination is not required and approximations are sufficient in most cases. Following this the BVH Line Space has better performance with mostly sufficient quality.

The main factor for quantitative and qualitative analysis is the value of the used depth parameter where Line Spaces are created and used. A deeper depth results in more Line Spaces, therefore causing higher build time



BUNNY DRAGON BUDDHA SIBENIK SPONZA CONFERENCE

Figure 4: The evaluated test scenes. Renderings were done in 720p. Primary rays were calculated with rasterization, shadow rays were rendered with a binary Line Space and indirect rays were produced with our technique.

Scene		BVH					NT (6, 3)			NT (10, 3)		
		pure	LS (9)		LS (12)		pure	LS (2)		pure	LS (2)	
Bunny	**init**	0,3	2,2		10,4		3,7	9,6		17,2	44,3	
69k tris	**size**	5,5	38,7		227,2		1,5	17,6		23,9	149,7	
	perf	76,7	194,7	**2,5x**	131,0	**1,7x**	20,6	89,4	**4,3x**	36,7	63,0	**1,7x**
Dragon	**init**	1,7	5,6		17,1		22,5	50,3		100,6	209,7	
871k tris	**size**	35,5	74,8		280,0		4,1	10,8		13,2	65,5	
	perf	45,9	169,4	**3,7x**	110,2	**2,4x**	1,4	107,9	**77x**	11,7	62,1	**5,3x**
Buddha	**init**	2,0	6,2		17,6		27,6	61,6		123,2	254,2	
1087k tris	**size**	40,7	80,0		285,0		4,5	10,9		11,6	57,5	
	perf	62,4	195,4	**3,1x**	128,0	**2,1x**	1,1	121,4	**108x**	12,1	69,4	**5,7x**
Sibenik	**init**	0,1	1,9		7,4		2,3	18,6		13,0	102,3	
75k tris	**size**	3,0	57,7		251,6		8,3	286,4		106,9	2114,3	
	perf	19,3	38,3	**2x**	28,4	**1,5x**	8,8	23,1	**2,6x**	8,6	12,8	**1,5x**
Sponza	**init**	0,5	3,0		9,2		7,1	28,5		36,3	153,0	
262k tris	**size**	10,1	54,1		231,2		8,9	347,6		133,3	2680,4	
	perf	16,7	48,5	**2,9x**	33,2	**2x**	5,8	27,4	**4,7x**	10,1	16,2	**1,6x**
Conference	**init**	0,7	3,2		9,6		7,3	27,0		36,8	115,2	
331k tris	**size**	13,4	61,3		213,2		6,3	170,9		86,3	1014,9	
	perf	16,0	44,6	**2,8x**	33,6	**2,1x**	1,4	26,3	**19x**	8,4	20,7	**2,5x**

Table 1: Test results of our evaluation. We measured the initialization time in seconds, the memory consumption in MB and the ray tracing performance in FPS for BVH and NTree as base data structures without and with the usage of the Line Space with varying depth parameter. Line Space values are already combined with the base structure. BVH initialization was optimized in terms of ray tracing performance and not initialization speed. The relative differences in comparison to the base data structure without Line Space acceleration are marked.

and memory consumption, as well as lower ray tracing performance. This is due to the fact, that with a deeper Line Space depth more nodes in the hierarchy need to be traversed. However the quality gets better when more Line Spaces are used. With this the Line Space depth can be used as an arbitrary parameter for setting a trade-off between quality and performance. This is especially true, when the base data structure produces a deep tree hierarchy, as it is done with BVHs. The NTree naturally only has a shallow tree hierarchy, therefore is not that suitable for a dynamic trade-off.

5 CONCLUSION AND FUTURE WORK

We presented the non-binary Line Space with visibility precomputation of scene information, which stores a single candidate as a representative per shaft. With this work, we explored the general approach of precomputing directional information per Line Space shaft in application of indirect and global illumination. Through the representative candidate precomputation, the need for intersection tests during traversal could be eliminated as far as possible. Although this technique results in approximation artifacts if the depth parameter is not high enough, we were able to show that these errors are nearly non-perceivable in the context of indirect illumination. When compared to the base data structure, this technique results in higher memory size and build time but is in all cases able to significantly surpass the base structure in terms of performance.

Moreover, we showed a generalization of the Line Space to all spatial data structures based on bounding boxes. We demonstrated this with an adaptation to a state-of-the-art BVH, resulting in higher performance in comparison to the NTree, the typically used base data structure of previous work.

Future Work

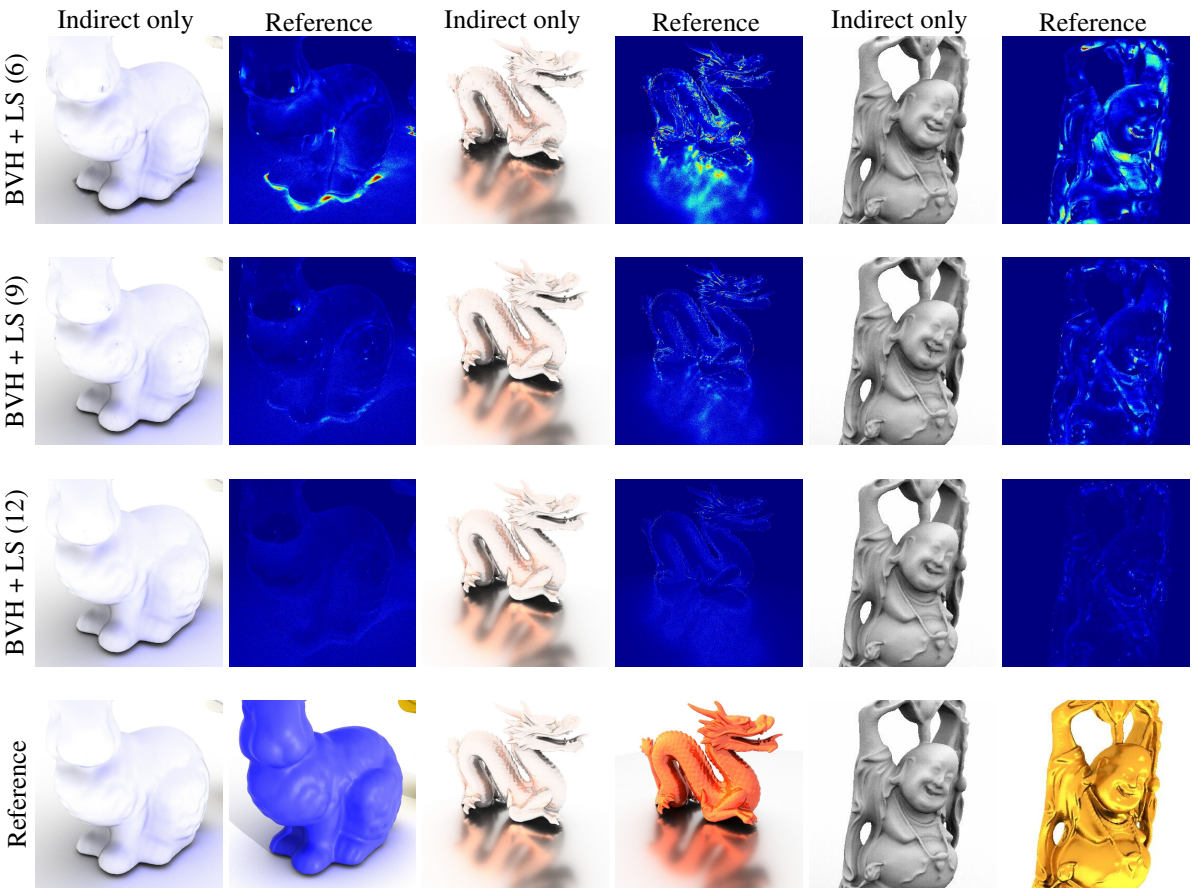


Figure 5: Some of the results of our evaluation. As illustrated, the buddha scene especially focuses on ambient occlusion, the bunny scene on diffuse materials and the dragon scene on glossy reflections. Nevertheless, all visual effects were indifferently produced with the same technique with the only difference in the object material. Therefore the results are not optimized to show specific effects. The maximum Line Space depth d within the BVH is shown in $LS(d)$.

The precomputation of scene information is only a beginning. By calculating and storing the lighting state in a shaft it may be possible to use a great variety of rendering techniques without further computation overhead during traversal. Although this leads to a significantly increase in memory consumption, the gain in tracing performance can be a big improvement in path tracing scenarios. Using a dynamic combination with the base data structure has the potential to accelerate most ray tracing systems, using the base structure in situations where correct information is needed and the Line Space where fast approximated data is sufficient.

An approach to further tackle the difficulties of memory size and initialization speed is to use the Line Space based on objects rather than the whole scene. With this each geometrical object has its own single Line Space. This grants the possibility for object instancing and a significant reduction of memory needed. Furthermore, Line Space information is more accurate, solving the teapot in a stadium problem. Object Line Spaces can be created beforehand and therefore significantly reducing initialization time. By combining the instancing aspect with local transformations creates the possibility to render dynamic scenes.

Another aspect that needs further investigation is the selection of the used parameter set. Currently, a fixed depth parameter is used for the whole Line Space tree, resulting in unnecessary high subdivision rate in sparsely filled areas. A dynamic subdivision scheme based on the number of candidates and the size of the current node would benefit the traversal and the Line Space accuracy.

6 REFERENCES

[Ail12] Aila, T., Laine, S., and Karras, T. Understanding the efficiency of ray traversal on gpus–kepler and fermi addendum. *NVIDIA Technical Report*, 2012.

[Ail13] Aila, T., Karras, T., and Laine, S. On quality metrics of bounding volume hierarchies. In *Proc. 5th High-Performance Graphics Conference*. 2013.

[Ama84] Amanatides, J. Ray tracing with cones. In *ACM Siggraph Computer Graphics*. 1984.

[Arv87] Arvo, J. and Kirk, D. Fast ray tracing by ray classification. In *ACM Siggraph Computer Graphics*. 1987.

[Bil16] Billen, N. and Dutré. Visibility acceleration using efficient ray classification. *Department of Computer Science, KU Leuven*, 2016.

[Dre97] Drettakis, G. and Sillion, F. Interactive update of global illumination using a line-space hierarchy. In *Proc. ACM SIGGRAPH*. 1997.

[Gai10] Gaitatzes, A., Andreadis, A., Papaioannou, G., and Chrysanthou, Y. Fast approximate visibility on the gpu using precomputed 4d visibility fields. *WSCG*, 2010.

[Gan15] Ganestam, P., Barringer, R., Doggett, M., and Akenine-Möller, T. Bonsai: rapid bounding volume hierarchy generation using mini trees. *Journal of Computer Graphics Techniques Vol 4*, 2015.

[Gar11] Garanzha, K., Pantaleoni, J., and McAllister, D. Simpler and faster hlbvh with work queues. In *Proc. ACM Siggraph Symp. High Performance Graphics*. 2011.

[Gu13] Gu, Y., He, Y., Fatahalian, K., and Blelloch, G. Efficient bvh construction via approximate agglomerative clustering. In *Proc. 5th High-Performance Graphics Conference*. 2013.

[Hai94] Haines, E. A. and Wallace, J. R. Shaft culling for efficient ray-cast radiosity. In *Proc. Second Eurographics Workshop on Rendering*. 1994.

[Hav00] Havran, V. *Heuristic ray shooting algorithms*. Ph.D. thesis, Ph.D. Thesis, Czech Technical University in Prague, 2000.

[Hec84] Heckbert, P. S. and Hanrahan, P. Beam tracing polygonal objects. *ACM Siggraph Computer Graphics*, 1984.

[Jev89] Jevans, D. and Wyvill, B. Adaptive voxel subdivision for ray tracing. In *Proc. Graphics Interface*. 1989.

[Kar13] Karras, T. and Aila, T. Fast parallel construction of high-quality bounding volume hierarchies. In *Proc. 5th High-Performance Graphics Conference*. 2013.

[Keu16] Keul, K., Müller, S., and Lemke, P. Accelerating spatial data structures in ray tracing through precomputed line space visibility. *Computer Science Research Notes, WSCG*, 2016.

[Keu17] Keul, K., Klee, N., and Müller, S. Soft shadow computation using precomputed line space visibility information. *Journal of WSCG*, 2017.

[Kwo98] Kwon, B., Kim, D. S., Chwa, K.-Y., and Shin, S. Y. Memory-efficient ray classification for visibility operations. *IEEE Transactions on Visualization and Computer Graphics*, 1998.

[Lai09] Laine, S., Siltanen, S., Lokki, T., and Savioja, L. Accelerated beam tracing algorithm. *Applied Acoustics*, 2009.

[Lau09] Lauterbach, C., Garland, M., Sengupta, S., Luebke, D., and Manocha, D. Fast bvh construction on gpus. In *Computer Graphics Forum*. 2009.

[Mei17] Meister, D. and Bittner, J. Parallel locally-ordered clustering for bounding volume hierarchy construction. *IEEE Transactions on Visualization and Computer Graphics*, 2017.

[Mor07] Mortensen, J., Khanna, P., Yu, I., and Slater, M. A visibility field for ray tracing. In *Computer Graphics, Imaging and Visualisation, CGIV'07*. 2007.

[Pan10] Pantaleoni, J. and Luebke, D. Hlbvh: hierarchical lbvh construction for real-time ray tracing of dynamic geometry. In *Proc. High Performance Graphics*. 2010.

[Pha16] Pharr, M., Jakob, W., and Humphreys, G. *Physically based rendering: From theory to implementation*. Morgan Kaufmann, 2016.

[Ren05] Ren, Z., Hua, W., Chen, L., and Bao, H. Intersection fields for interactive global illumination. *The Visual Computer*, 2005.

[Res05] Reshetov, A., Soupikov, A., and Hurley, J. Multi-level ray tracing algorithm. *ACM Transactions on Graphics (TOG)*, 2005.

[Rit12] Ritschel, T., Dachsbacher, C., Grosch, T., and Kautz, J. The state of the art in interactive global illumination. In *Computer Graphics Forum*. 2012.

[Sti09] Stich, M., Friedrich, H., and Dietrich, A. Spatial splits in bounding volume hierarchies. In *Proc. High Performance Graphics*. 2009.

[Vin16] Vinkler, M., Havran, V., and Bittner, J. Performance comparison of bounding volume hierarchies and kd-trees for gpu ray tracing. In *Computer Graphics Forum*. 2016.

[Wal03] Wald, I., Purcell, T. J., Schmittler, J., Benthin, C., and Slusallek, P. Realtime ray tracing and its use for interactive global illumination. *Eurographics State of the Art Reports*, 2003.

[Wan16] Wang, Y., Guo, P., and Duan, F. A fast ray tracing algorithm based on a hybrid structure. *Multimedia Tools and Applications*, 2016.

[Wod17] Wodniok, D. and Goesele, M. Construction of bounding volume hierarchies with sah cost approximation on temporary subtrees. *Computers & Graphics*, 2017.

[Yin14] Yin, M. and Li, S. Fast bvh construction and refit for ray tracing of dynamic scenes. *Multimedia tools and applications*, 2014.

[Yu09] Yu, I., Cox, A., Kim, M. H., Ritschel, T., Grosch, T., Dachsbacher, C., and Kautz, J. Perceptual influence of approximate visibility in indirect illumination. *ACM Transactions on Applied Perception (TAP)*, 2009.

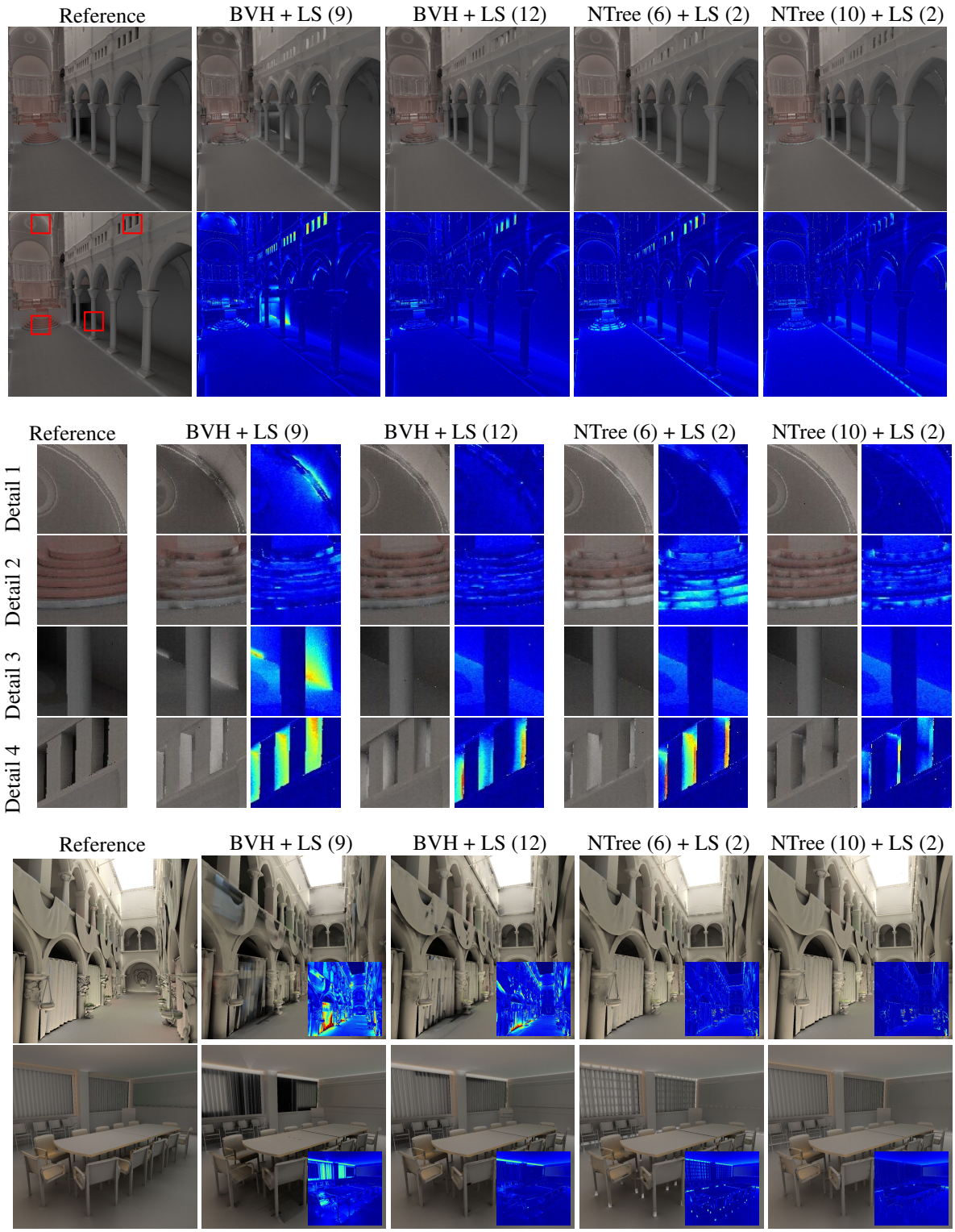


Figure 6: The test results with the architectural scenes. All images were rendered in 720p and only present indirect illumination. In bigger scenes using a lower parameter for the depth of the Line Space usage within the base data structure, more approximation artifacts due to shaft simplification occur. The detailed magnifications and the heatmaps specifically show the weaknesses of our technique using a low depth parameter. These artifacts especially occur in the transitions of different Line Spaces. However, a deeper Line Space depth improves image quality significantly, making Line Space accelerated results suitable for indirect illumination. Overall, the perception in indirect illumination given a suitable depth parameter is mostly similar to ground truth renderings, but granting significantly better performance.

ISSN 1213-6972 Journal of WSCG http://www.wscg.eu/ Vol.26, 2018, No.1

A Dynamic Non-Manifold Mesh Data Structure to Represent Biological Materials

Endre Somogyi
Dept. of Intelligent Systems Engineering, Indiana University Bloomington, IN 47405
{somogyie} @ indiana.edu

ABSTRACT

Computational models of biological materials enable researchers to gain insight and make testable predictions of quantitative dynamic responses to stimuli. These models are particularly challenging to develop because biological materials are (1) highly heterogeneous containing both biological cells and complex substances such as extra-cellular medium, (2) undergo structural rearrangement (3) couple biological cells with their environment via chemical and mechanical processes. Existing numerical approaches excel at either describing biological cells or solids and fluids, but have difficulty integrating them into a single simulation approach. We present a novel dynamic non-manifold mesh data structure that naturally represents biological materials with coupled chemical and mechanical processes and structural rearrangement in a unified way.

Keywords

Physically Based Modeling, Biological Simulation, Dynamic Meshing, Finite Element Simulation.

1 INTRODUCTION

Researchers increasingly build computational models of biological materials to gain insight and make testable predictions about responses to stimuli. Mechanistic models of biological tissues are particularly challenging to develop because biological materials are highly heterogeneous across a broad range of scales. Biological cells exist in a dynamic, spatial fluid environment and create and respond to a range of physical and chemical stimuli with complex behaviors, including movement, changes in morphology and mechanics, proliferation, death, differentiation and modification of the local environment. Biological materials combine *active agents* such as biological cells with highly heterogeneous visco-elastic substances such as extra-cellular medium (ECM), fluids and solids. We use the term *physical agent* to refer to parcels of biological material. Physical agents may be active or passive, may or may not have sub-structures, and may or may not have natural boundaries.

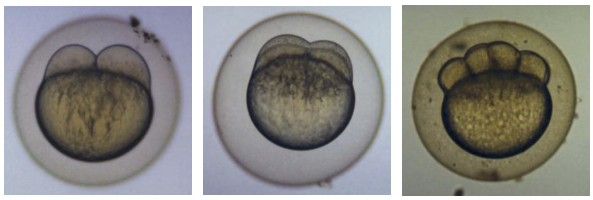
As a key enabling component of a tool for modeling biological materials, we have generalized existing manifold mesh data structures into a novel dynamic **non-manifold** mesh data structure that can consistently represent smooth deformations and dynamic topological rearrangement. This mesh forms the basis of our *Mechanica* [13] environment for simulation of cells and tissues. Our mesh represents physical agents as polytopes in contact through shared surfaces, where both sides of the surface maintain their own identity. We perform smooth mesh deformations via the explicit finite element method; we use three mesh update operations, *radial edge split*, *radial edge collapse*, and *vertex split* to perform discrete topological changes.

Biological material models are diverse, so tools to build these models must be extremely flexible and able to easily accommodate new constructs. Computational biomaterial model development tools thus need to be able to conveniently represent the interactions and dynamics of a wide range of different agents with varied material properties. Most numerical methods naturally represent only a limited and fixed subset of agent behaviors. In engineered applications designed for modularity and testability, this specialization does not usually pose a problem. For example, simulations of mechanical parts such as an aircraft wing, usually need to account primarily for mechanical and thermal properties; traditional finite-element simulators suffice for such simulations. Even complex engineered systems, such as combustion dynamics tend to involve a limited number of pre-specified physical processes such as reaction-advection-diffusion, for which finite-volume simulation is appropriate. Biological materials often lack strict modularity, and tend to be highly interdependent with large numbers of coupled chemical and mechanical processes. Fig.1, illustrates cell and tissue rearrangement and dynamics that occur during body elongation (*epiboly*) in early embryonic development. Epiboly's com-

https://doi.org/10.24132/JWSCG.2018.26.1.3

plex dynamics requires coordinated material transfer and rearrangement, formation and loss of intercomponent boundaries and intra- and extra-cellular regulation of coupled chemical and mechanical processes.

While building computational models of the complex behaviors and interactions found in nature will inevitably be cognitively challenging, the lack of tools that describe naturally the mechanics of materials composed of continually rearranging agents creates an additional and unnecessary computational burden on model developers. Many computational-mechanics



(a): 0.75 hours (b): 1 hours (c): 1.2 hours

Figure 1: **Cell rearrangement and identity change in embryonic development.** Time series of images $(0.8\,mm \times 0.8\,mm)$ of epiboly in a developing zebrafish embryo. During epiboly, ectodermal cells proliferate and flow from the dorsal to the ventral side of the embryo, as cell layers fold inwards and the yolk is incorporated into the embryo. Epiboly illustrate the complex interplay between biochemistry and mechanics during embryogenesis and the changes in agent number, shape, properties and relative positions which a tissue modeling platform must describe (from [4]).

simulation methodologies require the creation of meshes to approximate the shape and structure of physical agents. A *manifold mesh* is locally smooth and flat (homeomorphic to a disc), so a tessellated manifold surface mesh contains only edges of degree two; that is, exactly two faces share each edge. A *non-manifold mesh* is not restricted to locally-smooth surfaces, so a tessellated non-manifold surface can contains edges of degree one, two, three, or higher.

Existing dynamic manifold mesh data structures [10] often suffice to represent single cells in isolation, as long as their movement dynamics are not too complex. However biological tissues consist of large numbers of cells in contact, and these contacts continually form and disappear as cells rearrange, change shapes and adjacency, divide, merge and disappear. In *mesenchyme* (three dimensional connective tissue), both the topology and shape of cell-cell contacts can be complex. In *epithelia* (sheet-like tissues), cells contact each other via numerous locally flat surfaces, whose intersections often define geometric edges and vertices. In both cases, manifold mesh data structures are a poor match to the underlying physical reality.

When a biological cell contacts its neighboring cells and its environment Fig.1, its volume and surface both maintain their identities. With few exceptions, a membrane (and possible additional cell wall structures) separate cells from other cells and the surrounding environment. This membrane is thin relative to the size of the enclosed cell cytoplasm, with a typical size ratio of $\sim 2,000:1$. In many cases, when we are modeling cell and tissue-scale phenomena, we can approximate the membrane as a quasi-two-dimensional manifold surface embedded in three-dimensional space. The properties of living materials depend on the interplay of quasi-one-dimensional fibers (*e.g.*, in the extracellular matrix and intracellular cytoskeleton), quasi-two-dimensional sheets and membranes and fully three-dimensional structures (*e.g.*, biological cells, organelles, micelles or fluid droplets). Treating one- two- and three- dimensional agents consistently in the same mathematical and computational framework is challenging and defines the **aspect ratio problem**. This paper will focus on how our data structure represents the interaction of three-dimensional volumes with quasi-two-dimensional surfaces. We will discuss the use of the data structure to represent fibers in a subsequent paper.

In Fig.2, we categorize key identity changes that occur when physical agents consisting of a three-dimensional volume with a two-dimensional surface change adjacency. When two such initially separate agents contact each other, either: (1) the agents behave like water droplets: the contact surface between the agents disappears, while single volume and surface replace the original agents' volumes and remaining surfaces, (2) the agents behave like soap bubbles: the contact surface between the agents persists and the agents' volumes maintain their identity, but a single surface replaces the original agents' surfaces, or (3) the agents behave like biological cells: the contact surface between the agents persists and both agents maintain their volumes and surfaces. Traditional data structures usually naturally support only one of these adjacency-change processes (either 1) or 3)) and require awkward manipulations to implement the others. A single agent can also split partially or completely through the inverse of any of these processes. Our data structure enables us to efficiently represent all three types of adjacency-change.

2 RELATED WORK

Many numerical approaches address some of the computational challenges of representing biological materials, however no single existing approach can represent the variety of these materials in a self-consistent way.

The Finite Element Method (*FEM*) [14] is convenient for modeling solids under small deformations. The FEM discretizes materials into "elements" representing finite regions of space. An FEM solver generates

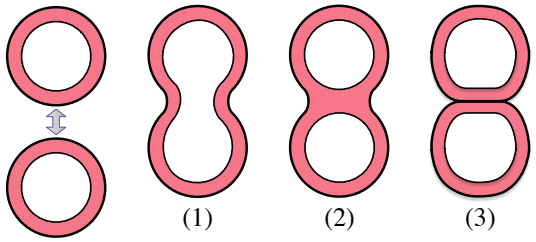


Figure 2: **Topological and identity changes on adjacency change.** When two physical agents come into contact: (1) They may merge both volumes and surfaces and lose their respective identities, like water droplets. (2) They may merge surfaces but maintain independent volumes, like soap bubbles. (3) They may maintain independent volumes and surfaces, like biological cells.

a set of nodes connecting adjacent elements, and a set of algebraic equations that define the time evolution of the nodal positions, velocities and properties at each node. Standard FEM discretizations can describe complex geometries and constitutive relations. However, most FEM methods can only naturally describe materials with a fixed dimensionality (one-, two or three dimensions). *E.g.*, to describe a composite agent with membrane and volume, standard FEM would represent the membrane using a very large number of small tetrahedra, which is computationally costly. Most FEM packages do not naturally handle large-scale material rearrangement or changing neighbor relationships.

The Finite Volume Method (*FVM*) [5] partitions space into a set of finite, connected volumes, and integrates governing equations for fluxes between volumes to calculate time evolution. FVM can represent fluid transport and reactions in complex geometries. However, most FVM approaches cannot naturally describe dynamic geometries or changing neighbor relationships.

Agent-based simulations of biological tissues typically employ lattice-, particle-, vertex-, or more recently FEM-based approaches to represent cells. Some of these approaches explicitly represent cell membranes, but many use *implicit* representations. Here we use the term *implicit* to mean data that is derived from explicit quantities. Because these are derived values, they do not have state variables, hence they can not define their own time evolution. Implicit surfaces have difficulty representing important biological processes such as surface chemistry, advection-transport, surface and edge contraction and adhesion, local signaling, etc. Explicit representation do not *ipso facto* mean that they support these biological processes, but rather that they *can* support them.

Lattice based approaches have explicit volumes and can represent both implicit and explicit surfaces, but edges and vertices are implicit. The Cellular Potts Model (CPM) usually defines an implicit representation of membranes, edges and vertices between voxels of different cells. Volume advection can be challenging in CPM as well.

Particle based approaches include center models [5] and sub-cellular element models [11]. Center models treat cells as single point particles that interact via non-bonded forces, and define the boundary of a cell implicitly. Subcellular element models represent cells as collections of point particles and permit both explicit and implicit surface representations.

Vertex models [6] represent biological cells as connected, relatively simple ($\sim 6-15$) faceted convex polyhedra, with implicit surfaces, but explicit vertices and volumes. Modern FEM approaches [1] have explicit vertices, edges and surfaces, with volumes explicit or implicit depending on the representation. With few exceptions [9], most vertex or FEM type biological cell simulations are hard-coded to solve specific biological problems and are not available as simulation environments.

Surface Evolver [3] is a program which determines the minimal energy configurations of surfaces such as soap films. Surface Evolver represents surfaces using a dynamic non-manifold mesh. However it can only implement case (2) in Fig. 2 and cannot handle explicit membranes in contact. It also does not support descriptions of the complex biochemistry of biological cells.

While existing dynamic manifold mesh data structures [10] can conveniently represent individual physical agents and sets of agents with a limited number of contacts, they typically do not directly support the variety of identity changes which occur when agents change adjacency. Most do not explicitly track cell neighbor relationships and contact areas, but calculate them as needed, which can be slow. We previously developed a numerical simulation engine using the deformable manifold mesh from the Bullet Physics library. We found that this data structure was able to calculate deformations for at most 20 cells in contact, before performance dropped to unacceptable levels. Benchmarking showed that because all cells are in physical contact with each other, collision detection was computationally expensive. Furthermore, determining cell neighbor relationships and contact areas between cells (which Bullet Physics required us to at each time step) was computationally costly. Compute time using our non-manifold boundary representation mesh scales linearly with the number of vertices plus the number of triangles; essentially, performance is proportional to the total surface area rather than the total volume.

3 APPROACH

To represent and simulate physical agents, we must consider two related questions: (1) How do we rep-

resent the *structure* of these agents when the physical properties of the agent's constitutive elements may all differ? and (2) How do we represent *dynamics* of these physical agents including deformation and structural rearrangement.

3.1 Physical Structure

Traditional numerical approaches do not adequately address agents with changing adjacency. Tissues contain many cells (composite agents) which frequently change their adjacency, so we need to efficiently represent: (1) multiple agents in contact, (2) individual cell surface chemical process occurring on each cell's surface, and (3) chemical processes occurring between neighboring cells. Physical agents (fluid droplets, soap bubbles, biological cells and tissues) can have a well-defined boundary that has intrinsic material and chemical properties distinct from the agents they envelop. We represent biological materials with an explicit boundary, dynamic non-manifold mesh data structure, inspired by Hun and Lee's partial entity [8] and Weiler's radial edge [16] structures. This mesh data structure enables us to faithfully represent changing neighbor relationships and chemical and mechanical processes in a unified way.

Consider two biological cells are in contact as in Fig.2.3. When we take a section of this contacting region, we can make the abstraction that there are basically three kinds of physical materials here: (1) the membrane that belongs to the first cell, (2) the interstitial material between the cells, and (3) the membrane of the second cell. Each of these regions is thin relative to the size of the cells, but still has finite thickness. We represent this stack of physical materials with the *Triangle* data structure. The triangle itself is a composite type that can be thought of as a "sandwich" formed from a left *Partial Triangle*, the triangle itself, and a right partial triangle. The triangle represents a complete section of this contact region, where the partial triangles represent one side of a boundary, i.e. the biological cell's membrane.

Our mesh data structure consists of four key data types: vertices, partial triangles, triangles and cells. Vertices represent a position in space, maintain a list of incident triangles and cells, and they presently do not store any other state variables. We *measure* mass at each vertex as the barycentric area (1/3 the area) weighted sum of each incident triangle and partial triangle's mass.

The partial triangle Fig.3 represents one side of a physical boundary, i.e., a biological cell membrane. Each partial triangle belongs to the boundary of a specific cell, and the cell's boundary is defined as a set of connected partial triangles that form a closed manifold surface. Each partial triangle has pointers to its base triangle, the cell that it belongs to, its opposing partial triangle, and to its three neighboring partial triangles that are also part of the same cell's boundary. A partial triangle has an explicit mass, and can contain other state variables such as chemical amounts. The triangle data structure represents a section of shared boundary between physical objects, and is itself a composite structure of two partial triangles. Each triangle contains pointers to three vertices. Like the partial triangle, the triangle has an explicit mass, and can contain a vector of attached chemical amounts. We call the edge where two or more triangles intersect a *radial edge*.

A cell represents a closed physical region of space demarcated by an explicit boundary composed of partial triangles. A cell type can represent a physical agent, such as fluid droplets, fluid volumes, soap bubbles, or biological cells. A cell contains pointers to the partial triangles that comprise the cell's boundary. Presently, we approximate physical agents as homogeneous deformable solids, but we plan to add more complex internal structures in future versions. The cell has an explicit mass and can contain a vector of chemical amounts. Two cells in contact form a manifold surface, three or more cells can intersect at a radial edge as in Fig.3.b forming a non-manifold intersection between three or more surfaces. The partial triangles in a non-manifold intersection are adjacent on only one side of the triangles incident to the radial edge.

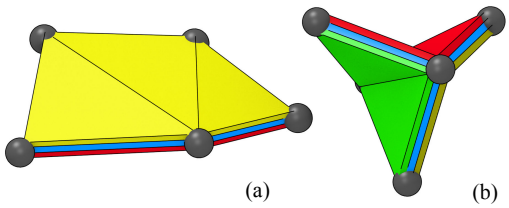


Figure 3: The partial triangle data structure enables us to represent both (a) manifold and (b) non-manifold meshes. (a) shows a manifold surface between a yellow and a red cell, (b) shows a non-manifold edge between a red, green and yellow cells.

3.2 Dynamics

The physical agents that we represent are not static – both their positions and their states evolve in time. They move around and past each other, and regularly come into and out of contact with each other (i.e., change neighbor relationships). We represent cell motion, membrane deformation, and chemical processes such as cell-signaling, membrane transport as continuous, ordinary differential equation defined processes. We implement large-scale structural rearrangement and topological change, such as large shape changes and cell attachment/detachment as discrete rule-based events. Vertices move according to the laws of classical mechanics, and we calculate their time

evolution using the *Propagator* data type. Chemical reaction-transport processes are well-studied, and we re-use our existing solver [12] to implement them. The *Rules Engine* performs large scale rearrangement and topological transformations with rule triggered mesh reconnect operations. Mesh reconnect operations alter the underlying mesh, they define how elements split and merge - that is, they define local topological rearrangement. Mesh operations can create and delete mesh objects (triangles, and ultimately cells) and alter their neighbor pointers and vertex positions. Exactly when to apply a mesh update operation is highly material and simulation specific. Users need the ability to define custom triggering rules for mesh updates in order to model a range of different physical materials. Thus, we have developed a general rule-based mesh update system that separates the mesh update operation from the triggering rule, and enables the user to readily add new mesh update operations and to write custom, material-specific trigger rules.

The Propagator calculates the continuous-time evolution of the material vertices, as well as the fluid materials and state variables located on the triangle, partial triangle and cell elements. In other words, the propagator calculates the subsequent simulation state based on the current state. The propagator queries the model for the net force on each vertex and the rate of change of each fluid or other state variables associated with the cells and surfaces. The propagator presently uses a Runge-Kutta integrator, however we plan on adding more sophisticated integrators in future versions.

In order to enforce constraints, the propagator implements a Position Based Dynamics (PBD) constraint solver [2]. The PBD solver enforce constraints by adjusting the positions of the mesh vertices. The solver applies a correction displacement to the vertex positions. The basic idea of the PBD constraint solver is to first project all of the mesh vertices forward in unconstrained motion, according to the net force acting on each vertex. The unconstrained motion most likely violates some constraints. Each constraint object contains two functions: a) a constraint function of the current model state that yields a scalar constraint value, and b) a function that calculates the rate of change of the constraint function relative to the mesh vertices, i.e. the Jacobian of the constraint function. The constraint is satisfied when the constraint function evaluates to zero. The propagator sequentially adjusts the vertex positions in a Gauss-Siedel fashion, according the constraint Jacobian until the constraint function reaches a tolerance. We have found that volume constraints are satisfied with only 2-3 iterations.

The Rules Engine is responsible for discontinuous time state changes in the mesh and other state variables. The rules engine maintains a list of rules that associate a trigger condition and energy function with a discrete mesh update operation. The Rules Engine monitors the mesh for positional and topological changes and looks for mesh configuration patterns that match a trigger condition. When a configuration matches a trigger, the rules engine applies the corresponding mesh operation to the mesh, thus modifying the mesh. We present three mesh update operations: radial edge split, radial edge collapse, and vertex split. Each mesh update rule also defines an energy function. We associate an energy with each mesh topological configuration, analogous to the way a potential energy is associated with different configurations in physics. For example, users may wish to define a rule that penalizes excessively long edges, where the energy function could be proportional to the square of the edge length, say kx^2, and then associate the edge split operation with this energy function. Here, the rules engine might find an edge, and perform a trial edge split. The initial energy of this edge would be kx^2, and the energy of the split edge would be $2 * k(x/2)^2 = (1/2)kx^2$, thus the change in energy is $-k/2x^2$. The rules engine determines that this is an energetically favorable operation and applies it. The rules engine stores all triggered rules in a priority queue, ordered energy value, such that the most energetically favorable rules are evaluated first. This approach is similar to [6], where they perform triangle to edge and edge to triangle operations, and store all pending operations in a priority queue ordered on edge length. Because the rules engine successively applies rules sorted by energy level, each time a rule is evaluated, the rules engine looks at which mesh objects were altered by the rule and removes any pending operation that also depends on these objects. If that pending operation is still valid, it will be triggered and queued in the next time step. These rules enable us to to represent objects separating and rearranging, and we intend to add more rules as needed.

3.2.1 Radial Edge Split

The triangles in an evolving mesh can become too large to adequately represent an object's surface which can result in increased numerical error. In order to accurately sample and represent spatially variant physical quantities, we must refine large triangles. Triangle refinement replaces a single large triangle with two or more smaller triangles. Numerous approaches subdivide a single triangle into three triangles, but the more common approach is split a single triangle into two triangles. In a manifold mesh, when a single triangle is split in two, the neighboring triangle incident to the split edge must also be split, hence the name, "edge split". The radial edge split operation is a generalization of the commonly used manifold edge split, with the manifold edge split being a special case of the radial edge split. The radial edge split creates a new vertex at the

midpoint of edge, and identifies all incident triangles around this edge, and splits each one of them into two triangles. The new radial triangles share the space as the original triangles did. Because the radial edge split does not move any existing vertices and only inserts a new vertex, the operation does not need to check for any topological or geometric violations - the radial edge split operation is always topologically valid. As such, it is one of the simplest mesh operation to implement.

For each triangle in a radial edge, the radial edge split operation identifies the outer-most vertex, and creates two new triangles and removes the original triangle. Each of the two new triangles share an edge formed by the new center vertex and the outer vertex. Radial edge split then reconnects these new triangle neighbor pointers to the triangles adjacent to the removed triangles.

3.2.2 Radial Edge Collapse

As a simulation evolves in time, a triangles can become excessively small. This over-refinement leads to wasted compute resources and performance degradation. The radial edge collapse operation removes small triangles and re-connects these triangles' neighbors. The radial edge collapse is a generalization of the conventional manifold edge collapse operation as studied by [15, 7], where the manifold edge collapse is a special case of the radial edge collapse. In a manifold edge collapse, an edge can be incident to exactly two triangles. A radial edge however can have any number of triangles arranged radially around an edge as in Fig.3. The idea however is the same: we want to remove the edge and re-connect all of the neighboring triangles. A variety of different trigger conditions may be appropriate for initiating a radial edge collapse. Say one would like to coarsen a mesh in areas of low curvature, or, say one wants to remove all triangles below a certain edge length threshold, as is the case in [6]

A radial edge collapse traverses every triangle in a radial edge and collapses the triangle along the edge side, and re-connects the neighboring triangles on the collapsed triangle's two remaining edges. The edge collapse removes a region of mesh which consists of the edge itself and its two incident triangles and enlarges the neighboring triangles to fill the void. The edge collapse operation reduces the dimensionality of a mesh. As such, it is only applicable under specific conditions. For an edge collapse (or any other operation) to be valid, it must not invert any triangles, i.e., it must not change any triangle's normal by more than 90 degrees. Vieira et al. [15] identified that a manifold edge collapse is valid if it does not violate the *link condition*. The link condition sates that an edge $e = \{u, v\}$ can be collapsed if and only if $link(u) \cap link(v) = link(e)$. In a triangular mesh, the star of a vertex v is the set of triangles and edges that are incident to v. The link of a vertex is the frontier of the star. The frontier of the star is the set of vertices that are incident to every edge and triangle in the star, not including v. The links of the edge and each vertex for non-manifold meshes are slightly more complex to calculate than the manifold case. We first build the edge link by iterating over every triangle in the radial edge and inserting the outer vertices into the edge link set. Because we only need to test that the intersection of the u and v link sets are equal to the edge link set, we do not need to build the entire link set for each vertex. Rather, we only need to build the $link(u)$ set out of the vertices adjacent to u, that are not in the $link(e)$ set. If we find a vertex adjacent to the other edge vertex v that is not in $link(e)$ and not in the $link(u)$, the radial edge violates the link condition.

In a radial edge collapse, the link condition alone does not guarantee that a vertex move will not invert a triangle. We must also explicitly test triangle incident to each of the radial edge endpoints. We test each of these triangles by test-moving the edge endpoint vertex into a trial position and testing for a change in the triangle normal direction. Presently, we also test to ensure that a radial edge collapse will not collapse a tetrahedron down to a triangle. We check this by looking at the partial triangles on each face of a radial edge triangle. If that partial triangle's two outer neighbors are adjacent to each other, then this partial triangle and its two neighbors define three faces of a tetrahedron, with an open base. The collapse of this triangle will result its two neighboring triangles collapsing down to each other - that is, collapsing a tetrahedron to a triangle.

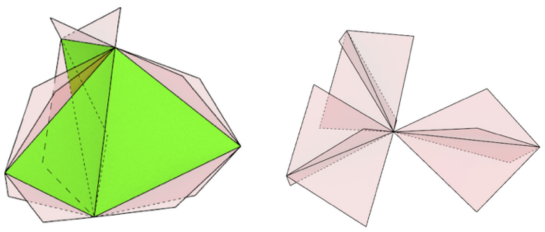


Figure 4: The radial edge collapse operation collapses a set of radial triangles (green) down to a single vertex, pulls in the remaining triangles (pink) to fill the newly created hole, and connects each remaining triangle to maintain mesh connectivity.

3.2.3 Vertex Split

The cell valence count of a vertex is the number of cells that share that vertex. The previously described radial edge collapse removes a vertex from a mesh and attaches all of this vertex's cells to another vertex, thus potentially increasing the cell valence count of the remaining vertex. Many mesh generation packages, tend to produce meshes that are predominantly pentahedron

and hexahedron, thus contain a large number of vertices with eight cell valences. Eight-valence vertices are unusual in nature, and high or unbalanced vertex cell valences also tend increase computational cost and numerical error. The vertex split operation here is a non-manifold generalization of Hoppe's [7] manifold vertex split. The vertex split operation here splits a single vertex into two vertices and detaches a cell from a vertex in order to reduce vertex cell valence count. We can see in in Fig. 5 an example of a vertex with a high cell valence count, (five in this case), and relaxed configuration after five vertex splits. The vertex split operation enables cells to reconnect to each other. In a manifold mesh, a vertex split is the exact inverse of the edge collapse operation. In our non-manifold mesh, the vertex split operation shares a similar relationship with the radial edge collapse, though they are not exact inverses of each other.

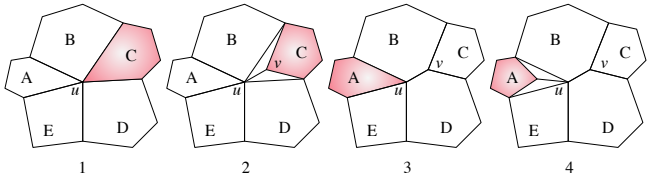


Figure 5: A sequence of vertex split operations in 2D. The central vertex initially has a cell valence count of 5, and the C cell (in red) is selected for ejection. The vertex split locates the shared face between C and its neighboring cells, creates new vertex towards the center of C, pulls the neighboring face of C to this new vertex, and creates a set of triangles to fill this void. The original vertex now has a cell valence count of 4. The vertex split then selects the A cell for ejection, creates a new vertex, moves the shared face, and fills the void again. The final images show the mesh after two subsequent vertex split operations. Note, the vertex split is a localized operation, the frontier elements of these cells are not modified.

A vertex split begins with a candidate vertex u and identifies a target cell C to disconnect and pull away from u. Vertex split will then split u into two vertices, u and v, where v is attached to the target cell, C, which is then no longer incident to u. As C is disconnected from u, u's cell valence count is reduced by one. The cell valence count of v is then the number of cells that that are both incident to the original vertex u, and adjacent to C, i.e. $valence\_count(v) = |incident\_cells(u) \cap adjacent\_cells(C)| + 1$. Thus, in order to reduce net valence counts, it is important for vertex split to choose C such that $valence\_count(v) < valence\_count(u)$. Otherwise, vertex split will indeed reduce the valence count of u, but it will also create another vertex v that has the same valence count that u originally had.

The most challenging task of the vertex split operation is filling the hole created by splitting and moving the vertex. This task is illustrated in Fig. 6. The vertex move creates a void between the cell being moved (C) and the set of cells adjacent to C, and incident to the original vertex u. In order to fill this void, the vertex split iterates around the triangle fan centered at v, composed of triangles that face C. Every triangle in this fan, by definition has exactly one face in the C cell surface. Consider a sequential pair of triangles in this fan. The non-C facing partial triangles either belong to a same cell, or they point to different cells. If both partial triangles point to the same cell on both sides, then there is nothing to do, these triangles simply shift slightly, as their center vertex gets attached to a new vertex v. If, however, there is a cell change, (i.e., the first triangle's non-C facing partial triangle points to a different cell than the second triangle's non-C facing partial triangle) then the vertex split inserts a new triangle at this edge. We say that triangle t_1 is incident to cells A and C, and triangle t_2 is incident to cells B and C. Vertex split then creates a new triangle, t_n with vertices v, u, and v_f, the vertex on the frontier of the fan that is incident to both t_1 and t_2. The t_n triangle faces cell A on one side, and cell B on the other. The vertex split will continue iterating around this fan until it encounters the starting triangle. Vertex split here creates a set of new triangles, one for each pair of triangles that have a cell change. All of these new triangles share the new $\{u,v\}$ edge, and when vertex split completes creating these new triangles, it then connects their partial triangles together appropriately. We can see that vertex split is essentially the inverse of the radial edge collapse, in that a radial edge collapse removes a set of triangles around a radial edge, and vertex split creates a set of triangles around a radial edge.

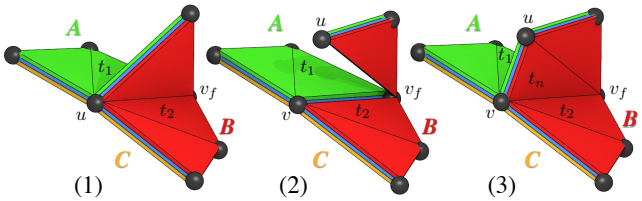


Figure 6: Three stages in a vertex split operation. This operation first identifies a target cell, C in yellow, and pulls this cell away, then splits the vertex u into u and v, and finally fills in the resulting holes with new triangles. Cell C is adjacent to cells A and B. Vertex split ensures that the partial triangles on the new triangles are connected to the correct cell boundaries.

3.3 Implementation Details

Biological cells and membranes participate in chemical processes that can often occur at significantly faster timescales than the motion of the objects themselves. In

biological material simulations, the most computationally expensive task is evaluating these continuous time chemical reactions local to cells, surface elements and the extracellular milieu, as well as the chemical fluxes between neighboring spatial objects. The topological rearrangement operations previously discussed perform a significant number of mesh traversals when evaluating mesh operation rule triggers, and when performing the mesh update operations themselves. In order to maximize the performance of the chemical process solver and mesh traversals, we choose to explicitly store all neighbor relationships in the mesh at the expense of increased memory usage and software complexity.

We provide two basic mesh traversal operations: **radial edge** and **triangle fan** traversals. A radial edge is an edge connecting two or more incident triangles. To enumerate these incident triangles, the radial edge traversal (1) starts with a partial triangle and a pair of vertices that define an edge, (2) finds the next partial triangle that is both a neighbor of the current partial triangle, and incident to the edge, (3) identifies the opposite partial triangle (4) continues the iteration until it come across the starting partial triangle. A triangle fan is the set of triangles that are incident to both the same vertex and the same cell. (i.e., a triangle fan is a set of triangles, all of which face the boundary of a cell and surround a specified vertex.) The triangle fan traversal (1) starts with a triangle, vertex and cell, (2) follows the partial triangle neighbor pointers until the original triangle is encountered again.

4 RESULTS

We have created a set of simple models that demonstrate the mesh reconnect rules. These models read an initial configuration from a gmesh file. These models all implement a surface tension term (a force that acts in a direction tangent to the surface), and a volume constraint (acts perpendicular to the surface).

We use Zheng's method [17] to calculate the surface tension force. This method calculates the surface tension contribution of a triangle's three barycentric regions to each corresponding vertex. Each barycentric region pulls its' incident vertex in towards the triangle's barycenter. The magnitude of each Voronoi region's surface tension is proportional to the 1/2 the length of opposite edge. For details, see page 39 in [17].

We define the volume constraint on a per cell basis as the difference between a target volume and the present volume, $C(X) = \lambda(v_t - v)$, where v_t is the target volume, v is the current volume, and λ represents the stiffness of the constraint. Cells with a lower λ are "softer", as the constraint solver adjusts the cell positions more rapidly in cells with a larger λ. We approximate the volume constraint Jacobian, as in [2] as area weighted surface normal to each vertex. We use the barycentric area of each triangle to estimate the surface area of each vertex, and we compute the total volume of each cell using the divergence theorem.

To illustrate the effect of the edge collapse and vertex split operations, we performed an experiment with four cells and applied a harmonic bond force to the center of mass of two cells in order to bring the cells in contact depicted in Fig. 7. The initial condition contained the two red cells topologically connected with each other - they share triangles, and the two blue cells are disconnected.

We then apply a harmonic force to the center of mass of each blue cell which caused the blue cells to come closer, and start to push the red cells apart. We can see the edge length shrinking between the blue cells. When the edge length drops below a threshold, the radial edge collapse rule culls these short edges. This operation however increases the cell valence count on the central vertexes to four. The increased valence count causes the rules engine to invoke the vertex split operation. We specified a rule that causes the cell with the highest curvature to be ejected from the vertex. The vertex split operation splits these vertices and creates the new triangles in yellow. We can then see that as these two blue cells continue moving towards each other, these central edges continue to fall below the threshold, and continue to be culled, thus invoking further vertex splits. In the last frame, we can see that all of the shared contacts between the red cells has been eliminated. At this point, we remove the force between the two blue cells, and allow system to equilibrate. We can see that in the final frame, the material is topologically very distinct from the initial configuration. In the final frame, the blue cells are not connected and the red cells are disjoint. The mesh operations provide one possible way to maintain a memory on materials, as these are atomic operations that alter the mesh structure and topology.

To illustrate the effects of differential surface tension and demonstrate the mesh update operations when cells separate, we created a model that mimics a simple biological model of cellular mitosis. Here, we represent a biological cell that is about to undergo mitosis as a pair of Mechanica cells in contact, separated by a membrane. We first create two cubic cells in contact and set the surface tension of the membrane that separates these two cells to zero and set the surface-tension of the membrane not in contact to a positive value and allow the simulation to relax. As expected, the two cells in contact in a relaxed configuration form a nearly perfect sphere as in Fig. 8. Because the shared membrane is initially not under tension, it does not exert any force on the outside membranes. Once the system is relaxed (no net motion), we increase the surface tension on the shared membrane to a positive value. We can see that this shared membrane begins to pull in-

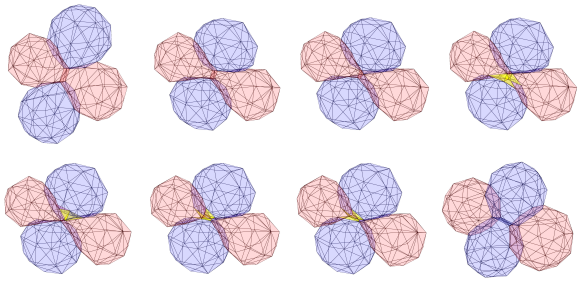


Figure 7: A series of frames from a simulation that starts with an initial configuration with the two red cells connected. We then apply a force between the two blue cells, which causes the blue cells to come towards each other and push the red cells out of the way. We can see the surface shrinking between the blue cells, as these edges shrink. The rules engine invokes the edge collapse operation to cull the short edges, which results in a number of vertices with a large cell valence count. The rules engine then invokes the vertex split operation to peel the red cells away. We then remove the harmonic bond between the blue cells, the system then relaxes on its own. The yellow regions indicate triangles that were generated by vertex split operations.

wards on the outer membrane, and the two Mechanica cells in contact begin to take the classic shape of a biological cell undergoing mitosis. We then increase the shared membrane surface tension to a value larger than the outer membrane surface tension. At this point, the shared membrane will continue pulling the outer membranes inwards, and shared contact area between the cells shrinks. As this shared area shrinks, the edge length of triangles within this shared area drop below the edge length cutoff, and the rules engine applies the radial edge collapse rule to cull these offending triangles. The shared surface area continues to shrink and pull inwards on the outer membranes until the shared area eventually disappears. The shared area disappears when the rules engine culls the last remaining triangle in that area. At this point, the two cells are no longer in contact, and their respective surface areas relax to approximate a sphere.

5 CONCLUSIONS

Researchers who simulate biological systems must take into account both mechanical and chemical processes. They must also represent objects that continually move and change neighbor relationships. Most existing numerical simulation approaches such as the finite element or the finite volume methods excel at representing certain aspects of physical objects. Finite element is ideal for simulating mechanical properties of materials with limited structural rearrangement. Finite volume methods excel at simulating complex fluids in relatively stationary geometries. Few, if any simulation

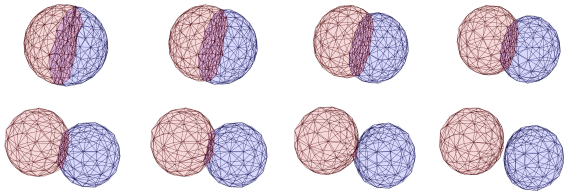


Figure 8: A simple cellular mitosis model. Two cells are initially in contact with a shared membrane, and each cell's individual membrane has a surface tension of σ, and the the shared membrane's net surface tension is zero. As the shared membrane is completely relaxed, the two cells in contact form a sphere. We then increase the surface tension of the membrane to a positive value which causes the membrane to shrink. As the membrane shrinks, the mesh update operations automatically cull the triangles with short edges and split large triangles. The mesh update operations enable the shared membrane to shrink and eventually disappear, at which point the cells separate.

platforms that can do both at once using the same simulation approach. The novel mesh data structure we have developed meets this challenge and enables us to represent a wide range of physical and biological materials in a unified way. As an example of the applicability of our platform, consider again the epiboly process depicted in Fig. 1. This biological process is challenging to represent using current simulation approaches because it involves a variety of different chemical and mechanical processes. Presently, researchers must resort to integrating a range of different numerical simulation methodologies and programs. Using our new numerical simulation engine, the representation of the epiboly process is much simpler, is done entirely self consistently, and requires no integration with external solvers. We are presently developing a cell-division mesh update operation that will split a single cell into two cells. When this new operation is ready, we will be able to directly represent this key biological process. Take, for example, Fig. 1. This image depicts a cluster of biological cells on top of a yolk sack. The cells start out as a blob on top of the yolk sack. This blob then gradually spreads out across the yolk, ultimately enveloping most of the yolk. We could represent each embryo cell as specific Mechanica cell type and represent the yolk as another cell type. We know that there are a number of chemical reaction processes that occur inside each embryonic cell. We presently support implementing chemical reaction networks inside our cell types. We also know that these embryo cells communicate with one another via intricate signaling pathways, and we implement these as chemical fluxes between cells. We also know that the embryo cells are initially blob-like, and later move to envelop the yolk. We can represent this transition with a surface tension

like force between the cells and the yolk. Initially we could have a high surface tension between the embryo cells and yolk, and lower surface tension between embryo cells of the same type. Later, possibly based on some chemical state variable in each embryo cell, we could have the embryo-yolk surface tension reduce, and the embryo-embryo surface tension increase. Here, we would expect the embryo cells to then decrease their affinity towards each other, and their affinity towards the yolk to increase and spread out over the yolk. The embryo development process exemplifies how biological processes couple chemical and mechanical interactions - a feature that the novel numerical simulation engine we have developed uniquely captures, enabling researchers to model these kinds of processes. The mesh data structure presented here lays the groundwork for this numerical simulation engine.

Our simulation code is currently under active development, all source code and binaries *will be made* freely available on the Mechanica website, (`http://www.mechanica.org`) under a open source (GPL) license.

6 ACKNOWLEDGMENTS

We acknowledge generous financial support the National Institutes of Health, National Institute of General Medical Sciences, grant R01 GM122424, and National Science Foundation grant 1720625. We thank Dr. David Umulis, Dr. Marie Gingras, Dr. Amit Hagar and Dr. James P. Sluka for their discussion and insights.

7 REFERENCES

[1] S. Alt, P. Ganguly, and G. Salbreux. Vertex models: from cell mechanics to tissue morphogenesis. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 372(1720):20150520, May 2017.

[2] J. Bender, M. Müller, M. A. Otaduy, M. Teschner, and M. Macklin. A Survey on Position-Based Simulation Methods in Computer Graphics. In *Computer Graphics Forum*, pages 228–251, Sept. 2014.

[3] K. A. Brakke. The surface evolver. *Experimental Mathematics*, 1(2):141–165, 1992.

[4] T. Braunbeck and E. Lammer. Fish Embryo Toxicity Assays. Technical report, German Federal Environment Agency, 2006.

[5] A. Ghaffarizadeh, R. Heiland, S. H. Friedman, S. M. Mumenthaler, and P. Macklin. Physicell: An open source physics-based cell simulator for 3-d multicellular systems. *PLoS computational biology*, 14(2):e1005991, 2018.

[6] H. Honda, M. Tanemura, and T. Nagai. A three-dimensional vertex dynamics cell model of space-filling polyhedra simulating cell behavior in a cell aggregate. *Journal of Theoretical Biology*, 226(4):439–453, 2004.

[7] H. Hoppe. *Progressive meshes*. ACM, New York, New York, USA, Aug. 1996.

[8] S. H. Lee and K. Lee. Partial Entity Structure: A Compact Boundary Representation for Non-Manifold Geometric Modeling. *Journal of Computing and Information Science in Engineering*, 1(4):356–365, 2001.

[9] R. Merks, M. Guravage, and D. Inzé. VirtualLeaf: an open-source framework for cell-based modeling of plant tissue growth and development. *Plant physiology*, 155:656–666, 2011.

[10] T. Odaker, D. Kranzlmueller, and J. Volkert. View-dependent simplification using parallel half edge collapses. In *Proceedings of the WSCG*, pages 63–72, 2015.

[11] S. A. Sandersius, C. J. Weijer, and T. J. Newman. Emergent cell and tissue dynamics from subcellular modeling of active biomechanical processes. *Physical biology*, 8(4):045007, July 2011.

[12] E. T. Somogyi, J.-M. Bouteiller, J. A. Glazier, M. König, J. K. Medley, M. H. Swat, and H. M. Sauro. libRoadRunner: a high performance SBML simulation and analysis library. *Bioinformatics*, 31(20):3315–3321, June 2015.

[13] E. T. Somogyi and J. A. Glazier. A modeling and simulation language for biological cells with coupled mechanical and chemical processes. In *Symposium on Theory of Modeling Simulation*, Virginia Beach, Apr. 2017. Society for Computer Simulation International.

[14] M. Verschoor and A. C. Jalba. Elastically Deformable Models based on the Finite Element Method Accelerated on Graphics Hardware using CUDA. *Journal of WSCG*, 2012.

[15] A. W. Vieira, L. Velho, H. Lopes, G. Tavares, and T. Lewiner. Fast Stellar Mesh Simplification. *SIBGRAPI*, pages 27–34, 2003.

[16] K. Weiler. *The radial edge structure: a topological representation for non-manifold geometric boundary modeling* . Geometric modeling for CAD . . . , 1988.

[17] W. Zheng, B. Zhu, B. Kim, and R. Fedkiw. A new incompressibility discretization for a hybrid particle MAC grid representation with surface tension. *Journal of Computational Physics*, 280:96–142, Jan. 2015.

ISSN 1213-6972 Journal of WSCG http://www.wscg.eu/ Vol.26, 2018, No.1

Radial Line Fourier Descriptor for Historical Handwritten Text Representation

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

Automatic recognition of historical handwritten manuscripts is a daunting task due to paper degradation over time. Recognition-free retrieval or word spotting is popularly used for information retrieval and digitization of the historical handwritten documents. However, the performance of word spotting algorithms depends heavily on feature detection and representation methods. Although there exist popular feature descriptors such as Scale Invariant Feature Transform (SIFT) and Speeded Up Robust Features (SURF), the invariant properties of these descriptors amplify the noise in the degraded document images, rendering them more sensitive to noise and complex characteristics of historical manuscripts. Therefore, an efficient and relaxed feature descriptor is required as handwritten words across different documents are indeed similar, but not identical. This paper introduces a Radial Line Fourier (RLF) descriptor for handwritten word representation, with a short feature vector of 32 dimensions. A segmentation-free and training-free handwritten word spotting method is studied herein that relies on the proposed RLF descriptor, takes into account different keypoint representations and uses a simple preconditioner-based feature matching algorithm. The effectiveness of the RLF descriptor for segmentation-free handwritten word spotting is empirically evaluated on well-known historical handwritten datasets using standard evaluation measures.

Keywords

Radial Line Fourier descriptor, word spotting, feature matching

1 INTRODUCTION

Automatic recognition of poorly degraded handwritten text is challenging due to complex layouts and paper degradations over time. Typically, an old manuscript suffers from degradations such as paper stains, faded ink and ink bleed-through. There is variability in writing style, and the presence of text and symbols written in an unknown language. This hampers the document readability, and renders the task of searching a word in a set of non-indexed documents i.e. word spotting, to be more difficult.

In literature [Gio17], word spotting approaches can either be segmentation-based where the search space consists of a set of segmented word images, or segmentation-free with the complete document image in the search space. This paper focuses on segmentation-free word spotting, which is typically preferred over segmentation-based methods when dealing with heavily degraded document images [Zag17]. However, the performance of word spotting algorithms significantly depends on the appropriate selection of feature detection and representation methods [Gio17]. In general, feature descriptors represent a region with distinct feature in a document image, coded into a numerical feature vector, which is subsequently compared with the feature vector of a reference image to perform matching.

Efforts have been made in the recent past towards research on feature detection and representation methods. Some popular methods include Scale Invariant Feature Transform (SIFT) [Low04], Speeded Up Robust Features (SURF) [Bay08] and Histograms of oriented Gradients (HoG) [Dal05]. SIFT and HoG contributed significantly towards the progress of several visual recognition systems in the last decade [Gir14]. However, these local descriptors were mainly designed for the representation of natural scene images, that possess structurally different characteristics from the document images. For example, the detection of the most important edges using pyramid scaling in SIFT creates local interest points between the text lines [Zag17]. The invariant properties of these descriptors amplify the noise in the degraded document images, rendering them

https://doi.org/10.24132/JWSCG.2018.26.1.4

more sensitive to noise and complex characteristics of historical manuscripts [Zag17]. The work by [Ley09] analyzed that the rotation-invariant features are more sensitive to noise in a document image, and perform poorly as compared to rotation-dependent features.

Since the existing descriptors are found to be unsuitable for representing handwritten text with high levels of degradations [Zag17, Ley09], it is important to design a descriptor to address this issue. This paper introduces a Radial Line Fourier (RLF) descriptor which is tailor-made for word spotting applications with fast feature representation and robustness to degradations. RLF is a fast and short-length feature vector of 32 dimensions, based on log-polar sampling followed by computing a few elements of the Discrete Fourier Transform (DFT) along each radial line. It does not require any orientation information from the feature detectors, and simple feature detectors can be used without compromising the descriptor and word spotting performance.

This paper is organized as follows. Section 2 reviews the state-of-art methods used in word spotting pipeline, with main focus on interest point detection and feature representation methods. Section 3 presents the proposed method based on the RLF descriptor for segmentation-free handwritten word spotting. Section 4 demonstrates the efficacy of the proposed method on well-known historical datasets using standard evaluation measures. Section 5 concludes the paper.

2 RELATED WORK

Appropriate selection of interest points (keypoints) and feature descriptors is indispensable for the performance of a word spotting system. This section discusses some popular interest point detection and feature representation methods with reference to word spotting systems. It is important to note that the segmentation-free word spotting framework presented herein is training-free, therefore training-based methods such as deep learning are not considered.

2.1 Interest Point Detection

Feature detection, or interest point detection refers to finding keypoints in an image that contain crucial information. There exist several interest point detectors in literature. For example, the Harris corner detector [Har88] is popularly used for corner points detection. It computes a combination of eigenvalues of the structure tensor such that the corners are located in an image. Shi-Tomasi corner detector [Shi94] is a modified version of Harris detector. The minimum of two eigenvalues is computed and a point is considered as a corner point if this minimum value exceeds a certain threshold. The Maximally Stable Extremal Regions (MSER) [Mat02] detector detects keypoints such that all pixels inside the extremal region are either darker or brighter than all the outer boundary pixels.

Typically, interest point based feature matching is performed by using a single interest point detector type. SIFT and SURF are the most popular detectors that capture the blob type of features in the image. SIFT uses the Difference of Gaussians (DoG) that computes the difference between Gaussian blurred images using different values of σ, where σ defines the Gaussian blur from a continuous point of view. SURF computes the Determinant of the Hessian (DoH) matrix, that defines the product of the eigenvalues. In principle, any combination of different keypoint detectors can be selected depending upon the application. This work uses a combination of four types of keypoint detectors for handwritten text representation, that consists of corner detectors, dark and bright blobs, saddle points, and the edges of text strokes.

2.2 Feature Representation

After a set of interest points has been detected, a suitable representation of their values has to be defined to perform word matching. In general, a feature descriptor is constructed from the pixels in the local neighborhood of each interest point. Fixed length feature descriptors are most commonly used that generate a fixed length feature vector, which can be easily compared using standard distance metrics (e.g. the Euclidean distance). Sometimes, fixed length feature vectors are computed directly from the extracted features without the need of a learning step [Gio17].

Gradient-based feature descriptors tend to be superior, and include SIFT [Low04], HoG [Dal05] and SURF [Bay08] descriptors. The 128-dimensional SIFT descriptor is formed from histograms of local gradients. SIFT is both scale and rotation invariant, and includes an intricate underlying framework to ensure this. Similarly, HoG computes a histogram of gradient orientations in a certain local region. An important difference between SIFT and HoG is that HoG normalizes the histograms in overlapping blocks, and creates a redundant expression. SURF descriptor is generally faster than SIFT, and is created by concatenating Haar wavelet responses in sub-regions of an oriented square window. SIFT and SURF are invariant to both scale and rotation changes. There are several variants of these descriptors that have been employed for word spotting [Rod08, Gio17].

Many feature descriptors use local image content in square areas around each interest point to form a feature vector [Has16]. Both scale and rotation invariance can be obtained in different ways [Gau11]. The Fourier transform has been used to compute descriptors that is illumination and rotation invariant, and scale-invariant to a certain extent [Car02, Car03]. In order

to overcome dimensionality issues that may arise in a high-dimensional space, binary descriptors are introduced that are faster, but less precise, for example the Binary Robust Invariant Scalable Keypoints (BRISK) descriptor [Leu11] and Fast Retina Keypoint (FREAK) descriptor [Ala12].

However, these descriptors with strict invariance properties are not suitable for handwritten document representation. This is mainly because the invariance property renders them more sensitive to noise in a degraded document, as has been carefully studied in [Zag17, Ley09].

A method for searching handwritten Arabic documents based on a set of binary shape features is presented in [Sri05], where a correlation distance based matching technique has been employed. However, it was argued by [Gau11] that the features that are dependent on word shape characteristics are not effective in dealing with multi-writer document collections. Instead, the texture information in a spatial context is considered more reliable than the shape information, as suggested in [Gau11, Lla12].

In [Ley07], the image zones representing the most informative parts in a document image are detected based on the gradient orientation computed by taking convolution of the image with the first and second derivatives of the Gaussian kernel. However, this method was found to be inefficient for short words with less than four characters, and therefore an improved version was proposed in [Ley09]. The feature matching algorithm in [Ley09] was found to be very sensitive to variations in handwriting and font sizes, and the overall matching process was too slow for processing large datasets. An interesting block-based document image descriptor was presented by [Gat09] where the query image was scaled and rotated to produce different word instances, and for each instance, a different set of feature vectors was computed. However, several versions of queries generated significant amount of noise in the final merging state, rendering the method inefficient for handling large writing style and font variations.

Inspired by Bag-of-Visual Words (BoVW) model, a patch-based framework that uses SIFT for local feature representation was presented in [Rus11]. The codebook generation step of BoVW model is expensive, and this method is also found to be unsuitable for handling query font size and handwriting variations [Zag17]. The performance of popular word descriptors in a BoVW context was evaluated in [Lla12], and it was suggested that the statistical BoVW approach generates the best result, but with significant increase in overhead in terms of memory requirements to store the descriptors.

The winning algorithm, [Kov14], for segmentation-free track of ICFHR 2014 Handwritten Keyword Spotting Competition [Pra14], employed HoG and Local Binary Patterns (LBP) descriptors, and the word retrieval is performed using the nearest neighbour search, followed by a simple oppression of extra overlapping candidates. The work by [Zag17] outperformed the winning algorithms from ICFHR 2014 Handwritten Keyword Spotting Competition [Pra14], and ICDAR2015 Handwritten Keyword Spotting Competition [Pui15]. They proposed a new approach towards handwritten word spotting, where the spatial information representing the current location of a feature point is taken into account, and is based on the texture information. However, it is unclear how well this method performs in challenging cases where a a word shares several letters with other different words. The RLF descriptor based method proposed herewith handles this issue by dividing a word into several parts (depending upon the size of the word) to eliminate false-positives, and perform reliable keypoint-based feature matching.

The performance of different features for word spotting applications was evaluated using Dynamic Time Warping (DTW) [Rod08] and Hidden Markov Models (HMMs) [Rod09]. It was found that the local gradient histogram features outperform other geometrical or profile-based features. These methods generally match features from evenly distributed locations over normalized words where no nearest neighbor search is necessary. This is because each point in a word has its corresponding point in some other word located in the very same position. Recently, a method based on feature matching of keypoints derived from the words was proposed [Has16], which requires a nearest neighbor search. In this case, a relaxed descriptor is required that is not over-precise, since the handwritten words are not normalized. This is due to complex characteristic of handwritten words, unlike simple Optical character recognition (OCR) text. Handwritten words across different documents are similar, but not identical due to variability in writing styles.

In an endeavor to address the issues discussed above, this work proposes the RLF descriptor, which is tailor-made for handwritten words representation. The main highlights of this work are as follows: (a) a segmentation-free and training word spotting approach is studied; (b) the proposed method uses a combination of different keypoint detectors to capture different characteristics in a handwritten document, which consists of both lines, corners and blobs; (c) the RLF descriptor is designed, which is a fast and short-length feature vector of 32 dimensions with several advantages; (d) a simple preconditioner-based feature matching algorithm is presented. Advantages of RLF descriptor include faster word spotting (due to short length of feature vector), robustness to degradations, flexibility to be employed with existing feature detectors, efficient memory utilization, and no increase

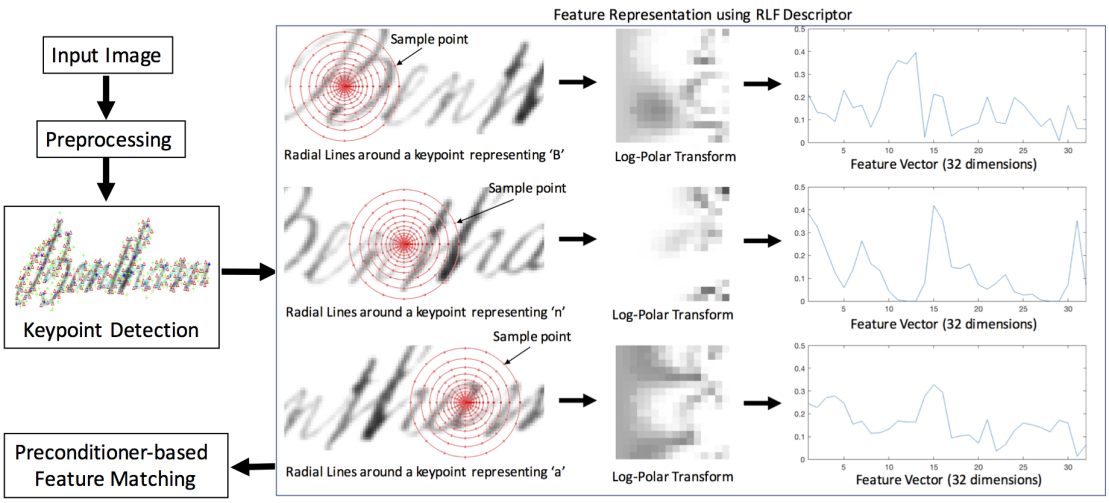


Figure 1: Radial Line Fourier (RLF) descriptor for feature representation in a word spotting framework. Each keypoint detected is represented using log-polar sampling scheme with 16 sampling points per ring. Each radial line, originating in the center and traversing each ring, is used to obtain a square (16 x 16) transformed image representation. In the next step, DFT is applied along each row (corresponding to radial lines) to compute the amplitude of a few elements for each row that constitute the feature vector. Finally, the feature vector generated is presented where x-axis denotes the feature vector length (i.e. 32), and y-axis denotes the amplitude of DFT.

in overhead for feature orientation estimation. The proposed methodology is discussed as follows.

3 METHODOLOGY

The pipeline of the word spotting framework is as follows. For an input document image, preprocessing is performed to remove background noise using two band-pass filtering approach [Vat17]. This is followed by keypoints detection, feature representation using RLF descriptor, and preconditioner-based feature matching. The framework of the proposed approach is pictorially described in Figure 1.

3.1 Preprocessing

Preprocessing is the initial step of the word spotting algorithm where the background noise is removed using a simple two band-pass filtering approach, as proposed in [Vat17]. A high frequency band-pass filter is used to separate the fine detailed text from the background, and a low frequency band-pass filter is used for masking and noise removal. The background removal is performed in such a way that the gray-level information crucial for the feature extraction is not affected. This allows the keypoint detector and the RLF descriptor to be more informative.

3.2 Keypoint Detection

To begin with, keypoints are detected for the document image and the query word. A combination of four different types of keypoint detectors is used to capture a variety of features that represent a handwritten document, and consists of lines, corners and blobs. Figure 2 presents the keypoint detectors used herein using an example image of a smoothed query word, *Bentham*. *Blue* * represents the Harris corner detector [Har88], *green* + represents the result of using the square of the Determinant of Hessian (DoH), which captures both dark and bright blobs, *red* Δ represents negative of DoH (-DoH) and finds the saddle points, and *cyan* + represents the result of an edge detector ($Assymetric^2$) [Has14b].

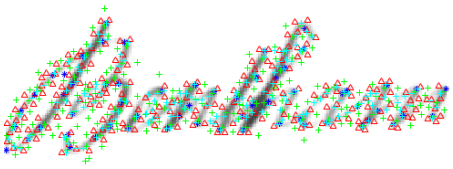


Figure 2: An example of a query word *Bentham* depicting four different types of keypoints. Blue * is a corner detector, green + finds the dark and bright blobs, red Δ finds the saddle points, and cyan + finds the edges of the text strokes.

3.3 Radial Line Fourier Descriptor

Radial Line Fourier (RLF) descriptor is a short-length feature vector of 32 dimensions, presented in this work for representation of handwritten words. RLF is inspired from a variant of Scale Invariant Descriptor (SID) [Kok08], known as SID-Rot [Tru13], and the idea is to perform log-polar sampling in a circular neighborhood around each keypoint. SID is a scale and rotation invariant descriptor, whereas SID-Rot is scale-invariant but rotation-sensitive descriptor. Typically, the Fourier transform can be applied over scales only to obtain a scale-invariant and rotation-dependent

descriptor, or a rotation-invariant and scale-sensitive descriptor. The Fourier transformation over scales render the SID-Rot to be rotation-sensitive, and the scale invariance is achieved by sampling over a large radius with a descriptor length of 3360. This method works well in representing natural scene images with scale changes and no rotations. However, strict invariance properties amplify noise in degraded document images [Zag17], and may lead to loss of useful information. Therefore, a relaxed feature descriptor, such as RLF, is required.

RLF descriptor computes a feature vector representation of an image feature, and is based on log-polar sampling followed by computing a few elements of the DFT along each radial line. It characterizes an image region as a whole using a single feature vector of fixed size, and no learning step is involved. Figure 1 presents the general framework of the RLF descriptor for feature representation in a word spotting pipeline, and discussed in detail as follows.

After the keypoints representing a document image have been detected, log-polar sampling is performed at each keypoint, where each radial line (going from the center, traversing each ring around the center along a line) is transformed into a square representation, as highlighted in Figure 1. The log-polar transform resampling resolution is set to 16 sample points per ring to obtain a square (16 x 16) transformed image. When sampling is done in a log-polar fashion, certain interpolation is required as the pixel coordinates are seldom in the center. One could for instance use a bilinear interpolation to achieve higher accuracy. In this work, sub-pixel sampling is computed using the Gaussian interpolation in a 3x3 neighborhood. In the next step, DFT is used to compute the amplitude of a few elements that constitute the feature vector.

The Fast Fourier Transform (FFT) performed efficiently in [Has14a] for creating descriptors that are relaxed. However, it was found to be impractical for high level applications with large amount of data [Has16]. This is because the FFT is rather slow in computations, such as computing the distance measures (i.e. phase correlation). In general, the FFT requires $O(N\,log(N))$ computations for a discrete series $f(n)$ with N elements. Therefore, this work improves and simplifies the computations needed to generate a faster feature representation, still benefiting from the advantages of the Fourier transform. We propose to use just a few elements from DFT of the sampled elements $f(n)$ along the radial line, and the computation required (using Euler's formula) is

$$\mathscr{F}[f(n)](k) = \sum_{n=0}^{N-1} f(n)\cos(2\pi nk/N) - i(f(n)\sin(2\pi nk/N)). \tag{1}$$

The value of k determines the frequency used to compute the Fourier element, where $k \in 0,2,4...$ Typically, noise in a document image has higher frequency as compared to the main text in the document image, therefore the second ($k = 2$) and third ($k = 4$) elements of the Fourier transform are selected to form the feature descriptor. DFT requires only $O(N)$ computations per element. Note that the Discrete Cosine (DC) component is obtained for $k = 0$ and is less informative. The trigonometric functions in the DFT do not have to be computed for each step, and the computation requires simple mathematical operations using the Chebyshev recurrence relation, same as the original Fourier Transform.

The RLF descriptor is thus constructed by computing the amplitude of a few elements of DFT:

$$|\mathscr{F}[f(n)](k)| = \sqrt{\Re(\mathscr{F}[f(n)](k))^2 + \Im(\mathscr{F}[f(n)](k))^2}. \tag{2}$$

The descriptor computation using only $k = 2$ suffices well for handwritten word representation under the test settings, and most importantly the descriptor is very short (length 32) with fast feature representation. However, experimentally it was found that by adding a second element for $k = 4$, the quality of the subsequent matching improved, even though the feature vector thus generated is twice as long. The advantage is that it makes it possible to sample in a smaller neighborhood, while still getting the same number of corresponding matches, with better accuracy. Nevertheless, adding a third element for $k = 6$ did not improve the accuracy significantly, and is found to be not worth the extra computational effort. This work uses RLF descriptor with length 32 for experimental analysis, taking into account the trade-off between computational cost and accuracy.

The RLF feature vector thus generated is presented in Figure 1, where x-axis denotes the feature vector length (i.e. 32 dimensions), and y-axis denote the frequency amplitudes of DFT. The advantages of the RLF descriptor are many-fold. The RLF descriptor computes a fast and short-length feature vector, to be able to perform quick feature matching in the nearest neighbor search. The RLF descriptor emphasizes on the pixels closer to the feature center, making it less sensitive to erroneous feature size estimation. It is resistant to high frequency changes, such as due to residuals from neighboring words, as it is based on the low frequency content in the local neighborhood. Nevertheless, it is insensitive to small differences in form and shape, as long as they are almost same, i.e. the low frequencies are sufficiently similar.

3.4 Feature Matching

A segmentation-free and training-free word spotting method based on the proposed RLF descriptor is studied herein. In general, no prior information is available about the potential word in the document that is to

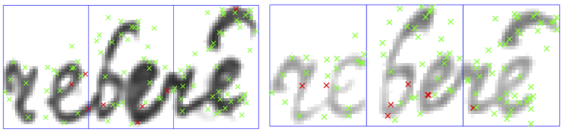


Figure 3: Matched points represented using the RLF descriptor for a sample word *reberé*. The matching keypoints (inliers) are in green, and the matches discarded (outliers) by the preconditioner are in red.

be matched with the query word. By using the RLF descriptor, the word matching problem is reduced to a much faster search problem. In this work, a simple preconditioner-based feature matching algorithm is employed.

To begin with, words are partitioned into several parts in order to avoid confusion between similar words and reduce false positives. This is to overcome the drawback of keypoint-based matching techniques [Zag17], where parts of the retrieved words may be very similar to some part of the query word, or where a word shares several letters with other different words, hence generate false positives. In the experiments, words are divided into several parts depending upon the length of the query word. For example, in Figure 3, a sample query word *reberé* and its corresponding retrieved word are divided into three parts, and the preconditioner-based matching is performed in the respective three different parts of both the words.

After the partitioning step, a nearest neighbor part-based search is performed in an optimal sliding window within the subgroups of the detected keypoints. The keypoint matching algorithm computes the extent of the the matching points in a word, and therefore is able to capture words that are partially outside the sliding window. Consequently, the matched points are removed from the set of points when a word is found, to avoid finding the same word again.

The resultant correspondences between the query word and the retrieved word in the sliding window obtained after a simple keypoint matching consists of many outliers and needs further refinement. A common approach is to use Random sample consensus (RANSAC) [Fis87] to learn transformations between the words. However, it is important to have a relaxed transformation instead, because the same word at different locations in a document can differ with small variations in font sizes, or even larger variations in a multi-writer scenario. Therefore, a deterministic preconditioner (inspired from [Has12a]) is used in this work that eliminates the need to use RANSAC and helps in removing the false matches. In [Has16], preconditioner had been used along with Putative Match Analysis (PUMA) [Has12b], which is found to be computationally expensive and increases overhead in computing false positives. To keep the matching algorithm simple yet effective, this work uses a matching algorithm that is solely based on the preconditioner.

Figure 4: Sample document images from the BH2M dataset [Fer14].

The preconditioner creates a cluster of corresponding matches in a two-dimensional space as positional vectors. This means that the correspondences between the query word and the retrieved word in the sliding window with same length and direction are potential inliers, that forms a two-dimensional cluster. However, the clusters are expected to be slightly scattered due to complex characteristics of words (e.g. words can differ in font and style), therefore the threshold must be relaxed or loosely set. The preconditioner finds the inliers efficiently and removes the outliers with fast computation speed. Figure 3 represents the matched points obtained from the proposed method, where the matching keypoints or inliers are highlighted in green and the outliers discarded by the preconditioner are in red. The preconditioner-based matching efficiently captures complex variations in handwriting by estimating the core text dimensions on-the-fly. The effectiveness of the proposed method has been experimentally demonstrated in the next section.

4 EXPERIMENTAL RESULTS

This section describes the datasets used in the experiments, and empirically evaluates the proposed method.

4.1 Datasets

For experimental analysis, the Barcelona Historical Handwritten Marriages dataset, and the Bentham dataset in two variants are taken into account. The former is heavily degraded, posing challenges for the word spotter, and the latter in both variants demonstrate multi-writer handwriting variations to a certain extent, along with document degradations. The datasets are discussed as follows:

- *Barcelona Historical Handwritten Marriages Dataset (BH2M)*: It consists of historical handwritten marriage records stored in the archives of Barcelona cathedral, written between 1617 and 1619 by a single writer in old Catalan. Figure 4 presents sample document images from the BH2M dataset. The reader is referred to [Fer14] for a deeper understanding of the dataset.

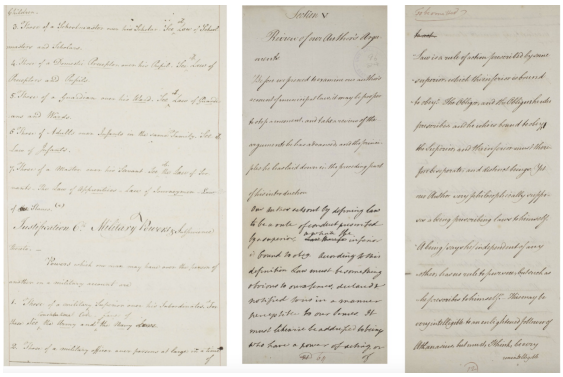


Figure 5: Sample document images from the Bentham dataset used in ICFHR 2014 Handwritten Keyword Spotting Competition [Pra14].

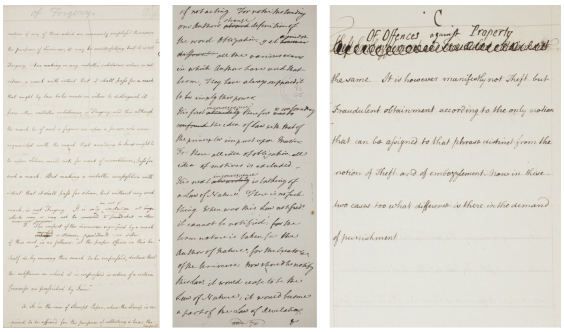


Figure 6: Sample document images from the Bentham dataset used in ICDAR 2015 Handwritten Keyword Spotting Competition [Pui15].

- *Bentham Dataset*: It consists of handwritten document pages from the Bentham collection, which have been prepared in the *tranScriptorium* project. The Bentham collection consists of manuscripts on law and moral philosophy handwritten by Jeremy Bentham (1748-1832) over a period of 60 years, and some handwritten documents from his secretarial staff. This dataset in first variant was used in ICFHR 2014 Handwritten Keyword Spotting Competition [Pra14], and the second variant in ICDAR 2015 Handwritten Keyword Spotting Competition [Pui15]. Figure 5 and 6 present sample images from the Bentham dataset used in ICFHR 2014 and ICDAR 2015 competitions, respectively. For the experiments, all pages from both variants of Bentham dataset used in the competitions are employed, which have been written by different authors in different styles, font-sizes, and contains crossed-out words.

4.2 Results

The performance of the proposed method is empirically evaluated against the winning algorithms of ICFHR 2014 Handwritten Keyword Spotting Competition [Pra14], and ICDAR 2015 Handwritten Keyword Spotting Competition [Pui15], along with the other state-of-the-art methods such as [Zag17, Ley09]. The evaluation measure used is the classic mean Average Precision (mAP) metric popularly used in document word spotting. In general, the retrieved regions of all the document pages are combined and re-ranked according to the score obtained. If a region overlaps more than 50% of the area of the ground truth corpora, it is classified as a positive region. The Precision and Recall values are first computed, and since a single value is preferable for comparison across different methods, the mAP of each method is calculated as the final result. A higher value of mAP is more desirable.

Method	**mAP**
[Alm12b]	0.513
[Zag17]	0.530
Proposed method	**0.783**

Table 1: Experimental results for BH2M dataset.

Method	**mAP**
[Ley09]	0.221
[How13]	0.409
[Kov14]	0.423
[Zag17]	0.517
Proposed method	**0.490**

Table 2: Experimental results for Bentham dataset used in ICFHR 2014 competition.

Method	**mAP**
PRG, TU Dortmund	0.293
CVC, Spain	0.116
[Zag17]	0.326
Proposed method	**0.786**

Table 3: Experimental results for Bentham dataset used in ICDAR 2015 competition.

Tables 1-3 present the segmentation-free handwritten word spotting results for various methods. In Table 1, the performance of the proposed method is evaluated on the BH2M dataset against the methods proposed in [Alm12b] and [Zag17]. The method proposed in [Alm12b] is based on exemplar-SVM framework for word spotting, and the method presented in [Zag17] is based on Document-oriented Local Features (DoLF). It is observed from Table 1 that the proposed method achieves higher mAP as compared to [Alm12b] and [Zag17]. This is mainly because the performance of [Alm12b] and [Zag17] is found to be weaker for challenging cases where a a word shares several letters with other different words. Typically, a higher mAP is achieved when search is performed on a long query word (e.g. *habitant*), as there is less possibility of finding the query word as part of other similar word. However, in an ideal scenario it is highly possible for a query word to share several characters with

other words, even with a longer word. A simple example of a query word from the BH2M dataset is *donsella*, where some characters are common with query words *fill* and *filla*. A much challenging case observed is the sequence of overlapping characters in the query words *fill* and *filla*, where *fill* is retrieved while searching for *filla*. The proposed method handles this effectively by dividing a word into several parts depending upon the length of the word, and then perform part-based keypoint matching. This simple approach reduces the false-positives by a significant margin, as is evident from the results in Table 1.

Table 2 presents the results obtained using different methods on the Bentham dataset from ICFHR 2014 Handwritten Keyword Spotting Competition [Pra14]. The performance of the proposed method is empirically evaluated against the state-of-the-art methods such as [Ley09], [How13], [Kov14] (i.e. winner of ICFHR 2014 competition), and [Zag17]. It is observed from Table 2 that the proposed method achieves higher mAP as compared to [Ley09], [How13] and [Kov14], and performs comparable against [Zag17] for all test images under the experimental settings. This is mainly because the relaxed nature of RLF allows it to capture more details in a degraded document image as compared to descriptors with stricter invariance properties that render them more sensitive to noise. This is important as the same query word at different locations in a document can differ with small variations in font sizes, or even larger variations in a multi-writer scenario. However, even though the proposed approach is observed to perform significantly in comparison with other methods discussed in Table 2, a mAP of 0.490 suggests further investigation. It is observed that the document images in the Bentham dataset from ICFHR 2014 competition consists of handwritten text from two or more authors, where the core text size in a document page differs across different locations in the same document page. This pose challenges for the algorithm in estimating the average core text size for each document page, as the normalization of text size might result in loss of information. The authors aim at investigating this issue further and working towards the improvement of the proposed algorithm as future work.

Table 3 evaluates the performance of the proposed method on the second variant of Bentham dataset introduced in the ICDAR 2015 Handwritten Keyword Spotting Competition [Pui15]. Unlike the first variant of the Bentham dataset discussed above, this dataset does not significantly suffer from the problem of highly variable core text size across a document page. This is evident from the higher mAP value achieved in Table 3. It is observed that the proposed method achieves higher accuracy in comparison with the winner algorithms from the competition, as well as a recent method [Zag17]. The RLF descriptor with relaxed feature description takes into account the handwriting variations to a considerable extent, and the standard core text size is estimated for each document page without significant errors.

Method	**mAP**
SIFT [Low04]	0.115
SURF [Bay08]	0.106
BRISK [Leu11]	0.035
ORB [Rub11]	0.098
KAZE [Alc12]	0.283
DoLF [Zag17]	0.517
Proposed RLF	**0.490**

Table 4: Performance evaluation of feature representation methods on Bentham dataset used in ICFHR 2014 competition.

Method	**mAP**
HoG [Alm12a]	0.584
Loci [Fer11]	0.419
Graph-based [Wan14]	0.565
FFT [Has16]	0.771
Proposed RLF	**0.783**

Table 5: Performance evaluation of feature representation methods on BH2M dataset.

In order to highlight the importance of the proposed RLF descriptor, a comparison is done with the existing feature representation methods such as SIFT [Low04], SURF [Bay08], BRISK [Leu11], Oriented FAST and Rotated BRIEF (ORB) [Rub11], KAZE [Alc12], DoLF [Zag17], HoG [Alm12a], Loci features [Fer11], graph-based [Wan14] and FFT [Has16]. Table 4 presents the experimental results to evaluate the feature representation methods used in the word spotting framework for the Bentham dataset (ICFHR 2014 competition), as an example. This is with reference to the mAP values published in a recent work [Zag17] under the given experimental set up. It is observed from Table 4 that the RLF descriptor achieves higher mAP in comparison with SIFT, SURF, BRISK, ORB and KAZE, and performs comparable against DoLF. Table 5 validates the performance of the RLF descriptor with respect to the BH2M dataset, and the experiments are performed under the same test settings where the matching algorithm is same for all feature representation methods. The RLF descriptor performs significantly in comparison with other methods, because of the advantages inherited from relaxed feature representation and efficient algorithm design. Nevertheless, with reference to the three historical handwritten datasets used in the experiments, the proposed method is observed to be most consistent and stable with high mAP.

5 CONCLUSION

This paper presented a fast and robust Radial Line Fourier descriptor, with a short feature vector of 32 dimensions, for segmentation-free and training-free handwritten word spotting. A simple preconditioner-based feature matching algorithm is employed, and the experimental results on a variety of historical document images from well-known datasets demonstrate the effectiveness of the proposed method. Under the experimental settings, the proposed RLF descriptor based method outperformed the state-of-the-art methods, including the winners of the popular keyword spotting competitions. As future work, the ideas presented herein will be scaled to aid word feature representation for heavily degraded archival databases with improvements using query expansion.

6 ACKNOWLEDGMENTS

This work was supported by the Swedish strategic research programme eSSENCE and the Riksbankens Jubileumsfond (Dnr NHS14-2068:1). The computations were performed on resources provided by SNIC through Uppsala Multidisciplinary Center for Advanced Computational Science (UPPMAX) under Project SNIC 2017/7-97.

7 REFERENCES

[Ala12] Alahi, A., Ortiz, R., Vandergheynst, P. Freak: Fast retina keypoint, in IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 510-517, 2012.

[Alc12] Alcantarilla, P.F., Bartoli, A., Davison, A.J. Kaze features, in European Conference on Computer Vision, Springer, pp. 214-227, 2012.

[Alm12a] Almazán, J., Fernández, D., Fornés, A., Lladós, J., Valveny, E. A coarse-to-fine approach for handwritten word spotting in large scale historical documents collection, in IEEE International Conference on Frontiers in Handwriting Recognition, pp. 455-460, 2012.

[Alm12b] Almazá, J., Gordo, A., Fornés, A., Valveny, E. Efficient exemplar word spotting, 2012.

[Bay08] Baya, H., Essa, A., Tuytelaarsb, T., Van Goola, L. Speeded-up robust features (surf). Computer Vision and Image Understanding, 110(3), 346-359, 2008.

[Car02] Carneiro, G., Jepson, A.D. Phase-based local features, in European Conference on Computer Vision, Springer. pp. 282-296, 2002.

[Car03] Carneiro, G., Jepson, A.D. Multi-scale phase-based local features, in IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 736-743, 2003.

[Dal05] Dalal, N., Triggs, B. Histograms of oriented gradients for human detection, in IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 886-893, 2005.

[Fer11] Fernández, D., Lladós, J., Fornés, A. Handwritten word spotting in old manuscript images using a pseudo-structural descriptor organized in a hash structure, in Iberian Conference on Pattern Recognition and Image Analysis, Springer, pp. 628-635, 2011.

[Fer14] Fernández-Mota, D., Almazán, J., Cirera, N., Fornés, A., Lladós, J. Bh2m: The barcelona historical, handwritten marriages database, in 22nd IEEE International Conference on Pattern Recognition, pp. 256-261, 2014.

[Fis87] Fischler, M.A., Bolles, R.C. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. Readings in computer vision, 726-740, 1987.

[Gat09] Gatos, B., Pratikakis, I. Segmentation-free word spotting in historical printed documents, in 10th IEEE International Conference on Document Analysis and Recognition, pp. 271-275, 2009.

[Gau11] Gauglitz, S., Höllerer, T., Turk, M. Evaluation of interest point detectors and feature descriptors for visual tracking. International Journal of Computer Vision, 94, 335-360, 2011.

[Gio17] Giotis, A.P., Sfikas, G., Gatos, B., Nikou, C. A survey of document image word spotting techniques. Pattern Recognition, 68, 310-332, 2017.

[Gir14] Girshick, R., Donahue, J., Darrell, T., Malik, J. Rich feature hierarchies for accurate object detection and semantic segmentation, in IEEE conference on Computer Vision and Pattern Recognition, pp. 580-587, 2014.

[Har88] Harris, C., Stephens, M. A combined corner and edge detector, in fourth alvey vision conference, pp. 147-151, 1988.

[Has12a] Hast, A., Marchetti, A. An efficient preconditioner and a modified ransac for fast and robust feature matching, in International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG'2012), pp. 11-18, 2012.

[Has12b] Hast, A., Marchetti, A. Putative match analysis: a repeatable alternative to ransac for matching of aerial images, in International Conference on Computer Vision Theory and Applications, pp. 341-344, 2012.

[Has14a] Hast, A., 2014. Robust and invariant phase based local feature matching, in 22nd IEEE International Conference on Pattern Recognition, pp. 809-814, 2014.

[Has14b] Hast, A., Marchetti, A. Invariant interest point detection based on variations of the spinor tensor, in 22nd International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG'2014), pp. 49-56, 2014.

[Has16] Hast, A., Fornés, A. A segmentation-free handwritten word spotting approach by relaxed feature matching, in 12th IAPR Workshop on Document Analysis Systems, pp. 150-155, 2016.

[How13] Howe, N.R. Part-structured inkball models for one-shot handwritten word spotting, in 12th IEEE International Conference on Document Analysis and Recognition, pp. 582-586, 2013.

[Kok08] Kokkinos, I., Yuille, A. Scale invariance without scale selection, in IEEE Conference on Computer Vision and Pattern Recognition, pp. 1-8, 2008.

[Kov14] Kovalchuk, A., Wolf, L., Dershowitz, N. A simple and fast word spotting method, in 14th IEEE International Conference on Frontiers in Handwriting Recognition, pp. 3-8, 2014.

[Leu11] Leutenegger, S., Chli, M., Siegwart, R.Y. Brisk: Binary robust invariant scalable keypoints, in IEEE International Conference on Computer Vision, pp. 2548-2555, 2011.

[Ley07] Leydier, Y., Lebourgeois, F., Emptoz, H. Text search for medieval manuscript images, Pattern Recognition, 40, 3552-3567, 2007.

[Ley09] Leydier, Y., Ouji, A., LeBourgeois, F., Emptoz, H. Towards an omnilingual word retrieval system for ancient manuscripts, Pattern Recognition, 42, 2089-2105, 2009.

[Lla12] Lladós, J., Rusinól, M., Fornés, A., Fernández, D., Dutta, A. On the influence of word representations for handwritten word spotting in historical documents, International Journal of Pattern Recognition and Artificial Intelligence, 26, 1263002, 2012.

[Low04] Lowe, D.G. Distinctive image features from scale-invariant keypoints, International Journal of Computer Vision, 60, 91-110, 2004.

[Mat02] Matas, J., Chum, O., Urban, M., Pajdla, T. Robust wide baseline stereo from maximally stable extremal regions, in British Machine Vision Conference, pp. 36.1-36.10, 2002.

[Pra14] Pratikakis, I., Zagoris, K., Gatos, B., Louloudis, G., Stamatopoulos, N. ICFHR 2014 competition on handwritten keyword spotting (h-kws 2014), in 14th IEEE International Conference on Frontiers in Handwriting Recognition, pp. 814-819, 2014.

[Pui15] Puigcerver, J., Toselli, A.H., Vidal, E. ICDAR2015 competition on keyword spotting for handwritten documents, in 13th IEEE International Conference on Document Analysis and Recognition, pp. 1176-1180, 2015.

[Rod08] Rodriguez, J.A., Perronnin, F. Local gradient histogram features for word spotting in unconstrained handwritten documents, in 1st International Conference on Frontiers in Handwriting Recognition, pp. 7-12, 2008.

[Rod09] Rodríguez-Serrano, J.A., Perronnin, F. Handwritten word-spotting using hidden markov models and universal vocabularies, Pattern Recognition, 42(9), 2106-2116, 2009.

[Rub11] Rublee, E., Rabaud, V., Konolige, K., Bradski, G. Orb: An efficient alternative to sift or surf, in IEEE International Conference on Computer Vision, pp. 2564-2571, 2011.

[Rus11] Rusiñol, M., Aldavert, D., Toledo, R., Lladós, J. Browsing heterogeneous document collections by a segmentation-free word spotting method, in IEEE International Conference on Document Analysis and Recognition, pp. 63-67, 2011.

[Shi94] Shi, J., et al. Good features to track, in IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 593-600, 1994.

[Sri05] Srihari, S., Srinivasan, H., Babu, P., Bhole, C. Handwritten arabic word spotting using the cedarabic document analysis system, in Symposium on Document Image Understanding Technology, pp. 123-132, 2005.

[Tru13] Trulls, E., Kokkinos, I., Sanfeliu, A., Moreno-Noguer, F. Dense segmentation-aware descriptors, in IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 2890-2897, 2013.

[Vat17] Vats, E., Hast, A., Singh, P., 2017. Automatic document image binarization using bayesian optimization, in 4th International Workshop on Historical Document Imaging and Processing, pp. 89-94, 2017.

[Wan14] Wang, P., Eglin, V., Garcia, C., Largeron, C., Lladós, J., Fornés, A. A coarse-to-fine word spotting approach for historical handwritten documents based on graph embedding and graph edit distance, in 22nd IEEE International Conference on Pattern Recognition, pp. 3074-3079, 2014.

[Zag17] Zagoris, K., Pratikakis, I., Gatos, B. Unsupervised word spotting in historical handwritten document images using document-oriented local features, IEEE Transactions on Image Processing, 26(8), 4032-4041, 2017.

Comparing the performance and accuracy of algorithms applied to tattoos images identification

Agnus Azevedo Horta
DCA, FEEC, University of Campinas
Av. Albert Einstein - 400
Brazil 13083-852, Campinas, SP
agnus@dca.fee.unicamp.br

Léo Pini Magalhães
DCA, FEEC, University of Campinas
Av. Albert Einstein - 400
Brazil 13083-852, Campinas, SP
leopini@fee.unicamp.br

ABSTRACT

This article presents results of the simulation of SIFT based algorithms in the context of the identification of tattoos. The algorithms studied are the SIFT - Scale Invariant Feature Transform, ASIFT - Affine SIFT, BOV - Bag of Visual Words and FV - Fisher Vector. The use of the OPF - Optimum-Path Forest and SVM - Support Vector Machine classifiers is exploited in conjunction with SIFT and ASIFT algorithms as well as BOV and FV. The present study uses the National Institute of Standards and Technology (NIST) Tatt-C dataset in a reduced and complete version. This work uses runtime and accuracy to compare the results of the simulations.

Keywords

Tattoo identification, SIFT, ASIFT, Bag of Visual Words, Fisher Vectors, Optimum-path forest classifier, Support Vector Machines, Soft biometric, Content-based image retrieval.

1 INTRODUCTION

The area of tattooing is a comprehensive area of activities that deal with the identification and detection of tattoos, search for similarities between tattoos, search for regions of interest in tattoos, treatment of tattoos on various materials/surfaces among their important applications. It is an area, for example, that supports important forensic activities related to law enforcement for offenders and victim support.

This paper is dedicated to the area of tattoo identification [1]. Although many important tattoo-related activities have been and are in progress in the scientific community, we believe that the comparison among algorithms we bring in this paper can be very useful in the choice of solutions for many applications in tattoo identification.

Most approaches that exist in the literature to solve the problem of identification of tattoos are strictly based on SIFT (Scale Invariant Feature Transform) and ASIFT (Affine SIFT) algorithms. This paper explores the idea to use both algorithms to provide reference results and to support other techniques - BOV (Bag of Visual words), FV (Fisher Vectors) and classifiers - to solve the tattoo identification problem. Thus we combined SIFT and ASIFT with BOV and FV approaches and additionally we applied these algorithms with the classifiers SVM (Support Vector Machine) and OPF (Optimum-Path Forest) to the tattoo identification problem. Our previous experience, in another area of application, showed a very encouraging performance of the OPF algorithm, compared to SVM, in that case with the descriptor BIC (Border/Interior pixel Classification) [DSFM11].

Section 2 gives a brief description of all simulated methods. These methods, as commented above, are the SIFT and ASIFT algorithms, the combination of both algorithms with the use of Bag of Visual words, Fisher Vectors, matching algorithms, and the use of the OPF and SVM classifiers. After that, Section 3 presents a comparison based on the accuracy and performance of the simulated approaches. For this comparison, the NIST Tatt-C dataset [NG15] is used in a reduced version and in its full version as explained in the section.

[1] "operational use-cases defined by the NIST challenge – 2015/2016: 1. Tattoo Similarity – matching visually similar or related tattoos from different subjects; 2. Tattoo Identification - matching different instances of the same tattoo image from the same subject over time; 3. Region of Interest - matching a small region of interest that is contained in a larger image; 4. Mixed Media - matching visually similar or related tattoos using different types of images (e.g. sketches, scanned print, computer graphics, or natural images); 5. Tattoo Detection - detecting whether an image contains a tattoo or not." [NG15]

https://doi.org/10.24132/JWSCG.2018.26.1.5

The final section reviews the results of the simulations and discusses the next developments.

2 RELATED WORKS

Tattoo is already being used in identification processes due to its soft biometric characteristic. Although several studies have already been developed in this area, this is still a developing area. For example, there is a small number of available image databases for the development and evaluation of tattoos applications. The NIST National Institute of Standards and Technology has developed the Tatt-C dataset, an important and well-established example of such a set of tattoo images. [NG15]

Figure 1: Examples of tattoos images from the site The Commons (`https://www.flickr.com/commons`).

Figure 1 shows images that are similar to those used in this work. Due to copyright restrictions, we do not present images belonging to the NIST Tatt-C dataset used in this paper.

Another important point is related to the search for algorithms to identify tattoos, which should have a good relationship between runtime and accuracy. These algorithms should treat the biometric characteristics of tattoos associating the attributes of low level of abstraction that describe the tattoo. This is the point that we address in this article.

One of the first proposals to replace human beings in the assignment of labels for classes of images in the process of identification of tattoos is of 2007([JLJ07]). The proposal was based on the use of attributes of low-level of abstraction - color, shape, texture - to characterize tattoos and on the use of a simple relevance feedback strategy.

Currently, most approaches to tattoo treatment are based on the use of the SIFT descriptor [LOW99] and extensions such as affine SIFT (ASIFT) ([YM11]) exploring the invariance of these methods for scaling and rotation of the image, robustness to distortions related to changes in illumination and, for ASIFT, its property of complete affine invariance. In both approaches, the attributes of an image are represented by the key points of the image.

Lee et al. [LJJ08] describe a content based image retrieval (CBIR) system for combining and retrieving tattoo images. Subsequently, Lee et al. [LJJT12] still using SIFT key points proposes more robust similarity measures that used along with associated metadata improves performance in image retrieval.

The use of "pure" SIFT for the identification of tattoos can still be found in recent works such as Han et al. [HJ13], Marcetic et al. [MRSP14], and in the NIST challenge results Ngan et al.[NQG16].

The ASIFT algorithm is defined by considering a geometric interpretation of the finite decomposition of an image, where the angles corresponding to the latitude (ϕ) and longitude (θ) are gradually changed to generate a set of sample views of the initial image, in which SIFT is applied (see Figure 2). ASIFT can be used in the same applications of SIFT if time is not a strict restriction. The Section 3 will present the simulation results of both algorithms.

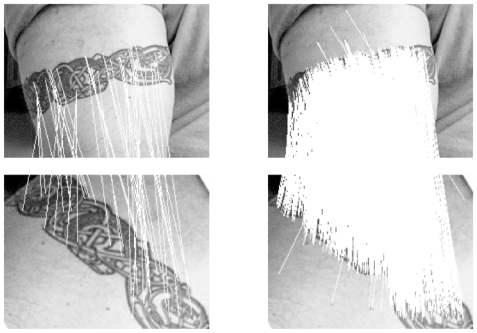


Figure 2: Images from the demo available at http://demo.ipol.im/demo/my_affine_sift/. Images showing the correspondence between the key points obtained by applying the SIFT (left) and ASIFT (right) in an original image and in its rotated image. The demo reports for the image at right 22938 ASIFT key points (top image) and 21577 ASIFT key points (bottom image); the key points computation was accomplished in 5 seconds, and 1482 matchings were identified. For the left image using SIFT 40 matchings were obtained and the key points matching was accomplished in 2 seconds.

Manger [Man12] also explored the categorization of tattoo images through an analogy with learning methods based on the categorization of text using the concept of a bag of words. In this analogy, one of the parameters is the size of a visual dictionary. SIFT is used to extract tattoo image characteristics that will be grouped using K-means. The model was inspired by the BOW model for text proposed by Sivic and Zisserman [SZ03] and extended to images by Csurka et al. [CDF04]. The original Manger modeling was discussed and improved in [MHW16], including an additional index based on Fisher vector coding. Fisher Vector (FV) derived from Fisher Kernel method [JH99]

encodes the visual vocabulary using Gaussian Mixture Models (GMM) and adds information related to local feature descriptors.

Regarding the classification process of images, the classifiers Support Vector Machines (SVM) and Optimum-Path Forest (OPF) are algorithms explored in this work. We use the libraries LIBSVM and LIBLINEAR for SVM and LIBOPF for OPF. LIBSVM Chang and Lin [CL11] implements algorithms for training and testing for SVM and supports several SVM formulations for classification, regression and distribution estimation. LIBLINEAR [FCH08] is an open-source library for large-scale linear classification. It supports logistic regression and linear SVM.

Perronnin and Dance [PD07] applied Kernels Fisher in visual vocabulary using nonlinear support machines (SVM) in the classification process. Schneider and Tuytelaars [ST14] successfully used FV together with the SVM classifier in sketches classification, surpassing the performance of existing techniques, a concept that can be applied to the recovery of tattoo images based on sketches. The present paper explores FV in tattoo identification.

OPF – Optimum-Path Forest classifier proposed by Papa et al. [PFS09] is a supervised classification method that represents each data class by its optimal path tree whose root nodes are called prototypes. The training samples correspond to the nodes of a complete graph whose arcs are weighted by the distances between its nodes.

The Section 3 will explore the combination of algorithms and classifiers using SIFT and ASIFT as base elements. The results will be compared considering accuracy and runtime.

3 RESULTS AND COMMENTS

The results of the simulations presented in this section were obtained by programs executed in a 4GB and 4-core virtual machine using a CPU-i7 with 8GB of memory, 4-core and 8 threads.

The simulations used the NIST Tatt-C dataset [NG15]; two datasets were generated, one for training processing and one for test processing. Each of the two sets was organized in five subsets - folds - to allow mean values across a five run-test, using the same procedure followed by Ngan et al. [NQG16].

In order to better understand and tunning the algorithms, the first set of simulations was performed using a "reduced" set of the Tatt-C dataset composed of, using the terminology of the NIST report, all #probes images (157 true images of interest), all #mates images (215 related images to probe-images) and none (zero) background images. In this way, the folds were defined for training containing approximately 302 images (157 minus 30 true images of interest added to 215 minus 40 images). For test processing, about 30 images were used (different true images for each fold). The results of these simulations are depicted in Table 1 and in Figures 3 to 5 always presenting mean and standard deviation of five rounds.

Considering the algorithms SIFT and ASIFT (lines 1 and 2 of Table 1) the process of searching for an image involves two basic steps, one denominated training which is the step in which the image database is processed and the image descriptor(s) is(are) generated, and the step denominated test that corresponds to the step where one or more images are displayed and the query is performed based on a match function. In the SIFT method, one aspect that influences in the results is the key point matching algorithm. In the experiments, we utilized the Knn Brute Force method of the OpenCV for both algorithms SIFT and ASIFT.

After the generation of the descriptors using SIFT or ASIFT we performed for the training phase the codebook generation for BOV based on KMeans and image coding, and for FV using the Gaussian Mixture Model (GMM) and image coding. Following, for the test phase, the image coding and rank generation (Figures 3 to 5) were performed for both methods. Numbers 75 to 1000 express the size of the codebook for BOV method and numbers 5 and 10 express the number of Gaussians for the FV method. Lines 3 to 12 depict the runtime for the different methods.

We have adopted in this paper the same definition of accuracy as NIST [NQG16]. Thus, accuracy is the amount of correctly identified images divided by the total number of probe images of the current fold.

The execution of the BOV for the reduced Tatt-C dataset with different codebook sizes showed us that for the BOV_SIFT the accuracy expressed by the Cumulative Matching Characteristic or CMC curve decreases with the increase of the codebook (Figure 3). In this way, the BOV_SIFT_75 has the best CMC curve compared to BOV_SIFT_500 and _1000. For ASIFT, a codebook of size 500 or 1000 resulted in almost the same CMC curve, overcoming codebooks with fewer elements (see Figure 4). Figure 5 showed us FV_SIFT_5 and _10 with similar behavior, and FV_ASIFT_5 with the best CMC for all FV.

In the next step, see Table 2, we applied classifiers SVM and OPF to the best algorithms - BOV_SIFT_75, BOV_ASIFT _500, FV_SIFT_5, FV_ASIFT_5 - obtained in the last step. The training and test phases show that the OPF classifier presents the best runtime and accuracy values compared to SVM. For training considering mean values, the best result for runtime was obtained for OPF + BOV_SIFT_75, followed by OPF + FV_ASIFT_5, OPF + FV_SIFT_5 and OPF + BOV_ASIFT_500. For test phase the

	Method	training (all images) mean	training (all images) std. deviation	test (one image) mean	test (one image) std. deviation
1	SIFT	207 *	– – –	25.56	1.63
2	ASIFT	200 *	– – –	536.51	38.87
3	BOV_SIFT_75	68.33	1.01	0.18	0.02
4	BOV_SIFT_500	104.24	2.64	0.30	0.03
5	BOV_SIFT_1000	165.15	3.56	0.41	0.04
6	BOV_ASIFT_75	339.27	5.48	0.78	0.05
7	BOV_ASIFT_500	659.64	13.17	1.34	0.06
8	BOV_ASIFT_1000	620.82	18.88	1.84	0.13
9	FV_SIFT_5	470.20	69.03	0.39	0.03
10	FV_SIFT_10	1,520.38	252.32	0.62	0.07
11	FV_ASIFT_5	2,855.31	768.71	1.85	0.30
12	FV_ASIFT_10	7,749.99	980.99	3.37	0.34

* single execution.

Table 1: Runtime in *seconds* for training and test phases using the reduced (no background images) Tatt-C dataset.

best result for runtime was obtained by OPF + BOV_SIFT_75, followed by OPF + FV_ASIFT_5, OPF + BOV_ASIFT_500 and OPF + FV_SIFT_5. Now, taking into account accuracy, again OPF presented better results than SVM. Considering accuracy mean values, OPF + BOV_ASIFT_500 presented the best result, followed by OPF + FV_ASIFT_5 and FV_SIFT_5, and BOV_SIFT_75.

Considering now the results provided by Table 1 in addition to the results discussed in the last paragraph (Table 2) we elected the algorithms BOV_SIFT_75 and FV_SIFT_5 to apply in the complete Tatt-C dataset in addition to the SIFT used as reference.

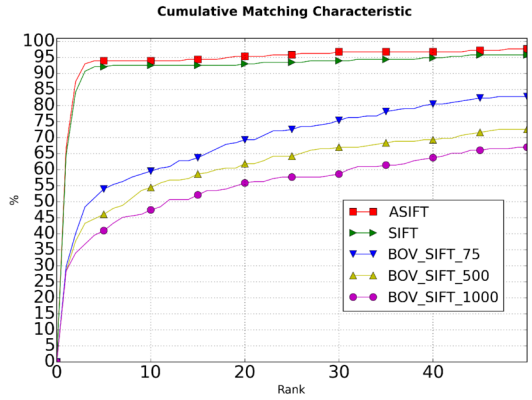


Figure 3: Cumulative Matching Characteristic for BOV_SIFT using the reduced Tatt-C dataset.

BOV_SIFT_75 and FV_SIFT_05 were chosen by the same reasoning of less runtime comparing with similar approaches (smaller dictionary and lower Gaussian number) (Tables 1 and 2) and comparable accuracy to the ASIFT approaches (Table 2). The use of the reduced dataset provided behavioral expectations of all algorithms addressed and, in this way, helped us to reduce the universe of simulation for the algorithms of interest to the complete dataset.

The second set of simulations was performed using the Tatt-C dataset composed by all #probes images (157 true images of interest), all #mates images (215 related images to probe-images) and background images (used to confuse the identification) as explained next. At first, a simulation was performed using all (100%) background images. In this way the folds were of approximate size of 4674 images (40 images of #mates images related to the 30 of #probes images, 215 minus 40 of #mates images, 157 minus 30 of #probes images and all 4332 background images) for training and

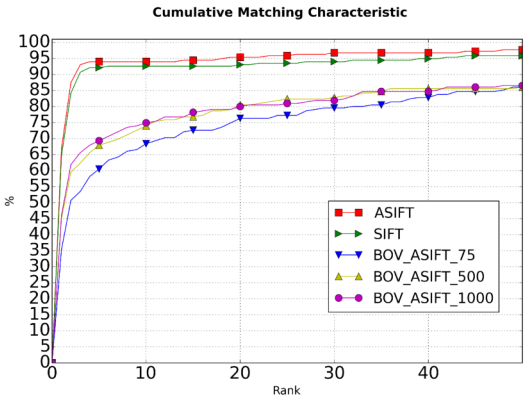


Figure 4: Cumulative Matching Characteristic for BOV_ASIFT using the reduced Tatt-C dataset.

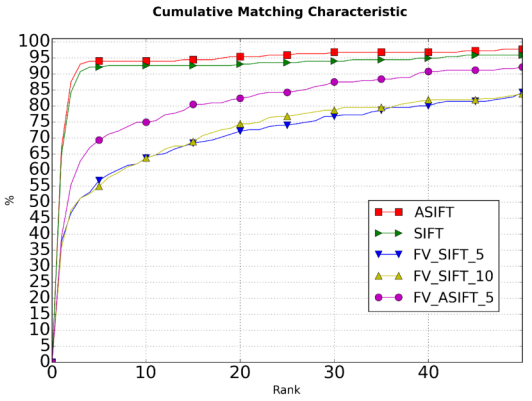


Figure 5: Cumulative Matching Characteristic for FV_SIFT and FV_ASIFT using the reduced Tatt-C dataset.

	OPF						SVM					
	Training time (s) (all images)		Testing time (s) (one image)		Accuracy (%)		Training time (s) (all images)		Testing time (s) (one image)		Accuracy (%)	
Method	**mean**	**std dev**	**mean**	**std dev**	**mean**	**std dev**	**mean**	**std dev**	**mean**	**std dev**	**mean**	**std dev**
BOV_SIFT_75	0.0293	0.0170	0.000054	0.000010	69.8517	2.3508	0.1066	0.0161	0.000347	0.000077	38.7097	5.2134
BOV_ASIFT_500	0.0668	0.0233	0.000199	0.000019	80.9683	2.6405	0.3181	0.0288	0.001290	0.000445	66.6667	4.1550
FV_SIFT_5	0.0490	0.0038	0.000225	0.000046	76.1766	2.3766	0.2979	0.0022	0.009501	0.000251	50.0000	3.2614
FV_ASIFT_5	0.0428	0.0089	0.000180	0.000056	76.3286	3.1973	0.2315	0.0619	0.000788	0.000203	59.3750	8.7565

Table 2: Runtime in *seconds* for the training and test phases for classifiers OPF and SVM with correspondent accuracy in % using the reduced Tatt-C dataset.

about 30 images (different for each fold) for test. After that, the simulation was repeated using now 25% of the background images (total of 1425 images) in the fold.

The use of a gradation in the number of background images corresponds to the fact that in a real world situation, non-tattoos exclusion algorithms can be applied before the tattoo identification process (with different degrees of failure, here represented by the notation 100% and 25% aggregated to the name of the algorithm).

It is important to note that in these two scenarios - amount of 100% and 25% of background images - we have reduced the amount of relevant images for training for BOV and FV. In the scenario of 100% the amount of relevant images was 0.9% (40/4674) and in the scenario 25% was 2.8% (40/1425). In the reduced Tatt-C dataset (first set of simulations) the ratio of relevant training image was 11.7% (40/342) in all cases.

Table 3 and Figure 6 presents runtime and accuracy values for SIFT, BOV_SIFT_75 and FV_SIFT_05 in both scenarios (100% and 25%). Table 3, as expected, shows that the use of the complete Tatt-C dataset has a significant impact on the performance of methods such as BOV and FV on the training phase (compare Tables 1 and 3). Figure 6 shows the effect in accuracy of the use of the background images. For FV_SIFT_5 (25%) and BOV_SIFT_75 (25%) the CMC curves were worse than in the case of the reduced dataset, for the case 100% of background the numbers were even worse (considering Figure 3 for BOV and Figure 5 for FV compared with Figure 6).

Next, we applied the above algorithms - BOV_SIFT_75, FV_SIFT_5 and SIFT for 100% and 25% of background with SVM and OPF classifiers; Table 4 presents runtime and accuracy results. The OPF classifier presents again better results in comparison to the SVM classifier in terms of runtime and accuracy. Considering accuracy and comparing reduced and complete dataset, see also Table 2, for BOV_SIFT_75 values were about 9% (100% background) and 3% (25% background) worse and for FV_SIFT_5 they were 10% and 6% worse.

Table 5 presents values of Rank 1 extracted from the values used for the construction of the CMCs. The best value for Rank 1 (72%) was obtained with the algorithm OPF + FV_SIFT_5(25%), according to line 10 of the table.

The next section concludes this article by summarizing the results obtained and pointing to the use of variations of the Tatt-C dataset as a tool for selection of candidate algorithms before exhaustive simulations.

Method	**training (all images)**		**test (one image)**	
	mean	**Std Dev**	**mean**	**Std Dev**
SIFT (100%)	2848 *		331.32	27.34
SIFT (25%)	790 *		103.50	8.67
BOV_SIFT_75 (100%)	1632.59	2.43	0.16	0.02
BOV_SIFT_75 (25%)	321.95	3.93	0.18	0.01
FV_SIFT_5 (100%)	2381.33	67.37	0.39	0.04
FV_SIFT_5 (25%)	920.41	62.17	0.44	0.03

* single execution.

Table 3: Runtime (in seconds) for the training and test phases.

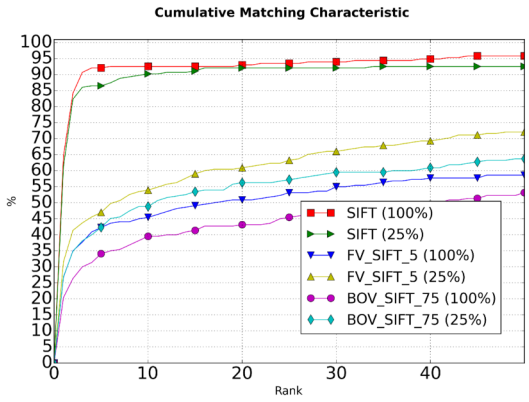


Figure 6: Cumulative Matching Characteristic considering 25% and 100% of background.

4 CONCLUSION AND NEXT STEPS

In this work, we studied the use of SIFT, ASIFT, BOV and FV algorithms and the use of BOV and FV with the OPF and SVM classifiers applied to tattos identification. SIFT and ASIFT provided base values as well as the descriptors used in the BOV and FV methods.

To optimize the simulation process, we defined a strategy based on the application of an initial set of candidate algorithms to a reduced configuration (without the background images) of the Tatt-C dataset. Then, based on cost (time) / benefit (accuracy) analysis we identified

	OPF						SVM					
	Training time (s) (all images)		Testing time (s) (one image)		Accuracy		Training time (s) (all images)		Testing time (s) (one image)		Accuracy	
Method	**mean**	**std dev**	**mean**	**std dev**	**mean**	**std dev**	**mean**	**std dev**	**mean**	**std dev**	**mean**	**std dev**
BOV_SIFT_75 (25%)	0.1542	0.0147	0.000211	0.000040	67.8313	2.7196	2.9266	0.0748	0.018228	0.000939	38.7097	7.9573
BOV_SIFT_75 (100%)	1.3295	0.0196	0.000648	0.000051	64.1376	1.9630	34.8473	1.4027	0.313152	0.006051	32.2581	6.3311
FV_SIFT_5 (25%)	0.6112	0.0236	0.000904	0.000087	71.6280	2.1479	7.3247	0.2340	0.021415	0.001260	43.7500	5.5115
FV_SIFT_5 (100%)	6.7016	0.0208	0.003131	0.000104	68.6940	2.5677	81.6082	1.4602	0.376530	0.015259	40.0000	5.4561

Table 4: Runtime in *seconds* for the training and test phases for classifiers OPF and SVM with correspondent accuracy in % using the complete Tatt-C dataset.

	Method	**Rank 1** (in %)
1	SIFT (100 %)	65
2	SIFT (25 %)	61
3	BOV _SIFT _75 (100 %)	20
4	BOV _SIFT _75 (25 %)	27
5	FV _SIFT _5 (100 %)	27
6	FV _SIFT _5 (25 %)	32
7	OPF + BOV _SIFT _75 (100 %)	64
8	OPF + BOV _SIFT _75 (25 %)	68
9	OPF + FV _SIFT _5 (100 %)	69
10	OPF + FV _SIFT _5 (25 %)	72
11	SVM + BOV _SIFT _75 (100 %)	32
12	SVM + BOV _SIFT _75 (25 %)	39
13	SVM + FV _SIFT _5 (100 %)	40
14	SVM + FV _SIFT _5 (25 %)	44

Table 5: Rank 1 list for the used algorithms.

the more suitable algorithms to be used in the simulations with the complete Tatt-C dataset.

As pointed in Section 3 we selected the SIFT (baseline), BOV_SIFT_75 and FV_SIFT_5 to be simulated with OPF and SVM algorithms using two Tatt-C dataset configurations, 100% and 25% of the background images. As already highlighted in Section 3, the idea of using two different amounts of background images was to simulate the quality of the image dataset (number of true tattoos).

The Rank 1 information provided by Table 5 in conjunction with the information provided by Table 3 and Table 4 allowed the completion of the experiment.

The training time column provided the runtime for the training phase of the different methods. So, for example to create the set of information used by OPF + FV_SIFT_5 (25%) we had to process the SIFT (runtime = 790s - Table 3), then the FV_SIFT_5 (25%) (runtime = 920s - Table 3) and then the OPF_FV_SIFT_5 (25%) (runtime = 0.61s - Table 4) totalizing 1710.6 seconds. The same goes for the other cases.

The test time columns are calculated in the same way. For example, for OPF + FV_SIFT_5 (25%), runtime is taken from Table 4, line 3 (0.000904s), while Table 5 points to this algorithm as the best for Rank 1 choice, presenting 72% accuracy.

Comparing (see Tables 3 and 4) the values obtained with the SIFT algorithm (baseline) and with SIFT combined with BOV or FV (without OPF) we observed a longer execution time and less precision compared to the same algorithms used in conjunction with OPF. In these cases, we obtain precision values that exceed the values obtained with SIFT (baseline). In the simulation results of Rank 1 presented in Table 5 using the Tatt-C data set with 100% of background images we obtained accuracy of 69% versus 65% (lines 9 and 1 respectively) and with a quantity of 25% of background images, accuracy of 72% versus 61% (lines 10 and 2 respectively).

In conclusion, it is important to bear in mind that the presented results were obtained with a relatively small image base and few cases of use per image. Thus, although we presented time values, for the test phase, by image, factors such as a considerable increase in the amount of images in the base will imply in the need to use memory management certainly impacting the access time of the images. On the other hand factors such as preprocessing in the images and optimization in the libraries, elements not used here, can contribute to the minimization of the times.

As a continuation of this work we see some scenarios. For example, the use of Deep Learning in the area of tattoos identification; the current literature presents studies in other use-cases - detection, similarity and de-identification [DP16], [HBRM16], [XGXHK16] - possibly due characteristics of current image databases (for example the NIST Tatt-C dataset presents a reduced number of tattoos for detection-training). A scenario we are already considering, taking advantage of a previous work [DSFM11] and also results related by Jain et al. [JLJ07], intend to couple the use of feedback by relevance to the identification process of tattoos leading the user to the decision loop.

5 ACKNOWLEDGMENTS

The research is supported by the National Counsel of Technological and Scientific Development (CNPQ) (141071/2015-0). The dataset was provided by the National Institute of Standards and Technology (NIST) and the Federal Bureau of Investigation (FBI).

6 REFERENCES

[CDF04] Gabriella Csurka, Christopher R. Dance, Lixin Fan, Jutta Willamowski, and Cédric Bray. Visual categorization with bags of keypoints. In Workshop on Statistical Learning in Computer Vision, ECCV, pages 1–22, 2004.

[CL11] Chih-Chung Chang and Chih-Jen Lin. Libsvm: A library for support vector machines. ACM Trans. Intell. Syst.Technol., 2(3):27:1–27:27, May 2011.

[DP16] Xing Di and Vishal M. Patel. Deep Tattoo Recognition. 2016 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW) , Pages: 119 – 126, Year: 2016

[DSFM11] André Tavares da Silva, Alexandre Xavier Falcão, and Léo Pini Magalhães. Active learning paradigms for CBIR systems based on optimum path forest classification. Pattern Recognition, 44(12):2971 – 2978, 2011.

[FCH08] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. Liblinear: A library for large linear classification. J. Mach. Learn. Res., 9:1871–1874, June 2008.

[HBRM16] Tomislav Hrkac, Karla Brkic, Slobodan Ribaric, Darijan Marcetc. Deep Lerning Architectures for Tattoo Detection and De-identification. 1St Int. Conf on Sensing, Processing and Learning for Intelligent Machines – SPLINE, July 6-8, 2016, Aalborg Univ, Denmark (pp. 45-49)

[HJ13] H. Han and A. K. Jain. Tattoo based identification: Sketch to image matching. In 2013 International Conference on Biometrics (ICB), pages 1–8, June 2013.

[JH99] Tommi S. Jaakkola and David Haussler. Exploiting generative models in discriminative classifiers. In Proceedings of the 1998 Conference on Advances in Neural Information Processing Systems II, pages 487–493, Cambridge, MA, USA, 1999. MIT Press.

[JLJ07] Anil K. Jain, Jung-Eun Lee, and Rong Jin. Tattoo-ID: Automatic Tattoo Image Retrieval for Suspect and Victim Identification, pages 256–265. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

[LJJ08] J.-E. Lee, A. K. Jain, and R. Jin, "Scars, marks and tattoos (SMT): Soft biometric for suspect and victim identification," Proceedings of the Biometrics Symposium, pp. 1–8, September 2008, Tampa, Florida, USA.

[LJJT12] J.-E. Lee, R. Jin, A. K. Jain, and W. Tong, "Image retrieval in forensics: Tattoo image database application," IEEE MultiMedia, vol. 19, no. 1, pp. 40–49, 2012.

[LOW99] D. G. Lowe. Object recognition from local scale-invariant features. In Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on, volume 2, pages 1150–1157 vol.2, 1999.

[Man12] D. Manger. Large-scale tattoo image retrieval. In Computer and Robot Vision (CRV), 2012 Ninth Conference on, pages 454–459, May 2012.

[MHW16] D. Manger, C. Herrmann, and D. Willersinn. Towards extending bag-of-words-models using context features for an 2d inverted index. In 2016 International Conference on Digital Image Computing: Techniques and Applications (DICTA), pages 1–5, Nov 2016.

[MRSP14] D. Marcetic, S. Ribaric, V. Struc, and N. Pavesic. An experimental tattoo de-identification system for privacy protection in still images. In 2014 37th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), pages 1288–1293, May 2014.

[NG15] M. Ngan and P. Grother. Tattoo recognition technology - challenge (tatt-c): an open tattoo database for developing tattoo recognition research. In Identity, Security and Behavior Analysis (ISBA), 2015 IEEE International Conference on, pages 1–6, March 2015.

[NQG16] Mei Ngan, George W. Quinn, and Patrick Grother. Tattoo recognition technology – challenge (tatt-c) outcomes and recommendations. revision 1.0. technical report nist 8078. Technical report, National Institute of Standards and Technology, sep 2016.

[PD07] F. Perronnin and C. Dance. Fisher kernels on visual vocabularies for image categorization. In 2007 IEEE Conference on Computer Vision and Pattern Recognition, pages 1–8, June 2007.

[PFS09] J. P. Papa, A. X. Falcão, and C. T. N. Suzuki. Supervised pattern classification based on optimum-path forest. Int. J. Imaging Syst. Technol., 19(2):120–131, June 2009.

[ST14] Rosália G. Schneider and Tinne Tuytelaars. Sketch classification and classification-driven analysis using fisher vectors. ACM Trans. Graph., 33(6):174:1–174:9, nov 2014.

[SZ03] J. Sivic and A. Zisserman. Video google: a text retrieval approach to object matching in videos. In Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on, pages 1470–1477 vol.2, Oct 2003.

[XGXHK16] Qingyong Xu, Soham Ghosh, Xingpeng Xu, Yi Huang, Adams Wai Kin Kong. Tattoo detection based on CNN and remarks on the NIST Database. 2016 International Conference on Biometrics (ICB) , Pages: 1 – 7, 2016

[YM11] Guoshen Yu and Jean-Michel Morel. ASIFT: An Algorithm for Fully Affine Invariant Comparison. Image Processing On Line, 1, 2011.

ISSN 1213-6972 Journal of WSCG http://www.wscg.eu/ Vol.26, 2018, No.1

Intuitive Transfer Function Editing Using Relative Visibility Histograms

Shengzhou Luo
Graphics Vision and Visualisation Group (GV2)
Trinity College Dublin
Ireland
luos@tcd.ie

Subhrajyoti Maji
Graphics Vision and Visualisation Group (GV2)
Trinity College Dublin
Ireland
majis@tcd.ie

John Dingliana
Graphics Vision and Visualisation Group (GV2)
Trinity College Dublin
Ireland
John.Dingliana@tcd.ie

ABSTRACT

In this paper, we present an interactive approach for intuitively editing colors and opacity values in transfer functions for volume visualization. We introduce the concept of a relative visibility histogram, which represents the difference between the global visibility distribution across the full volume and the local visibility distribution within a user-selected region in the viewport. From this measure, we can infer what subset of the 3D volume the user intends to select when they click on a region in the 2D rendered image of the data set, and use this to modify relevant parts of the transfer function. We use this selection mechanism for two alternative purposes. The first is to allow output-driven editing of the transfer function, whereby a user can change the opacity values and colors of features without directly having to manipulate the transfer function itself. The second is to extract visually dominant features in any user-selected region of interest, so that the user may individually edit their appearance and then merge these to create new transfer functions. Our approach is lightweight compared to similar techniques and performs in real-time.

Keywords

Volume rendering, transfer function, visibility, visibility histograms

1 INTRODUCTION

A recurring challenge in volume visualization is defining effective transfer functions (TF), which assign color and opacity (alpha value) to specific data ranges for visualization. Due to the indirect relationship between the transfer function and the resultant rendering, the process of editing transfer functions is often counter-intuitive, typically necessitating a trial-and-error process. This may be addressed using an output sensitive approach where the user can more directly control the appearance of the visualization, without explicit knowledge of the transfer function.

In this paper, we propose a technique which enables us to infer a user's intended changes to the visualization when they click or select a region in the rendered image of a 3D volume data set. 3D selection is a non-trivial process in volume visualization due to the presence of many overlapped layers of transparent data. This is achieved in our solution by weighting the data in the selected region based on the proportion of materials visible to the user within that region. We introduce the concept of a *relative visibility histogram*, derived from the relationship between the global visibility and the local visibility of data in the user-selected region. Based on this weighting, the user can directly modify colors and opacity values in the rendered image of the volume data, in a manner analogous to painting a 3D scene. In addition, we introduce an automated technique for creating transfer function components from relative visibility histograms to represent features of interest in the selected regions. This technique allows users to edit transfer function on a feature level by manipulating the colors and opacity values of the components and merging them to create new transfer functions. Compared to other similar techniques, our approach is relatively lightweight, requiring only intermediate information about the visibility of data samples. It is thus simple to implement and performs in real-time.

2 RELATED WORK

Transfer function specification is an essential part of the volume visualization pipeline. The specification is often achieved by a trial-and-error process, which in-

https://doi.org/10.24132/JWSCG.2018.26.1.6

volves a significant amount of tweaking of colors and opacity values, and the resultant visualization largely depends on how well the transfer function captures features of interest [KKH02] [KWTM03].One of the challenges for such an approach is highlighted by Mindek et al. [MMGB17] who argue that often small changes in input parameters (i.e. the transfer function variables) can lead to disproportionately large changes to the visualization. While Mindek et al. propose a data-sensitive solution that addresses this disproportionality, others take the route of output-driven transfer function editing, that is, manipulation of what is rendered, without the user being explicitly exposed to the underlying changes to the transfer function.

For instance, Guo et al. [GMY11] proposed a sketch-based approach that allows direct manipulation of transfer functions by brushing strokes on top of volume rendered images, which is similar to the operations in painting applications for 2D images. Later, Guo and Yuan [GY13] extended the sketch-based technique for specifying local transfer functions for topology regions using contour trees. Wu and Qu [WQ07] presented an approach that allows users to select sample images rendered using predefined transfer functions and generates new transfer functions by fusing multiple features in distinct volume renderings. Bruckner and Gröller [BG07] presented style transfer functions, which allow the user to specify styles extracted from actual illustrations in the transfer function. Ropinski et al. [RPSH08] proposed a stroke-based approach for specifying transfer functions by drawing strokes near silhouettes on a monochromatic view of the volume and generating transfer function components [CKLE98] that later can be modified and combined to explore the volume.

Many of the aforementioned approaches require, among other things, a model of what is visible to the user from a particular view direction, in other words, the visibility of features in volume data. The visibility of a sample refers to the alpha contribution of a sample to the final image, taking into account the degree to which it is occluded by other samples. This can be computed during ray-casting as the difference between the accumulated alpha of a sample and the accumulated alpha of the previous sample along a ray in the view direction [Ems08]. Correa and Ma presented the general notion of visibility histograms [CM11] which represent the distribution of visibility over intensity ranges in a volume rendering image. Wang et al. [WZC+11] extended visibility histograms to feature visibility histograms for measuring the influence of features on the resultant volume rendered images. Wiebel et al. [WVFH12] found that the user usually perceives features at a screen position with the highest visibility along a ray and exploited this information for volume picking.

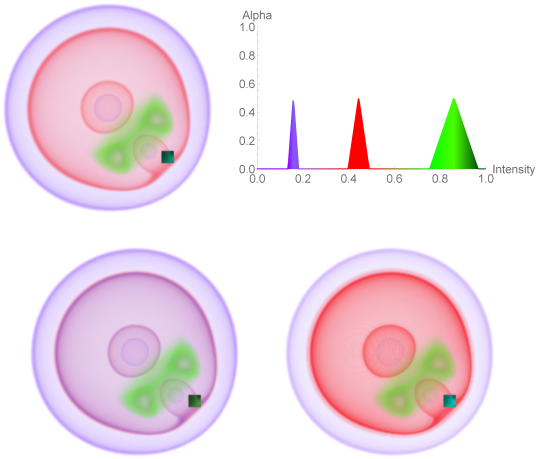


Figure 1: Sample operations using our technique. Top row: a nucleon with a selected region and its TF. Bottom left: Selected material colored in blue; Bottom right: Opacity of selected material enhanced

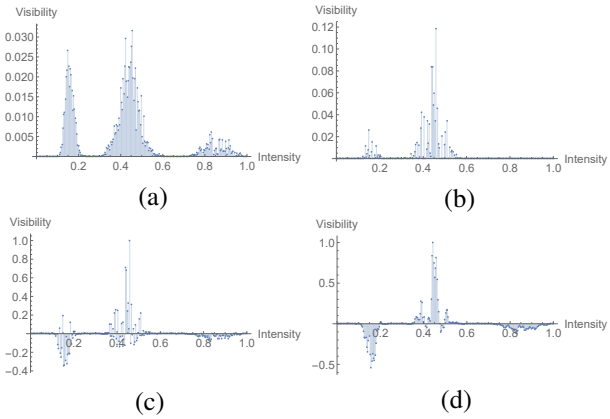


Figure 2: (a) Global Visibility Histogram of data set shown in Figure 1; (b) Local Visibility Histogram of selected region; (c) Relative Visibility Histogram; (d) Relative visibility histogram after smoothing

3 RELATIVE VISIBILITY HISTOGRAMS

In this section, we define a novel concept called the *Relative Visibility Histogram*, which is used as a mechanism for allowing users to select a subset of a 3D volume data set based on a selection in a 2D rendered view of the data. This is a key component that facilitates the two techniques, presented in subsequent sections, for intuitively manipulating transfer functions in volume rendering.

A visibility histogram [CM11] represents the visibility distribution of all voxels in the viewport when rendered from a given view, in other words, how visible any voxel is, given its opacity and the degree to which it is occluded by other voxels in the view direction. We will use the more specific term *Global visibility histogram*, H, to describe such a distribution and *Local visibility*

histogram, H_L, to describe the histogram representing the local visibility distribution for the voxels that contribute to a region of interest (ROI) in the rendered image, e.g., for a rectangular ROI on screen, this would be all the voxels that lie in the frustum extended by the rectangle. Furthermore, we introduce the concept of a Relative Visibility Histogram, derived from the former which is defined as the difference between H_L and H divided by the maximum of the absolute value in the difference, i.e.,

$$H_R = H_r / max(abs(H_r))$$

where $H_r = H_L - H$. The relative visibility histogram is scaled to the range $[-1,1]$ by dividing by the maximum absolute value in the histogram.

The purpose of the relative visibility histogram is as follows: firstly the local H_L component captures the dominant intensities in the ROI. Secondly, the subtraction and normalization against the global context capture a representation of which intensities are particularly densely distributed within the ROI and not elsewhere in the view. Essentially, we assume that when a user selects any particular ROI to select a subset of the volume, there is a strong likelihood that they will choose an area that contains a high number of voxels of the intensity ranges that they are interested in and that stands out as clearly dominated by that intensity range compared to the rest of the view.

An example is illustrated in Figure 1 and 2. Figure 1 shows a nucleon data set, its associated transfer function and sample modifications using our technique. The global visibility histogram is shown in Figure 2(a) and the local visibility histogram for the region of interest (the rectangle in inverted color) is shown in Figure 2(b). The relative visibility histogram is shown in Figure 2(c).

In order to smooth the histogram, we apply a Gaussian kernel to H_r and then scale it to the range $[-1,1]$. So the smoothed relative visibility histogram is

$$H_G = H_g / max(abs(H_g))$$

where $H_g = Gaussian(H_r, n, \sigma)$, n is the size and σ is the standard deviation of the Gaussian kernel (see Figure 2(d)).

Henceforth, this smoothed histogram H_G will be referred to as the relative visibility histogram, and $H_G(i)$, which is the value of the relative visibility histogram H_G at intensity i, will be referred to as the relative visibility of intensity i.

4 OUTPUT-DRIVEN COLOR AND ALPHA EDITING

The first application of the relative-visibility histogram is in allowing users to manipulate the visualization with no explicit knowledge of the transfer function. In this use-case, the user simply needs to select regions of interest on the rendered image that they wish to emphasize or color. Then, a single pass of volume ray casting is done to calculate the global visibility histogram for the whole volume and the local visibility histogram for the selected region in the viewport. From this we calculate the relative visibility histogram, which provides a measure of visible materials within the selected region. This is used to infer features that the user intends to edit in the visualization. More precisely, H_G is used as a weighting function to blend the colors or opacity values in the original transfer function with a user-selected target color or alpha value.

The user-selected target color is blended with the original transfer function for intensity ranges that have positive values in the relative visibility histogram as below:

$$C_i = \begin{cases} C_i + H_G(i)(C_s - C_i) & \text{if } H_G(i) > 0 \\ C_i & \text{otherwise} \end{cases}$$

where $H_G(i)$ denotes the relative visibility at intensity i in H_G, C_s is the user-selected target color and C_i the color of intensity i in the original transfer function.

Similarly, the alpha (A_i) of the transfer function is increased in intensity ranges that have positive relative visibility values, and decreased for ranges with negative relative visibility values, as follows:

$$A_i = \begin{cases} A_i + H_G(i)(1 - A_i) & \text{if } H_G(i) > 0 \\ A_i - H_G(i)(0 - A_i) & \text{otherwise} \end{cases}$$

Note that the color blending and alpha blending operations can be applied separately. Figure 1(c) displays the result of only applying the color blending to the volume rendering of the nucleon data set in Figure 1(a), and Figure 1(d) displays the result of only applying the alpha blending to the original.

Furthermore, as the blending process is fast, any changes can be applied by the user iteratively, analogous to a paintbrush-like tool for a large number successive of operations.

5 TRANSFER FUNCTION COMPONENTS

In addition to low-level output-driven editing of the transfer function, the relative visibility histogram allows us to support editing the transfer function at a feature level, which is a desirable use case in many volume visualization applications. For instance, Ropinski et al. [RPSH08] reported that physicians had high-level requests such as emphasizing, adding or removing specific features in collaboratively adapting visualizations.

As previously discussed, relative visibility histograms reveal what intensity ranges are concentrated in the

view frustum behind a selected 2D viewport region. The visually dominant features in the selected region can be represented by *transfer function components* proportional to the positive parts of relative visibility histograms.

A discrete set of such component transfer functions can be composed into a new transfer function for the full data set, and then the colors and opacity values of individual components can be separately modified.

5.1 Feature Specification Using Transfer Function Components

In this approach, features are automatically generated as transfer function components based on the relative visibility histograms created from user-selected regions. Let $F(i)$ denote a transfer function component derived from the relative visibility histogram H_G.

$$F(i) = \begin{cases} H_G(i) & \text{if } H_G(i) > 0 \\ 0 & \text{otherwise} \end{cases}$$

where $H_G(i)$ is the value of the relative visibility histogram H_G at intensity i.

Figure 3 displays two examples of the transfer function components created by a rectangular selection within the rendered volume image respectively. Figure 3 (a) and (b) show the two selected regions in the volume rendering. Figure 3 (c) and (d) show the two transfer function components which represent the relative visibility distributions of features in the two user-selected regions respectively. The regions are rectangles centered around a single point in 2D screen space, and the region sizes can be modified according to the user's need.

5.2 Image-Space Clustering for region of interest selection

Heretofore, we have assumed a rectangular region of interest centered on a point such as the location of the mouse cursor when a user clicks on the screen. While this is a reasonable representation of the user's interest for the purposes of low-level iterative editing proposed in the previous sections, a more robust representation of the user's intended selection may be obtained by a more generalized representation of this region. We propose that one alternative of the region of interest can be obtained by segmenting the rendered image into contiguous visual objects in 2D (we avoid using the term "feature" here to avoid confusion with the transfer function features discussed previously).

In order to achieve visual object selection, image segmentation is performed on volume rendered images and the resultant segments are stored as individual masks

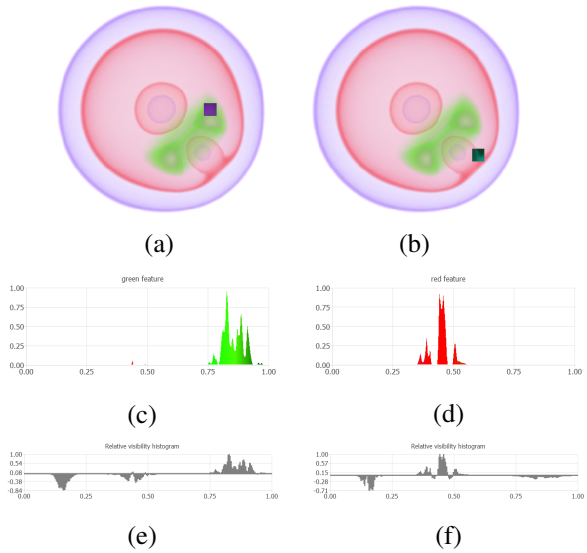


Figure 3: Examples of rectangular region selections. Left: A TF component (c) created from a green rectangular region (a) (highlighted in inverted colors) and its relative visibility histogram (e); Right: A TF component (d) created from a red region (b) and its relative visibility histogram (f).

for object selection. More specifically, a GPU accelerated k-means clustering implementation is used to accomplish interactive segmentation of volume rendered images. The distance metric used for the segmentation is the Euclidean distance in the RGB color space.

Now, when the user clicks on a position of the volume rendered image, a selected region is formed by all pixels that belong to the same segment as the pixel at the mouse position. Figure 4 displays the regions selected by clicking on the same screen positions as in Figure 3.

Compared to the rectangular regions in Figure 3 (a) and (b), the selected regions in Figure 4 (a) and (b) are heterogeneously-shaped segments with colors similar to the pixels at the respective mouse positions. These screen regions are larger in size, resulting in larger view frustums with more voxels being selected, and the colors across the selection tends to be more homogeneous due to the clustering. As a result of this, we note that the resultant transfer function components in Figure 4 (c) and (d) are more continuous compared to those in Figure 3 (c) and (d) for the rectangular selections. The relative visibility histograms in Figure 4 (e) and (f) are also smoother than those in Figure 3 (e) and (f). To disambiguate the two select techniques, we refer to them henceforth as *rectangular selection* and *generalized selection*.

5.3 Merging Transfer Function Components

A new transfer function can be created by merging several transfer function components.

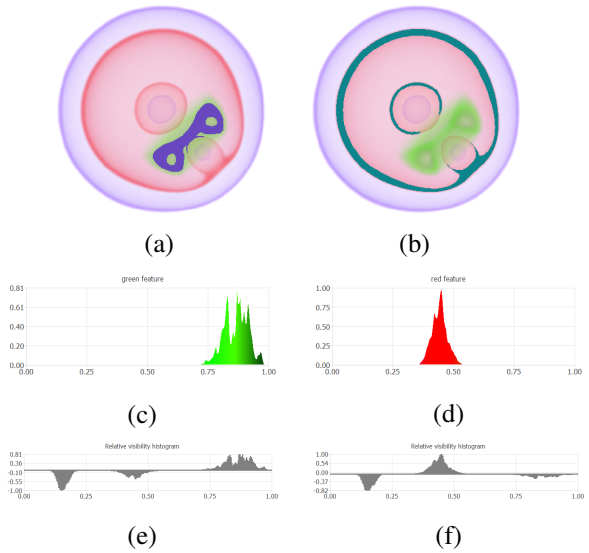


Figure 4: Examples of generalized segment selections. Left: A TF component (c) created from a green generalized selection segment (a) (highlighted in inverted colors) and its relative visibility histogram (e); Right: A TF component (d) created from a red segment (b) and its relative visibility histogram (f).

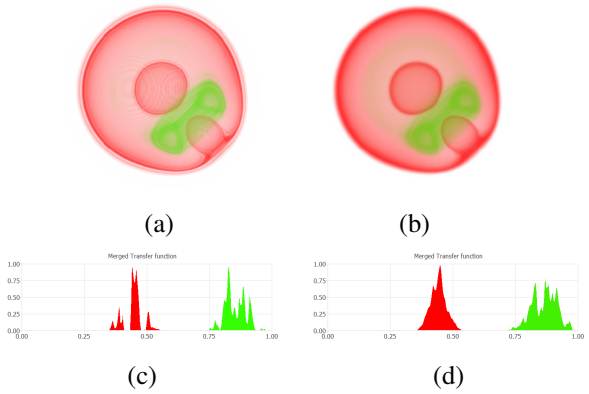


Figure 5: (a) and (c): Volume rendering and its TF obtained from merging the TF components in Figure 3 using *rectangular selection*; (b) and (d): Volume rendering and TF based on *generalized selection* in Figure 4

The opacity function A_i is defined by a weighted sum of transfer function components clipped to the range $[0,1]$.

$$A_i = \begin{cases} a(i) & \text{if } a(i) \in [0,1] \\ 0 & \text{if } a(i) < 0 \\ 1 & \text{if } a(i) > 1 \end{cases}$$

where $a(i)$ is the weighted sum of transfer function components, i.e.

$$a(i) = \sum_{j=1}^{n} w_j F_j(i)$$

where w_j ($w_j \geq 0$) is the weight of transfer function component F_j, $F_j(i)$ is the value of F_j at intensity i, and n is the number of transfer function components.

Figure 5 shows the results of merging the transfer function components in Figure 3 and Figure 4 respectively.

Note that there are "wood grain" artifacts in the volume rendered image in Figure 5 (a), especially in the red feature. The artifacts are due to the gaps in the transfer function components, as shown in Figure 5 (c). In contrast, the volume rendered image in Figure 5 (b) does not have noticeable artifacts, because the transfer function components in Figure 5 (d) are smoother.

There may be overlaps between transfer function components. Two methods for deciding the color of the overlaps in the merged transfer functions are described below.

5.3.1 *Blending colors of transfer function components*

The first method is blending the colors of the transfer function components based on the weights and the values of the transfer function components.

In order to keep the blended colors in a valid range, the weights for blending the colors of transfer function components are normalized using weighted averages of the weights and the values of the transfer function components. The normalized weight of transfer function component F_j is defined by

$$W_j = \frac{w_j F_j(i)}{\sum_{k=1}^{n} w_k F_k(i)}$$

where w_j ($w_j \geq 0$) is the weight of transfer function component F_j, $F_k(i)$ is the value of the transfer function component F_k at intensity i, and n is the number of transfer function components.

Hence, the color function C_i is defined by

$$C_i = \sum_{j=1}^{n} W_j c_j$$

where W_j is the normalized weight of transfer function component F_j and c_j is the color of F_j, and n is the number of transfer function components.

Using this method, new colors that do not exist in the settings of transfer function components may be introduced due to the blending of colors of overlapping transfer function components. Moreover, a feature, represented by a transfer function component, may have various colors in the final volume rendering.

5.3.2 *Using colors of the dominant components*

In some cases, using a single color per feature is preferable for better distinction of features in the volume rendering.

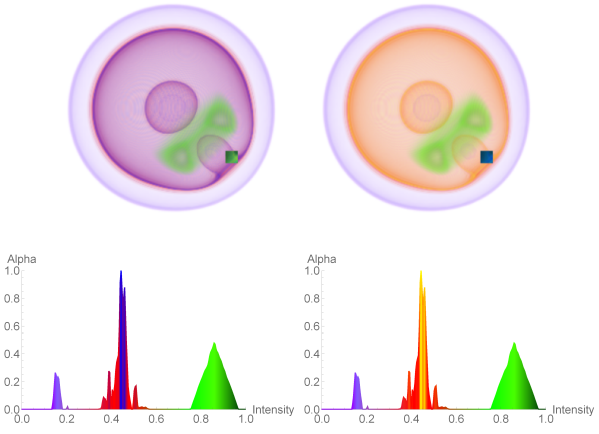


Figure 6: Combining operations. Left: Blue applied to selected region of TF in Figure 1 and opacity enhanced; Right: Yellow applied and opacity enhanced. The modified TF is shown for each case below the rendering

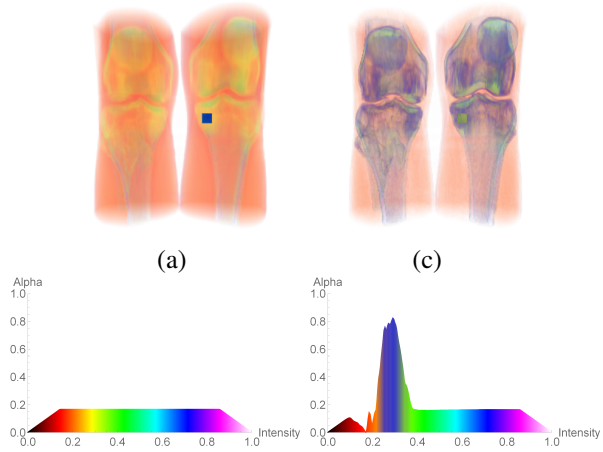


Figure 7: Left: CT-knee data set and basic TF; Right: Volume rendering and TF after blue applied and opacity enhanced for the selected region

Thus an alternative to blending colors is to use the color of the visually dominant transfer function component as the merged color.

The color function C_i is defined by

$$C_i = c_j, \quad j = \underset{k\in\{1,\ldots,n\}}{\operatorname{argmax}}\, w_k F_k(i)$$

where c_j is the color and w_j ($w_j \geq 0$) is the weight of transfer function component F_j with $w_j F_j(i)$ at intensity i that is maximum among the n transfer function components.

With this method, different features would have distinct colors, so that they are distinguishable by colors in the volume rendering.

6 RESULTS

Our solution comprises a ray-cast volume renderer and a visibility computation module, both implemented on the GPU using CUDA. The implementation is lightweight and achieves real-time performance at 30 to 40 frames per second on a computer equipped with an Intel Xeon E3-1246 v3 CPU and an NVIDIA Quadro K4200 graphics card.

We present some results to demonstrate the effectiveness of our approach on the nucleon (voxel dimensions: $41 \times 41 \times 41$), CT-knee ($379 \times 229 \times 305$) data sets, one time-step of a simulated turbulent vortex flow ($128 \times 128 \times 128$) and one time-step of a simulated supernova ($432 \times 432 \times 432$). Our implementation was able to handle all the data sets at interactive rates.

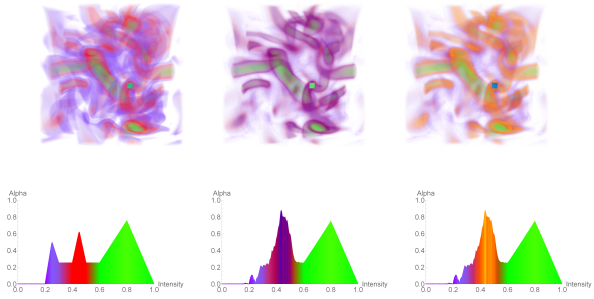
6.1 Output-driven Transfer Function Editing

Figure 6(left) displays the result of both applying color blue and adjusting alpha of the TF in Figure 1. Note that the intensity ranges with initial red color in the middle of the transfer function have been blended with blue and have become purple. Similarly, Figure 6(right) shows the result of applying yellow and adjusting alpha. Here, the intensity ranges in the middle have become orange after blending with yellow. In both cases, the alpha of the relevant parts of the transfer function is increased and the alpha of the less relevant parts is decreased in order to emphasize the materials of interest.

Figure 7(left) shows a rendered image of a CT-knee data set with a selected region over the bone and the initial transfer function. Figure 7(right) shows the image and the transfer function after applying a blue color and alpha adjustment. The bone material becomes mostly blue and is emphasized due to increased opacity, while the materials around the bone with lower relative intensity ranges are de-emphasized.

Figure 8 shows results of enhancing one time-step of a turbulent vortex data set. Figure 8(a) shows the rendered image and original transfer function. Figure 8(b) shows a clear visualization, and respective TF, of the materials of interest blended with blue and emphasized with higher alpha. Similarly, Figure 8(c) shows the materials of interest blended with yellow and emphasized with higher alpha.

The examples show that the technique can be applied effectively to a range of different data sets and transfer functions. Although we only show single step examples due to space constraints, it should be noted that changes can be applied by the user iteratively, analogous to a paintbrush-like tool for a large number successive of operations.



(a) (b) (c)

Figure 8: (a)Turbulent vortex data set and initial TF; (b)Blue applied to selected material and opacity enhanced; (c) Yellow applied and opacity enhanced.

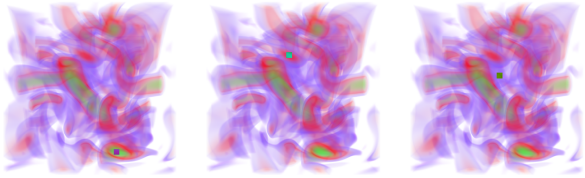
6.2 TF-components editing

Figure 9, Figure 10 and Figure 11 show results of creating and merging transfer function components on the time-step of the turbulent vortex data set.

Figure 9 displays three transfer function components created from a green region, a red region and a purple region, which are rectangular regions highlighted in inverted colors, in the volume rendering respectively. In contrast, Figure 10 displays three transfer function components created from three generalized segments based on image segmentation results. The three segments are visual objects with similar colors and are selected by clicking on the same positions as in the rectangular regions in Figure 9.

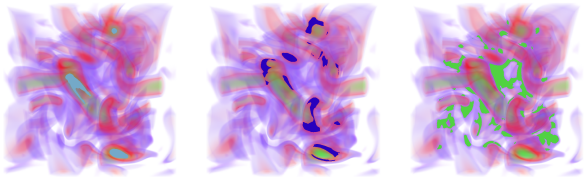
Figure 11 (a) and (d) show the volume rendered image and the transfer function created from merging the three transfer function components in Figure 9. The user interface for editing and merging transfer function components is displayed in Figure 11 (d), which shows the individual transfer function components with their weights and colors on the left, and the resultant transfer function at the bottom. Similarly, Figure 11 (b) and (e) display the volume rendered image and the transfer function created from merging the three transfer function components in Figure 10 with color blending, and Figure 11 (c) and (f) display the results of merging the three transfer function components in Figure 10 using colors of the dominant components. In Figure 11, the three transfer function components are merged with weights {2,1,0.2}, so that the purple feature is de-emphasized, the green feature is emphasized, and the red feature remains the same level of opacity.

Figure 13 displays the results of merging two transfer function components, i.e. the red feature and the green feature, with weights {1, 1}. The transfer function components are created from the user-selected rectangular regions in Figure 12 (a) and (b), and the user-selected generalized segments in Figure 12 (c) and (d) respectively. Figure 12 (e) shows the initial transfer function used for volume rendering.



(a) (b) (c)

Figure 9: User-selected regions highlighted in inverted colors. (a): a rectangular region in the green material; (b): a rectangular region in the red material; (c): a rectangular region in the purple material



(a) (b) (c)

Figure 10: Generalized segments selected at the same positions as in Figure 9, highlighted in inverted colors. (a): a segment in the green material; (b): a segment in the red material; (c): a segment in the purple material

From Figure 11 and Figure 13, we observe that the transfer function components created from segments are smoother and have wider intensity ranges than those created from rectangular regions.

7 CONCLUSION

In this paper, we introduce relative visibility histograms for inferring user intentions and present interactive techniques for editing colors and opacity values in transfer functions for volume visualization. We describe an output-driven color and alpha editing technique as well as a higher level technique that involves creating and merging transfer function components which represent features in the volume rendering.

Our color and alpha editing approach described in section 4 has a similar interaction paradigm to that proposed by Guo et al. [GMY11] in terms of emphasizing features and applying colors to features. However, the feature definition in Guo et al.'s approach relies on clustering of four attributes, i.e. depth, visibility, alpha and intensity. The clustering of attributes of voxels may be computationally heavy particularly for large volume data sets. In contrast, our approach identifies relevant intensity ranges of the transfer function based purely on visibility information, thus requiring a much more lightweight approach.

Our transfer function components approach discussed in section 5 is similar to the work by Ropinski et al. [RPSH08] in how the transfer function components are modified and merged to create new transfer functions. However, the two approaches differ in how features are

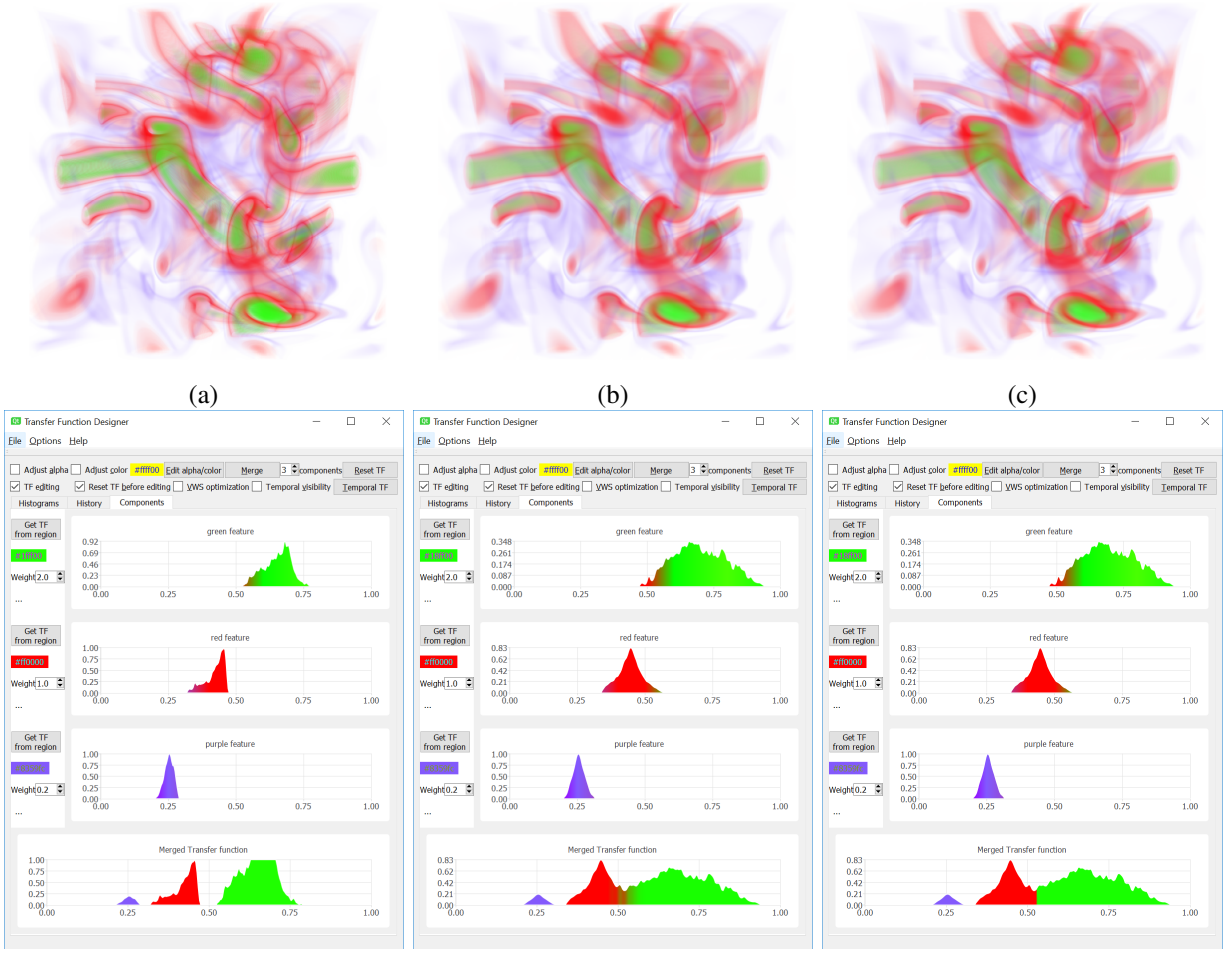


Figure 11: Merging 3 features with weights {2, 1, 0.2} and colors from peaks of TF components; (a) and (d): Volume rendering and TF from merging the TF components in Figure 9; (b) and (e): Volume rendering and TF from merging the TF components in Figure 10 with color blending; (c) and (f): Volume rendering and TF from merging the TF components in Figure 10 using colors of the dominant components

identified and transfer function components are generated. The approach by Ropinski et al. generates two further strokes which are both parallel to the user-drawn stroke along the silhouette and positioned in the same distance on its opposite sides. They hypothesize that the inner stroke covers the feature of interest in image space while the outer stroke does not cover it. In some cases, such as a complex flow visualization, it may be difficult to draw a stroke along the silhouette and determine a distance so that the two further generated strokes would be one inside the feature of interest and the other outside of it.

In our transfer function components approach, only a region inside the feature of interest is needed for creating a transfer function component to represent the feature. Moreover, apart from selecting a rectangular region around the mouse position, the user can also select segments generated from k-means clustering in image space. The segments often cover more pixels and thus lead to smoother transfer function components.

Both the visibility-driven transfer functions by Correa and Ma [CM11] and the feature visibility technique by Wang et al. [WZC+11] utilize iterative optimizations to refine the transfer functions. Correa and Ma derived the target visibility distribution from the user-defined transfer function, while Wang et al. allow the user to specify the target feature visibility. The transfer functions are then refined by minimizing the difference between the visibility distribution of the current volume rendering and the target visibility distribution. In contrast, our approaches do not involve an iterative optimization process. The relative visibility histogram is only computed once in both of the uses cases we presented.

However, the proposed techniques are subject to the limitations of 1D transfer functions, e.g. it is not possible to separate features of interest that overlap in the 1D transfer function domain. Both the color and alpha edit-

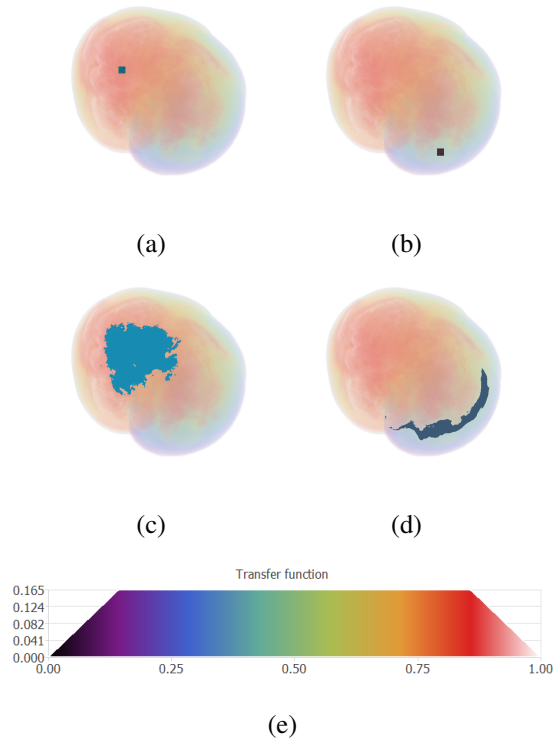


Figure 12: User selections on the rendering of a supernova data set; (a) & (b): User-selected rectangular regions highlighted in inverted colors; (c) & (d): User-selected generalized segments highlighted in inverted colors; (e): The initial transfer function

ing and transfer function components techniques are based on relative visibility histograms, which indicate the difference between the global visibility distribution across the volume and the local visibility distribution within the user-selected region. Therefore, only materials that are visible in the initial transfer function can be captured and edited by the proposed techniques.

In future, we would like to conduct user studies to evaluate the effectiveness of the proposed techniques, in particular with expert users in specific domains that use volume data. We believe our approach is particularly suited for tasks where there is no clear a priori search target, which might be the case in many complex fluid visualizations. We used some standard approaches to some component mechanisms of our solution and further study may be warranted to determine if the use of alternative segmentation techniques or a perceptually-based color space such CIE-LAB may improve the quality of the overall solution.

8 ACKNOWLEDGEMENTS

The Supernova data set is made available by Dr. John Blondin at the North Carolina State University through US Department of Energy's SciDAC Institute for Ultrascale Visualization. Other data sets were obtained from the Volume Library courtesy of Stefan Roettger, the Stanford Volume Data archive and the Time-varying data repository at UCDavis. The nucleon data set was obtained from the free distribution of the Voreen engine. We would like to thank the respective owners for making these data sets available.

This research has been conducted with the financial support of Science Foundation Ireland (SFI) under Grant Number 13/IA/1895.

9 REFERENCES

[BG07] S. Bruckner and M. E. Gröller. Style transfer functions for illustrative volume rendering. *Computer Graphics Forum*, 26(3):715–724, 2007.

[CKLE98] Silvia Castro, Andreas König, Helwig Löffelmann, and Eduard Gröller. Transfer Function Specification for the Visualization of Medical Data. Technical Report, Universität Wien, 1998.

[CM11] Carlos D. Correa and Kwan-Liu Ma. Visibility histograms and visibility-driven transfer functions. *IEEE Transactions on Visualization and Computer Graphics*, 17(2):192–204, 2011.

[Ems08] Gerlinde Emsenhuber. *Visibility Histograms in Direct Volume Rendering*. Master's Thesis, Institute of Computer Graphics and Algorithms, Vienna University of Technology, November 2008.

[GMY11] Hanqi Guo, Ningyu Mao, and Xiaoru Yuan. WYSIWYG (What You See is What You Get) Volume Visualization. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2106–2114, 2011.

[GY13] Hanqi Guo and Xiaoru Yuan. Local WYSIWYG volume visualization. In *Visualization Symposium (PacificVis), 2013 IEEE Pacific*, pages 65–72, February 2013.

[KKH02] Joe Kniss, Gordon Kindlmann, and Charles Hansen. Multidimensional transfer functions for interactive volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 8(3):270–285, July 2002.

[KWTM03] Gordon Kindlmann, Ross Whitaker, Tolga Tasdizen, and Torsten Möller. Curvature-based transfer functions for direct volume rendering: Methods and applications. In *Proceedings of the 14th IEEE Visualization 2003 (VIS'03)*, VIS '03, pages 513–

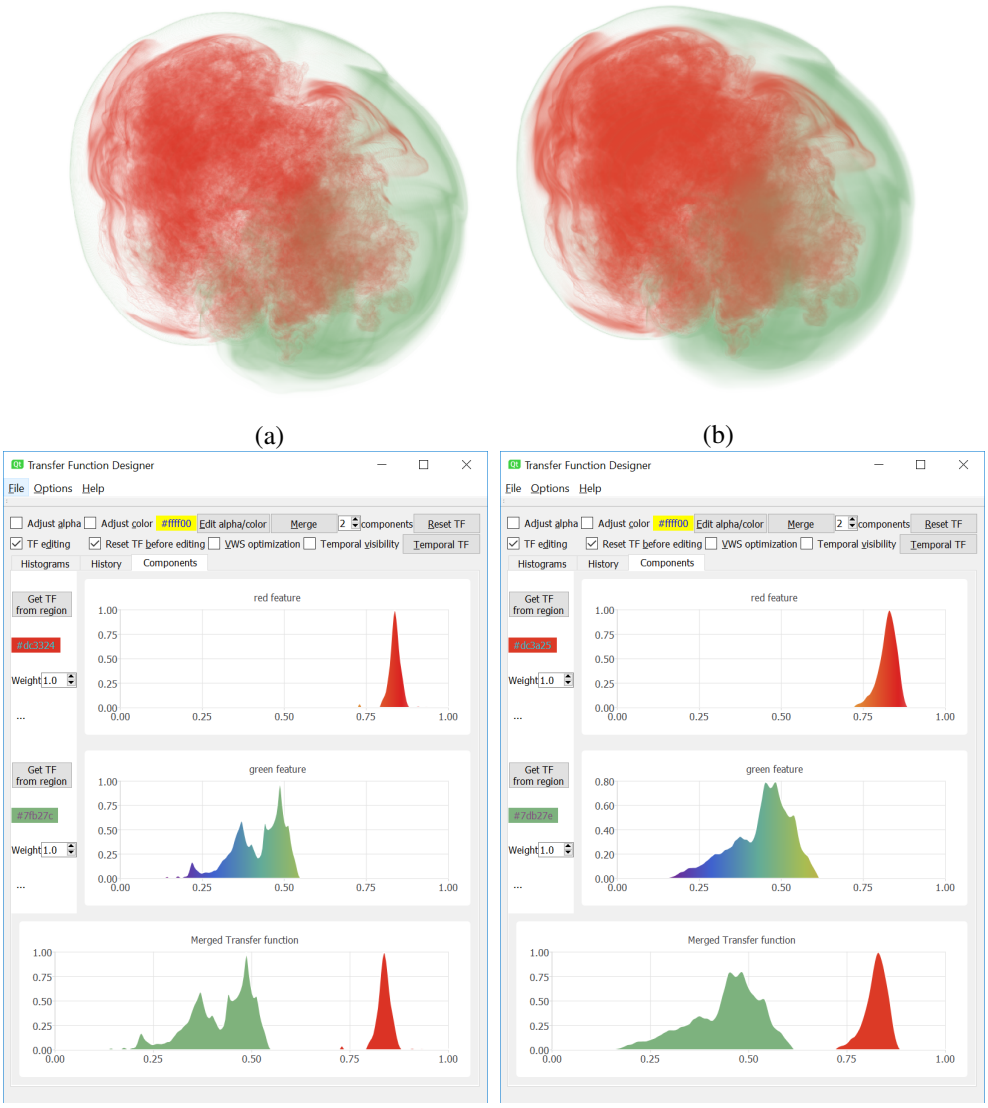


Figure 13: Merging 2 features with weights {1, 1} and colors from peaks of TF components; (a) & (c): Volume rendering and TF from merging the TF components in Figure 12 (a) and (b); (b) & (d): Volume rendering and TF from merging the TF components in Figure 12 (c) and (d)

520, Washington, DC, USA, 2003. IEEE Computer Society.

[MMGB17] Peter Mindek, Gabriel Mistelbauer, Eduard Gröller, and Stefan Bruckner. Data-sensitive visual navigation. *Computers & Graphics*, 67:77–85, October 2017.

[RPSH08] Timo Ropinski, Jörg-Stefan Praßni, Frank Steinicke, and Klaus H. Hinrichs. Stroke-Based Transfer Function Design. In *IEEE/EG International Symposium on Volume and Point-Based Graphics*, pages 41–48. IEEE, 2008.

[WQ07] Yingcai Wu and Huamin Qu. Interactive transfer function design based on editing direct volume rendered images. *IEEE Transactions on Visualization and Computer Graphics*, 13(5):1027–1040, 2007.

[WVFH12] A. Wiebel, F.M. Vos, D. Foerster, and H.-C. Hege. WYSIWYP: What You See Is What You Pick. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2236–2244, 2012.

[WZC+11] Yunhai Wang, Jian Zhang, Wei Chen, Huai Zhang, and Xuebin Chi. Efficient opacity specification based on feature visibilities in direct volume rendering. *Computer Graphics Forum*, 30(7):2117–2126, 2011.

ISSN 1213-6972 Journal of WSCG http://www.wscg.eu/ Vol.26, 2018, No.1

Dynamic Correction of Image Distortions for a Kinect-Projector System

Jihoon Park
Gwangju Institute of Science and Technology
123 Cheomdangwagiro, Bukgu
Gwangju, Republic of Korea (61005)
parkbi16@gist.ac.kr

Seonghyeon Moon
Gwangju Institute of Science and Technology
123 Cheomdangwagiro, Bukgu
Gwangju, Republic of Korea (61005)
moonsh@gist.ac.kr

Kwanghee Ko
Gwangju Institute of Science and Technology
123 Cheomdangwagiro, Bukgu
Gwangju, Republic of Korea (61005)
khko@gist.ac.kr

ABSTRACT

This paper addresses the problem of correcting distortion in an image projected onto a target screen without using a camera. Unlike a camera-projector system that projects a special pattern on the screen and acquires it using a camera for distortion correction, the proposed system computes the amount of correction directly from the geometric shape of the screen, which is captured by a Kinect device, a scanner that produces 3D points of the screen shape. We modified the two-pass rendering method that has been used for the projector-camera system. An image is created on the Kinect plane. Next, the image is mapped to the 3D points of the screen shape obtained by the Kinect device using the ray-surface intersection method. Finally, a corrected image is obtained by transforming the image on the 3D point set to the projector plane. The proposed method does not require a marker or a pattern and can be used in a dynamic environment where the shape of the screen changes, or either the viewer's position and direction change. Various tests demonstrate the performance of the proposed method.

Keywords

Geometric Alignment, Projector-Kinect system, Projection in dynamic environment, C^2-continuous surface.

1 INTRODUCTION

Projection displays an image or information on the surface of an object such as a flat white screen or a wall. It has been used in various fields such as education, presentation, and performing arts. A projector is installed in such a way that the optical axis of the projector and the flat screen are perpendicular to each other. Sometimes a non-flat surface is considered such as displaying images or other contents on the outer surface of a building in a media-art performance show. In this case the images from the projector should be adjusted to minimize any distortion caused by the relative relation between the geometric shape of the surface, the positions and directions of the projector and the viewer. Many researchers have focused on methods for automatic correction of distortions in an image that is projected on the surface of an object considering the projector and the viewer.

Most methods use a projector-camera system, a system consisting of a projector for projecting an image and a camera that captures the geometric distortion. The distortion is then corrected based on the detected information [An16]. [Ahm13] are focused on the approximation of a shape using a higher order B-spline surface. The method corrects the distortion through the difference between the two sets of feature points; one contains the origin points on the projector plane, and the other has the extracted features from the pattern image. The extracted points are then transformed to the projector plane through homography. Thereby, the distortion in the image can be adjusted by warping the original image based on the difference between the origin and the corresponding transformed feature points. Kaneda *et al.* [Kan16] considers the case that the projector's optical axis and the screen are not perpendicular to each other when a planar screen is used. In this method, a distorted image and the geometry information of the screen are obtained through a camera and a Kinect device. The method also uses homography to correct the distorted image. Unlike the previous two methods, depth information obtained by the Kinect is used to decide the orientation of the planar screen. A

https://doi.org/10.24132/JWSCG.2018.26.1.7

normal vector is estimated from the depth values. Then, two perpendicular vectors that define the plane of the screen are obtained using the normal vector. The shape of the corrected image is determined using the vectors by considering the relative geometric relation to the optical axis. The method adjusts the distorted image into a rectangular one with respect to the orientation of the screen. However, the method is only focused on the planar screen only.

Methods using surface reconstruction are similar to the aforementioned ones. The main concept is to transform a rendered image in the world space to the projector space. This process is called the two pass rendering method [Ras98]. First, an image is mapped to the surface of the screen model through rendering. Here, the screen model can be modeled directly or reconstructed from a set of measured points. Then, the image on the model surface is transformed to the projector plane. In [Bro05], [Ras99] and [Ras00], the two pass rendering method for a projector-camera system is used for correcting the distortion in an image. Here, a 3D screen model is obtained by the camera pair, and a projection matrix is established from the relation between the projector and the position of the viewer. Then, an image without distortion is created. The first pass is finished by rendering the desired image on the 3D screen model. Next, the rendered image is transformed to the projector plane through the projection matrix. The image transformed to the projector plane becomes the corrected image at the viewer's viewpoint. If the projection matrix and the position of the viewer are known, the method can correct the distorted image whenever the viewer moves.

The methods based on the projector-camera system, however, do not extend well to the dynamic environment where the lighting condition, the shape of the screen, and the viewer's position and orientation change. The change of the shape of the screen may alter the reflection pattern of the screen; the intensity or color of the screen in the image may change. The same phenomenon happens when the view position changes. In such cases, the feature extraction step, which is used for estimating 3D points of the shape of the screen, may fail to detect the features for the estimation because image processing methods used in the feature detection are not stable to the lighting condition and therefore the 3D shape may not be obtained robustly. In addition, the projector-camera system requires projecting a pattern on the surface of the screen for generating the 3D shape of the screen. It means that whenever the shape of the screen or the position and the orientation of the viewer change, the pattern should be projected and processed to obtain the geometric shape, which means that we cannot expect a continuous projection of images on the screen.

Kundan and Reddy [Kun13] proposed a geometric compensation method of a non-planar surface with a Kinect device. The compensated image is obtained by warping a mesh model of the target surface that is calculated from a depth map generated by the Kinect. The method does not require any pattern for acquiring the 3D surface model because it can be directly obtained by the Kinect.

We propose a method for correcting the image distortion when an image is projected onto the surface of a screen. Here, we consider two cases: the change of the screen shape with a static viewer's position and orientation, and the change of the viewer's position and orientation with a static screen position and shape. The proposed method uses a Kinect v2 device for acquisition of the 3D shape of the screen and a projector for image projection on the screen. No projection and acquisition of a pattern are required and the method can obtain the 3D shape of the screen in real time.

The contributions of the proposed method are twofold. First, the proposed method can handle the cases mentioned above. In addition, it can process the dynamically changing environment in the distortion correction computation. Second, a registration based method is proposed to estimate the relative change of the viewer's position and orientation in the distortion correction step. Based on these technical contributions, the method is demonstrated with several examples.

2 PROPOSED METHOD

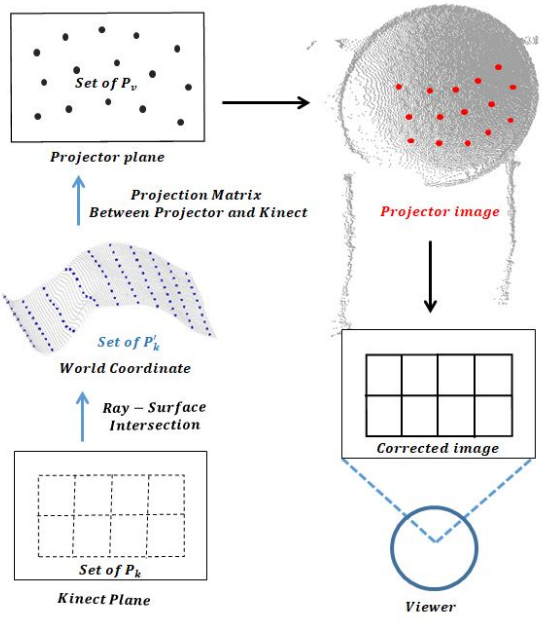


Figure 1: Overview of correcting a distorted image.

In this work, we consider a projector-Kinect system and modify the two pass rendering method introduced

in [Ras98] for the proposed system. We take some assumptions. First, the Kinect coordinate coincides with the viewer's. Second, the screen is a C^2-continuous surface. In addition the screen is always in front of the Kinect. Figure 1 shows the overall procedure of the proposed method.

The 3D geometric shape of the screen is obtained using the Kinect device. Here, we assume that the viewer's position and direction are the same as those of the Kinect device. An image that is needed to be projected on the screen is assumed to be in the Kinect plane, a virtual 2D plane that the viewer watches. The pixels of the image in the Kinect plane are denoted P_k. They are mapped on the screen surface through the ray-surface intersection to produce P'_k, which correspond to the rendered image on the 3D surface. Then P'_k are transformed to the projector plane by a transformation matrix to yield P_v. When P_v is projected on the surface of the screen, an image whose distortion has been corrected is displayed on the surface.

2.1 Kinect Device

There are two types of Kinect, Kinect v1 and Kinect v2. Kinect v2 is an improved version of Kinect v1. The devices are affordable and easy to use. However, they have inherent noise in the measurement, which prevents them from being used in applications that require high accuracy. The quantitative analysis of the noise of Kinect v1 and Kinect v2 over the scan distance is made as shown in Fig. 2 [Pag15]. According to [Pag15], Kinect v2 has the noise of $0.02m \sim 0.03m$ in the maximum depth of $5m$, and the point cloud obtained by Kinect v2 becomes sparse as the depth increases.

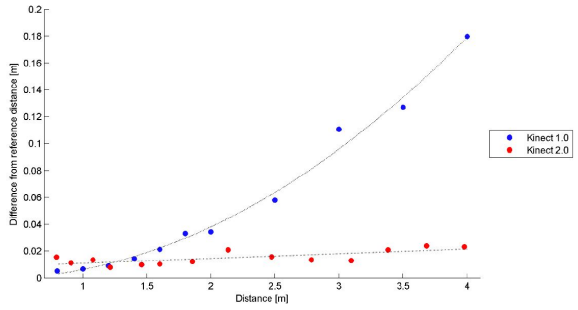


Figure 2: Comparison of the noise levels of Kinect v1 and Kinect v2 with respect to the distance to the target. [Pag15]

Moreover, the maximum range that Kinect v2 can robustly cover is 5m according to the specification of the device and [Pag15]. Therefore, the effective space where the proposed system works would be limited.

2.2 Geometric Correction

The projection matrix is necessary to transform an image in the Kinect space to the plane of a viewer. We calculate the projection matrix $\mathbf{T}(x)$ relating the initial position of the viewer to the projector once before correction [Jon14]. Next, we obtain the shape of the screen for distortion correction. The point set representing the screen is extracted from the depth values measured by the Kinect. Grid points, which cover an image to be projected on the screen, are created on the Kinect plane. Next, rays are shot from the Kinect origin $(0,0,0)$ through each of the grid points. From the rays, a virtual frustum is created, which intersects the point set of the measured point set. Then, the points within the frustum are collected and used as the points that lie on the screen. For this process, the kd-tree data structure is used for an efficient computation [Pha10]. The angle of the line segment connecting a point in the measurement and the Kinect origin is considered. Namely, the angles of the segment with respect to the xy, xz and yz planes are computed. If the angles are close to those of a ray, then the point is considered to be within the frustum and taken as a point on the screen.

The Kinect device produces measurements with some noise. The noise may cause a serious problem in the distortion correction process. In particular computation of the intersection between the screen and the ray, one of the steps in the proposed method, is mostly compromised when the raw measured points are used directly. Therefore, the noise level in the measurement data set should be controlled. In this study, we use a hierarchical B-spline approximation method by [Lee97] to avoid such a problem. Here, the approximation of the points using a B-spline surface is used as a low pass filter. Suppose that we have a set of points with some noise. Representing the shape defined by the points accurately may require a function of high order or a B-spline surface with many control points because the high frequency components of the noise should be considered. Unless they are part of the surface, they do not have to be represented in the surface definition and should be smoothed out to obtain the underlying geometric structure. A B-spline surface with a reasonable number of control points can be used to filter out the high frequency noise components and to approximate the given points with satisfactory accuracy. For this purpose we consider the hierarchical B-spline approximation method. Approximation is started with a small number of control points such as 4×4. If the error of approximation is larger than the user defined tolerance, the control net is refined to be 8×8, and the points are approximated again. This refinement step is repeated until a surface with reasonable accuracy is obtained. The approximated surface is used as a virtual screen in the proposed method. Next, an image on the Kinect plane is rendered on the virtual screen to obtain the positions of the pixels of the image in the world coordinate space. The positions correspond to the intersections between the rays and the virtual surface. The virtual

screen is given as a cubic B-spline surface, and the intersection points P'_k between the rays and the surface are calculated through Newton-Raphson method [Pre92]. Finally, the corrected points P_v on the projector plane is obtained by transforming P'_k to the projector plane through the projection matrix **T**. A viewer can see the desired image by projecting P_v to the screen.

The aforementioned process can correct the distortion of an image in a static environment, where the shape of the screen and the position and orientation of the viewer do not change. A new projection matrix must be computed to show a corrected image on the screen whenever either the shape or the position and direction change. This computation process requires the calibration process, which hinders the continuous projection of an image on the screen. A solution to this problem is proposed for two cases.

2.3 Case 1: Change of Screen Shape with Static View Position and Orientation

When the shape of a screen is changed, the parameters of the projection matrix are constant because the orientation and position of the projector and the viewer (Kinect) do not change. Instead, new intersection points with respect to the changed shape are computed. They are obtained in the same way as presented in the previous section. However, if the points are calculated at each frame, the computation time is increased, and the frame rate of projection is decreased. So, it is necessary to determine if the shape of the screen has been changed or not. First, the mean of the depth values is calculated at the current frame. Then, the difference between the current mean and the previous values is calculated. If the difference value is lower than the threshold, we determine that the current shape be equal to the previous one, do not perform the step of obtaining the new intersection points at the current frame and return to the acquisition step. Otherwise, we decide that the current shape be changed. Hence, new intersection points are calculated at the current frame. Then, a corrected image is obtained with respect to the changed shape by multiplying the projection matrix to the new intersection points. Figure 3 shows the overall process for Case 1.

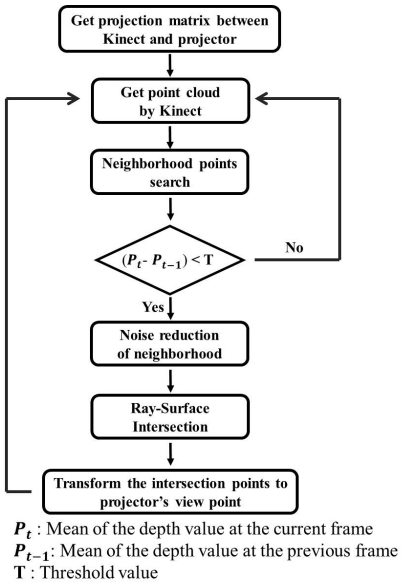


Figure 3: Flowchart for handling Case 1.

2.4 Case 2: Change of Position and Orientation of Viewer with Static Screen

When a viewer moves, the parameters of the projection matrix should be changed accordingly. The projection matrix consists of an intrinsic and an extrinsic matrices. The intrinsic matrix is related to the device properties such as the focal length, the principal point, etc. For this reason, it is not influenced by the position or orientation of the viewer. The extrinsic matrix, however, should be modified with respect to the new viewer's position and orientation because it captures the relation between the viewer and the projector. In this study, a registration algorithm is employed to estimate the relation.

Suppose that the viewer has moved from pos_1 to pos_2. At pos_1, the shape of the screen S_1 has been measured by the Kinect. After the movement, the screen shape S_2 is measured by the Kinect at pos_2. The relation of the viewer's position and orientation can be estimated from S_1 and S_2. Since S_1 and S_2 are point clouds with some overlap of the same shape, they can be registered to form one point cloud of the screen shape in the reference coordinate system, which produces the rigid body transformation that registers S_2 onto S_1 as closely as possible. This transformation provides the relative relation of the viewer at pos_1 and pos_2, which can be translated into the relation of the viewer at the new position to the projector. The point-to-plane algorithm [Low04] is used for computing the transformation matrix to register S_1 and S_2. The new projection matrix at pos_2 is calculated by multiplying the inverse of **M** to the extrinsic matrix at pos_1. The process should only be performed when the viewer's position and orientation have been changed much, which can be decided by checking the change of depth values as is performed for Case 1 because the change of the viewer's position and orientation is equivalent to the relative change of the shape of the screen. The step by step procedure of the proposed method for Case 2 is illustrated in Fig. 4.

3 RESULT AND DISCUSSIONS

The proposed method is implemented in C++. The workstation used for testing has a 4-GHz Intel Core i7 CPU with 8GB RAM. We use a Microsoft Kinect v2 to obtain a point cloud and a Panasonic PT-DX1000

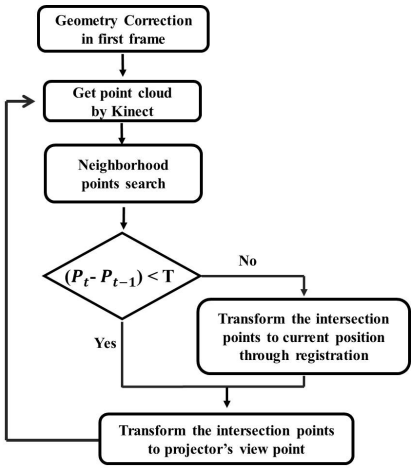


Figure 4: Flowchart for handling Case 2.

projector. Two types of screen are considered in the test. One is a spherical screen. The distance between the Kinect and the spherical screen is determined such that the projected image covers the maximum area of the screen. The other is a curtain screen the shape of which can be changed arbitrarily. Here, the distance between the Kinect and the screen is about 2 meters. To simulate the dynamic environment, the spherical screen is moved or rotated, and the shape of the curtain was changed with a hand by pushing or pulling it behind. A viewer also moves within a valid range of the projector.

We use the overlap ratio between the ideal and the corrected images for error evaluation, which is denoted R. The overlap ratio quantifies how much similar the corrected image is to the ideal one. The ideal image is represented as a grid whose numbers of the columns and rows are equal to the feature points. The ratio is calculated by dividing the number of the overlapped pixels by the number of the ideal pixels. It is expressed as

$$R = \frac{\sum_i P_{overlap}}{\sum_i P_{ideal}} \times 100, \quad (1)$$

where $P_{overlap}$ are the pixels in the overlapping area and P_{ideal} are the ideal pixels. Figure 5 shows the corrected image by the proposed method.

The overlap ratios before and after noise reduction are 66.66% and 89.5%, respectively. The result without noise reduction shows that the grid lines are not straight due to the inaccurate intersection points as shown in Fig. 6(a). On the other hand, the result after noise reduction has a higher overlap ratio as 20% as shown in Fig. 6(b).

Table 1 shows the computation time of each step for Cases 1 and 2. The three steps of searching the adjacent points, generating a surface with noise reduction

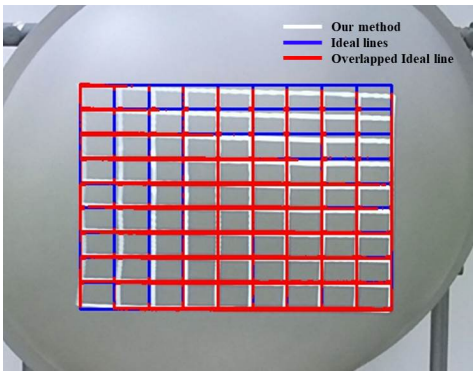
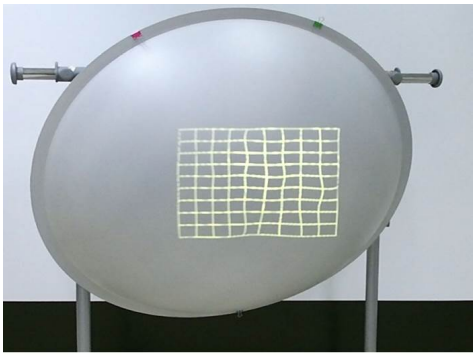
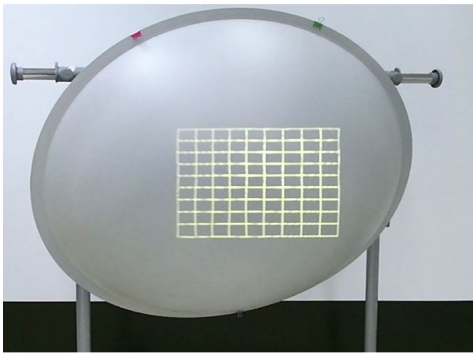


Figure 5: Corrected image by the proposed method. The blue lines in the figure show an ideal grid and the white lines show a corrected grid. The overlapped ideal lines are represented in red color.



(a) Corrected image before noise reduction



(b) Corrected image after noise reduction

Figure 6: Corrected images with and without noise reduction. The resolution of figures is [395 x 307]

and calculating intersection points through Newton-Raphson method are the same for the two cases. The process of searching and generating only takes less than 0.01s. Most of the computation time is spent for registering the two point clouds as 0.78s. The computation time is affected by the number of grid points and a generated cubic B-spline surface.

The number of grid points increases the computation time of the entire process but can contribute to the higher accuracy of correction. More grid points can obtain more accurate information about the distortion

Case 1	
Process	Computation time(second)
Search the region	0.011s
Noise reduction	0.003s
Ray-surface intersection	0.0571s
Total	Over 0.06s
Case 2	
Process	Computation time(second)
Search the region	0.011s
Noise reduction	0.003s
Registration	0.78s
Total	Under 1.00s

Table 1: Computation time of the proposed method

because more intersection points are used. However, a large number of grid points may result in a discontinuous surface and makes the intersection computation fail from time to time. Therefore, a tradeoff between the number of grid points, the computation time and the accuracy should be taken into account. We have chosen the 10×10 grid points empirically in the proposed method. We tested various numbers of grid points to analyze the influences to the distortion correction. After a series of experiments, we found that the accuracy is almost unchanged although the number of grid points is increased from 10×10. However, the computation time is quite sensitive to the number of grid points because intersection should be computed the number of times proportional to the number of grid points. For example, the accuracy of the result is converged to 89.5% although the number of grid points is increased from the 10×10 grid points in Fig. 6. However, the computation time is significantly increased. If the number of grid points is changed to 15×15 from 10×10, the computation time grows to $0.79s$ from $0.06s$.

A curtain is used to test the performance of the proposed method as shown in Fig. 7. In this test, the shape of the curtain is changed, and the position and orientation of the viewer are fixed. Two different types of shapes are considered for the experiments. Figures 7-(a) and (b) show a distorted and a corrected images on the curtain of one shape, respectively. Similarly, Figures 7-(c) and (d) show images before and after distortion correction on another shape of the curtain. Here, the resolution of the images is 376×297. As shown in the figures, the proposed method corrected the distortions of the images and produced corrected ones on the different shapes of the curtain, respectively. The correction was performed at the speed of 10 FPS (Frames Per Second).

Figure 8 shows the case when two different screen shapes are considered. In this test, a spherical and a curtain are used with a different image. As shown in the figure, the proposed method successfully corrects the distortions and shows the corrected images on the screen.

Figure 9 shows the corrected images when the position and orientation of the viewer change. Here, the shape of the curtain is maintained. The method successfully produces the corrected images at the three different positions and orientations of the viewer (pos1, pos2, and pos3) as shown in Figs. 9-(a), (b), and (c). In this test, the resolution of the figures used in this test is 376×297. The process runs at about 1 FPS. The drop of FPS in this case mostly attributes to the estimation of transformation between two positions using the registration method.

4 CONCLUSION

In this study, we propose a method of correcting the distorted projector image in a dynamic environment using a Kinect device. The dynamic environment includes the two cases: that the shape of the screen changes and that the position and orientation of the viewer change. Additionally, the proposed method can compensate the distortion of the two cases simultaneously during execution.

The proposed method uses the Kinect for obtaining the 3D shape of the screen in real time, which is an advantage of the proposed method over others that use a camera-projector configuration. Therefore, the acquisition step is not influenced by lighting conditions. Moreover, when the position and orientation of the viewer change, the proposed method estimates the projection matrix only by considering the relative motion between the positions and orientations before and after the viewer moves, which is computed by a registration method.

However, there are a few limitations of the proposed method. The method cannot differentiate Cases 1 and 2 automatically because they address the same problem from the theoretical viewpoint. It means that we must select one of the two cases before executing the proposed method. As a possible solution, an additional sensor such as an accelerometer or a gyro sensor can be employed to detect the position or the orientation change of the viewer. Moreover, the implementation of the method needs to be refined for improving computation time to yield a higher frame rate for real-time operation. Alternatively, a parallel computation scheme can be introduced in the intersection computation between a ray and a surface for reducing the computation time. Finally, the proposed system has been designed to consider one projector, which limits the screen area that the system can cover. Overcoming the range of one projector can be achieved by using multiple projectors, and each of the projectors can be handled individually using the multiple thread framework. These problems need to be solved before the proposed method is used in practice, which is recommended for future work.

5 ACKNOWLEDGMENTS

This research was supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Science and ICT(2017R1A2B4012124) and by the MSIP(Ministry of Science, ICT and Future Planning), Korea, under "Development of a smart mixed reality technology for improving the pipe installation and inspection processes in the o shore structure fabrication(S0602-17-1021)" supervised by the NIPA (National IT industry Promotion Agency).

6 REFERENCES

[Ahm13] A. Ahmed, R. Hafiz, M. M. Khan, Y. Cho and J. Cha, Geometric Correction for Uneven Quadric Projection Surfaces Using Recursive Subdivision of Bezier Patches, ETRI Journal 35, 6 (2013)

[An16] H. An, Geometrical Correction for Arbitrarily Curved Projection Surface By Using B-Spline Surface Fitting, Master's thesis, Gwangju institute of Science and Technology, 2016

[Kan16] T. Kaneda, N. Hamada and Y. Mitsukura, Automatic Alignment Method for Projection Mapping on Planes with Depth, 2016 IEEE 12th International Colloquium on Signal Processing Its Applications (CSPA), pp.111-114, 2016.

[Ras98] R. Raskar, G. Welch, M. Cutts, A. Lake, L. Stesin and H. Fuchs, The Office of the Future: A Unified Approach to Image-based Modeling and Spatially Immersive Displays, SIGGRAPH '98, pp.179-188, 1998.

[Ras00] R. Raskar, Immersive planar display using roughly aligned projectors, Proceedings IEEE Virtual Reality 2000 (Cat. No.00CB37048), New Brunswick, NJ, pp.109-115, 2000.

[Ras99] R. Raskar, M. S. Brown, R. Yang, W.-C. Chen, G. Welch, H. Towles, B. Scales and H. Fuchs, Multi-projector displays using camera-based registration, Visualization '99. Proceedings, San Francisco, CA, USA, pp.161-522, 1999.

[Kun13] S. D. Kundan and G. R. M. Reddy, Projection and Interaction with Ad-hoc Interfaces on Non-planar Surfaces, 2013 2nd International Conference on Advanced Computing, Networking and Security, Mangalore, pp.1-6, 2013.

[Bro05] M. Brown, A. Majumder and R. Yang, Camera-based calibration techniques for seamless multiprojector displays, in IEEE Transactions on Visualization and Computer Graphics, vol. 11, no. 2, pp.193-206, March-April 2005.

[Jon14] B. Jones, R. Sodhi, M. Murdock, R. Mehra, H. Benko, A. Wilson, E. Ofek, B. MacIntyre, N. Raghuvanshi and L. Shapira, RoomAlive: Magical Experiences Enabled by Scalable, Adaptive Projector-camera Units, Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology, Honolulu, Hawaii, USA, pp.637-644, 2014.

[Pha10] M. Pharr and G. Humphreys, Physically Based Rendering, Second Edition: From Theory To Implementation, 2nd ed. Morgan Kaufmann Publishers Inc., 2010.

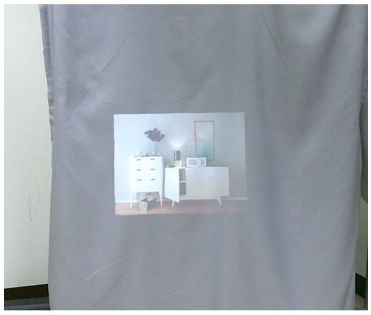
[Pag15] D. Pagliari and L. Pinto, Calibration of Kinect for Xbox One and Comparison between the Two Generations of Microsoft Sensors. Zlatanova S, ed. Sensors (Basel, Switzerland). 2015.

[Lee97] S. Lee, G. Wolberg and S. Y. Shin, Scattered data interpolation with multilevel B-splines, IEEE Transactions on Visualization and Computer Graphics, vol. 3, no. 3, pp.228-244, 1997.

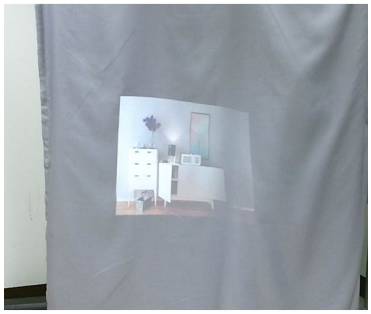
[Pre92] W. H. Press, S. A. Teukolsky, W. T. Vetterling and B. P. Flannery, Numerical Recipes in C, 2nd ed. Cambridge University Press, 1992.

[Low04] K. L. Low, Linear Least-Squares Optimization for Point-to-Plane ICP Surface Registration. Chapel Hill, University of North Carolina, vol. 4, 2004.

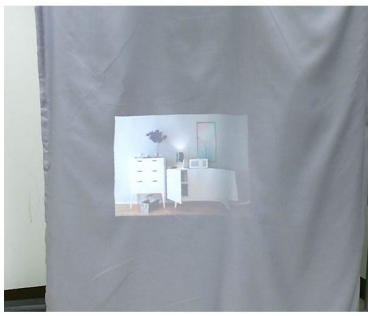
(a) Distorted image on $shape_1$



(b) Corrected image on $shape_1$



(c) Distorted image on $shape_2$



(d) Corrected image on $shape_2$

Figure 7: Distorted and the corrected images by our method on the different shapes of a curtain. The images of (a)-(b) and (c)-(d) use the same shape of the screen, respectively. (a) and (c) show the distorted images with respect to the shape and (b) and (d) show the corrected images, respectively. The resolution of figures is 376×297.

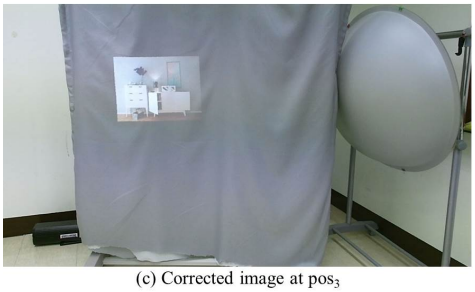
(a) Corrected image on the spherical screen. The resolution : [427,326]

(b) Corrected image on the curtain. The resolution : [676 x 512]

Figure 8: Results of our method using the curtain and the spherical screen.

(a) Corrected image at pos_1

(b) Corrected image at pos_2



(c) Corrected image at pos_3

Figure 9: Results of the distortion correction when the position and orientation of the viewer change. (a), (b) and (c) show the corrected images at three different positions and orientations of the viewer. The resolution of the figures used in this test is 376×297.

[KJ13] Nilay Khatri and Manjunath V. Joshi. Image super-resolution: Use of self-learning and gabor prior. In Kyoung Mu Lee, Yasuyuki Matsushita, James M. Rehg, and Zhanyi Hu, editors, *Computer Vision – ACCV 2012*, pages 413–424, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[KJ14] N. Khatri and M.V Joshi. Efficient self-learning for single image upsampling. In *22nd International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG 2014)*, pages 1–8, 2014.

[LHG+10] Xuelong Li, Yanting Hu, Xinbo Gao, Dacheng Tao, and Beijia Ning. A multi-frame image super-resolution method. *Signal Processing*, 90(2):405 – 414, 2010.

[MPSC09] Dennis Mitzel, Thomas Pock, Thomas Schoenemann, and Daniel Cremers. Video super resolution using duality based tv-l1 optical flow. *Proceedings of the 31st DAGM Symposium on Pattern Recognition*, pages 432–441, 2009.

[NM14] Kamal Nasrollahi and Thomas B. Moeslund. Super-resolution: A comprehensive survey. *Machine Vision Applications*, 25(6):1423–1468, August 2014.

[NMG01] Nhat Nguyen, P. Milanfar, and G. Golub. A computationally efficient superresolution image reconstruction algorithm. *IEEE Transactions on Image Processing*, 10(4):573–583, Apr 2001.

[NTP17] Mattia Natali, Giulio Tagliafico, and Giuseppe Patan. Local up-sampling and morphological analysis of low-resolution magnetic resonance images. *Neurocomput.*, 265(C):42–56, November 2017.

[PC12] P. Purkait and B. Chanda. Super resolution image reconstruction through bregman iteration using morphologic regularization. *IEEE Transactions on Image Processing*, 21(9):4029–4039, Sept 2012.

[RIM17] Y. Romano, J. Isidoro, and P. Milanfar. Raisr: Rapid and accurate image super resolution. *IEEE Transactions on Computational Imaging*, 3(1):110–125, March 2017.

[RU90] S. P. Raya and J. K. Udupa. Shape-based interpolation of multidimensional objects. *IEEE Transactions on Medical Imaging*, 9(1):32–42, Mar 1990.

[SLJT08] Qi Shan, Zhaorong Li, Jiaya Jia, and Chi-Keung Tang. Fast image/video upsampling. *ACM Transactions on Graphics*, 27(5):153:1–153:7, December 2008.

[Sob68] Feldman G. Sobel, I. A 3x3 isotropic gradient operator for image processing. *Stanford Artificial Intelligence Project*, 1968.

[TAe17] R. Timofte, E. Agustsson, and L. V. Gool et.al. Ntire 2017 challenge on single image super-resolution: Methods and results. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1110–1121, July 2017.

[TDSVG15] Radu Timofte, Vincent De Smet, and Luc Van Gool. A+: Adjusted anchored neighborhood regression for fast super-resolution. *Computer Vision – ACCV 2014: 12th Asian Conference on Computer Vision, Singapore, Singapore, November 1-5, 2014, Revised Selected Papers, Part IV*, pages 111–126, 2015.

[War03] Greg Ward. Fast, robust image registration for compositing high dynamic range photographs from handheld exposures. *Journal of Graphics Tools*, 8:17–30, 2003.

[YMY14] Chih-Yuan Yang, Chao Ma, and Ming-Hsuan Yang. Single-image super-resolution: A benchmark. In *Proceedings of European Conference on Computer Vision*, 2014.

[YSL+16] Linwei Yue, Huanfeng Shen, Jie Li, Qiangqiang Yuan, Hongyan Zhang, and Liangpei Zhang. Image super-resolution: The techniques, applications, and future. *Signal Processing*, 128:389 – 408, 2016.

[YZS12] Q. Yuan, L. Zhang, and H. Shen. Multi-frame super-resolution employing a spatially weighted total variation model. *IEEE Transactions on Circuits and Systems for Video Technology*, 22(3):379–392, March 2012.

[ZC14] H. Zhang and L. Carin. Multi-shot imaging: Joint alignment, deblurring, and resolution-enhancement. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2925–2932, June 2014.

[ZWZ13] H. Zhang, D. Wipf, and Y. Zhang. Multi-image blind deblurring using a coupled adaptive sparse prior. In *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1051–1058, June 2013.