Augmenting the Creation of 3D Character Motion

By Learning from Video Data

XiaoLong Chen

A Thesis

In the Department

of

Computer Science and Software Engineering

Presented in Partial Fulfillment of the Requirements
For the Degree of
Master of Computer Science at
Concordia University
Montreal, Quebec, Canada

June 2012

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By: XiaoLong Chen

Entitled: Augmenting the Creation of 3D Character Motion by Learning from Video Data

and submitted in partial fulfillment of the requirements for the degree of

Master of Computer Science

complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

_____ Chair
Dr. Volker Haarslev

_____ Examiner
Dr. Thomas Fevens

_____ Examiner
Dr. Dhrubajyoti Goswami

_____ Supervisor
Dr. Sudhir P. Mudur

_____ Supervisor
Dr. Thiruvengadam Radhakrishnan

Approved by _____
Chair of Department or Graduate Program Director

_____
Dr. Robin A. L. Drew, Dean
Faculty of Engineering and Computer Science

Date _____

Abstract

Augmenting the Creation of 3D Character Motion by Learning from Video Data

XiaoLong Chen

When it comes to character motions, especially articulated character animation, the majority of efforts are spent on accurately capturing the low level and high level action styles. Among the many techniques which have evolved over the years, motion capture (mocap) and key frame animations are the two popular choices. Both techniques are capable of capturing the low level and high level action styles of a particular individual, but at great expense in terms of the human effort involved. In this thesis, we make use of performance data in video format to augment the process of character animation, considerably decreasing human effort for both style preservation and motion regeneration. Two new methods, one for high-level and another for low-level character animation, which are based on learning from video data to augment the motion creation process, constitute the major contribution of this research. In the first, we take advantage of the recent advancements in the field of action recognition to automatically recognize human actions from video data. High level action patterns are learned and captured using Hidden Markov Models (HMM) to generate action sequences with the same pattern. For the low level action style, we present a completely different approach that utilizes user-identified transition frames in a video to enhance the transition construction in the standard motion graph technique for creating smooth action sequences. Both methods have been implemented and a number of results illustrating the concept and applicability of the proposed approach are presented.

Acknowledgements

I would like to thank many people for their help to bring this research to completion. First of all, I would like to thank my two supervisors Dr. Sudhir P. Mudur and Dr. Thiruvengadam Radhakrishnan for their exceptional support and great guidance throughout this research. This project was funded through Dr. Mudur via research grants from NSERC, GRAND NCE, and Concordia University's ENCS Faculty. I would also like to thank PHD student Kaustubha Mendhurwar for his participation throughout the whole course of this academic exploration with remarkable efforts. I also thank everyone who had participated in the construction of the kickboxing action database. My thanks also go out to all my lab mates who have been there for me, days and nights. Last and foremost, I thank my family, especially my dearest one, for the trust, encouragement and patience. Without them, the completion of this research would have not been possible.

Table of Contents

TABLE OF CONTENTS ..... **V**

LIST OF FIGURES ..... **VII**

LIST OF TABLES ..... **VIII**

1 INTRODUCTION **1**

1.1 Motivation and Problem Statement 1

1.2 Methodology 5

1.3 Summary of Contributions 6

1.4 Thesis Organization 7

2 BACKGROUND AND RELATED WORK **9**

2.1 Computer Animation Techniques and algorithms 9

2.1.1 Interpolation Based Animation *10*

2.1.2 Motion Capture *12*

2.1.3 Motion Graph *13*

2.1.4 Motion Editing *14*

2.1.5 Physics Based Animation *15*

2.1.6 Data Driven Animation *16*

2.1.7 Behavior Modeling *16*

2.1.8 Concluding Remarks on Character Animation Techniques *17*

2.2 Relevant Topics in Computer Vision and Algorithms 17

2.2.1 Object Recognition *18*

2.2.2 Action Recognition from Video *19*

2.2.3 2D/3D Pose Estimation *20*

2.2.4 Motion Tracking *21*

2.2.5 Summarizing Our Use of Computer Vision Techniques *22*

2.3 Relevant Topics in Machine Learning 22

2.3.1 Supervised Learning *23*

2.3.2 Unsupervised Learning *23*

2.3.3 Summarizing Our Use of Machine Learning Techniques *25*

3 REVIEW OF APPLICABLE TECHNIQUES AND MAJOR CHALLENGES **26**

3.1 Challenges in Action Recognition 26

3.2 Challenges in 3D Pose Estimation 28

3.3 Challenges in Motion Graph 30

3.4 Challenges in Clustering 31

3.5 Challenges in SVM 32

3.6 Concluding Remarks on Challenges 32

4 LEARNING ACTION STYLE FROM VIDEO DATA AND ITS APPLICATION **33**

4.1 Feature Word Vocabulary Creation 35

4.1.1 Orientation Assignment *36*

4.1.2 Descriptor Representation *39*

4.2 Action Classifier Training 40

4.3 Frame-wise Action Recognition 42
4.4 Capturing the Behavior Pattern With HMM 43

5 AUGMENTING MOTION GRAPH WITH VIDEO DATA 46
5.1 Motion Graph 46
5.1.1 Identifying Candidate Transition Nodes 49
5.1.2 Selection of Transition Points 51
5.1.3 Creating Transitions 51
5.1.4 Pruning the Graph 53
5.2 Improving Motion Graph Technique Using Video Data 54
5.2.2 Manual Annotation of Transition Frames 56
5.2.3 2D Pose Estimation from Video Data 57
5.2.4 Viewpoint estimation 59
5.2.5 Matching 2D Pose Information with 3D Motion Data 60
5.2.6 Transition frame identification 61
5.2.7 Incorporating the extra transition node 63

6 IMPLEMENTATION 64
6.1 Learning Action Style from Video and Its Application 64
6.1.1 Preparing a video action database 64
6.1.2 Action Vocabulary Construction 65
6.1.3 Action Classifiers 66
6.1.4 Action Recognition 68
6.1.5 HMM Training and Action Sequence Generation 69
6.1.6 Creating New Motion Sequences 70
6.2 Augmenting the Motion Capture with Video Data 72
6.2.1 Motion Capture Action Database 72
6.2.2 Motion Graph Implementation 72
6.2.3 Pose Estimation from Video Data 73
6.2.4 Character Viewpoint Estimation in Video Frame 74
6.2.5 Mapping 2D Body Part Data to 3D Joint Data 75
6.2.6 Incorporating the New Transition Node into the Standard Motion Graph 76
6.3 Conclusion 77

7 CONCLUSIONS 78

REFERENCES 82

List of Figures

Figure 1: A High Level Action Style Learning Model. 34
Figure 2: 2D to 3D SIFT descriptors [7] 38
Figure 3: Visualization of the Abbreviated Descriptor [7]. 40
Figure 4: Frame-wise Action detection 43
Figure 5: HMM [16] 44
Figure 6: A High Level View on the Method 47
Figure 7: Motion Graph Building Process [13] 48
Figure 8: A Simple Motion Graph [13] 49
Figure 9: Blending To Create Transition 53
Figure 10: Strongly Connected Component 54
Figure 11: Transition Frame Manual Annotation 56
Figure 12: Body Model 59
Figure 13: Pose Estimation Example Results 59
Figure 14: 2D to 3D mapping 61
Figure 15: The Enhanced Distance Function 62
Figure 16: Sample Frames of the Kickboxing Video Database 65
Figure 17: Sample signature of a video frame 67
Figure 18: Grouping Histogram 67
Figure 19: Frame Level Action Recognition Results 68
Figure 20: Subject Wearing the Shapewrap System 71
Figure 21: Synthesis Results From a Sample Video Sequence 71
Figure 22: Transition Created In Standard Motion Graph 73
Figure 23: Sample Pose Estimation Result 74
Figure 24: A Complete 2D to 3D Matching Process 76
Figure 25: Augmented Motion Graph Output 76

List of Tables

Table 1: Confusion Matrices 66
Table 2: Confusion Matrices for Frame-wise Action Recognition Accuracy 69
Table 3: A Sample Training Sequence Fed to HMM 69
Table 4: An Example Sequence Output from HMM 70
Table 5: Body Part Detection Accuracy Table 74
Table 6: Confusion Matrix for Viewpoint Detection 75

1 Introduction

The medium of animation, compared with other media such as images or text, can quickly convey a large amount of information to the audience. Computer animation, especially articulated character animation, has been a vital element attributing to the success of video games and movies in these days. Long, complex realistic human motions create the required sense of character realism. A motion sequence is considered realistic when it does not contain visual discontinuities in motion, complies with physical constraints and most importantly, faithfully depicts the behavior style of the object. A realistic articulated character animation sequence depicting an individual's style is particularly difficult to produce. This is not only because of the large number of degrees of freedom which a human body possesses, but also because of the human viewers' discernibility in perceiving human-like motion due to the high level of familiarity human beings have to their own actions. The research work reported in this thesis addresses some aspects of this problem.

1.1 Motivation and Problem Statement

The available solutions to create articulated character animation can be divided by the level of detail in its specification. Hand drawing animation frames is a very traditional animation technique, which is clearly the lowest level as every detail in the required animation has to be specified by the animator. Although it clearly requires enormous amount of human efforts; even today, some artists, such as the Japanese manga artist and

prominent film director and animator, Hayao Miyazaki [24], insist on using the hand drawing technique. Motion capture, or mocap for short, is considered to be another form of low level of specification, where the spatial-temporal locations and orientations of the key human joints are directly recorded. However, motion capture data is usually only suitable for a direct play back. Motion capture data alteration is an active and difficult research field. On the other hand, a high level specification for computer animation refers to a process of creating animations by simply specifying a set of higher level attributes. Behavior models are considered as high level specification where animators only need to specify the rules and constraints and computers take over the rest of work, and do the required simulation to create the whole animation sequence. Behavior models have been successfully applied in many domains to create animations dynamically. However, much like low level specification techniques, behavior models for human characters require intensive manual calibration to ensure that human eyes do not perceive any unnatural human actions. Moreover, humans are also very good at detecting distinctive action traits, which are very important when modeling subjects who are familiar to the audience. To conclude, generating human character animation generally requires major human efforts, with or without high level specification methods.

How to decrease the human efforts when creating 3D articulated character animations is the focus of this thesis. Let us look at the root of the problem from closer range. Creating articulated character animations mostly involves reproducing character action styles. Action styles can be divided into low level action styles and high level action styles. Low level action styles are the styles that bear the detailed characteristics of how a particular character performs an action, e.g. such as how the arms swing or the legs stride

when the character does a running action. Most of the low level actions can be captured/created/modified from individual performances and repeatedly used as many time as required by the animator. It is the remaining variances that make this a difficult problem for autonomous characters. They are difficult to capture/create since the variances are not as constant as the individual actions. High level styles are often known as behavioral patterns. Behavioral patterns refer to the logical groupings of recognition, of process or action that has actually worked for us in a previous case, given a set of circumstances [25]. They are the conscious decisions or instinct reflects an individual makes under a certain context, out of personal experiences. For example, a sequence of kick-boxing actions can very well be viewed as a mixture of the unconscious and conscious reactions of the performer because the order in which the actions had been performed and their effects convey his/her observations and experience, which later have formed into a distinct style that is specific to him/her. To bring a particular character to life, this distinct style has to be properly taken care of because it is clearly perceptible to the audience. Therefore, the motivation of capturing this pattern and the goal of generating new sequences following the same pattern are easily understandable and important in character animation creation.

At the first glance, Motion Graph [13] seems to be providing the required solution. It connects together short motion clips of the basic actions, assuming they have been captured/created to faithfully reproduce the low level styles of the performer, to form a long, smooth and complex animation sequence. Each captured/created action is represented as a linear graph. The motion graph technique introduces additional links between every two actions, say, A and B, for transitioning from A to B. These links are

determined based on similarity between the “from frame” in A, and the “to frame” in B. If these transition frames are well chosen and the clips are carefully ordered, the resulting concatenation can be made to follow the high level behavior pattern of a performer. However, at this time, it still involves extensive human efforts. First of all, the animator has to analyze human action patterns and generalize them into one or more sequential order to be followed when concatenating individual motion. Secondly, transition thresholds required for selecting the transition frames have to be manually set in order to balance the responsiveness and smoothness of a motion graph for obtaining the desired result in the sequence creation process. Finally, physical constraints to be met by the character (say satisfying gravity, or joint angle constraints, etc.) are also set manually with a post data-cleaning process to ensure constraint compliances. In spite of all this human involvement, standard motion graph technique based on geometric interpolation is limited, as it cannot always generate realistic motion sequences. For example, transition from a walking action to a hand flipping action would likely involve a braking action in-between. This action cannot be produced by the simple geometric interpolations between the two actions. Animators would have to manually create the braking action in between.

In this thesis, we propose two new methods requiring considerably reduced human efforts in producing realistic new motion sequences for character animation. In the first method, we utilize action recognition and machine learning techniques to recognize action patterns from video data and to automatically synthesize realistic high level 3D human action sequences from motion clips. In the second method, we address one of the low level animation requirements, namely, incorporating the transition styles of an individual when concatenating actions into a longer sequence. For this, we employ recent

advancements in pose estimation from video data to retrieve style defining transition poses from a 3D mocap database and help augment the standard motion graph model so as to improve the realism of motion transitions, again with reduced human efforts.

1.2 Methodology

The methodology we have followed in this research is described in the following. Firstly, we assume that video and motion capture clips for basic actions are labeled and available in a database (for our experiments, we created our own databases). Our goal in the first method is to learn the high level motion behavior of a human performing various action sequences, captured in video, and to transfer this behavior to a virtual character, usually a non-playing character (NPC) in a game. We have chosen to use video data for learning, because it is much easier to capture than 3D motion capture. The strategy is to employ action recognition techniques to detect actions at frame level from the video sequence. For this, we extract suitable features from video frames and use them to train a support vector machine (SVM) classifier to recognize actions to which a frame belongs. The result is a coded action sequence in which each frame of the video sequence is assigned the label corresponding to the action it belongs to. After a noise cleaning process, the results are fed into a Hidden Markov Model (HMM, a Finite State Machine with transition probabilities) to train it with action transition probabilities. This way, the high level behavior pattern in the human performance is captured and can be used to synthesize new action sequences.

In the second method, we propose an augmented motion graph called AugMG. We use pose estimation techniques to derive a 2D pose with joint location information and viewpoint information for each of the manually annotated transition frames in the video. Once again, there is an extensive training phase to train the body parts detectors and the viewpoint classifiers. With a large motion clip database, for each of these 2D poses, we assume that we can retrieve a corresponding 3D skeleton pose from the database. The newly retrieved skeleton poses are used as reference points and their neighboring frames in the motion clip serve as enhancement nodes. These enhancement nodes lead to more realistic transitions with little further human efforts. Both methods have been implemented and as proof of concept, a complete end-to-end experiment was conducted and the results in the form of realistic character animation have been created.

1.3 Summary of Contributions

The most significant contributions of this research are the formulation, design and presentation of two new methods targeted towards improving articulated character animation creation: incorporating high-level and low-level action style into character animation. We also provide various experimental results using our implementation demonstrating the applicability of our methods. Our work integrates various techniques from computer vision and machine learning into the process of animation creation for articulated characters, and to the best of our knowledge, these are the pioneer efforts that use video data to augment articulated character animation.

As the applicability of our methods relies greatly on the action recognition rate, a second major contribution is the enhancement of the current action recognition technique to improve the action recognition rate at frame level. Moreover, we also present a simplified 2D to 3D pose matching algorithm that is highly effective in the context of our problem.

1.4 Thesis Organization

As already mentioned above, this research touches two different computer science fields: Computer Animation and Computer Vision. Therefore in Chapter 2, we first provide a comprehensive overview of these two fields and then focus on the particular sub-areas that we are interested in – 3D motion creation using motion capture and key frame animation techniques, along with computer vision techniques for action recognition, pose estimation and pose reconstruction from video data.

Chapter 3 discusses in more details our motivation for adopting our approach. We explain what exactly we are trying to achieve by conducting the investigation on using computer vision techniques to help improve the current process of articulated character animation construction.

Chapter 4 and Chapter 5 provide the details of the two new methods developed as part of this research. In Chapter 4 we present our methods for transferring character behavior pattern from video data onto 3D characters. In Chapter 5 we present our AugMG method in detail.

Chapter 6 offers a detailed description about the prototype implementations of the two methods, from behavior pattern extraction, synthesis of animation sequences, and motion graph augmentation for incorporating individual style into transitions. We also present the results of the various experiments conducted using our implementation.

Chapter 7 concludes the thesis by summarizing the contributions. It also presents the limitations of our approach and future research directions.

2 Background and Related Work

The major objective of computer animation is to bring a lifeless object to life, through the use of computer systems. While our research efforts are mainly focused on tackling the issues in character animation, our methods heavily rely on computer vision and machine learning techniques. In this section, we give a general introduction to character animation, computer vision and machine learning fields. As all three fields are challenging subjects spanning a wide range of areas, we focus on the particular sub-areas that are of immediate relevance in achieving the objective of this thesis: augmenting the creation of 3D character motion by learning from video data.

2.1 Computer Animation Techniques and algorithms

Computer animation is a process used to generate animation frames/images by using computer graphics [1]. It is a re-application of the famous stop motion techniques on the computer graphics pipeline with digital models and frame to frame animations. There are 12 basic principles of animation: squash and stretch, anticipation, staging, straight ahead action and pose to pose, follow through and overlapping action, slow in and slow out, arcs, second action, timing, exaggeration, solid drawing and appeal [53]. Each of the following techniques was introduced to fulfill one or several of these principles.

2.1.1 Interpolation Based Animation

Key frame animation is probably the earliest technique employed to produce animation. In a traditional key frame animation, more experienced and talented animators provide the key frames that depict key dynamic changes in the sequence, and other animators fill in the supporting frames. The process of filling in the supporting frames between the key frames is known as in-betweening or tweening. Nowadays, while the key frames are often defined by animators, the in-between frames can be generated by a computer through straightforward techniques. Therefore, the basic foundation of key frame computer animation is the interpolation of values, which refer not just to spatial values, but could also include arc length, image pixel colors, and shape parameters [534].

The simplest interpolation of values is point to point linear interpolation, which forms the basis of all other complex interpolations as well. The interpolation between two data points, say $(x_a, y_a)\ and\ (x_b, y_b)$ is given by

$$y = y_b + (y_b - y_a)\frac{(x-x_a)}{(x_b-x_a)} \text{ at the point } (x, y) \quad (1)$$

However, linear interpolation alone is usually insufficient to generate a sophisticated animation sequence. Therefore, polynomial interpolation and spline interpolation were employed, where spline interpolation is preferred over polynomial interpolation because the interpolation error can be made small even when using low degree polynomials for the spline. A curve to curve interpolation is more powerful but also much more complex. It requires information such as the point correspondence and speed of interpolation.

Moving point constraints [13] is an algorithm of this kind that allows the users to specify these parameters.

The shape transformation or deformation requires more sophisticated techniques. There are physics based systems providing rather high level abstractions to simulate the effects of the external forces acting on objects. However, they are usually computationally expensive and the outcomes might be unpredictable or non-ideal depending on the circumstances [39, 57, 59], which provides little control to the animators over the exact transformation of the shapes. There are numerous techniques introduced to modify the shapes of the objects, such as affine transformation and non-affine transformation, vertex displacement, Free Form Deformation (FFD) [55, 56, 58]. The morphing between 3D shapes is even more complicated. Surface based techniques and volume based techniques are two basic techniques used for 3D shape transformation [60]. In summary, shape transformation problems must address two fundamental sub-problems: the correspondence problem and the interpolation problem [48]. Automatically establishing the correct correspondence requires feature identification and matching, and obtaining the correct interpolated result depends on the control parameters used in the interpolation scheme. For capturing individual movement styles, considerable human involvement would be required.

2.1.2 Motion Capture

Motion capture (mocap) is a popular means to record movement of objects or persons. It is especially the case since the motion capture devices are becoming more accessible nowadays. There are two major motion capture approaches: electromagnetic motion capture technologies and optical tracking systems. Electromagnetic motion capture systems use transmitters to establish magnetic fields within a space and use sensors to detect the position and orientation within the space based on these fields [27]. It requires a more closed-in environment to avoid magnetic field distortion. The information from electromagnetic sensors can be displayed in real time with the expense of possible distortion and accuracy suffering. Electromagnetic sensors are usually heavy and clumsy, which limit the type of the actions that can be performed while wearing them. Optical tracking systems use visual markers and a number of special cameras to determine the 3D location of the markers [27]. Optical markers are passive objects that reflect a specific color of light which is to be received by the cameras. Optical markers are less intrusive than electromagnetic sensors as they are much lighter and easier to wear and they can be used in a much larger range. However, they cannot provide real time feedback and the data is more error prone and noisy.

The motion capture data is applied to a skeletal structure to drive its motion. While motion capture can reproduce the real object's action, its main disadvantage is that it is more intrusive, and its data is extremely difficult to modify. Recently Microsoft released a low resolution motion recognition system called Kinect that requires no markers. Bundled with the Xbox 360, Kinect projects an invisible infra-red laser pattern for depth

recovery and motion acquisition. Together with the video images captured, Kinect provides a full-body 3D motion capture. Its sensing range is adjustable, though limited to an indoor setup. It also has particular requirements on the room spacing and the background color [35].

Of course, there is the other method of motion capture: computer vision-based motion capture. This technique processes video streams using image analysis algorithms to synthesize 3D motion data. The major advantage of vision-based motion capture is the ready availability of video data and its non-intrusiveness nature. However, vision-based motion capture has not been very successful due to its limited accuracy and extremely demanding computing efforts. Taking this into consideration, in our methods, the 2D spatial-temporal pose position and orientation information estimated from video data are used only to improve the quality of the action transition instead of directly trying to generate 3D animation data from 2D video data.

2.1.3 Motion Graph

Long and complex natural looking human like motions are required to create the sense of character realism in movies, games and virtual worlds simulations, etc. For autonomous characters which react to events in the virtual world, it is labor intensive to create all the possible animation sequences using motion capture or key frame animation. A popular solution is motion concatenation. Motion concatenation is the technique of piecing together short motion clips to generate a long motion sequence with spatio-temporal variances. Similar frames between the motion clips can be used as transition points. Using a graph structure to represent motion clips was proposed in several parallel studies

[13, 37, 38.]. Among which, [13] provided a unique distance function that incorporates the velocities and acceleration of the character into account when computing transition positions between two actions. It also ensures responsiveness and smoothness of the animation sequence. Responsiveness is concerned with the time taken for a virtual autonomous character, which is the middle of performing an action, to react to a sensed event and transit to a new action. Smoothness is concerned with the visual continuity when changing from one action to another. With suitable data cleaning as a post-processing step to incorporate real world requirement and constraints, the motion graph technique is capable, in principle, of creating interactive animation sequences with smooth transitions.

2.1.4 Motion Editing

The common misperception is that motion editing is only required because the motion data is not perfect. Motion editing is necessary even when the motion data perfectly represents a real-life performance. There are multiple reasons for which motion editing is required: motion data re-use, creating infeasible motions, imperfections of reality, change of intent and addition of "secondary" motion [27].

The common techniques for motion editing allow users to make changes to the existing motions while maintaining the integrity of the motion by imposing some restrictions on the motion. The integrity of a motion is rather subjective. It could refer to the physical correctness, the nuance of the motion or any other aspects of the motions perceivable to the viewer. Depending on how the original motion is constructed, motion editing techniques can vary.

A motion generated by key frame animation techniques is comparatively simpler to edit. Users can directly manipulate the key frames using the same system with which the animation sequences were originally generated. Motion displacement techniques ensure that neighboring frames are also altered correctly. Inverse kinematics methods can be used to ensure that editing respects the skeletal structure of the model and other physical constraints.

Motion capture data, on the other hand, is very difficult to edit. First of all, it registers joint orientation and location information for each frame in a motion sequence, and that is certainly very inconvenient for editing. Secondly, there is little indication in the data to show the correspondences between the data and the properties of the motion. Thirdly, sensor errors and other failures lead to "dirt" motion. Lastly, it is difficult to correct the motion because we do not have the exact record of what happened [27]. In spite of this, in principle, motion editing techniques could also be used to create realistic and smooth animation sequences from captured data.

2.1.5 Physics Based Animation

A physics based animation is built based on the principle of physics, and it focuses on animation of an object by satisfying the laws of motion. Some of the attributes are not physics related but constraints applied to make the sequences to have a particular feeling. The major advantage of the system is that it allows the animators to create as many different sequences as possible by just adjusting a few parameters and at the same time frees them from concerns related to the low level animation details. However, modeling a physics-based system usually requires a lot of computation resources and real life

experience. A small offset in a physical parameter of the model can create a major distortion in the resulting animation. There is also only finite number of characteristics known to the animators to model the system. The physics based animation systems of today still are difficult to use for most animators and still cannot produce the same look and feel of the motion capture.

2.1.6 Data Driven Animation

Data driven animation methods use mocap data to animate characters. They are able to reproduce the low level style and mannerisms of a particular individual. However, basic data driven animation can only play back the captured animation sequence and cannot adapt it to new circumstances. For this reason, a number of methods have been used, such as physical simulation, multiple motion interpolation, etc. in order to modify the current motion capture data. As we have earlier discussed in the motion editing section, finding the correspondence between the motion data and the properties of interest is not trivial. Instead of modifying the motion capture data, motion graph techniques [13, 37, 38] use simple interpolation to sequence motion clips together to create a new motion. The order of actions in the sequences is based on circumstances. The simple interpolations between the transition points avoid the issue of finding the correspondence.

In our experiments, we have used data driven animation methods for the low level actions.

2.1.7 Behavior Modeling

Development of behavioral animation is today largely driven by computer games, film industry and real life simulations. There are different types of behaviors: vision aware

behavior, aggregate behavior, flocking behavior, prey predator behavior and autonomous behavior [61]. Many of these behaviors share common properties but they are usually applied for different purposes. For example, autonomous behavior shares many traits with flocking behavior but it is applied on much fewer objects with much more complex cognitive reactions. It usually maintains an internal state that models the time-varying needs, desires and emotions, and at the same time an external state that models the perceivable but changing environment that affects its cognitive reasoning [54]. In this research, the autonomous behavior pattern is learned from video and is used to drive the generation of new action sequences.

2.1.8 Concluding Remarks on Character Animation Techniques

Many different character animation techniques are inter-related, and it is very common to employ multiple animation techniques to achieve a single purpose. For example, motion graph technique utilizes the interpolation technique to generate smooth transition frames in between the motion capture segments. Physics based animation models frequently utilize motion capture data. One of the open challenges in the computer animation is to produce a realistic animation sequence that is indistinguishable from a video clip.

2.2 Relevant Topics in Computer Vision and Algorithms

Computer vision is a field that is concerned with the use of visual data, captured from the real world, towards extracting and interpreting information on a real time basis [3]. Its research is dedicated to open the “eyes” of computers. Computer vision has many

applications, such as surveillance, input for user interface, bio-mechanical analysis, navigation, organizing information, etc.

Like computer animation, computer vision is a vast computer science field that overlaps with human vision, image processing, pattern recognition and machine learning. It requires a deep understanding of several domains such as artificial intelligence, machine learning, mathematics, neurobiology, physics, signal processing, and control robotics. Computer vision research can be split into many sub-domains, such as object and action recognition, motion tracking, pose estimation, image restoration, event detection, etc. A complete survey of computer vision is a major research effort by itself. Therefore, we will only discuss relevant work in the sub-domains that are of particular relevance to our research.

2.2.1 Object Recognition

Humans recognize objects despite the fact that the objects might be translated, rotated, scaled or even partially occluded. Object recognition in computer vision is concerned with the task of equipping computer with the same ability. The exact mechanism that accounts for successful object recognition by human beings is still under active research. It could be certain traits of the object, such as shapes, colors, patterns or some other not yet known features that play the active roles in the process. Since exactly how the human brain recognizes objects still remains to be an active research field, object recognition employs many different methods in order to accomplish the same goal [2]. Appearance-based methods use appearance of the example images of the objects to train a computational model to perform recognition [47]. Shape-based methods represent the

objects by their shapes and contours [47]. Pedestrian detection using shaped-based methods are described in [48]. Model-based methods represent the objects as a collection of primitives (boxes, spheres, etc.). Feature-based methods extract distinctive features from the objects during training and search for these features during testing to find feasible matches. A particular descriptor called Scale Invariant Feature Transform (SIFT) is of specific interest to us in our work on video feature extraction for its invariance to affine transformation and partial invariance to viewpoint and illumination changes [22].

2.2.2 Action Recognition from Video

The goal of action recognition is to recognize actions performed in a video. It can be considered as an extension of object recognition. The interest in this subject is motivated by the promise of many applications such as man-machine interaction, video data indexing and retrieval, video conference, and automatic surveillance, etc. Action feature extraction is considered the building block of feature based action recognition methods. Actions are often regarded as three dimensional entities – spatial and temporal; therefore, the features of human actions are often in three dimensional volumes. These features are usually complex and involve manual inferences and significant computational efforts. There are a number of proposed features such as optical flow, Histogram of Oriented Gradient (HOG), and 3D SIFT [7].

The computational process for action recognition is often divided into object detection, object tracking, object activity recognition and a high-level activity evaluation [3]. Each step can be difficult to tackle depending on the application domain. For example, object detection becomes very complicated when the backgrounds or viewpoints are constantly

changing. Object tracking is very difficult when there are continuous occlusions of the object. Therefore, there are no generic solutions for action recognition. The methods are often carefully developed for one particular application domain.

There are a number of well-known efforts dedicated to this field. Aggarwal et al [5] discussed human body structure analysis, motion tracks and action recognition in depth. Bobick [6] presented a taxonomy that covers movement recognition, action recognition and activity recognition. Silhouette shape based image presentation feature is introduced by Yilmaz et al [8]. Paul et al [7] introduced a 3-dimensional (3D) SIFT descriptor to represent the video data. The Bag of Words approach [9] was borrowed from text categorization to apply to the action recognition paradigm. A local SVM approach [10] was introduced to recognize complex motion patterns. It constructs the video representations with space-time features and integrates such representations with SVM classifications. An extensive survey on vision based human action recognition is presented in [11].

2.2.3 2D/3D Pose Estimation

There are many applications such as human-machine interaction and robotics in which only recognizing the object or its actions are not sufficient. The estimates of the object's position and orientation are also required. This specific task of determining the pose of an object in an image or an image sequence is referred to as pose estimation [4]. While action recognition is usually a classification problem, pose estimation is considered to be a regression problem. Aggarwal et el [5] use a taxonomy with three categories: body

structure analysis, tracking and recognition. Research efforts are devoted to two correlated fields: 2D pose estimation and 3D pose estimations. The process of 2D pose estimation for articulated characters usually involves skeletal model construction, body part detection, and joint data estimation. In the literature, there are several surveys available that provide an overview of the different approaches to pose estimation [12, 29].

In the case of 3D pose estimation, camera location and orientation, and 3D joint data extraction are also required, besides the steps mentioned above for 2D pose estimation. The 3D pose estimation is often restricted to large body chunks, such as torso, arms, legs, etc. There are two main approaches in 3D pose estimation: model-based and learning based. A model-based approach employs a priori information about the human model, including a kinematic structure, the body dimensions, and a function that describes how the human body appears in the image domain [12]. In contrast, learning based approaches estimate a model that directly recovers the 3D pose estimation from observable image evidence. Example-based learning approaches estimate 3D pose estimation by comparing the given input image with the example images whose 3D poses are pre-known [50, 51, 52].

2.2.4 Motion Tracking

Tracking motion in a video sequence has been an important topic in computer vision. It is concerned with locating a moving object over a sequence of consecutive video frames. It has a variety of uses, some of which are: human-computer interaction, security and surveillance, video communication and compression, augmented reality, traffic control, medical imaging and video editing [28]. There are various tracking algorithms developed,

such as blob tracking, kernel-based tracking and contour tracking. Each of these algorithms has its strengths and weaknesses, and the choice of using one over another is dependent on the applications. Tracking techniques can become rather complicated because it usually involves object recognition. It is also computationally demanding due to the amount of data that needs to be extracted and processed from video. However, tracking techniques can greatly reduce the solution space in action recognition and pose estimation problems and consequently improve the accuracy of these algorithms.

2.2.5 Summarizing Our Use of Computer Vision Techniques

In our research, we use 3D SIFT for creating the keypoint descriptor used for action recognition in combination with the Bag of Words approach to represent the video data at frame level. The second method proposed by us also employs pose estimation and 2D pose to 3D pose mapping techniques to search for potential enhancement points in the motion capture database. As motion tracking techniques are often applied to remove noise in a sequence of detections, they can be used to complement our methods; however, we have not incorporated them in our prototype implementation.

2.3 Relevant Topics in Machine Learning

Machine learning is a scientific discipline concerned with the design and development of algorithms that allow computers to evolve behaviors based on empirical data, such as data obtained from sensors, through logging or from a large database [36]. Machine learning allows the machine to learn via inductive inference based on observing data that represents incomplete information about statistical phenomenon and generalize it to rules

and to make predictions on missing attributes or future data. Machine learning algorithms can be organized into a taxonomy based on the desired outcome of the algorithm [14]: supervised learning, unsupervised learning, semi-supervised learning, reinforcement learning, etc. As we have used both supervised and unsupervised learning in our methods, we shall briefly review these two further.

2.3.1 Supervised Learning

Supervised Learning is a machine learning task of inferring a function from supervised training data. In supervised learning, each training example is a pair consisting of an input object (typically a vector) and a desired output value (also called the supervisory signal). A supervised learning algorithm analyzes the training data and produces the inferred function, which is called a classifier if the output is discrete or regression function if the output is continuous [23]. Support vector machine (SVM) is a concept that is often used in supervised learning. The standard SVM takes a set of input data and predicts, for each given input, which of two possible classes forms the input, making the SVM a non-probabilistic binary linear classifier [30].

2.3.2 Unsupervised Learning

Unsupervised learning refers to the problem of trying to find hidden structure in unlabeled data. Because the training data are not pre-labeled when given to the learner, there is no error or reward signal to evaluate a potential solution. This distinguishes the unsupervised learning from supervised learning. The approaches used in unsupervised learning include clustering and blind signal separations. Among them, k-means clustering,

principal component analysis and Hidden Markov Models are the ones we have used in our methods. These are briefly discussed further below.

K-means clustering

K-means clustering is a method of cluster analysis which aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean. It can be viewed as data space been partitioned into Voronoi cells [31]. The problem is computationally difficult (NP-hard).

Principal Component Analysis

Principal Component Analysis (PCA) is a mathematical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principle components. The transformation is defined in such a way that the first principal component has the largest possible variance and each succeeding component in turn has the next highest variance possible under the constraints that it be orthogonal to the preceding components [32].

Viterbi learning

The Viterbi algorithm is a dynamic programming algorithm for finding the most likely sequence of hidden states – called the Viterbi path – that results in a sequence of observed events, especially in the context of Markov information sources [33], and more generally, Hidden Markov Models. HMM is a statistical Markov model in which the system being modeled is assumed to be a Markov process with unobserved (hidden) states. An HMM can be considered as the simplest dynamic Bayesian network [34].

2.3.3 Summarizing Our Use of Machine Learning Techniques

In our research, we have used Support Vector Machine (SVM) methods for frame-level action recognition and viewpoint classification. We employ Hidden Markov Model methods in combination with other learning techniques such as k-means clustering method and Principal Component Analysis (PCA) to learn the higher level behavior patterns. The following chapter elaborates further on the major challenges that we are faced with when using these techniques for our goals.

3 Review of Applicable Techniques and Major Challenges

The objective of our research in the broadest sense is to automate the creation of the realistic 3D human character animation by learning from real world human performances captured through the medium of digital video. The tasks involved include action recognition, motion capture and motion graph, pose estimation, and behavior modeling. The techniques for automatically carrying out these tasks constitute a minimal set of technology elements required to achieve our objective. While these fields are still under active research, as discussed in the previous chapter, there are some inherited challenges which we have to recognize.

3.1 Challenges in Action Recognition

Action recognition can be considered as an extension of object recognition. Object recognition is extensively researched and is usually carried out in one single image. Vision-based action recognition usually involves a sequence of continuous frame images and is considered to be a more difficult topic.

The choices of features and classification algorithms used in action recognition are greatly dependent on the application domain. There are a few challenges that influence their choice.

First of all, one of the impediments in action recognition that usually lead to un-desirable action classification result is the adequacy of intra and inter-class variations. While we

expect the variations between different actions to be large, sometimes they could be indistinguishable. For example, walking actions could be similar to running actions because they seem only different by the speed and stride length, which are features largely dependent on individuals. We also expect the variations between different individuals performing the same actions to be small, given that humans recognize them as the same so well. Unfortunately, sometimes the anthropometric differences between the individuals for the same action can be large. For example, the running actions performed by one person can be very different from another in terms of movement speed and frequency of steps in a given time interval. A good human action recognition approach should be able to generalize over variations within one class and distinguish between actions of different classes [11]. The solutions usually depend on the choices of features and action representations.

Environmental and recording setups also have great impact on the accuracy of the action classification. It is much harder to localize a person in a changing or cluttered background. A dynamic moving camera makes the localization even more difficult, compared to a static camera. Environmental illumination can also affect the image features. Different viewpoints can lead to different observations on the same action. While there is not a well-known image representation or classification algorithm that can handle all these variations, these issues can be avoided by restricting the diversity in the training video and testing video, according to the application domain. However, such restrictions also mean compromises in the generality of the solutions.

Further, an activity sequence consists of actions of different classes. Action segmentation in a video sequence can be a problem. For example, in an activity of “kick-boxing”, a “kick” action usually takes more time than a “punch” action does. However, this is also individual-dependent. There can be a substantial variation in the rate of an action being performed. A robust recognition algorithm must take into account of the rate of the action being executed [11].

Keeping the challenge in action recognition in mind, we chose 3D SIFT as the keypoint descriptor because of its well-known properties of being invariant to illumination, viewpoint, and affine transformation. We also recommend using actions with rather distinct features as basic actions in a sequence to avoid the sharing of style traits. A static background should be imposed in the recording setup to reduce noise in object detection and action recognition. We carry out action recognition operation at the frame level, and action segmentation is still a potential issue to be resolved well. As HMM is a probabilistic model, certain amounts of detection and segmentation errors are tolerated.

3.2 Challenges in 3D Pose Estimation

Pose estimation shares many of the challenges that action recognition faces. The task gets even more challenging when it comes to deriving 3D poses from 2D images. 3D pose estimation is an ill-posed problem because the depth information of the object is lost during video recording. Until today, available 3D pose estimation algorithms can only achieve the goal under highly restricted conditions, and manual annotations are commonly required [42, 43, 44].

First of all, since frame images are the projection results of 3D objects, the depth information of the objects is lost in a frame image. Therefore, many 3D poses can appear to be the same when been projected into a 2D plane. Depth sensors or multiple concurrent cameras are often deployed to derive the depth information. However, depth estimation from single monocular images still remains a very active research subject.

Secondly, despite the fact that kinematic human model is the most natural choice for parameterizing the human body state, static and dynamic constraints are employed to complement the human model to reduce the possible solution space by specifying joint dependencies and the parameter ranges. A human body is a highly articulated structure with more than 690 degree of freedom (DOFs), consisting of 230 movable parts with three degrees of freedom each. Even a simplified version could still contain around 80 DOFs. How to recover these parameters is the main problem in 3D pose reconstruction and estimation.

Thirdly, there could be many self-occlusions between body parts in a 2D image space. Hands could be occluded by the torso, and legs could be occluded by each other. These self-occlusions complicate the tracking problem and make it difficult to segment different body parts to have a well separated body structure in the image.

Lastly, the dynamic background, changing light conditions, moving cameras, etc. could easily affect the result of a pose estimation algorithm. Many solutions assume highly controlled background and lightings, which make the solutions tightly coupled with specific application domains. A generic and robust solution which is invariant to environment and set up changes is very much under research.

In our work, we use a body model consisting of 10 body parts and 6 joints. This condensed version of body structure improves the 2D pose estimation result and simplifies the overall 2D to 3D pose mapping process. The possible false matches introduced in the process can be removed by taking into account the adjacent motion frames when calculating the distance value used in the mapping. A predefined kinematic tree and boosted body part detectors effectively solve the self-occlusion issues to a great extent. As we have explained before, we avoid having dynamic background, illumination changes, or moving cameras during action video recording to avoid having to deal with the related action recognition challenges, as that is not within the scope of this research.

3.3 Challenges in Motion Graph

The motion graph technique can generate smooth, controlled animations using a database of 3D motion clips. The 3D motion clip database usually has hundreds of thousands of motion capture frames to have a truly diversified set of actions for an animated character. One of the main challenges in motion graph is the enormous computation efforts required in finding candidate transitional points. The similarity calculation needs to be carried out on each pair of the N frames in the database, which in total involves $O(N^2)$ operations. Storage space and lookup time for the transitional points also increases in proportion to the size of the database. Furthermore, different kinds of motions might have different fidelity requirements [13]. Given a fairly complex set of motion activities, manually setting the threshold for each transition point may be overly laborious. How to automate the thresholds and inform this annotation process for setting transition between motion clips is still an active research topic.

While motion graph is capable of creating smooth transitions, it does not always produce realistic animation because of the fact that action transition cannot always be viewed as a mathematical interpolation of two or even multiple motions. A transition motion could exhibit unique body pose traits that are not present in the source and target actions. For example, a turning transition might involve a braking. While we could use the post data-cleaning phase to address this issue through setting complex constraints, how to clearly express the constraints and how to solve them remain a difficult issue. Dynamic motion animation, a specific type of parametric synthesis, has the potential to solve this issue because it incorporates a set of real-world constraints on mathematical models to simulate human motions [39, 40, 41]. However, it would still need the animator to set the parameters for the desired style. Training or hand tuning of such a mathematical model is often non-trivial given the large motion space of human actions and the complexity of human body structure.

3.4 Challenges in Clustering

There are two central issues involved in clustering: assessment of the meaningfulness of a certain cluster structure found in the data and problem of choosing k – the number of clusters – which best categorizes a given data set. Cluster stability is the common approach to provide answers to these two questions. The idea behind this is that a good choice of cluster algorithm and cluster number should produce a rather stable cluster structure when being applied repeatedly on the sample data points [63]. The idea was thoroughly analyzed on k-means clustering. However, in [64] this common belief is challenged with counter examples which show that poor choices for the number of

clusters can still yield cluster stability. In our research, we have used a trial and error experimental method, by comparing the action recognition results, to set the right number of the clusters.

3.5 Challenges in SVM

SVM is a useful technique for data classification. However, sometimes incorrect results are obtained. The choice of kernel function is one of the major challenges. Linear, Gaussian or polynomial kernel is usually the common option but more complicated kernels might be needed depending on the input data structure. Secondly, a reasonable amount of labeled data is often needed for effective SVM training, and these data can be expensive to obtain. Finally, disk space and memory space may be an issue when training large-scale SVM. In our experiments, reasonably satisfactory results were obtained by using Gaussian kernel. Moreover, the distinct nature of the feature vectors extracted from kick-boxing video data used in our experiments allows effective classification without involving a large set of the training data.

3.6 Concluding Remarks on Challenges

As the focus of this research is to improve the process of 3D character animation generation, understanding the challenges faced when using techniques from related fields is essential to enable us to make a careful selection of the best fitting solutions available today and meeting with our goals.

4 Learning Action Style from Video Data and Its Application

Utilizing both 3D motion data and monocular video data to augment the creation of new 3D character animation sequences is the main focus of this thesis. Our research can be divided into two independent parts: action style learning using video data and its application to character motion generation and motion graph augmentation with video data for incorporating action transition style.

In this chapter, we present a detailed description of our framework for action style learning, extraction and its application in motion sequence generation. In [20], Marco Gilles introduced HMM which uses the transition probabilities of Finite State Machines to govern the character's behavior. The machine learning and motion graph components of our framework are built on the work presented in [20]. One major difference between the preceding approaches and the proposed model lies in the fact that learning in our case is done through video data, an easy-to-acquire media type, as opposed to motion capture data used in the above works. The basic idea is to analyze the behavior pattern of a performer by recognizing his/her actions from video sequences. The "Bag of Words" paradigm is used to categorize actions, and the 3D Scale-Invariant Feature Transform (3D SIFT) [7] is chosen as the keypoint feature for its well-known property of invariance to viewpoint, illumination and affine transformations. Figure 1 describes the complete steps involved in our model including extracting the behavior patterns from video frames to generate a new animation sequence.

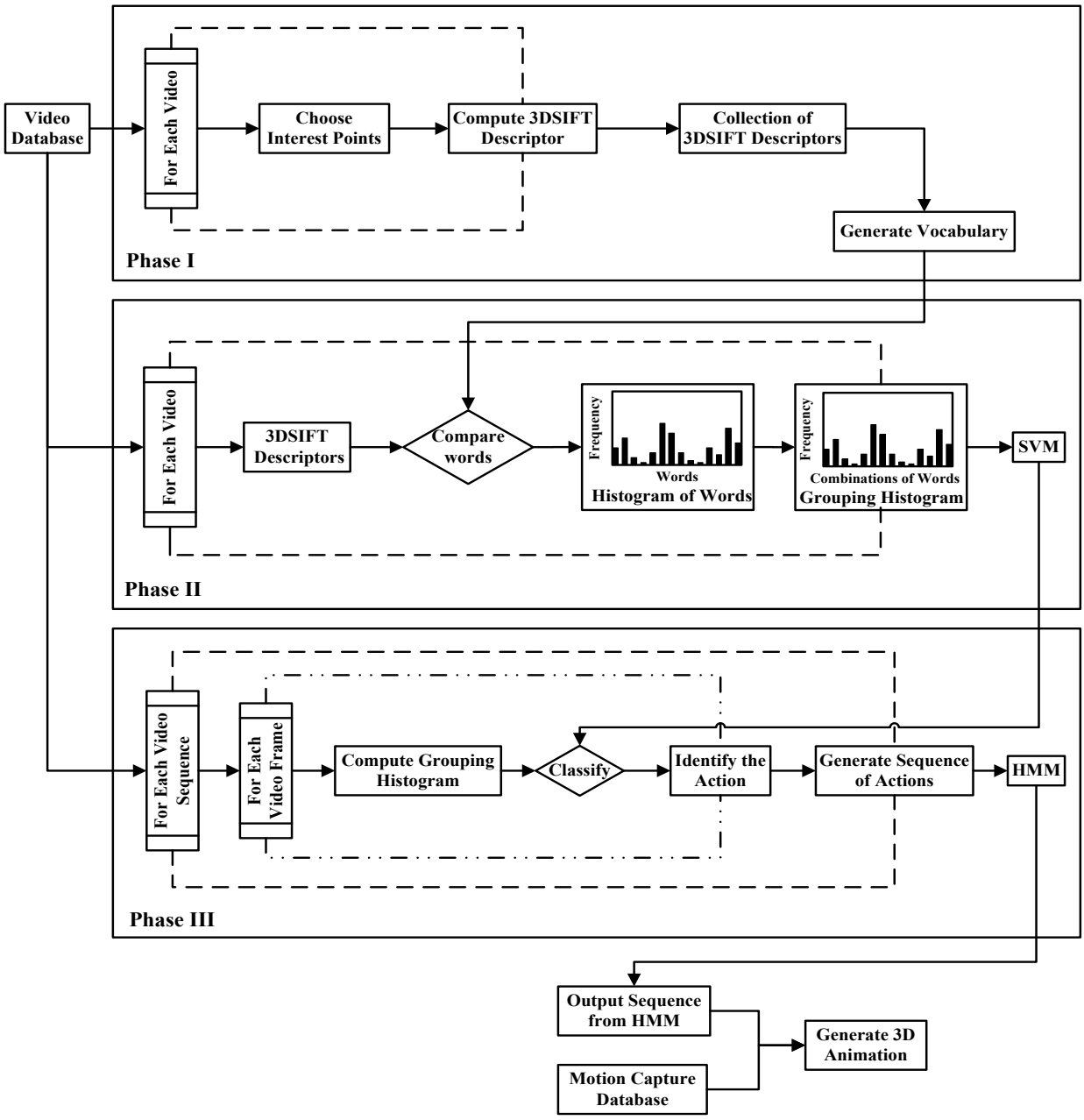


Figure 1: A High Level Action Style Learning Model.

4.1 Feature Word Vocabulary Creation

We start with a repository of training videos. All the actions in the video database have to be specific to the current domain of interest for animation. For example, if this model were to be applied to generate new kick boxing sequences, the action repository should contain different type of kick-boxing actions, such as punching and kicking actions, and different defense poses, etc. If this model were to be applied to a salsa dance motion generation, the videos involved would be different basic actions of salsa dance. This video repository is split into two sets A and B because we would need two different set of video clips in the training and testing phases to avoid biased inference results.

For each video clip in group A, we first have to identify the salient points on each frame. From the perspective of an action, the salient points of a frame are those pixels that have the most significant (largest displacement) movements [15]. Thus, we use optical flow to detect the salient points since it effectively captures the most dynamic movement of the object. It is clear that these active pixels usually lie on the silhouettes of the action frames; therefore, to increase performance and decrease search space, we apply the optical flow algorithm only on the silhouettes of the video frames. A silhouette is extracted by simply doing a subtraction between two adjacent frames. For each frame, the pixels that have the highest optical flow values and are sufficiently apart from each other are chosen as the salient points for further processing. However, the number of salient points chosen for each frame is to be decided empirically. The computation time is proportionate to this number. During our experiments with kick-boxing videos, we chose ten salient points per frame that have the highest optical flow values and are sufficiently apart.

The SIFT descriptor [7] can effectively encode the spatiotemporal information. We compute the 3D SIFT descriptors on each of the salient points and cluster them into a pre-specified number of clusters. Hierarchical k-means, an unsupervised clustering method, is employed. Cluster centers are referred to as "words", while the set of all these cluster centers is referred to as 'vocabulary'. Our kick-boxing experiments have shown that vocabulary size of 1000 yields the best results. Again, the size of the vocabulary is empirical, and it is domain dependent. The following sub-sections describe 3D SIFT descriptor calculation [7] in more detail.

4.1.1 Orientation Assignment

The 2D gradient magnitude and orientation of each pixel are calculated using the equations defined as follows:

$$m_{2D}(x,y) = \sqrt{L_x^2 + L_y^2} \qquad \theta(x,y) = tan^{-1}(\frac{L_y}{Lx}) \tag{1}$$

where L_x and L_y are computed using finite difference approximation $L(x+1,y,t) - L(x-1,y,t)$ and $L(x,y+1,t) - L(x,y-1,t)$. Treating video data as 3D $(x, y \; and \; t)$, the spatial-temporal gradient (L_x, L_y, L_t) can be computed following the same approach, where L_t is approximated by $L(x, y, t+1) - L(x, y, t-1)$. Now the gradient magnitude and orientations can be computed as follows:

$$m_{3D}(x,y) = \sqrt{L_x^2 + L_y^2 + L_t^2} \tag{2}$$

$$\theta(x, y, t) = tan^{-1}(\frac{L_y}{L_x}) \tag{3}$$

$$\emptyset(x, y, t) = tan^{-1}(\frac{L_t}{\sqrt{L_x^2 + L_y^2}}) \tag{4}$$

This means, there will be two values (θ, $\emptyset$) representing the direction of the gradient for each pixel in 3D environment. A weighted histogram, similar to [22], is constructed in the next step. This histogram is the representation of a given interest point taking into account of its surrounding pixels in three dimensions. Figure 2 shows the standard 2D SIFT descriptor on the left. The center image shows the direct application of the 2D SIFT descriptor method in the context of video data. The image on the right shows the 3D SIFT descriptor with its 3D sub-volumes, with each sub-volume accumulated into its own sub-histogram. These histograms together constitute the final descriptor of an interest point in 3D. There are many ways of constructing the weighted histogram. [7] uses meridians and parallels, which are created by dividing θ and $\emptyset$ into equal sized bins.

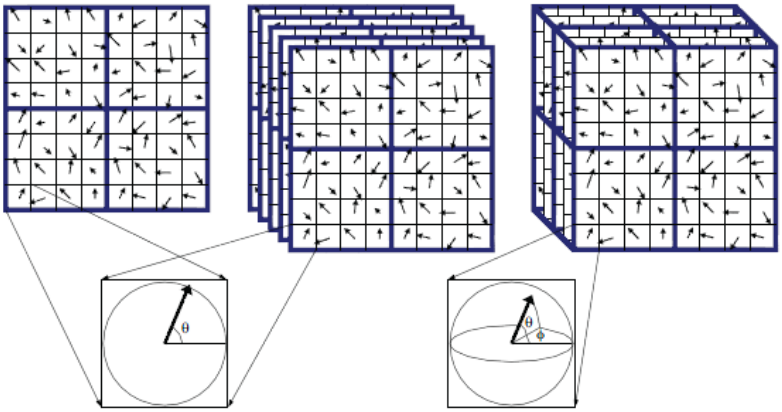


Figure 2: 2D to 3D SIFT descriptors [7]

The normalization of the values added to each of the bins is mandatory to equalize the weight distribution on the orientation histogram. The bins are normalized by their solid angle(ω), which is the area of the bins defined as follows:

$$\omega = \int_{\emptyset}^{\emptyset+\Delta\emptyset} \int_{\theta}^{\theta+\Delta\theta} \sin\theta \, d\theta d\emptyset = \Delta\emptyset \int_{\theta}^{\theta+\Delta\theta} \sin\theta d\theta \tag{5}$$

$$= \Delta\emptyset \, [-\cos\theta]_{\theta}^{\theta+\Delta\theta} = \Delta\emptyset(\cos\theta - \cos(\theta + \Delta\theta))$$

$$hist(i_\theta, i_\emptyset) += \frac{1}{\omega} m_{3D}(x', y', t') e^{\frac{-((x-x')^2+(y-y')^2-(t-t')^2)}{2\sigma^2}} \tag{6}$$

The actual value added to the histogram is shown in Equation 6, where (x, y, t) represents the location of the interest point, and (x', y', t') represents the location of the pixel being added to the orientation histogram. The peaks of the histogram are considered as the dominant orientations. They are stored for use in the following step to rotate the neighboring points around the key points to a unified orientation. This creates rotationally invariant features.

4.1.2 Descriptor Representation

In this step, the SIFT descriptor is computed. We start by calculating the orientation sub-histograms. First of all, the 3D neighborhood surrounding the key point has to be rotated so that the dominant orientation points in the direction of $\theta = \emptyset = 0$. The rotation matrix is shown as the following. This is achieved by taking each neighboring point and multiplying it by the following matrix

$$\begin{bmatrix} \cos\theta\cos\emptyset & -\sin\theta & -\cos\theta\sin\emptyset \\ \sin\theta\cos\emptyset & \cos\theta & -\sin\theta\sin\emptyset \\ \sin\emptyset & 0 & \cos\emptyset \end{bmatrix} \quad (7)$$

The sub-regions surrounding the interest point are sampled and the orientations values in each sub-region are accumulated to form a sub-histogram. What was originally trilinear interpolation becomes now quintalinear (five dimensional) interpolation. The final descriptor of an interest point is obtained by vectorization of the sub-histograms. Figure 3 illustrates the concept. Each of the 8x4 sub-plots represents an orientation bin, and each

gray value in these subplots represents the value of the 2x2x2 sub-histogram (reshaped to 4x2) [7].

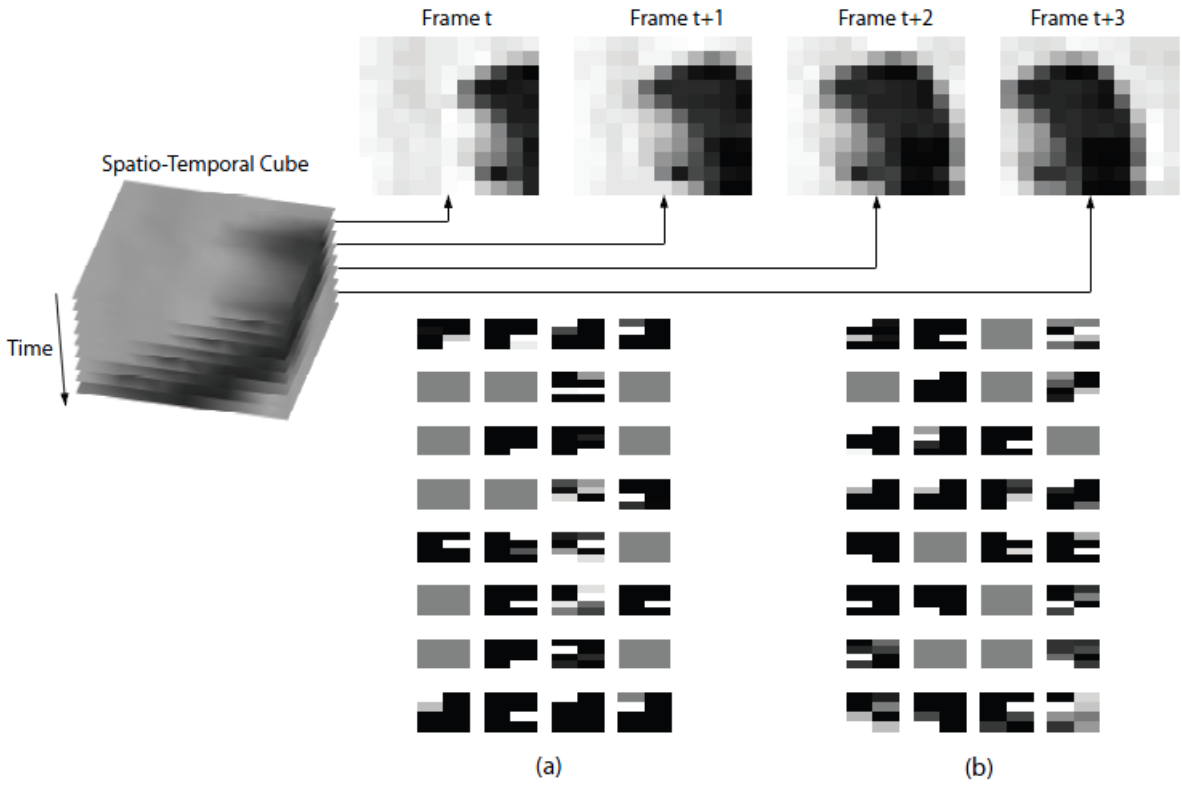


Figure 3: Visualization of the Abbreviated Descriptor [7]. (a) Shows the descriptor before global reorientation by the overall maximum orientation direction. (b) Shows the descriptor after global reorientation.

4.2 Action Classifier Training

After the vocabulary has been built, we compute the 3D SIFT features of the salient points from the video data for a different training set B. Experiments show that better frame-wise action recognition accuracy can be achieved by involving the adjacent frames

into the classification. For example, say, the frame x of video m is in question for action classification, then the 3D SIFT features on the salient points of frame $n-2$, $n-1$, n, $n+1$, and n+2 are grouped together, and the results are matched against the action vocabulary to compute the histograms. These histograms basically show the frequency of the words appearing in the current frame and the adjacent ones. Rather than using these histograms directly as the feature vector of the action, they are used to populate co-occurrence histograms. The observation is that, if two words have similar contextual distributions for a particular action that means these two words are capturing something similar and therefore related to each other. If we can discover such co-occurrences among the words, we can build more discriminative representations of the given action video frames. To quantize this, the correlation between the distribution vectors of any two words is computed [7]. If the value is above a certain threshold, then we consider two words are contextual related, the corresponding frequency counts of the two words are added together. The result of this process is a feature grouping histogram. This, after using Principle Component Analysis (PCA) to reduce dimension, serves as the feature vector of the current frame n.

These feature vectors, with the same action labels as their training videos, are used in SVM learning to train a representation model for each action. The "one against one" and the "one against all" are the two most popular strategies for multi-class SVM. In our model, we first compare these feature vectors against the rest of feature vector, to build a set of one against all SVM classifiers. One against one SVM classier is also trained to supplement the one against all SVM classifier in case they output multiple positive

results. For example, when one against all SVM classifiers for action A and one against all SVM classifier for action B both yield positive results for an action frame, one against one SVM classifier for action A versus action B is then used to further distinguish the actions.

4.3 Frame-wise Action Recognition

In this step, the feature vector is computed for every frame in the testing sequence using the same operations that were performed in the training phase. The feature vector is then classified to an action category using the SVM classifiers built in the previous step. For a sequence of video frames, action classification is executed and each video frame is automatically labeled with the detected action. For example, given a video of 300 frames, the classification process will be executed 300 times and the recognition process labels each frame with an action. A windowing operation is then performed on this numerical output vector to reduce the classification error by updating and/or neglecting the misclassified frames. The idea behind is that the actions of the neighboring frames should be consistent and any abrupt action changes are considered as classification errors to be either corrected or neglected in the later steps. The choice of the window length is experimental and varies by application domain. Figure 4 depicts the frame-wise action recognition process, where each frame in $(x, x+1, x+2, \ldots, x+n-1, x+n)$ is labeled as , b, c, etc.

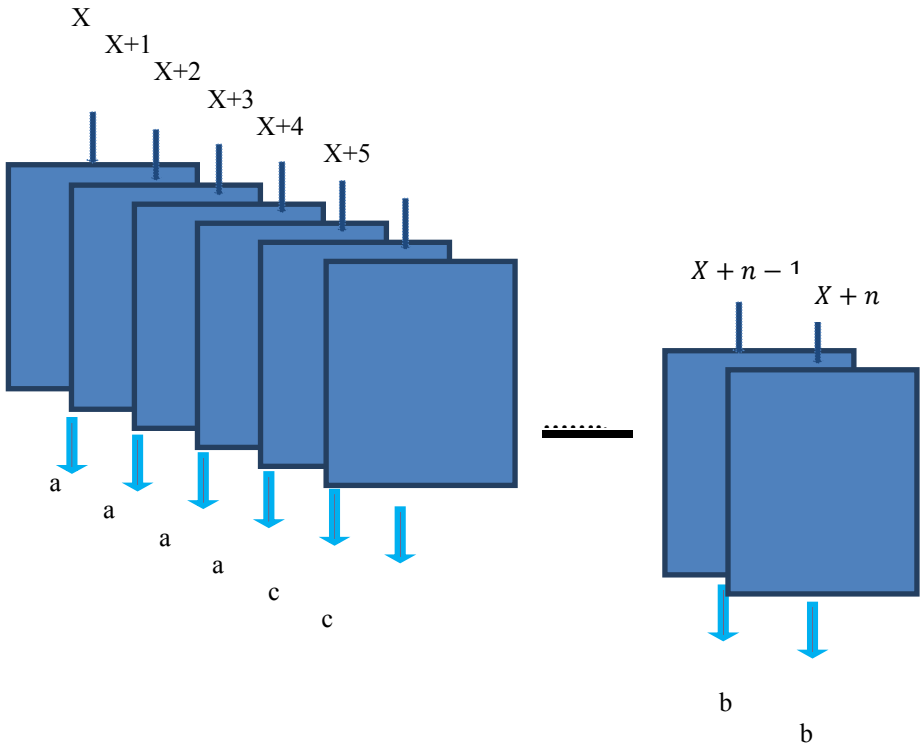


Figure 4: Frame-wise Action detection

4.4 Capturing the Behavior Pattern With HMM

After categorizing the actions performed in the video sequence, we are left with a list of numerous labels (See Figure 4). The next step is to extract and capture the behavior pattern from this labeled list. Our model is inspired by [20]. The control of our characters is based on Finite State Machine (FSM) because the structure of FSM states and the learned probabilities of transitions can embed the behavior pattern of the character quite well. The FSM consists of a set of states $X = \{x_0 \ldots x_m\}$ that the character could possibly be in. The exact meaning of the states is application dependent. There is also as a set of

events that the character can respond to $Y = \{y_0 \ldots y_m\}$. The events correspond to things happening in the environment. Hidden Markov model (HMM) is employed to learn the transition probability between the states in the FSM. HMM is a variant of probabilistic Finite State Machines (FSMs). It is a statistical Markov model in which the system being modeled is assumed to be a Markov process with unobserved (hidden) states [16]. HMM is especially known for its application in temporal pattern recognition such as speech, handwriting, gesture recognition, etc. HMMs are generative models, in which the joint distribution of observations and hidden states, or equivalently both the prior distribution of hidden states (the transition probabilities) and conditional distribution of the observation given states (the emission probabilities), is modeled [16].

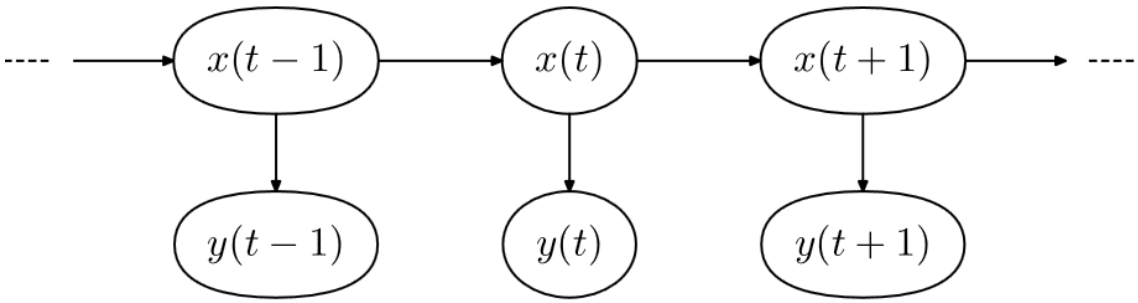


Figure 5: HMM [16]

Figure 5 illustrates that, for a sequence of state transitions, a given state is dependent only on the previous state. $x(t)$ represents the state and $y(t)$ the observation. In our model, we are not concerned with the emission probabilities as we match each state with one single action. We are also only interested in the sequence that was carried out by the

performer spontaneously rather than compulsive reactions driven by an event. Therefore, there is only one probability distribution that requires to be learned in our FSM: the conditional probability of the current state given the previous state (called the transition probability). The Expectation-Maximization (EM) algorithm is used as it is the standard method of learning HMM, and Markov Chain Monte Carlo sampling is used to estimate the expected value of the state variable. During the new motion generation step, we provide an initial state of the HMM, and the state transitions are automatically carried out based on the transitional probabilities of the learned HMM model. This process continues until a user defined length of action is generated. The result is the vector of action labels that faithfully follow the behavior pattern of the performance in the testing video.

5 Augmenting Motion Graph with Video Data

Motion capture has been used very often to reproduce the stylistic movements of an individual. However, motion capture by itself only captures previously executed motions and it does not create new motion sequences. Therefore, the motion graph technique was introduced to stitch individual motion capture actions together.

5.1 Motion Graph

Motion graph is a data structure that allows new animations to be generated by re-sequencing existing motion clips [20]. Motion graph can also generate real-time responsive animation because transition motions can be synthesized dynamically at run time. Figure 6 illustrates the motion graph building process. Consider a motion graph built from two initial clips. The initial clip on the top is divided into two smaller clips by inserting a node. The two different initial clips or the different parts of the same initial clip can be joined together by a transition node [13].

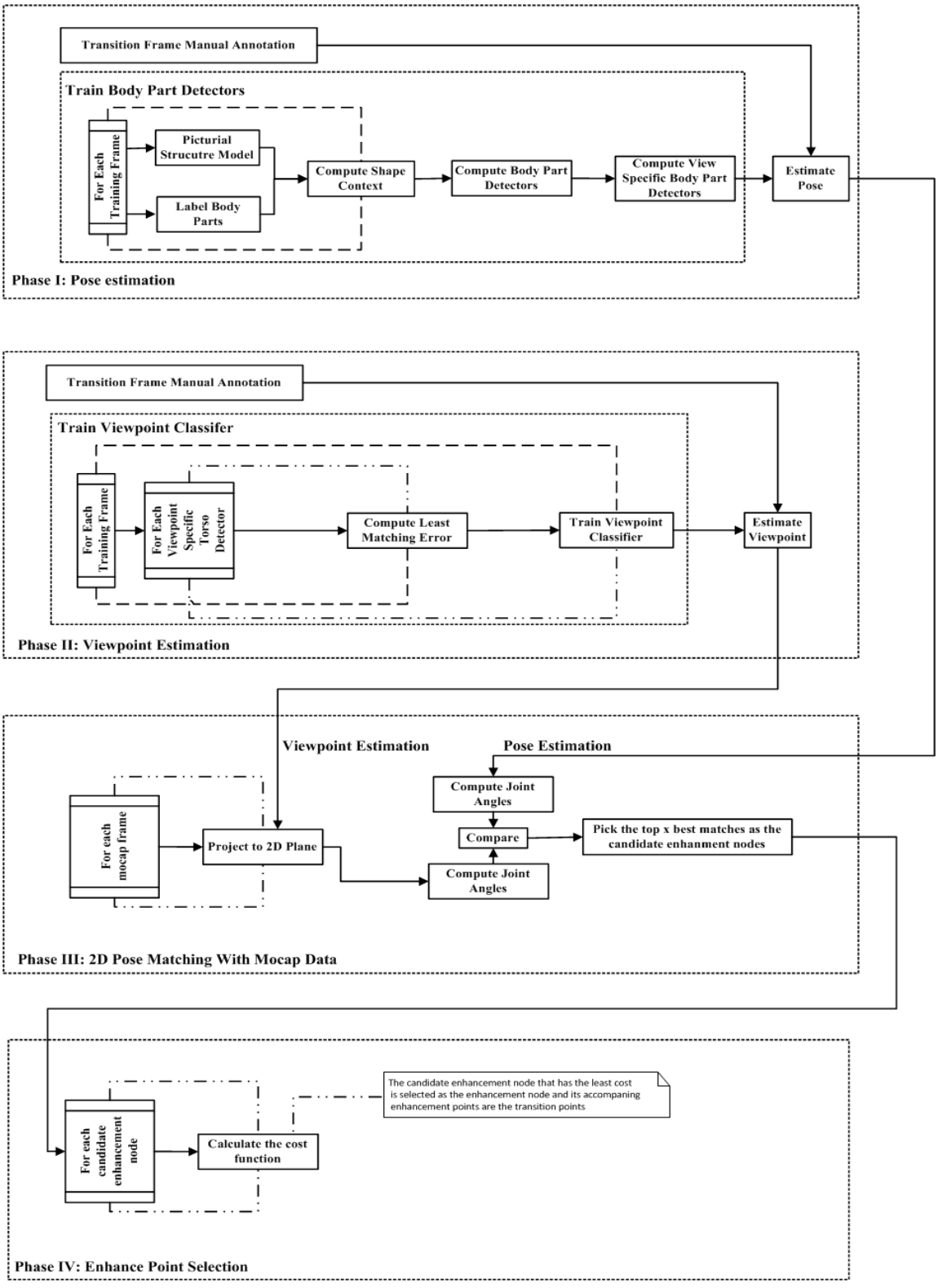


Figure 6: A High Level View on the Method

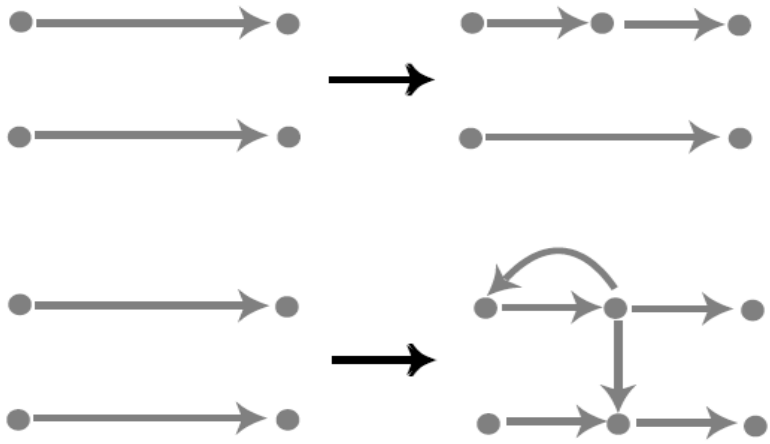


Figure 7: Motion Graph Building Process [13]

Figure 6 gives the complete high level view on our method. Figure 7 illustrates a simple motion graph built from two motion clips. As can be seen from this figure, a motion graph is a directed graph where all edges correspond to motion clips. Nodes serve as potential transition points connecting these clips. Each outgoing edge is potentially the successor to any incoming edge [13]. The arced arrows and vertical arrows between the transition points show the possibilities of transitioning from one point to another. In Figure 8, node 1, 2, 3, 4, and 5 are the transition points of the first motion clip, and node 6, 7, and 8 are the transition points of the second motion clip. In the example graph, the first clip can only transit to the second clip through node 1, and the second clip can transit to the first clip through node 8.

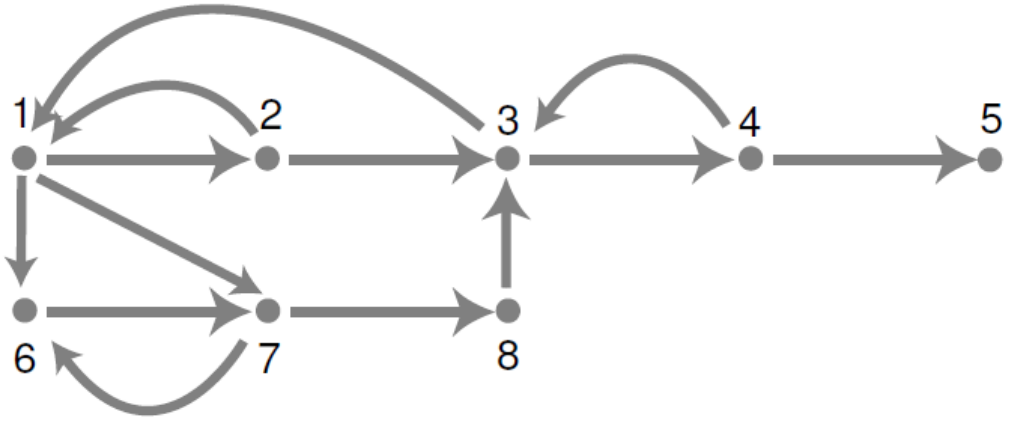


Figure 8: A Simple Motion Graph [13]

5.1.1 Identifying Candidate Transition Nodes

Motion capture data is usually represented with a skeleton hierarchy representation along with the root positions and associating joint rotations information of each frame. We identify the candidate transition points by calculating the distance value between each pair of frames, and the minimal distance is considered as the candidate transition points.

$$\min_{\theta, x_0, z_0} \sum_i w_i || p_i - T_{\theta, x_0, z_0} p_i' ||^2 \tag{8}$$

This optimization has a closed-form solution:

$$\theta = arctan\frac{\sum_i w_i(x_i z_i' - x_i' z_i) - \frac{1}{\sum_i w_i}(\bar{x}\bar{z}' - \bar{x}'\bar{z})}{\sum_i w_i(x_i z_i' + x_i' z_i) - \frac{1}{\sum_i w_i}(\bar{x}\bar{x}' + \bar{z}'\bar{z})} \tag{9}$$

$$x_0 = \frac{1}{\sum_i w_i}(\bar{x} - \bar{x}'\cos\theta - \bar{z}'\sin\theta) \tag{10}$$

$$z_0 = \frac{1}{\sum_i w_i}(\bar{z} + \bar{x}'\cos\theta - \bar{z}'\sin\theta) \tag{11}$$

where $\bar{x} = \sum_i w_i x_i$ and the other barred terms are defined similarly. The distance between two frames is calculated by computing the weighted sum of square distances between the corresponding joint locations p_i and p_i' in the two skeletons, given that an arbitrary rigid 2D transformation may be applied to the skeletons containing p_i and p_i'. T_{θ,x_0,z_0} rotates the point p_i' around the y axis by θ degrees and then translates it by (x_0, z_0). This means the torso rotation and (x, z) translation of a 3D pose should not be taken into consideration when calculating the distance value. The weights w_i can be assigned according to the level of significance a joint has on the whole skeleton hierarchy. For example, torso joint would probably have higher weight value compared to wrist joints because the former affects the final pose more.

When we calculate the distance value for a particular frame, its neighboring frames should be taken into consideration as they are used in the interpolation process. As suggested in [13], the same distance calculation is carried out on the neighboring frames

to ensure a smooth transition. The summation of all the distance values yields the final distance value for the frame in question.

5.1.2 Selection of Transition Points

Using the frame that has the global minimum of the distance value as the only transition point is not ideal. Although it can best ensure the smoothness of the transition, it sacrifices the responsiveness of the motion graph. Therefore, we always consider the local minimum within a window of frames. A simple solution is to only use local minimum below an empirically determined threshold [13]. This threshold is dependent on the type of motions that we try to produce as certain motions would require higher fidelity requirement. Take running action as example, any inaccuracy in a running sequence is likely to be noticed by human eyes because human beings are so familiar with this action. On the other hand, some activities do not require high fidelity. A roundhouse kick following a butterfly kick would probably be a good example because most people do not know how to perform them; neither do they see them often. One important point to note is that the threshold value highly affects the responsiveness and smoothness of the character animation; therefore, it has to be set carefully according to the activity requirements.

5.1.3 Creating Transitions

As shown in Figure 9, when $D(a_i, \beta_j)$ meets the threshold requirement, a transition can be created by inclusively blending frames a_i to $a_{i+k+1}$with frames β_{j-k+1} to β_j. The first step is to apply the appropriate aligning 2D transformation to motion β . Then on frame p

of the transition $(0 \leq p < k)$, we linearly interpolate the root positions and perform spherical linear interpolation on joint rotations.

$$R_p = a(p)R_{a_{i+p}} + [1 - a(p)]R_{\beta_{j-k+1+p}} \tag{10}$$

$$q_p^i = slerp(q_{a_{i+p}}^i + q_{\beta_{j-k+1+p,}}^i a(p)) \tag{11}$$

where R_p is the root position on the p^{th} transition frame and q_p^i is the rotation of the i^{th} joint on the p^{th} transition frame. A blending weight $a(p)$ is chosen to maintain the continuity of the transition according to the conditions that $a(p) = 1$ for $p \leq -1$ and $a(p) = 0$ for $p \geq k$, and $a(p)$ has c^1 continuity everywhere. This requires

$$a(p) = 2(\frac{p+1}{k})^3 + 3(\frac{p+1}{k})^2 + 1, -1 < p < k \tag{12}$$

It is important to note that dynamic constraints imposed on the motion clips may be violated by this linear interpolation. As already mentioned earlier, a post data-cleaning step might be necessary to ensure that the generated transitions also comply with the physical fidelity.

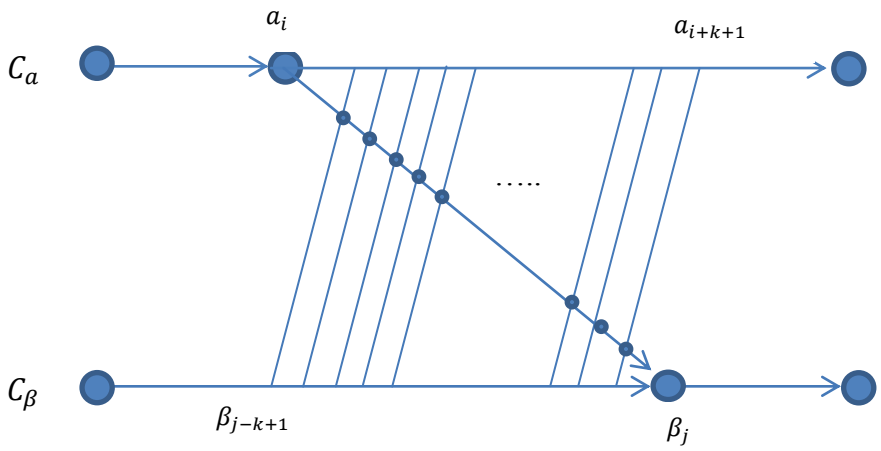


Figure 9: Blending To Create Transition

5.1.4 Pruning the Graph

A motion graph is constructed with motion clips connected to each other. However, there might be nodes that are not part of any cycles. Such nodes have to be pruned from the motion graph because they create dead ends to the motion sequence. Other nodes might have very limited out-bounding connections. They also should be pruned from the motion graph because once this type of a node is entered; the motion is confined to a small part of the database, which will decrease the flexibility of the motion. Such nodes are called sink nodes (See Figure 10). For example, a motion sequence of kickboxing with high diversity needs to ensure that there are multiple choices of actions following a "punching" action. A punching action that can only transit to a kicking action would violate this requirement. Figure 10 illustrates this with some examples - the blue circle indicates the largest strongly connected component in the graph, which is [1, 2, 3, 6, 7, and 8]. Node 4 is a sink and 5 is a dead end.

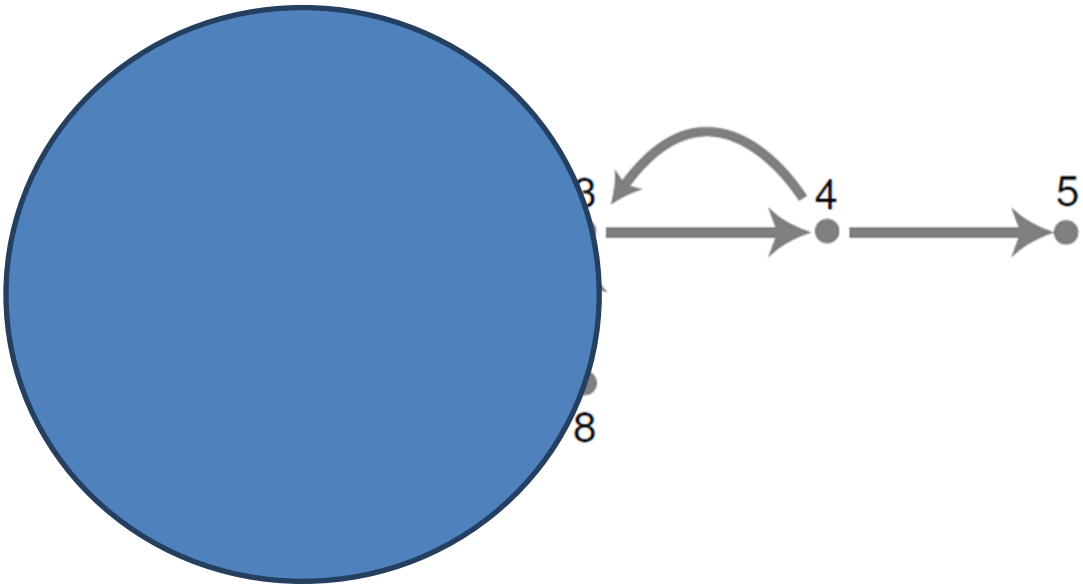


Figure 10: Strongly Connected Component

5.2 Improving Motion Graph Technique Using Video Data

As we have explained before, the standard motion graph technique does not always create realistic transition frames because the transitioning behavior of human beings cannot be always produced by a simple geometric blend of two actions. It has much more than that. A walking to jumping transition will involve stopping and slightly crouching actions. A running to hand flipping transition could involve stopping and braking actions. Normally the animator would need to manually create these transitions. As already mentioned earlier, a physics-based model could be used to solve this problem, but it requires precise specification of parameters (which animators find rather difficult to specify), significant computational power and robust control systems making its application to real-time interactions involving complex activities nearly impossible.

In this thesis, we present a different approach that makes use of the transition frames in video data to allocate the possible ***enhancement nodes*** in the motion database to supplement the blending algorithm in the standard motion graph. This approach creates much more realistic transitions and once all the pre-computations are done, it can be applied in real-time. We call our model AugMG (*i.e.* an augmented motion graph). To illustrate the idea, we use the same example described above. Consider two motions m_1 and m_2, in which we know that the transition between the two would involve poses which cannot be obtained by simply interpolating between the two motion clips. With a large enough 3D motion database, there is a great chance that some or all of the poses in the transition motion exist somewhere in the motion database; however, manually searching for it in a large database is a tedious job. We propose to use video data to help automate the search process and thus considerably reduce human efforts in creating realistic transitions.

Identifying transition frames between actions in a video sequence is difficult to achieve automatically, but rather simple to do manually as human beings are quick in recognizing actions. These transition frames are used for pose-estimation, and the results are later matched with the motion capture database. The best matching frames in the motion capture database are treated as ***reference points***, and they are inserted into the motion graph for use in the transition creation process. As example, we shall use a running to hand flipping transition to illustrate our technique. We assume that some clips containing the braking action exist somewhere in the 3D motion database. An automated process to extract the transition action from thousands of mocap frames is described in detail in the following sections.

5.2.1 Manual Annotation of Transition Frames

While the current advances in action recognition enable us to do frame level action recognition quite well, accurate recognition of the transition frames is still rather difficult, as we mentioned before. By their very nature, transition poses combine traits of multiple actions which create ambiguities during recognition. After careful investigation and many different experiments, we concluded that at this stage automatic detection of transition frames may not be worth pursuing. Therefore, in this step, given a video sequence of the actions from which the individual's style of transitioning between actions is to be embedded into the motion graph, the transition frames will be manually identified (See Figure 11). This does need a little effort from the human animator, but it is much easier than specifying the required physics constraints.

Figure 11: Transition Frame Manual Annotation

5.2.2 2D Pose Estimation from Video Data

Given a transition video frame, we derive the 2D pose of the character in it as follows. First of all, a pictorial structure model is required. Pictorial structures [17] present kinematic restrictions on the body parts to restrict the solution space. Pictorial structures represent human being as a flexible configuration of N different parts Lm $\{l_{mo}, l_{m1}, \ldots, l_{mN}\}$ where m denotes the current frame of the sequence. The state of part i is given by $I_{mi} = \{x_{mi}, y_{mi}, \theta_{mi}, s_{mi}\}$, where x_{mi} and y_{mi} denotes its image position, θ_{mi} the absolute orientation, and s_{mi} the part scale [18]. The posterior probability of the 2D part configuration L_m given the single frame image evidence D_m is given as

$$p(L_m|D_m) \propto p(D_m|L_m)\, P(L_m) \tag{13}$$

The body configuration $p(L_m)$ has a tree structure and represents the kinematic dependencies between body parts. It factorizes into a unary term for the root part (here, the torso) and pairwise terms along the kinematic chains:

$$p(L_m) = p(l_{m0}) \prod_{(i,j)\epsilon K} p(l_{mi}|\, l_{mj}) \tag{14}$$

where K is the set of edges representing kinematic relationships between parts. $p(l_{m0})$ is assumed to be uniform and the pairwise terms are taken to be Gaussian in the transformed space of the joints between the adjacent parts. The likelihood term is assumed to factorize into a product of individual part likelihoods presented in Eq. (14).

$$p(D_m|\, L_m) = \prod_{i=0}^{N} p(d_{mi}|\, l_{mi}) \tag{15}$$

$$p(L|\, M) \propto \; p(l_0) \prod_{i=0}^{N} p(d_i|\, l_i) \cdot \prod_{(i,j)\in E}^{N} p(l_i|\, l_j) \tag{16}$$

This simplifying assumption is valid as long as occlusions are not significant; nevertheless, it enables an efficient and accurate inference and leads to competitive experimental results. The posterior over the configuration of parts factorize as formulated by Eq. (15).

To define the part likelihood, we rely on the boosted part detectors from [19] with the shape context initially introduced in [2] as the feature descriptor. As mentioned before, our method only considers the major body parts: left/right lower and upper legs, torso, head and left/right upper arms and forearms, as shown in Figure 12. Detectors are trained for each of these body parts.

According to [18], viewpoint specific body part detectors can significantly improve the performance of pose estimation. Figure 13 shows the experimental results of pose estimation using generic body part detectors (a) and viewpoint specific body part detectors (b). Notice that the viewpoint specific body part detectors clearly outperform the generic body part detectors. Both viewpoint specific and generic body part detectors are trained at this step for use in the next steps of our method.

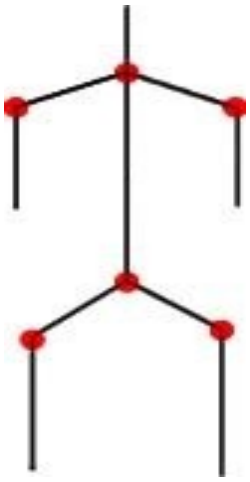
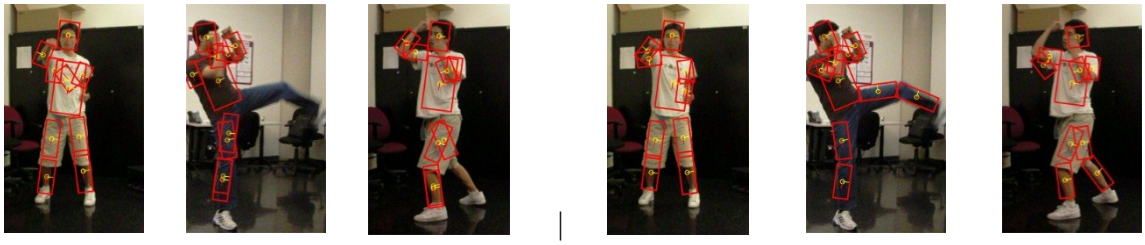


Figure 12: Body Model



(a) (b)

Figure 13: Pose Estimation Example Results

5.2.3 Viewpoint estimation

Knowing the viewpoint of the actor is essential later for 3D pose matching. It will allow the use of viewpoint specific body part detectors to achieve better performance in 2D pose estimation. During the training phase, we annotate the viewpoints for each training

video and build body part detectors accordingly with viewpoint specific kinematic priors. Now, we run over a given video frame using all the viewpoint specific torso detectors, and this yields a multi-dimensional vector representing the least matching errors in finding the torso in different viewpoints. We train the one against all and one against one SVM classifiers using these vector outputs as features. During the testing phase, these SVM classifiers are employed to identify the view points of the video frames in question.

5.2.3.1 Matching 2D Pose Information with 3D Motion Data

Recovering 3D joint points from 2D is an active research field [42, 43, 44]. The complexity involved in the process is built-in due to the loss of 3D information in the monocular projection. Body part self-occlusions and difficulties in limb segmentation increase the ambiguities. The exemplar-based approaches are usually limited by the number of exemplars that are created before-hand [43], and they require further validation in more dynamic environments.

We propose a simple and effective method at the cost of a little estimation accuracy. Firstly, all the motion capture frames are projected to a 2D plane, according to the viewpoint detected in the previous step. This projection is necessary to bring the 3D motion capture data and 2D pose data into the same dimension space. Figure 14 depicts this step with a running action as example. The 3D motion capture data (red skeleton) is projected to a 2D plane (green skeleton) based on the viewpoint estimation (front view in this example). The gray shape is the result of 2D pose estimation. Now the comparison between the 2D pose estimation and the 3D motion capture data is possible. This averts the complexity and ambiguities involved in the processes of directly deriving 3D joint

information from 2D pose estimation results. With each joint location and orientation known, we calculate the angle information between the connecting joints. The red dots in Figure 12 indicate all the joints in our 10 body part model. Using the joint angle information instead of Euclidean distance of joint position avoids the issues of scaling and length of limbs, typically involved in motion data matching.

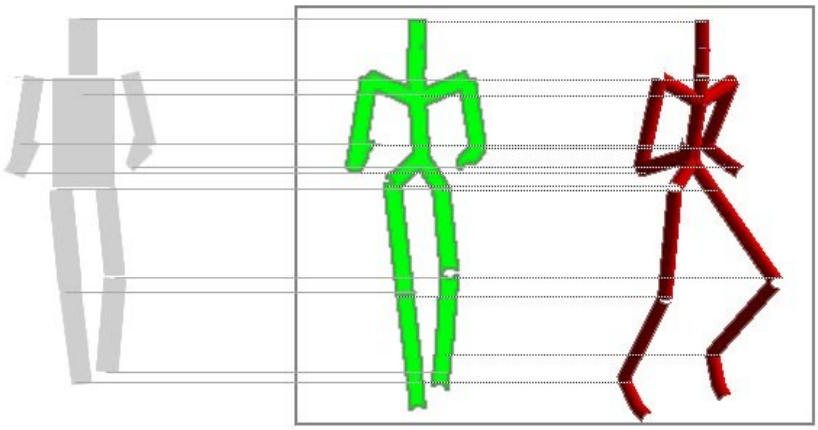


Figure 14: 2D to 3D mapping

5.2.4 Transition frame identification

The 2D pose estimation and 3D motion data mapping usually lead to multiple matches, including some false positives, especially given a large motion database. It is mainly due to the information loss during the 3D to 2D projection. We refer to these matches as **candidate reference points**. It is essential to remove the false positives to ensure a natural transition. Figure 15 illustrates our algorithm. Equation 16 describes our distance function in details. With C_a and C_β as the two actions in question and frame ω_{z_1} as one of

the 3D poses matching with the 2D pose estimation, we compute the distance value for every pair of frames between C_a and C_ω as well as every pair of frames between C_ω and C_β within a certain window size k. The use of frame window effectively incorporates derivative information into the metric. We use the distance function and the close-form solution described in [13] with extra constraints to locate the candidate reference points. The range values x and y are considered to be the responsiveness measure. The larger x and y are, the slower the character transitions between two actions. Therefore, our similarity metric consists of the distance measures along with x and y values. The matching point $\boldsymbol{\omega_{z_1}}$ with the minimal distance value and (x, y) value is considered the **reference point**. The nearest transition points $\boldsymbol{\omega_{z_0}}$ and $\boldsymbol{\omega_{z_2}}$ are considered as the **enhancement nodes**.

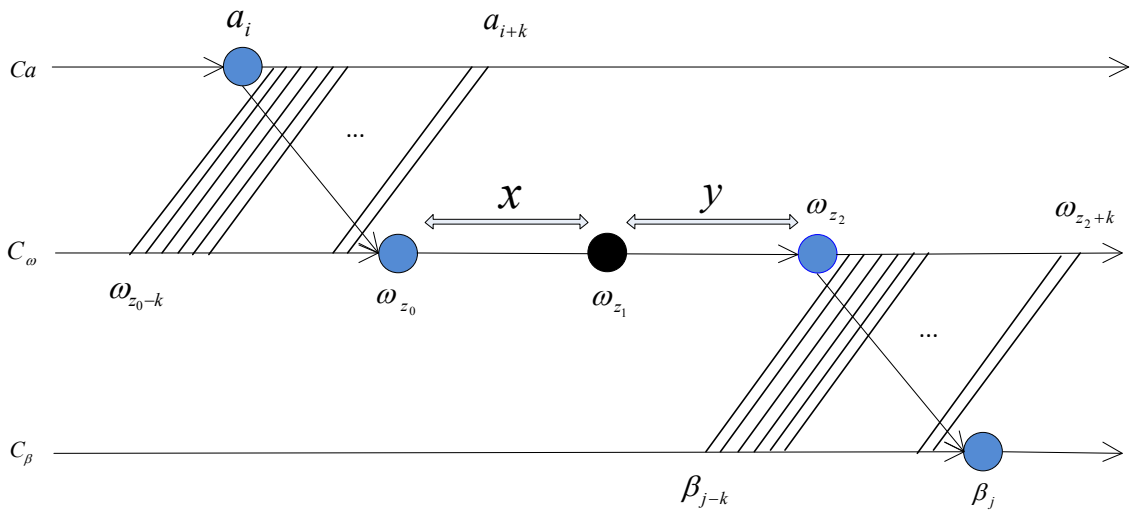


Figure 15: The Enhanced Distance Function

The distance function and close-form solution [13] are given in Equations 9, 10, 11.

5.2.5 Incorporating the extra transition node

The first half of the transition clip is created by blending frames $\boldsymbol{\alpha_i}$ to $\boldsymbol{\alpha_{i+k}}$ with ω_{z-k} to ω_z. Frame ω_z to ω_{z+k} with β_{j-k} to β_j are blended together to form the second half of the transition clip. Data cleaning may be required as explained in Motion Graph [13] because the constraints in the original motions might be violated due to the simple interpolation that is done.

Moreover, noise could be introduced due to the incorrect pose estimation and view point detection. This could result in undesired transitions. These types of unnatural transitions are easy to be identified by human eyes. During the offline creation of AugMG, animators can simply discard the problem-causing enhancement nodes. As the whole process does not require extensive human efforts, a complete rerun can be conducted with the animator selecting a new transition frame from video, to obtain a more realistic and smoother transition.

6 Implementation

We have implemented all the techniques discussed in earlier chapters and have conducted a number of experiments using this implementation. Description of these experiments, the results, and analysis of results follows.

6.1 Learning Action Style from Video and Its Application

In order to demonstrate how our first method supports learning of high level motion behavior from video, we have chosen the example of a performer doing kick-boxing actions in a sequence that is characteristic of his/her individual style.

6.1.1 Preparing a video action database

A repertoire of video clips containing basic actions was created for action recognition purposes. Our video database contains 240 video sequences consisting of six basic kickboxing actions - Jab, Hoop, Uppercut (UC), Defense (Def), Lower Kick (LK), and Roundhouse Kick (RK) recorded for 10 actors from 3 different views (front view and two side views). Their lengths are between 15 to 20 seconds. All the videos are taken with a stationary camera; however, backgrounds as well as actors' clothes/positions/orientations may differ.

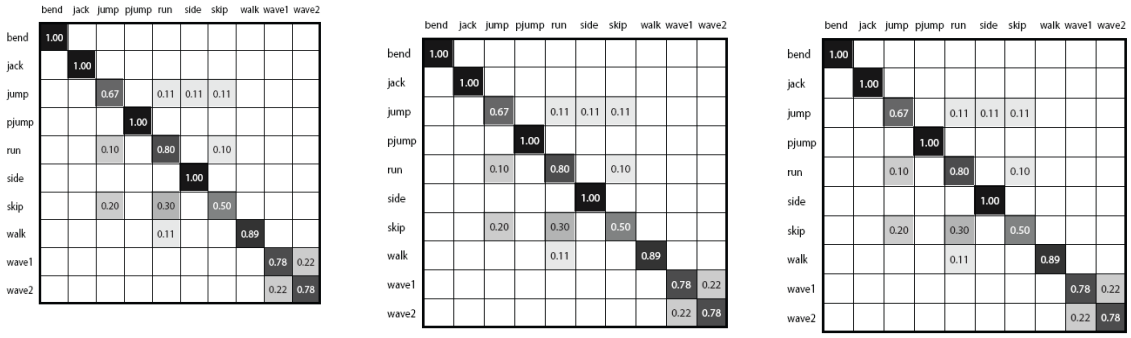
Figure 16: Sample Frames of the Kickboxing Video Database

The videos are down-sampled to spatial resolution of 160X120 pixels and a frame rate of 20fps. The primary reason for this choice of spatial resolution and frame rate is to reduce the computational cost, and to minimize training and learning time. Note that the spatial resolution and the frame rate are proportional to the complexity of the action performed. Figure 16 shows sample frames from the kickboxing video database.

6.1.1 Action Vocabulary Construction

This phase takes an input of a set of training videos from the video database. First and foremost step in this phase involves selection of salient points. As mentioned earlier, we implemented the optical flow algorithm on the silhouette extracted from the video to select the salient points. Silhouette is extracted by simple subtraction of next and previous frames. For each frame, the top ten pixels that have the highest optical flow values were chosen as the salient points. 3D SIFT descriptors were computed on these interest points to be used as feature vectors. This process was repeated for all the videos to generate a

collection of 3D SIFT descriptors. These 3D SIFT descriptors were then quantized by clustering into a pre-specified number of clusters. The cluster centers are referred to as ‘words’, while the collection of these cluster centers is referred to as the ‘vocabulary’. Table 1 shows the three combinations that we tried for varying number of cluster size: 500 (a), 1000 (b), and 1500 (c). As one can see from the confusion matrices, increasing cluster numbers does not necessarily increase the accuracy. In this experiment, we set the number of clusters at 1000 as it yields a better recognition result for kickboxing actions.



(a) (b) (c)

Table 1: Confusion Matrices

6.1.2 Action Classifiers

This next phase receives a different set of training videos from the video database. The word frequency histograms were first built; they are also referred to as signatures. Figure 17 shows a typical signature for a video frame. The grouping histogram is shown in Figure 18. One against one and one against all SVM classifiers were trained using the grouping histograms. For this we have used the Bioinformatics Toolbox from Matlab.

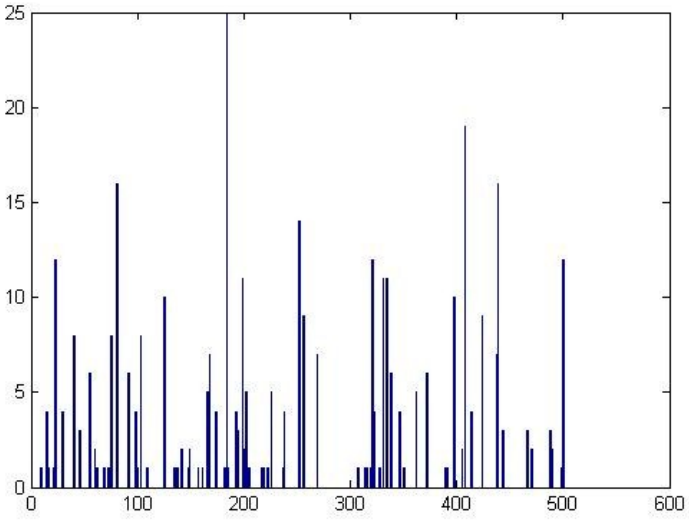


Figure 17: Sample signature of a video frame

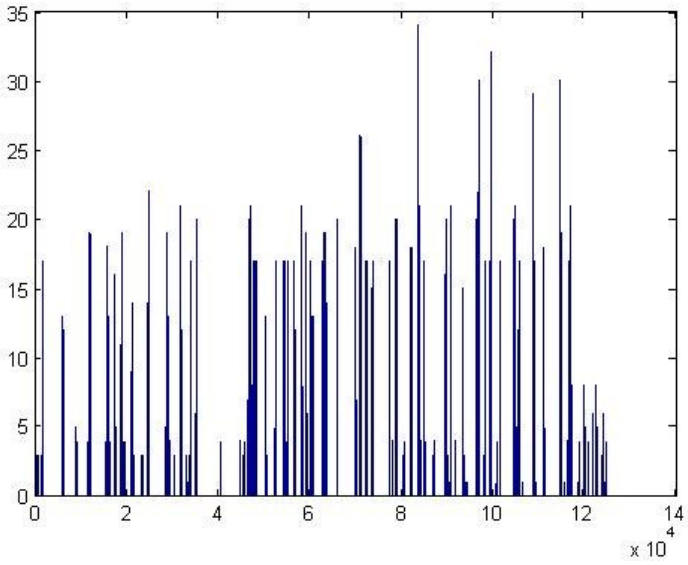


Figure 18: Grouping Histogram

6.1.3 Action Recognition

In the testing phase, we again construct a co-occurrence histogram as the feature descriptor for each frame in the testing video sequence. We used the trained SVM classifiers to identify the action category of the frames in question. The accuracy of SVM classifiers at frame level action recognition was found to be 64% and confusion matrix for the same is shown in Table 2 (a). A windowing operation was then performed with a window size of 6 to remove noise and improve accuracy as average action length was found to be 6-10 frames in our video database. The selection of the window size depends on frame rate of the video and also complexity of the action being performed. This operation increases the overall accuracy to 70% and the confusion matrix obtained after windowing operation is illustrated in Table 2 (b). This somewhat low accuracy of frame level action detection does not affect the process of pattern learning with HMM greatly, as only the actions classified with certainty are fed to the HMM. Figure 19 shows the action recognition output result of the proposed framework for a sample video sequence where (a), (b), (c), (d), (e) are representative frames of the kickboxing action videos.

Figure 19: Frame Level Action Recognition Results

	Jab	UC	Hoop	Def	RK	LK
Jab	0.85	0	0.03	0	0.07	0.05
UC	0.17	0.56	0.15	0	0	0.12
Hoop	0	0.12	0.62	0	0.26	0
Def	0.06	0	0	0.66	0.06	0.22
RK	0.04	0.16	0	0.20	0.6	0
LK	0.04	0.21	0	0.20	0	0.55

(a)

	Jab	UC	Hoop	Def	RK	LK
Jab	0.90	0	0.02	0	0.04	0.04
UC	0.12	0.66	0.11	0	0	0.11
Hoop	0	0.12	0.70	0	0.18	0
Def	0.06	0	0	0.71	0.05	0.18
RK	0.04	0.14	0	0.20	0.62	0
LK	0.04	0.20	0	0.17	0	0.59

(b)

Table 2: Confusion Matrices for Frame-wise Action Recognition Accuracy

6.1.4 HMM Training and Action Sequence Generation

The implementation of HMM is also in Matlab, and it supports inference and learning of HMMs with discrete outputs. The typical input sequence fed into the HMM for it to learn the action sequence pattern is presented in Table 3 where C1, C2, C3, C4, C5, C6, C7 represent different actions in the sequence. The output sequence generated by the trained HMM is presented in Table 4 with C4 as the starting state. As can be seen from the data in Table 4, the output sequence generated by HMM reflects very well the pattern of the input sequence.

C2|C1|C3|C1|C1|C2|C2|C2|C4|C4|C4|C4|C5|C4|C4|C4|C4|C5|C5|C3|C4|C4|C3|C3|C5|C7| C6|C7|C7

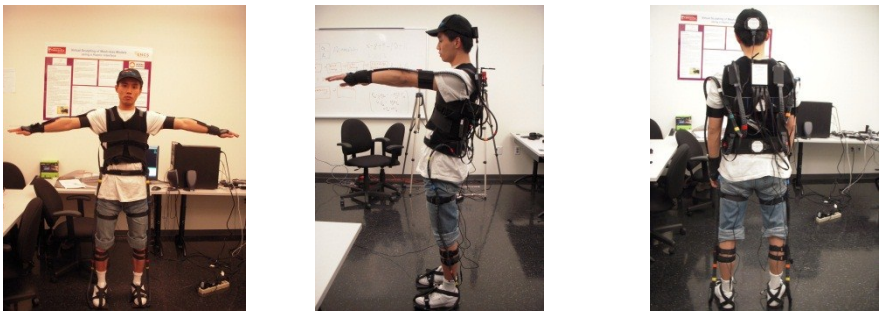
Table 3: A Sample Training Sequence Fed to HMM

C4|C3|C4|C4|C3|C5|C4|C3|C5|C4|C3|C5|C4|C4|C3|C5|C4|C4|C4|C4|C4|C5|C4|C5|C3|C4| C5|C5|C4|C4|C4|C4|C3|C4|C5|C4|C4|C4

Table 4: An Example Sequence Output from HMM

6.1.5 Creating New Motion Sequences

To demonstrate the new sequences generated by HMM, we created a repertoire of mocap clips for individual actions of a performer using Measureand Inc's Shapewrap system [26]. The repertoire consists of the aforementioned basis kick boxing actions: Jab, Hoop, Uppercut (UC), Defense (Def), Lower Kick (LK), and Roundhouse Kick (RK). We use the motion graph [13] techniques to connect the motion together to create a new kick-boxing sequence embedded with the high level behavior pattern of the performer. This resulted into a novel kickboxing motion sequence following the behavior pattern of the actor in the video sequence. Figure 20 shows the subject with the shape wrap system in (a) front view, (b) side view, and (c) back view. Figure 21 shows a complete synthesis example with (a) a video sequence input (b) action recognized, (b) a novel 3D motion sequence generated following the behavior pattern.



(a) (b) (c)

Figure 20: Subject Wearing the Shapewrap System

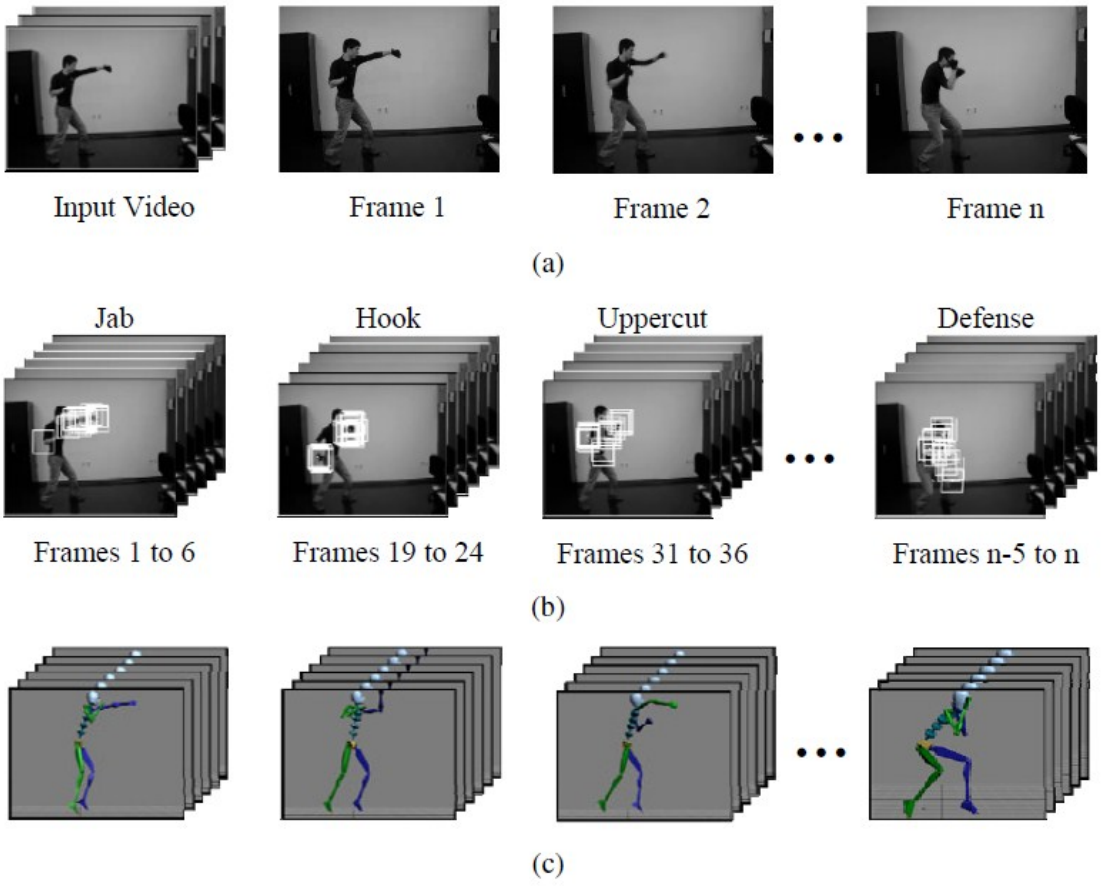


Figure 21: Synthesis Results From a Sample Video Sequence

6.2 Augmenting the Motion Capture with Video Data

We implemented the standard motion graph algorithm as well our method AugMG. We applied both methods to generate the transition from running to hand flipping actions. The results clearly show the advantage of our method in creating more realistic transitions.

6.2.1 Motion Capture Action Database

We used the online motion capture database provided by CMU Graphics Lab for obtaining the basic 3D actions. As the BVH file seems to be a popular format for motion capture data, we use the BVH motion clips provided in the database. One major reason for using this well-recognized motion capture database is to provide the common ground for comparing with other available solutions. Also, it has a wide repertoire of motion captured action clips.

6.2.2 Motion Graph Implementation

For comparison purpose, the complete motion graph solution, including distance calculation, transition nodes identification, and transition frames blending, was implemented in C++. As example, we have chosen the transition from running action to hand flipping action. The transition generated using the standard motion graph algorithm is shown below in Figure 22, where the transition frames between the two actions generated by using original motion graph algorithm. Although the transition is quite

smooth, it is not realistic because people usually have to slow down from running, or even come to a complete stop before performing the hand flipping motion.

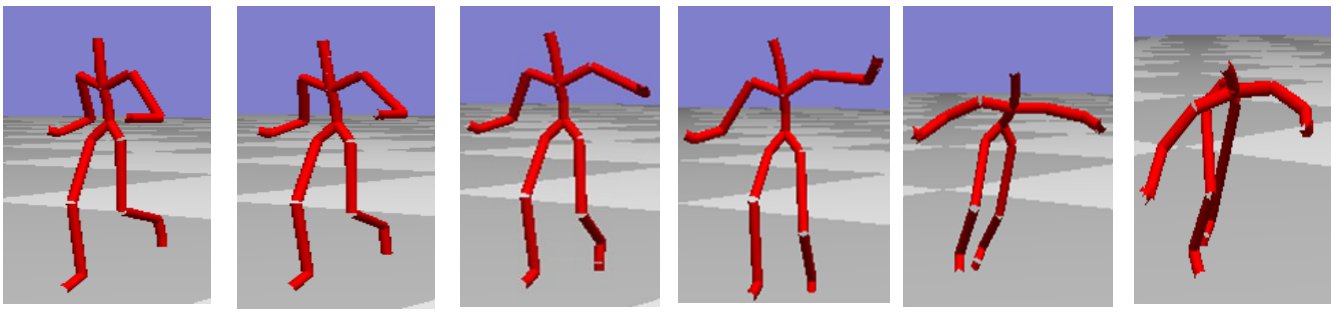


Figure 22: Transition Created In Standard Motion Graph

6.2.3 Pose Estimation from Video Data

The 2D pose estimation code we have used is the one was provided in [19]. For demonstration purposes, we trained the body part detectors for front, left, right, and generic viewpoints. The 2D positions and orientations of the body parts detected in a video frame are shown in Figure 23. This is the result of performing pose estimation to one of the side-punch frames in the previously mentioned kick-boxing video database, using front view specific body part detectors. The complete detection accuracy ratio is shown in Table 5.

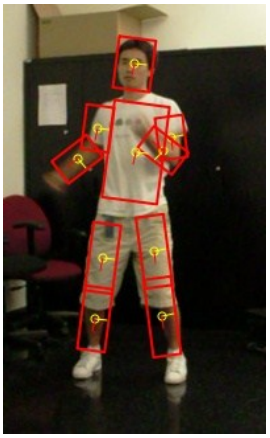


Figure 23: Sample Pose Estimation Result

	LLL	LUL	RUL	RLL	LFA	LUA	RUA	RFA	Torso	Head
Front	0.90	1	0.90	0.81	0.81	0.90	0.90	0.57	1	0.95
Left	0.82	0.82	0.77	0.73	0.82	0.86	0.95	0.91	1	0.95
Right	0.76	0.88	0.94	0.94	0.76	0.88	0.82	0.88	1	1
Generic	0.77	0.89	0.67	0.44	0.44	0.44	0.89	0.67	1	1

Table 5: Body Part Detection Accuracy Table

6.2.4 Character Viewpoint Estimation in Video Frame

Table 5 clearly shows that viewpoint specific body part detectors greatly improve the accuracy of the detection, compared to generic body part detectors. Viewpoint information is also crucial for 2D to 3D data matching. Therefore, our next step is viewpoint detection.

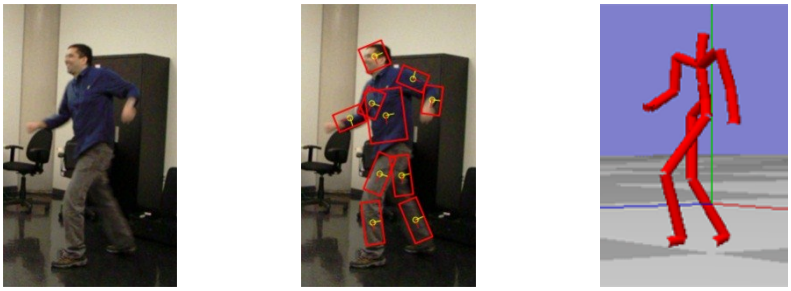
We implemented the viewpoint estimation based on the methods proposed in [18]. The result of the viewpoint estimation is presented in Table 6. As can be seen from this table, we obtain reasonable high accuracy.

	Front	**Left**	**Right**
Front	0.82	0.09	0.09
Left	0.045	0.91	0.045
Right	0.14	0.14	0.72

Table 6: Confusion Matrix for Viewpoint Detection

6.2.5 Mapping 2D Body Part Data to 3D Joint Data

In this step, the 3D joint data was projected into a 2D plane, in accordance to the viewpoint information detected in the previous step. Figure 24 illustrates the complete 2D to 3D matching process with (a) as the transition video frame, (b) as the pose estimation result with the body part position and orientation indicated with red triangle boxes, and (c) as the 3D matching pose.



(a) (b) (c)

Figure 24: A Complete 2D to 3D Matching Process

6.2.6 Incorporating the New Transition Node into the Standard Motion Graph

The following figure shows the motion sequences created using AugMG model, where a braking reference point was inserted into the existing motion graph to create a realistic transition between a running and hand-flipping actions.

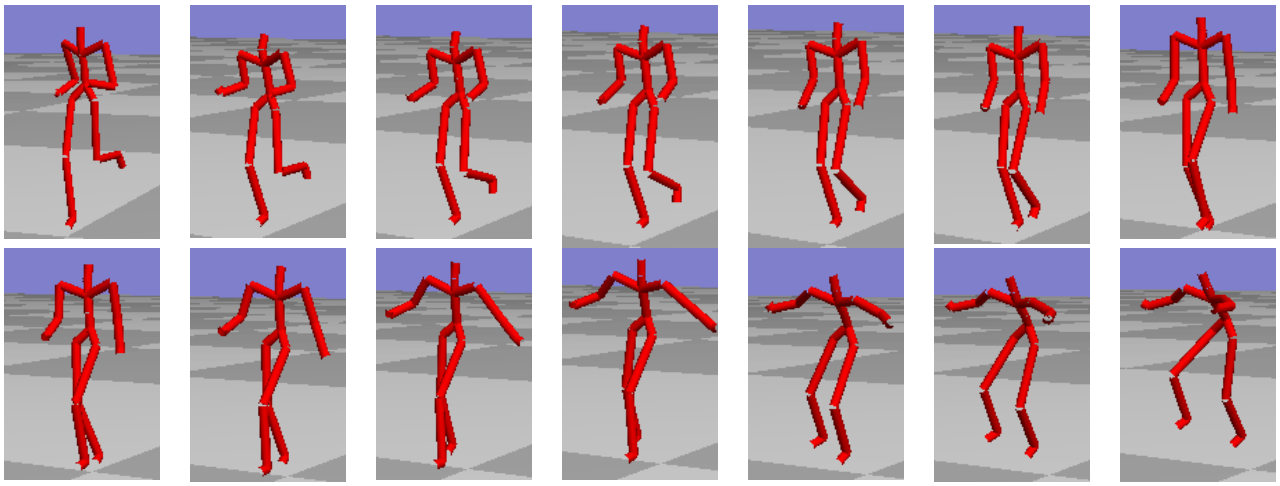


Figure 25: Augmented Motion Graph Output

6.3 Conclusion

The experimental results shown in this chapter provide substantial evidence of the applicability of our methods in generating new motion sequences by learning action styles from video and for generating more realistic transition between actions with significantly reduced human efforts.

7 Conclusions

In this thesis, we have proposed two new methods in order to augment the creation of new articulated character animation sequences by learning from video data.

The first method focuses on learning high-level human styles from video of human actors performing activities and then embedding this style into 3D game characters. This has been achieved by recognizing the action sequence by training SVM classifiers and then suitably training an HMM for synthesizing new action sequences according to that style. Its demonstration on kickboxing action sequences confirms its usability. In the current implementation, our framework has training and learning phases that typically last for a few hours. Training and learning duration varies in proportion to the complexity of the actions. However, it must be noted that these are the preprocessing tasks. A new video action sequence can then be synthesized in real time.

Creating very large volumes of 3D motion data for the learning phases is expensive. In our framework by using video data to learn, this problem is alleviated. Video is non-intrusive, and so much easier to capture these days. Our choice of using salient points on the silhouette works considerably very well for action recognition, as the SVM classifier for recognizing actions has yielded a satisfactory accuracy of about 90% at the video level and of about 70% at individual frame level. 70% accuracy rate at the frame level means that the generated action sequences could have occasional mistakes. This may prevent our framework to be applied in application domains that require highly accurate reconstruction of action styles.

As a future work of this framework we plan to explore the possibilities of using other descriptors to obtain higher recognition rates. We also would like to deploy other known faster clustering techniques [45] to identify optimum vocabulary size for the data under consideration. Techniques such as relevance feedback [62] have the potential to improve accuracy of action recognition and we would like to incorporate these into our framework in the future. Action tracking could be employed to eliminate the false matches at run time. Also as the silhouettes are produced by neighboring frame subtraction, this restricts the videos to only have static background and viewpoint.

The motivation of the second method AugMG is to augment the standard motion graph by 3D poses learned from video data to create realistic transitions. This can overcome the problem caused by the interpolation method for transition generation in motion graphs, which do not always generate realistic transitions. With a lot of interest in the field of human character animation for virtual applications, realistic transitions are of utmost importance. Our algorithm makes use of the transition information available in video to provide additional enhancement nodes to the standard motion graph. Transition frames from a video are manually annotated and body parts detectors with various viewpoints have been trained. Outputs of these detectors constitute a feature vector which is then used to train viewpoint SVM classifiers. Given a transition video frame, the viewpoint specific body part detector set outputs the skeleton structure. Along with the viewpoint recognized using the SVM classifiers, this pose estimation information is matched with the motion capture database to find the closest possible motion data frame. The matches are filtered by putting the neighboring frames and responsiveness of the transition into account. The best matching motion frames are used as reference points and their

neighboring nodes called as enhancement nodes are made available to the motion graphs in order to produce realistic transitions. An example of the application of AugMG has been presented in the experimental results section to show the usefulness of AugMG. Our AugMG algorithm has been demonstrated through a specific set of videos; however, it is independent of the videos used and can be applied to any set of actions as long as both video and motion capture data for those actions are available.

Our method has many advantages: (a) video data is less intrusive and it is readily available and easy to produce (b) pose estimation, 2D data to 3D data matching, and motion frame extraction and utilization processes can be fully automated (c) animators can select multiple transition frames to try and see the final transition result and dispose them at will until the most naturally looking transitions are found (d) this model can be used to a certain extent to impose physical constraints on the transition frames generation.

In our current method we require a manual identification of the transition frames in the video data. In the future, we would like to automate this process. Furthermore, currently we use a body model constituted of 10 body parts. We believe that a body model incorporating feet and hands will provide more accurate pose estimation and therefore better transition nodes for the transition. We would also like to incorporate weights into the comparison step to put emphasis on the joints that contribute more to the final pose. The current method is also limited by the number of viewpoints. Nevertheless, our method is the first one that utilizes the technique of learning from video data to improve the performance of the motion graph. It is efficient in creating realistic motion transitions

with considerably reduced human efforts. This model could also be easily extended to satisfy any required constraints in the transition generation.

References

[1] "Computer Animation," Wikipedia, accessed May 30, 2012. http://en.wikipedia.org/wiki/Computer_animation.

[2] K. Mikolajczyk and C. Schmid, "A Performance Evaluation of Local Descriptors," *PAMI*, 27(10): 1615–1630, 2005.

[3] Linda G. Shapiro and George C. Stockman, *Computer Vision* (Prentice Hall, 2001).

[4] "Activity Recognition," Wikipedia, accessed April 15, 2012. http://en.wikipedia.org/wiki/Activity_recognition.

[5] J. Aggarwal and Q. Cai, "Human motion analysis: A review," *IEEE Proc. No rigid Articulated Motion Workshop*, 1997, pp. 90–102.

[6] A. Bobick, "Movement, Activity, and Action: The Role of Knowledge in the Perception of Motion," *Philosophical Trans. Royal Soc. London B*, vol. 352, pp. 1257-1265, 1997.

[7] P. Scovanner, S. Ali, and M. Shah, "A 3-dimensional SIFT Descriptor and its Application to Action Recognition," Proceedings of *the 15th international conference on Multimedia*, pages 357–360, 2007.

[8] A. Yilmaz and M. Shah., "Actions Sketch: A Novel Action Representation," *CVPR*, Volume 1, pp. 984-989, 2005.

[9] G. Csurka et al., "Visual Categorization with Bags of Keypoints," *ECCV International Workshop on Statistical learning in Computer Vision*, 2004.

[10] C. Schuldt, I. Laptev, and B. Caputo, "Recognizing Human Actions: A Local SVM Approach," *Proc. ICPR'04*, pp. III: 32–36, 2004.

[11] Poppe, R., “A Survey on Vision-based Human Action Recognition,” *Image and Vision Computing* 28, pp. 976-990, 2010.

[12] Poppe, R. “Vision-based Human Motion Analysis: An Overview,” *Computer Vision and Image Understanding*, 108:4–18, 2007.

[13] Kovar, L., Gleicher, M., and Pighin, F., “Motion Graphs,” *SIGGRAPH* 2002.

[14] T. Mitchell, *Machine Learning* (McGraw Hill. p.2. 1997).

[15] XL. Chen, K. Mendhurwar, S. Mudur, T. Radhakrishnan, P. Bhattacharya “Learning Human Action Sequence Style from Video for Transfer to 3D Game Characters,” *MIG*, v. 6459/2010, pp. 422-433.

[16] Lawrence R. Rabiner, “A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition,” *Proceedings of the IEEE* 77 (2): 257–286.

[17] P. F. Felzenszwalb and D. P. Huttenlocher, “Pictorial Structures for Object Recognition,” *IJCV*, 61:55–79, Jan. 2005.

[18] Andriluka, M., Roth, S. and Schiele, B., “Monocular 3D pose estimation and tracking by detection,” *2010 IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 623-630, San Francisco, USA, 2010.

[19] M. Andriluka, S. Roth, and B. Schiele, “Pictorial Structures Revisited: People Detection and Articulated Pose Estimation,” *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR),* pp. 1014-1021, 2009.

[20] Gillies, M., “Learning Finite-state Machine Controllers from Motion Capture Data,*” IEEE Transactions on Computational Intelligence and AI in Games 1(1),* pp. 63–72, 2009.

[21] S.S. Beuchemin and J.L. Barron, “The Computation of Optical Flow,” *ACM Computing Surveys*, 27 (1995), pp. 433–467.

[22] D. G. Lowe, “Distinctive Image Features from Scale-Invariant Keypoints," *IJCV*, 60, 2 (2004), pp. 91-110.

[23] S. Kotsiantis, "Supervised learning: A Review of Classification Techniques," *Informatica*, vol. 31, pp. 249-268, 2007.

[24] “Hayao Miyazaki,” Wikipedia, accessed March 21, 2012 http://en.wikipedia.org/wiki/Hayao_Miyazaki.

[25] Ebbi Thomas, “Behavior Pattern Recognition,” accessed March 10 2012 http://www.effective-mind-control.com/behavior-pattern-recognition.html.

[26] “Shapewrap Motion Capture System,” Shapewrap, accessed March 21, 2012 http://www.motion-capture-system.com/shapewrap.html

[27] M. Gleicher, “Animation from Observation: Motion Capture and Motion Editing,” *Computer Graphics*, 4(33):51–55, November 1999.

[28] “Video tracking,” Wikipedia, accessed March 19, 2012. http://en.wikipedia.org/wiki/Video_tracking.

[29] D. Ramanan, “Part-based Models for Finding People and Estimating Their Pose,” *Visual Analysis of Humans*. Springer, Oct 2011.

[30] Cortes, Corinna and Vapnik, Vladimir N.; "Support-Vector Networks", *Machine Learning*, 20, 1995.

[31] Anderberg, M. R., “Cluster analysis for applications,” *Academic*, New York, 1973.

[32] Abdi. H., & Williams, L.J., "Principal component analysis," *Wiley Interdisciplinary Reviews: Computational Statistics*, 2: 433–459, 2010.

[33] Forney GD (March 1973), "The Viterbi algorithm," *Proceedings of the IEEE 61* (3): 268–278. DOI:10.1109/PROC.1973.9030.

[34] Lawrence R. Rabiner, "A tutorial on Hidden Markov Models and selected applications in speech recognition," Proceedings of the IEEE 77 (2): 257–286.

[35] "Kinect," Wikipedia, accessed Feb. 11, 2012. http://en.wikipedia.org/wiki/Kinect.

[36] T. Mitchell, *Machine Learning* (McGraw Hill. p.2. 1997).

[37] Arikan, O. and Forsythe, D. A. "Interactive Motion Generation from Examples," *ACM Transactions on Graphics*, 2002.

[38] Lee, J., Chai, J., Reitsma P., Hodgins, J., and Pollard, N., "Interactive Control of Avatars Animated with Human motion Data," In *ACM Transactions on Graphics*, vol. 21, 491–500, 2002.

[39] Petros Faloutsos , Michiel van de Panne and Demetri Terzopoulos, "Composable controllers for physics-based character animation," *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pp.251-260, August 2001.

[40] Marc H. Raibert, Jessica K. Hodgins, "Animation of Dynamic Legged Locomotion," *Proceedings of the 18th annual conference on Computer graphics and interactive techniques*, pp.349-358, July 1991.

[41] Rachel Heck , Michael Gleicher, "Parametric motion graphs, *"Proceedings of the 2007 symposium on Interactive 3D graphics and games*, April 30-May 02, 2007, Seattle, Washington.

[42] Ankur Agarwal, Bill Triggs, "Recovering 3D human pose from monocular images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28 (1) (2006), pp. 44–58.

[43] L. Sigal, M.J. Black, "Predicting 3D people from 2D pictures," *International Conference on Articulated Motion and Deformable Objects*, Springer LNCS 4069, Andratx, Mallorca, Spain, July 11–14, 2006.

[44] G. Mori and J. Malik, “Recovering 3D human body configurations using shape contexts,” *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 28(7):1052-1062, 2006.

[45] Suryavanshi, B.S., Shiri, N., Mudur, S.P., “An Efficient Technique for Mining Usage Profiles Using Relational Fuzzy Subtractive Clustering,” *IEEE WIRI (Web Information Retrieval and Integration)*, Washington, DC, pp. 23–29 (2005).

[46] W. T. Reeves, “Inbetweening for Computer Animation Utilizing Moving Point Constraints,” *Proceedings of SIGGRAPH 81*, pages 263-269. July 1981. ACM.

[47] P.M. Roth and M. Winter, “Survey of Appearance-based Methods for Object Recognition,” tech. report ICG-TR-01/08, Inst. for Computer Graphics and Vision, Graz University of Technology, Austria, 2008.

[48] S. Belongie, J. Malik and J. Puzicha, “Shape Matching and Object Recognition Using Shape Contexts,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 2, no. 4, pp. 509-522, Apr. 2002.

[49] E. Seemann, B. Leibe, K. Mikolajczyk, and B. Schiele, “An Evaluation of Local Shape-based Features for Pedestrian Detection,” *BMVC*, September 2005.

[50] V. Athitsos and S. Sclaroff, “Estimating 3D Hand Pose From a Cluttered Image,” *Proceedings of IEEE on CVPR*, v. 2, pp. 432-439, 2003.

[51] G. Mori and J. Malik, “Estimating Human Body Configurations Using Shape Context Matching,” *European Conf. Computer Vision*, volume 3, pp. 666–680, 2002.

[52] G. Shakhnarovich, P. Viola, and T. Darrell, “Fast Pose Estimation with Parameter Sensitive Hashing,” *ICCV*, v. 2, pp. 750, 2003.

[53] Allan, Robin, *Walt Disney's Nine Old Men & The Art Of Animation*, Animation World Network. October 21, 2011.

[54] R. Parent, “Computer Animation: Algorithms and Techniques,” Morgan-Kaufmann, 2007.

[55] P. Faloutsos, M. Van De Panne, and D. Terzopoulos, ”Dynamic Free-Form Deformations for Animation Synthesis”. *IEEE Trans. on Vis. and Comp. Graphics 3*, 3, 201–214. 1997.

[56] S. F. Gibson and B. Mirtich, “A Survey of Deformable Modeling in Computer Graphics,” Technical report, TR-97-19, MERL, November 1997.

[57] Barbič, J., and Popović, J, “Real-time Control of Physically Based Simulations Using Gentle Forces,” A*CM Transactions on Graphics* (2008)*,* 27- 5, 2008.

[58] T. Igarashi, T. Moscovich and J. F. Hughes, “As-rigid-as-possible Shape Manipulation,” *ACM Transactions on Graphics* (*ACM SIGGRAPH* 2005) 24(3):1134–1141, 2005.

[59] A. Nealen, M. Müller, R. Keiser, E. Boxerman, M. Carlson, “Physically Based Deformable Models in Computer Graphics,” *Eurographics* 2005 *state of the art report*, 2005.

[60] Oosterhof NN, Wiestler T, Downing PE, Diedrichsen J, “A Comparison of Volume-Based and Surface-based Multi-voxel Pattern Analysis,” *NeuroImage*, 56, 593-600.

[61] X. Tu, “Artificial Animals for Computer Animation: Biomechanics, Locomotion, Perception and Behavior,” PhD thesis, University of Toronto, May 1996.

[62] “Relevance Feedback,” Wikipedia, accessed March 17, 2012. http://en.wikipedia.org/wiki/Relevance_feedback.

[63] U. von Luxburg, “Clustering Stability: An Overview,” *Foundations and Trends in Machine Learning*, Vol. 2, No. 3, pp. 235–274.

[64] S. Ben-David, D. Pal, and H.-U. Simon, “Stability of K-means Clustering,” *COLT*, LNCS, pages 20–34, 2007.