3D content recovery and complexity reduction

Tan, Cheen Hau

2015

Tan, C. H. (2015). 3D content recovery and complexity reduction. Doctoral thesis, Nanyang Technological University, Singapore.

https://hdl.handle.net/10356/62230

https://doi.org/10.32657/10356/62230

3D Content Recovery and Complexity Reduction

TAN CHEEN HAU

School of Electrical and Electronic Engineering

A thesis submitted to the Nanyang Technological University

in partial fulfilment of the requirement for the degree of

Doctor of Philosophy

2015

Acknowledgements

First and foremost, I would like to thank my supervisor Assoc. Prof. Chau Lap Pui for this supervision and support throughout the period of my study. I am grateful for his patience and understanding throughout my research.

I would like to thank my parents for their unwavering support and encouragement through the difficult times I had in my study.

I would like to thank my friends in NTU for their support and encouragement, and for accompanying me on this journey.

I would also like to express gratitude to the lab technician Ms How for her assistance when my computer is uncooperative.

Finally, I would also like to thank Nanyang Technological University for providing a scholarship and good facilities for my study.

Abstract

3D content, such as motion capture data and 3D meshes, are widely used in a number of sectors, such as entertainment and sports. Due to issues related to the acquisition of 3D content, postprocessing is often required on captured 3D content before it can be used in applications. For human motion capture (mocap) data acquired using optical systems, entries of the data might be missing due to occluded body parts or markers. For scanned 3D meshes, the complexity, or resolution, of the mesh could be much higher than necessary for the intended application; this will consume a large amount of computational resources. Hence, this thesis focuses on using mocap recovery algorithms to recover any missing mocap data, and on using mesh simplification algorithms to reduce the resolution of meshes.

For mocap recovery, we focus on using low rank matrix completion with Singular Value Thresholding (SVT) optimization to recover mocap data. Previously, mocap data is arranged in the form of a matrix, where each frame forms a column to allow low rank matrix completion to recover missing entries. We present three strategies to extend and improve the performance of this mocap recovery framework, namely using a trajectory-based matrix representation, applying skeleton constraints, and using subspace constraints. Our mocap recovery methods target two types of missing data: random missing data, where each joint in a sequence are missing at random, and block missing data, where each joint

is missing for long intervals of time.

For the case of random missing data, we propose to arrange the mocap data matrix into columns of short trajectories, which we call the trajectory-based representation, for matrix completion recovery. We show that this representation produces a matrix with a lower rank than the previous frame-based matrix representation. Since matrix completion performs better on matrices with lower rank, this fact allows the SVT matrix completion method, by using the proposed representation, to recover missing mocap data at a much lower error.

Both frame-based and trajectory-based matrix completion exploit different types of correlations; frame-based matrix completion relies on the correlation between different frames, whereas trajectory-based matrix completion relies on the correlation between trajectories. We propose to exploit both types of correlation simultaneously by constraining the solution of trajectory-based matrix completion in the subspace formed by the solution of frame-based matrix completion. The proposed method shows significant improvement over both frame-based and trajectory-based matrix completion.

For block missing data, the effectiveness of matrix completion in recovering missing data decreases significantly when mocap data entries are missing for extended periods of time. To alleviate this problem, we exploit the fact that human bones are rigid and constrain inter-joint distances of connected joints. To this end, we extend the SVT matrix completion method to include skeleton constraints. The proposed method improves on the Singular Value Thresholding method significantly, especially when mocap data joints are missing for many consecutive frames.

Image-Driven Simplification is a 3D mesh simplification method that simplifies meshes based on their visual appearance. It renders images of a mesh from each of 20 surrounding viewpoints to estimate the visual appearance of the mesh. Hence, this method is

very time consuming since it requires repeated renders and image readback cycles during simplification. We propose to accelerate Image-Driven Simplification by using only one adaptively placed viewpoint instead of 20 viewpoints with fixed locations. Two computationally efficient methods are proposed to dynamically compute the viewpoint location so that simplification performance is not compromised. Both single viewpoint simplification methods run at an estimated twenty-five times faster than the original method at competitive simplification performance.

Contents

List of Figures

List of Tables

List of Abbreviations

SVT	Singular Value Thresholding
FSVT	Frame-based Singular Value Thresholding
TSVT	Trajectory-based Singular Value Thresholding
SCFSVT	Skeleton Constrained Frame-based Singular Value Thresholding
CFTSVT	Combined-Frame-and-Trajectory-based Singular Value Thresholding
IDS	Image-Driven Simplification
MA-IDS	Maximum Angle Image-Driven Simplification
ME-IDS	Maximum Error Image-Driven Simplification
LOD	Level of Detail
GPU	Graphics Processing Unit

List of Notations

We follow a standard rule for notations of variables in this thesis. Scalars are denoted in italics (s), vectors are denoted in boldface lowercase letters ($\mathbf{v}$), and matrices are denoted in boldface uppercase letters ($\mathbf{M}$). All vectors are column vectors. Vectors or matrices with italicized subscripts denote an element of the vector or matrix. For example, $\mathbf{v}_i$ is an element of vector $\mathbf{v}$, and $\mathbf{M}_i$ is an element or vector in matrix $\mathbf{M}$. Vectors or matrices with normal font subscripts denote the type of vector or matrix. For example, $\mathbf{M}_{\mathrm{f}}$ and $\mathbf{M}_{\mathrm{t}}$ denote an input matrix in frame-based representation, and a matrix in trajectory-based representation, respectively, in Chapter 3.

Below, we list a summary of key notations for each chapter.

Chapter 3

σ	Singular value
$\mathbf{f}$	Frame of a mocap sequence
$\mathbf{t}$	Trajectory segment
$\mathbf{M}$	Matrix representation of input mocap sequence
$\mathbf{M}_{\mathrm{f}}$	Frame-based matrix representation of input mocap sequence
$\mathbf{M}_{\mathrm{t}}$	Trajectory-based matrix representation of input mocap sequence

Chapter 4

σ	Singular value
$\mathbf{A}$	Sampling matrix
$\mathbf{C}$	Skeleton constraint matrix
$\mathbf{b}$	Observed entries of input mocap matrix
vec	linear operator which maps a matrix to a vector
$\mathcal{L}$	Lagrangian

Chapter 5

$\mathbf{M}_{\mathrm{f}}$	Frame-based matrix representation of input mocap sequence
$\mathbf{M}_{\mathrm{t}}$	Trajectory-based matrix representation of input mocap sequence
$\mathbf{A}$	Sampling matrix
$\mathbf{b}$	Observed entries of input matrix
$\hat{\mathbf{X}}_{\mathrm{f}}$	Solution of FSVT
vec	linear operator which maps a matrix to a vector
$\mathcal{L}$	Lagrangian
$\mathbf{T}_{\mathrm{tf}}$	Matrix that transforms a vector from trajectory-based to frame-based representation

Chapter 6

$\hat{M}$	Original mesh
M^k	A stage of a simplified mesh
ecol	Edge collapse operation
$I_{\mathrm{X}}, I_{\mathrm{Y}}$	Image rendered from a viewpoint
v_0	Vertex to be collapsed
$R_{\mathrm{X}}, R_{\mathrm{Y}}$	1-ring region around vertex to be collapsed
PA_f	Projected area of f
n_f	Face normal
n_v	Vertex normal
v_p	Viewpoint vector

Chapter 1

Introduction

1.1 Motivation

For the past few decades, the prevalent visual media used by industries and consumers, such as images and videos, are spatially two dimensional. However, the research and application of spatially three dimensional (3D) content, such as 3D meshes and motion capture data, have been growing. Due to the advantages offered by 3D content, 3D content is being increasingly adopted by researchers and professionals in various sectors, such as entertainment and sports. Furthermore, the recent releases of 3D content acquisition hardware, for example, the Microsoft Kinect and the Makerbot Digitizer, to the consumer market allow non-professionals to easily acquire 3D content. These releases, coupled with consumer-targeted 3D manipulation software and 3D printers, encourage the adoption of 3D content and its technology by non-professionals.

3D content is more useful than 2D content, although more computational power is required to process 3D content. Since 3D content encodes complete spatial information, it is a more accurate representation and model for physical phenomena. For example,

Figure 1.1: A basic mesh representation (left) and a textured mesh (right)

a more accurate analysis can be done on an athlete's performance if his motions are captured and analyzed in 3D instead of 2D. In movies, computer generated worlds and characters are modeled in 3D before they are projected into a two-dimensional video representation for viewing.

A 3D mesh is a basic digital representation of a 3D model; other properties like color and texture can be added to a mesh to improve its realism. Fig. 1.1 shows a basic mesh representation and the same mesh when augmented with textures. 3D meshes are used in creating virtual scenes, which are in turn are used for applications such as Computer Generated Imagery (CGI) and virtual worlds.

Motion capture (mocap) data, on the other hand, specifies the motion of a moving skeletal structure, or skeleton, of a human or other animated objects. This is done by encoding the position of the skeleton joints at each point in time. Fig. 1.2 shows skeletons extracted from a human mocap sequence at different points in time. Mocap technology can capture human motions for analysis or viewing, thus it is helpful in a number of

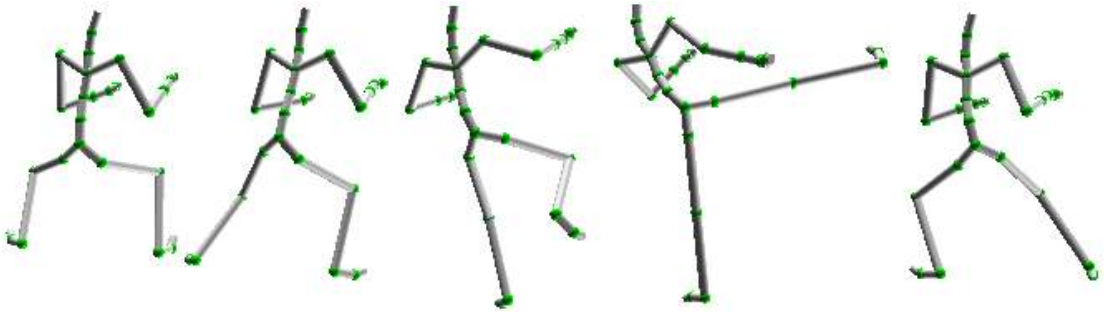


Figure 1.2: A stick figure illustration of a mocap data sequence, where the joints are green.

applications. For example, an athlete's training sequence can be captured and analyzed to find weaknesses in the athlete's pose and movement. Another example application is action recognition, where mocap technology is used to obtain and classify the pose of a human. For example, the action of elderly people falling down can be identified, and following that an alarm can be triggered [1].

A major application that draws upon both mocap and mesh data is the animation of 3D meshes using mocap data. Motion is extracted from human actors, which is then used to animate virtual characters in a realistic way. This technique is used to create computer generated characters, such as Gollum in the movie The Lord of the Rings. In video games, the method can be used to map a player's motions to player-controlled virtual characters.

Both mocap and mesh data can be manually created or acquired from physical samples. For mesh data, 3D modelling software can be used to draw a model; however, if a physical model is already available, it is more convenient to use 3D mesh scanning to obtain its mesh. For mocap data content creation, the trajectory of skeleton joints (specific points on a skeleton) can be specified to manually create a motion. This is, however, a laborious process and the results might not look realistic. Thus, mocap data is usually

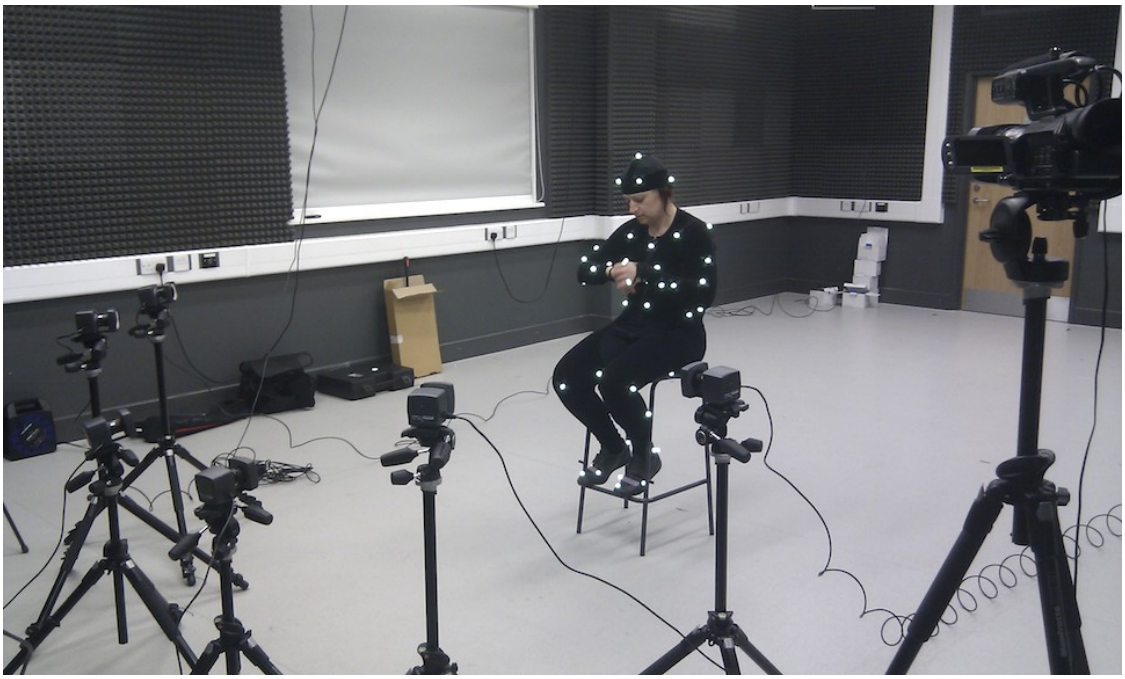


Figure 1.3: An optical-based mocap acquisition system

captured from live actors and then edited later if required.

In this thesis, we deal with issues related to the acquisition of 3D data. The acquisition of human mocap data involves a performer acting out the required motions while a mocap acquisition system captures the data. Most commonly used professional mocap acquisition systems, such as the Vicon System [2], utilize optical motion capture. In such systems, a performer has markers attached to his body and the trajectory of each marker is tracked using a set of surrounding cameras, such as shown in Fig. 1.3. A passive system uses reflective markers, whereas an active system uses LED markers. Less accurate and reliable systems, such as The Microsoft Kinect, which uses a RGB camera and a depth camera, can also be used directly for mocap acquisition [3]. Some works have demonstrated the capture of mocap data using a single RGB camera with the aid of machine learning [4, 5].

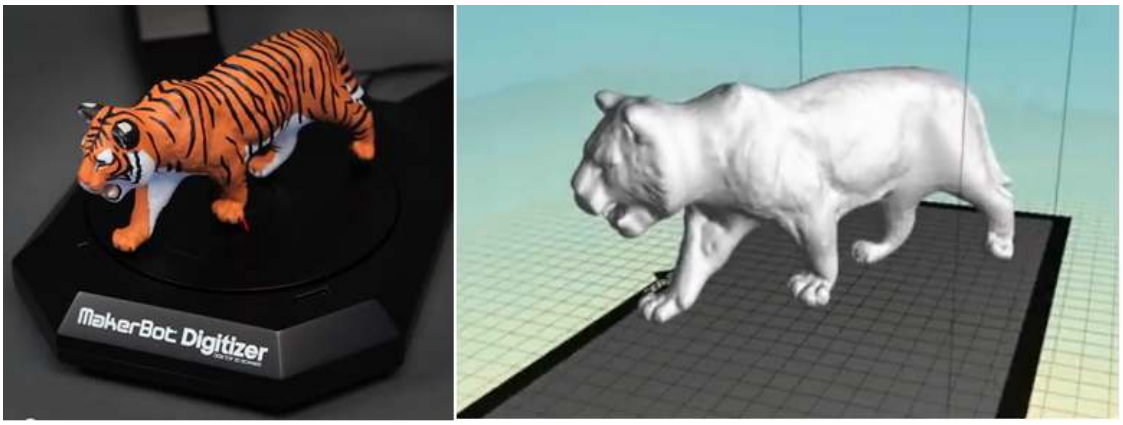


Figure 1.4: (left) A 3D mesh scanner scanning a physical model; (right) The acquired 3D mesh

After acquisition, mocap data has to be 'repaired' before it can be used, even if it is captured using professional systems. This is because the acquired data could be too noisy, missing, or it could contain outliers. The main reason for missing markers is occlusion by body parts or other objects. For outdoor mocap systems, environmental effects such as rain could affect the visibility of markers. Noisy or outlier entries could occur when two markers are very close together; the acquisition system could mistakenly identify a marker for another when two markers are very close together. Certain data entries could be too noisy, or missing altogether due to occluded markers or body parts. As an example, for the Kinect system, mocap data for an arm would be unavailable when the arm is swung behind the body. Hence, in this thesis, several methods are proposed to recover mocap data that is missing.

The acquisition of a 3D mesh is comparatively more straightforward; the object is placed on the receptacle of the scanner and scanned, such as shown in Fig. 1.4. Commercially available scanners include the Cyberware Model Shop 3D scanner and the Makerbot Digitizer.

After a 3D mesh has been acquired and repaired, it is often desirable to lower the

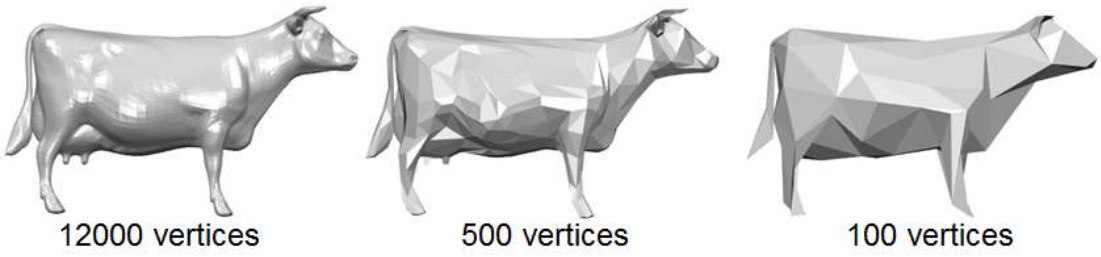


Figure 1.5: 3D mesh of a cow model at different resolutions

complexity, or resolution, of the mesh; reduction of mesh resolution is done using a process called mesh simplification. A model is captured by a regular sampling of its surface, resulting in a mesh with a roughly regular resolution across its surface. This resolution could be much higher than required for the application the mesh is used in. Furthermore, a mesh is more optimally represented, i.e., a mesh has a higher quality for a given overall resolution, if its resolution is allowed to vary across it's surface. Each part of a model requires a different resolution for accurate representation, depending on the complexity of the part. Compared with smoothly varying parts like the torso, detailed or complex features, such as the ears and the eyes, would require higher resolution to represent accurately. Through a process called mesh simplification, or mesh decimation, we can reduce the resolution of a mesh while striving to retain the fidelity of the represented model. The advantage of mesh simplification is that it lowers the computational time and memory required to process the mesh in a later stage, for example, in mesh rendering and editing. Fig. 1.5 shows a 3D mesh at different resolutions after mesh simplification. Although the cow mesh with 12000 vertices (a measure for mesh resolution) is more accurate, a 500 vertex model might be detailed enough. In this thesis, we proposed improvements to a mesh simplification method.

An application which utilizes both mocap recovery and mesh simplification is the animation of scanned 3D models. In the Pinocchio animation system [6], a skeleton is

automatically rigged in a humanoid-shaped 3D meshed model. This allows the 3D mesh to be animated by specifying the motion of the skeleton. The mesh representation of the model can be obtained by scanning a physical model, repairing any errors, and then simplifying the model to the required resolution. The user of the system can perform a set of motions which are captured with a motion capture system and represented as mocap data. Any missing data is restored using mocap recovery. The mocap data can then be used to animate the skeleton, which in turn animates the 3D mesh. The end result is an animated mesh that is able to imitate the motions of the user.

1.2 Contribution and Scope

This thesis aims to develop effective techniques for mocap data recovery and 3D mesh simplification. For mocap data recovery, we simulate missing joint data by randomly removing a number of entries from a mocap data. With the incomplete mocap data as input, we recover the missing joint data using mocap data recovery algorithms. For 3D mesh simplification, given high resolution meshes at up to 20000 vertices as input, we simplify the meshes to various target resolutions. Fig. 1.5 gives an illustration of a cow mesh simplified to different resolutions.

In terms of scope, for mocap data recovery, we limit our scope to human mocap data. We also focus on methods that do not require a database of prior motions. Although machine learning-based methods are likely to perform better than non-machine learning based methods, machine learning-based methods have a number of limitations. Besides the obvious requirement of a mocap database, machine learning based methods usually require the processed motion and prior motions to have the same marker set. The performance of these methods are also reliant on the similarity of prior motions and processed motions. If the processed motion is captured from an actor with a different size, or if

there are no prior motions of the same type as the processed motion, recovery performance might be degraded significantly.

For 3D mesh simplification, we focus on incremental decimation methods. This class of methods can be used for a wider range of applications, such as rendering and mesh transmission, since it can generate a progressive mesh. A progressive mesh is a structure from which meshes of continuously varying resolutions can be extracted.

The contributions of the thesis can be are as follows.

- Previously, low rank matrix completion with Singular Value Thresholding (SVT) optimization was applied on mocap data in a frame-based matrix representation to recover missing data in the sequence. We restructure mocap data into a trajectory-based representation for matrix completion instead, and show that the trajectory-based matrix has a lower rank. This enables matrix completion to recover mocap data more effectively. We show that the proposed trajectory-based SVT method achieves significant improvement over the previous frame-based SVT method for randomly missing data.

- The effectiveness of SVT matrix completion in recovering missing data decreases significantly when data entries are missing for extended periods of time. We show that longer periods of missing data are correlated with larger violations of the rigid skeletal structure. To alleviate this problem, we extend the frame-based SVT method to include skeleton constraints. We show that the proposed method improves on the frame-based SVT method significantly, especially when data entries are missing for an extended time.

- Frame-based SVT and trajectory-based SVT utilize different correlations; frame-based SVT relies on the correlation between different frames, whereas trajectory-based SVT relies on the correlation between trajectories. We propose to exploit

both types of correlation simultaneously by constraining the solution of trajectory-based SVT in the subspace spanned by the frames recovered using frame-based SVT. The proposed method is able to outperform both trajectory-based SVT and frame-based SVT for randomly missing data.

- Image-Driven Simplification (IDS) is very time consuming because it requires repeated renders and image captures from each of its 20 viewpoints during simplification. We propose to accelerate IDS by using only a single adaptively placed viewpoint instead of 20 statically placed ones. We propose two methods to place the viewpoint dynamically so that simplification performance is not compromised. The two proposed single viewpoint IDS methods run at an estimated twenty-five times faster than IDS at comparable simplification performance.

1.3 Organization of the Thesis

In the thesis, Chapters 3, 4, and 5 deal with mocap data recovery, whereas Chapter 6 covers mesh simplification. In more detail, the rest of the thesis is organized as follows.

Chapter 2 explains the background for mocap recovery and 3D mesh simplification. We review the representation for mocap data, and describe the current developments in mocap recovery. We also describe the 3D mesh structure and simplification methods.

Chapter 3 describes our work on using trajectory-based low rank matrix completion. We first explain the theory and intuition behind the matrix completion method, the method that forms basis of all our mocap recovery algorithms. We also explain the SVT method, which is the method used to solve the optimization problem posed by matrix completion. Next, we describe the proposed method, trajectory-based matrix completion, and compare it with the previous frame-based matrix completion.

In Chapter 4, we aim to improve the performance of matrix completion for data entries

that are missing for extended periods of time. To that end, we describe how to impose skeleton constraints in the matrix completion problem, and how to solve it using SVT.

In Chapter 5, we aim to effectively combine together two mocap recovery methods which exploits different characteristics of mocap data. We then describe our method which first recovers mocap data using frame-based SVT, and then use the output to guide mocap recovery of a subspace constrained trajectory-based SVT.

Chapter 6 describes the Image-Driven Simplification (IDS) algorithm and its deficiency which is long processing times. We explain how that IDS can be accelerated significantly if only one viewpoint is used instead of 20. Two efficient methods are proposed to find the viewpoint placement.

Chapter 7 concludes the thesis and suggests some avenues for future work.

Chapter 2

Background and Literature Review

2.1 Introduction

This chapter provides the background for human motion capture data and 3D mesh representation, and then reviews previous research done for motion capture (mocap) data recovery and 3D mesh simplification. We first describe the representations of mocap data in Section 2.2 and the literature review on mocap recovery methods in Section 2.3. In Section 2.4, we describe the sequences used and performance evaluation details for mocap recovery experiments. After that, we describe 3D mesh representations in Section 2.5. We then gave an overview of 3D mesh simplification and a brief coverage of literature for less related classes of simplification algorithms in Section 2.6, and a more in-depth coverage of previous works of the more relevant class of simplification in Section 2.6.1. Last, we wrap up with a Conclusion in Section 2.7.

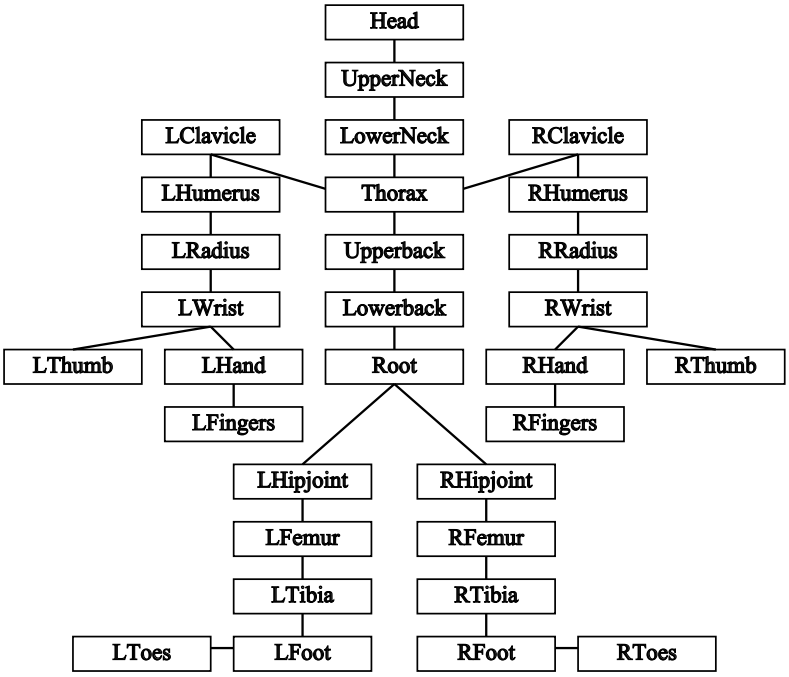


Figure 2.1: Mocap skeleton topology. Each label indicates a joint, and 'L' and 'R' prefixes denote left-sided and right-sided, respectively.

2.2 Motion Capture Data Representation

Human mocap data is used to describe the motion of a human over time. A mocap data sequence encodes the motion of a moving skeletal structure by specifying the pose, or frame, of the skeletal structure at each point in time. A skeletal structure, or skeleton, is a simple representation of an animated object, where each skeleton pose specifies the positions of a set of points (called joints), and the connectivity between the joints. In Fig. 1.2 (in Chapter 1), each joint is illustrated with a green sphere, and the bars connecting pairs of joints are referred to as bones. The structure and joints of the skeleton used in this thesis is shown in Fig. 2.1. Obviously all poses in a mocap sequence share the same skeleton structure.

There are two ways to represent a pose. The first representation, which we call joint-position representation, explicitly specifies the position of each joint. File formats supporting this representation include C3D. The second representation, which we call joint-angle representation, specifies the position and orientation of the root joint, and the angle of each child joint relative to its parent joint. For this representation, we can obtain the position of each joint using its angle and distance from its parent joint. Hence, the distance (bone length) between each parent-child joint pair has to be specified in the skeleton as well. File formats supporting joint-angle representation include ASF+AMC and BVH.

For professional motion capture systems, a number of markers are attached to the body of an actor. When the actor is performing the required motions, surrounding cameras acquire the positions of each marker over time. The positions of the markers are then used to generate the positions of skeleton joints. In some systems that do not use markers, such as the Kinect, joint information is directly acquired by processing the RGB and depth video. Thus mocap acquisition systems usually output mocap data with joint-

position representation. Thus, we will use this representation for mocap data. We also assume that the skeleton structure is known and does not need to be recovered; thus, in this thesis, mocap data encodes a set of skeleton poses, where each pose specifies the positions of all joints of the skeleton.

In motion editing, the joint-angle representation is used instead since it can be manipulated more naturally and easily. Instead of working on mocap data frame-by-frame, splines can be fitted to the angle of a joint across time, yielding a continuous representation that can be specified using fewer parameters.

2.3 Motion Capture Data Recovery Literature Review

Due to the imperfections of mocap acquisition systems, mocap markers or body parts might be occluded, and therefore certain mocap joint entries might be missing. Thus there is a need for mocap recovery methods.

A number of works have been proposed to recover missing mocap data. Wiley and Hahn [7] used interpolation to estimate missing data. To allow for real-time missing marker recovery, Piazza et al. [8] predicted whether the immediate motion is linear, circular or both, and then applied extrapolation for recovery. These methods recover each joint individually, and do not consider the fact that certain joints might be correlated with each other. Yii et al. [9] used a form of regression by modeling all data in a time window as a linear model and predicting missing elements using linear least squares. Papadimitriou et al. [10] improved on this method by using frequency analysis to automatically set the window size. Although the relationship between joints is also accounted for in this model, these two methods only use information from frames that are temporally near the mocap data to be recovered. Data that lie further away could still contain useful information to reconstruct missing data.

Some works utilize local linear models extracted from training data to predict missing data. Liu and McMillan [11] formed a global linear model and a set of locally linear models from a database. Missing values are initially recovered using the global model, then a classifier assigns a local model to each frame. By assuming that all frames lie in their respective models, a frame can be recovered by least squares estimation of model coefficients from available data, and then recovery of missing data using said coefficients. Instead of precomputing locally linear models, Chai and Hodgins [12] retrieved poses (frames) that are similar to the processed pose at runtime. These poses, which form a linear model, were used as a prior in energy minimization optimization to recover the processed pose. The pose retrieval efficiency of this framework was improved by Krüger et al. [13] by using a kd-tree to speed up retrieval, and the recovery performance was improved by Baumann et al. [14] by additionally considering the velocity and acceleration of each pose.

Since mocap data is not strictly linear, some researchers applied more complex models to model mocap data more accurately. Grochow et al. [15] learned a global nonlinear model to form a low-dimensional space using Gaussian Process Latent Variable Models (GPLVM) [16] and maximized a likelihood function over poses. Wang et al. [17] observed that GPLVM treated each mocap pose as independent data points, and proposed using Gaussian Process Dynamical Models, which considers the temporal structure of mocap data, to model mocap data instead. Lou et al. [18] extended the concept of linear model of poses to a linear model of consecutive sets of frames, i.e., a spatio-temporal model. With the aid of robust statistical methods, spatio-temporal bases from a training set were used to estimate missing values of mocap data. Taylor et al. [19] proposed that mocap data should be modeled using a non-linear model and used Conditional Restricted Boltzmann Machine to achieve this aim. Xiao et al. [20] proposed that a sparse representation of

a training set can be used to describe mocap frames. Given an incomplete frame, they solved for the sparsest representation for the frame with the constraint that the recovered frame is as similar as possible to the incomplete frame. This method was improved by Hou [21] by using a trajectory-based representation for the sparse representation.

The methods described above require a training database to learn a mocap data model. Such methods have a number of limitations; they require the processed data and training data to have the same marker set. The performance of these methods are also reliant on the similarity of training data and processed data. If the processed data is captured from an actor with a different size, or if there are no prior motions that are similar to the processed motion, recovery performance might be degraded significantly.

There exists several methods that attempt to directly create a model from the processed mocap data. Li et al. [22] modeled the entire processed sequence as a linear dynamical system; a sequence of frames is modeled as a sequence of latent variables of lower dimension. Consecutive latent variables are related by a linear map, and each frame is related to a latent variable by a linear projection. The objective function to be maximized is the likelihood of the observed data with respect to the model. The solution is computed using expectation maximization: iteratively estimate the latent variables and model parameters, and estimate the missing values until convergence. The method BOLERO [23] improved on this system by enforcing bone length constraints for the recovered data. This is done by including bone constraints in the objective function. Since the bone constraints are formulated as nonlinear constraints, the objective function becomes nonlinear; thus the method is computationally slow.

Lai et al. [24] arranged mocap data to form of an input matrix with missing entries. They obtain the recovered matrix by solving for the lowest ranked matrix that conforms to the observed entries of the input matrix. This is the matrix completion approach [25],

and they solve it using Singular Value Thresholding [26]. Since this approach models the data recovery as a convex problem, convex optimization can be used to solve the problem; hence it is faster than methods posing the recovery problem as a nonlinear one. A faster but less accurate approach of obtaining a low rank approximation of an incomplete matrix is proposed by Srebro and Jaakkola [27].

A related class of works deal with recovering occluded sensor markers during optical motion capture, and are tailored for the motion capture system used. Thus, the methods cannot be directly used for alternative motion capture systems. Herda et al. [28, 29] use a pre-specified skeleton model to estimate the position of missing neighboring markers. In the system proposed by Hornung et al. [30], sets of markers with fixed relative positions are found automatically without a manually specified skeleton structure; this information is used to aid missing marker estimation. In some works, extended Kalman filters [31, 32] and unscented Kalman filters [33] are used to track markers and to recover any missing markers.

2.4 Motion Capture Data Recovery Experimental Settings

We will be using a set of mocap sequences in our experiments in the next few chapters; thus we first describe the sequences and performance evaluation details here.

In all of our experiments, we use mocap sequences from the CMU mocap database [34]. Each sequence contains 31 joints (see Fig. 2.1), and is sampled at 60 frames per second (fps) or 120 frames per second. In order to maintain the same frame rate for all sequences, we downsample 120 fps sequences down to 60 fps.

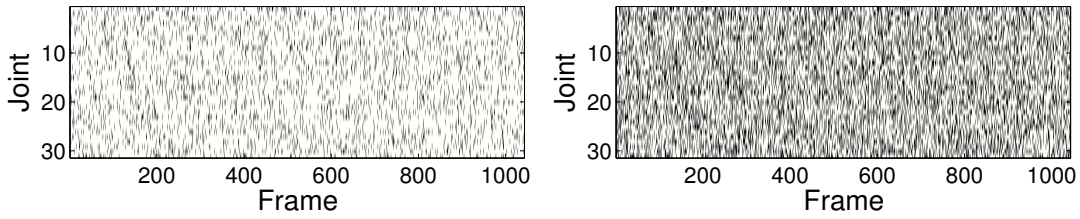
We select two sets of sequences; the first set is used for estimating parameters in our proposed algorithms (Table 2.1), whereas the second set is used for performance evaluation (Table 2.2). The number of frames shown are the number of frames under

Sequence	Description	Number of frames
20_01	Chicken dance	642
54_01	Monkey actions	960
56_01	Walk around	753
135_10	Syutouuke martial arts walk	1000
143_41	Sneak	583
42_01	Stretch, rotate, head, arms, legs	1134
55_01	Dance, whirl	903
85_01	JumpTwist	499
85_06	Kickflip	475
143_24	Kicking	502

Table 2.1: Sequences used for parameter estimation.

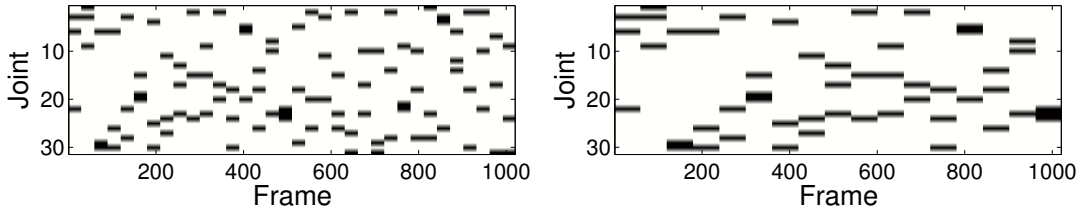
Sequence	Description	Number of frames
14_01	Boxing	2797
143_04	Run figure 8	543
135_02	Empi martial arts	2601
61_05	Salsa dance	790
85_02	Jumptwist	405
49_02	Jumping and one foot hopping	1043
135_11	Yokogeri martial arts walk	1223
88_04	Acrobatics	1240

Table 2.2: Sequences used for performance evaluation.



(a) 10% missing data (b) 30% missing data

Figure 2.2: Random missing data mask



(a) $l = 30$ (b) $l = 60$

Figure 2.3: Block missing data mask for 3 missing joints. l is the length of each block of missing data

60fps sampling rate. We select sequences of different complexity, encompassing a variety of actions, and of different lengths (from 6 seconds to 46 seconds). Here, we loosely define an action as a simple, short motion, such as a punch or a kick; a complex motion can contain a number of different actions.

In the performance evaluation mocap set, sequences of vastly different length are chosen to better evaluate computational efficiency of the tested algorithms. With regards to motion complexity, except for 85_02, 135_02, and 88_04, which consist of a variety of different actions, the other sequences consist of repetitions of similar action. For example, in sequence 14_01, the figure performs repeated punching actions of different types, such as straight punch and uppercut. On the other hand, in 135_02, the figure performs a variety of poses, kicks, and punches.

We simulate missing data by removing joint entries $\{x, y, z\}$ from mocap sequences

using two methods. In the first method, which we call *random missing data*, missing data is simulated by randomly removing a number of joints so that the required missing data rate is simulated. Sample masks for random missing data at 10% and 30% missing data rates are shown in Fig. 2.2. Entries for mocap sequences are removed at corresponding blacked out areas. Note that since each joint has 3 entries (x, y, z coordinate) in a mocap sequence, each removed entry in a missing data mask corresponds to 3 entries in the mocap sequence. For random missing data, missing entries at lower missing data rates are a subset of the missing entries at higher missing data rates.

In the second method, which we call *block missing data*, we partition a sequence into time intervals of equal length l, and for each interval we randomly choose a set of joints to remove. Each joint remains missing for the entire duration of an interval. Sample masks for block missing data at $l = 30$ and $l = 60$ for 3 missing joints (equivalent to $\frac{3}{31} \approx 10\%$ missing data rate) are shown in Fig. 2.3. Note that for this case, the missing entries for shorter interval lengths are not a subset of missing entries for longer interval lengths. For mocap sequences with frames that are not a multiple of l, we truncate them so that they are a multiple of l frames.

Block missing data allows us to judge the robustness of mocap recovery methods towards joints missing for extended periods of time. The interval length l can be increased to allow us to judge the robustness of mocap recovery towards such 'bursty' errors under the same error rate. The number of removed joints determines the error rate of the corrupted mocap, e.g., removing three joints induces an error rate of $\frac{3}{31} \approx 10\%$.

We use Root Mean Square Error (RMSE) to quantify the distortion of a recovered matrix.

$$\text{RMSE}(\hat{\mathbf{X}}, \mathbf{M}) = \sqrt{\frac{1}{n_1 \times n_2} \sum_{j=1}^{n_2} \sum_{i=1}^{n_1} \left(\hat{\mathbf{X}}_{ij} - \mathbf{M}_{ij}\right)^2} \tag{2.1}$$

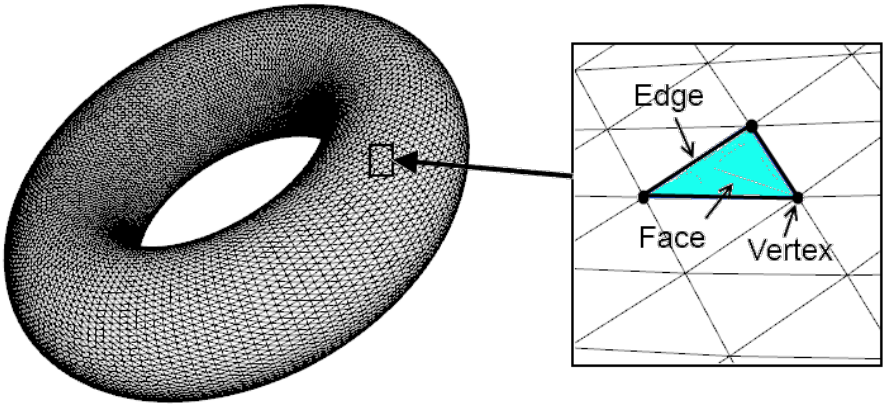


Figure 2.4: An illustration of a vertex, edge, and face of a 3D mesh.

where $\hat{\mathbf{X}}$ and $\mathbf{M}$ are the matrices of the recovered and original mocap respectively. All mocap data values shown in the thesis are expressed in the provided default units in the CMU mocap database. To obtain values in millimeters, multiply the values by 56.44. To get a better idea of the significance of the values, we can consider the height of the skeletons, which is around 26.6 units (1.5 meters).

2.5 3D Mesh Representation

There are a number of ways to mathematically describe the surface of a 3D model. In CAD applications, for example, usually spline functions are used to model the surface. However, the most common representation for 3D models is the triangular mesh, which uses a set of triangles to model a surface in a piecewise manner. In this thesis, we use the term 3D mesh, or just mesh, to refer specifically to the triangular mesh.

A 3D mesh is fully described by two components, a set of vertices, $\mathcal{V} = \{v_1, \ldots, v_V\}$, which are points in 3D space, and a set of triangular faces, $\mathcal{F} = \{f_1, \ldots, f_F\}$, $f_i \in \mathcal{V} \times \mathcal{V} \times \mathcal{V}$, where each face specifies the connectivity of three vertices. Each vertex specifies a position in 3D space, $p_{v_i} = [x_{v_i}, y_{v_i}, z_{v_i}]^T$. Sometimes, instead of faces, connectivity

information is represented using a set of edges, $\mathcal{E} = \{e_1, \ldots, e_E\},\ e_i \in \mathcal{V} \times \mathcal{V}$, where each edge connects two vertices. An illustration of the vertices, edges and faces of a mesh is shown in Fig. 2.4.

The complexity or resolution of a 3D mesh is defined by the number of vertices, edges or faces that it has. The number of vertices V, edges E, and faces F of a closed and connected mesh is related by Euler's formula

$$F + V - E = 2(1 - g)$$

where g is the genus, or number of handles, of an object. Since the genus of most meshes is small, it is sufficient to use either the number of vertices, edges, or faces to quantify mesh resolution. In this thesis, we use the number of vertices as a measure of mesh resolution.

2.6 3D Mesh Simplification Literature Review

A 3D mesh obtained through 3D scanning or tomography could have a complexity that is much higher than required for its application; this unnecessarily high complexity slows down processing operations on the mesh. Thus, after mesh acquisition and removal of mesh errors, the complexity of a mesh is reduced through a process called mesh simplification or mesh decimation.

Mesh simplification works by either generating a mesh with the lowest distortion subject to the required number of vertices, or by generating a mesh with the smallest number of vertices subject to a certain quality criteria. Mesh simplification algorithms can be classified into three main categories [35] — vertex clustering algorithms, remeshing algorithms, and incremental decimation algorithms. We will focus more on incremental

decimation algorithms since it is the most relevant to our work, but we will also briefly describe the other classes of methods.

Vertex clustering algorithms are usually very fast, with ($O(n)$) complexity, and robust. However, these algorithms, compared with other methods, generate simplified meshes with higher error. The initially proposed method [36] works by superimposing a grid on a model and assigning an importance value to each vertex. It then clusters all the vertices in each grid cell into the vertex with the highest importance value. Low and Tan [37] sorted the vertices according to their importance value to determine the clustering order and ensured that two important vertices are not clustered together. Lindstrom [38] used a quadric error metric for all faces in a grid cell to determine the position of a cell.

Remeshing methods aim to generate meshes that are regular, where mesh faces are nearly the same size, and each vertex has the same number of neighbors. The regularity of the resulting mesh allows a natural 2D parameterization to be defined on 3D surfaces, thus remeshing is used in mesh parameterization algorithms. This regularity also makes the mesh easier to compress; hence remeshing is also helpful in applications like mesh transmission [39].

Remeshing methods resample a mesh and reconnects the sampled points in a regular way. Hence, remeshing can also generate a lower resolution mesh by under-sampling a mesh. The connectivity structure can be constrained to obey certain patterns, depending on the required application. For example, a mesh can be remeshed to have isotropic, or nearly equilateral, triangles [40], quadrilaterals [41], or squares [42]. Instead of using vertices to represent a resampled mesh, Cohen-Steiner [43] proposed to use a small set of simple geometric shapes as proxies to represent the mesh. The interested reader could refer to this literature [44] for a more complete survey of remeshing.

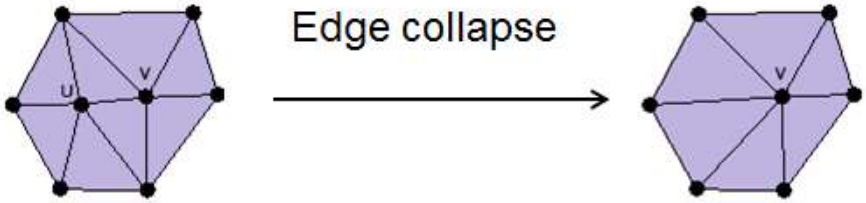


Figure 2.5: An illustration of an edge collapse operation.

2.6.1 Incremental Decimation

Hoppe [45] introduced the incremental decimation framework where a mesh is simplified incrementally by removing a vertex at a time until the target vertex count is reached. Incremental decimation has a very useful characteristic: it can generate a sequence of simplified meshes, where each mesh has one less vertex than the previous mesh. This characteristic has two useful implications. First, if the vertices of a lower resolution mesh is a subset of the vertices of a higher resolution mesh, a straightforward multi-resolution representation of a mesh can be encoded easily. Second, the appearance of a mesh changes smoothly while the resolution of the mesh is scaled up or down. The first implication facilitates compression and progressive transmission [46] of a mesh, whereas the second implication enables accelerated rendering through a system called Level of Detail (LoD) [45].

In incremental decimation, the vertex is removed through an operation called edge collapse (see Fig. 2.5); it is merged with an adjacent vertex (a vertex that it shares an edge with) to form a new vertex. In half-edge collapse, the new vertex takes the position of one of the parent vertices, whereas in full-edge collapse, the new vertex is subsequently moved to a more optimal position. While full edge collapse can yield higher quality meshes, the resulting vertex is not a subset of its parent vertices. Hence, systems relying on this characteristic will need to compute and encode the position of the new vertex,

and so will be less efficient. In this thesis, we employ the half-edge collapse. After the two vertices are merged, the connectivity information are updated so that the new vertex has the same neighboring vertices as the two parent vertices.

Edge collapse operations are applied to edges in a greedy manner — at each step, the vertex pair that causes the least error after collapse is chosen to be collapsed. This error is usually defined as $E(\hat{M}, M^k)$, where $\hat{M}$ is the original mesh, and M^k is the mesh after the kth collapse. How this error is computed determines the sequence of edges to be collapsed; hence the edge collapse error metric determines the performance of incremental decimation.

Initially, euclidean distances between two mesh surfaces are used to measure edge collapse error. Garland and Heckbert proposed the widely used quadric error metric [47], which efficiently computes the distance between planes of the original and simplified mesh. Later, Lindstrom and Turk [48] used a memoryless error metric computed on volume distortion, which defines edge collapse error using $E(M^{k-1}, M^k)$, i.e., the candidate mesh is compared with the mesh in the previous step instead of the original mesh. Although this definition is unintuitive, experiments showed that the method performs well.

Since the appearance of a mesh is also influenced by other mesh properties, such as normals and texture coordinates, some works tried to account for these properties. The quadric error metric was augmented with normals, colors, and texture coordinates in [49, 50]. The errors of the various measures are weighted and summed to obtain the edge collapse error. More recently, Wei [51] incorporated normals into the metric using feature-sensitive space. Combining spatial error and mesh appearance property error together into a single error metric is difficult; these errors are different in nature and have perceptual affects on each other.

There are also methods that focus exclusively on normals or curvature as an error

metric. Hussain [52] selected vertices for collapse based on the variation of normals in a local area. Kim et al. [53] used the distortion in local curvature as the error metric. To preserve texture appearance, texture was adapted to each edge collapse during simplification [54, 55].

Other researchers [56, 57, 58] mapped the mesh surface onto a parametric domain. The algorithms try to minimize the attribute (either spatial position or normal vector) differences between corresponding points in the parametric domain of the original and simplified mesh. These methods, however, are susceptible to parameterization distortion; mesh attributes are not perfectly mapped to the parametric domain.

Instead of computing spatial error and mesh appearance error separately, and then trying to combine them, Lindstrom and Turk [59] proposed the image-driven simplification method, which directly measures the visual appearance of a mesh. Visual appearance is obtained using a set of images captured from an evenly distributed set of cameras around the mesh. Thus, this method does not need to deal with how various measures of a mesh contributes to the visual appearance of the mesh.

Since image-driven simplification was introduced, several other methods that utilize images captured from surrounding viewpoints were proposed. Zhang [60] used these images to obtain a visibility measure of all faces in the mesh. This measure is used to weigh the quadric error metric to prioritize collapses for edges that are unlikely to be seen (e.g., interior regions of a mesh). Castello [61] applied an information-theoretic measure on the viewpoint-projected face area. A mutual information measure is defined between the set of faces and the set of viewpoints. Following that, edge collapse error is defined as the total variation of mutual information for all viewpoints.

As mentioned earlier, incremental decimation is useful for the level-of-detail (LOD). View-independent LOD [45] encodes a mesh using incremental decimation, and, at low

computational cost, extracts the mesh at varying resolutions. In applications like video games, view-independent LOD is used to improve rendering speed; when the mesh is a large distance away from the viewpoint, and the details of the mesh are imperceptible, the resolution of the mesh is decreased.

Later works introduced a new class of LOD known as view-dependent LOD [62, 63, 64], along with a variation of incremental decimation required to support it. Unlike the view-independent LOD framework, this system can simplify a mesh with non-uniform resolution over the mesh. In this system, an error threshold that varies over the mesh is computed at render time, and edges are collapsed until this threshold is reached. For more efficient rendering, the extracted mesh is set to a high resolution at areas near the viewpoint and to a low resolution at areas further away from the viewpoint. Hence, view-dependent LOD generates a perceptibly higher quality mesh than view-independent LOD for a given vertex count, but at a much higher processing cost during mesh extraction.

For more information regarding LOD and simplification, we recommend the following literature [65, 66, 35].

2.7 Conclusion

In this chapter we provide the background to our work in following chapters. We describe the different motion capture data representations when used for different applications. We then cover the literature review of motion capture data recovery and briefly compare the two categories of methods — methods that require a database of motions for training, and methods that do not have such requirements. We also describe the mocap sequences that are used in the following chapters and missing data masks that are used to simulate missing entries in mocap data. After that, we describe the triangular 3D mesh representation used in our works. We then describe the different classes of 3D mesh simplification.

We give a brief coverage of literature for the vertex clustering and remeshing class of simplification algorithms and a more in-depth treatment of previous works for incremental decimation, which is more related to our research.

Chapter 3

Mocap Recovery using Trajectory-based Singular Value Thresholding

3.1 Introduction

Human mocap data is frequently used in movies, video games, virtual worlds, sports and other domains that require synthesis and analysis of natural human animation. Mocap data is usually obtained from humans who perform the required motions; usually via markers attached to actors, which provide information on the trajectory of the actor's movements. However, even for professional systems, the mocap acquisition process is not perfect — acquired data could be noisy or incomplete due to occluded markers.

Lai et al. [24] arranged mocap data as a sequence of frames in matrix form and then recovers the missing data entries by solving a low rank matrix completion [25] problem to recover the matrix; the Singular Value Thresholding algorithm [26] was used to solve

the problem.

Since low rank matrix completion recovers missing matrix entries more reliably when the rank of the matrix is low, we propose to reduce the rank of the mocap matrix by representing it as a sequence of trajectory segments instead. We show that the rank of the trajectory-based matrix is indeed lower than its frame-based counterpart. We also show that the optimal trajectory segment length is dependent on the fraction of missing mocap data and suggested a model for this dependency. Using segment lengths derived from our model, our proposed trajectory-based Singular Value Thresholding method shows substantial improvement over the previous frame-sequenced method in mocap recovery. We are able to achieve up to 80% improvement for some sequences, and 68% improvement on average.

We first describe the low rank matrix completion method and Singular Value Thresholding, the implementation of low rank matrix completion that we use, in Section 3.2. We then describe our trajectory-based matrix completion in Section 3.3. In Section 3.4, we present experimental results and wrap up with a conclusion in Section 3.5

3.2 Matrix Completion

Given a matrix with a number of missing entries, how can we recover the complete matrix? If it is known that the matrix is low ranked (or the matrix has a low rank), low rank matrix completion theory [25] says that the solution is the lowest ranked matrix that matches the observed entries. Put in another way, the missing values should be filled in such a way that the resulting matrix has the lowest possible rank.

The rank of a matrix is defined as the number of linearly independent columns (or rows) of the matrix. In other words, the rank of a matrix $\mathbf{X}$ is the size of the set of maximally linearly independent vectors that can be used to reconstruct each column of

$\mathbf{X}$, i.e.,

$$\mathbf{x}_i = \sum_{j}^{k} \alpha_j \mathbf{c}_j, \quad \forall \mathbf{x}_i \tag{3.1}$$

where $\mathbf{x}_i$ is the ith column of $\mathbf{X}$, k is the rank of $\mathbf{X}$, and $\mathbf{c}_j \in \mathcal{C}$, where $\mathcal{C}$ is the set of k maximally linearly independent vectors, also known as a basis set. We can view $\mathcal{C}$ as the set of factors that contribute to $\mathbf{X}$; the degrees of freedom of the entries of $\mathbf{X}$ are limited by k, that is, the size of $\mathcal{C}$. Intuitively, a small k limits the degrees of freedom of the entries of $\mathbf{X}$; this means that a partial observation of its entries is sufficient to recover a low rank matrix $\mathbf{X}$.

The low rank property of a matrix is dependent on the value of k relative to the dimensions of the matrix, and not the absolute value of k. This is because the matrix dimensions limit the maximum rank of the matrix. A matrix with dimensions $n_1 \times n_2$ has a maximum rank of $\min(n_1, n_2)$, that is, the smallest dimension of the matrix. If a matrix has $k = \min(n_1, n_2)$, it has the maximum rank possible; it is a full ranked matrix. Assuming matrices where $n_2 > n_1$, we can say that a matrix with $k = 10$ and $n_1 = 93$ is low ranked, but a matrix with $k = 10$ and $n_1 = 11$ is not, since it is nearly full ranked.

We note here that there is no standard definition or threshold on how small the value of k should be (relative to the maximum possible rank) for a matrix to be qualified as low ranked. Furthermore, matrices of mocap data are only approximately low ranked; strictly speaking, they are full ranked, but can be approximated very closely by a low ranked matrix. Nevertheless, these matrices can still be accurately recovered by low rank matrix completion.

From a vector space point of view, for Eq. 3.1, we can also say that a k ranked matrix has columns (or rows) that lie in a k-dimensional subspace, where this subspace is maximally spanned by the vectors in $\mathcal{C}$. The matrix is low ranked if the subspace

dimension k is small relative to the dimension of $\mathbf{x}_i$, n_1.

Mocap data has been viewed as a set of points in a low dimensional subspace in previous works on mocap recovery. For example, in [11], the basis set $\mathcal{C}$ is retrieved from a mocap database to form a low dimensional linear model, the values of α are estimated from the observed entries for each $\mathbf{x}_i$, and then the missing entries in $\mathbf{x}_i$ are obtained using α, by using Eq. 3.1. We can also interpret matrix completion as estimating the basis set $\mathcal{C}$ directly from the input matrix, which is the smallest basis set that can form $\mathbf{X}$, and then estimating α and the missing entries of $\mathbf{X}$.

Although it is not wrong to interpret Eq. 3.1 in terms of row vectors, i.e., $\mathbf{x}_i^T = \sum_j^k \alpha_j \mathbf{c}_j^T, \ \forall \mathbf{x}_i^T$, where $\mathbf{x}_i^T$ is the row i of matrix $\mathbf{X}$, it is not helpful to do so when $n_2 \gg n_1$, which is the case for all matrices used to represent mocap data in this thesis. The dimension of a subspace k required to contain a set of vectors is always smaller or equal to the number of vectors in the set. Since the number of row vectors in $\mathbf{X}$ is n_1, then $k \leq n_1$. If the matrix has dimensions such that $n_2 \gg n_1$, then $n_2 \gg k$. Thus for any matrix with such dimensions, row vectors will always lie in a low-dimensional (compared to row space dimension, which is n_2) subspace, but the matrix is not necessarily low ranked since the maximum possible rank of the matrix is $\min(n_1, n_2) = n_1$. Thus it is not useful to use row vectors in Eq. 3.1 to find out whether a matrix has low rank, and the applicability of low rank matrix completion on said matrix. Intuitively, if we have k vectors spanning a k-dimensional subspace, all the vectors are linearly independent, and we do not have any redundancy to recover missing data in any of these vectors.

Matrix completion can be used to estimate missing data in cases where the data can be properly represented in the form of matrix, and that the nature of the data causes the matrix to be low ranked. In practice, although a data matrix is almost never exactly low ranked, an approximately low ranked matrix is sufficient for matrix completion. For

example in a movie recommender system, matrix completion can estimate missing entries of a matrix that represents user ratings of various movies, where users correspond to rows and rated movies correspond to columns of the matrix. The matrix is low ranked since only a few factors contribute to a user's movie preference. Matrix completion has also been used for video denoising [67], where pixels deemed noisy are discarded and then recovered using matrix completion, and medical imaging [68], where data is sparsely sampled and recovered using matrix completion.

However, not all low-ranked matrices can be recovered; the matrices have to fulfill additional criteria. First, the sampled entries should be uniformly distributed, and it is necessary that there is at least one observation for each row and one observation for each column. Second, the singular vectors should be sufficiently spread to minimize the number of sampled entries required for recovery, i.e., the singular vectors should be uncorrelated with the standard basis.

As shown in [25], matrix completion recovers the matrix exactly with high probability provided that the number of observed entries m is sufficiently large, or if the matrix rank r is sufficiently small, i.e.,

$$m \geq Cn^{6/5} r \log n \tag{3.2}$$

where C is a constant, and $n = \max(n_1, n_2)$ where (n_1, n_2) are the dimensions of the matrix. Eq. 3.2 shows that a matrix with a lower rank can be recovered more reliably.

3.2.1 Singular Value Thresholding

The solution for matrix completion is the lowest ranked matrix that matches the observed entries, i.e.,

$$\begin{aligned} \text{minimize} \quad & \text{rank}(\mathbf{X}) \\ \text{subject to} \quad & \mathbf{X}_{ij} = \mathbf{M}_{ij} \quad (i,j) \in \Omega \end{aligned} \tag{3.3}$$

where $\mathbf{X}$ is the decision variable, rank($\mathbf{X}$) is the rank of $\mathbf{X}$, and Ω is the set of locations of the observed entries $\mathbf{M}_{ij}$ of input matrix $\mathbf{M}$.

Since this problem is NP-hard, an alternative and equivalent optimization problem is proposed [25], i.e.,

$$\begin{aligned} \text{minimize} \quad & \|\mathbf{X}\|_* \\ \text{subject to} \quad & \mathbf{X}_{ij} = \mathbf{M}_{ij} \quad (i,j) \in \Omega \end{aligned} \tag{3.4}$$

$\| \bullet \|_*$ denotes the nuclear norm, which is the sum of singular values of a matrix arranged in decreasing order, i.e.,

$$\|\mathbf{X}\|_* = \sum_{k=1}^{n} \sigma_k(\mathbf{X}) \tag{3.5}$$

Problem 3.4 is convex and can thus be solved efficiently using convex optimization. Matrix completion is thus a class of methods that recovers a matrix by solving Problem 3.4. There are a number of algorithms to solve Problem 3.4, e.g., the Alternating Direction Method of Multipliers (ADMM) [69] and Fixed Point Continuation with Approximate SVD (FPCA) [70].

We solve this problem by using the same algorithm as Lai et al. [24], which is the Sin-

gular Value Thresholding (SVT) algorithm [26]. SVT minimizes $f(\mathbf{X}) \coloneqq \tau\|\mathbf{X}\|_* + \frac{1}{2}\|\mathbf{X}\|_F^2$, where $\|\mathbf{X}\|_F$ is the Frobenius norm of $\mathbf{X}$, instead of $f(\mathbf{X}) \coloneqq \|\mathbf{X}\|_*$; thus, this algorithm computes an inexact solution, but it is efficient in both time and memory. Obviously, a more accurate solution can be obtained by setting a larger value for parameter τ, but doing this comes at cost of longer processing time.

Algorithm 1 Singular Value Thresholding

Input: $\mathbf{M}$, location of observed entries Ω
Parameter: δ, τ, $k_{\max}$, *tol*
Output: Recovered mocap sequence $\hat{\mathbf{X}}$
Initialization:
$\mathbf{Y}^0 \leftarrow \delta P_\Omega(\mathbf{M})$
$k \leftarrow 1$

while $k < k_{max}$ **and** $\text{RMSE}(P_\Omega(\mathbf{X}^k), P_\Omega(\mathbf{M})) > tol$ **do**
    $\mathbf{X}^k = \mathcal{D}_\tau(\mathbf{Y}^{k-1})$ // update $\mathbf{X}$
    $\mathbf{Y} \leftarrow \mathbf{Y}^{k-1} + \delta P_\Omega(\mathbf{M} - \mathbf{X}^k)$ // update $\mathbf{Y}$
    $k \leftarrow k + 1$
end while
$\hat{\mathbf{X}} \leftarrow \mathbf{X}^k$

function $(P_\Omega(\mathbf{X}))$
    if $(i, j) \in \Omega$ **then**
        $\mathbf{Y}_{ij} = \mathbf{X}_{ij}$
    else
        $\mathbf{Y}_{ij} = 0$
    end if
    Return $\mathbf{Y}$
end function

function $(\mathcal{D}_\tau(\mathbf{X}))$
    $\sum_i \sigma_i \mathbf{u}_i \mathbf{v}_i^T \leftarrow \mathbf{X}$ // singular value decomposition
    $\mathbf{Y} \leftarrow \sum_i \max(\sigma_i - \tau, 0)\mathbf{u}_i \mathbf{v}_i^T$
    Return $\mathbf{Y}$
end function

The algorithm for SVT is presented in Algorithm 1. Increasing the value for step size δ will lower the number of iterations to convergence, but if the value is too large,

the solution might diverge. The algorithm terminates when the intermediate recovered matrix $\mathbf{X}^k$ has a low error (which is thresholded by *tol*) compared with the observed entries of $\mathbf{M}$, or when the number of iterations has reached k_{max}.

3.3 Proposed method

A mocap sequence is composed of a sequence of frames where each frame holds the 3D coordinates $\{x_i, y_i, z_i\}$ of a set of joints; as such, a mocap sequence with n_2 frames and j joints can be represented in the form of a matrix $\mathbf{M} \in \mathbb{R}^{n_1 \times n_2}$, $n_1 = 3j$

$$\mathbf{M}_{\mathrm{f}} = [\mathbf{f}_1\ \mathbf{f}_2\ \ldots\ \mathbf{f}_n] \tag{3.6}$$
$$\text{where } \mathbf{f} = [x_1\ y_1\ z_1\ x_2\ y_2\ z_2\ \ldots\ x_j\ y_j\ z_j]^T$$

We denote this matrix as a *frame-based* representation. Intuitively, throughout a mocap sequence, the poses that are adopted by a person has limited degree of freedom. For example, a punching sequence might contain a ready pose, arm outstretch pose, and arms pulled in pose. The limited degree of freedom of the set of poses means that the rank of the matrix is low. From the perspective with regard to Eq. 3.1, each frame $\mathbf{x}_i$ could be constructed relatively accurately from just $\mathbf{c}_i$ consisting of the three poses mentioned.

Other researchers have done studies regarding the low rank characteristic of frames of mocap data. Barbic et al. [71] showed that frames belonging to the same action lie in a low-dimensional subspace. This means that these frames are highly correlated, and that the frame-based mocap matrix has a low rank. This low rank property of the frame-based representation makes it suitable for matrix completion, as used by Lai et al. [24].

Since matrix completion performs better on matrices with lower rank (Eq. 3.2), we aim

to lower the rank of the mocap matrix. First, we suggest that frame-based representation has high frame correlation only for motion sequences that, a) have a small variation of frames, which means that all frames can be grouped, with low error, into a small set of clusters, and b) are highly repetitive. In fact, Barbic et al. [71] segmented motion into different action sequences by exploiting the fact that a combination of two different actions has a significantly higher dimensionality than either of the individual sequences.

Next, we suggest that trajectories of different joints over a short time window are highly correlated. The intuitive reason is that human joints are arranged in a hierarchical manner; the motion of a joint is dependent on the motion of its parent joint. For example, when we swing a punch, the shoulder, elbow and wrist will follow a similar trajectory of an arc. Furthermore, some sequences perform similar repeated actions, thus there is high correlation between trajectories of these actions.

Hence, in general, it is likely that a representation based on trajectory segments have lower rank than the frame-based representation. We thus arrange the mocap data as a sequence of trajectory segments $\mathbf{t}$, i.e.,

$$\begin{aligned} \mathbf{M_t} =& [\mathbf{t}_1\ \mathbf{t}_2\ \ldots\ \mathbf{t}_k] \\ \text{where } \mathbf{t} =& [x_{i,s}\ y_{i,s}\ z_{i,s}\ x_{i,s+1}\ y_{i,s+1}\ z_{i,s+1}\ \ldots\ x_{i,s+L-1}\ y_{i,s+L-1}\ z_{i,s+L-1}]^T, \\ k =& n_1 \times n_2/3L \end{aligned}$$

where L is the length of a trajectory segment. Each $\mathbf{t}$ is the trajectory segment of a joint from frame s to frame $s + L - 1$. We denote this matrix as a *trajectory-based* representation. Fig. 3.1 shows a graphical comparison of frame-based and trajectory-based mocap matrices. Each block represents the data of a joint for L frames. Thus instead of operating on a set of frames, matrix completion operates on a set of joint

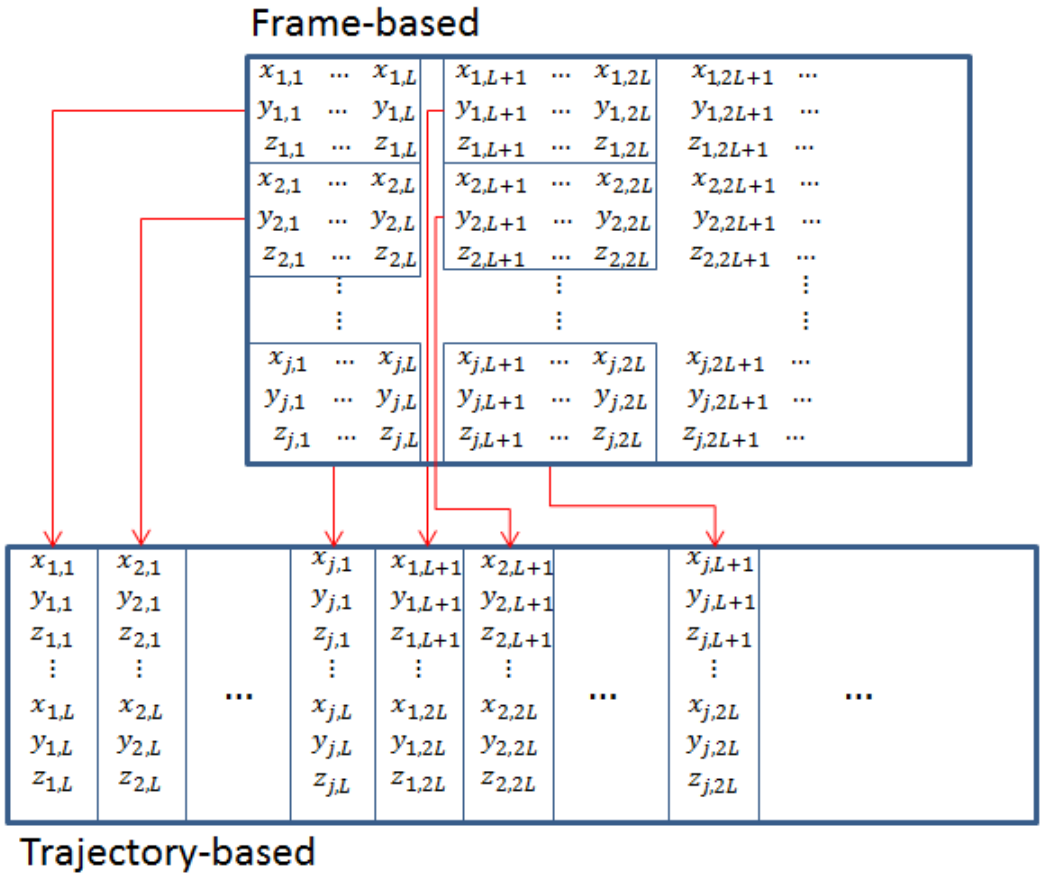


Figure 3.1: A comparison of the structure of frame-based and trajectory-based representations of a mocap matrix

trajectories.

Since matrix completion does not care whether a row or column corresponds to an independent data unit, recovery of a frame-based matrix $\mathbf{M}_{\mathrm{f}}$ and its transpose $\mathbf{M}_{\mathrm{f}}^T$ is equivalent and produces the same result. However, it is less accurate to interpret a column of $\mathbf{M}_{\mathrm{f}}^T$ as a proper trajectory, since each column corresponds to x, y or z dimension of a joint and lacks physical meaning. Considering the $\{x, y, z\}$ of a joint together as a unit gives us the joint's location in space, and grouping the $\{x, y, z\}$ of a joint for a continuous sequence gives us the trajectory of the joint. Moreover, note that the trajectory-based representation $\mathbf{M}_{\mathrm{t}}$ with $L = n_2$ is distinct from $\mathbf{M}_{\mathrm{f}}^T$.

3.4 Results and Discussion

We first investigate the optimal trajectory length for our proposed method in Section 3.4.1. Then we evaluate our proposed method using this length in Section 3.4.3.

We recover the mocap matrix using the Singular Value Thresholding algorithm (Algorithm 1) for both frame-based and trajectory-based representations, which we call frame-based SVT (FSVT) and trajectory-based SVT (TSVT), respectively. We set the parameters close to the values by recommended by Cai et al. [26]. The τ parameter is set to $8\sqrt{n_1 n_2}$ for FSVT, and $8\sqrt{m_1 m_1}$ for TSVT where (m_1, m_2) are the dimensions of the trajectory-based matrix. For both methods, we set $tol = 0.01$ and $\delta = 1.9$. Implementations of both FSVT and TSSVT are written in matlab and the experiments are run on a computer with an Intel i7-3770 CPU and 8GB memory.

We list again the sequences used for optimal trajectory length estimation (Table 3.1) and for performance evaluations (Table 3.2).

As described in the previous chapter, we simulate random missing data by randomly removing a number of joints so that the required missing data rate is simulated. We

Sequence	Description	Sequence	Description
20_01	Chicken dance	42_01	Stretch, rotate, head, arms, legs
54_01	Monkey actions	55_01	Dance, whirl
56_01	Walk around	85_01	JumpTwist
135_10	Syutouuke martial arts walk	85_06	Kickflip
143_41	Sneak	143_24	Kicking

Table 3.1: Sequences used in estimating trajectory length

Sequence	Description	Sequence	Description
14_01	Boxing	85_02	Jumptwist
143_04	Run figure 8	49_02	Jumping and one foot hopping
135_02	Empi martial arts	135_11	Yokogeri martial arts walk
61_05	Salsa dance	88_04	Acrobatics

Table 3.2: Sequences used for testing algorithm performance

simulate block missing data by partitioning a sequence into time intervals of the same length, and for each interval we randomly remove a fixed number of joints. The number of removed joints determines the error rate of the corrupted mocap sequence, and the interval length determines for how many frames is a joint missing. We use Root Mean Square Error (RMSE) to quantify the distortion of a recovered matrix.

$$\text{RMSE}(\hat{\mathbf{X}}, \mathbf{M}) = \sqrt{\frac{1}{n_1 \times n_2} \sum_{j=1}^{n_2} \sum_{i=1}^{n_1} \left(\hat{\mathbf{X}}_{ij} - \mathbf{M}_{ij}\right)^2} \tag{3.7}$$

where $\hat{\mathbf{X}}$ and $\mathbf{M}$ are the matrices of the recovered and original mocap, respectively.

3.4.1 Trajectory Length Evaluation

Since our aim is to improve the performance of matrix completion by using a matrix representation with lower rank, we first verify that a trajectory-based representation has lower rank. In practice, a mocap matrix is nearly always full rank, so we compute the

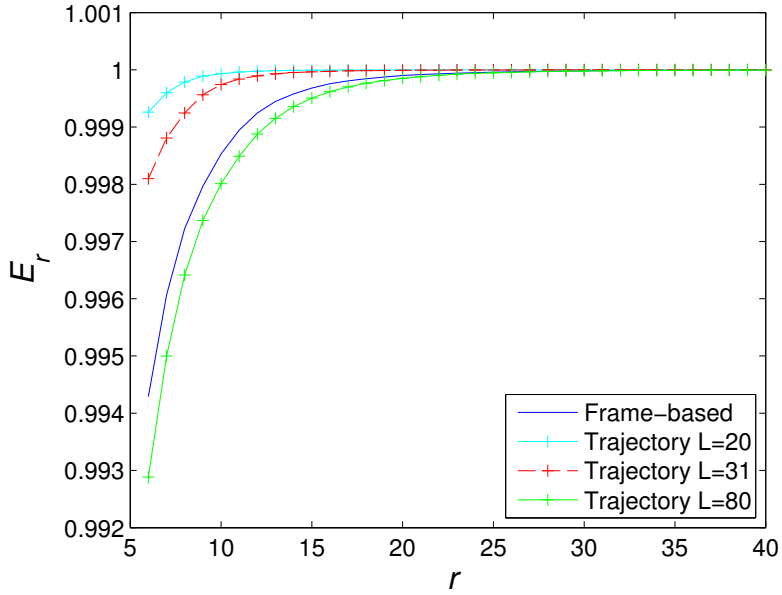


Figure 3.2: Comparison of average E_r of mocap matrices of rank r for frame-based matrices and trajectory-based matrices at different trajectory lengths L.

energy E_r of the closest approximating matrix of a lower rank r, for different values of r. E_r is obtained by computing the Singular Value Decomposition of a matrix, arranging the singular values in decreasing order, and then evaluating Eq. 3.8.

$$E_r = \frac{\sum_{i=1}^{r} \sigma_i^2}{\sum_{i=1}^{j_{\max}} \sigma_i^2} \tag{3.8}$$

E_r represents the amount of information retained by the closest approximating matrix of rank r; we can say that a matrix has rank r where its E_r is very close to 1. In Fig. 3.2, we say that a matrix has a lower rank if its energy converges to 1 earlier (at lower values of r) compared to other matrices.

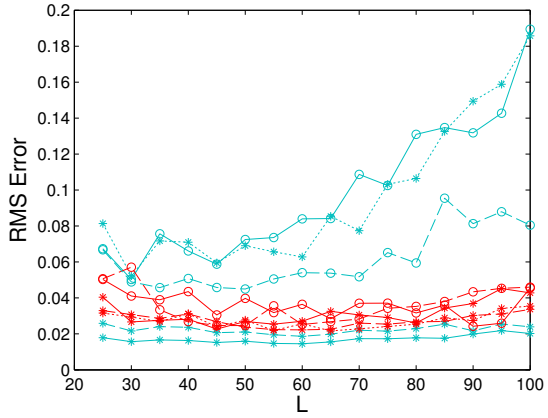
We pick a set of mocap sequences and compare the average energy of both frame-based and trajectory-based matrices at different ranks r and trajectory lengths L. From the figure, the rank of the trajectory-based matrix is dependent on the trajectory length

L used; a representation with lower L has a lower rank. However, since a lower L also decreases the maximum rank of a representation, this conclusion is not meaningful; as explained in Section 3.2, the rank of a matrix should be considered in the context of its maximum rank.

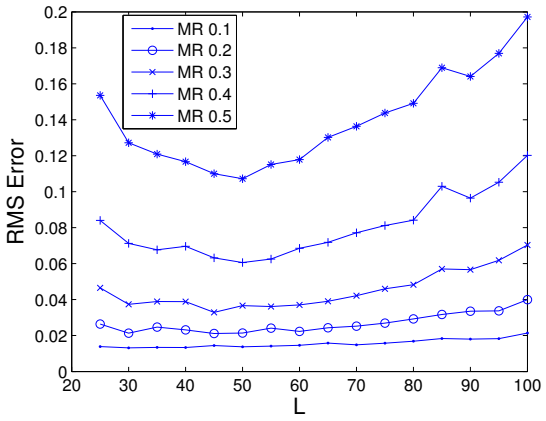
Nevertheless, we can compare frame-based and trajectory-based matrices of the same dimensions and maximum rank by setting $L = 31$. As shown in Fig. 3.2, the trajectory-based matrix with $L = 31$ converges earlier than the frame-based matrix; thus the trajectory-based representation has a lower rank than the frame-based representation.

To estimate the optimal value of L, we use a set of mocap sequences (listed in Table 3.2) with simulated missing entries. For each sequence, we simulate missing entries at a certain missing rate using random missing data and then check the RMS error of the sequence recovered using different values of L (Fig. 3.3(a)). For each missing rate, we obtain the average of RMS errors over all sequences (Fig. 3.3(b)). Finally, we take the points with minimum L for each missing rate and fit a line to them using least squares (Fig. 3.3(c)). From the figure, we see that the optimal L increases with increasing missing rate. Evidently, shorter trajectory lengths are less robust towards increasing error rate; RMS error rises more quickly when missing data rate increases (Fig. 3.3(b)).

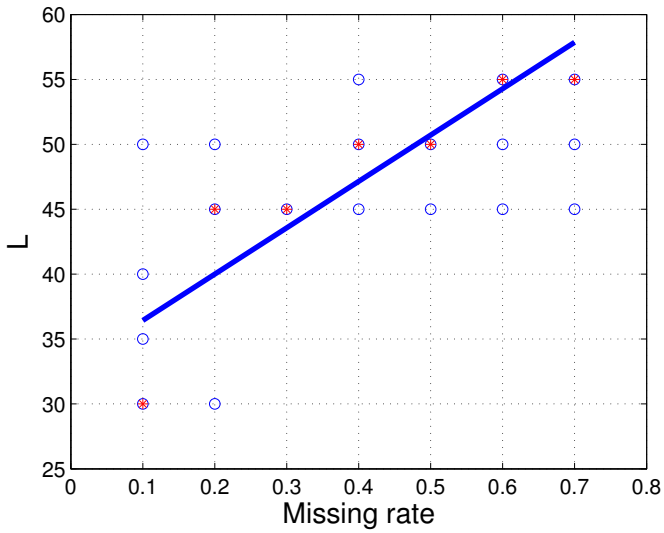
We also see that the value of L has less impact at the lowest missing rate of 10%; the RMS error for a wide range of L is within 5% of the minimum RMS error. Hence we set the value of L to 50, which we think is sufficiently accurate, instead of changing it linearly with the missing rate. A constant value is preferable to a linear function since it is a simpler way to explain the data.



(a) RMS errors of various mocap sequences with 30% missing rate



(b) Mean RMS errors



(c) Line fitted to L with minimum RMS error (shown in asterisks). Circles show L with RMS error within 5% of the minimum RMS error

Figure 3.3: Estimation of optimal value for trajectory length L.

Sequence	N	FSVT			TSVT			mean increase
		0.1	0.3	0.5	0.1	0.3	0.5	
85_02	405	7	7	6	9	11	11	32%
143_04	543	9	9	8	13	15	13	38%
49_02	1043	15	17	17	29	32	32	48%
135_11	1223	18	20	18	27	36	34	42%
88_04	1240	16	15	12	34	36	34	59%
61_05	1678	24	23	20	42	49	51	52%
135_02	2601	34	30	27	71	72	66	56%
14_01	2797	48	46	42	76	82	77	42%

Table 3.3: Processing time (seconds) of FSVT and TSVT at missing data rates of 0.1, 0.3, and 0.5 of random missing data. N denotes the number of frames in a sequence

Sequence	N	FSVT			TSVT			mean increase
		0.1	0.3	0.5	0.1	0.3	0.5	
85_02	405	1153	1274	1032	1212	1512	1543	18%
143_04	543	1153	1260	1105	1290	1555	1319	15%
49_02	1043	1101	1300	1320	1892	2114	2141	40%
135_11	1223	1152	1297	1243	1524	1834	1939	30%
88_04	1240	990	1004	811	1898	2004	1911	52%
61_05	1678	1093	1091	985	1700	2050	2181	46%
135_02	2601	993	916	806	1876	1937	1838	52%
14_01	2797	1228	1248	1165	1919	2105	2037	40%

Table 3.4: Number of iterations to convergence of FSVT and TSVT at missing data rates of 0.1, 0.3, and 0.5 of random missing data. N denotes the number of frames in a sequence

3.4.2 Processing Time Evaluation

From Table 3.3, we can see that the processing times of FSVT and TSVT are relatively constant even when missing data rates are increased for random missing data. Compared with FSVT, TSVT has a higher processing time by 46% on average. The number of iterations to convergence for TSVT is increased over FSVT by 36% on average, as seen in Table 3.4. Hence, although the increase in processing time is mainly due to the increase in the number of iterations to convergence, there is also an increase in the processing time per iteration. This is due to the effect of the different dimensions of frame-based and trajectory-based matrices on the Singular Value Decomposition operation (Line 16 of Algorithm 1), which is the most time consuming operation in SVT by far.

3.4.3 Performance Evaluation

Here, we compare matrix completion of frame-based mocap matrix with the proposed trajectory-based representation method using mocap sequences listed in Table 3.2. The RMS error of mocap data recovered from both methods are compared for varying fractions of missing data. We first show the results for random missing data. From Table 3.5, the sequences with the highest error are 85_02, 135_02, and 88_04. This conforms with expectations, since these sequences contain a variety of motions with little repetition, whereas all other motions have a higher degree of repetition. The idea is that if there is more than one exactly the same frames in a mocap sequence, with both having different missing elements, it is easier to recover both frames. Generally there is a slight drop in accuracy at missing rates above 40%. Comparing methods, the trajectory-based method is much better than the frame-based method, with up to 80% improvement and 68% improvement on average. This shows that short joint trajectories are more highly correlated than frames.

Sequence	Method	Missing data					
		10%	20%	30%	40%	50%	60%
14_01	A	0.052	0.092	0.171	0.264	0.404	0.614
(Boxing)	B	0.014	0.023	0.038	0.066	0.121	0.211
	C	74.19%	75.44%	77.52%	75.08%	70.15%	65.64%
85_02	A	0.041	0.094	0.192	0.402	0.661	1.193
(Jumptwist)	B	0.020	0.035	0.070	0.179	0.333	0.678
	C	51.69%	62.80%	63.42%	55.50%	49.63%	43.13%
143_04	A	0.047	0.096	0.175	0.284	0.459	0.857
(Running)	B	0.020	0.040	0.075	0.154	0.314	0.781
	C	57.04%	58.41%	57.26%	45.87%	31.70%	8.89%
49_02	A	0.044	0.074	0.118	0.178	0.265	0.438
(Jumping)	B	0.014	0.022	0.034	0.053	0.099	0.201
	C	66.90%	70.03%	71.05%	70.36%	62.85%	54.16%
135_02	A	0.087	0.151	0.262	0.415	0.645	1.025
(MartialArts)	B	0.022	0.034	0.053	0.081	0.139	0.265
	C	74.97%	77.54%	79.91%	80.41%	78.46%	74.13%
135_11	A	0.050	0.078	0.141	0.240	0.380	0.683
(Kicking)	B	0.017	0.029	0.049	0.084	0.130	0.254
	C	65.22%	63.03%	65.43%	65.08%	65.78%	62.87%
61_05	A	0.056	0.103	0.175	0.275	0.417	0.657
(Salsa)	B	0.014	0.019	0.027	0.039	0.070	0.135
	C	75.79%	81.82%	84.77%	85.67%	83.24%	79.38%
88_04	A	0.088	0.149	0.297	0.494	0.774	1.201
(Acrobatics)	B	0.018	0.031	0.052	0.088	0.154	0.308
	C	79.67%	79.19%	82.36%	82.28%	80.15%	74.38%

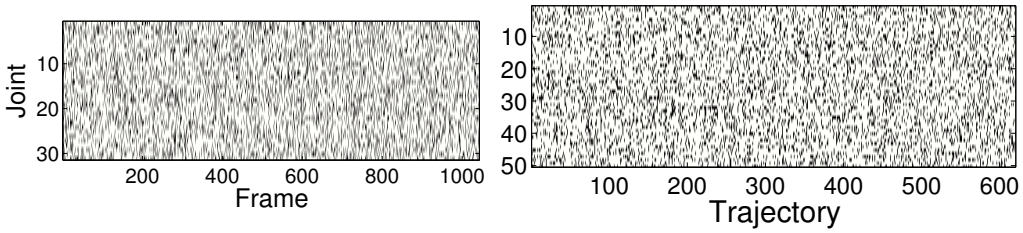
Table 3.5: Performance comparison of matrix completion of (A) frame-based representation, and (B) trajectory-based representation, and (C) the percentage of improvement under random missing data.

Sequence	Method	Missing data		
		l=5	l=10	l=20
14_01	A	0.044	0.044	0.056
	B	0.151	0.389	0.716
	C	0.061	0.152	0.331
49_02	A	0.048	0.049	0.053
	B	0.162	0.425	0.929
	C	0.067	0.089	0.165
88_04	A	0.079	0.103	0.175
	B	0.258	0.631	1.253
	C	0.069	0.166	0.438

Table 3.6: Performance comparison of matrix completion of (A) frame-based representation, (B) trajectory-based representation with $L = 50$, and (C) trajectory-based representation with $L = 200$ under block missing data at a missing rate of 10%. l is the length of each missing data interval.

Next, we show the results for block missing data in Table 3.6. First, for each sequence, even though the missing data rate is fixed at 10%, both frame-based and trajectory-based methods show increased error when the missing data interval length l increases. Second, we can see that the trajectory-based representation is not able to recover missing data as effectively as the frame-based representation under this type of error. The performance of TSVT improves when the value of L is increased to 200; nevertheless, its performance is still worse than that of FSVT. Furthermore, increasing L defeats the strength of the trajectory-based representation in utilizing the low rank property of short trajectories.

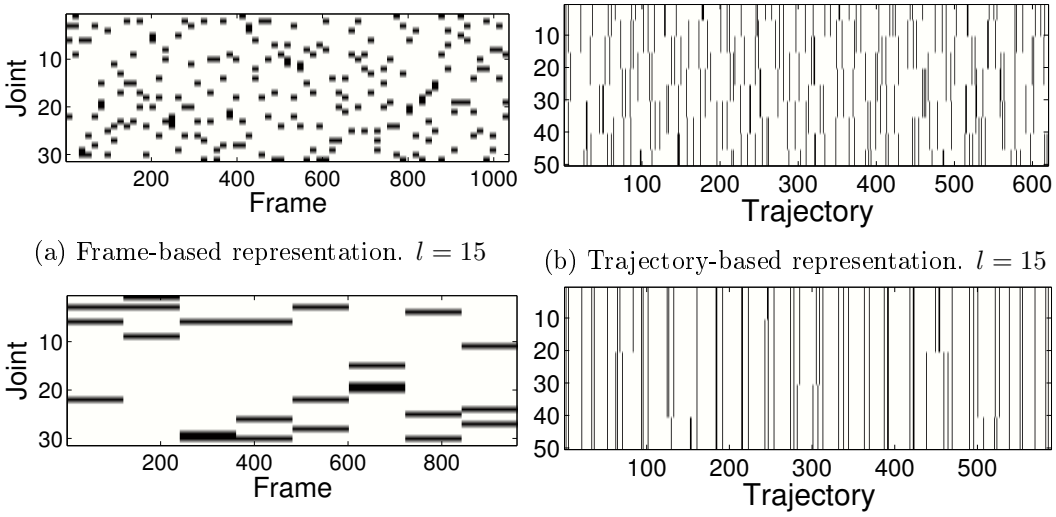
It could be quite surprising that TSVT performs worse than FSVT, given that it has been shown earlier that TSVT has lower rank than FSVT. We would like to point out that for the case of block missing data, the missing data distribution is no longer uniform, which is a requirement for Eq. 3.2 to be valid. We compare the missing data mask for a frame-based representation and its corresponding trajectory-based representation in



(a) Frame-based representation

(b) Trajectory-based representation

Figure 3.4: Random missing data at 20% error rate



(a) Frame-based representation. $l = 15$

(b) Trajectory-based representation. $l = 15$

(c) Frame-based representation. $l = 120$

(d) Trajectory-based representation. $l = 120$

Figure 3.5: Block missing data with 3 missing joints

Fig. 3.4 for random missing data. The missing entry distribution for the trajectory-based representation is as uniform as that for the frame-based distribution. For block missing data, as shown in Fig. 3.5, the trajectory-based representation is much less uniform than the frame-based representation. At interval length $l = 15$, some trajectories (columns) have zero missing entries while some have more than half of their entries missing. This situation becomes worse at higher values of l, such as $l = 120$. Many trajectories are have all their entries missing; these trajectories are impossible to recover.

3.5 Conclusion

Low rank matrix completion can reliably recover matrices even though large fractions of its entries are missing. However, in order to better utilize this method, the matrix should be structured to have a lower rank. In this chapter, we propose to improve the performance of low rank matrix completion with Singular Value Thresholding for mocap recovery by representing mocap data as sets of short trajectory segments instead of the previously used sets of frames. We show that at the same matrix dimensions, the trajectory-based representation has a lower rank than the frame-based representation. Using the estimated optimal trajectory length, the proposed trajectory-based representation is universally better than the frame-based representation by a huge margin for randomly missing joints. In our experiments, we achieve significantly better recovery results in all tested mocap sequences, at 68% on average and up to 80% in some cases, under various fractions of randomly missing data. However, for the case block missing data, the trajectory-based representation performs worse, due to the uneven nature of the missing data when it is presented in trajectory-based form.

For block missing data, both the frame-based representation and trajectory-based representation decrease in performance significantly when the missing data interval length

is increased. Since using a trajectory-based representation is not a good way to deal with this type of error, we need an alternative method to improve the SVT framework towards block missing data. We shall present the method in the next chapter.

Chapter 4

Mocap Recovery Using Skeleton Constrained Singular Value Thresholding

4.1 Introduction

In the previous chapter we showed that the trajectory-based matrix completion outperforms the frame-based matrix completion for mocap recovery by exploiting the correlation between trajectories. However, for cases where joints are missing for long intervals, both frame-based and trajectory-based matrix completion is not robust when the interval length is increased. Thus, in this chapter, we aim to improve the performance of matrix completion for mocap recovery by using an alternative method that is robust towards such missing data.

As a recap, Candès and Recht [25] showed that a low rank matrix can be accurately recovered from observations of a small fraction of its entries by solving for the matrix

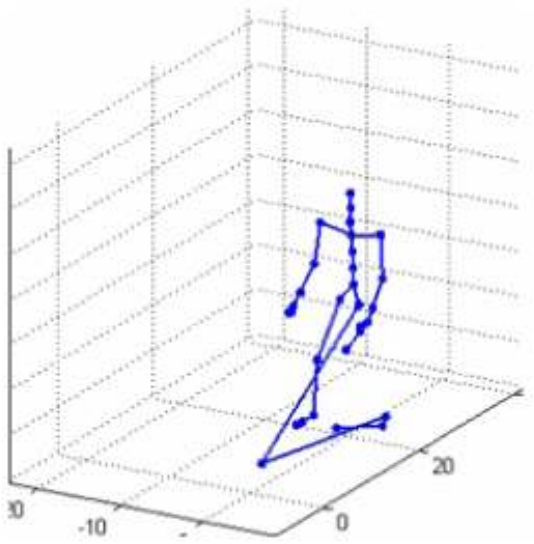


Figure 4.1: A frame of a mocap sequence recovered using SVT. The position of the left knee joint and the length of the bones connected to the joint are severely distorted.

with the lowest nuclear norm that conforms to the observations; an approach known as matrix completion. Lai et al. [24] showed that mocap data, when arranged in the form of a matrix, is a low rank matrix. They then exploit this property to recover mocap data by using a matrix completion method known as Singular Value Thresholding (SVT) [26].

Since the human skeleton is rigid, the distance between any two adjacent joints is constant throughout a motion sequence. However, this inter-joint distance is not preserved for mocap data recovered using SVT, especially when mocap entries are missing for long intervals. We observe that the bone length of the skeleton of a recovered mocap data might undergo extension and shrinkage throughout a sequence; this observation can be observed in Fig. 4.1, and is verified quantitatively in Fig. 4.2.

We remove consecutive entries of mocap data for certain interval lengths, recover the entries using SVT, and plot the RMS error for the recovered data and the RMS error of inter-joint distances of all pairs of adjacent joints from the recovered data. As shown

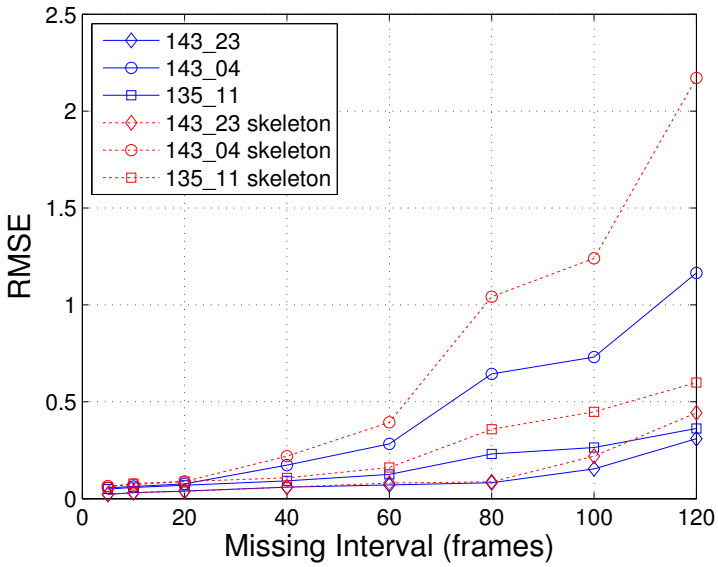


Figure 4.2: RMS error and skeleton RMS error of three mocap sequences recovered using Singular Value Thresholding over varying missing data intervals. The skeleton RMS error is the RMS error of inter-joint distances of all pairs of adjacent joints in the recovered mocap sequence.

in the figure, both errors increase when the interval length is increased; increases in fluctuation of the inter-joint distance is correlated with increases in error of the recovered mocap data.

In this chapter we propose a skeleton constrained SVT method, which attempts to preserve bone inter-joint distances when recovering mocap data. We formulate the bone inter-joint distances as additional constraints, taking care to ensure that the problem is still convex, and show how to solve the problem within the frame-based SVT framework. Experiments show that the proposed skeleton constrained frame-based SVT (SCFSVT) outperforms the frame-based SVT (FSVT) method by a significant margin, especially for cases where mocap data is missing for long periods. SCFSVT performs competitively with BOLERO [23], a mocap recovery method modeled on linear dynamical systems that also utilizes skeleton constraints, but at a much lower computational cost.

In Section 4.2, we will discuss the problem formulation and solution of the basic SVT mocap recovery method. After than, we extend the problem to accommodate skeleton constraints and derive the solution in Section 4.3. In Section 4.4, we will compare and discuss the performance of the evaluated mocap recovery methods and wrap up with a conclusion in Section 4.5.

4.2 Mocap Recovery Using SVT

To recover a mocap sequence using matrix completion, we can first represent it using a frame-based matrix representation $\mathbf{M} \in \mathbb{R}^{n_1 \times n_2}$, $n_1 = 3j$

$$\mathbf{M} = [\mathbf{f}_1 \ \mathbf{f}_2 \ \ldots \ \mathbf{f}_{n_2}] \tag{4.1}$$
$$\text{where } \mathbf{f} = [x_1 \ y_1 \ z_1 \ x_2 \ y_2 \ z_2 \ \ldots \ x_j \ y_j \ z_j]^T$$

where each frame $\mathbf{f}$ holds the 3D coordinates $\{x, y, z\}$ of j joints. The mocap matrix has low rank; thus it can be effectively recovered using matrix completion. The mocap recovery problem is posed as the following optimization problem[1]

$$\min \quad f(\mathbf{X}) \tag{4.2}$$
$$\text{s.t.} \quad \mathcal{S}(\mathbf{X}) = \mathbf{b} \tag{4.3}$$

where $\mathcal{S} : \mathbb{R}^{n_1 \times n_2} \mapsto \mathbb{R}^{n_b}$ is a linear map that extracts n_b entries from a $n_1 \times n_2$ matrix. $\mathbf{b}$ contains the m observed entries of the matrix $\mathbf{M}$ to be recovered, i.e., $\mathbf{b} = \mathcal{S}(\mathbf{M})$. $f(\mathbf{X}) \coloneqq \|\mathbf{X}\|_*$, where $\|\mathbf{X}\|_*$ is the nuclear norm of the decision variable $\mathbf{X} \in \mathbb{R}^{n_1 \times n_2}$. The problem can be solved using the Singular Value Thresholding (SVT) algorithm [26],

[1] This is the same optimization problem as Eq. 3.4, but is casted in a different way.

which computes an approximate solution by solving for $f(\mathbf{X}) \coloneqq \tau\|\mathbf{X}\|_* + \frac{1}{2}\|\mathbf{X}\|_F^2$, where $\|\mathbf{X}\|_F$ is the Frobenius norm of $\mathbf{X}$. However, it is efficient in both time and memory. A better approximation is obtained by setting a larger value for parameter τ.

The solution for Problem 4.2 using the SVT method is

$$\mathbf{X}^k = \mathcal{D}_\tau \mathcal{S}^*(\mathbf{y}^{k-1}) \tag{4.4}$$

$$\mathbf{y}^k = \mathbf{y}^{k-1} + \delta(\mathbf{b} - \mathcal{S}(\mathbf{X}^k)) \tag{4.5}$$

The recovered mocap matrix $\hat{\mathbf{X}} = \mathbf{X}^{k_{\text{final}}}$ is obtained by iteratively solving for $\mathbf{X}^k$ in Eq. 4.4 and $\mathbf{y}^k$ in Eq. 4.5 until convergence. $\mathcal{S}^* : \mathbb{R}^{n_b} \mapsto \mathbb{R}^{n_1 \times n_2}$ is the adjoint of $\mathcal{S}$, while δ is the step size parameter. $\mathcal{D}_\tau$, a characteristic operator of SVT, soft-thresholds the singular values σ_i of input $\mathbf{X}$ with parameter τ. It is defined as

$$\mathcal{D}_\tau(\mathbf{X}) \coloneqq \mathbf{U}\mathcal{D}_\tau(\mathbf{\Sigma})\mathbf{V}^T, \quad \mathcal{D}_\tau(\mathbf{\Sigma}) = \text{diag}((\sigma_i - \tau)_+) \tag{4.6}$$

where $(\sigma_i - \tau)_+ = \max(0, \sigma_i - \tau)$, and orthogonal matrices $\mathbf{U}$ and $\mathbf{V}$, and matrix of singular values Σ are obtained using Singular Vector Decomposition (SVD), i.e., $\mathbf{X} = \mathbf{U}\Sigma\mathbf{V}^T$. In other words, $\mathcal{D}_\tau(\mathbf{X})$ factorizes $\mathbf{X}$ using SVD and sets all singular values in Σ that are less than τ to zero.

Note that the equations used to solve for SVT in Chapter 3, as presented in Algorithm 1, is cast in a different, but equivalent, way. Instead of Eq. 4.4 and Eq. 4.5, the equations are

$$\mathbf{X}^k = \mathcal{D}_\tau(\mathbf{Y}^{k-1}) \tag{4.7}$$

$$\mathbf{Y}^k = \mathbf{Y}^{k-1} + \delta P_\Omega(\mathbf{M} - \mathbf{X}^k) \tag{4.8}$$

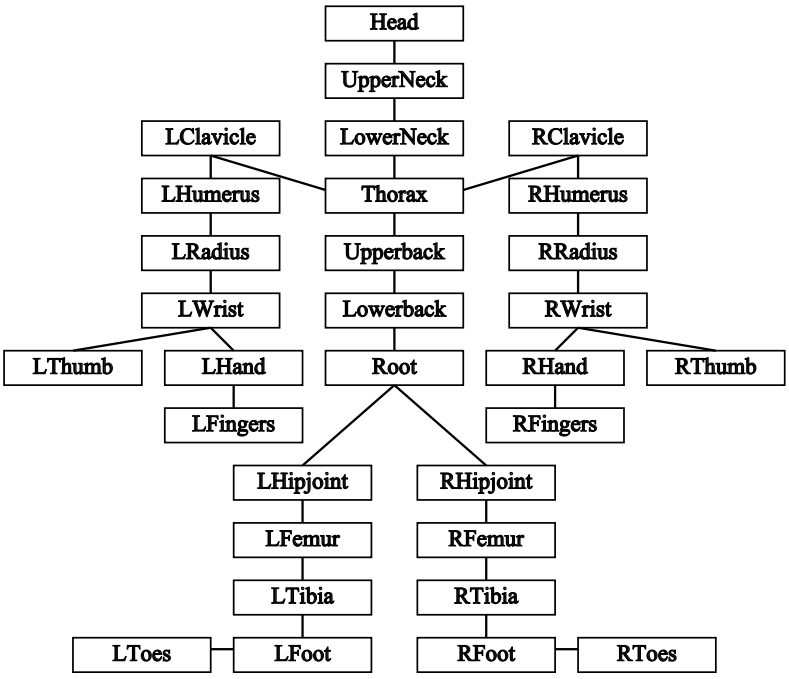


Figure 4.3: Mocap skeleton and its joints. Each label indicates a joint, and 'L' and 'R' prefixes denote left-sided and right-sided, respectively.

where P_Ω sets the matrix elements corresponding to missing entries Ω to zero. The recovered mocap matrix $\hat{\mathbf{X}} = \mathbf{X}^{k_{\text{final}}}$ is obtained by iteratively solving for $\mathbf{X}^k$ and $\mathbf{Y}^k$ until convergence (see Algorithm 1).

4.3 Skeleton Constrained SVT

4.3.1 Problem Formulation

As mentioned earlier, the previous method of mocap recovery using SVT does not enforce preservation of skeleton constraints. We overcome this weakness by applying skeleton constraints to Problem 4.2. First, we rewrite $\mathcal{S}(\mathbf{X})$ in Eq. 4.3 as a composition of sampling

matrix $\mathbf{A}$ and a linear map vec

$$\mathcal{S}(\mathbf{X}) = \mathbf{A}\,\mathrm{vec}(\mathbf{X}) = \mathbf{A}\mathbf{x} \tag{4.9}$$

$$\text{where} \quad \mathrm{vec} : [\mathbf{f}_1 \ \dots \ \mathbf{f}_{n_2}] \mapsto \begin{bmatrix} \mathbf{f}_1 \\ \vdots \\ \mathbf{f}_{n_2} \end{bmatrix} \tag{4.10}$$

vec maps $\mathbf{X} \in \mathbb{R}^{n_1 \times n_2}$ to $\mathbf{x} \in \mathbb{R}^{n_1 n_2}$, and $\mathbf{A} \in \mathbb{R}^{n_b \times n_1 n_2}$ extracts the n_b observed entries $\mathbf{b} \in \mathbb{R}^{n_b}$ from $\mathbf{M}$, i.e., $\mathbf{b} = \mathbf{A}\,\mathrm{vec}(\mathbf{M})$.

The mocap skeleton used in this chapter is shown in Fig. 4.3, where each label is a joint, and each edge is a bone connecting adjacent joints. For each joint that is missing, we insert each of its adjacent edges into a set $\mathcal{E}$. For each edge $e_i \in \mathcal{E},\ i = 1 \dots p$, we have its length d_i and connected joint pair $(\alpha, \beta)_i$. Then, for each e_i, we define skeleton constraint matrix $\mathbf{C}_i \in \mathbb{R}^{3 \times n_1 n_2}$, which extracts the inter-joint distance of $(\alpha, \beta)_i$ from $\mathbf{x}$, i.e., $\mathbf{C}_i\mathbf{x} = [x\ y\ z]^T_{\alpha_i} - [x\ y\ z]^T_{\beta_i}$. Applying constraints $\mathbf{C}_i$ to Problem 4.2, we get Problem 4.11 as follows

$$\min \quad f(\mathbf{X}) \tag{4.11}$$

$$\text{s.t.} \quad \mathbf{x} = \mathrm{vec}(\mathbf{X}) \tag{4.12}$$

$$\mathbf{A}\mathbf{x} = \mathbf{b} \tag{4.13}$$

$$\|\mathbf{C}_i\mathbf{x}\|_2 \leq d_i \qquad i = 1...p \tag{4.14}$$

where p is the number of pairs of adjacent joints. d_i can be obtained in a preprocessing step by simple averaging of inter-joint distances of non-missing joints

4.3.2 Solution of Problem

In a similar way to [26], we derive the solution to Problem 4.11 by stating the solution given by the gradient algorithm applied to the Lagrangian, and then recasting it into the solution obtained using SVT. The gradient algorithm applied to the Lagrangian for a problem with Lagrangian $\mathcal{L}(\mathbf{X}, \mathbf{y})$ can be expressed as

$$\mathbf{X}^k = \arg\min_{\mathbf{X}} \mathcal{L}(\mathbf{X}, \mathbf{y}^{k-1}) \tag{4.15}$$

$$\mathbf{y}^k = \mathbf{y}^{k-1} + \delta\, \partial_{\mathbf{y}} \mathcal{L}(\mathbf{X}^k, \mathbf{y}) \tag{4.16}$$

where $\mathbf{y}$ is the Lagrange multiplier of $\mathcal{L}(\mathbf{X}, \mathbf{y})$, and δ is the step size parameter. The solution proceeds by iteratively solving for $\mathbf{X}$ that minimizes $\mathcal{L}(\mathbf{X}, \mathbf{y})$ and moving $\mathbf{y}$ in the direction of the gradient. Upon convergence, the solution $\hat{\mathbf{X}} = \mathbf{X}^{k_{\text{final}}}$ is obtained.

To obtain the Lagrangian for Problem 4.11, we first express the quadratic skeleton constraints in Eq. 4.14 in a convex form, i.e., in the form of a second order cone $\mathcal{K}$ (see [26]).

$$\begin{bmatrix} \mathbf{C}_i \mathbf{x} \\ d_i \end{bmatrix} \in \mathcal{K}_i \tag{4.17}$$

Following that, the Lagrangian is

$$\begin{aligned}
\mathcal{L}(\mathbf{X}, \mathbf{y}_\mathrm{a}, \mathbf{y}_\mathrm{c}, \mathbf{s}) =& f(\mathbf{X}) + \langle \mathbf{y}_\mathrm{a}, \mathbf{b} - \mathbf{A}\,\mathrm{vec}(\mathbf{X}) \rangle \\
&+ \sum_i^p \left(\langle \mathbf{y}_{\mathrm{c}_i}, -\mathbf{C}_i\,\mathrm{vec}(\mathbf{X}) \rangle - s_i d_i \right) && (4.18) \\
=& f(\mathbf{X}) + \langle \mathbf{y}_\mathrm{a}, \mathbf{b} \rangle - \langle \mathbf{y}_\mathrm{a}, \mathbf{A}\,\mathrm{vec}(\mathbf{X}) \rangle \\
&- \sum_i^p \left(\langle \mathbf{y}_{\mathrm{c}_i}, \mathbf{C}_i\,\mathrm{vec}(\mathbf{X}) \rangle + s_i d_i \right) && (4.19) \\
=& f(\mathbf{X}) + \langle \mathbf{y}_\mathrm{a}, \mathbf{b} \rangle - \left\langle \mathrm{vec}^*(\mathbf{A}^T \mathbf{y}_\mathrm{a}), \mathbf{X} \right\rangle \\
&- \sum_i^p \left(\left\langle \mathrm{vec}^*(\mathbf{C}_i^T \mathbf{y}_{\mathrm{c}_i}), \mathbf{X} \right\rangle + s_i d_i \right) && (4.20)
\end{aligned}$$

where $< \mathbf{X}, \mathbf{Y} >$ is the standard inner product between matrices $\mathbf{X}$ and $\mathbf{Y}$. $\mathbf{y}_\mathrm{a}$ and $\mathbf{y}_{\mathrm{c}_i}$ are the Lagrange multipliers associated with Eq. 4.13 and the ith constraint of Eq. 4.14, respectively. Taking $f(\mathbf{X}) \coloneqq \tau \|\mathbf{X}\|_* + \frac{1}{2}\|\mathbf{X}\|_F^2$, we have

$$\begin{aligned}
&\arg\min_X \mathcal{L}(\mathbf{X}, \mathbf{y}_\mathrm{a}, \mathbf{y}_\mathrm{c}, \mathbf{s}) \\
&= \Bigg\{ \mathbf{X} \;\Bigg|\; \tau\, \partial_X \|\mathbf{X}\|_* + \mathbf{X} && (4.21) \\
&\quad - \mathrm{vec}^*(\mathbf{A}^T \mathbf{y}_\mathrm{a}) - \sum_i^p \mathrm{vec}^*(\mathbf{C}_i^T \mathbf{y}_{\mathrm{c}_i}) = 0 \Bigg\} && (4.22) \\
&= \left\{ \mathbf{X} \;\middle|\; \tau\, \partial_X \|\mathbf{X}\|_* + \mathbf{X} - \mathrm{vec}^* \left(\begin{bmatrix} \mathbf{A}^T & \mathbf{C}^T \end{bmatrix} \begin{bmatrix} \mathbf{y}_\mathrm{a} \\ \mathbf{y}_\mathrm{c} \end{bmatrix} \right) = 0 \right\} && (4.23) \\
&\text{where} \quad \mathbf{C} = \begin{bmatrix} \mathbf{C}_1 \\ \vdots \\ \mathbf{C}_p \end{bmatrix},\ \mathbf{y}_\mathrm{c} = \begin{bmatrix} \mathbf{y}_{\mathrm{c}_1} \\ \vdots \\ \mathbf{y}_{\mathrm{c}_p} \end{bmatrix} && (4.24)
\end{aligned}$$

It has been shown in [26] that

$$\mathcal{D}_\tau(\mathbf{Y}) = \arg\min_X \ \tau\|\mathbf{X}\|_* + \frac{1}{2}\|\mathbf{X}-\mathbf{Y}\|_F^2 \tag{4.25}$$

$$= \{\mathbf{X} \mid \tau\, \partial_X \|\mathbf{X}\|_* + \mathbf{X} - \mathbf{Y} = \mathbf{0}\} \tag{4.26}$$

By comparing Eq. 4.23 with Eq. 4.26, we can solve Eq. 4.23 using the singular value soft thresholding operator $\mathcal{D}_\tau$

$$\arg\min_X \mathcal{L}(\mathbf{X}, \mathbf{y}_\mathrm{a}, \mathbf{y}_\mathrm{c}, \mathbf{s}) = \mathcal{D}_\tau \operatorname{vec}^* \left(\begin{bmatrix} \mathbf{A}^T & \mathbf{C}^T \end{bmatrix} \begin{bmatrix} \mathbf{y}_\mathrm{a} \\ \mathbf{y}_\mathrm{c} \end{bmatrix} \right) \tag{4.27}$$

Thus, to update $\mathbf{X}^k$, we use the Lagrangian in Eq. 4.27 in place of the Lagrangian in Eq. 4.15

$$\mathbf{X}^k = \mathcal{D}_\tau \operatorname{vec}^* \left(\begin{bmatrix} \mathbf{A}^T & \mathbf{C}^T \end{bmatrix} \begin{bmatrix} \mathbf{y}_\mathrm{a}^{k-1} \\ \mathbf{y}_\mathrm{c}^{k-1} \end{bmatrix} \right) \tag{4.28}$$

To update $\mathbf{y}_a$, we use the Lagrangian in Eq. 4.18 in place of the Lagrangian in Eq. 4.16.

$$\mathbf{y}_\mathrm{a}^k = \mathbf{y}_\mathrm{a}^{k-1} + \delta\, \partial_{\mathbf{y}_\mathrm{a}} \mathcal{L}(\mathbf{X}^k, \mathbf{y}_\mathrm{a}, \mathbf{y}_\mathrm{c}, \mathbf{s}) \tag{4.29}$$

$$= \mathbf{y}_\mathrm{a}^{k-1} + \delta(\mathbf{b} - \mathbf{A}\operatorname{vec}(\mathbf{X}^k)) \tag{4.30}$$

Each $\mathbf{y}_{c_i}$ is updated by orthogonal projection $\mathcal{P}_i$ onto their respective cones $\mathcal{K}_i$ (see [26])

$$\begin{bmatrix} \mathbf{y}_{c_i}^k \\ s_i^k \end{bmatrix} = \mathcal{P}_i \left(\begin{bmatrix} \mathbf{y}_{c_i}^{k-1} \\ s_i^{k-1} \end{bmatrix} + \delta \begin{bmatrix} -\mathbf{C}_i \operatorname{vec}(\mathbf{X}^k) \\ -d_i \end{bmatrix} \right), i = 1...p \tag{4.31}$$

$$\text{where} \quad \mathcal{P}_i : (\mathbf{y}_c, s)_i \mapsto \begin{cases} (\mathbf{y}_c, s), & \|\mathbf{y}_c\| \leq s \\ \frac{\|\mathbf{y}_c\|+s}{2\|\mathbf{y}_c\|}(\mathbf{y}_c, s), & -\|\mathbf{y}_c\| \leq s \leq \|\mathbf{y}_c\| \\ (\mathbf{0}, 0), & s \leq -\|\mathbf{y}_c\| \end{cases} \tag{4.32}$$

The proposed method, an algorithm to solve Problem 4.11 and recover the mocap sequence, is summarized in Algorithm 2. The inputs are the n_b observed entries of a mocap matrix, $\mathbf{b} \in \mathbb{R}^{n_b}$ and the inter-joint distance of p joint pairs, $\mathbf{d} \in \mathbb{R}^p$. Matrix $\mathbf{A}$ and map vec are described in Eq. 4.9, and matrix $\mathbf{C}$ is described when discussing Eq. 4.14.

A larger τ improves the accuracy of the solution at the cost of increased number of iterations to convergence. A higher δ reduces the number of iterations to convergence, but if the value is too large, the solution will diverge. We empirically set τ to $8\sqrt{n_1 n_2}$ for a reasonable compromise between performance and speed. The value of δ is discussed in Section 4.4.1. The RMSE operator in the condition statement of the while loop is defined in Eq. 4.33, and the tolerance *tol* value used is 0.01. The FSVT algorithm without skeleton constraints can be obtained by removing $\mathbf{C}$, $\mathbf{y}_c$ and s_i from Algorithm 2.

4.4 Results and Discussion

In this section we evaluate the performance of the proposed method in recovering missing mocap data. Recall that each sequence contains 31 joints and 30 skeleton constraints (see Fig. 4.3). We generate the 3D coordinates of each joint from joint angle representation data, so the length of each bone d_i is known exactly. We use a set of sequences of

Algorithm 2 Skeleton Constrained SVT

Input: $\mathbf{A}, \mathbf{C}, \mathbf{b}, \mathbf{d}$
Parameter: δ, τ, $k_{\max}$, tol
Output: Recovered mocap sequence $\hat{\mathbf{X}}$
Initialization:
$\mathbf{y}_\mathrm{a} \leftarrow \mathbf{b}$
$\mathbf{y}_\mathrm{c} \leftarrow \mathbf{0}$
$\mathbf{s}_c \leftarrow 0$
$k \leftarrow 1$

while $k < k_{max}$ **and** $\mathrm{RMSE}(\mathbf{A}\,\mathrm{vec}(\mathbf{X}^k), \mathbf{b}) > tol$ **do**
  $\mathbf{Y} \leftarrow \mathrm{vec}^*\left(\begin{bmatrix}\mathbf{A}^T & \mathbf{C}^T\end{bmatrix}\begin{bmatrix}\mathbf{y}_\mathrm{a}^{k-1} \\ \mathbf{y}_\mathrm{c}^{k-1}\end{bmatrix}\right)$ // update $\mathbf{X}$
  where $\mathbf{C} = \begin{bmatrix}\mathbf{C}_1 \\ \vdots \\ \mathbf{C}_p\end{bmatrix}, \mathbf{y}_\mathrm{c} = \begin{bmatrix}\mathbf{y}_{\mathrm{c}_1} \\ \vdots \\ \mathbf{y}_{\mathrm{c}_p}\end{bmatrix}$
  $\mathbf{X}^k \leftarrow \mathcal{D}_\tau(\mathbf{Y})$
  $\mathbf{y}_\mathrm{a}^k \leftarrow \mathbf{y}_\mathrm{a}^{k-1} + \delta(\mathbf{b} - \mathbf{A}\,\mathrm{vec}(\mathbf{X}^k))$ // update $\mathbf{y}_\mathrm{a}, \mathbf{y}_\mathrm{c}$
  $\begin{bmatrix}\mathbf{y}_{\mathrm{c}_i}^k \\ s_i^k\end{bmatrix} \leftarrow \mathcal{P}_i\left(\begin{bmatrix}\mathbf{y}_{\mathrm{c}_i}^{k-1} \\ s_i^{k-1}\end{bmatrix} - \delta\begin{bmatrix}\mathbf{C}_i\,\mathrm{vec}(\mathbf{X}^k) \\ d_i\end{bmatrix}\right)$
  $k \leftarrow k + 1$
end while
$\hat{\mathbf{X}} \leftarrow \mathbf{X}^k$

function $(\mathcal{P}_i(\mathbf{y}, s))$ // see Eq.4.32
  if $\|\mathbf{y}\| \leq s$ **then**
    $(\mathbf{z}, t) \leftarrow (\mathbf{y}, s)$
  else if $-\|\mathbf{y}\| \leq s \leq \|\mathbf{y}\|$ **then**
    $(\mathbf{z}, t) \leftarrow \frac{\|\mathbf{y}\|+s}{2\|\mathbf{y}\|}(\mathbf{y}, s)$
  else
    $(\mathbf{z}, t) \leftarrow (\mathbf{0}, 0)$
  end if
  Return $(\mathbf{z}, t)$
end function

function $(\mathcal{D}_\tau(\mathbf{X}))$
  $\sum_i \sigma_i \mathbf{u}_i \mathbf{v}_i^T \leftarrow \mathbf{X}$ // singular value decomposition
  $\mathbf{Y} \leftarrow \sum_i \max(\sigma_i - \tau, 0)\mathbf{u}_i\mathbf{v}_i^T$
  Return $\mathbf{Y}$
end function

different complexity and composed of a variety of actions for performance evaluations (see Table 2.2 in Chapter 2).

We compare the proposed skeleton constrained frame-based SVT mocap recovery method (SCFSVT) against the previous frame-based SVT mocap recovery method (FSVT) [24]. We also compare against BOLERO[2] [23], a method which models the problem as a linear dynamical system with skeleton constraints.

We use the BOLERO code provided publicly by the authors and the implementation which enforces hard constraints (BOLERO-HC). However, for our test data, since the algorithm only terminates long after the result has practically converged, for better fairness, we remove the stricter termination criterion that checks for a decrease in a likelihood funtion (likelihood increases logarithmically during optimization), and retain the criterion which checks whether the rate of change of likelihood has dropped below a threshold value. Modifying the code in this way decreases computation time substantially with very little change in performance. Implementations of all the compared algorithms are written in matlab and the experiments are run on a computer with an Intel i7-3770 CPU and 8GB memory.

In the same way as described in the previous chapter, we simulate random missing data by randomly removing a number of joints so that the required missing data rate is simulated. We simulate block missing data by partitioning a sequence into time intervals of the same length, and for each interval we randomly remove a fixed number of joints. The number of removed joints determines the error rate of the corrupted mocap sequence, and the interval length determines for how many frames is a joint missing. We vary the interval length from 15–120 frames (0.25–2 seconds). We use Root Mean Square Error

[2]Mocap data values are scaled down for algorithm stability as recommended, but are not transformed to local coordinates. All parameters are left at default values.

(RMSE) to quantify the distortion of a recovered matrix.

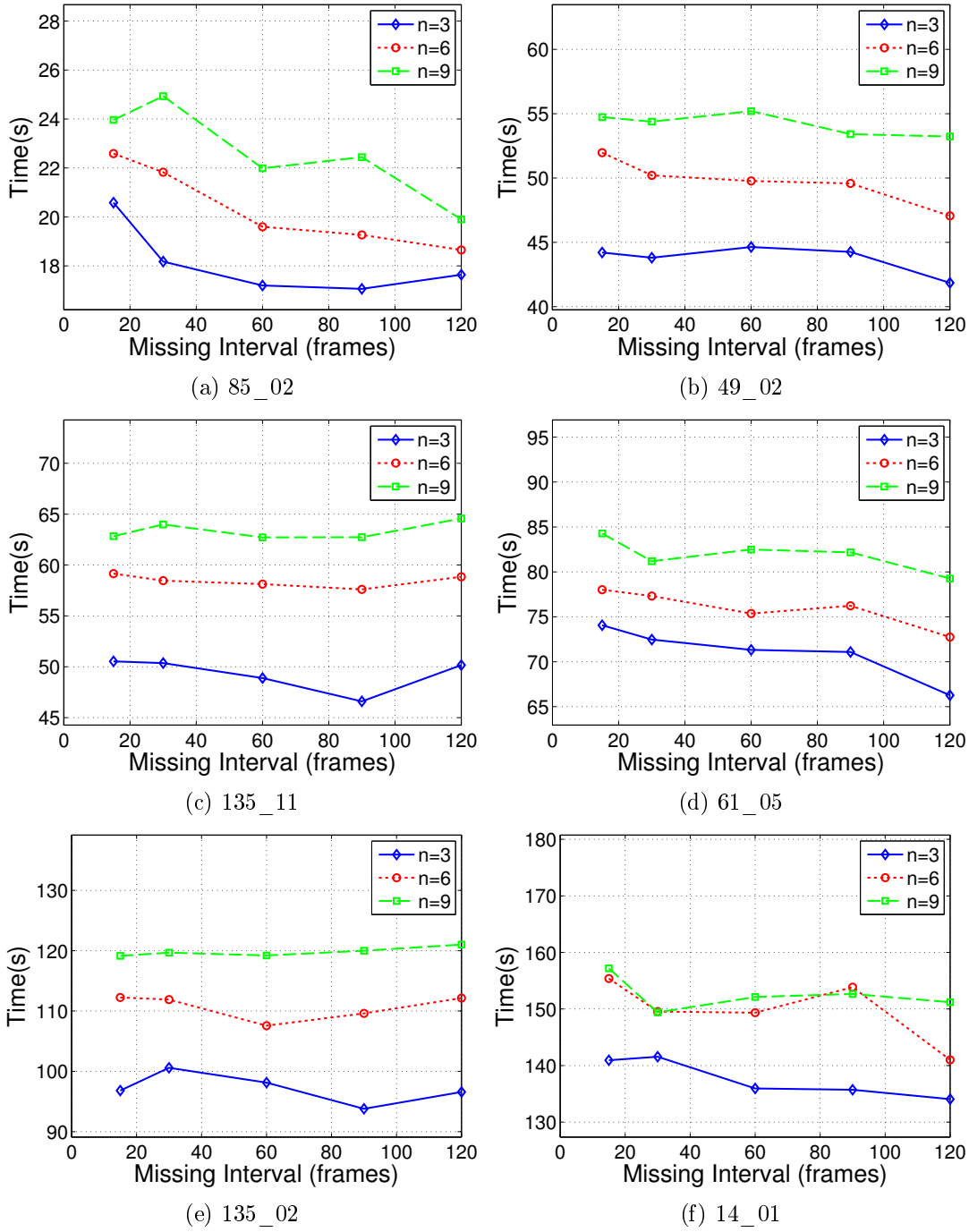
$$\text{RMSE}(\hat{\mathbf{X}}, \mathbf{M}) = \sqrt{\frac{1}{n_1 \times n_2} \sum_{j=1}^{n_2} \sum_{i=1}^{n_1} \left(\hat{\mathbf{X}}_{ij} - \mathbf{M}_{ij}\right)^2} \tag{4.33}$$

where $\hat{\mathbf{X}}$ and $\mathbf{M}$ are the matrices of the recovered and original mocap, respectively.

4.4.1 Processing Time Evaluation

The processing time for SCFSVT is strongly dependent on the number of frames of the mocap sequence, weakly dependent on the error rate, and generally not dependent on the missing interval length, as shown in Fig. 4.4. In the figure, the error rate is $\frac{n}{31}$, where n is the number of missing joints. This observation can be explained from the operations listed in Algorithm 2. When the error rate $1-\frac{m}{n_1 n_2}$ is increased, the dimension of $\mathbf{y_c} \in \mathbb{R}^{3p}$ and $\mathbf{C} \in \mathbb{R}^{3p \times n_2}$ increases while the dimension of $\mathbf{y} \in \mathbb{R}^m$ decreases. Thus the resulting processing time increase is largely due to the projection operation (step 5 in Algorithm 2). The most time consuming operation by far is the singular value decomposition. It scales with the number of frames n_2, and is independent of the error rate. The fluctuations across varying interval lengths, especially for a very short sequence like 85_02, can be attributed to the experimental setup; for each experiment, each sequence is truncated to multiples of missing interval length.

Compared with FSVT, the computation of SCFSVT requires the projection operation and operations associated with $\mathbf{y_c}$ and $\mathbf{C}$. Since the singular value decomposition operation is the same for both FSVT and SCFSVT, the complexity for each iteration is the same for both algorithms; the processing time is somewhat higher for SCFSVT, but not greatly so. However, SCFSVT has a lower δ threshold for convergence than FSVT. With a lower value of δ, SCFSVT requires many more iterations to converge; hence, it



(a) 85_02 (b) 49_02

(c) 135_11 (d) 61_05

(e) 135_02 (f) 14_01

Figure 4.4: Processing time of SCFSVT for varying interval lengths and number of missing joints n for six sequences.

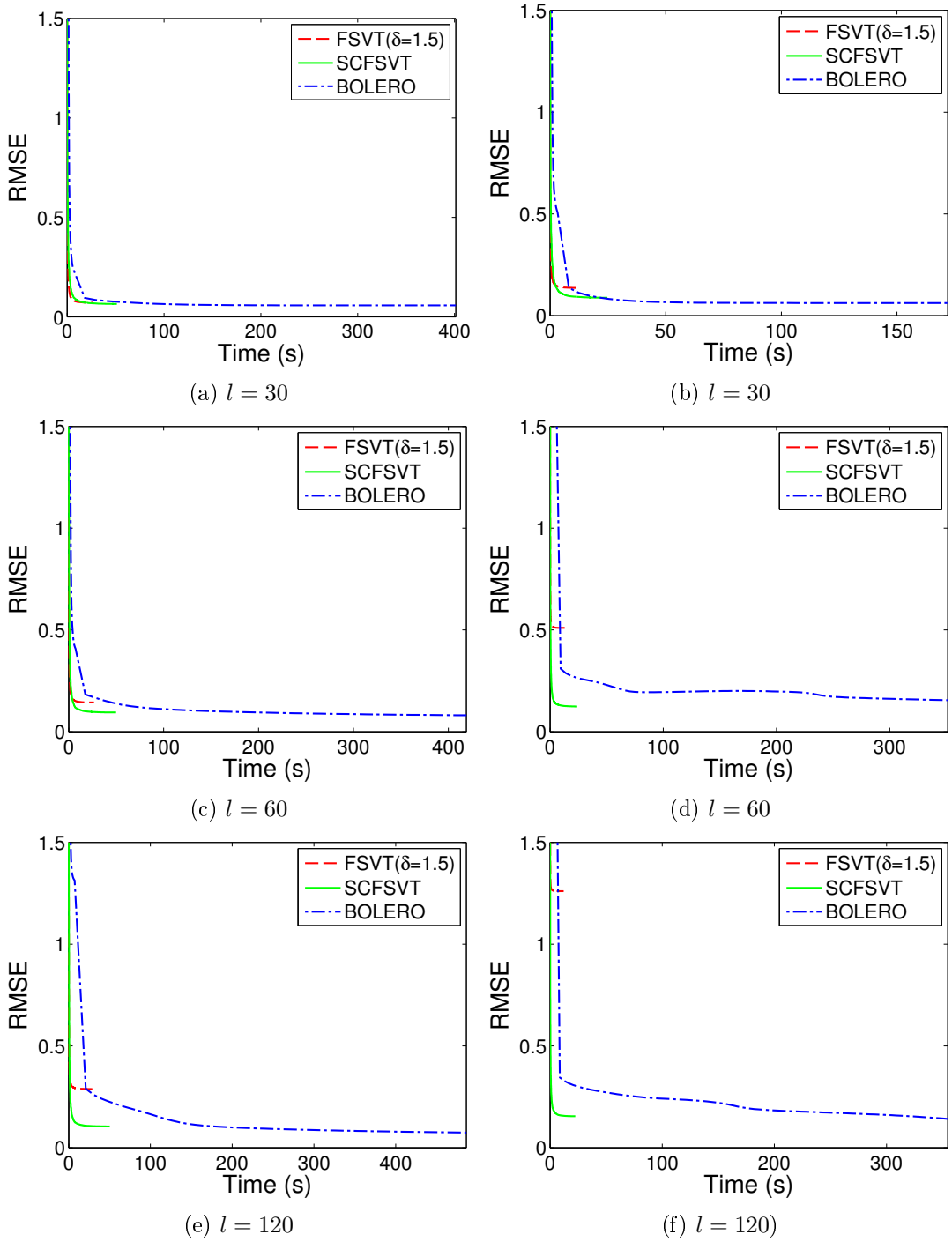
Sequence	N	A1	A2	B	C
85_02	405	6	15	18	164
143_04	543	8	21	25	291
49_02	1043	15	38	44	156
135_11	1223	17	42	49	438
88_04	1240	16	40	46	266
61_05	1678	23	57	71	241
135_02	2601	31	78	97	367
14_01	2797	44	110	138	480

Table 4.1: Average processing time (seconds) of (A1) FSVT($\delta = 1.9$), (A2) FSVT($\delta = 0.75$), (B) the proposed SCFSVT, and (C) BOLERO for 3 missing joints. N denotes the number of frames in a sequence.

requires a longer processing time.

We compare the processing times of SCFSVT($\delta = 0.75$) against that of FSVT($\delta = 0.75$), FSVT($\delta = 1.9$), and BOLERO in Table 4.1 of various sequences for three missing joints. The average processing time over interval lengths of {15, 30, 60, 90, 120} are shown. FSVT($\delta = 1.9$) is approximately $\frac{1.9}{0.75} \approx 2.5$ times as fast as FSVT($\delta = 0.75$), which is in turn up to 25% faster than SCFSVT. Nevertheless, SCFSVT is 3 to 11 times faster than BOLERO.

Fig. 4.5 shows how the compared algorithms converge for three missing joints. Both FSVT and SCFSVT converge smoothly, and the convergence rate is independent of interval length l. FSVT converges faster than SCFSVT, hence it has a lower RMS error initially, but eventually the RMS error of SCFSVT decreases to a smaller value. BOLERO converges more slowly than both FSVT and SCFSVT. Generally, convergence time of BOLERO is also longer for longer l.



(a) $l = 30$ (b) $l = 30$

(c) $l = 60$ (d) $l = 60$

(e) $l = 120$ (f) $l = 120$)

Figure 4.5: RMS error for sequences 135_11 (a, c, e) and 143_04 (b, d, f) at different interval lengths l for 3 missing joints.

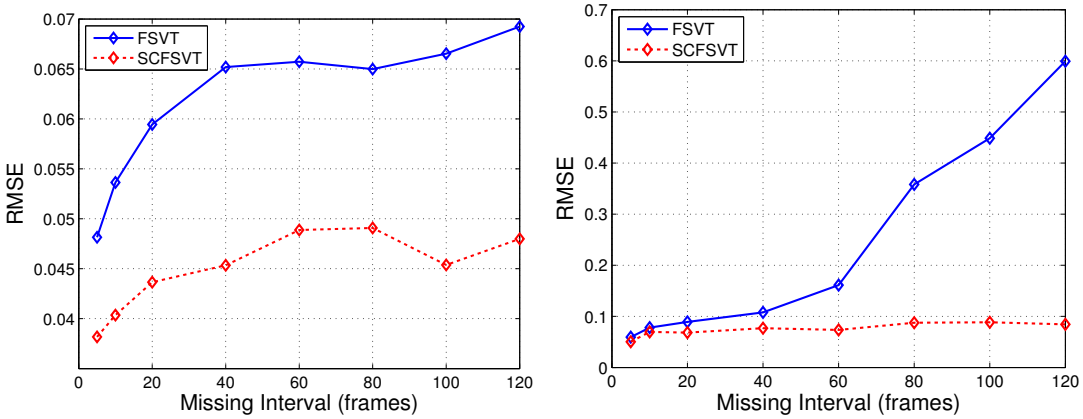
4.4.2 Performance Evaluation

We first show the effectiveness of the proposed method in constraining the inter-joint distance of joint pairs. We compute the skeleton RMS error

$$\mathrm{RMSE}_{\mathrm{skel}} = \sqrt{\frac{1}{k}\sum_{i=1}^{30}\left(\left\|\begin{bmatrix}x\\y\\z\end{bmatrix}_{\alpha_i} - \begin{bmatrix}x\\y\\z\end{bmatrix}_{\beta_i}\right\|_2 - d_i\right)^2} \tag{4.34}$$

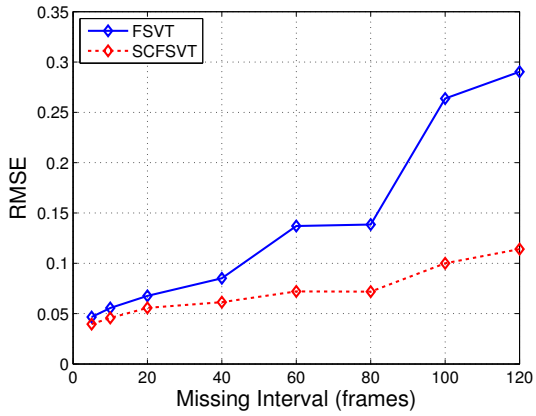
where for each ith bone its joints are α_i and β_i, and its length is d_i. There are 30 bones, as shown in Fig. 4.3.

As can be seen in Fig. 4.6, the skeleton RMSE for FSVT and SCFSVT generally increases with increasing interval length. However, the error for SCFSVT is lower and increases at a much lower rate.

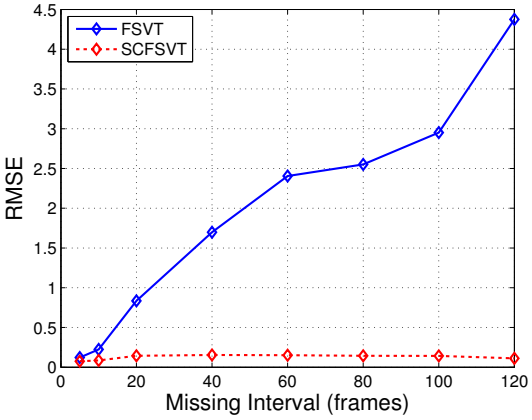


(a) 14_01

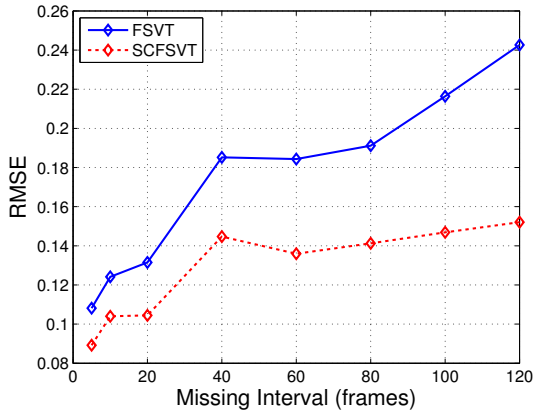
(b) 135_11



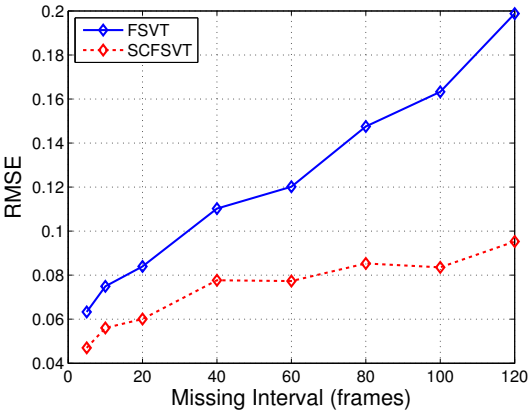
(c) 49_02



(d) 85_02



(e) 135_02



(f) 61_05

Figure 4.6: Skeleton RMS error for FSVT and SCFSVT for 3 missing joints.

Sequence		n=3, l=15	n=3, l=30	n=3, l=60	n=3, l=90	n=3, l=120	n=6, l=15	n=6, l=30	n=6, l=60	n=6, l=90	n=6, l=120
14_01	A	0.052	0.051	0.053	0.056	0.061	0.098	0.118	0.138	0.128	0.140
(Boxing)	B	**0.046**	**0.043**	**0.045**	**0.045**	**0.048**	**0.085**	**0.104**	**0.118**	**0.102**	**0.105**
	C	0.068	0.066	0.068	0.063	0.072	0.100	0.112	0.143	0.112	0.130
85_02	A	0.281	0.613	1.563	2.038	2.499	0.716	1.119	2.204	3.060	3.745
(Jumptwist)	B	**0.120**	0.191	0.230	**0.204**	**0.249**	**0.248**	**0.315**	**0.354**	**0.485**	**0.321**
	C	0.145	**0.154**	**0.214**	0.266	0.326	0.290	0.340	0.366	0.538	0.397
143_04	A	0.075	0.136	0.510	0.380	1.261	0.156	0.340	1.148	1.634	3.280
(Run fig 8)	B	0.059	0.088	**0.124**	**0.128**	0.155	0.117	0.193	0.220	**0.270**	**0.290**
	C	**0.057**	**0.062**	0.155	0.135	**0.141**	**0.105**	**0.143**	**0.174**	0.314	0.403
49_02	A	0.046	0.065	0.111	0.091	0.238	0.128	0.182	0.239	0.255	0.423
(Jumping)	B	**0.039**	**0.053**	0.086	0.068	0.093	0.119	0.140	**0.132**	0.146	**0.153**
	C	0.057	0.058	**0.074**	**0.064**	**0.079**	**0.103**	**0.099**	0.139	**0.133**	0.233
135_02	A	0.097	0.120	0.168	0.166	0.209	0.201	0.246	0.404	0.428	0.453
(MartialArts)	B	**0.083**	**0.096**	**0.139**	**0.131**	**0.151**	**0.181**	**0.211**	0.333	**0.247**	**0.280**
	C	0.138	0.144	0.144	0.161	0.195	0.216	0.228	**0.287**	0.266	0.282
135_11	A	0.057	0.071	0.144	0.261	0.288	0.132	0.180	0.326	0.500	0.608
(Kicking)	B	**0.052**	0.065	0.095	**0.095**	0.104	0.125	0.150	**0.184**	0.160	0.157
	C	0.054	**0.058**	**0.081**	0.099	**0.074**	**0.092**	**0.102**	0.383	**0.154**	**0.114**
61_05	A	0.070	0.086	0.102	0.167	0.174	0.159	0.206	0.233	0.390	0.344
(Salsa)	B	**0.060**	**0.063**	**0.080**	**0.100**	**0.102**	0.137	0.168	**0.144**	0.185	**0.179**
	C	0.073	0.085	0.086	0.100	0.112	**0.123**	**0.145**	0.156	**0.151**	0.209
88_04	A	0.131	0.202	0.274	0.342	0.347	0.363	0.617	0.576	1.036	0.949
(Acrobatics)	B	**0.101**	**0.123**	**0.146**	**0.147**	**0.141**	**0.265**	0.432	**0.317**	0.314	**0.261**
	C	0.155	0.143	0.201	0.178	0.177	0.270	**0.286**	0.341	**0.302**	0.297

Table 4.2: Performance comparison in terms of RMSE of recovered mocap data beween (A) FSVT, (B) the proposed SCFSVT mocap recovery, and (C) BOLERO under block missing data. n and l denote the number of missing joints and interval length, respectively.

Table 4.2 shows a comparison of mocap recovery performance under block missing data. First, we observe that in most cases the performance of FSVT decreases substantially when the time interval increases, whereas the performance of SCFSVT degrades more slowly. This shows that skeleton constraints improves the resistance of FSVT towards long bursts of joint data loss. Overall, SCFSVT outperforms FSVT by up to 90% and at 41% on average, with larger gains at longer missing joint intervals. Comparing SCFSVT with BOLERO, SCFSVT outperforms BOLERO in a slight majority of our tests with a 4% improvement on average.

While in general the RMSE of all algorithms increase with increased interval lengths, there are cases where RMSE fluctuates. For example, for sequence 85_02 at $n = 3$, the RMSE for SCFSVT rises from 0.191 to 0.230 at $l = 60$, and then falls to 0.204 at $l = 90$. This is not surprising since the simulated missing data entries are not the same for different l, and certain data entries might be more important for recovering the mocap sequence. For example, for $l = 60$, important mocap entries might be missing, but these entries are available for $l = 90$, hence in this case $l = 90$ might result in lower RMSE. The fact that this phenomenon occurs more frequently for SCFSVT and BOLERO shows that skeleton constraints are effective in reducing the performance degradation caused by longer interval lengths.

Table 4.3 shows a comparison of mocap recovery performance under random missing data. We only show experiments up to 40% missing data rate since BOLERO becomes unstable at higher missing data rates. In this test, although SCFSVT outperforms FSVT in all cases, the performance advantage of is smaller than in the block missing error test, at an average of 7% improvement. SCFSVT outperforms BOLERO at 10% and 20% missing rate, but loses out at higher missing rates.

Sequence	Method	Missing data			
		10%	20%	30%	40%
14_01	A	0.052	0.092	0.171	0.264
	B	**0.048**	**0.084**	0.159	0.248
	C	0.057	0.086	**0.115**	**0.143**
85_02	A	0.041	0.094	0.192	0.402
	B	**0.035**	**0.086**	0.182	0.379
	C	0.081	0.120	**0.159**	**0.199**
143_04	A	0.047	0.096	0.175	0.284
	B	**0.043**	0.089	0.166	0.269
	C	0.046	**0.074**	**0.103**	**0.130**
49_02	A	0.044	0.074	0.118	0.178
	B	**0.041**	0.070	0.114	0.172
	C	0.044	**0.064**	**0.085**	**0.131**
135_02	A	0.087	0.151	0.262	0.415
	B	**0.079**	**0.137**	0.242	0.388
	C	0.112	0.165	**0.225**	**0.335**
135_11	A	0.050	0.078	0.141	0.240
	B	**0.046**	0.074	0.135	0.231
	C	0.047	**0.071**	**0.099**	**0.118**
61_05	A	0.056	0.103	0.175	0.275
	B	**0.050**	**0.093**	0.162	0.258
	C	0.064	0.098	**0.127**	**0.159**
88_04	A	0.088	0.149	0.297	0.494
	B	**0.080**	**0.137**	**0.280**	0.470
	C	0.118	0.179	0.342	**0.464**

Table 4.3: Performance comparison of recovered mocap data beween (A) FSVT, (B) the proposed SCFSVT mocap recovery, and (C) BOLERO under random missing data.

4.4.3 Discussion

BOLERO is an extension of a system [22] that models the entire processed sequence as a linear dynamical system; a sequence of frames is modeled as a sequence of low-dimensional latent variables. Consecutive latent variables are related by a linear map, and each frame is related to a latent variable by a linear projection. The objective function to be maximized is the likelihood of the observed data with respect to the model. BOLERO improves on this system by including bone constraints in the objective function. Since the bone constraints are formulated as nonlinear constraints, the objective function becomes nonlinear, and is solved by using either Newton's method or the coordinate descent method. In comparison, our method formulates the skeleton constraint as an inequality constraint, which is convex, and we model mocap recovery as a convex optimization problem. Hence we can utilize convex optimization to find a global optimal solution efficiently.

With regard to the skeleton constraints in SCFSVT, it is likely that the performance of SCFSVT can be improved by tightening the bound of the constraint in Eq. 4.14, $\|\mathbf{C}_i\mathbf{x}\|_2 \leq d_i$, to

$$d_i - \epsilon \leq \|\mathbf{C}_i\mathbf{x}\|_2 \leq d_i + \epsilon \tag{4.35}$$

In other words, each bone is constrained to a certain length d_i with tolerance ϵ, instead of being constrained to have length of less than d_i. However, if we do so, we would be changing the constraint from a convex constraint (Eq. 4.14) into a non-convex constraint (Eq. 4.35). As a result, the problem becomes non-convex, and thus we would not be able to apply convex optimization; we lose the efficiency of convex optimization. Nevertheless, even with skeleton constraints with looser bound, SCFSVT still improves on FSVT by a large extent, and its overall performance is comparable to BOLERO.

4.5 Conclusion

Low rank matrix completion methods, such as Singular Value Thresholding (SVT), work well in recovering mocap data with missing values. However, the performance of SVT decreases substantially when joints are missing for a long time interval, due to fluctuating bone lengths of the recovered sequence. We alleviate this drawback by constraining the inter-joint distances of specific joint pairs in the SVT method. For situations where joints are missing for long intervals, experiments show that our proposed skeleton constrained frame-based SVT method (SCFSVT) improves on the frame-based SVT method (FSVT) by 40% on average, with higher gains at longer missing joint intervals. Compared to the recent state of the art algorithm, BOLERO[23], our method offers competitive performance and is 3 to 11 times faster.

The skeleton constraints are independent of the structure of the mocap matrix. Thus it is possible to impose skeleton constraints on a trajectory-based mocap matrix. However, since a trajectory that is missing entirely could not be reliably recovered under matrix completion, even the application of skeleton constraints might not be enough to obtain a good performance for long intervals of missing data.

Chapter 5

Subspace Constrained Mocap Recovery

5.1 Motivation

In Chapter 3, we introduced the trajectory-based representation for SVT matrix completion, which exploits different properties for mocap recovery compared with the frame-based representation; the trajectory-based representation exploits the correlation between short trajectory segments, whereas the frame-based representation exploits the correlation between frames of the sequence. Naturally, the next step would be to combine both methods in a way that both correlations between trajectory segments and correlations between frames could be exploited. A naive method to do so is by weighted addition of mocap data $\mathbf{X}_{\mathrm{A}}$ and $\mathbf{X}_{\mathrm{B}}$ recovered by method A and method B to obtain the final recovered data $\mathbf{X}_C$, i.e., $\mathbf{X}_{\mathrm{C}} = \alpha\mathbf{X}_{\mathrm{A}} + (1-\alpha)\mathbf{X}_{\mathrm{B}}$. However, this is not a good solution since properties of both methods are not considered together, and $\mathbf{X}_C$ will never outperform both $\mathbf{X}_A$ and $\mathbf{X}_B$.

In this chapter, we propose a method to 'combine' two different mocap recovery methods to exploit both correlation between frames and correlation between short trajectory segments. The property of correlation between frames means that, regardless of the representation of a mocap matrix, the frames, or entries corresponding to frames, of the matrix lie approximately in a low dimensional subspace. This means that for a frame-based matrix, the columns lie in a low dimensional subspace, whereas for a trajectory-based matrix, the entries corresponding to frames are constrained so that they lie in a low dimensional subspace. This subspace can be estimated from the columns of the matrix recovered by FSVT. Thus, we recover an initial solution using Frame-based Singular Value Thresholding (FSVT), and then recover the matrix again using Trajectory-based Singular Value Thresholding (TSVT) while constraining the solution of TSVT to lie in the frame correlation subspace estimated from the initial solution. We will refer to this combined method as Combined-Frame-and-Trajectory Singular Value Thresholding (CFTSVT).

We first describe how the low dimensional subspace is obtained from FSVT and formulated as a constraint in Section 5.2.1. In Section 5.2.2, we show how to reformulate TSVT to accept frame-based constraints. The value estimation of parameter $r_{\max}$ is described in Section 5.3.1, and experimental results are shown in Section 5.3.2. Finally we conclude the work in Section 5.4.

5.2 Proposed method

The proposed method recovers mocap data in two stages; first, the input mocap matrix is arranged in a frame-based representation $\mathbf{M}_{\mathrm{f}} \in \mathbb{R}^{n_1 \times n_2}$. FSVT is then used to recover $\mathbf{M}_{\mathrm{f}}$ to provide an inital solution, $\hat{\mathbf{X}}_{\mathrm{f}} \in \mathbb{R}^{n_1 \times n_2}$, and the low-dimensional subspace spanned by the frames of $\hat{\mathbf{X}}_{\mathrm{f}}$ is extracted (Section 5.2.1). At the second stage, we rearrange the input mocap matrix into a trajectory-based representation $\mathbf{M}_{\mathrm{t}} \in \mathbb{R}^{m_1 \times m_2}$, and then solve for

the final recovered mocap using the subspace constrained TSVT. In our implementation, as we will show later in Section 5.2.2, this rearranging operation step can be skipped by using a transformation matrix. .

5.2.1 Subspace Extraction

After computing $\hat{\mathbf{X}}_{\mathrm{f}}$, we first take its Singular Value Decomposition (SVD),

$$\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T = \mathrm{SVD}(\hat{\mathbf{X}}_{\mathrm{f}}) \tag{5.1}$$

The columns of $\mathbf{U} = [\mathbf{u}_1 \dots \mathbf{u}_{n_1}] \in \mathbb{R}^{n_1 \times n_1}$ span the subspace that the frames of $\hat{\mathbf{X}}_{\mathrm{f}}$ lie in. The optimal rank r approximation of this subspace is obtained by retaining the first r columns of $\mathbf{U}$. In practice, the rank of $\hat{\mathbf{X}}_{\mathrm{f}}$ can be as high as 60, so we obtain a low dimensional subspace $\mathcal{U}$ by setting the maximum rank $r_{\max}$,

$$\mathcal{U} = \{\mathbf{u}_1, \dots, \mathbf{u}_r\} \tag{5.2}$$

$$\text{where} \quad r = \min(r_{\max}, \mathrm{rank}(\hat{\mathbf{X}}_{\mathrm{f}})) \tag{5.3}$$

Given a frame $\mathbf{x}_{\mathrm{f}}$ that does not lie in $\mathcal{U}$ the error vector of its orthogonal projection on $\mathcal{U}$, $\mathbf{e}_{\mathcal{U}}(\mathbf{x}_i)$, is

$$\mathbf{e}_{\mathcal{U}}(\mathbf{x}_{\mathrm{f}}) = \mathbf{P}_{\mathcal{U}^\perp}\mathbf{x}_{\mathrm{f}} \tag{5.4}$$

$$\text{where} \quad \mathbf{P}_{\mathcal{U}^\perp} = \mathbf{I} - \mathbf{U}_r\mathbf{U}_r^T \tag{5.5}$$

with $\mathbf{U}_r = [\mathbf{u}_1 \dots \mathbf{u}_r]$. Hence the subspace constraint for the second stage recovery method is

$$\|\mathbf{P}_{\mathcal{U}^\perp}\mathbf{x}_{\mathrm{f}_i}\|_2 = 0 \tag{5.6}$$

where $\mathbf{P}_{\mathcal{U}^\perp} \in \mathbb{R}^{n_1 \times n_1}$ is the orthogonal projector onto $\mathcal{U}^\perp$ and $\mathbf{X}_\mathrm{f} = [\mathbf{x}_{\mathrm{f}_1} \dots \mathbf{x}_{\mathrm{f}_{n_2}}] \in \mathbb{R}^{n_1 \times n_2}$ is the decision variable in a frame-based representation. Each frame of the decision variable is constrained to lie in $\mathcal{U}$.

5.2.2 Subspace Constrained TSVT

We first formulate the problem for a subspace constrained FSVT method, and then extend it to subspace constrained TSVT.

Since we want to apply the subspace constraints to every frame of the decision variable, we have

$$\|\mathbf{P}_{\mathcal{U}^\perp}\mathbf{x}_{\mathrm{f}_i}\|_2 = 0, \quad i = 1...n_2 \tag{5.7}$$

Now we try to denote the constraint in terms of $\mathbf{x}_\mathrm{f}$. Consider $\mathbf{P}_\mathrm{f}\mathbf{x}_\mathrm{f}$, where

$$\mathbf{P}_\mathrm{f} = \begin{bmatrix} \mathbf{P}_{\mathcal{U}^\perp} & 0 & \dots \\ 0 & \mathbf{P}_{\mathcal{U}^\perp} & \dots \\ \vdots & \vdots & \ddots \end{bmatrix}, \quad \mathbf{x}_\mathrm{f} = \mathrm{vec}(\mathbf{X}_\mathrm{f}) = \begin{bmatrix} \mathbf{x}_{\mathrm{f}1} \\ \vdots \\ \mathbf{x}_{\mathrm{f}n} \end{bmatrix} \tag{5.8}$$

Denoting $(\mathbf{P}_\mathrm{f})_i$ as the ith block-row of $\mathbf{P}_\mathrm{f}$, where each block row is a block of n_1 rows, we

have $(\mathbf{P}_\mathrm{f})_i\mathbf{x}_\mathrm{f} \coloneqq \mathbf{P}_{\mathcal{U}^\perp}\mathbf{x}_{\mathrm{f}_i}$. We can thus formulate the subspace constrained FSVT

$$\begin{aligned} \min \quad & f(\mathbf{X}_\mathrm{f}) && (5.9)\\ \text{s.t.} \quad & \mathbf{x}_\mathrm{f} = \mathrm{vec}(\mathbf{X}_\mathrm{f}) && (5.10)\\ & \mathbf{A}_\mathrm{f}\mathbf{x}_\mathrm{f} = \mathbf{b} && (5.11)\\ & \|(\mathbf{P}_\mathrm{f})_i\mathbf{x}_\mathrm{f}\|_2 \leq 0, \quad i = 1...n_2 && (5.12) \end{aligned}$$

where $\mathbf{A}_\mathrm{f} \in \mathbb{R}^{n_b \times n_1 n_2}$ extracts the n_b observed entries $\mathbf{b} \in \mathbb{R}^{n_b}$ from the input matrix $\mathbf{M} \in \mathbb{R}^{n_1 \times n_2}$, i.e., $\mathbf{b} = \mathbf{A}_\mathrm{f}\,\mathrm{vec}(\mathbf{M})$. Instead of specifying the subspace constraint using an equality constraint (in a similar way to Eq. 5.7), we use an inequality constraint (Eq. 5.12); this is because the equality constraint is non-convex. Moreover, notice that $(\mathbf{P}_\mathrm{f})_i\mathbf{x}_\mathrm{f}$ has the same form as the skeleton constraint in Eq. 4.14.

Now we cast the formulation to a trajectory-based representation by changing the decision variable in Problem 5.9 to $\mathbf{X}_\mathrm{t}$

$$\begin{aligned} \min \quad & f(\mathbf{X}_\mathrm{t}) && (5.13)\\ \text{s.t.} \quad & \mathbf{x}_\mathrm{t} = \mathrm{vec}(\mathbf{X}_\mathrm{t}) && (5.14)\\ & \mathbf{A}_\mathrm{f}\mathbf{x}_\mathrm{f} = \mathbf{b} && (5.15)\\ & \|(\mathbf{P}_\mathrm{f})_i\mathbf{x}_\mathrm{f}\|_2 \leq 0, \quad i = 1...n_2 && (5.16) \end{aligned}$$

where $\mathbf{b} = \mathbf{A}_\mathrm{f}\,\mathrm{vec}(\mathbf{M})$. Note that Constraint 5.15 is equivalent to $\mathbf{A}_\mathrm{t}\mathbf{x}_\mathrm{t} = \mathbf{b}_\mathrm{t}$, where $\mathbf{b}_\mathrm{t} = \mathbf{A}_\mathrm{t}\,\mathrm{vec}(\mathbf{M}_\mathrm{t})$ and $\mathbf{M}_\mathrm{t}$ is the input matrix in trajectory-based representation.

The decision variable $\mathbf{X}_\mathrm{t}$ is a matrix in a trajectory-based representation, whereas the subspace constraint in Eq. 5.16 is formulated in terms of $\mathbf{x}_{\mathrm{f}i}$, which is expressed in a frame-based representation. We need to express all equations using the same representation.

Thus we define a permutation matrix $\mathbf{T}_{\mathrm{tf}} \in \mathbb{R}^{n_1 n_2 \times n_1 n_2}$ that transforms a trajectory-based representation to a frame-based representation,

$$\mathbf{T}_{\mathrm{tf}}\mathbf{x}_{\mathrm{t}} = \mathbf{x}_{\mathrm{f}} \tag{5.17}$$

$$\text{where} \quad \mathbf{x}_{\mathrm{f}} = \mathrm{vec}(\mathbf{X}_{\mathrm{f}}), \tag{5.18}$$

$$\mathbf{x}_{\mathrm{t}} = \mathrm{vec}(\mathbf{X}_{\mathrm{t}}) \tag{5.19}$$

$$\mathrm{vec} : [\mathbf{x}_1 \dots \mathbf{x}_n] \mapsto \begin{bmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_n \end{bmatrix} \tag{5.20}$$

Substituting Eq. 5.17 into $\mathbf{P}_{\mathrm{f}}\mathbf{x}_{\mathrm{f}}$, we have

$$\mathbf{P}_{\mathrm{f}}\mathbf{x}_{\mathrm{f}} = \mathbf{P}_{\mathrm{f}}\mathbf{T}_{\mathrm{tf}}\mathbf{x}_{\mathrm{t}} \tag{5.21}$$

$$(\mathbf{P}_{\mathrm{f}})_i\mathbf{x}_{\mathrm{f}} = (\mathbf{P}_{\mathrm{f}}\mathbf{T}_{\mathrm{tf}})_i\mathbf{x}_{\mathrm{t}} \tag{5.22}$$

where $(\mathbf{P}_{\mathrm{f}}\mathbf{T}_{\mathrm{tf}})_i$ is the ith block row.

Substituting Eq. 5.17 and Eq. 5.22 into Problem 5.13, we have the problem for subspace constrained TSVT

$$\min \quad f(\mathbf{X}_{\mathrm{t}}) \tag{5.23}$$

$$\text{s.t.} \quad \mathbf{x}_{\mathrm{t}} = \mathrm{vec}(\mathbf{X}_{\mathrm{t}}) \tag{5.24}$$

$$\mathbf{A}_{\mathrm{f}}\mathbf{T}_{\mathrm{tf}}\mathbf{x}_{\mathrm{t}} = \mathbf{b} \tag{5.25}$$

$$\|(\mathbf{P}_{\mathrm{f}}\mathbf{T}_{\mathrm{tf}})_i\mathbf{x}_{\mathrm{t}}\|_2 \leq 0, \quad i = 1...n_2 \tag{5.26}$$

where $\mathbf{b} = \mathbf{A}_{\mathrm{f}}\,\mathrm{vec}(\mathbf{M})$, $(\mathbf{P}_{\mathrm{f}}\mathbf{T}_{\mathrm{tf}})_i \in \mathbb{R}^{n_1 \times n_1 n_2}$, and n_2 is the number of frames in the mocap sequence. Eq. 5.26 constrains each frame of $\mathbf{X}_{\mathrm{f}}$ to lie in a certain subspace, but is

expressed in terms of $\mathbf{x}_t$.

Notice that Problem 5.23 has the same form as Problem 4.11 in Chapter 4. Thus the derivation to the solution of this problem can be referred to in Chapter 4. The pseudocode for the subspace constrained TSVT is shown in Algorithm 3. Note that Algorithm 3 does not include the subspace extraction procedure, and that the recovered matrix is in trajectory-based representation.

5.3 Results and Discussion

Similar to the earlier chapters, use a set of sequences (Table 2.1) to estimate the value of $r_{\max}$, and another set of sequences (Table 2.2) for performance evaluations. Implementations of all compared algorithms are written in matlab, and the experiments are run on a computer with an Intel i7-3770 CPU and 8GB memory. The τ parameter for FSVT and subspace constrained TSVT algorithms are set to $8\sqrt{n_1 n_2}$ and $8\sqrt{m_1 m_2}$, respectively. The trajectory length for TSVT is set to $L = 50$.

The simulated missing markers are the same as previous chapters. For random missing data, missing data is simulated by randomly removing a number of marker entries. For block missing data, a sequence is partitioned into time intervals of the same length, and for each interval a fixed number of markers is removed. The number of removed markers determines the error rate of the corrupted mocap data.

We use Root Mean Square Error[1] (RMSE) to quantify the distortion of a recovered matrix.

$$\mathrm{RMSE}(\hat{\mathbf{X}}_2, \mathbf{M}) = \sqrt{\frac{1}{n_1 \times n_2} \sum_{j=1}^{n_2} \sum_{i=1}^{n_1} \left(\hat{\mathbf{X}}_{ij} - \mathbf{M}_{ij}\right)^2} \tag{5.27}$$

where $\hat{\mathbf{X}}_2$ and $\mathbf{M}$ are the matrices of the recovered and original mocap, respectively.

[1]We use the provided default units for mocap values. To obtain values in millimeters, multiply them by 56.44

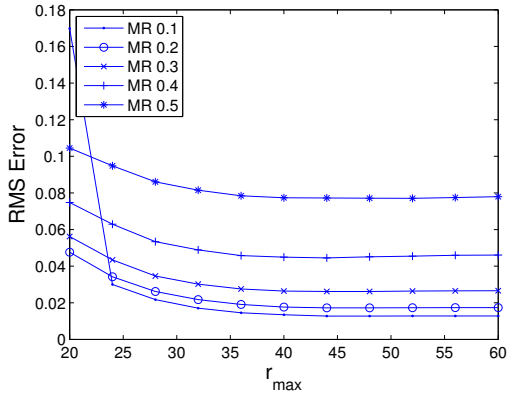
Algorithm 3 Subspace Constrained SVT

Input: $\mathbf{A}_\mathrm{f}, \mathbf{P}_\mathrm{f}, \mathbf{T}_\mathrm{tf}, \mathbf{b}$
Parameter: δ, τ, k_max, tol
Output: Recovered mocap sequence $\hat{\mathbf{X}}$
Initialization:
$\mathbf{y}_\mathrm{a}^0 \leftarrow \mathbf{b}$
$\mathbf{y}_\mathrm{c}^0 \leftarrow \mathbf{0}$
$\mathbf{s}_c^0 \leftarrow 0$
$k \leftarrow 1$
$\mathbf{A} \leftarrow \mathbf{A}_\mathrm{f}\mathbf{T}_\mathrm{tf}$
$\mathbf{C} \leftarrow \mathbf{P}_\mathrm{f}\mathbf{T}_\mathrm{tf}$

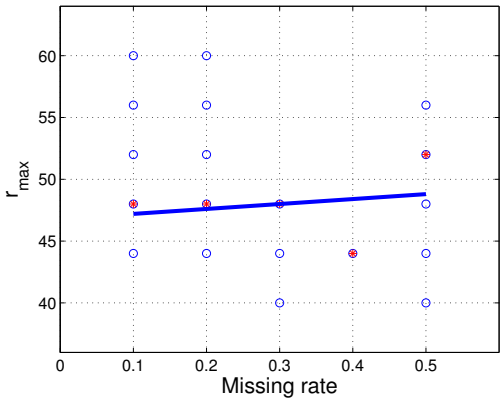
while $k < k_{max}$ **and** $\mathrm{RMSE}(\mathbf{A}\,\mathrm{vec}(\mathbf{X}^k), \mathbf{b}) > tol$ **do**
  $\mathbf{Y} \leftarrow \mathrm{vec}^*\left(\begin{bmatrix}\mathbf{A}^T & \mathbf{C}^T\end{bmatrix}\begin{bmatrix}\mathbf{y}_\mathrm{a}^{k-1} \\ \mathbf{y}_\mathrm{c}^{k-1}\end{bmatrix}\right)$ // update $\mathbf{X}$
  where $\mathbf{C} = \begin{bmatrix}\mathbf{C}_1 \\ \vdots \\ \mathbf{C}_p\end{bmatrix}, \mathbf{y}_\mathrm{c} = \begin{bmatrix}\mathbf{y}_{\mathrm{c}_1} \\ \vdots \\ \mathbf{y}_{\mathrm{c}_p}\end{bmatrix}$ // $\mathbf{C}_i \in \mathbb{R}^{n_1 \times n_1 n_2}$
  $\mathbf{X}^k \leftarrow \mathcal{D}_\tau(\mathbf{Y})$
  $\mathbf{y}_\mathrm{a}^k \leftarrow \mathbf{y}_\mathrm{a}^{k-1} + \delta(\mathbf{b} - \mathbf{A}\,\mathrm{vec}(\mathbf{X}^k))$ // update $\mathbf{y}_\mathrm{a}, \mathbf{y}_\mathrm{c}$
  $\begin{bmatrix}\mathbf{y}_{\mathrm{c}_i}^k \\ s_i^k\end{bmatrix} \leftarrow \mathcal{P}_i\left(\begin{bmatrix}\mathbf{y}_{\mathrm{c}_i}^{k-1} \\ s_i^{k-1}\end{bmatrix} - \delta\begin{bmatrix}\mathbf{C}_i\,\mathrm{vec}(\mathbf{X}^k) \\ d_i\end{bmatrix}\right)$
  $k \leftarrow k + 1$
end while
$\hat{\mathbf{X}} \leftarrow \mathbf{X}^k$

function $(\mathcal{P}_i(\mathbf{y}, s))$ // see Eq.4.32
  if $\|\mathbf{y}\| \le s$ **then**
    $(\mathbf{z}, t) \leftarrow (\mathbf{y}, s)$
  else if $-\|\mathbf{y}\| \le s \le \|\mathbf{y}\|$ **then**
    $(\mathbf{z}, t) \leftarrow \frac{\|\mathbf{y}\|+s}{2\|\mathbf{y}\|}(\mathbf{y}, s)$
  else
    $(\mathbf{z}, t) \leftarrow (\mathbf{0}, 0)$
  end if
  Return $(\mathbf{z}, t)$
end function

function $(\mathcal{D}_\tau(\mathbf{X}))$
  $\sum_i \sigma_i \mathbf{u}_i \mathbf{v}_i^T \leftarrow \mathbf{X}$ // singular value decomposition
  $\mathbf{Y} \leftarrow \sum_i \max(\sigma_i - \tau, 0)\mathbf{u}_i\mathbf{v}_i^T$
  Return $\mathbf{Y}$
end function



(a) Mean RMS errors



(b) Line fitted to $r_{\max}$ with minimum RMS error (shown in asterisks). Circles show $r_{\max}$ with RMS error within 1% of the minimum RMS error

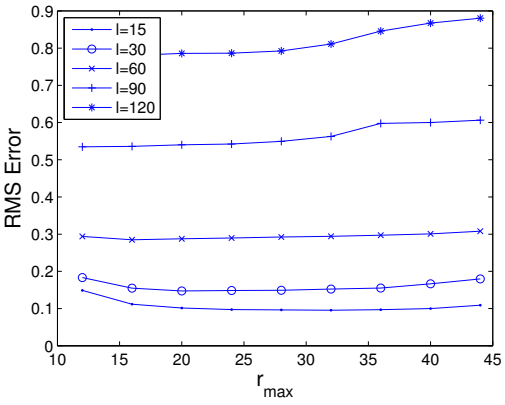
Figure 5.1: Estimation of optimal value for $r_{\max}$ of constraining subspace for random missing data.

We first investigate the optimal rank $r_{\max}$ for our proposed method in Section 5.3.1. Then we evaluate our proposed method using this length in Section 5.3.2.

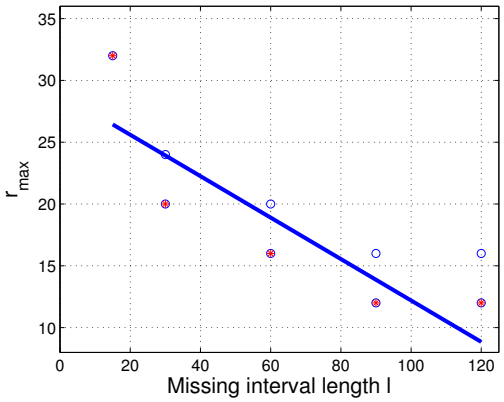
5.3.1 Subspace Rank Evaluation

In this section, we estimate the optimal value for $r_{\max}$ (Eq. 5.3) separately for random missing data and block missing data. For random missing data, for each missing rate, we obtain the average of RMS errors over all sequences (Fig. 5.1(a)). Then, we take the r_{max} that gives the minimum RMS error for each missing rate and fit a line to them using least squares (Fig. 5.1(b)). We can see that the optimal value of $r_{\max}$ is nearly constant for different missing rates, so we set $r_{\max} = 48$.

For block missing data, for each $l = 15$ to $l = 120$ at three missing markers, we repeat the same procedure (Fig. 5.2). In this case, $r_{\max}$ is clearly decreasing with increasing missing interval length l. Thus, we vary $r_{\max}$ according to the interval length. For the



(a) Mean RMSE



(b) Line fitted to $r_{\max}$ with minimum RMSE (shown in asterisks). Circles show $r_{\max}$ with RMSE within 1% of the minimum RMSE

Figure 5.2: Estimation of optimal value for $r_{\max}$ of constraining subspace for block missing data.

following experiments, we set $r_{\max}$ to (26,24,19,14,9) corresponding to missing interval lengths of (15,30,60,90,120).

5.3.2 Performance Evaluation

In this section, we compare the proposed CFTSVT with frame-based SVT (FSVT) and trajectory-based SVT (TSVT). It can be seen from Table 5.1 that for random missing data, CFTSVT outperforms FSVT by a wide margin in all cases, and except for a few cases at low missing data rates, outperforms TSVT. The performance advantage increases at higher missing data rates. On average, CFTSVT outperforms FSVT by 76% and TSVT by 23%.

For block missing data, as shown in Table 5.2, TSVT has the highest error. FSVT performs the best overall and outperforms CFTSVT mostly at long values of l, and by 13% on average. Still, CFTSVT outperforms FSVT in 143_04 (Running) and performs nearly the same for 85_02 (Jumptwist).

Sequence	Method	Missing data					
		10%	20%	30%	40%	50%	60%
14_01	A	0.052	0.092	0.171	0.264	0.404	0.614
(Boxing)	B	0.014	0.023	0.038	0.066	0.121	0.211
	C	**0.013**	**0.017**	**0.025**	**0.037**	**0.055**	**0.105**
85_02	A	0.041	0.094	0.192	0.402	0.661	1.193
(Jumptwist)	B	0.020	0.035	0.070	0.179	0.333	0.678
	C	**0.017**	**0.027**	**0.055**	**0.144**	**0.254**	**0.556**
143_04	A	0.047	0.096	0.175	0.284	0.459	0.857
(Run fig 8)	B	0.020	0.040	0.075	0.154	0.314	0.781
	C	**0.016**	**0.026**	**0.049**	**0.093**	**0.170**	**0.373**
49_02	A	0.044	0.074	0.118	0.178	0.265	0.438
(Jumping)	B	0.014	0.022	0.034	0.053	0.099	0.201
	C	**0.014**	**0.018**	**0.023**	**0.033**	**0.045**	**0.079**
135_02	A	0.087	0.151	0.262	0.415	0.645	1.025
(MartialArts)	B	**0.022**	0.034	0.053	0.081	0.139	0.265
	C	0.024	**0.033**	**0.044**	**0.064**	**0.102**	**0.202**
135_11	A	0.050	0.078	0.141	0.240	0.380	0.683
(Kicking)	B	0.017	0.029	0.049	0.084	0.130	0.254
	C	**0.016**	**0.025**	**0.041**	**0.063**	**0.088**	**0.166**
61_05	A	0.056	0.103	0.175	0.275	0.417	0.657
(Salsa)	B	**0.014**	**0.019**	0.027	0.039	0.070	0.135
	C	0.015	0.019	**0.025**	**0.035**	**0.054**	**0.107**
88_04	A	0.088	0.149	0.297	0.494	0.774	1.201
(Acrobatics)	B	**0.018**	0.031	0.052	0.088	0.154	0.308
	C	0.020	**0.028**	**0.041**	**0.064**	**0.103**	**0.222**

Table 5.1: Performance comparison of (A) FSVT, and (B) TSVT, and (C) CFTSVT under random missing data.

Sequence	Method	Missing data				
		l=15	l=30	l=60	l=90	l=120
14_01	A	**0.052**	**0.051**	**0.053**	**0.056**	**0.061**
	B	0.465	1.163	2.325	2.855	3.081
	C	0.068	0.074	0.095	0.136	0.285
85_02	A	0.281	**0.613**	**1.563**	2.038	**2.498**
	B	0.776	2.053	3.673	3.842	4.139
	C	**0.272**	0.623	1.576	**2.038**	2.515
143_04	A	0.075	0.136	0.510	0.380	1.261
	B	0.946	2.484	4.220	4.807	5.225
	C	**0.07**	**0.112**	**0.491**	**0.315**	**1.239**
49_02	A	**0.046**	0.065	0.111	**0.091**	**0.238**
	B	0.521	1.263	2.334	2.543	2.555
	C	0.046	**0.064**	**0.108**	0.122	0.286
135_02	A	**0.097**	**0.120**	**0.168**	**0.166**	**0.209**
	B	0.646	1.692	3.318	3.952	4.165
	C	0.121	0.141	0.189	0.267	0.492
135_11	A	0.057	0.071	0.144	**0.261**	**0.288**
	B	0.686	1.903	3.641	4.155	4.536
	C	**0.055**	**0.071**	**0.142**	0.269	0.339
61_05	A	**0.070**	**0.086**	**0.102**	**0.167**	**0.174**
	B	0.462	1.375	2.835	3.339	3.615
	C	0.078	0.096	0.124	0.187	0.289
88_04	A	**0.131**	0.202	**0.274**	**0.342**	**0.347**
	B	0.887	1.865	3.095	3.580	3.672
	C	0.132	**0.193**	0.274	0.389	0.650

Table 5.2: Performance comparison of (A) FSVT, and (B) TSVT, and (C) CFTSVT under block missing data.

The dimension of the constraint subspace determines the relative contribution of the outputs of the two stages of CFTSVT. A lower dimensional subspace constrains the output of stage two TSVT to be very close to the results of stage one FSVT; hence, using a lower dimension increases the relative contribution from FSVT.

5.4 Conclusion

Different mocap recovery methods exploit different characteristics of the mocap data to do recovery; frame-based Singular Value Decomposition (FSVT) exploits the correlation between frames whereas trajectory-based Singular Value Thresholding (TSVT) exploits the correlation between short trajectory segments. In this chapter, we proposed the method Combined-Frame-and-Trajectory Singular Value Thresholding (CFTSVT), which exploits both properties. A mocap matrix has a correlation between frames; this correlation induces the frames of a mocap matrix lie in a low dimensional subspace, which we can approximate using the matrix recovered using FSVT. This correlation is then exploited in TSVT, which by itself only exploits correlation between trajectories, by constraining the frames in the solution of TSVT to lie in the low dimensional subspace extracted previously. In short, the output of FSVT is used to constrain the output of TSVT. The proposed CFTSVT method significantly outperforms both FSVT and TSVT in random missing data tests.

Chapter 6

Single Viewpoint Image-Driven Simplification

6.1 Introduction

Mesh simplification is the process of reducing the resolution of polygonal meshes. In the incremental decimation framework, a mesh is simplified by repeated vertex removal in a greedy fashion; at each vertex removal step, the vertex removed is the one that causes the least distortion when removed. Incremental decimation algorithms and their performances differ by how they define distortion at each step of vertex removal. The majority of algorithms use geometry-based error metrics, which measure spatial distortion such as volume distortion and distance distortion. To account for perceived distortions due to distortion in other properties that affect appearance, like normals and texture, a few works [49, 50, 51] proposed a separate 'appearance distortion' and combine it with spatial distortion using weighted summation to obtain an aggregate distortion.

However, the relationship between perceived distortion and the measured distortions,

i.e., 'appearance distortion' and spatial distortion, is not straightforward; 'appearance distortion' can affect the perceptibility of spatial distortion. Besides, 'appearance distortion' and spatial distortion refer to measures that are very different in nature. Hence, summing both types of distortion is a compromised approach.

Lindstrom and Turk suggested an alternative approach — Image-Driven Simplification (IDS) [59], where the appearance of a mesh is measured directly. Hence IDS measures perceived distortion directly by measuring the change in appearance of a mesh.

The mesh appearance can be fully described by a set of radiance samples emanating from the mesh. Since it is impractical to capture all the radiance samples, IDS only aims to capture a subset of these samples. IDS approximates mesh appearance using mesh images that are captured from a set of evenly distributed viewpoints. The appearance error between two meshes is obtained by aggregating the differences between their corresponding images. Increasing the number of viewpoints (and captured images) will increase the amount of captured radiance samples; this increment improves the accuracy of the approximation of mesh appearance. Thus, using more viewpoints improves simplification performance at a cost of longer processing time. Incidentally, the approach of using a set of images as a proxy for a 3D mesh has also been used successfully in content-based mesh retrieval [72, 73].

Incremental decimation is done by repeatedly applying an edge collapse operation whereby two vertices are merged together. Since only a small region is distorted by an edge collapse, obviously only a small set of radiance samples are affected by one such operation. In IDS, 20 viewpoints are used to capture mesh appearance. We observe, however, that in most cases, for each edge collapse, only a few viewpoints will register any image distortion.

We thus propose that instead of placing the viewpoints at a fixed location and ori-

entation around the mesh, we should place the viewpoints relative to each edge collapse region. In this way, we only need a few viewpoints to capture the same amount of appearance distortion as the previous method. We go one step further and propose that with proper positioning of the viewpoint, a single viewpoint is sufficient for good simplification quality. By reducing the number of viewpoints to one, we achieve significant savings in processing time.

We first describe the Image-Driven Simplification method in Section 6.2. Then we explain our faster single viewpoint method in Section 6.3. In Subsection 6.3.1 and Section 6.3.2, we proposed two methods to place the viewpoint, and in Section 6.3.3 we described a method to accelerate our single viewpoint methods. We then present processing time improvements in Section 6.4.1 and simplification quality comparisons in Section 6.4.2, followed by a Conclusion in Section 6.6.

6.2 Image-Driven Simplification

In the incremental decimation framework, the final simplified mesh M^n is obtained from the original mesh $\hat{M}$ through a series of edge collapse (ecol) operations, where a vertex is collapsed into an adjacent vertex, removing a vertex from the mesh. Hence we have a set of intermediate meshes, each with one less vertex than the previous mesh.

$$\hat{M} \overset{\text{ecol}}{\rightarrow} ... \overset{\text{ecol}}{\rightarrow} M^{k-1} \overset{\text{ecol}}{\rightarrow} M^k \overset{\text{ecol}}{\rightarrow} ... \overset{\text{ecol}}{\rightarrow} M^n$$

The simplification process from $\hat{M}$ to M^n is done in a greedy manner. At any stage k of simplification $M^{k-1} \overset{\text{ecol}}{\rightarrow} M^k$, the collapse errors $E(\text{ecol}(x, y))$ of all candidate vertex pairs (x, y) are computed, and then edge collapse is applied on the pair with the smallest collapse error to obtain M^k. This collapse error is defined as the difference between the

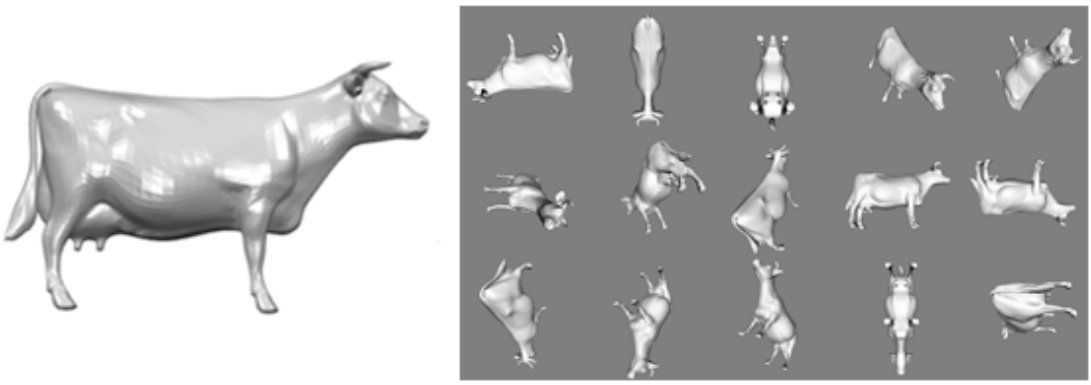


Figure 6.1: (left) Cow mesh; (right) Images rendered from a set of equally distributed viewpoints around the mesh.

original mesh $\hat{M}$ and the mesh $M^k(x,y)$, where $M^k(x,y)$ is the mesh after the collapse of (x,y) on M^{k-1}. Formally, we have

$$(x,y)^k = \arg\min_{x,y} E(\text{ecol}(x,y)) \tag{6.1}$$

$$= \arg\min_{x,y} E(\hat{M}, M^k(x,y)) \tag{6.2}$$

Note that (x,y) has to be collapsed first in order to evaluate the collapse error of (x,y), so all candidate pairs have to be independently collapsed first for the error evaluation. After error evaluation, the pair with the smallest collapse error is collapsed to obtain M^k.

Image-Driven Simplification [59] models the difference between two meshes as the difference in appearance between them. Mesh appearance is modeled using images rendered from a set of viewpoints distributed equally around the mesh, as shown in Fig. 6.1. Therefore, the difference between meshes M_X and M_Y, $E_\text{IDS}(M_\text{X}, M_\text{Y})$, is defined as the

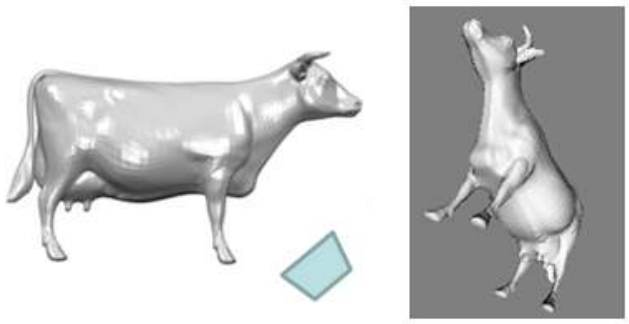


Figure 6.2: (left) Cow mesh; (right) Image of the mesh rendered from a single viewpoint

difference between their image sets, that is,

$$E_{\mathrm{IDS}}(M_{\mathrm{X}}, M_{\mathrm{Y}}) = \|E_I(I_{\mathrm{X},1}, I_{\mathrm{Y},1}), \ldots, E_I(I_{\mathrm{X},k}, I_{\mathrm{Y},k})\|_2 \tag{6.3}$$

$$= \sqrt{\sum_{h}^{k} E_{\mathrm{I}}^2(I_{\mathrm{X},h}, I_{\mathrm{Y},h})} \tag{6.4}$$

$$\text{where} \quad E_{\mathrm{I}}(I_{\mathrm{X}}, I_{\mathrm{Y}}) = \sqrt{\sum_{u}^{m}\sum_{v}^{n}(i_{\mathrm{X},u,v} - i_{\mathrm{Y},u,v})^2} \tag{6.5}$$

$E_{\mathrm{I}}(I_{\mathrm{X}}, I_{\mathrm{Y}})$ is the difference between two images, I_{X} and I_{Y}, and $i_{\mathrm{X},u,v}$ is the intensity of the pixel (u, v) of image I_{X}. In the Image-Driven Simplification paper, k is set to 20. For rendering each image, lightning is provided by a single light source that is positioned near the viewpoint.

6.3 Proposed Method

We observe that in IDS, most viewpoints do not capture any distortion for an edge collapse since it is occluded from their view. Since the region distorted by an edge-collapse is small, we propose that a single image is sufficient to capture this difference.

To do so effectively, we adaptively position the viewpoint relative to the distorted region for each edge collapse; in contrast, for IDS, viewpoints are positioned at fixed locations relative to the simplified mesh throughout the simplification process. Adaptively placing the viewpoint also ensures that all distortions are adequately captured. In a fixed-location viewpoint configuration, there could be regions that are not captured by any viewpoint.

In Image-Driven Simplification, the error of a candidate edge collapse is defined as the difference between the original mesh and the mesh after the edge collapse (Eq. 6.2) as viewed from a fixed set of viewpoints. Thus for a region (or image) of each intermediate mesh, there exists the same corresponding region (or image) for the original mesh. This correspondence is determined by the location and orientation of the viewpoint. However, this correspondence is not preserved for an adaptively-placed viewpoint. Without the same correspondence for all intermediate meshes, it does not make sense to compute the distortion relative to $\hat{M}$. Moving the viewpoint changes $E(\hat{M}, M^{k-1})$, which voids any information that we have obtained from $E(\hat{M}, M^k(x, y))$; we do not know whether the error is due to the edge collapse or simply due to a change in viewpoint location.

Hence we adopt the memoryless approach [48], where edge collapse error is computed relative to the mesh in the previous step rather than to the original mesh. The memoryless approach has been shown to perform at least as well as the standard approach for spatial error metrics [48, 50]. Using the memoryless approach, we now define the error of edge collapse at step k relative to M^{k-1}.

$$(x, y)^k = \arg\min_{x,y} E_{\mathrm{SV}}(M^{k-1}, M^k(x, y)) \tag{6.6}$$

This formulation avoids the issue of correspondence, since the error of M^{k-1} is always zero by definition. Experiments show that this approach also works well for our single viewpoint methods. In our Single Viewpoint Image-Driven Simplification method, the

difference between meshes M_{X} and M_{Y} is defined as the difference between the image of M_{X}, I_{X}, and the image of M_{Y}, I_{Y}, that is,

$$E_{\mathrm{SV}}(M_{\mathrm{X}}, M_{\mathrm{Y}}) = E_{\mathrm{I}}(I_{\mathrm{X}}, I_{\mathrm{Y}}) \tag{6.7}$$

$$= \sqrt{\sum_{u}^{m}\sum_{v}^{n}(i_{\mathrm{X},u,v} - i_{\mathrm{Y},u,v})^2} \tag{6.8}$$

where both M_{X} and M_{Y} are captured from the same viewpoint. Fig. 6.2 illustrates the captured image from a single viewpoint. In the next two subsections, we describe two methods to derive the location and orientation of the viewpoint. Since the light source affects the appearance of a mesh as well, we also derive the position of the light source.

6.3.1 Maximum Angle Viewpoint

In this section, we describe a heuristic to position the viewpoint. We suggest that any heuristic should work reasonably well if it meets the following criteria. 1) The viewpoint is fixed relative to the position and orientation of the distorted region. We assume that the position (and orientation) of any vertex, edge and face of the region can serve as a reference point. 2) Spatial distortions (as opposed to texture or color distortions, which are view-independent) are always captured from the viewpoint. In the following analysis, we assume that a single directional light source (light source is defined by a vector v_{light}), orthographic projection, and diffuse lighting are used for rendering.

Consider the local region R that is distorted when collapsing vertex v to a neighbor (Fig. 6.3a). R is the one-ring neighborhood of v, and its normal n_v is the average normal of faces of R. Since the boundary of R does not change after the edge collapse, v serves as the center of R. We thus constrain the viewpoint to point towards v.

In most cases, a viewpoint will capture the distortion of R if R is not occluded from

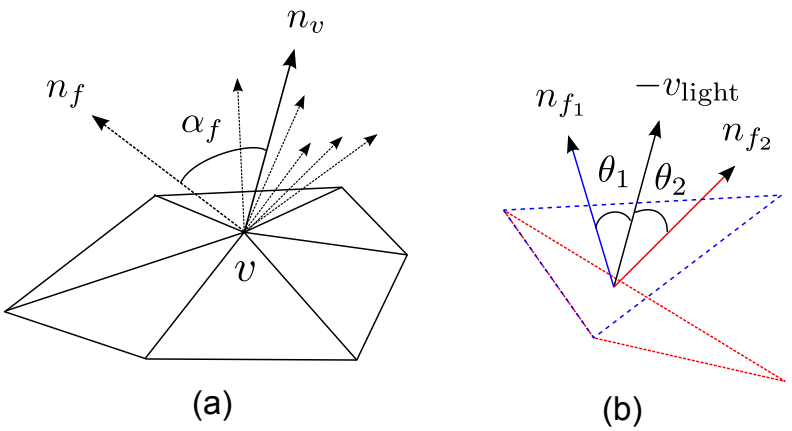


Figure 6.3: (a) The region R that is distorted when collapsing v to a neighboring vertex. Arrows indicate normals of faces in R. n_v is the average normal of faces of R, and n_f is an instance of a face normal with angle α_f from n_v. (b) The orientation of a face before and after an edge collapse and their normals, n_{f_1} and n_{f_2}, respectively, and the light source direction v_{light}. The appearance of the face is unchanged if $\theta_1 = \theta_2$

view. However, there are two exceptions. Exception 1 occurs if the angle between the face normals and light vector is larger than 90 degrees. This means that the faces of R are not illuminated; thus any distortion would not be visible. We avoid this exception by constraining the light vector v_{light} and the viewpoint vector v_{view} to be equal.

Fig. 6.3b shows exception 2 where a face has normals n_{f_1} and n_{f_2} before and after an edge collapse, and is illuminated by v_{light}. Assuming diffuse lighting is used, if $\theta_1 = \theta_2$, the face appearance will be unchanged by the edge collapse. It is likely that the face normals in R will converge towards n_v after an edge collapse. Thus to reduce the occurrence of exception 2, we set v_{light} (and v_{view}) to be the negative of the face normal n_f with the largest angle α_f to n_v (see Fig. 6.3a). Although it might be more intuitive to set v_{view} and v_{light} to $-n_v$ so that all faces of R are always visible, our tests indicate that this generates poor results due to exception 2.

By using orthographic projection for rendering, the viewpoint's distance to R does not affect the rendered image. We set this distance to be an arbitrarily small value.

Similarly, by using a directional light source to light the mesh, the distance between the light source and R does not matter. Thus we set the light source and viewpoint position at a small distance from v in the direction of face normal that has the largest angle to n_v.

To account for silhouette distortions consistently, we render the background and mesh using different alpha values. A difference in alpha value for a pixel between two images indicates a silhouette being covered or uncovered. For such pixels, we set the error to $s \times (\text{max luminance value})^2$. We empirically set the value of s to 0.25.

6.3.2 Maximum Error Viewpoint

In this section, we describe a more structured approach to obtain the placement of the viewpoint. IDS casts mesh difference as the L_2 norm of the difference between their image sets (Eq. 6.4). We recast this difference by applying the L_∞ norm on the set of image differences.

$$E_{\text{IDS-INF}}(M_{\text{X}}, M_{\text{Y}}) = \|E_{\text{I}}(I_{\text{X},1}, I_{\text{Y},1}), \ldots, E_{\text{I}}(I_{\text{X},k}, I_{\text{Y},k})\|_\infty \tag{6.9}$$

$$= \max_h E_{\text{I}}(I_{\text{X},h}, I_{\text{Y},h}) \tag{6.10}$$

$$\text{where} \quad E_{\text{I}}(I_{\text{X}}, I_{\text{Y}}) = \sqrt{\sum_u^m \sum_v^n (i_{\text{X},u,v} - i_{\text{Y},u,v})^2} \tag{6.11}$$

If we let $k \to \infty$, there would be an infinite number of viewpoints to choose from. Instead of choosing the viewpoint with the largest E_{I} (Eq. 6.10), we can, equivalently, compute the viewpoint that yields the largest E_{I}.

To validate our proposition that using $E_{\text{IDS-INF}}$ (Eq. 6.10) with memoryless decimation (Eq. 6.6) is viable, we place k viewpoints around an Armadillo mesh and simplify the mesh to 1000 vertices using the L_∞ metric, i.e., by choosing the largest image error to

k	12	20	62	242
error	0.013	0.013	0.010	0.008

Table 6.1: Hausdorff error of $E_{\text{IDS-INF}}$ with k number of viewpoints for the Armadillo mesh simplified to 1000 vertices.

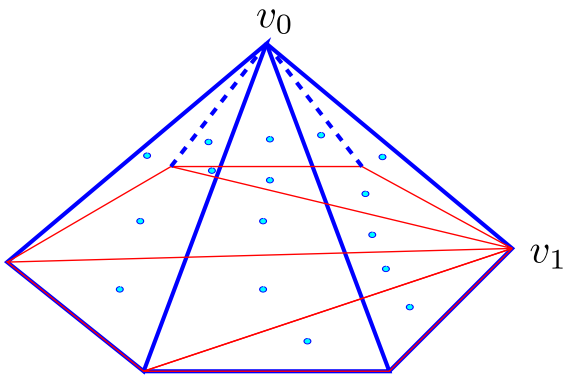


Figure 6.4: The figure shows the overlap of projections of R_{X} and R_{Y} (distorted region before and after v_0 is collapsed to v_1). The overlay of edges partitions the overlapped projections into a set of cells, where each cell is marked with a dot.

be the collapse error. As shown in Table 6.1, increasing the number of viewpoints, k, improves performance. Computing the viewpoint with the largest error is equivalent to using the L_∞ metric at $k \to \infty$.

Now we describe a method to estimate the viewpoint position and lightning position that maximizes $E_{\text{I}}(I_{\text{X}}, I_{\text{Y}})$. In the context of our simplification framework, M_{X} corresponds to M^{k-1} and M_{Y} corresponds to $M^k(x, y)$. Similarly to the Largest Angle Viewpoint method (Section 6.3.1), we orient the viewpoint towards v_n (Fig.6.3) and use orthographic projection for rendering so the viewpoint's distance to the distorted region is irrelevant; only the view vector is required. We also use a single infinite light source and set its vector to be the same as the view vector, so we are left with finding a single unit vector, v_{view}. Moreover, we use diffuse lighting and flat shading so the color of all pixels within each rendered face is the same.

Let R_{X} and R_{Y} denote the distorted region before and after collapsing, respectively. v_0 to v_1 in Fig. 6.4. R_{X} consists of the set of faces connected to v_0, whereas R_{Y} consists of the faces connected to v_1. Notice that R_{X} and R_{Y} share a common boundary around the polyhedron-shaped base. When viewed from an arbitrary viewpoint, the overlay of edges of the projected R_{X} and R_{Y} partitions the projections into a set of cells. The difference between R_{X} and R_{Y}, $E_{\mathrm{I}}(I_{R_{\mathrm{X}}}, I_{R_{\mathrm{Y}}})$, is the aggregated differences of the rendered faces of R_{X} and R_{Y} for all the cells.

$$E_{\mathrm{I}}(I_{R_{\mathrm{X}}}, I_{R_{\mathrm{Y}}}) \tag{6.12}$$

$$= \sqrt{\sum((i_{\mathrm{X},c} - i_{\mathrm{Y},c})^2 \times PA_c)} \tag{6.13}$$

$$= \sum_{f_{\mathrm{X}}} \sum_{f_{\mathrm{Y}}} [(\max(v_p \cdot n_{f_{\mathrm{X}}}, 0) - \max(v_p \cdot n_{f_{\mathrm{Y}}}, 0))^2 \times (PA_{f_{\mathrm{X}}}(v_p) \cap PA_{f_{\mathrm{Y}}}(v_p))] + B(v_p) \tag{6.14}$$

where $i_{\mathrm{X},c}$ denotes the color intensity of R_{X} corresponding to cell c in Fig. 6.5, and PA_c denotes the projected area of c. Factoring out and ignoring material reflectivity constants, we can expand Eq. 6.13 to get Eq. 6.14, where f_{X} and f_{Y} are the faces of R_{X}, and R_{Y} and B is the error attributed to any background uncovered by the edge collapse, or more specifically, the area corresponding to $(PA_{f_{\mathrm{X}}} \cup PA_{R_{\mathrm{Y}}}) - (PA_{f_{\mathrm{X}}} \cap PA_{R_{\mathrm{Y}}})$.

Instead of maximizing Eq. 6.14, we simplify the task by making a few approximations. Instead of optimizing E_I for all the cells, we only consider the largest face of R_{X}, which we denote as f_{M}. By assuming that R_{Y} is flat, and that the shape and area of f_{M} are similar to the shape and area of the underlying R_{Y}, we can constrain v_p to lie on the plane spanned by $n_{f_{\mathrm{M}}}$ and $n_{R_{\mathrm{Y}}}$; hence the problem is reduced to a one-dimensional one

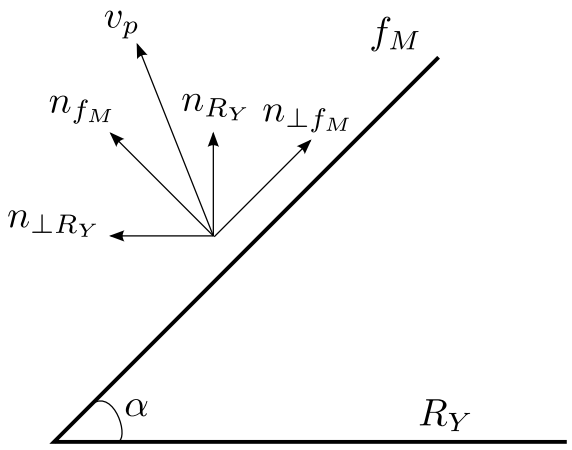


Figure 6.5: The simplified problem for computing v_p where v_p is constrained to lie on the plane spanned by n_{f_M} and n_{R_Y}

(Fig. 6.5). The square root in Eq. 6.13 is irrelevant, so we drop it. Hence we have

$$\arg\max_{v_p} E_\mathrm{I}(I_{R_\mathrm{X}}, I_{R_\mathrm{Y}})$$

$$\approx \arg\max_{v_p}(\max(v_p \cdot n_{f_\mathrm{M}}, 0) - \max(v_p \cdot n_{R_\mathrm{Y}}), 0)^2 \times (PA_{f_\mathrm{M}}(v_p) \cap PA_{R_\mathrm{Y}}(v_p)) \tag{6.15}$$

$$\approx \arg\max_{v_p}(\max(v_p \cdot n_{f_\mathrm{M}}, 0) - \max(v_p \cdot n_{R_\mathrm{Y}}, 0)) \times \min(\max(v_p \cdot n_{f_\mathrm{M}}, 0), \max(v_p \cdot n_{R_\mathrm{Y}}, 0)) \tag{6.16}$$

Note that the projected area of a face f is equals to $(v_p \cdot n_f) \times \mathrm{Area}_f$. By dropping the square terms in Eq. 6.15, we get Eq. 6.16, which has an analytical solution. The solutions are

$$v_p = \widehat{(n_{f_\mathrm{M}} + n_{R_\mathrm{Y}})} + n_{\perp R_\mathrm{Y}} \tag{6.17}$$

$$v_p = \widehat{(n_{f_\mathrm{M}} + n_{R_\mathrm{Y}})} + n_{\perp f_\mathrm{M}} \tag{6.18}$$

where $\widehat{\ }$ is a normalization operator, and $n_\perp$ is normal to n. We verify the solutions by

simulating $E_{\mathrm{I}}(I_{R_{\mathrm{X}}}, I_{R_{\mathrm{Y}}})$ (Eq. 6.16) for different values of v_p and α, and then confirming that two equal maximum values for $E_{\mathrm{I}}(I_{R_{\mathrm{X}}}, I_{R_{\mathrm{Y}}})$ are given by Eq. 6.17 and Eq. 6.18. We use Eq. 6.17 to set the viewpoint since it generally provides better simplification results.

If R_{X} is concave, it is possible for f_{M} to be occluded by other faces of R_{X} when rendered from view vector v_p. Thus for strictly concave regions, we reverse the sign of v_p and render the backside of R. To check for concavity, we find the vector of each incoming neighboring edge of v_0. If all incoming edges are in the opposite direction with $n_{R_{\mathrm{Y}}}$, the region is concave. In our implementation, $n_{R_{\mathrm{Y}}}$ is estimated by taking the average normals of f_{X}.

6.3.3 Speed Optimizations

In IDS, the mesh is selectively rendered to accelerate the algorithm; for each viewpoint, only faces that intersect the 2D bounding box of the distorted region in the captured image are rendered. These faces are identified efficiently by indexing mesh faces for each image pixel and using vertex projection onto the image plane to estimate the bounding box of the distorted area. This index structure has to be updated after each edge collapse operation throughout simplification. Instead of using this acceleration structure, we take a faster but less accurate approach of rendering only the faces in the distorted patch R for our Single Viewpoint methods. We also accelerate the image read-back operation (from GPU to CPU) by using pixel buffer objects and copying only the pixels of the bounding box of the distorted patch from the frame buffer.

6.4 Results and Discussion

Here we compare the simplification performance of Image-Driven Simplification (IDS), Maximum Angle Viewpoint Image-Driven Simplification (MA-IDS) (Section 6.3.1), Max-

imum Error Viewpoint Image-Driven Simplification (ME-IDS) (Section 6.3.2), and the Quadric Error Metric simplification method [47]. The IDS algorithm and both proposed single viewpoint algorithms are implemented based on the OpenMesh [74] decimation framework. We use the publicly available Qslim [47] for the implementation of the Quadric Error Metric method. For all image-driven methods, for non-textured meshes, we render using flat shading and diffuse lighting. For textured meshes, we render using smooth shading and diffuse lighting. All images are rendered at a resolution of 256x256. Our test platform is a Core 2 Quad 2.66 GHz CPU with a Geforce 8600GT GPU.

The meshes in this test are obtained courtesy of Stanford Graphics Group, AIM@SHAPE Shape Repository [75], Hugues Hoppe and Cyberware. These meshes are not textured; thus in order to evaluate performance on textured meshes, we apply texture on the meshes by using CGAL's [76] public implementation of the least squares conformal mapping [77] texture mapping method. We use a gray and white checkerboard pattern as texture.

We evaluate the simplification algorithms using two metrics. The first is the Hausdorff distance, which is the largest of all distances from a point in a mesh to the closest point in another mesh. It can be interpreted as the maximum distance between corresponding points of two meshes. This metric is computed using the program Metro [78], and results are in units of percentage of the bounding box size of the mesh. The second metric, which is also used in the IDS paper [59], evaluates the image RMS error. This error measures perceived error from multiple viewpoints and intrinsically accounts all kinds of perceivable error, such as texture distortion. The evaluated mesh and reference mesh are rendered from 60 evenly distributed viewpoints of 512x512 resolution. The error is the Root Mean Square difference (RMS error) between the images of the evaluated mesh and reference mesh.

Model	Vertices		Time				
	Original	Final	(A)	(B)	(C)	(D)	(E)
Armadillo	20000	1000	20987	824	829	88	87
Bunny	20000	500	18176	707	664	86	90
Cow	11610	254	8511	269	271	52	51
Dinosaur	20000	800	18249	712	758	82	87
Horse	19851	311	15811	565	570	84	88
Rocker-arm	10044	300	5793	168	167	46	45

Table 6.2: Processing times (seconds) of image-driven methods (A) IDS, (B) MA-IDS-S, (C) ME-IDS-S, (D) MA-IDS, and (E) ME-IDS

6.4.1 Processing time comparison

Here we show the main benefit of the single viewpoint methods, which is faster processing time. Our implementation of IDS is not accelerated by selective face rendering (see Section 6.3.3) Hence for a fairer comparison, we also implemented non-accelerated versions of the single viewpoint methods, which we denote as MA-IDS-S and ME-IDS-S. The non-accelerated single viewpoint methods render all faces of the mesh instead of just the faces of the distorted region.

Table 6.2 shows a comparison of the processing times of the image-driven methods. Both single viewpoint methods have comparable speed, regardless of whether they are accelerated or not. The non-accelerated single viewpoint methods are around 25 times faster than IDS, which falls in line with expectations; we expect simplification speed to improve 20 times from using 1 viewpoint instead of 20 viewpoints. We can also deduce that the viewpoint computation methods are computationally fast enough so that the total computation speed is not significantly affected. Both MA-IDS and ME-IDS should show an even larger speedup compared with an accelerated IDS, since the acceleration structure of IDS has to be updated for each edge collapse, and so imposes a speed cost.

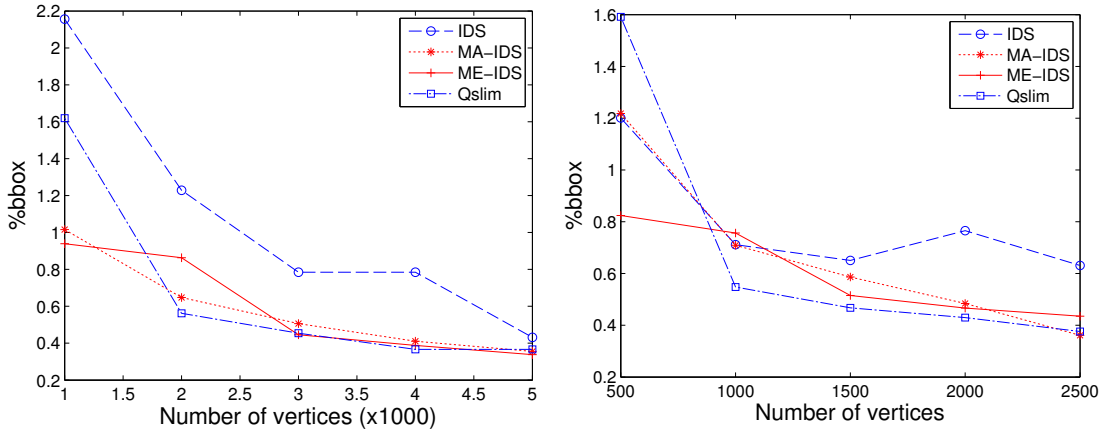
Model	Hausdorff distance				Image RMS error			
	(A)	(B)	(C)	(D)	(A)	(B)	(C)	(D)
Armadillo	2.16	1.02	**0.94**	1.62	17.9	18.0	**17.6**	17.8
Bunny	1.20	1.22	**0.82**	1.59	14.6	14.4	**13.8**	14.4
Cow	2.11	**1.61**	1.62	2.02	14.2	**14.0**	14.1	14.4
Dinosaur	1.12	**0.81**	1.11	1.10	12.4	12.6	12.6	**12.3**
Horse	2.62	1.23	**0.98**	1.47	12.0	12.1	**12.0**	12.3
Rocker-arm	1.54	**1.15**	1.73	2.33	12.7	**12.0**	12.6	12.9

Table 6.3: Error comparisons between simplification results of (A) IDS, (B) MA-IDS, (C) ME-IDS, and (D) Qslim.

6.4.2 Quality comparison

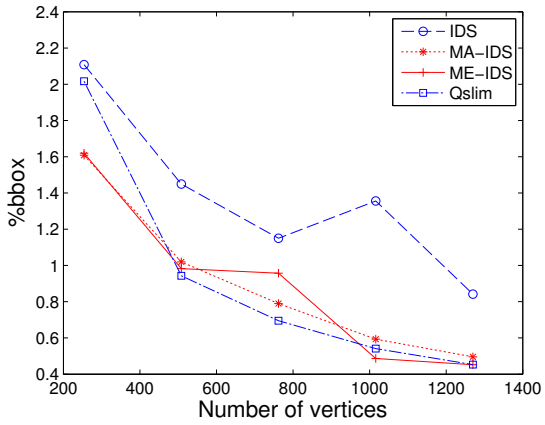
In the following results, we will refer to untextured meshes as plain meshes. For the first quality comparison, we simplify a number of meshes down to a perceptually minimally acceptable resolution. The meshes and their final number of vertices are shown in Table 6.2. The qualitative results of the simplified plain meshes are shown in Table 6.3. In terms of Hausdorff distance, MA-IDS has the lowest error for Cow, Dinosaur, and Rocker-arm, whereas ME-IDS has the lowest error for Armadillo, Bunny, and Horse. Both MA-IDS and ME-IDS perform better than IDS and Qslim in almost all cases. In terms of RMS error, the results are much closer. Still, ME-IDS performs best overall with the lowest error for Armadillo, Bunny, and Horse.

We also compute comparisons at five stages of simplification: the lowest resolution, twice the lowest resolution, and so on up to five times the lowest resolution. Results for Hausdorff distance and RMS image error are shown in Fig. 6.6 and Fig. 6.7 for plain meshes. Over various resolutions, for Hausdorff distance, both MA-IDS and ME-IDS perform similarly to Qslim and outperform IDS. For image RMS error, overall, ME-

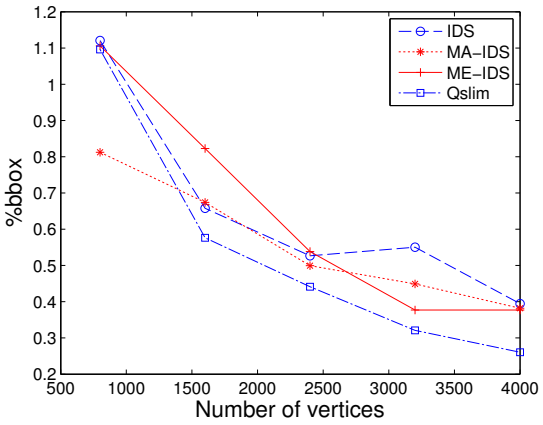


(a) Armadillo

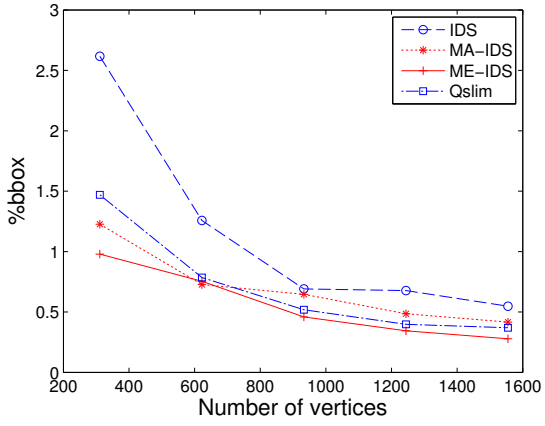
(b) Bunny



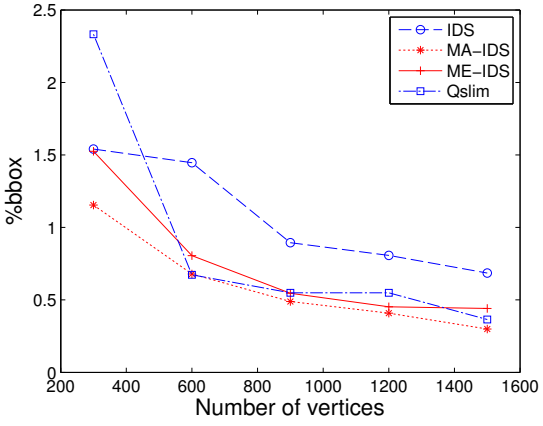
(c) Cow



(d) Dinosaur

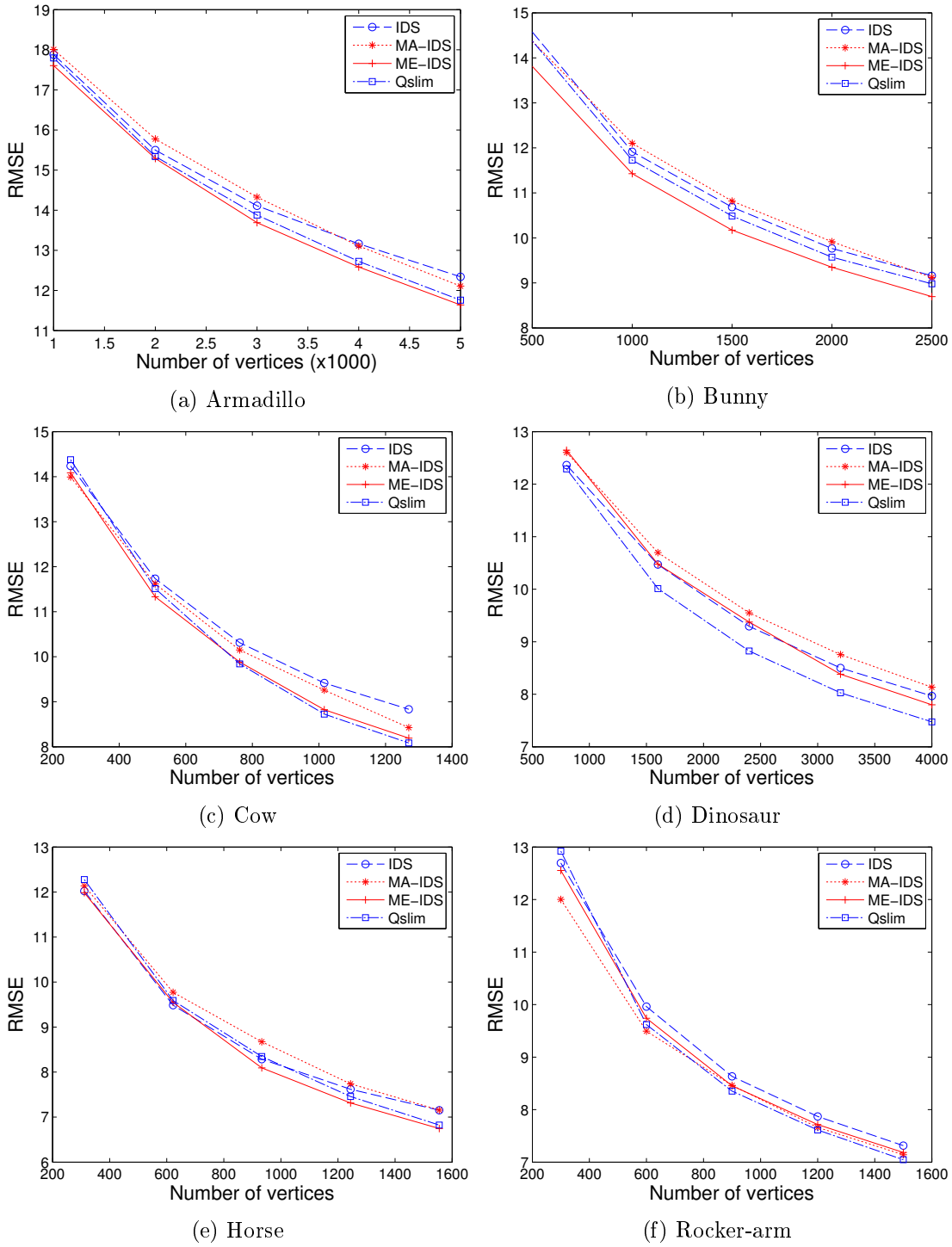


(e) Horse



(f) Rocker-arm

Figure 6.6: Hausdorff distance of various plain meshes at five stages of simplification



(a) Armadillo

(b) Bunny

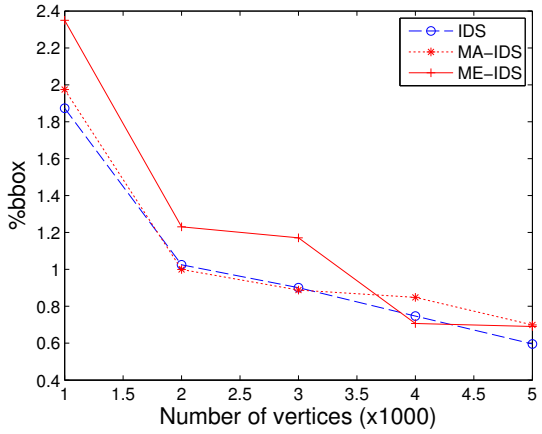
(c) Cow

(d) Dinosaur

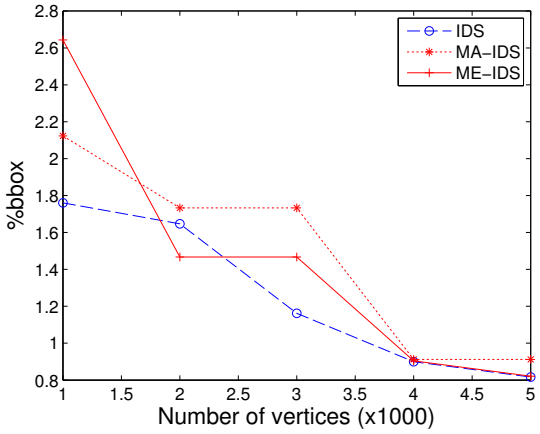
(e) Horse

(f) Rocker-arm

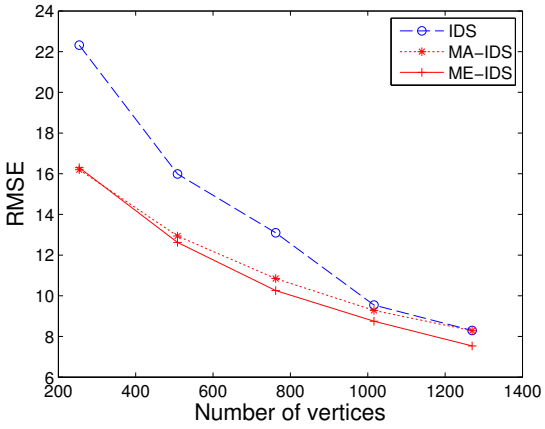
Figure 6.7: Image RMS error of various plain meshes at five stages of simplification



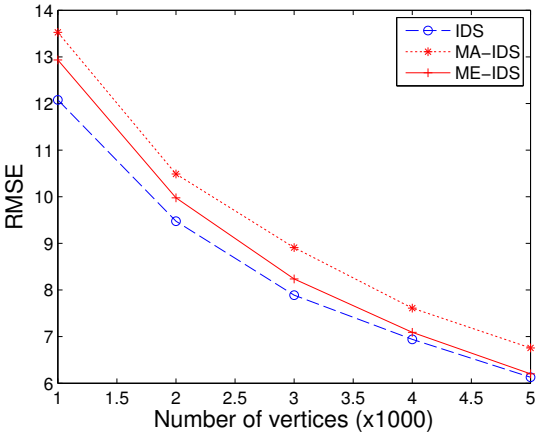
(a) Armadillo



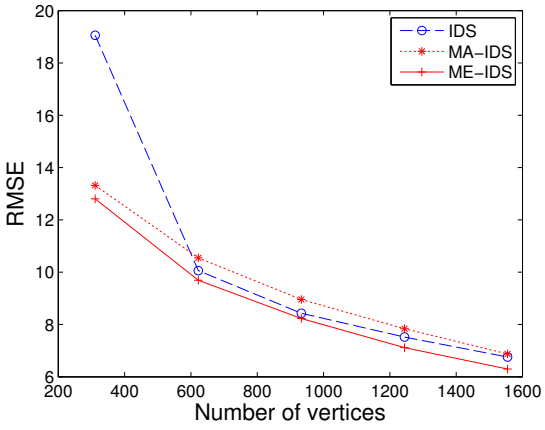
(b) Bunny



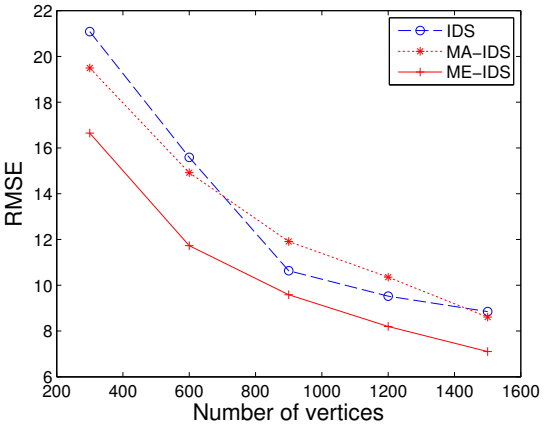
(c) Cow



(d) Dinosaur

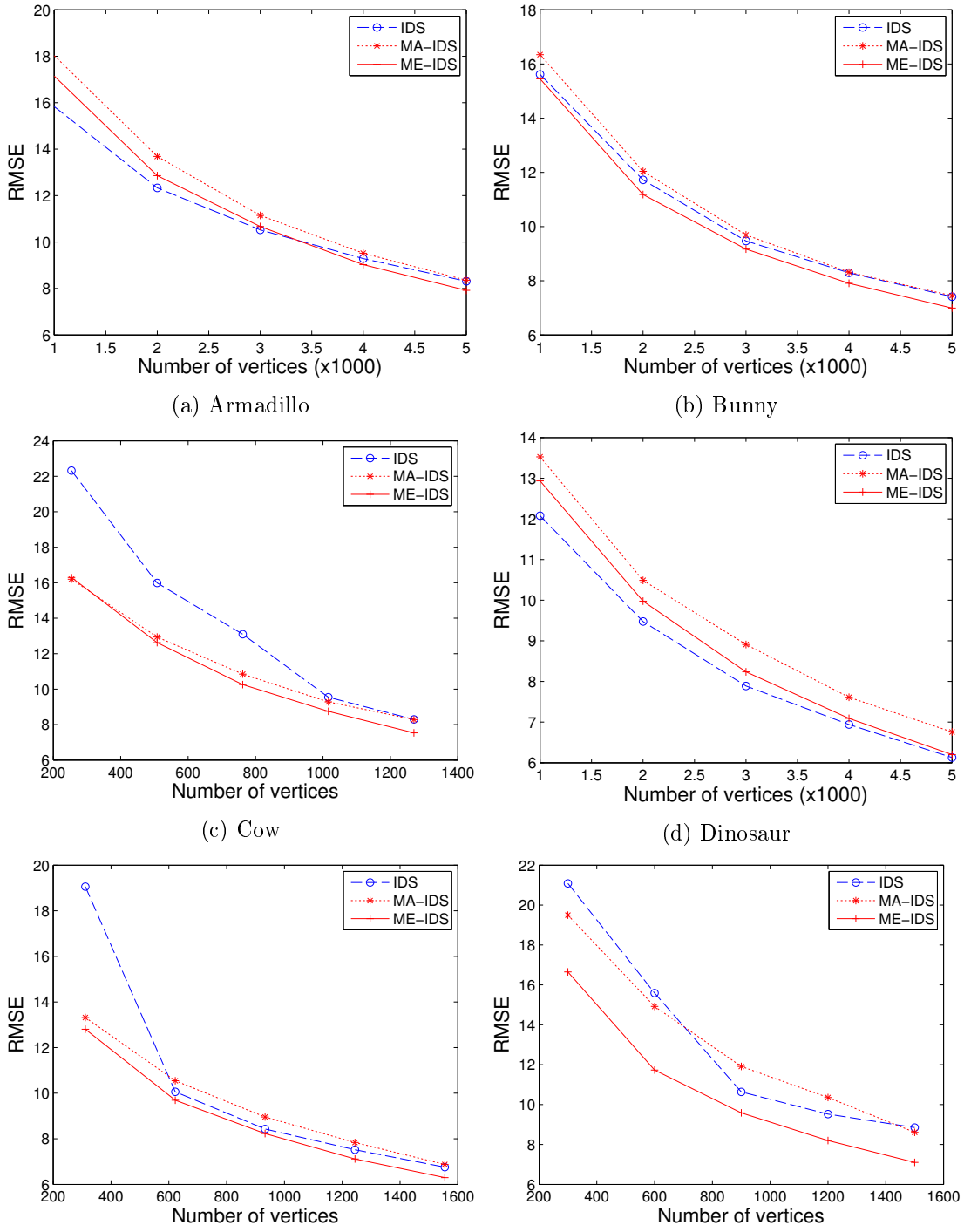


(e) Horse



(f) Rocker-arm

Figure 6.8: Hausdorff distance of various textured meshes at five stages of simplification



(a) Armadillo

(b) Bunny

(c) Cow

(d) Dinosaur

(e) Horse

(f) Rocker-arm

Figure 6.9: Image RMS error of various textured meshes at five stages of simplification

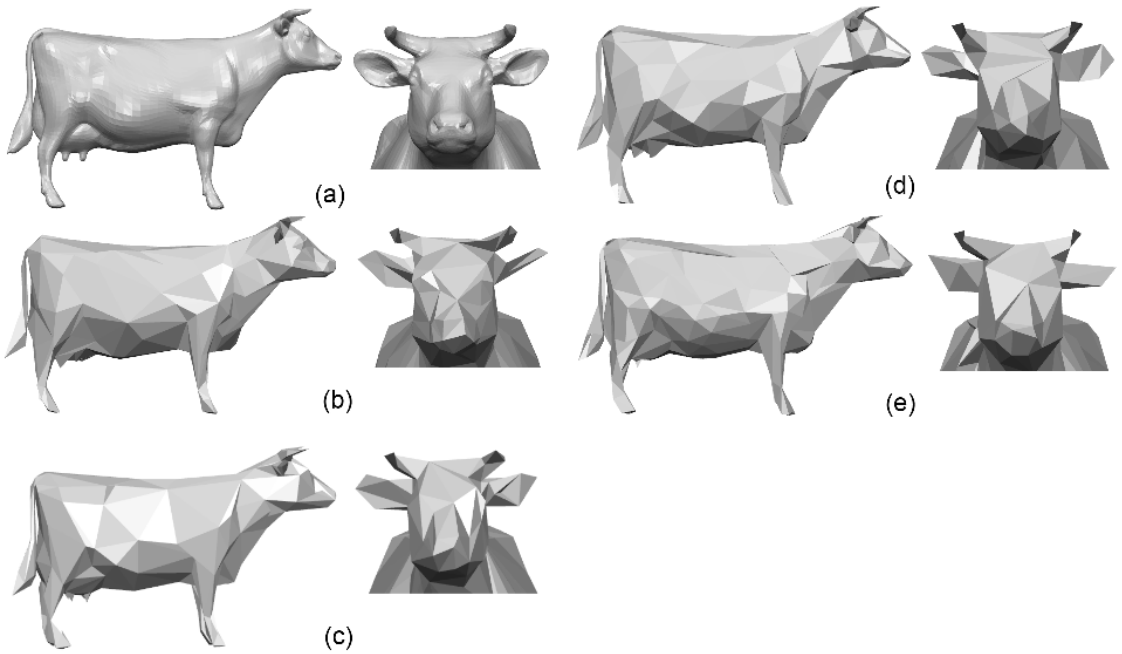


Figure 6.10: Comparison of original and simplified plain cow models (254 vertices). (a) original model (b) IDS (c) MA-IDS (d) ME-IDS (e) Qslim

IDS has the lowest error for Armadillo, Bunny and Horse, whereas Qslim has the best performance for Dinosaur.

Next, we show simplification results on textured meshes in Fig. 6.9 at five resolutions. Since Qslim does not for textures during simplification, we do not evaluate Qslim for this case. We can see that ME-IDS outperforms MA-IDS in most of the cases, and is competitive with IDS.

From a visual comparison of the simplified plain meshes we find that both MA-IDS and ME-IDS are generally the best at preserving mesh fidelity and do well in preserving mesh features such as eyes and ears. IDS on the other hand generally preserves smoothly varying regions well although sharp details could be lost. Fig. 6.10 shows a comparison of the simplified Cow models. Observe that the nipples and ears of the cow are better

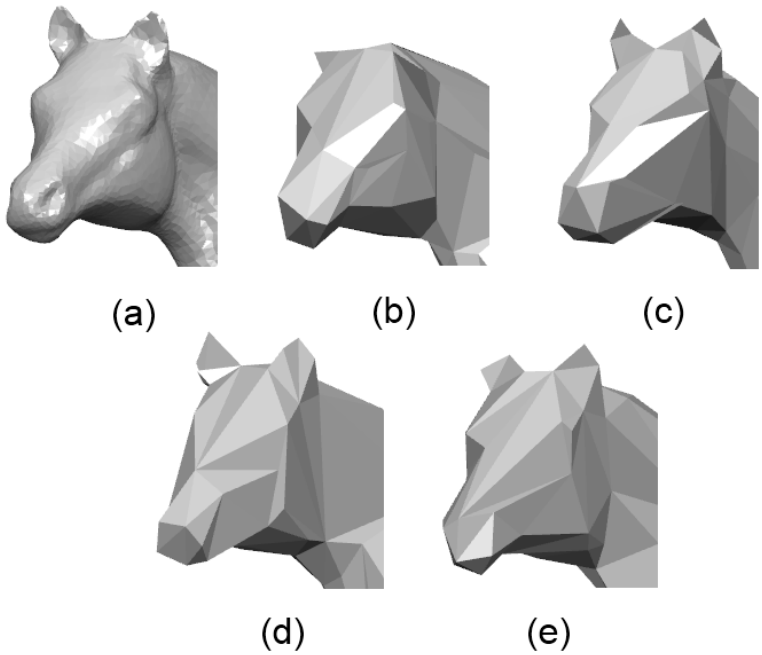


Figure 6.11: Comparison of original and simplified plain horse models (311 vertices). (a) original model (b) IDS (c) MA-IDS (d) ME-IDS (e) Qslim

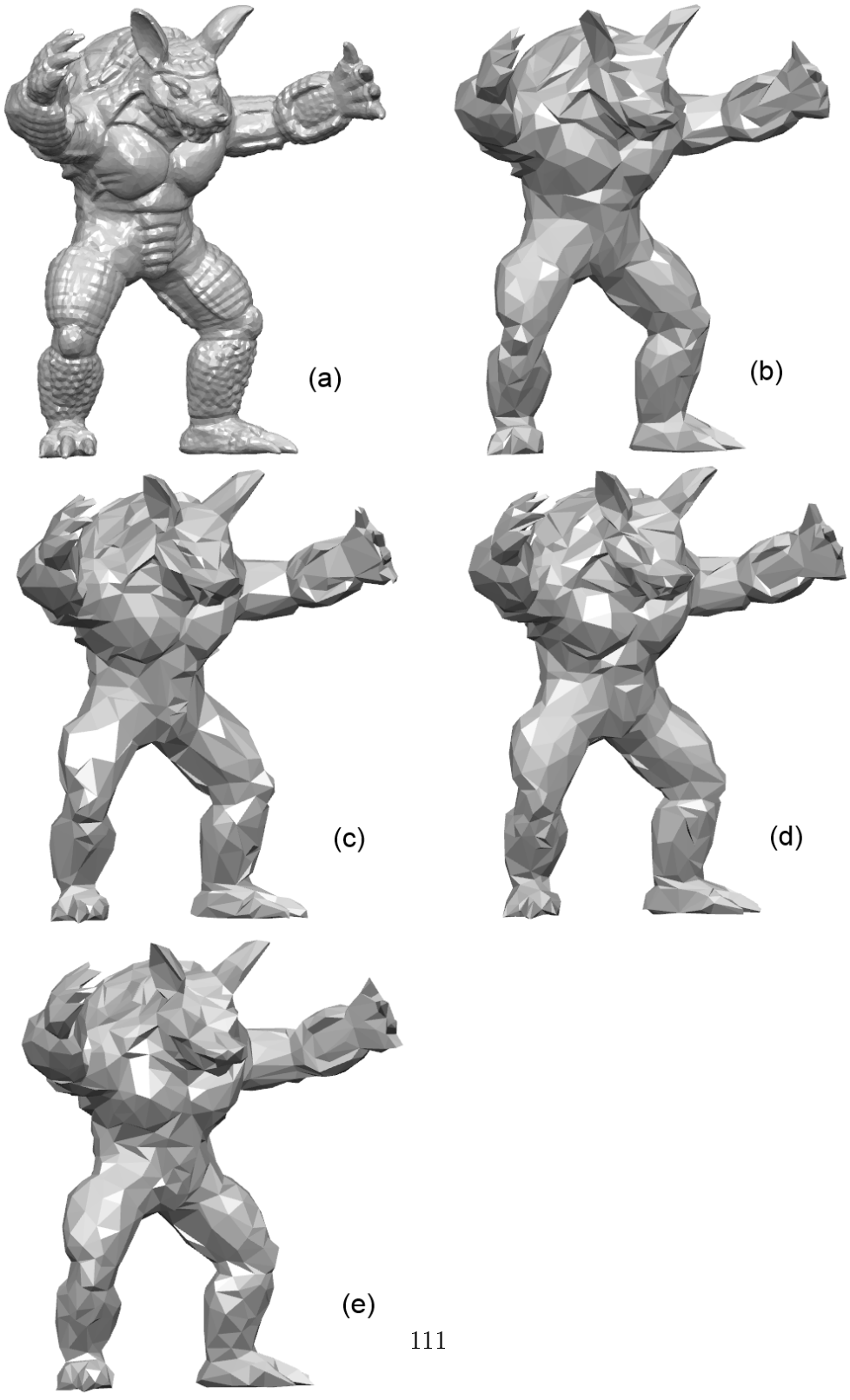


Figure 6.12: Comparison of original and simplified plain armadillo models (1000 vertices). (a) original model (b) IDS (c) MA-IDS (d) ME-IDS (e) Qslim

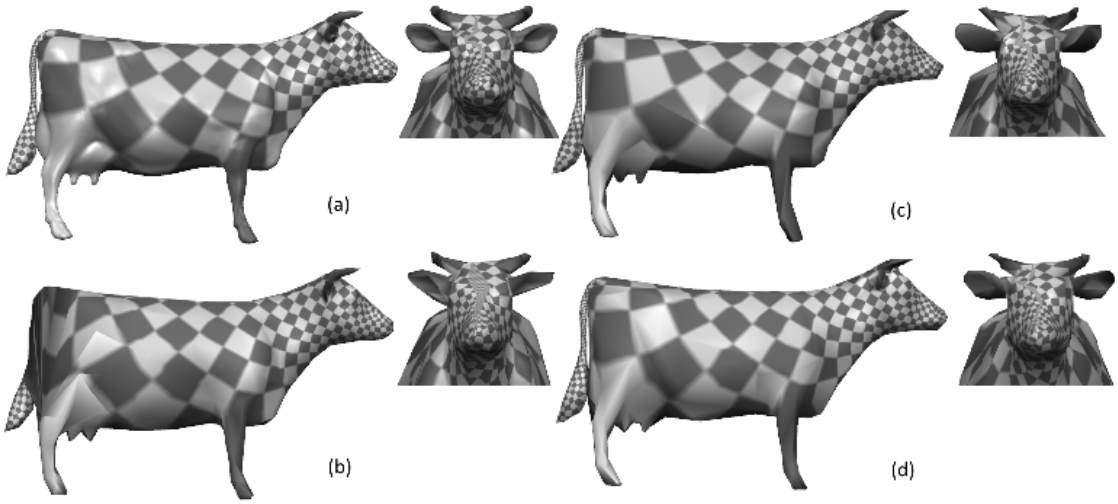


Figure 6.13: Comparison of original and simplified textured cow models (508 vertices). (a) original model (b) IDS (c) MA-IDS (d) ME-IDS

preserved for both MA-IDS and ME-IDS. Similarly for the Horse model in Fig. 6.11, the important features, i.e, the ears, are better preserved by MA-IDS and ME-IDS. In Fig. 6.12, the eyes are only reasonably well-preserved by MA-IDS and ME-IDS; IDS and Qslim blur out the eyes. Both ME-IDS and Qslim preserve the impression on the left arm better than the other methods. The torso is smoother for IDS compared with other algorithms.

A visual comparison of textured meshes show that subjective evaluation agrees with objective results. For the armadillo model (Fig. 6.14), IDS preserves the left armadillo hand slightly better than the single viewpoint methods. On the other hand, for the cow model (Fig. 6.13), IDS skews the ears of the cow and the tail is partly merged with the body, while the single viewpoint methods perserve the ears and tail well. It is notable that the texture patterns are relatively well preserved at the expense of geometric features. This is could be due to stronger appearance distortion when textured patterns are distorted, hence these areas are less likely to be simplified.

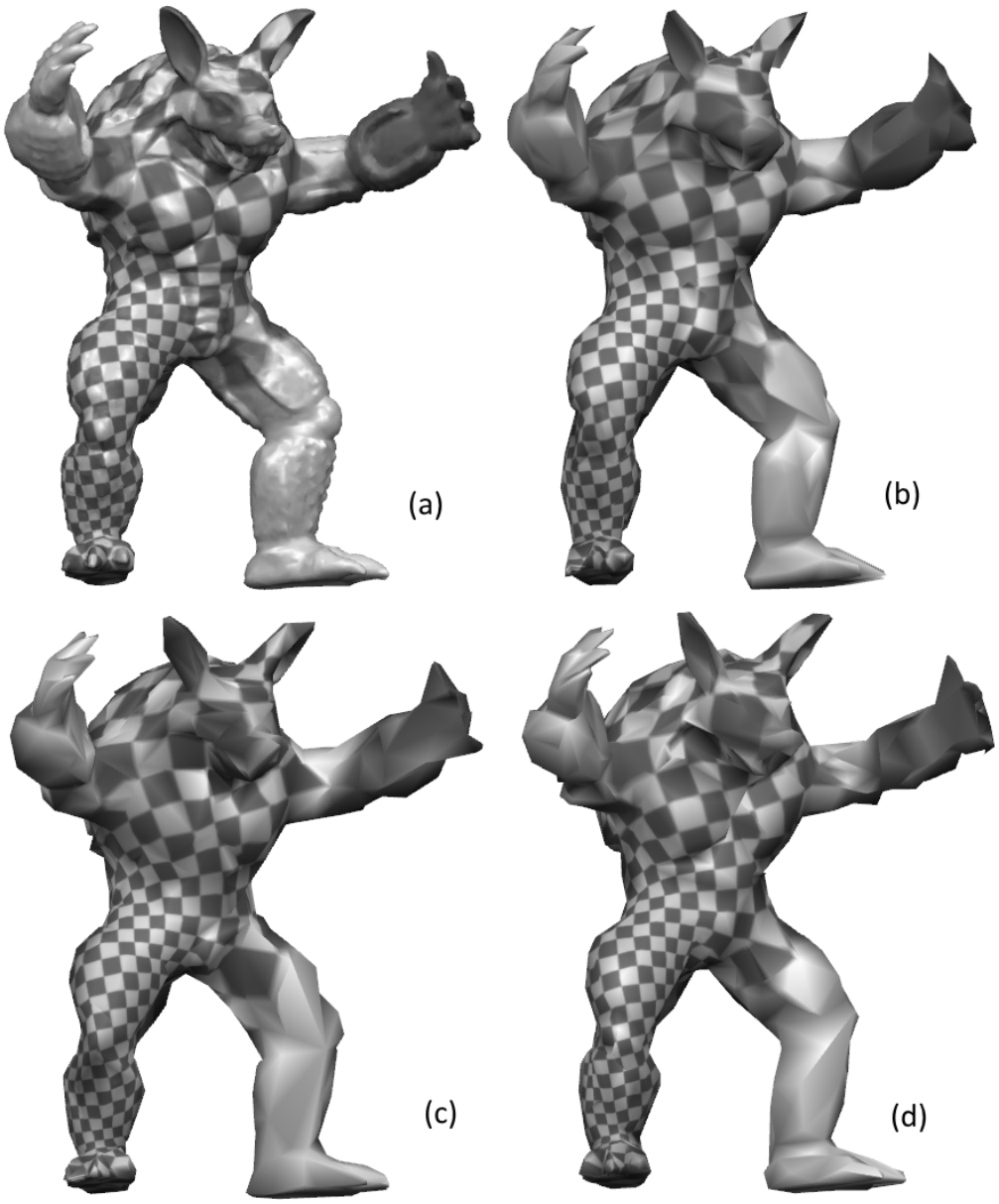


Figure 6.14: Comparison of original and simplified textured armadillo models (1000 vertices). (a) original model (b) IDS (c) MA-IDS (d) ME-IDS

Comparing models simplified with and without texture, quantitative results show that, for all algorithms, Hausdorff errors are higher on textured meshes compared to plain meshes. This is because textures add extra details to flat and featureless areas of the mesh. In order to preserve these details, extra vertices have to be allocated to those areas, whereas for untextured meshes, these vertices can be allocated to better preserve geometric features.

6.5 Discussion

Although our formulation for the viewpoint selection equation for ME-IDS does not factor in texture explicitly, theoretically the presence of texture has no effect on the precision of our viewpoint estimation equation. The reason is that the color contribution from texture is view-independent; the color due to the texture is not affected by the viewpoint and light position. Thus, the degration of performance of the single image methods relative to twenty viewpoint IDS is likely due to the reliance of single viewpoint IDS on a background error value for silhouette distortion detection. For MA-IDS and ME-IDS, if a silhouette pixel becomes a background pixel, it is assigned a constant background error value (instead of the difference in pixel values). This value weighs the tradeoff between preserving likely silhouttes, which are most likely to protruding features, and flatter regions of the mesh. Since texture modulates the color, and thus the amount of appearance distortion, of the flatter regions, the background error value used might be less optimal for textured meshes.

Although our viewpoint determination methods could appear to be too grossly approximated, it is very fast to compute; it does not involve any expensive trigonometric functions which a more accurate solution might require. As shown in the results, our algorithms using one approximated viewpoint are good enough to at least match the 20

viewpoint IDS method, while being much faster.

6.6 Conclusion

As opposed to geometry-based algorithms like Qslim, Image-Driven Simplification (IDS) intrinsically factors in the appearance properties of a mesh like normals and texture during simplification. In this chapter, we propose to accelerate Image-Driven Simplification by using only a single viewpoint instead of 20.

We propose two methods to determine the position and orientation of the viewpoint. For the first method, used in Maximum Angle Viewpoint Image-Driven Simplification (MA-IDS), we suggest a few guidelines for viewpoint placement and propose a heuristic that conforms to these guidelines.

For the second method, we reformulate the Image-Driven Simplification error metric by casting it from a L_2 norm to a L_∞ norm. We then propose that the single viewpoint should be placed to maximize this L_∞ error. By making a few approximations, we then obtained the Maximum Error Viewpoint placement method to approximate this viewpoint for Maximum Error Viewpoint Image-Driven Simplification (ME-IDS)

Our Single Viewpoint Image-Driven methods are around two hundred times faster than a non-accelerated version of IDS and an estimated twenty-five times faster than the accelerated version. For non-textured meshes, objective measures and visual inspection show that both Single Viewpoint Image-Driven methods generally have lower distortion than IDS, which uses 20 viewpoints. For textured meshes, both ME-IDS is performs competitively with IDS, whereas MA-IDS performs worse. Visual inspection shows that textures of simplified meshes of the single viewpoint methods are well preserved.

Chapter 7

Conclusions and Future Work

In this chapter, we summarize the main developments and results in Section 7.1, and present some directions for future research in Section 7.2.

7.1 Conclusions

3D content, such as motion capture data and 3D meshes, are widely used in a number of sectors, such as entertainment and sports. With the recent availability of 3D content acquisition hardware for consumers, such as the Microsoft Kinect and Makerbot Digitizer mesh scanner, the acquisition and use of 3D content will become even more widespread. However, due to issues related to the acquisition of 3D content, postprocessing is often required on captured 3D content before it can be used in applications. For human motion capture (mocap) data acquired using optical systems, entries of the data might be missing due to occluded body parts or markers. Thus, mocap data recovery algorithms are needed to restore any missing data. For scanned 3D meshes, the complexity, or resolution, of the mesh could be much higher than necessary for the intended application; this will consume a large amount of computational resources. Therefore, mesh simplification algorithms are

required to reduce mesh resolution.

For mocap recovery, we focus on using low rank matrix completion, along with the Singular Value Thresholding (SVT) method, to recover mocap data. Matrix completion poses the problem for recovering mocap data in a matrix form, while SVT is an optimization method that solves the problem. We present three strategies to extend and improve the performance of this mocap recovery framework, namely using a trajectory-based matrix representation, applying skeleton constraints, and using subspace constraints. Although we use SVT to solve the matrix completion problem, it is equally valid to use other optimization methods instead. However, if an alternative optimization method is used, the extensions applied to SVT in our improved methods have to be similarly applied to this alternative method.

Our mocap recovery methods target two types of missing data: random missing data, where each joint in a sequence are missing at random, and block missing data, where each joint is missing for long intervals of time. For random missing data, we simulate missing rates of up to 60%, whereas for block missing data, we simulate missing joint intervals of up to 2 seconds at low missing rates.

For the case of random missing data, we propose in Chapter 3 to arrange the mocap matrix into columns of short trajectories for matrix completion recovery, which we call the trajectory-based representation. We show that the proposed trajectory-based representation has a lower rank than the previous frame-based representation. Since matrix completion recovers data better on matrices with lower rank, this fact allows the SVT matrix completion method, by using the proposed representation, to recover missing mocap data at a much lower error. The proposed method, trajectory-based SVT (TSVT) shows up to 80% improvement and 68% improvement on average compared to the previous frame-based SVT (FSVT) method.

FSVT and TSVT methods exploit different properties of mocap data; TSVT exploits the correlation between short trajectory segments, whereas the FSVT exploits the correlation between frames of the sequence. In order to exploit both properties for better mocap recovery performance, in Chapter 5, we propose to extract the correlation between frames property from FSVT, and then apply it in the form of subspace constraints during TSVT optimization. In this proposed Combined Frame and Trajectory based SVT (CFTSVT) method, FSVT recovery is first applied, followed by TSVT. The TSVT method is modified so that mocap recovery is constrained by a subspace modeled by the output of FSVT. Experiments showed that CFTSVT outperforms FSVT by an average of 76% and TSVT by an average of 23%.

For the case of block missing data, performances of both FSVT and TSVT decrease drastically when joints are missing for many consecutive frames, even when the rate of missing data is unchanged. Hence, in Chapter 4, we explain that the distances between mocap joints are constant due to skeleton rigidity. However, this distance is not preserved in the recovered mocap data when the missing data interval increases. Thus, we extend the FSVT method to include skeleton constraints that constrain the distances between connected joints. As explained in Chapter 3, TSVT fares poorly against such patterns of missing data, and thus methods derived from TSVT are not suitable for this case. Experiments show that the proposed skeleton-constrained SVT (SCSVT) method outperforms FSVT by 40% on average. Compared to the recent state of the art algorithm, BOLERO, our method is 3 to 11 times faster, and offers competitive performance.

Chapter 6 focuses on a method of 3D mesh simplification, the Image-Driven Simplification (IDS) algorithm. We find that the IDS algorithm has a long processing time because of repeated render and image readback cycles from each of its 20 viewpoints. To reduce the processing time of IDS, we propose that with proper placement of the

viewpoint, the single viewpoint version can reduce processing time significantly while not compromising on simplification quality. Two single viewpoint algorithms with different viewpoint placement methods, MA-IDS and ME-IDS, are proposed. Timing tests shows that both single viewpoint algorithms are an estimated 25 times faster than IDS. Objective and subjective evaluations show that both single viewpoint algorithms generally have lower distortion simplification results than IDS for non-textured meshes. For textured meshes, ME-IDS is competitive with IDS, whereas MA-IDS performs worse. Regardless, the textures are still well preserved for the single viewpoint algorithms.

7.2 Future Work

Based on the studies of the thesis, the following areas can be investigated in the future.

7.2.1 Extensions for Mocap Recovery

In Section 3.3, it was explained that the trajectory-based method works well because the trajectory of joints are correlated. However, the start and end of trajectories for various joints over the sequence might not be synchronized. For example, in a walking motion, the trajectories of the right and left feet follow a similar cyclical pattern, but they are not synchronized; the start of a cycle for the left foot might coincide with the middle of a cycle for the right foot. If the trajectories of the joints are better synchronized, overall correlation of the set of trajectories will be much higher, thus yielding improved performance for trajectory-based matrix completion. A possible approach would be to create an over-represented matrix which has overlapping trajectories. During recovery, an optimization process will then pick out the optimal set of trajectories which generates the best results for mocap recovery.

In this thesis, we work with mocap sequences that are relatively homogeneous; each

mocap sequence contains a limited set of actions. For example, a sequence only encodes a punching action, and not punching, followed by running and flipping. It is possible to extend the framework for more effective recovery of mocap sequences that contain more diverse sets of actions. The sequence of frames in a mocap sequence forms a high-dimensional continuous curve in frame space, where each point on the curve represents a single frame, or pose [79]. If the mocap sequence contains a small range of actions, this curve lies in a low dimensional subspace, which allows low rank matrix completion to recover mocap data effectively. However, if the motion sequence contains a variety of actions, the set of frames in the sequence lies in a higher dimensional subspace; this subspace is a union of low dimensional subspaces, where each low dimensional subspace is spanned by a set of frames corresponding to an action.

Therefore, it might be more effective to segment a mocap sequence into sets of frames, where each set spans a low dimensional subspace, and recover missing data for each set individually, rather than try to recover all frames at once. A straightforward way to segment a sequence is to partition it sequentially at equal time intervals; this could work since a small continuous set of frames lie in a low dimensional subspace. A better way is to use more advanced mocap segmentation methods [71, 80]. However, these methods also partition a mocap sequence sequentially; they ignore the overlaps or relationships between subspaces of each individual segment. It is possible to obtain even better segmenting for low rank matrix completion by using subspace clustering. Subspace clustering [81, 82] clusters a set of high dimensional data points into sets of points, where the sets are disjoint, and each set span a low-dimensional subspace. A mocap sequence can be segmented as such into sets of frames, and each set recovered individually.

Another avenue for further investigation is in the targeted missing data patterns. In this thesis, we only focus on two types of missing data patterns. It would be beneficial to

investigate whether the proposed methods are beneficial for recovering from other types of missing data patterns, and also investigate methods tailored for those types of missing data patterns. This would allow a wider range of application for mocap data recovery. For example, the methods could be used to improve the resilience of network delivery of mocap data.

7.2.2 Extensions for Mesh Simplication

For Image-Driven Simplification, although reducing the number of viewpoints to one shortens processing time significantly, the algorithm is still quite slow. The bottleneck of Image Driven Simplification is the transfer of rendered images from GPU to CPU for error computation. By moving the computation of candidate edge collapse errors entirely to the GPU, the processing time can be reduced even further. Even faster processing times can be achieved if part of or the entire simplification algorithm is moved to the GPU itself. There has been research in implementing simplification on the GPU. Shontz et al. [83] moved part of the simplification workload to the GPU. DeCoro and Tatarchuk [84] implemented a vertex clustering type simplification algorithm entirely on the GPU. However, it is less straightforward to implement an efficient progressive mesh type simplification method, such as IDS, on a GPU due to its non-parallel nature.

The MA-IDS viewpoint placement is purely heuristical, whereas the ME-IDS viewpoint placement is solved by making a number of approximations since we prioritized speed of computation over accuracy of viewpoint placement. Since the processing time of Single Viewpoint Image-Driven Simplification is still quite long, it might be better to derive a more accurate solution at a higher computation cost for better simplification performance.

Author's Publications

1. C.-H. Tan, J. Hou, and L.-P. Chau, "Human motion capture data recovery using trajectory-based matrix completion," Electronics Letters, vol. 49, no. 12, pp. 752–754, 2013.

2. C.-H. Tan, J. Hou, and L.-P. Chau, "Motion capture data recovery using skeleton constrained singular value thresholding," The Visual Computer, pp. 1–12, 2014.

3. C.-H. Tan and L.-P. Chau, "Single Viewpoint Image-Driven Simplification," International Journal of Image and Graphics, vol. 14, no. 03, 2014.

4. C.-H. Tan and L.-P. Chau, "Image-driven simplification with single viewpoint," in 2012 IEEE International Symposium on Circuits and Systems (ISCAS), 2012, pp. 858–861.

5. C.-H. Tan and L.-P. Chau, "Saliency guided image-driven simplification," in Communications and Signal Processing (ICICS) 2011 8th International Conference on Information, 2011, pp. 1–4.

Bibliography

[1] Z. Bian, J. Hou, L. Chau, and N. Magnenat-Thalmann, "Fall detection based on body part tracking using a depth camera," *IEEE Journal of Biomedical and Health Informatics*, vol. PP, no. 99, pp. 1–1, 2014.

[2] Vicon, "Vicon motion system," *http://www.vicon.com*, 2013.

[3] T. Dutta, "Evaluation of the kinect™ sensor for 3-d kinematic measurement in the workplace," *Applied Ergonomics*, vol. 43, no. 4, pp. 645–649, Jul. 2012.

[4] A. Yao, J. Gall, and L. Van Gool, "Coupled action recognition and pose estimation from multiple views," *International Journal of Computer Vision*, vol. 100, no. 1, pp. 16–37, 2012.

[5] Y. Sheikh, M. Sheikh, and M. Shah, "Exploring the space of a human action," in *Tenth IEEE International Conference on Computer Vision, 2005. ICCV 2005*, vol. 1, 2005, pp. 144–149 Vol. 1.

[6] I. Baran and J. Popović, "Automatic rigging and animation of 3D characters," in *ACM SIGGRAPH 2007 Papers*, ser. SIGGRAPH '07. New York, NY, USA: ACM, 2007.

[7] D. Wiley and J. Hahn, "Interpolation synthesis for articulated figure motion," in *Virtual Reality Annual International Symposium, 1997., IEEE 1997*, Mar. 1997, pp. 156–160.

[8] T. Piazza, J. Lundström, A. Kunz, and M. Fjeld, "Predicting missing markers in real-time optical motion capture," in *Proceedings of the 2009 international conference on Modelling the Physiological Human*. Springer-Verlag, 2009, p. 125–136.

[9] B.-K. Yi, N. Sidiropoulos, T. Johnson, H. Jagadish, C. Faloutsos, and A. Biliris, "Online data mining for co-evolving time sequences," in *16th International Conference on Data Engineering, 2000. Proceedings*, 2000, pp. 13–22.

[10] S. Papadimitriou, A. Brockwell, and C. Faloutsos, "Adaptive, hands-off stream mining," in *Proceedings of the 29th international conference on Very large data bases - Volume 29*, ser. VLDB '03. VLDB Endowment, 2003, pp. 560–571.

[11] G. Liu and L. McMillan, "Estimation of missing markers in human motion capture," *The Visual Computer*, vol. 22, no. 9-11, pp. 721–728, Sep. 2006.

[12] J. Chai and J. K. Hodgins, "Performance animation from low-dimensional control signals," *ACM Trans. Graph.*, vol. 24, no. 3, pp. 686–696, Jul. 2005.

[13] B. Krüger, J. Tautges, A. Weber, and A. Zinke, "Fast local and global similarity searches in large motion capture databases," in *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, ser. SCA '10. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2010, p. 1–10.

[14] J. Baumann, B. Krüger, A. Zinke, and A. Weber, "Data-driven completion of motion capture data," *Proceedings of Workshop on Virtual Reality Interaction and Physical Simulation (VRIPHYS)*, Dec. 2011.

[15] K. Grochow, S. L. Martin, A. Hertzmann, and Z. Popović, "Style-based inverse kinematics," in *ACM SIGGRAPH 2004 Papers*, ser. SIGGRAPH '04. New York, NY, USA: ACM, 2004, p. 522–531.

[16] N. Lawrence, "Gaussian process latent variable models for visualisation of high dimensional data," in *Advances in Neural Information Processing Systems 16*, S. Thrun, L. Saul, and B. {schölkopf}, Eds. MIT Press, 2004.

[17] J. Wang, D. Fleet, and A. Hertzmann, "Gaussian process dynamical models for human motion," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 2, pp. 283–298, Feb. 2008.

[18] H. Lou and J. Chai, "Example-based human motion denoising," *IEEE Transactions on Visualization and Computer Graphics*, vol. 16, no. 5, pp. 870 –879, Oct. 2010.

[19] G. W. Taylor, G. E. Hinton, and S. Roweis, "Modeling human motion using binary latent variables," in *Advances in Neural Information Processing Systems*. MIT Press, 2007, p. 1345–1352.

[20] J. Xiao, Y. Feng, and W. Hu, "Predicting missing markers in human motion capture using $l1$-sparse representation," *Computer Animation and Virtual Worlds*, vol. 22, no. 2-3, p. 221–228, Apr. 2011.

[21] J. Hou, L.-P. Chau, Y. He, J. Chen, and N. Magnenat-Thalmann, "Human motion capture data recovery via trajectory-based sparse representation," in *Proc. IEEE International Conference on Image Processing (ICIP)*, 2013.

[22] L. Li, J. McCann, N. S. Pollard, and C. Faloutsos, "DynaMMo: mining and summarization of coevolving sequences with missing values," in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, ser. KDD '09. New York, NY, USA: ACM, 2009, pp. 507–516.

[23] L. Li, J. McCann, N. Pollard, and C. Faloutsos, "BoLeRO: a principled technique for including bone length constraints in motion capture occlusion filling," in *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, ser. SCA '10. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2010, p. 179–188.

[24] R. Lai, P. Yuen, and K. Lee, "Motion capture data completion and denoising by singular value thresholding," in *Eurographics 2011-Short Papers*, 2011, pp. 45–48.

[25] E. Candès and B. Recht, "Exact matrix completion via convex optimization," *Foundations of Computational mathematics*, vol. 9, no. 6, pp. 717–772, 2009.

[26] J.-F. Cai, E. J. Candès, and Z. Shen, "A singular value thresholding algorithm for matrix completion," *SIAM J. on Optimization*, vol. 20, no. 4, pp. 1956–1982, Mar. 2010.

[27] N. Srebro and T. Jaakkola, "Weighted low-rank approximations," in *In 20th International Conference on Machine Learning*. AAAI Press, 2003, p. 720–727.

[28] L. Herda, P. Fua, R. Plankers, R. Boulic, and D. Thalmann, "Skeleton-based motion capture for robust reconstruction of human motion," in *Computer Animation 2000. Proceedings*, 2000, pp. 77 –83.

[29] L. Herda, P. Fua, R. Plänkers, R. Boulic, and D. Thalmann, "Using skeleton-based tracking to increase the reliability of optical motion capture," *Human Movement Science*, vol. 20, no. 3, pp. 313–341, Jun. 2001.

[30] A. Hornung, S. Sar-Dessai, and L. Kobbelt, "Self-calibrating optical motion tracking for articulated bodies," in *IEEE Virtual Reality, 2005. Proceedings. VR 2005*, 2005, pp. 75–82.

[31] K. Dorfmüller-Ulhaas, "Robust optical user motion tracking using a kalman filter," in *10th ACM Symposium on Virtual Reality Software and Technology*, 2003.

[32] A. Aristidou, J. Cameron, and J. Lasenby, "Real-time estimation of missing markers in human motion capture," in *The 2nd International Conference on Bioinformatics and Biomedical Engineering, 2008. ICBBE 2008*, May 2008, pp. 1343–1346.

[33] A. Aristidou and J. Lasenby, "Real-time marker prediction and CoR estimation in optical motion capture," *The Visual Computer*, vol. 29, no. 1, pp. 7–26, Jan. 2013.

[34] CMU, "CMU graphics lab motion capture database," *http://mocap.cs.cmu.edu/*, 2012.

[35] M. Botsch, M. Pauly, L. Kobbelt, P. Alliez, B. Lévy, S. Bischoff, and C. Rössl, "Geometric modeling based on polygonal meshes," in *ACM SIGGRAPH 2007 Courses*, ser. SIGGRAPH '07. New York, NY, USA: ACM, 2007.

[36] J. Rossignac and P. Borrel, "Multi-resolution 3D approximations for rendering complex scenes," in *Modeling in Computer Graphics: Methods and Applications*. Springer-Verlag, 1993, pp. 455–465.

[37] K.-L. Low and T.-S. Tan, "Model simplification using vertex-clustering," in *Proceedings of the 1997 symposium on Interactive 3D graphics*. Providence, Rhode Island, United States: ACM, 1997, pp. 75–ff.

[38] P. Lindstrom, "Out-of-core simplification of large polygonal models," in *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*. ACM Press/Addison-Wesley Publishing Co. New York, NY, USA, 2000, pp. 259–262.

[39] Y. He, B.-S. Chew, D. Wang, C.-H. Hoi, and L.-P. Chau, "Streaming 3D meshes using spectral geometry images," in *Proceedings of the seventeen ACM international conference on Multimedia*. Beijing, China: ACM, 2009, pp. 431–440.

[40] P. Alliez, E. de Verdiere, O. Devillers, and M. Isenburg, "Isotropic surface remeshing," *SMI 2003: SHAPE MODELING INTERNATIONAL 2003, PROCEEDINGS*, pp. 49–58, 2003.

[41] S. Dong, P.-T. Bremer, M. Garland, V. Pascucci, and J. C. Hart, "Spectral surface quadrangulation," *ACM Trans. Graph.*, vol. 25, no. 3, pp. 1057–1066, 2006.

[42] X. Gu, S. J. Gortler, and H. Hoppe, "Geometry images," *ACM Trans. Graph.*, vol. 21, no. 3, pp. 355–361, 2002.

[43] D. Cohen-Steiner, P. Alliez, and M. Desbrun, "Variational shape approximation," in *ACM SIGGRAPH 2004 Papers*. Los Angeles, California: ACM, 2004, pp. 905–914.

[44] P. Alliez, G. Ucelli, C. Gotsman, and M. Attene, "Recent advances in remeshing of surfaces," in *Shape Analysis and Structuring*, ser. Mathematics and Visualization, L. D. Floriani and M. Spagnuolo, Eds. Springer Berlin Heidelberg, Jan. 2008, pp. 53–82.

[45] H. Hoppe, "Progressive meshes," in *Proceedings of SIGGRAPH*, vol. 96, 1996, pp. 99–108.

[46] R. Pajarola and J. Rossignac, "Compressed progressive meshes," *IEEE Transactions on Visualization and Computer Graphics*, vol. 6, no. 1, pp. 79–93, Jan. 2000.

[47] M. Garland and P. S. Heckbert, "Surface simplification using quadric error metrics," in *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*. ACM Press/Addison-Wesley Publishing Co. New York, NY, USA, 1997, pp. 209–216.

[48] P. Lindstrom and G. Turk, "Fast and memory efficient polygonal simplification," in *Proceedings of the conference on Visualization '98*. Research Triangle Park, North Carolina, United States: IEEE Computer Society Press, 1998, pp. 279–286.

[49] M. Garland and P. S. Heckbert, "Simplifying surfaces with color and texture using quadric error metrics," in *Proceedings of the conference on Visualization '98*.

Research Triangle Park, North Carolina, United States: IEEE Computer Society Press, 1998, pp. 263–269.

[50] H. Hoppe, "New quadric metric for simplifiying meshes with appearance attributes," in *Proceedings of the conference on Visualization'99: celebrating ten years*. IEEE Computer Society Press Los Alamitos, CA, USA, 1999, pp. 59–66.

[51] J. Wei and Y. Lou, "Feature preserving mesh simplification using feature sensitive metric," *Journal of Computer Science and Technology*, vol. 25, pp. 595–605, 2010, 10.1007/s11390-010-9348-7.

[52] M. Hussain, "Efficient simplification methods for generating high quality LODs of 3D meshes," *Journal of Computer Science and Technology*, vol. 24, pp. 604–604, 2009, 10.1007/s11390-009-9249-9.

[53] S.-J. Kim, C.-H. Kim, and D. Levin, "Surface simplification using a discrete curvature norm," *Computers & Graphics*, vol. 26, no. 5, pp. 657–663, Oct. 2002.

[54] C.-C. Chen and J.-H. Chuang, "Texture adaptation for progressive meshes," *Computer Graphics Forum*, vol. 25, no. 3, pp. 343–350, 2006.

[55] N. Coll and T. Paradinas, "Accurate simplification of multi-chart textured models," *Computer Graphics Forum*, vol. 29, no. 6, pp. 1842–1853, Sep. 2010.

[56] J. Cohen, D. Manocha, and M. Olano, "Simplifying polygonal models using successive mappings," in *Proceedings of the 8th conference on Visualization '97*, 1997, pp. 395–402.

[57] J. Cohen, M. Olano, and D. Manocha, "Appearance-preserving simplification," in *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*. ACM New York, NY, USA, 1998, pp. 115–122.

[58] R. Klein, A. Schilling, and W. Strasser, "Illumination dependent refinement of multiresolution meshes," in *Proceedings of the Computer Graphics International 1998*, 1998, pp. 680–687.

[59] P. Lindstrom and G. Turk, "Image-driven simplification," *ACM Transactions on Graphics*, vol. 19, no. 3, pp. 204–241, 2000.

[60] E. Zhang and G. Turk, "Visibility-guided simplification," in *Proceedings of the conference on Visualization'02*. IEEE Computer Society Washington, DC, USA, 2002, pp. 267–274.

[61] P. Castelló, M. Sbert, M. Chover, and M. Feixas, "Viewpoint-driven simplification using mutual information," *Computers & Graphics*, vol. 32, no. 4, pp. 451–463, Aug. 2008.

[62] H. Hoppe, "View-dependent refinement of progressive meshes," in *Proceedings of the 24th annual conference on Computer graphics and interactive techniques - SIGGRAPH '97*, Not Known, 1997, pp. 189–198.

[63] D. Luebke and C. Erikson, "View-dependent simplification of arbitrary polygonal environments," in *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*. ACM Press/Addison-Wesley Publishing Co. New York, NY, USA, 1997, pp. 199–208.

[64] J. El-sana and A. Varshney, "Generalized view-dependent simplification," *Computer Graphics Forum*, 1999.

[65] D. P. Luebke, *Level of detail for 3D graphics*. Morgan Kaufmann, 2003.

[66] M. Garland, "Multiresolution modeling: Survey & future opportunities," *State of the Art Report, Eurographics '99*, 1999.

[67] H. Ji, C. Liu, Z. Shen, and Y. Xu, "Robust video denoising using low rank matrix completion," in *2010 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2010, pp. 1791 –1798.

[68] J. Haldar and Z.-p. Liang, "Spatiotemporal imaging with partially separable functions: A matrix recovery approach," in *2010 IEEE International Symposium on Biomedical Imaging: From Nano to Macro*, 2010, pp. 716–719.

[69] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Found. Trends Mach. Learn.*, vol. 3, no. 1, p. 1–122, Jan. 2011.

[70] S. Ma, D. Goldfarb, and L. Chen, "Fixed point and bregman iterative methods for matrix rank minimization," *Mathematical Programming*, vol. 128, no. 1-2, pp. 321–353, Jun. 2011.

[71] J. Barbic, A. Safonova, J.-Y. Pan, C. Faloutsos, J. K. Hodgins, and N. S. Pollard, "Segmenting motion capture data into distinct behaviors," in *Proceedings of Graphics Interface 2004*, ser. GI '04. School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada: Canadian Human-Computer Communications Society, 2004, p. 185–194.

[72] D. Chen, X. Tian, Y. Shen, and M. Ouhyoung, "On visual similarity based 3D model retrieval," *Computer Graphics Forum*, vol. 22, no. 3, pp. 223–232, Sep. 2003.

[73] Y. Gao, Q. Dai, and N.-Y. Zhang, "3D model comparison using spatial structure circular descriptor," *Pattern Recognition*, vol. 43, no. 3, pp. 1142–1151, Mar. 2010.

[74] M. Botsch, S. Steinberg, S. Bischoff, and L. Kobbelt, "OpenMesh - a generic and efficient polygon mesh data structure," in *1st OpenSG Symposium*, 2002.

[75] AIM@SHAPE, "AIM@SHAPE repository," *http://shapes.aimatshape.net/*, 2013.

[76] "CGAL, Computational Geometry Algorithms Library," http://www.cgal.org.

[77] B. Lévy, S. Petitjean, N. Ray, and J. Maillot, "Least squares conformal maps for automatic texture atlas generation," *ACM Trans. Graph.*, vol. 21, no. 3, pp. 362–371, Jul. 2002.

[78] P. Cignoni, C. Rocchini, and R. Scopigno, "Metro: Measuring error on simplified surfaces," *Computer Graphics Forum*, vol. 17, pp. 167–174, Jun. 1998.

[79] K. Forbes and E. Fiume, "An efficient search algorithm for motion data using weighted PCA," in *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*. New York, NY, USA: ACM, 2005, pp. 67–76.

[80] F. Zhou, F. De la Torre, and J. Hodgins, "Hierarchical aligned cluster analysis for temporal clustering of human motion," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 35, no. 3, pp. 582–596, 2013.

[81] R. Vidal, "Subspace clustering," *IEEE Signal Processing Magazine*, vol. 28, no. 2, pp. 52–68, 2011.

[82] G. Liu, Z. Lin, S. Yan, J. Sun, Y. Yu, and Y. Ma, "Robust recovery of subspace structures by low-rank representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 1, pp. 171–184, 2013.

[83] S. M. Shontz and D. M. Nistor, "CPU-GPU algorithms for triangular surface mesh simplification," in *Proceedings of the 21st International Meshing Roundtable*, X. Jiao and J.-C. Weill, Eds. Springer Berlin Heidelberg, Jan. 2013, pp. 475–492.

[84] C. DeCoro and N. Tatarchuk, "Real-time mesh simplification using the GPU," in *Proceedings of the 2007 Symposium on Interactive 3D Graphics and Games*, ser. I3D '07. New York, NY, USA: ACM, 2007, p. 161–166.