INSTITUTO NACIONAL DE MATEMÁTICA PURA E APLICADA

ChoreoGraphics

An Authoring Environment for Dance Shows

Adriana Schulz
instructed by **Luiz Velho**

Rio de Janeiro, June 2011

ב"ה

Agradecimentos

Ao professor Luiz Velho, pelos ensinamentos, pela dedicação, pelo incentivo, pelo carinho, pelo exemplo. Por inspirar e orientar esse trabalho com envolvimento e entusiasmo e por contribuir decisivamente para minha inserção na vida acadêmica.

Ao professor Eduardo A. B. da Silva, pelos ensinamentos, pelo estímulo, pela amizade e pelas orientações, tão fundamentais para minha formação.

Aos professores Luiz Henrique de Figueiredo, Paulo Cezar Carvalho, Diego Nehab e Carlos Gustavo Moreira, pelos ensinamentos e pelo apoio ao longo de todo curso.

À dançarina e coreógrafa Raquel Leão, pela valorosa colaboração.

Aos colegas Aldo Zang, Bruno Madeira, Carlos Eduardo da Cruz, Djalma Lúcio, Fernanda Groetars, Juliano Kestenberg, Marcelo Cicconet e Vanessa Simões, pelas contribuições técnicas e artísticas.

Aos colegas do Visgraf, pela presença sempre alegre e pelas enriquecedoras discussões.

Ao IMPA, pelo maravilhoso ambiente acadêmico.

Ao CNPq e à FAPERJ, pelo apoio financeiro.

A minha querida família, pela intensa participação em todas as minhas conquistas.

Abstract

This work describes an authoring environment for dance shows that can both be used as a mechanism to synthesize virtual performances and as a tool to assist the planning of choreographies, allowing design and visualization of full motion sequences.

To this end, we explore adaptations of techniques for leveraging motion capture data to an application that makes extensive use of musical references. We also discuss methods for controlling the way dancers move on stage as a group, creating formations and following trajectories. Finally, we propose an integrated platform, which suggests a new form of collaboration between artists, allowing the show to naturally evolve from iterative contributions of dancers, musicians and choreographers.

In this way, we indicate how graphics technologies can be used not only to facilitate creation but also to suggest new forms of artistic expressions.

Resumo

Este trabalho descreve um programa de autoria para espetáculos de dança que pode ser usado tanto para a síntese de performances virtuais como para auxiliar o planejamento de coreografias, possibilitando o design e a visualização de sequências completas de movimentos.

Para isso, exploramos adaptações de técnicas de aproveitamento de dados de captura de movimento para uma aplicação que faz forte uso de referências musicais. Também discutimos métodos para controlar a forma como os dançarinos se deslocam no palco como um grupo, criando formações e seguindo trajetórias. Finalmente, propomos uma plataforma integrada, que sugere uma nova forma de colaboração entre artistas, possibilitando que o espetáculo evolua naturalmente de contribuições iterativas de dançarinos, músicos e coreógrafos.

Desta forma, indicamos como tecnologias de computação gráfica podem ser usadas não apenas para facilitar a criação, mas também para sugerir novas formas de expressões artísticas.

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

The availability of Motion Capture technologies has had a very significant impact on the area of computer animation. Nowadays, fast and cheap MoCap setups have become widely available and innumerable efficient techniques for leveraging motion capture data were and are being published. These novel graphics technologies are being strongly used by the movie industry, allowing the combination of real actors with virtual ones that are synthesized from captured performances.

We observe, however, that these applications are still very naive, demanding a lot of work from the actors and the artists who manually manipulate the data. In movies like James Cameron's *Avatar*, for example, the performance of the actors are directly used to drive the movements of the virtual characters with very little editing.

On the other hand, many recent papers have proposed efficient techniques for manipulating MoCap data, allowing a large number of different motions to be generated from a relatively small database and with very accurate control. These new methods for data driven animation suggest a new paradigm for content creation, allowing the synthesis of sophisticated and realistic human motion with very little effort from artists. However, in order for this revolution to in fact take place it is essential to adapt these techniques to real scenarios and develop authoring environments that serve as a bridge between artists and the emerging technologies.

Although applications of these resources are mostly observed in movies and have so far been little explored in other media, recent trends indicate that artists will also begin to use them creatively to produce new visual effects in dance shows, concerts, musicals and other spectacles. In this context, we propose an adaptation of the existing methods for leveraging MoCap data to the context of creating dance content from artistic input and propose an authoring environment for dance shows. With this, we advocate how graphics technologies can be used not only to facilitate creation but also to suggest new forms of artistic expressions.

1.2 Contributions

The contributions of our work are related to both the structuring of the authoring platform and to the development and adaptation of techniques for motion synthesis.

1.2.1 The Authoring Environment

To the best of our knowledge, we are the first to propose an authoring and collaboration platform for dance shows that integrates the creative elements that compose a choreography: music, dance and group motion. In this work, we combine the contribution of the dancers (whose performance creates the motion database), with the contribution of musicians (who guide both the dancers' rhythm and the choreography's high level control parameters), and the contribution of choreographers (who design the dance show using a proposed interface that merges all three inputs).

This platform can be used both as a mechanism to synthesize virtual performances and as a tool to assist the planning of dance shows, allowing design and visualization of full motion sequences. Though it is highly applicable in conventional shows, guiding the artists through conception, production and execution, it also suggests a new paradigm for creation. In offering an integrated platform, this work promotes a new form of collaboration between the artists, allowing the show to naturally evolve from iterative specifications.

Taken to a real–time scenario, this framework bridges the gap between conception and execution, suggesting a new paradigm for content creation. With this system, artists can propose shows that combine real and virtual dancers whose movements are controlled by instantaneous contributions from different presenters and even from spectators.

1.2.2 Dance Motion Synthesis

Dance is a very interesting type of motion because its structure is very hard to describe. In this sense, it is very different from other types of motion, such as walking, lifting, or sports movements, which can be easily associated with specific intentions. These motions are much simpler to define and, therefore, to reproduce using physical models which optimize goal, balance, and energy. Dance, on the other hand, cannot easily be interpreted as an optimization of any kind, but is at the same time not at all random, since each nuance of the movement is important to the resulting *expression*.

We have chosen to use a data driven system both because this allows the strong interaction between dancers and choreographers proposed in this work, and because we believe that the nature of the dance movements make this technique more efficient than physically based animation methods.

One of our most significant contributions is that we approach the problem of analyzing dance motion by exploring the fact that dance has a strong relation and synchronism with

music, which is a very structured signal. We take advantage of this for segmenting motion into parts (or dance steps) and this segmentation is very useful not only for making these movements available to the choreographer in the interface, but also because this structure can be extensively explored for making editing tools simpler and more efficient.

Another aspect that we explore in this work is how we can take advantage of the platform to make the database created by the dancer match the choreographer's desires as much as possible. The drawback of several data driven approaches is that they are inefficient or limited when the necessary amount of data is not available. Applications that use such approaches often require a very large amount of data, or are restricted to very specific scenarios. We get around this problem by proposing an environment that gives feedback to the artists and not only informs the choreographers of the available dance movements based on the analysis of the dancer's performance, but also suggests what the dancer can do in order to allow the result to be consistent with what the choreographer is specifying. Since our platform suggests iterative contribution of artists, we are able to have a very broad application (that can be used for practically any dance style) without demanding large amounts of data.

1.2.3 Group Motion Synthesis

Another interesting aspect of dance shows is the visual effects generated by the combination of the individual performances, i.e., the way the dancers move on stage as a group, interacting, creating formations and following trajectories.

In this work, we study the different kinds of group motions that occur during dance shows and propose an interface that allows choreographers to easily specify them. We explore declarative ways of positioning the dancers on stage, creating transitions between formations and specifying trajectories.

It is important to point out that the group motion can be regarded as the movement of particles on the 2D plane and, therefore, is by itself a interesting animation problem, which has some very interesting related work[1]. We stress that creating dance content using the computer is interesting because it allows more efficient design (allowing fast specification and visualization of the results), and also because it allows us to draw ideas from several other resources. Thus, in this work, we also explore behavioral animation methods that allow the dancers to follow attraction/repulsion forces and spread out on the stage avoiding neighbors and obstacles.

With this, we suggest a new paradigm for content creation, allowing choreographers to design new types of motions that would be impossible or very hard to specify by hand.

[1]It is important to point out that many choreographies involve nontrivial interactions between the dancers that cannot be modeled in this manner. In this work, however we do not explore such dance styles.

1.3 Thesis Structure

In the next chapters, we will present the work we have developed in detail and discuss some applications and the achieved results. We also invite the reader to view the illustrative videos available at `www.impa.br/~aschulz/ChoreoGraphics/videos`.

In Chapter 2, we describe the basic principles involved in the authoring tool, its functionalities and applications. We also introduce some background on dance analysis and design in order to justify the chosen framework and interface.

In Chapter 3, we investigate the mechanisms for character animation. Most specifically, we describe MoCap data leveraging techniques, including motion graphs, interpolations, and other signal processing editing tools. We propose variations and adaptations of these techniques to allow synthesis of multiple dancers.

In Chapter 4, we approach the problem of creating group motions. We examine both declarative and procedural methods and investigate the elements involved in specification as well as mathematical and computational foundations necessary for motion synthesis.

In Chapter 5, we demonstrate the applications of our research with experiments that illustrate the different tools that were developed.

Finally, we revisit our contributions and point out future directions in Chapter 6.

Chapter 2

The Authoring Environment

In this chapter, we will present the basic principles underlying our authoring environment. We will first discuss (Section 2.1) the applications of our platform and how it relates to the state-of-the art in dance composition. In Section 2.2, we will study the fundamental aspects involved in the process of creating choreographies in order to determine (Section 2.4) our proposed platform.

2.1 Overview

Dance shows are becoming increasingly collaborative and the responsibilities of each artist increasingly fuzzy. Choreographers, for example, used to control in almost every aspect of the process of creation, being typically the only ones responsible for designing the dance. Nowadays, however, they tend to assume the position of editor, instead of composer, and the dancers' role in the composition has become much more fundamental.

We observe that there are usually at least four types of artists involved in designing a dance show: choreographers, dancers, musicians and set designers. Allowing them to interact easily contributing to each other's work is becoming progressively more desirable. In this context, we propose a platform for designing content of dance shows which covers all elements of the creative process and promotes the communication between these different artists, allowing the show to naturally evolve from iterative contributions.

Technology has been applied in many different ways to facilitate artistic works. Here, we propose a creative tool that permits planning, editing and visualizing dances, thus guiding the artists through conception, production and execution. This can improve the quality of the results significantly, since, equipped with a system that allows easy input and instant feedback, the artists can concentrate uniquely on their craft.

Still, once we transfer the creation process to the computer, we are offered a whole new range of tools that can not only make dance composition easier and more efficient, but also suggest innovative approaches. In this work we explore, for example, methods for simulating

behaviors of autonomous agents in order to control the locomotion of the performers on the stage.

Though these contributions are significant in advancing the state-of-the-art in dance design, they are still restricted to the same conventional steps involved in its production. There are, however, some very unordinary ways with which these tools can be applied, suggesting a whole new paradigm in dance composition.

An interesting application of our framework and its extensions are "on stage" productions and improvisations, which would join efforts from choreographers, dancers and musicians in real time. An example of this would be a performance that combines live and virtual dancers projected on stage, whose movements are guided by the combination of different artistic inputs. In such scenarios, the artists would be able to influence not only the virtual dancers' movements, but also one another through a framework of instant feedback.

Ultimately, we can use this system to mediate between performers and spectators. By allowing real time collaborations, we bridge the gap between conception and execution, making it possible for the performance to be produced and presented at the same time. Hence, we can also allow the public to contribute to the dance show either controlling the dance, the music, or the set design.

2.1.1 Artistic Input

In this work, we concentrated in exploring the contributions of dancers, musicians and choreographers. In terms of set design, we considered that all the dances are performed in a rectangular empty stage. Of course, there are many interesting ways to explore set designs, considering different types of stages and various elements that can be dynamically placed on the set. However, we leave these ideas for future work.

We assume that the role of the dancers is to determine the steps that will be executed during the performance, while the role of the choreographer is to plan how these steps will be combined or sequenced, and how the multiple dancers will interact on stage (determining what we refer to as *group motions*). The role of the musician is, of course, to compose the show's soundtrack.

A very important aspect that must be analyzed is how each of these artists should input their creative abstractions into the computer. The most simple input is perhaps the musicians', whose music can be represented by a simple audio signal. Of course, there are several other layers of information that could also be described and we could even suggest a method for automatic composition of the songs based on high level specifications. Music composition, however, is a problem on its own and the topic of several other works. Hence, we decided not to explore such mechanisms during this research. Instead, we consider the musicians' input to be a musical piece with several annotations. These annotations consist of the music's segmentation into measures and also control signals that indicate musical events that can be

synchronized with dance movements. These signals are essential for promoting the interchange of ideas between choreographers and musicians, as will be discussed in the following sections.

Dance specification is somewhat more complex, since movements have many degrees of freedom which cannot be easily determined by a standard notation. In this work, however, we used a MoCap setup to capture the movement of the dancers, therefore allowing dance steps to be fully determined by example. We also propose methods for storing the captured clips based on annotations made by the dancers. This results in a structured data set that can be used by the choreographer during dance design and also by the system's motion synthesis algorithms that allow visualization of animated dancers.

Finally, it is necessary to determine the choreographers' input. Analogously to dance input, this is also a very complex information with no existent standard notation that describes it. In this case, however, there is no available set up that allows us to easily capture the choreographer's specifications and transform it into a signal, as is done with MoCap. We also observe that the choreographers' role is much more central to the dance, since they are the ones who usually coordinate how the input of the other artists should be combined. Hence, we decided to develop an interface for choreographers that permits them to specify group motions and determine how the dance steps should be sequenced in a way that is consistent with the musical reference.

The interface for choreographic input is a fundamental tool in the proposed platform. To develop such a tool, it is very important to understand the choreographer's process of creation in order to translate it to a language the computer understands. In short, we need to specify and classify the relevant components of a group dance and find a way of making them available for use in our authoring environment. This is somewhat challenging, first because there is no established list of all motion elements that may be present in a group dance and, second, because there is no standard notation that choreographers use to determine the dance steps and group motions. For these reasons, we have decided to approach the interface design problem by first making a careful study of dance elements.

2.2 Dance Analysis

In this section, we will describe methods for characterizing dance and discuss how we classified the essential dance elements.

2.2.1 Approach

The most important reference in dance analysis is the Laban Movement Analysis (LMA), which is a method and language for interpreting, describing, visualizing and notating all forms of movement, created by Rudolf Laban. LMA determines four basic elements of dance: *body*, which refers to structural and physical characteristics of the human body while moving; *effort*,

which refers to the strength, control and timing of the movement (would discriminate an angry punch from a reaching motion); *shape*, which refers to the designing of the body as it exists in space; and *space*, which involves motion in connection with the environment, spatial patterns and pathways.

Though this formalization has had a great impact on the dance community and even been used by computer graphics researchers [34], we decided not use the LMA in our work for a couple of reasons. The first reason why this approach is not efficient for our application is that it is mostly concentrated in analyzing movements of a single human while our main interest consists in specifying group motions. The second and more important reason why we abandoned the idea of using Laban is that it is a system built for human observers and cannot be easily translated into a computer. Is it greatly important for our application to use a language that not only describes dance efficiently but also one that can be used as a bridge between choreographers and computers. The discussion about how much human-designed knowledge should be used in a computational environment is not recent. A famous quote from the speech scientist Fred Jelinek that dates back to the 1980's states: "Every time I fire a linguist, the performance of our speech recognition system goes up." Though this quote can definitely be interpreted as an overstatement, it indicates that one should be careful when trying to apply formalizations from other fields to computer science.

Finally, it is important to mention that the LMA was developed in order to allow dance movements to be recorded and repeated in the future. This is a big challenge because dance, unlike music, does not have a simple standard notation and often choreographers have to memorize whole dance sequences or use video recordings to register their compositions. Therefore Labanotation came out as a very important innovation. Our application, however, requires a language that allows creation and, hence, should not be based on the dance elements that are needed for notation purposes, but on the elements that are required for dance design and movement specifications.

Our approach, therefore, was to create a new vocabulary for describing dance. We started by analyzing a series of dance performances while, at the same time, studying the previous work on motion synthesis (see Chapters 3 and 4) and drew, from both sources, a classification of dance group movements. It is our intention to validate, in the future, our approach with feedback from dancers and choreographers.

Before describing our classification, it is important to emphasize that we are distinguishing between two different motion levels: group motions and individual motions. Group motions refer to the visual effects that result from the combination of individual movements, while individual motions refer to the actual dance steps that are being performed. This distinction is not only useful to help formalizing the problem, but also contributes to the construction of our interface.

As previously mentioned, we are interested in allowing choreographers to fully specify group

motions using high level commands. The individual movements, on the other hand, should be designed only by determining how the dance steps will be sequenced, while the actual dance steps are left to be specified by the dancers, using the MoCap setup described above. Hence, we have concentrated in analyzing only the elements of group motions.

2.2.2 Group Motion Modeling

We propose the classification for group motion modeling illustrated in Figure 2.1. We base our classification on the nature of specification, which can be: spatial, temporal or categorical.

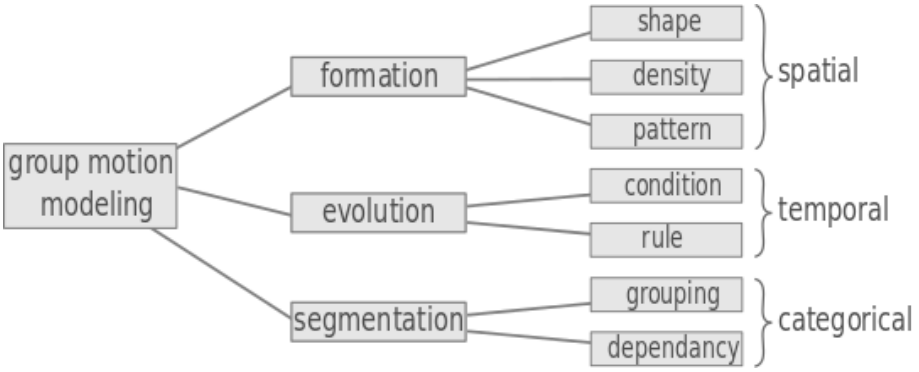


Fig. 2.1: Modeling elements.

The spatial elements of group motions refer to the way the dancers are positioned and distributed in space at a given time, creating formations on the stage. A first aspect of spatial specifications is determining the geometry of the overall shape formed by the dancers' positions. Secondly, the choreographers should determine the density, i.e., the number of dancers that should be positioned on the given shape. Finally, the choreographers should define the distribution pattern, designating rules for arranging the dancers (e.g., maximizing distances between them or following a grid).

Temporal elements refer to ways of specifying how the dancers' positions evolve in time. This is done by designating the evolution conditions and rules. We observe that evolutions can be specified based on fixed, boundary, or initial conditions. Fixed conditions refer to selecting a formation that the dancers should assume for a determined time period. The choreographers can also specify an evolution where the only restrains are the initial and/or final conditions. In these cases, the choreographers should also specify the evolution rules that will define the trajectories that the dancers should follow from a given initial state or methods for evolving between two boundary conditions.

The final kind of group motion element that we analyzed is categorical, referring to different ways of rearranging the group of dancers into smaller subgroups and creating dependencies between them. This way, the choreographers can specify different formations and evolutions to different subgroups. The group motions associated to these subgroups of dances can be inde-

pendent or have an intimate relation given by dependency rules (e.g., symmetric movements).

2.3 Extensions

In truth, authoring group motions involves not only modeling, but also visualization, i.e., determining projection and staging. This is very important for the design interface, since choreographers' input usually involves sketching and moving objects in different ways. Visualization is even more important during presentation and therefore authoring tools should allow choreographers to preview the resulting motion from different points of view.

Authoring dance shows, therefore, involves modeling and visualization of the different elements that compose the show: dance steps, group motions, music and set design. All of these elements should be specified in parallel and, preferably, within a collaborative environment (see Figure 2.2).

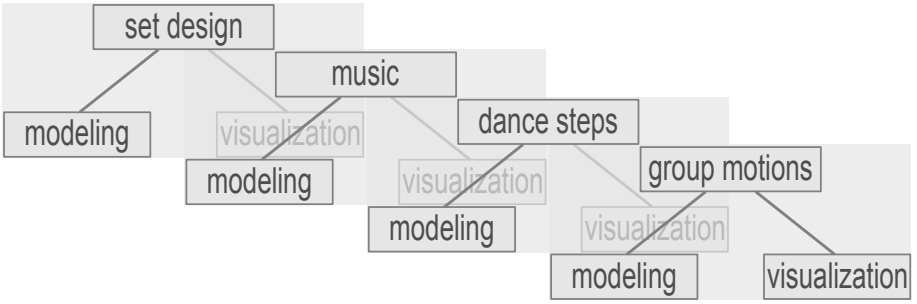


Fig. 2.2: Dance shows authoring.

As previously argued, we will not discriminate all the modeling and visualization aspects of these other elements, since the previous discussion on modeling motion groups will suffice for designing the authoring tool we propose in this work.

2.4 The Proposed Platform

In this section we will discuss the main functionalities of our framework. As mentioned before, we have suggested a platform that integrates music, dance, and choreography. In what follows we will discuss how each of these elements are explored in our authoring tool.

2.4.1 Music

Songs are mostly very structured, organized in measures (or bars) that are sets of beats[1] and dance steps usually have the same duration as an exact number of measures.

Typically, choreographers start to plan a dance show by first selecting a musical piece. Then, they sit down with a blank sheet of paper and extract the rhythm of the song by making a mark on the paper every time they count four (or eight) beats. After segmenting the song into measures, they use this structure to design all aspects of the dance: steps, group motions, etc.

In accordance with this, we make extensive use of the music's rhythmic structure in several aspects of our work. We use a discretized timeline as the foundation of our interface, as shown in Figure 2.3. This timeline allows choreographers to plan the elements of the dance show using the musical structure described above.

Fig. 2.3: Musical reference. From top to bottom: representation of the music segmented by the measures; control signals; the authoring timeline (i.e., a sequence of small boxes appear that correspond to musical measures and are filled in with elements that indicate to the choreographers' specifications).

We also use the music's rhythm to guide the segmentation of the MoCap data into segments that correspond to motion measures. Although the purpose of this segmentation is to facilitate the combination of motion sequences in an application that makes extensive use of musical references, this structure can also be explored for motion editing, as will be discussed in Chapter 3.

In addition to the rhythm, we also explore the music's melody, which is used to determine control signals that should be synchronized with motion events (see Figure 2.3). These signals can either be extracted from the selected music and used as a reference to guide the dance, or can be iteratively edited by both the musician and the choreographer in a scenario where the show is designed in a collaborative effort of both artists.

2.4.2 Dance

As previously discussed, we use a motion capture system to acquire the dance motions. In order to be able to synchronize the dance with the musical rhythm, we capture the movements of the dancers while they perform to the music or a "tack-tack" audio signal that counts the musical beats.

[1]Depending on the score, measures can have any fixed number of beats, specified in a musical score by the time signature, which appears at the beginning of the piece, as stacked numerals. However, for the purpose of this discussion we will consider songs that have four beats per measure, which are the most popular. The arguments would follow in the same way, had we chosen to work with any other time signature.

We can synchronize the music with the dance by having the artists perform some very specific movements before starting to dance. In this work, we asked the dancers to clap to the sound of the beats. This allows us to determine the exact musical reference in the acquired motion signal, as shown in Figure 2.4. With an accurate synchronization, we can segment the data into motion sequences that correspond to musical measures.

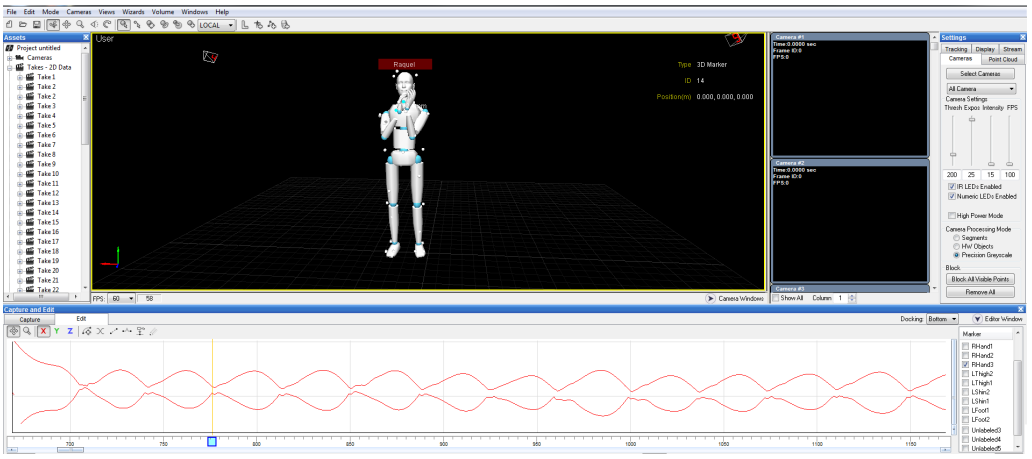


Fig. 2.4: Synchronization with music while dancers clap their hands. Notice that we can determine the beats of the song by finding the moments in time when the distance between the two hands reaches its minimum (screen shot from Arena software).

In addition to this segmentation, we also have the dancer annotate the data, specifying which motion step the sequence refers to. The final structure of our motion database is, therefore, a set of annotated clips, where each clip has an integer number of measures and corresponds to a specific dance step. Notice that steps can have an arbitrary number of measures, since in many cases it may be useful to capture a combination of movements that will be performed one or several times during the choreography, but always in the same order.

It is also important to point out that each dance step may be captured several times if the dancers want to perform different variations (e.g. steps that perform trajectories, may need to be performed facing several directions). These variations are also annotated and will be used in many different ways during motion synthesis. We will discuss these aspects in detail in Chapter 3.

2.4.3 Choreography

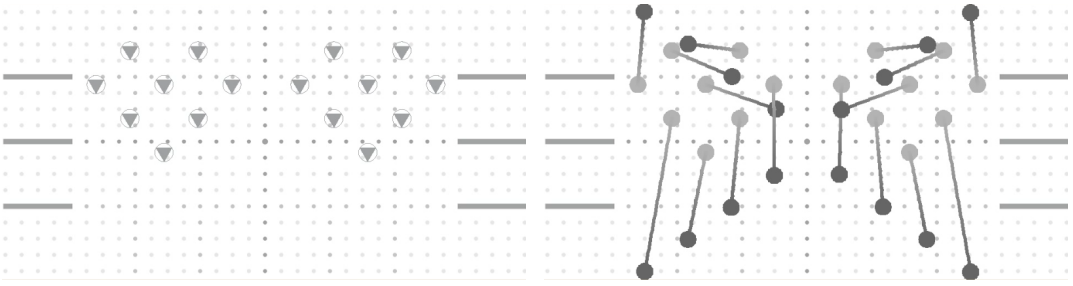
As discussed above, creating the choreography involves both designating which steps each dancer will perform at any given time and the way the group will move along the stage creating formations and following trajectories. By exploring the classification described in the previous section, we propose an interface in which choreographers can specify both of these elements.

Creating an interface for dance design is challenging because we want an environment that

is natural for choreographers to use and, therefore, it should be similar to the language they use to annotate compositions. Since there is no standard notation for group dance specification, we decided to propose an interface that, to the best of our knowledge, approximates what we have found to be natural for these artists.

In our discussions with choreographers and review of related work [5], we observed that group motions are usually specified with images of an upper view of the stage, where the dancers are represented as circles (often color coded) and lines or arrows indicate trajectories. Typically, these images are staked together creating a storyboard that indicates the sequence of motions.

In view of this, we created an interface for specifying group motion that has a window that represents the stage and where the choreographers can position the dancers (represented as circles with inner triangles that indicate orientation) and determine trajectories (represented by gradient colored lines). This is illustrated in Figure 2.5. Right bellow this window we place the musical reference with the authoring timeline. By clicking at any segment of the timeline, the choreographers can view the formations and trajectories specified for this segment.



(a) Representation of dancers. (b) Representation of trajectories.

Fig. 2.5: Window that represents the stage on the interface.

In terms of representing individual motions, we have found that there is no notation, except full written descriptions or videos, that would be efficient for all dance forms. For this reason, many choreographers have chosen to film the performance of the dance steps in order to be able to reproduce them later on. Based on this idea, we have suggested presenting the choreographer with a list of the names of the captured dance steps. By clicking on any one of these names, the choreographer can preview an animation of the motion (see Figure 2.6) or see information such as duration, available variations, etc.

Hence, our interface has two modes: group motion specification and dance movement specifications (see Figure 2.7). In both of these modes, specifications are made using the authoring timeline shown in Figure 2.3. In the dance motion mode, choreographers can specify dances movement at each measure by filling the boxes on the authoring timeline using a drag-and-drop motion from the list of the steps. In the group motion mode, the choreographers also have to fill in these boxes, but with elements that correspond to evolutions. Notice that

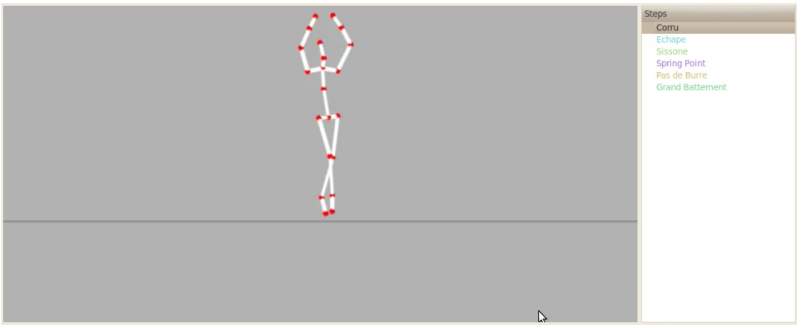
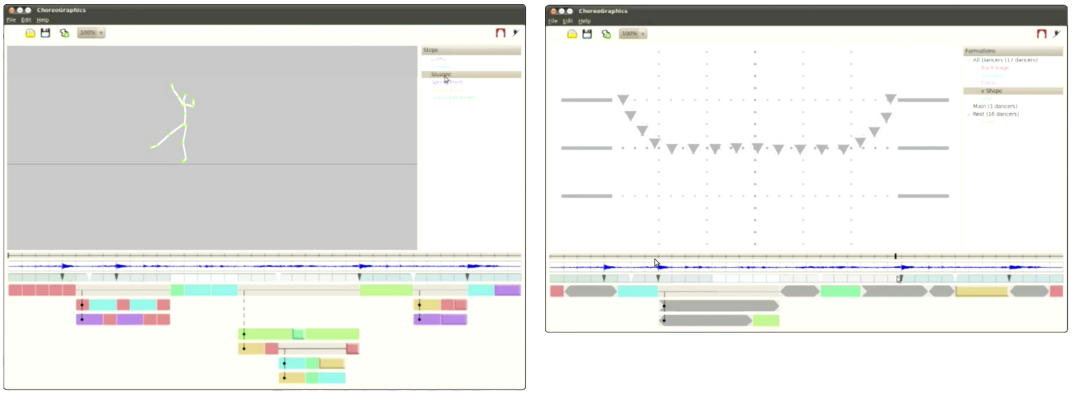


Fig. 2.6: Preview of dance steps. Notice the list of dance steps on the right hand side.

evolutions often depend on formations and, therefore, instead of a list of steps, the interface's mode for group motion specification presents a list of formations, which choreographers can create by determining shapes, densities and patterns. We will discuss the methods for specifying formations and evolutions in detail in Chapter 4.



(a) Dance steps mode.

(b) Group motions mode.

Fig. 2.7: The two modes of the interface

We explore the categorical aspects of group motions by allowing the dancers to be grouped and regrouped in different ways throughout the dance. We developed a hierarchical structure for storing these segmentations and make them visible on the interface by subdividing the timeline, as show in Figure 2.8. We will also discuss the techniques involved in this process in Chapter 4.

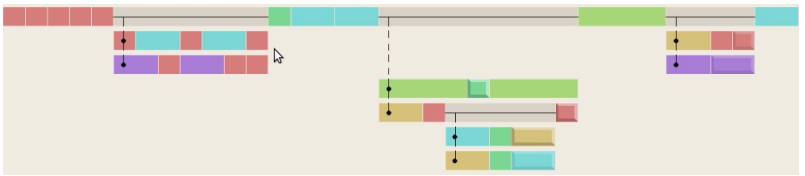
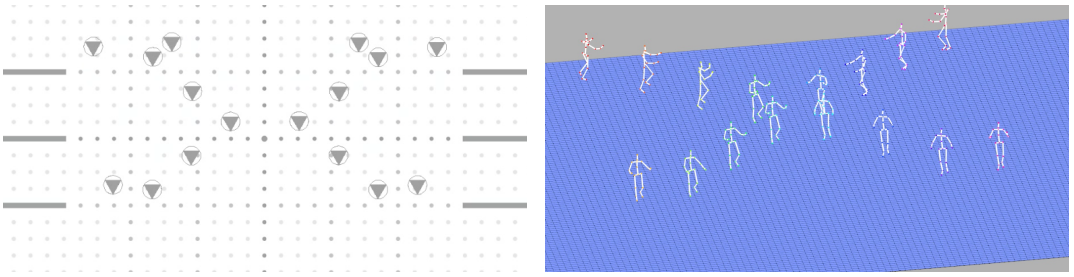


Fig. 2.8: Visualizations of subgroups.

It is important to point out that choreographers should be able to specify dance steps to

groups, subgroups or individual dancers. Hence, we use the same hierarchical structure to allow them to divide and subdivide the group on the dance movement specification mode. Notice, however, that these groupings do not have to be the same (i.e., choreographers can specify the same dance step but different group motions to different parts of the group and vice-versa).

We also allow users to visualize the resulting 3D animation in a separate window and to preview the group motion by playing an animation of the resulting 2D movement of the circles, as shown in Figure 2.9. This way, we explore the visualization elements discussed in Section 2.3. Notice that, although we chose a single (top) view for the 2D motion preview, we allow choreographers to visualize the 3D motions from several different viewpoints.



(a) 2D motions (b) 3D motions

Fig. 2.9: Two different motion previews.

Chapter 3

Dance Motion

In this chapter we will discuss our approach for the synthesis of the individual dancers' motion. We will start with a brief discussion of character animation, describing the current techniques for data-driven animation (Section 3.1.2), and most specifically Motion Graphs (Section 3.1.3), which was the method we explored in this work. We will then propose an extension of this technique that allows efficient synthesis of dance motions, in Section 3.2. Finally, we will discuss the proposed motion editing tools (Section 3.3) and the proposed methods for combining motion segments (Section 3.4).

3.1 Overview and Related Work

By definition, to animate is to bring to life, in this case, to make a lifeless object (a graphics model) move. Realistic animation of human motion is a challenging task, firstly, because human movements can be quite complex since joints have many degrees of freedom and, secondly, because people are skilled at perceiving the subtle details of human motion. For example, people are capable of recognizing friends at a distance purely from the way they walk and can perceive personality and mood from body language. This implies that synthetic human motion needs to be very accurate in order to appear real.

It is the opinion of many researchers in the area that synthesis of realistic human motion can only be made possible by an approach that makes extensive use of data from the real world [14]. In this context, it has been quite appealing for animators to use methods of copying the movements of real life objects.

An early example of such a technique is Rotoscoping, in which animators trace over live-action film movement, frame by frame, in order to create realistic motion. Though archives show that most of Snow White's movements were traced from actors motions using this technique, Walt Disney never admitted to it. In fact, this was considered not only "cheating", because it was not produced by the imagination of the artist, but also "cheep animation" because animations were supposed to be "bigger than life", not merely a "copy of life" [3].

Rotoscoping was, naturally, a precursor of MoCap and the controversy around it has the same origin. On the one hand, there is the need for creating engaging and expressive characters and, on the other hand, the need for synthesizing realistic motion efficiently and fast [10].

The techniques for motion creation are usually divided into three basic categories: manual specification (key framing), motion capture and simulation [13].

Key framing borrows its name from traditional hand animation and is, in fact, very similar to it in the sense that, while computers reduce some of the labor by automatically interpolating between frames, the animator has to specify critical (or key) positions for the objects. This requires great training and talent, since characters usually have many controls (e.g, each of the main characters of the movie Toy Story, which were animated in this fashion, had more than 700 controls). The advantage of this method is that the artist is able to control subtle details of the motion. Nevertheless, it is very hard to make the characters look real.

Motion Capture, as we have mentioned above, is the process of recording movement and transferring it to an animated object. Some of the advantages of motion capture over traditional computer animation are [1]:

- more rapid, even real time results;
- real, natural movement data;
- extremely accurate 3-D data that permits the study of the essence of the movements; and
- data formats that require minimal storage and allow for easy manipulation and processing.

The physically based approach uses laws of physics to generate motion through simulation and optimization. This technique is largely used, not only for human motion, but also to animate fluids, fire, explosions, face and hair. Simulation techniques supply physical realism, while MoCap allows for natural looking motion. Currently, many applications merge both techniques together in order to create models of human motion that are flexible and realistic [14].

In this work we make extensive use of MoCap techniques. For additional information on MoCap system and the setup that we have used in the Visgraf lab, refer to Appendix A.

3.1.1 Representing Motion

The most standard way of representing human motion is using a hierarchical structure [11]. The human skeleton is thus described by a root node and a sequence of connecting joints, while the pose is fully determined by the 3D position of the root node and the rotation of each joint.

The BVH file format is one of the most popular formats for representing human motion and is compatible with many programs, such as Arena, Blender and Maya, which were used

during this project. It specifies first the human skeleton, i.e, the hierarchical node structure and the distances between each node, and then the motion sequence, i.e, the pose at each frame. For more details on the BVH file notation, see Appendix B.

3.1.2 Data-driven Animation

Though Motion Capture is very powerful in the sense that it allows rendering natural and realistic motion, MoCap by itself is nothing more than a method for reproducing acquired movements. Therefore, much effort has been and is currently being put into extending the applications of MoCap data.

There are many reasons that make editing captured motion extremely important. Firstly, it is usually necessary to eliminate artifacts generated during acquisition. Secondly, it is important to match time and space of computer generated environments, overcome spatial constraints of capture studios and allow for the existence of motions that would be extremely hard for an actor to perform, such as the ones used in special effects. Finally, it is interesting to be able to reuse motion data in different occasions. For example, given a walking scene, it should be possible to generate a walk on an uneven terrain or stepping over an obstacle.

In addition to editing [37, 4], it is the interest of many researchers in the field to synthesize new streams of motion from previously acquired data and, therefore, be able to create new and more complex motions. Motion synthesis strategies include constructing models of human motion [21, 2], interpolating motion to create new sequences [16] and reordering motion clips employing a motion graph [17].

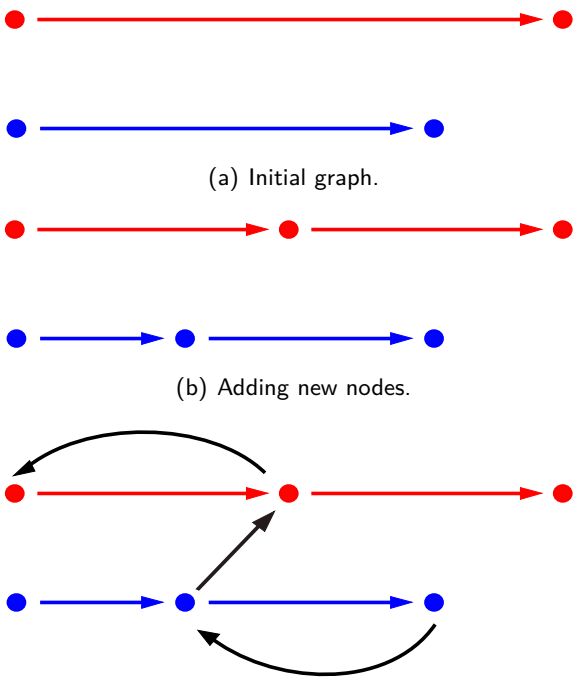
In the next section we describe the motion graph technique, which was the one we explored during this work.

3.1.3 Motion Graphs

Motion graphs were introduced in [17], in order to encapsulate connections among a database. In this graph, edges correspond to motion clips (sequence of frames) and nodes to choice points (specific frames) connecting these clips. After selecting the similar frames, or windows of frames, and creating the graph connections, a walk along the graph allows us to re-assemble the captured clips, creating new motion.

Figure 3.1 illustrates the construction of a motion graph. Initially we have two clips of motion from a MoCap database. The resulting graph is the one shown in Figure 3.1(a), which is disconnected and quite simple. Notice that it is possible to add nodes to the graph, simply by breaking the initial motion clips into two or more small clips, as shown in Figure 3.1(b). Nevertheless, this graph is still disconnected.

To achieve greater connectivity, we introduce transition clips, as shown in Figure 3.1(c). Notice that these new edges also represent clips of motion which have to be synthesized by



(a) Initial graph.

(b) Adding new nodes.

(c) Create greater connectivity.

Fig. 3.1: Construction of a motion graph.

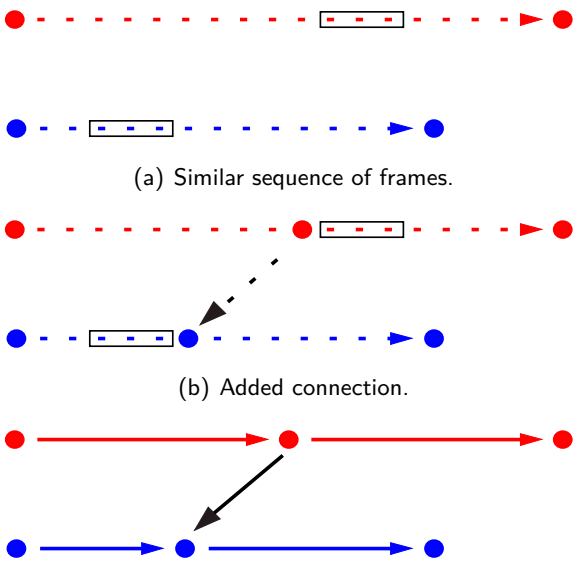
the technique. Therefore, in order to create these new edges we first have to select the nodes which can be effectively connected and then create the motion clips to which the transition edges will correspond.

Following the ideas proposed in [17], we select the nodes which are sufficiently similar so that straightforward blending is almost certain to provide valid transitions and use a similarity norm that takes into account a sequence of frames and a point cloud driven by the skeleton.

As in the BVH file, a motion frame is fully described by the position and orientation of the root node and the rotation of the other joints. However, an attempt to use this information to calculate similarity between frames will be unsuccessful because some of these parameters have a much greater overall effect on the character's position than others (for example, a knee rotation is usually much more significant than a wrist rotation). In addition, it is quite hard to set fixed weights that work well in all cases, since the effect of a joint rotation on the shape of the body depends on the current configuration of the body.

To solve this problem, we use a point cloud, which describes the skeleton pose (ideally this point cloud is a downsampling of the mesh defining the character), and a vector distance norm. It is important to notice that a motion is fundamentally unchanged if we translate it along the floor plane or rotate it about the vertical axis. Therefore, before comparing the distance between two point clouds we apply a rigid 2D transform to the second point cloud that sets it at the optimal position so that weighted sum of squared distances is minimal.

In addition, it is important to consider not only the current position but also derivative



(a) Similar sequence of frames.

(b) Added connection.

(c) Resulting graph.

Fig. 3.2: Addition of a transition edge.

information, such as velocity and acceleration. We incorporate this information to the similarity metric by using a window of frames, i.e, we take N frames from one clip ($F_1(n)$, for $n = 1, \ldots, N$) and N frames from the other ($F_2(n)$, for $n = 1, \ldots, N$) and calculate $\sum_{n=1}^{N} \|F_1(n) - F_2(n)\|$. This also helps the blending procedure. An example is shown in Figure 3.2. The similarity metric selects the two sequences of frames as shown in Figure 3.2(a). Therefore we can add an edge that connects the beginning of the first sequence of frames to the end of the second one, shown in Figure 3.2(b). The resulting motion associated to this transition edge is created by blending the two sequences of frames:

$$F_{\text{result}}(n) = \alpha(n)F_1(n) + (1 - \alpha(n))F_2(n), \quad \text{for } n = 1, \ldots, N,$$

where $\alpha(n) = \frac{n-1}{N-1}$

Finally, after the edges are created, we prune the graph by computing the strongly connected components (SCCs). We use only the largest SCC so as to guarantee that there are no dead ends, and therefore we can synthesize the motion indefinitely[1].

3.2 Our Approach

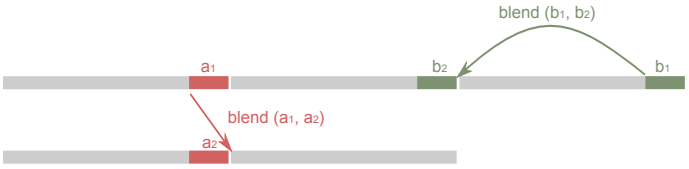
As discussed in the previous chapter, one of the most essential aspects that distinguishes dance from other types of motion is the rhythm and the synchronization with music. Since choreographers design dances based on this rhythmic structure, it is essential to incorporate this into

[1] Although some of the results presented in this work were acquired with this technique, towards the end of this research we found that we were able to eliminate this step using a controlled graph search which we will describe at the end of this chapter

our model for motion synthesis. In this work we chose to explore the motion graph technique for creating new motions from a collection of MoCap data. Although a straightforward use of this technique to our problem would create a plausible motion, it would not create a plausible dance, since it would combine pieces for motion of random duration which would not likely fit into any rhythmic structure.

For this reason, we propose, an extension of a motion graph, which is *structured* and *measure-synchronous*. As described in Section 2.4.2, we organize the captured data as a group of clips that consist of sequences of measures (i.e, the duration of each clip corresponds to an exact number of musical measures). Hence, we create a *measure-synchronous* motion graph by, instead of creating nodes at any frame, only allowing nodes to be placed at the end of each measure (see Figure 3.3(a)). With this we guarantee that even a random walk on this graph will result in a motion that is coherent with the music metric.

(a) Graph Connections



(b) Actual Trasitions

(c) Stored Clips

Fig. 3.3: The *measure-synchronous* motion graph.

It is important to mention that, as discussed in the previous section, the graph transitions are created by interpolating a sequence of frames (typically we have used 20 frames), as shown in Figure 3.3(b). This means that, in order to create a transition to a specific measure, we need to store the frames that immediately precede it.

We create a *structured* motion graph by cropping the input motion stream so that each stored clip corresponds only to a single step and by annotating each clip with the name of the dance step and its variations. Notice that clips may have different sizes because, as discussed in Section 2.4.2, the steps may last for any integer number of measures. Moreover, clips that are annotated with the same step name always have the same size. The size of each clip is the number of measures in the step plus one (the extra measure is the one necessary for creating

the graph transitions to the initial measure, as shown in Figure 3.3(c)).

With this structure, we can represent the motion graph as a set of clusters, as shown in Figure 3.4. Each cluster represents a set of instances of the same step. The instances may be different in several ways. For example, a step that involves moving along the stage may be performed in different directions. Since human body is symmetric, most steps can be performed with (or start with) the right foot or left foot. Finally, the beginning and ending of each step depends on the way it is being combined with other steps in the choreography. For example, if a ballerina plans to do a Couru after an Échappé, she will conclude the step with a Sous-sur (a step that rises up), if not, she will finish the step in a Plié (bending of the knees).

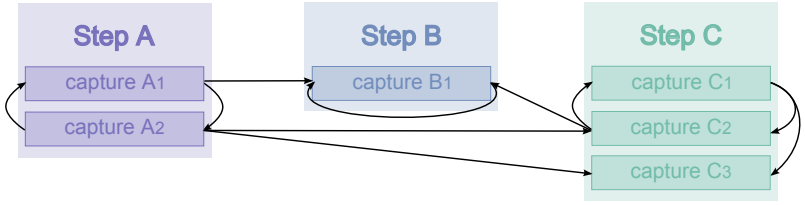


Fig. 3.4: Clusters of dance steps.

Notice that these different step variations are very important to capture since they will be essential for motion synthesis. Having multiple examples of the same step allows us to create more complex movements, as for example making the dancers follow different trajectories on stage while dancing. Also, as illustrated in Figure 3.4, the more instances of the same step we have, the greater the connection between the graph clusters.

It is noteworthy that the effectiveness of our motion synthesis method is highly dependent on the captured data. Instead of trying to get around this problem, by allowing some kind of suboptimal synthesis, we have explored the fact that our platform is integrated and that the choreographer can give feedback to the dancer while they are designing the choreography. Therefore, we have suggested a system for informing the artists when a given specification cannot be accurately synthesized and what kind of input they should provide to resolve the problem.

In this way, we argue that our synthesis technique will always find a solution and that this solution will be optimal in the sense that it will fit the exact specification. We will discuss this in more detail in the following sections.

3.3 Motion Editing Tools

As we mentioned before, one of the advantages of the structure that we developed is that it allows us to use very simple motion editing tools based on signal processing techniques and get very realistic results. In this work, we have explored such methods for interpolating motions, for inserting rotations, for combining upper and lower body motions, and for inserting amplitude

variations and time warpings to make the group dance more natural.

3.3.1 Interpolations and Combination

The greatest challenge with interpolating and combining upper and lower body motions is that the results will not be seamless unless there is a reasonable alignment between the two motion segments and that may require nontrivial warpings. In this work, we argue that since we have clips of the same dance step which have equal durations, we guarantee that they are trivially aligned by construction.

For combining motions, this means that we can trivially replace the joint rotations of the upper body part of a motion clip with that of another motion clip that corresponds to the same dance step. This allows us, for example, to capture several iterations of a dance step with a single arm motion and only one iteration of the same dance step with another arm motion, but at the end have all the iterations we desire with both arm motions.

For interpolation, this means that simple blending yields very accurate results. Motions are represented by a 3D position and a set of joint rotations. Therefore, to guarantee a smooth interpolation, we use quaternion representations and the Spherical Linear Interpolation (SLERP), which is a geodesic that connects the mapping of the two points in S^3. For a detailed discussion of rotations and interpolations, refer to [32].

It is important to mention that this is only done for clips of the same dance step that have the same structural variation (it would not be possible, for example, to interpolate a sequence that is performed with the right foot with one of the left foot). This technique is largely used in our application to allow dancers to follow the group motion, i.e., to synthesize a step that makes the character move in any direction from only a few captured instances. For example, we can create an accurate diagonal walk by interpolating a side and a front walk. We can also create any kind of intermediate step length by interpolating short and long steps.

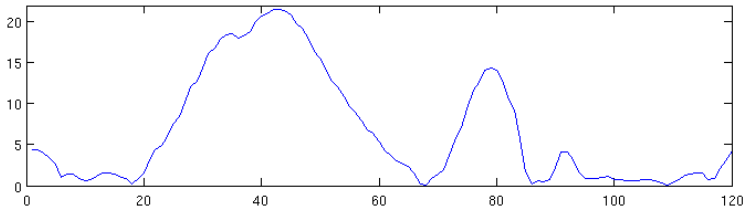
3.3.2 Rotations

Although dancers usually face the public during the show, it is often necessary to allow them to choose different orientations. Also, although interpolating allows us to synthesize dancers moving in different directions while still facing forwards, it is often more natural to move while facing the path. For these reasons, we have explored efficient tools for editing orientation.

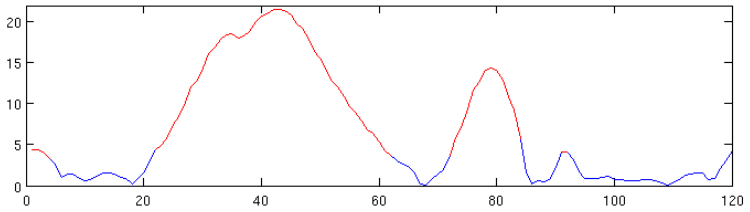
One of the greatest challenges with making the characters rotate while performing a step is maintaining floor contact. It is not uncommon that editing techniques produce the undesirable "Michael Jackson" effect, known as foot-sliding. Although these artifacts do occur often and methods for efficiently removing them have already been proposed, it is still very important to try to minimize them when manipulating motion.

We observe that, in a walking step, rotations occur when only one foot is on the ground.

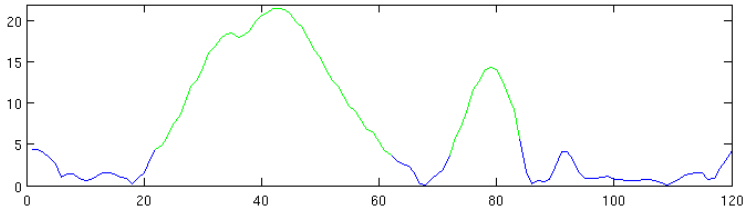
Therefore our approach was to analyze the foot contact, select the segments in which there is only one foot touching the floor and apply a rotation and translation transform to the motion sequence that changes the orientation of the dancer while maintaining the foot contact.



(a) Distance between the height of the right and left foot.



(b) Regions where the distance is greater than 20% of the maximum distance value.



(c) Final selected regions.

Fig. 3.5: Selecting rotation regions. Graphs indicate the absolute value of the distance between the two feet in centimeters along the number of frames.

Figure 3.5 illustrates this procedure. We first calculate the distance between the height of the right and left foot and select the regions where this value is greater than 20% of the maximum value (see Figure 3.5(b)). Notice that although this allows us to select the regions where there is a significant distance between the two feet, errors do occur and a couple of scattered frames that are only slightly above this threshold end up being selected. To get around this problem, we explore the fact that steps tend to be smooth and therefore a significant amount of time is required to take a foot off the ground and then place it back. Hence, we ignore all the segments that we consider too small (we used 10 frames as a threshold) and are therefore able to eliminate the small errors.

In our experiments, this technique has been very efficient to select the frames where rotations can be inserted. Notice that we are editing the motions at each measure. This is very important because if we had a very large motion clip we would have to select a certain moment

or a certain movement to apply the transformations. However, since we know that a whole sequence corresponds to a specific dance step, we can distribute the rotation angle uniformly between all the selected frames.

Given a desired rotation angle θ and N selected frames, we apply a rotation of θ/N to each frame. We then calculate, also at each frame, the distance between the position of the contact foot before the transformation and after and calculate a translation that eliminates it. The full transformation (rotation + translation) is then applied to the whole sequence starting from the selected frame.

Finally we should mention that we do not take into account the last few frames, since they are used during the interpolation process for the motion graph connections.

3.3.3 Variations

When replicating the movements of a single dancer to synthesize multiple characters performing on the stage, it is essential to make some modifications on the data to make the group dance look natural. In this work, we have explored time warping mechanism for desynchronizing the data and amplitude variation methods for varying the upper body motion. We chose to apply the latter tools only to the upper body part because they are less susceptible to undesirable artifacts, since there are no floor contact restrictions. If we applied the same editing methods to the lower body motion, we would probably get some undesirable foot-sliding effects and could even have the dancer penetrate the floor or "fly" over it.

Time Warping

There are two ways of approaching the time warping problem: considering that editing is done independently at each measure, or making an overall editing operation after the whole motion is created. Editing at the measure level may be simpler to implement, as it can be done prior to the graph search. However, we observe that the last few frames of each measure (which we use to create the graph transitions) cannot be warped because the synchronization of the measures is essential for making the blending smooth. Therefore, when time warping is done at the measure level, the resulting motion is not as desynchronized, since at the end of each measure there is a sequence of frames in which the motions match exactly.

A function that time warps the dance movements should randomly oscillate around the identity function. These oscillations should be significantly small because accelerating and decelerating the motion too much would make the dance look unnatural. We also want a function that is monotonically increasing because this will guarantee that the sequence of movements is done in the right order, i.e., that the motions are not played backwards at any moment.

To guarantee causality, we suggest a warping function that is the sum of an identity and a

sine function:

$$w(x) = x + A\sin(\frac{2\pi f x}{N}),$$

where x is the frame number, N is the total number of frames, and A and f are a chosen amplitude and frequency.

Notice that

$$w'(x) = 1 + A\frac{2\pi f}{N}\sin(\frac{2\pi f x}{N}),$$

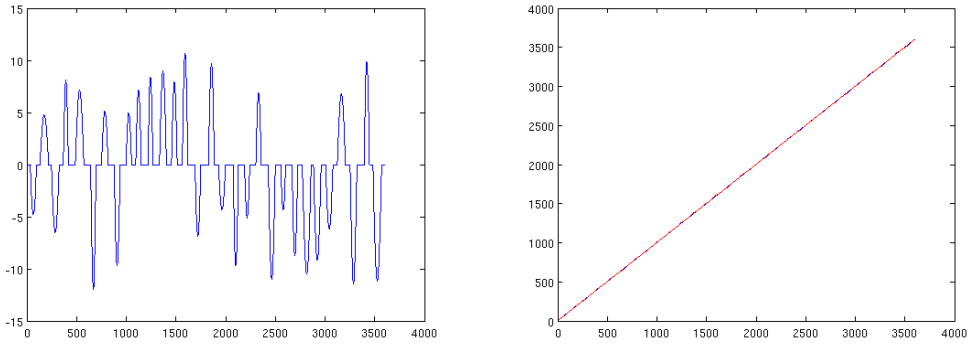
which is greater than zero if $A\frac{2\pi f}{N} < 1$. Hence, if we restrict the values of A and f, we can guarantee that this function is monotonically increasing.

When working at the measure level, we choose A from a uniform distribution, such that $0 \leq A \leq A_{max}$ and f from a uniform distribution, such that $0.5 \leq f \leq f_{max}$ (typically, we used $A_{max} = 7$ and $f_{max} = 2$, for $N = 100$). By restricting $0.5 \leq f$, we guarantee that there is at least one half sine cycle between frames 1 and N. Let N_0 be the size of this cycle, i.e, $N_0 = \frac{N}{2f}$ and x_0 a random number between 0 and N_0. Then, the final warping function we suggest is:

$$w(x) = \begin{cases} x & \text{if } \ 0 < x \leq x_0 \\ x + S(y)A\sin(\frac{2\pi f(x-x_0)}{N}) & \text{if } \ x_0 < x \leq x_0 + N_0 \\ x & \text{if } \ x_0 + N_0 < x \leq N \end{cases}$$

where y is a random number between $(-1, 1)$ and $S(y)$ returns the sign of y. Notice that $w(x)$ is continuous and monotonically increasing at every segment. Hence, we can conclude that it is monotonically increasing.

Figure 3.6 illustrates the graph of this function when we apply it to a sequence of measures. We observe that it fits the specification as it randomly oscillates around the identity function.



(a) Subtracted from the identity function.

(b) Final warping function.

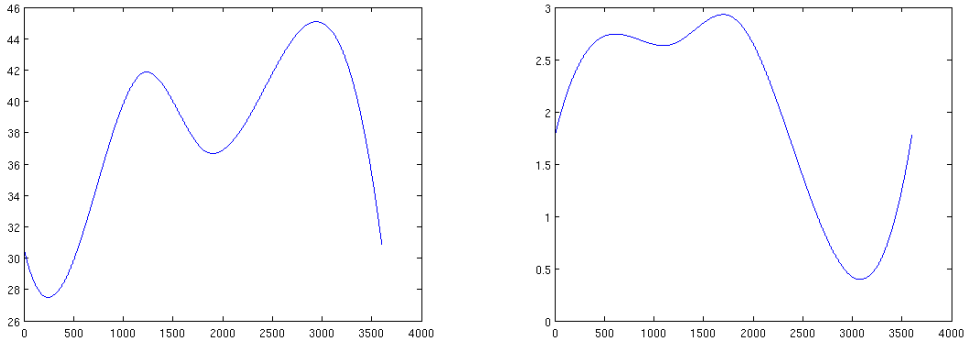
Fig. 3.6: Graph of warping function when warping is done at the measure level.

In the case where we consider that editing can be done after the whole dance sequence is acquired, we make the function random by varying A and f. In this case, the final function is:

$$w(x) = x + A(x)\sin(2\pi f(x)\frac{x}{N}),$$

Proving that this modified function still is monotonically increasing is a little bit more tricky and we will not do this here. However, we argue that if $A(x)$ and $f(x)$ are very smooth functions and $A_{max}\frac{2\pi f_{max}}{N} \ll 1$, then this is likely be the case. We are able to verify empirically that this was true for our chosen values of $A(x)$ and $f(x)$.

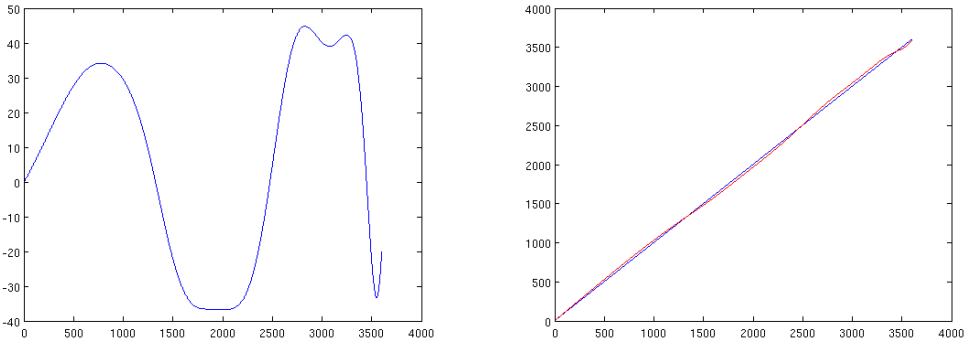
We create $A(x)$ and $f(x)$, by randomly choosing values that they should assume at a very sparse distribution and interpolating these values with a cubic spline. Figures 3.7 illustrate these two function and Figure 3.8 illustrates the resulting warping function.



(a) Graph of $A(x)$.

(b) Graph of $f(x)$.

Fig. 3.7: Graphs of the auxiliary functions.



(a) Subtracted from the identity function.

(b) Final warping function.

Fig. 3.8: Graph of warping function when warping is done for the entire sequence.

Amplitude Variation

In addition to desynchronizing the motion of the different characters, we also felt the need for slightly altering the movements themselves. In truth, when we see a dance spectacle the movement of the individual dancers differs not only because it is impossible for humans to be precisely coordinated, but also because each performer have his/her own style. Ideally, we should be able to understand style and movement as two different components. In this way, we could segment MoCap data so that we could transfer the style from a captured database to another while preserving the steps of the dance. Though efforts have been done in this direction [15, 6], this is still a very difficult open problem and therefore we have not tried to implement it here.

Instead, we took a simpler approach, which was, at the same time, quite effective. In our work, we explore signal processing editing tools to alter the intensity of the movements. In this way, we can have dancers lift their arms lower or higher and make gestures that are more modest or more exaggerated.

Our first experiment was to use low-pass and sharpening filters on the MoCap data. Although, based on previous work [36], we would expect this approach to be successful, using filters proved to be quite unsuitable for our application. We observed, for instance, that low pass filters would either have barely discernible results or would remove the details of the movement, which are an essential part of the dance steps. This happens because our intention is not to exaggerate or remove the details (high-frequency) of the movements, but to alter the intensity of the movements themselves.

It follows from this analysis that the most effective mechanism is to simply alter the amplitude of the signals. Analogous to the time warping function, the amplitude alteration function should also be random and can be applied at the measure level or to the overall resulting motion. Also analogous to the time warping function, this modification can be done at the measure level or to the overall motion sequence.

In this work, we have proposed a rescaling function that oscillates between $1 - R_{max}$ and $1 + R_{max}$. We subtract the mean value of the segment from it and multiply the resulting signal by the scaling function. Then, we add the mean value once again to the resulting motion. Typically we have used $R_{max} = 0.2$.

Empirically, we found that, while observing this threshold was important (since greater rescaling would make the dance look unnatural) intermediary rescaling values were barely noticeable. Hence, we decided to use a function that would randomly assume one of the three values: $1 - R_{max}$, 1, and $1 + R_{max}$. We create this function by randomly selecting these values from a uniform distribution and sparsely distributing them, making sure that they repeat two by two. Then, we interpolate these points using a Piecewise Cubic Hermite Interpolating Polynomial. Figure 3.9 illustrates the resulting rescaling function.

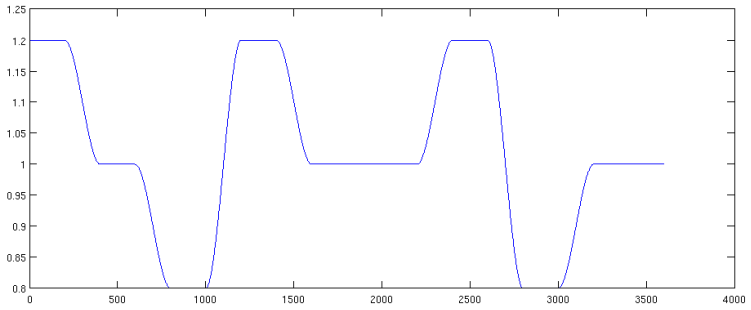


Fig. 3.9: Rescaling function used when working with the full motion sequence.

Notice that to work at the measure level we can simply take each measure and rescale it using a factor of $1 - R_{max}$, 1, or R_{max}. The interpolation that is part of the motion graph connectivity method will naturally take care of smoothing out the artifacts created by using different rescaling values for consecutive measures.

3.4 Combining Motion Segments

In this section, we will describe the mechanisms for searches in the motion graph. As previously stated, we propose a *structured measure-synchronous* motion graph. By *structured* we mean that each clip corresponds to a single step and is annotated with the name of this step and the relevant variations. There are three types of variations: stylistic, structural and locomotion.

Stylistic variations refer to variations that significantly alter the motion in a way that it may be appreciated by untrained spectators as a different dance step. Examples of this would be a jump instead of a walk or changes in the upper body motion. Since their visual effects are very significant, these variations should be specified by the choreographer and, therefore, to the extent that graph search is concerned, they can be interpreted as completely different dance steps. The only important motion synthesis technique that is related to stylistic variations is then one discussed in Section 3.3.1.

The other two variations are thoroughly explored for the combining of motion segments. Locomotion variations refer to changes in the path that the dancer follows while performing a given motion and they are critical to allow group motions. Structural changes correspond to modifications that are essential for graph connectivity. For example, when the same step is performed with the left or right foot or when the beginning and finishing poses are changed to allow different step sequences (see discussion is Section 3.2) .

In synthesizing motion, it is important to follow the two specifications set by the choreographers: the group motion and the sequence of dance steps. Hence, we can create a graph search that guarantees that each dancer is able to follow a specified trajectory and that the dance steps can be combined in the desired order. In what follows, we will discuss these two

aspects individually.

3.4.1 Sequencing Dance Steps

To discuss the sequencing of dance steps, we will interpret the data as clusters of dance components, in which each cluster corresponds to a specific dance step and each component is a structural variation this dance step (see Figure 3.10). As previously argued, the only variations that are relevant to this discussion are the structural changes. Notice that stylistic variations can be described as new clusters and that locomotion variations can be interpreted as iterations of a single dance component.

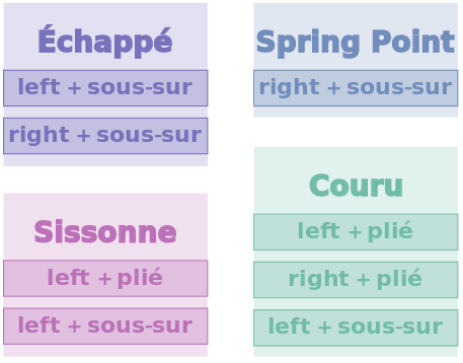


Fig. 3.10: Example of cluster representation.

Since each dance component is associated to a different structural variation, we can determine connections between them and represent them as a graph, as shown in Figure 3.11. Notice that this is a simplification of the actual motion graph, since each component may correspond to several dance clips.

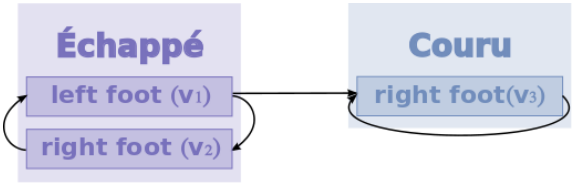


Fig. 3.11: Example of connections between the dance components.

The choreographer determines the sequence of dance steps and our goal is to create a path in the graph that matches this description (i.e., the order of the visited clusters should correspond exactly to the specified order of dance steps). Our approach is to use an modified depth-first search (DFS) algoritm, a widely known method for graph search which is *complete*, i.e., if there is a solution, the search will find it regardless of the kind of graph [30].

Let $d(n)$ be a discrete function, for $0 < n \leq N$, where N is the number of dance steps and $d(n)$ is the nth specified dance step. We want to find a path that has exactly N nodes. Let p be the resulting path, where $p(n)$ is the dance step which the nth node of p corresponds

to. The cost of the path is then the total number of nodes n, such that $p(n) \neq d(n)$.

DFS algorithms exhaustively search the entire graph by expanding and examining each node in the graph, producing a *spanning tree* of the nodes reached during the search. Since our goal is not to reach a specific node , but to find a path, we use the spanning tree structure to find the sequence of visited nodes that has zero cost. Hence, we stop expanding a node when it's depth in the spanning tree is equal to N.

Notice that, since we are interested only in results that match exactly the choreographer's specification, we only want to find zero cost paths. This information is useful because it means that we can abandon the search in a tree node, when we guarantee that the cost is greater than zero for all the subsequent paths. Hence, if we reach a tree node[2] t_0 at the tree level n, such that the dance step which correspond to t_0 is different from $d(n)$, we know that any path that contains this node has least one unmatched step. Since the cost of each path is the total number of unmatched steps, we can guarantee that the total cost of any path that contained this node will be greater or equal to one and, therefore, we do not need to expand this node.

Figure 3.12 illustrates the algorithm for the case where the data is as shown in Figure 3.11 and the desired steps are {Échappé; Échappé; Couru}.

In this work, we assume that the data was captured from a group of dancers that are working together with the choreographers and, therefore, it should always be possible to combine the captured clips in a way that matches the exact specification. However, it would be interesting to be able to find the optimal solution in case the zero cost path did not exist. This is definitely something that should be explored in future work, as this information can be used to give feedback to the dancers, i.e., provide the dancers with a list of steps that should be added to the database in order to allow the synthesis of the desired choreography.

3.4.2 Following Trajectories

One of the essential aspects of this work is to synthesize group motions, i.e., to synthesize the way the dancers move on stage. In the next chapter we will discuss all the technical aspects related to specifying the trajectories that each individual dancer should follow in the 2D plane represented by the stage. However, as far as individual motion synthesis is concerned, we still need to study how to make the dancers follow a given a path.

If we analyze this problem without the discussed framework, it does not seem at all trivial. Basically what we are asking for is a mechanism that allows characters to follow trajectories of different lengths and curvatures, while performing determined steps in synchronization and reaching the objective position at the same time. However, since our data is structured and measure-synchronous, we are able to find a very simple solution to this problem.

Notice that the choreographers' specifications conform to a timeline that is discretized ac-

[2] A tree node is a node in a spanning tree.

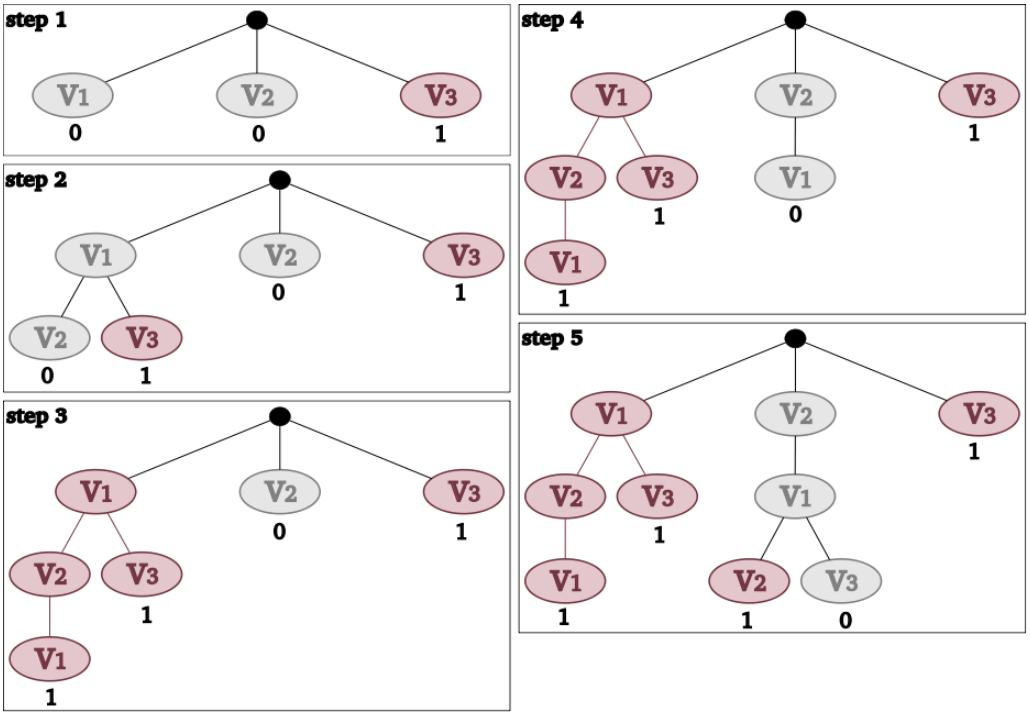


Fig. 3.12: In **Step 1**, we calculate the cost of starting at each of the nodes. Since the first specified step is an Échappé and node v_3 corresponds to a *Courru*, its cost is 1 and we eliminate it. In **Step 2** we pick node v_1 and calculate the cost of the two possible subsequent nodes: v_2 and v_3. Since the second specified step is also an Échappé, we eliminate node v_3. In **Step 3** we observe that the only edge that leaves v_2 goes to v_1 and since its cost is 1, we eliminate this whole branch and go back to the node v_2 on the first tree level. In **Step 4** we notice that the only edge that leaves v_2 goes to v_1 and its cost is 0 and finally, in In **Step 5** we are able to select the path $\{v_2, v_1, v_3\}$ that has zero cost.

cording to musical measures. Therefore, the specified durations of each path are also measure-synchronous and we can create a list of positions and orientations for each individual dancer.

In Figure 3.13, we illustrate this process with an example. Let $p(t) = (x, y)$ be a path the choreographer specified and $N = 3$ the number of measures that the dancer should take to perform it. Also, consider that in the beginning the character is facing the side (i.e. its orientation is $\theta_{\text{begin}} = 90^o$) and that we want it to be facing the public by the end of this trajectory ($\theta_{\text{end}} = 0^o$). We approach this problem by segmenting the path into N segments of the same length and annotating the (x, y) position at the end of each segment (Figure 3.13). We also specify that each measure should have a rotation of $\theta = (\theta_{\text{end}} - \theta_{\text{begin}})/N$. The final specification is therefore a discrete list $S(n) = (x_n, y_n, \theta_n)$, for $n = 1, \ldots, N$. Notice that in a more complex scenario $S(n)$ is formed by a concatenation of multiple path specifications.

With this information, and knowing the dance component each measure corresponds to from the above discussion, we are able to synthesize each measure. We will do this exploring the editing tools for interpolation and rotation discussed in the previous section.

It is important to emphasize that we use the desired final position at the end of each

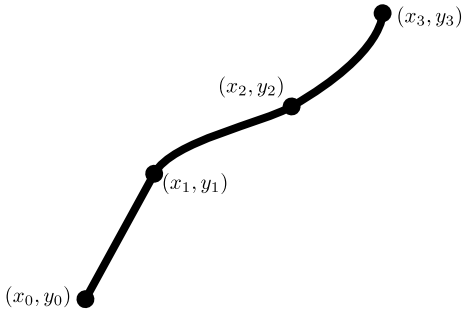


Fig. 3.13: Example of path segmentation.

measure and not the distance that the character should travel at each measure. This is done because we cannot guarantee that the motion we synthesize will follow the exact path, since small errors resulting from restrictions in the database do occur. Hence, if we use distances instead of positions, these errors may add up and we may end up with a trajectory very different from the one we started with. On the other hand if we focus on preserving the position, the small errors will tend cancel each other as the dance grows longer.

For each measure n we have a group of clips that correspond to the dance component we are interested in (each clip with a different locomotion variation). We also have the information of the final position and orientation at the previous measure, $S_{\mathsf{actual}(n-1)}$, and the desired final position and orientation of this measure, $S(n)$. Hence, we can calculate the desired locomotion and rotation for this measure.

Typically, we have only a few iterations of locomotion variation (e.g, one step with no locomotion, a short step moving forward/sideways/backwards and a long step moving forward/sideways/backwards). However, we can enlarge the database significantly by interpolating these different clips. We interpolate all the clips in which the directions are not opposite (e.g., a step moving forward should not be interpolated with a step moving backwards). We can also use infinitely different weights for the interpolation. In this work we found that the weights $(.25, .75)$, $(.5, .5)$, $(.75, .25)$, were sufficient, since changes produced by a finer scale would be barely noticeable.

Finally, we take each of these clips and calculate their total stage locomotion when the desired rotation is inserted and choose the one that best matches our solution.

We assume that if a dance step is to be performed while the characters are moving significantly along the stage, then some of the clips in our database will have locomotion of different lengths and in different directions. If this is not true and we are unable to find a suitable clip at some point, we may still be able to render the desired motion if the errors at each measure cancel themselves out, as previously mentioned. However, if we observe that the errors in the stage position are increasing during the motion synthesis and surpass a certain threshold, our system informs this to the choreographer, who will then ask the dancer to perform the additional necessary motions.

Chapter 4

Group Motion

In this chapter we will discuss the techniques involved in synthesizing group motions. After reviewing the area and related work (Section 4.1), we will discuss our approach and aspects regarding the implementation of the interface (Section 4.2). We then detail the declarative (Section 4.3) and procedural (Section 4.4) techniques for group motion synthesis.

4.1 Overview and Related Work

When viewed together, a group of dancers moving on stage gives the impression of a dynamic, complex object. Therefore, an important aspect of dance motion synthesis is determining how the dancers move as a group, creating formations, following trajectories, interacting without colliding, etc.

Several works have proposed methods for simulating the motion of a group of objects that together assume an *emergent behavior* [24]. *Particle system* techniques are often used to simulate clouds, water, fire or very large crowds. Typically, the particles have a finite life span, being generated and terminated during the animation. These methods deal with a very large number of objects and, therefore, make simplified assumptions for controlling motions (e.g., using physical models which include gravity, collision, and other local and global forces).

Other techniques for controlling group motions involve autonomous behavior models. Such methods often account for and rely less on physics and more on artificial intelligence. These models can be quite complex and applications range from few participants to single character control. Typically, members have a limited "vision" allowing them to perceive the environment and the other members and, thus, plan reactions to their circumstances.

Control mechanisms for simulating small groups (as is the case of dancers on stage) can be considered an intermediary level between the two previous strategies. In this scenario, characters behave according to more sophisticated physics and only a restricted amount of intelligence. Such mechanisms are often used to simulate groups of characters, such as herds or flocks, allowing the synthesis of lifelike motion with very little input. Previous research in

this area is, however, somewhat more scarce than the other two.

These models are largely referred to as *procedural animation*, where the motions are calculated at each state based on conditions of the previous state and updating rules. Although these methods are very efficient for generating motions that are more natural or "organic", they allow less control over individual frames.

For this reason, we have explored both procedural and declarative methods. The declarative methods involve direct specifications of movements and are therefore very much dependent on the requirements of the specific application (in this case, dance). For this reason, related research is fairly limited. In terms of procedural animation tools, the previous works resources are much richer. In our approach for procedural group dance animation, we have widely explored Craig Reynold's work on *steering behaviors* [29, 28].

4.1.1 Steering Behaviors

In [29, 28], the motion of autonomous characters, or *agents*, are divided into three layers: *action selection* (considering goals and strategies and planning the motion), *steering* (determining the path), and *locomotion* (actual animation).

In terms of *locomotion*, the agents (in this case, the dancers) are viewd as simple "vehicles" that move in a manifold (in this case, a 2D manifold represented by the stage). These vehicles are very simple structures that can only translate and rotate, therefore determining the position of the individual dancers on stage and their orientation.

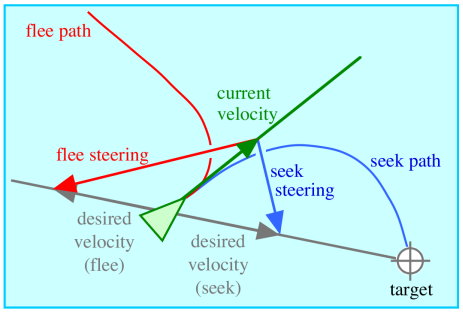


Fig. 4.1: Steering force: difference between the desired velocity and the current one (seek and flee example extracted from [28]).

The *steering* behaviors are described in terms of a vector that represents the desired steering force, as shown in Figure 4.1. There are several different types of forces that allow agents to reach different goals. We will describe here the ones that were more significant during our experiments with dance (see Figure 4.2):

Seek: adjusts the agent so that its velocity is radially aligned with a specific point (target).

Flee: the inverse of seek.

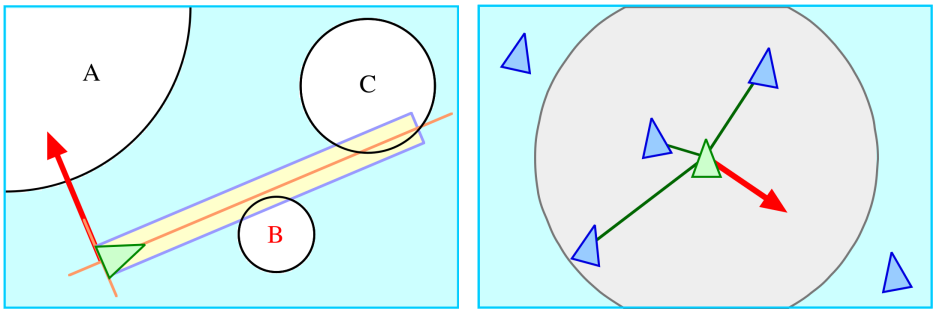
Obstacle avoidance: creates a fleeing behavior only when the obstacles are directly in the path of the agent and both are sufficiently close.

Wandering: creates a random walk by making small random displacements to the steering direction at each frame.

Flow field following: steers the character to align its motion with the tangent of a vector field.

Unaligned collision avoidance: calculates potential collisions of moving agents and steers the characters to avoid the site of the nearest predicted collisions.

Separation: combines repulsion forces from other characters within a specific neighborhood (usually used for flocking behavior animation together with cohesion and alignment steering.)



(a) Obstacle avoidance (b) Separation

Fig. 4.2: Examples of steering behaviors (extracted from [28]).

Finally, the *action selection* layer consists in combining the steering forces in order to allow agents to reach higher level goals. The are two ways of combining behaviors: blending steering forces together or switching between different behavior modes according to circumstances. Both of them can be very efficient depending on the situation. For example, it may be necessary to ignore a seeking force momentarily to control collision avoidance. On the other hand, blending flow field following and separation behaviors may be quite effective. There are several ways in which blending can be done, however simple weighted linear combinations have been shown to work very well.

4.2 Our Approach

Since our goal is to propose a natural environment for dance design, it is important to develop an interface that takes into account the standard specification methods that choreographers use. Also, since we want to create not only a tool that makes dance specification simple, but also one that allows novel creative expressions, we draw ideas from related work in procedural animation to suggest different ways of specifying dance.

Hence, in this work, we propose a series of declarative and procedural methods for group motion specification and we will describe both of these methods in the following sections.

The interface for choreographic input was developed in C++ using the Qt toolkit. The

basic elements and usage of the interface have been described in Chapter 2. There are, however, a couple of important implementation aspects related to group motion specifications that are worth describing in further detail.

4.2.1 Interface Implementation Aspects

Abstraction Levels

We interpret group motion in three levels of abstraction[1]: *motion intention*, *motion segment*, and *locomotion*.

Motion intention refers to the essential identity or objective of the motion. For example: assuming a given formation, following a given trajectory, blending between formations using a given optimization method, following a given flow fied with collision avoidance, etc. The *motion intention* contains the essential specifications of the motion, but not its contextualization as part of the overall dance.

We create a *motion segment*, by taking *motion intention* specifications and determining how it fits in the dance timeline. For example, for how long the dancers assume a given position or trajectory, what are the initial positions that should be blended, to which subgroup of dancers these specifications are related to, etc. In our interface, this refers to dragging, positioning and sizing the elements in the sequence of small boxes that are part of the authoring timeline and correspond to the musical measures.

Finally, after the sequence of *motion segment* are combined, we can determine the actual *locomotion*, i.e., the 2D stage position and orientation of each character at each frame.

Data Structure

Although the *locomotion* level contains all the necessary information for motion synthesis we have chosen to create a data storage format that aggregates all the three abstraction levels. This is very important because it allows us to distinguish between the different inputs and therefore allow the data to be reused (e.g, a specific formation may occur several times during the dance) as well as edited. In an application that suggests iterative contributions of different artists, it is essential to permit editing.

The most essential element of our data structure is a list of *motion segments*. Notice that each motion segment refers to an *evolution* (see characterization in Section 2.2.2) and can, therefore, be of one of the three kinds: fixed formations, boundary conditions, and initial conditions. The different types of *motion segments* correspond to different *motion intentions* and have, therefore, different attributes.

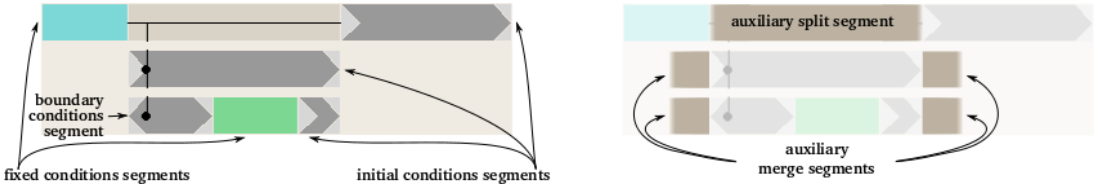
Since motion intentions can be viewed as attributes of motions segments, they don't have to necessarily be stored separately. However, the advantage of creating a data representation

[1]Notice that we could draw a parallel of this division with Cray Reinold's motion layers.

of motions intentions is that, in this way, they can be referenced by more than one segment. In our previous discussions of the group dance specifications (see Chapter 2), we verified that formations are one of the essential elements that choreographers must design. For this reason, we decided to store formations separately, allowing them to be repeated effortlessly during the dance. This way, if the choreographer chooses to edit a formation latter on in the dance, all the segments that refer to it will be altered automatically.

The *locomotion* information is stored as the list of 2D positions and orientations described in Section 3.4.2. The latter is regenerated after the edited and guides the synthesis of the articulated motion.

The final element of our data structure is the hierarchical representation of the different groupings. We store this information by creating auxiliary *motion segments* every time a group is subdivided, as shown in Figure 4.3. These segments indicate how the groups are connected and where the divisions take place. This way, instead of having different fragments, we maintain the format of sequenced *motion sequences* throughout the whole timeline simplifying the implementation of auxiliary functions.



(a) Original *motion segments*.

(b) Auxiliary *motion segments*

Fig. 4.3: Examples of *motion segments* created for hierarchical representation.

4.3 Declarative Methods

In this section, we will discuss the declarative methods proposed for group motion synthesis. We will use the formalism given in Chapter 2, considering formation and evolutions and segmentations.

4.3.1 Formations

Formations refer to they way dancers occupy the stage and are maybe the most essential element that choreographers explore for creating different group motions. Determining formations involves positioning the characters on the stage and specifying relative distances between them. Some interesting effects of editing formations include positioning the main dancer in a more prominent place and designing shapes on the stage (e.g., ballerinas organized in concentric circles forming the shape of a flower). We specify formations by determining shape, density, and pattern.

Shape refers to the geometry that the group of dancers assume on the floor. For input, we allow the choreographer to draw lines or sketches on the canvas represented by the stage. Examples of shapes are illustrated in Figure 4.4. Notice that it is also important to specify whether the dancers will be placed on or inside the drawn lines.

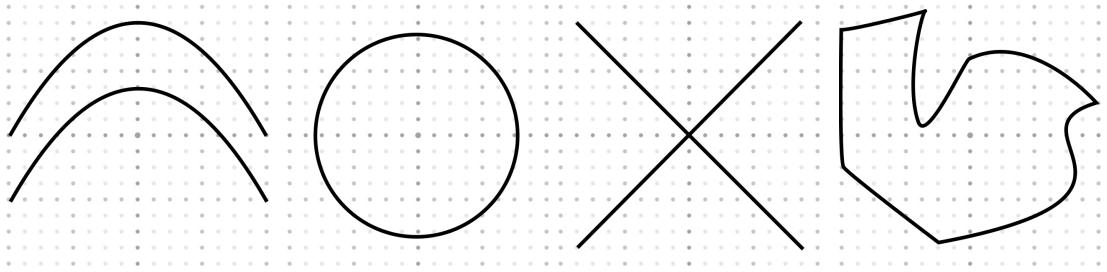
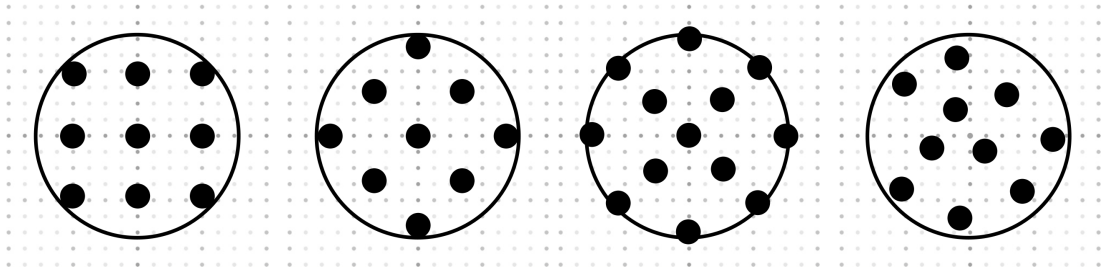


Fig. 4.4: Examples of different shapes.

Density refers to the number of dancers that will occupy a given stage area.

Pattern refers to the way the dancers are distributed on the designed shapes. For example, the dancers could be equally distributed along the lines or more strongly concentrated in different areas. In closed shapes the distribution of the dancers becomes even more interesting, as not only different concentrations are plausible but also grid arrangements. Figure 4.5 illustrates different patterns within the same shape.



(a) Uniform grid (b) Diagonal grid (c) Concentric silhouettes (d) Poisson disk sampling

Fig. 4.5: Examples of different patterns on a circle.

Finally, we also observe that choreographers have typically explored symmetries in these specifications. We allow the choreographer to specify them by drawing a symmetry line and reflecting over it the elements that are drawn on either side (see Figure 4.6).

4.3.2 Evolutions

We consider three types of group motions: fixed conditions, boundary conditions and initial conditions.

Each *motion segment* belongs to one of these three major categories and we distinguish between them on the interface by representing formations as colored blocks, movements based

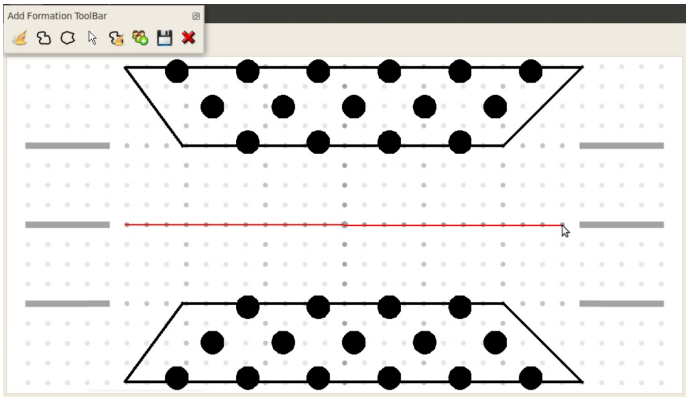


Fig. 4.6: Symmetry example.

on boundary conditions as hexagons, and movements determined by an initial condition as arrows (see Figure 4.7).

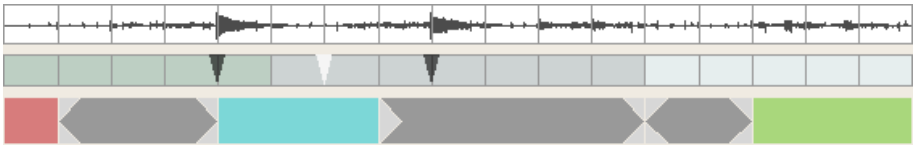


Fig. 4.7: Example of motion segments. Notice that the colors distinguish between different formations.

Fixed Conditions

We determine a fixed condition by referencing one of the designed formations. Notice, however, that a formation by itself is only a motion intention. Hence, we allow choreographers to specify how this intention becomes a part of the dance by determining how it fits within the timeline.

Boundary Conditions

Choreographers usually determine initial and final sets of positions and orientations of the dancers on stage and want to create a group motion that allows them to naturally evolve from one to the other. These initial and final conditions can be specified formations or the final setting that results from a different group motion.

There are three basic steps in creating a motion based on boundary conditions: matching the two sets of positions, synthesizing a trajectory, and determining the orientations.

Although in many situations choreographers may want to declare the matching themselves using the IDs of each performer, in most cases they are simply interested in determining the positions of a group or subgroup. Hence, they choose which dancer goes where by somehow optimizing the path that they should follow. The most typical method is choosing a matching between the dancers that guarantees that the overall distances traveled will be minimum.

We implement this idea using a bipartite graph matching algorithm [7]. A bipartite graph is a graph whose vertices can be divided into two disjoint sets U and V such that every edge connects a vertex in U to one in V (see Figure 4.8). Each vertex in our graph refers to an individual position on stage, the ones in U referring to the initial and the ones in V to the final ones. Our graph is *complete* because we create an edge from every initial to every final position and *weighted*, because we determine a value for each edge (namely, the cost associated to the trajectory it determines). We can thus calculate the minimum edge weight matching using the Hungarian Algorithm and, in this way, determine the optimal matching between the initial and final positions.

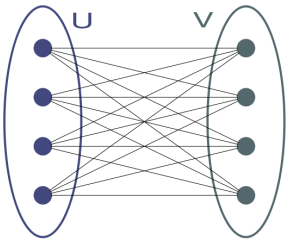


Fig. 4.8: Example of a bipartite graph.

Of course, the cost function being optimized is determined by the weights of each edge. We can set these weights to be equivalent to the distance between the two positions, i.e, :

$$W_{\text{overall minimal distance}} = \|P_i - P_j\|\,,$$

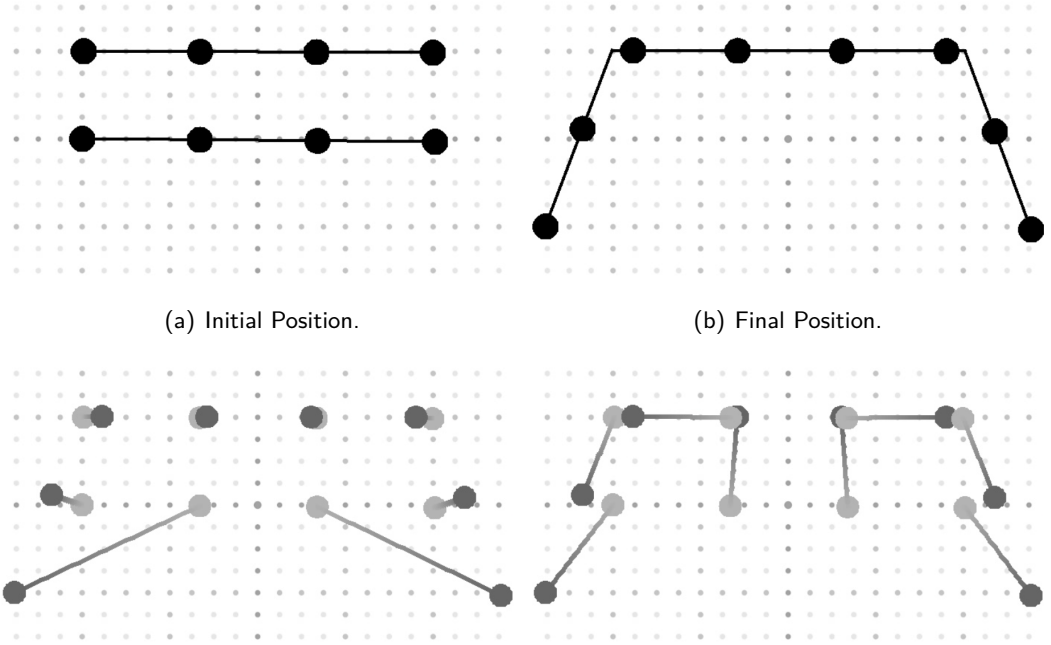
where P_i and P_j are, respectively, the 2D positions of the initial and final points connected by this edge. Notice that, in this case, the algorithm returns the matching that guarantees that, if the dancers travel in a straight line, the sum of all the trajectories will be minimum.

In several cases, this is precisely what choreographers desire. However, in many circumstances this may result in a few dancers moving far along the stage while the others stay at the same place. Hence, depending on the scenario, it may be more important to guarantee that the distance traveled by the individual dancers are optimally equivalent. To implement this, we should specify the weight of each edge as the norm of the difference between the distance traveled with this matching and the average distance traveled by the dancers:

$$W_{\text{equal distances}} = \left\| W_{\text{overall minimal distance}} - \bar{D} \right\|,$$

where $\bar{D}$ is the average traveled distance.

Calculating the actual average distance is challenging since it naturally depends on the chosen matching. However, we can approximate it by taking the average of the distances traveled when we choose the matching that results from optimizing the overall minimal distance. Empirically, we have observed that this yields very interesting results, as can be seen in Figure 4.9.



(a) Initial Position. (b) Final Position.

(c) Optimizing overall minimal distance. (d) Optimizing equal distances.

Fig. 4.9: Examples of different patterns on a circle.

Notice that we could also have a cost function that results from adding the previous two calculated weights:

$$W_{\text{optimizing both}} = W_{\text{overall minimal distance}} + W_{\text{equal distances}}$$

This would result in optimizing both the minimum distances and the equivalence between the trajectories at the same time. Of course, we could also have a weighted average instead of a sum. Empirically, however, we observed that, since we usually work with a small number of dancers (usually not greater than 50), different weights do not produce very significantly different results. For this reason, we choose to make the interface simpler and only allow choreographers to choose between one of these three options (see Figure 4.10).

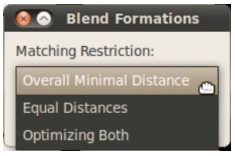
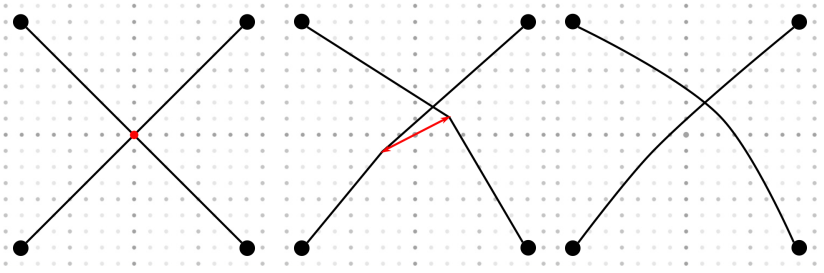


Fig. 4.10: Window for creating group motions based on boundary conditions.

After matching the initial and the final positions we have to create a path that connects them. Although straight lines are the natural choice and the most efficient one in most cases, they may result in collisions. Hence we implemented a simple method for collision avoidance based on [25].

The technique is based on a "divide and conquer" approach. For each pair of paths, we check if there is a collision half way along the trajectory. If there is, we choose a random direction and shift both paths according to it, as shown in Figure 4.11. We repeat this process for the $1/4$ and $3/4$ positions, and so on. Afterwards, we smooth the resulting path.



(a) Initial paths. (b) Shift on random direction. (c) Resulting smoothed path.

Fig. 4.11: Collision avoidance method.

Finally, we have to establish the orientation of the dancers during the path. This can be done in several different ways and has a very strong impact on the overall result. In this work, we have suggested two ways of automatically determining the orientations. Of course, we could have any other results from declarative specifications.

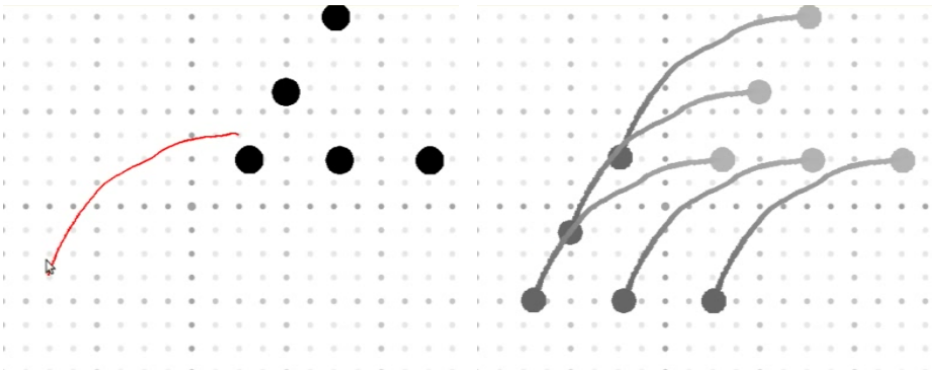
The first method we suggested is a simple linear interpolation between the initial and final orientations, allowing the dancers to smoothly rotate while translating on the stage. The other standard rotation method we developed is to make the orientations tangent to the direction lines. This usually results on the dancers rotating before the motion starts, then traveling facing their paths and rotating again, at the end of the motion, to face the determined final orientation.

Initial Conditions

The final type of group motion we explored is the one based on initial condition and evolution rules. This allows choreographers to plan very complex movements based not on the final positions, but on the actual motion effect that they desire.

In terms of declarative rules, we implemented pauses and trajectories. Pauses refer to simply making the dancers maintain the initial condition formation (i.e., the formation that resulted from the previous group motion) for a longer period of time. Although simple, creating pauses is very useful. Trajectories are designed by allowing the choreographers to sketch a path on the stage (see Figure 4.12). The orientations during these trajectories can be specified along the path, or we can use one of the automatic methods described on the previous section.

Movements based on initial condition and evolution rules are perhaps the richest of all three types of group motion specifications. This happens because we can have not only declarative



(a) Defining sketch. (b) Resulting trajectory.

Fig. 4.12: Example of a trajectory specification.

rules but also rules drawn from procedural animation methods as we will describe in Section 4.4.

4.3.3 Segmentation

In many cases, it may be desirable to divide the group of dancers into smaller subgroups and specify different group motions for each one of those. This grouping may depend on the identity of each performer, (e.g., groupings based on gender or height) or simply on position (e.g., specifying symmetrically opposite motions for dancers who are on the left and right sides of the stage).

Groups of dancers can be divided and subdivided in different ways through the dance. To allow choreographers to easily specify these groupings we developed a hierarchical structure for arranging and rearranging dancers into subgroups that are specified either based on individual identities or positions on stage.

We have a list of dancers, each of which is annotated with a specific ID. The choreographers specify subgroups of dancers by declaring different attributes, such as ability (e.g, a main dancer versus the chorus), gender, costumes, etc. Hence, each dancer has a list of attributes, or groups to which it belongs. Since these are general characteristics attributed to each dancer, we can divide and redivide a subgroup based on any segmentation (e.g., take the main dancers and divide it into two groups based on gender). An example of this structure is shown in Figure 4.13.

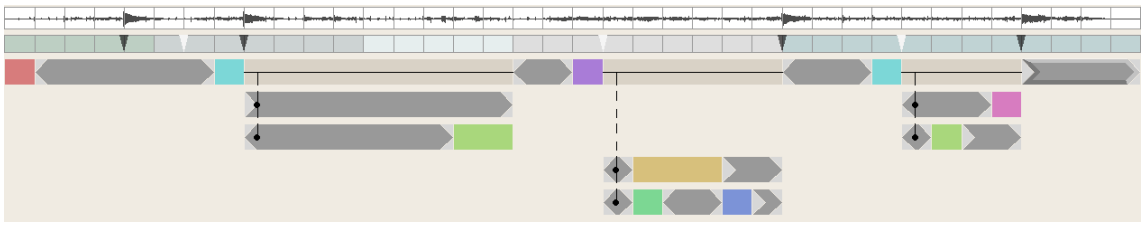


Fig. 4.13: Example of groupings along the timeline. Notice how we can create different segmentations for the entire group, but also repeat groupings throughout the dance.

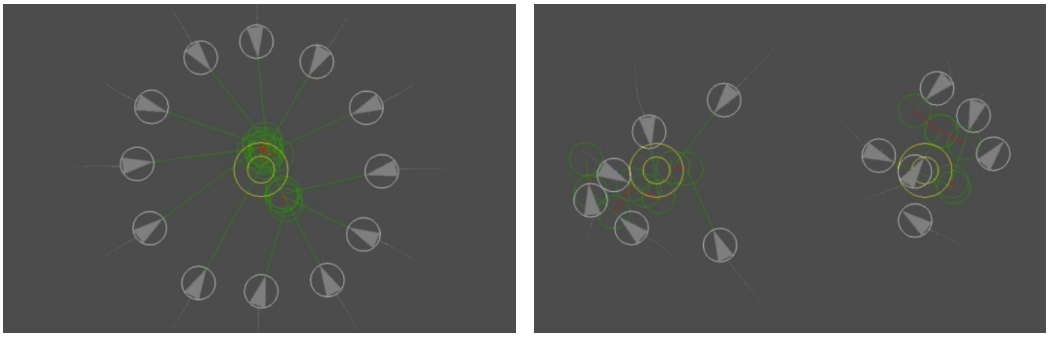
4.4 Procedural Methods

In this work we explored behavioral animation methods for specifying evolution rules. As discussed in Section 4.1.1, these methods regard each dancer as an autonomous agent that travels along the 2D manifold represented by the stage according to combined steering forces.

There are, of course, innumerable evolution rules that could be designed based on these behaviors. For the purpose of illustrating the applications of this method, we have implemented the following group motions: following attraction/repulsion forces, spreading out on the stage, and crossing over.

4.4.1 Following Attraction and Repulsion Forces

The choreographers specify attraction and repulsion forces by positioning attractors or repulsors on stage and determining for how long their effects will last (i.e, setting the measures in which they are active). Notice that the same hierarchical structure discussed previously is still valid in this context, allowing different groups to be drawn to different attractors (see Figure 4.14).

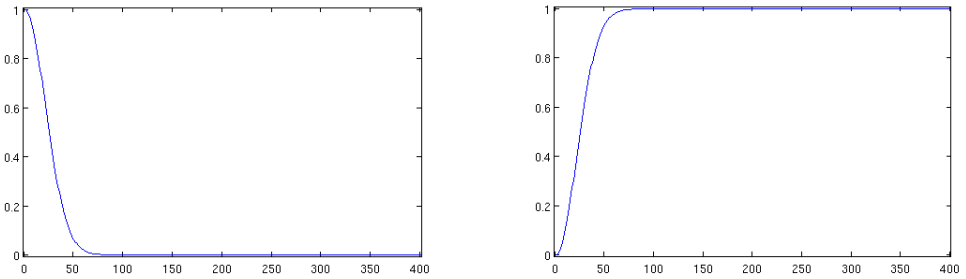


(a) One attractor. (b) Two attractors.

Fig. 4.14: Example of attractors. In (a) the whole group is drawn to a single attractor, while in (b) two sets of subgroups are drawn to two different attraction forces.

We use an exponentially decaying function, $f(x)$, to control the repulsion forces, making them more intense the closer the agents are to the source. Analogously, we use the function $1-f(x)$ for attractors, making the agents move faster towards the attractors when they are far away and slow down as they approach the destination. We plot the graphs of these functions in Figure 4.15. Empirically, we observed that this creates a more natural group motion.

It is important to point out that attractors and repulsors are single points on the stage. However, we can create regional attraction/repulsion forces by combining these elements. For example, we can draw dancers to a straight line by aligning a set of attraction forces, as shown in Figure 4.16. We combine the forces of multiple attractors/repulsors by selecting the one which is nearest to each autonomous agent. Notice that this is much more effective than linearly interpolating the forces because such interpolation would be equivalent to a force



(a) Repulsion control function. (b) Attraction control function.

Fig. 4.15: Functions that control repulsion (b) and attraction (a) forces.

directed toward the center of mass of all the elements.

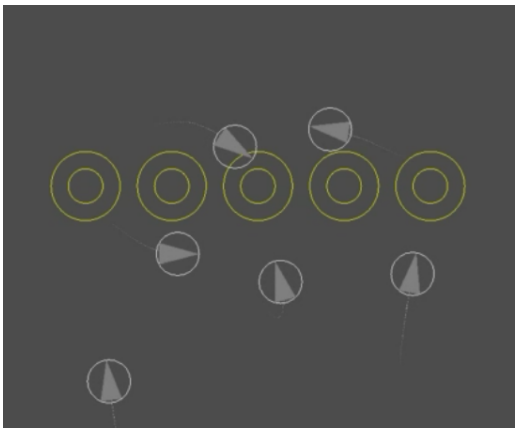


Fig. 4.16: Example of a combination of attractors.

While following these forces, it is essential to control collisions with neighboring dancers and obstacles (which are usually the edges of the stage, but could also be any other item in the set). We also use separating behaviors which guarantee that the dancers will not be cramped together during the dance.

To combine these steering forces we observe that, when a collision is imminent, avoiding it should take precedence over all other behaviors, i.e., we should steer the characters to avoid collisions and ignore the other driving forces momentarily. As discussed in Section 4.1.1, collision avoidance forces are active only when the obstacles are "sufficiently close". However, determining this minimal distance is not trivial and highly influences the resulting motion. If we choose a distance which is too small, there may not be enough time for the steering to successfully prevent the collision. On the other hand, if we make this parameter too large, we will lose the attraction/repulsion behaviors since their corresponding steering forces will be inactive for too long.

To get around this problem, we suggest specifying two minimal distance parameters: a small one, d_s, and a larger one d_f. If the distance between the colliding objects is smaller

than d_s, we ignore the other behaviors and steer the characters to avoid collisions. However, if the distance is smaller than d_f, but larger than d_s, we combine all the steering forces allowing the character to slowly move away from the obstacles, while continuing to follow the other behaviors. We also suggest limiting the velocity of the agents when the distance is smaller than d_s, making the characters slow down when they get close to a potential collision. This change of speed was very effective, because it allowed use to use a smaller value for d_s while still successfully avoid collision.

4.4.2 Spreading Out on the Stage

Having the characters spread out on the stage assuming random positions is a typical specification used even in very classic dances, such as the ballet. This is also something that is hard to be accomplished by an unexperienced group of dancers, since they tend to be crowded up in certain areas leaving other regions of the stage empty[2].

Our first approach in creating spreading motions using procedural methods was combining separation and wandering behaviors with collision control. Though this had an effective result in the sense that the characters assumed random positions without colliding, we ended up with the same effect that results when real dancers occupy the stage with random motions: several areas become overpopulated while others are empty.

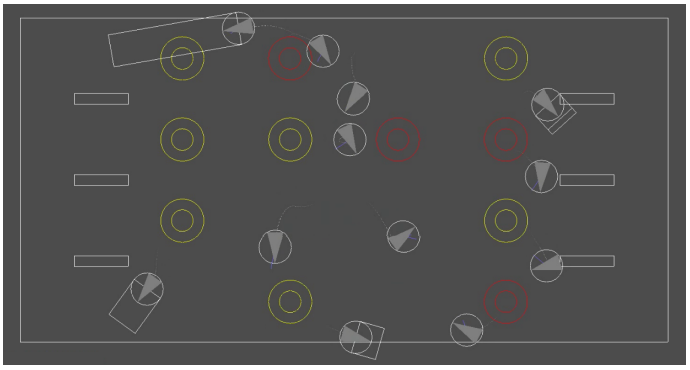


Fig. 4.17: Example of characters spreading out on the stage.

To create a spending that guarantees at the same time randomness and a uniform distribution of the dancers, we propose a method based on segmenting the stage into a grid. The idea is that each block of this grid should be occupied by an approximate number of dancers. Hence, if it is overpopulated we place a repulsor in the center of the grid and if it is underpopulated, we place an attractor (see Figure 4.17). We combine the attraction/repulsion forces with wandering and separation behaviors. We also use the collision avoidance method described in the previous section. We observed that this method was much more effective than

[2]We learned this from conversations with the dancer and choreographer Raquel Leão, who enthusiastically encouraged us to make experiments with spreading motions.

the previous one.

4.4.3 Crossing Over

Crossing over refers to specifying a line that divides the stage into two parts and have the dancers migrate from one of them to the other. We do this by defining two symmetrically opposite flow fields, which are parallel to the specification line. Hence, dancers that are on one side of the line are drawn to the other, and vice-versa (see Figure 4.18).

This behavior is also combined with the collision avoidance forces described above. It should be emphasized that, in this scenario, the concept of distinguishing between potential collisions which are closer or further away is very important. Since the characters have to move facing each other, if potential collision forces made the characters steer in other directions, they would never be able to complete this motion.

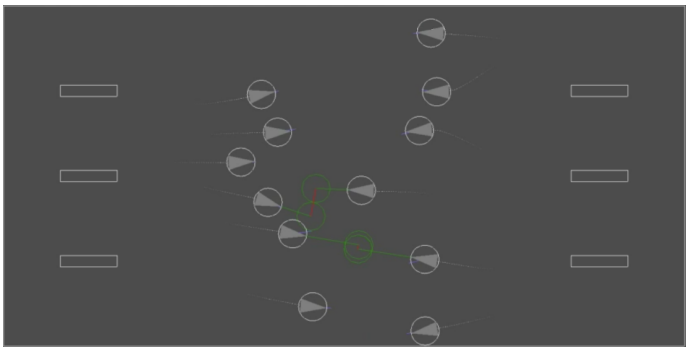


Fig. 4.18: Example of characters crossing over.

Chapter 5

Experiments

In this section, we will discuss the experiments that were made during the course of this work and that exemplify applications of our research. In each experiment, we have tried to explore a different aspect of our authoring platform. The videos that illustrate them are available at `www.impa.br/~aschulz/ChoreoGraphics/videos`.

5.1 Experiment 1: Motion Editing Tools

In the first experiment, we explored creating a *measure-synchronous* motion graph as well as editing tools for altering MoCap data.

We used a samba style dance captured from Vanessa Simões, who was asked to dance to three songs with the same time signature composed by Marcello Cicconet. We created a simple (not structured) *measure-synchronous* motion graph by segmenting the data according to musical measures and observing segment boundary similarities. Hence, we were able to synthesize new dance sequences by following a random walk on this graph. We found that, since we connected the segments obeying the measure structure, the final dance motion could be synchronized with a song that has this metric [31]. We thus validated the idea that segmentation based on musical measures is efficient for designing dance shows.

With the resulting animation, we created a chorus line by replicating the motion and inserting small alterations to make the group dance more natural. We used the amplitude variation and time warping methods discussed in Section 3.3.3.

In addition, we observed that, since segments were randomly combined, the orientation of the resulting motion was also random. This made the result seem somewhat unusual, for chorus lines usually face the public. To resolve this problem, we calculated the average orientation of the motion sequence and rotated the characters so that the average orientation became zero. Though this significantly improved the results, the dancers still seemed to be staring at strange directions. Therefore, we subsequently modified the rotation of the neck, to give the impression that they where facing the public while dancing.

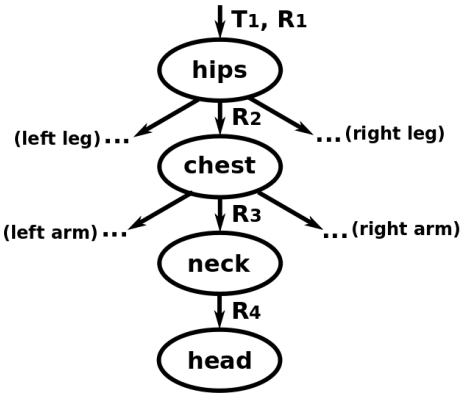


Fig. 5.1: Hierarchical structure.

To do this, we first calculate the rotation θ around the y-axis that should be applied to the neck joint at each frame in order to guarantee that the character is facing the stage. Since the motion is represented by the hierarchical structure shown in Figure 5.1, we know that the overall rotation of the neck is:

$$R_{\text{head}} = R_4 R_3 R_2 R_1,$$

Therefore, we can calculate the orientation vector of the head (see Figure 5.2):

$$[v_x \ v_y \ v_z] = [1 \ 0 \ 0] R_{\text{head}}$$

and the desired theta:

$$\theta = -\text{arctg}\left(\frac{v_z}{v_x}\right)$$

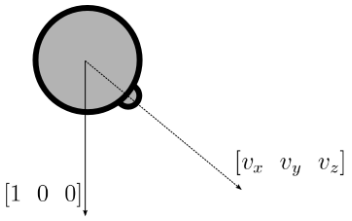
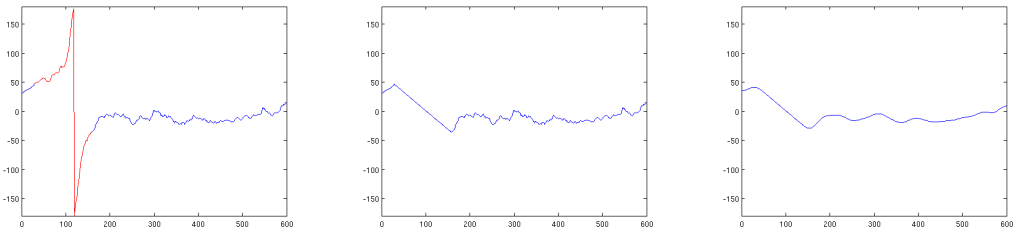


Fig. 5.2: Head orientation.

Notice that the character's orientation varies significantly, since dancers turn and even spin while dancing. Hence, we should be careful not to apply this transformation directly, so that we do not generate impossible human movements. In this work, we chose to threshold θ, so that the final head rotation ranged between -60 and 60 degrees. Also, since we calculated the θ frame by frame, it is important to smooth the resulting $\theta(t)$ function.

Hence, we processed the data by first eliminating the regions where $\theta(t)$ surpassed the threshold and filling in the gaps with a liner interpolation. Then, we applied a Gaussian filter to remove the high frequencies (see Figure 5.3).



(a) Threshold. (b) Interpolations. (c) Smoothing.

Fig. 5.3: Steps in processing the additional rotation value (frames 40 to 140 refers to a spin movement).

Finally, after calculating θ we had to alter the hierarchical representation of the motion by determining a new value for R_4. Since the new value for R_{head} is $R'_{\text{head}} = R_4 R_3 R_2 R_1 R_\theta$, then $R'_4 = R'_{\text{head}} R_1^{-1} R_2^{-1} R_3^{-1}$.

5.2 Experiment 2: Declarative Group Motions

In this experiment, we explored the use of declarative methods for specifying the group motions. We synthesized the movement of the characters using the discussed techniques for combining motion segments in order to allow the dancers to follow trajectories on stage.

We specified a very simple group motion inspired by the Irish dance group "Lord of the Dance". We created three formations shown in Figure 5.4 and interpolated them optimizing the overall minimum distances (see Section 4.3.2).

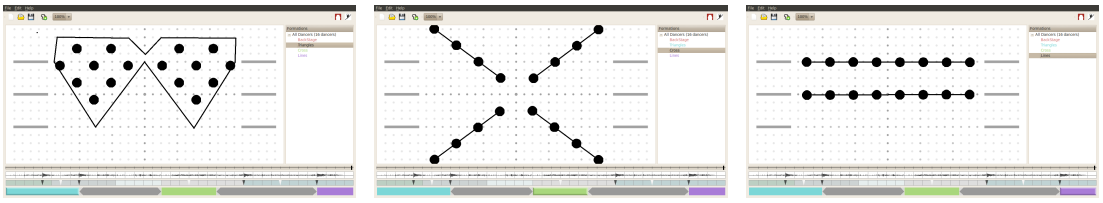


Fig. 5.4: Specified formations.

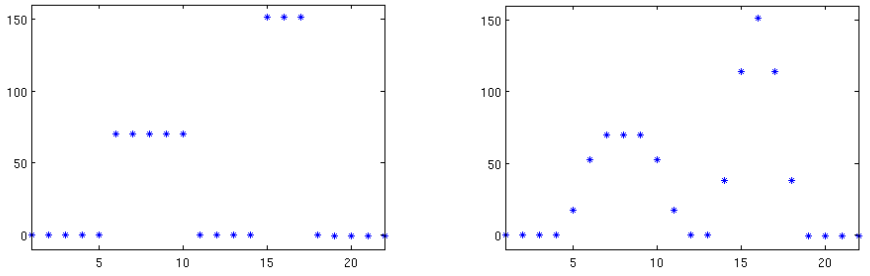
We specified orientations by assuming that the dancers would face the public during a formation and that they would face the trajectory while moving along the stage. Hence, we specified the orientation as being zero if no locomotion was determined and tangent to the directions lines, otherwise. As a result, the dancers rotate to face the path before starting the trajectory and then rotate again at the end of the motion to face the stage once more.

We observed, however, that this specification made the dancer turn very abruptly at the beginning and at end of each trajectory. To solve the problem, we suggested smoothing the orientations values, with a weighted mean filter:

$$\mathcal{O}^{'}(n) = 0.25\mathcal{O}(n-1) + 0.5\mathcal{O}(n) + 0.25\mathcal{O}(n+1),$$

where $\mathcal{O}(n)$ is the function that represented the specified orientation of the dancer at the musical measure n

Notice that, to synthesize motions that observe these orientations specifications, we have to apply a rotation at each measure n which is the difference between $\mathcal{O}(n)$ and $\mathcal{O}(n-1)$. Hence, this smoothing allows us to distribute the rotations between neighboring measures. Figure 5.5 illustrates this with an example.



(a) Original values. (b) Filtered values.

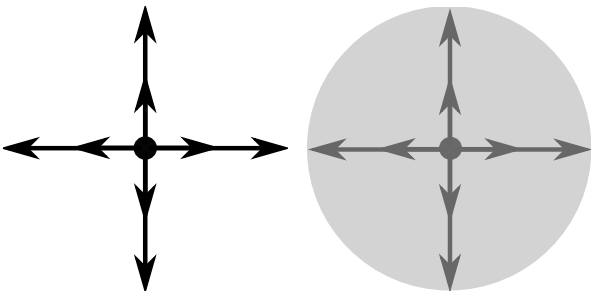
Fig. 5.5: Filtering example. Notice that for the first five musical measures the dancer is facing the public and, for the next five measures, he is facing sideways. In figure (a), we illustrate how these original values would suggest an 80^o rotation at the sixth frame. In figure (b), we show how we can distribute this rotation using a mean filter. As a result, the character prepares the spinning move at the fifth measure with a small rotation of 20^o. Then, at the sixth measure, he performs the major part of the spin by rotating 40^o. Finally, at the seventh measure, he concludes the motion with another 20^o rotation.

We created a structured measure-synchronous motion graph from capturing a single tap step called the *time step*, which was performed with the left and right foot (see Figure 5.6). In this example, the dancer was the author of this thesis.

Fig. 5.6: Graph representation

We chose a time signature for our composition, created a "tack-tack" audio signal that counted the music beats, and used it to guide the MoCap session. We asked the dancer to perform the step while moving in nine different directions/distances. Then, we interpolated

these motions to allow the dancers to move in approximately any 2D direction, with the step lengths that ranged from zero to the largest performed step (see Figure 5.7).



(a) Captured directions (b) Interpolated directions

Fig. 5.7: In (a) we show the nine different captured directions: one step with no locomotion, four short and four long steps with different directions (forward, right side, left side, backwards). In (b) we illustrate the approximate region of the stage where we are able to move the character with one single step.

With the database created with both the captured and the interpolated motions, and using the rotation mechanisms described in Section 3.3.2, we were able to synthesize the trajectories on stage. Finally, we applied the amplitude variation and time warping methods discussed above to make the group dance appear more natural.

An interesting aspect of this experiment is the relationship with music. Since we started out only with the time signature, we could have a musician create any kind of music that matched this dance. The final video was fashioned using a song called The Lark in the Morning, from James Morrison, that Marcelo Cicconnet edited to fit the resulting dance. We also captured the sound tap dance shoes made while the dancer performed the *time step* and used it to create an interesting audio effect.

5.3 Experiment 3: Procedural Techniques

In this experiment, we explored procedural animation methods for synthesizing group motions. Although we could combine declarative and procedural specifications, we chose to design the entire dance only using behavioral animation methods.

This experiment was done in collaboration with the dancer and choreographer Raquel Leão and the musician Marcelo Cicconet. We started out by defining to the choreographer which were the procedural elements she could use to create the group motions. We suggested four elements: attractors (create attraction forces), repulsors (create repulsion forces), spreaders (stimulate the dancers to spread out on the stage), and crossing-over bars (define a line that divides the stage into two parts and forces which draw the dances to migrate from one side to the other).

The choreographer then suggested a first version of the group motion, designing a storyboard which contained these elements and their duration, i.e., for how many measures the effects created by each of them should last. Figure 5.8 illustrates this first storyboard.

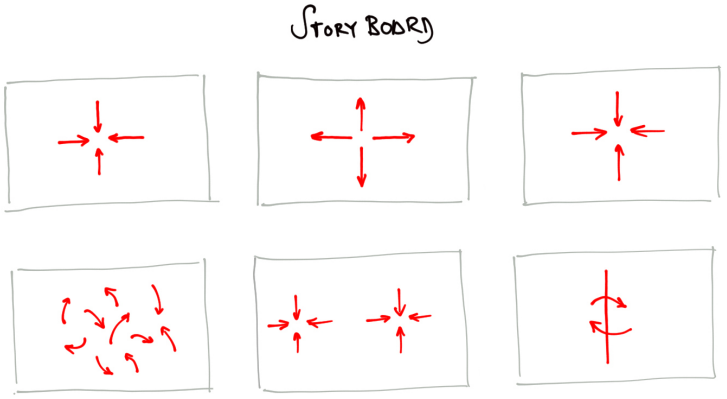


Fig. 5.8: Simplified version of the first storyboard.

This information was then given to the musician, who analyzed the storyboard and the used procedural elements. He suggested different musical instruments and effects that would go well which each of these elements, chose a time signature for the composition, and synthesized the first version of the music.

Having the time signature, we were able to start the MoCap sessions. To simplify the experiment, we chose to use a single step for the whole choreography. The step was suggested by the dancer, who proposed a motion that seemed fluid and expressive. The idea was to explore the fact that procedural animation methods would go well with the contemporary dance style, which suggest movements based on the natural, organic motion of the human body.

During the MoCap sessions, the music's time signature was altered several times until the dancer was comfortable with performing the step she had chosen for the exact duration of a single measure.

To implement the effects of the different procedural elements on the group of autonomous characters we used the OpenSteer [27] library which provides a toolkit of steering behaviors and a plug-in framework, which allows a visualization of the resulting motions. While simulating the effects of each element to the group of autonomous characters, the choreographer came up with several other ideas that would make the group dance more interesting and therefore iterative editing was made not only to insert new procedural elements on the storyboard but also to change the duration of these elements.

After the final group motion was designed, the musician was asked to recompose the music, so that it would match the new specifications. Finally, we synthesized the motion using the mechanisms for combining segments and inserting variations described above.

5.4 Experiment 4: Combining Dance Steps

In this experiment, we explored combining different dance steps during the choreography. For simplification, we chose not to work with group motion specifications which were already illustrated in the previous examples. Instead, we synthesized the dance for only a single character.

We chose to work with Ballet movements and were assisted once again by Raquel Leão, who chose a classic music for the composition and selected four Ballet steps: *Couru*, *Échappé*, *Spring Point* and a *Sissone* combination. We observed that each of these steps have very different structures. For example, each of these steps has a different duration. The *Couru* and the *Spring Point* are smaller steps that can be performed in the duration of a single musical measure, while the *Échappé* lasts for two measures. The final step suggested by the dancer is a combination for ballet movements in which the fundamental step is called *Sissone*. The performed *Sissone* combination has the duration equivalent to four musical measures.

Another very important difference is that the *Couru* and the *Spring Point* allow different movements along the stage and the other two steps can only be performed while the dancer rests in a single position. This is very important to know when capturing the dance, since we can ask the dancer to perform steps that allow locomotion several times and in different directions. In view of the previous examples, we suggested the nine directions shown in Figure 5.7. However, we learned that certain steps like the *Couru* can only be performed while moving to the left if the leading foot is the right one, and vice-versa.

Finally, the *Échappé* and the *Spring Point* are weight transfer motions, while the other two steps are not. Notice that, to sequence several recurrences of steps with weight transfer, it is necessary to alter between instances that start with the right and ones that start with the left foot. On the other hand, if weight transfer does not occur, the same instance can be repeated sequentially since the step always finishes with the weight on the same foot it begins with. Figure 5.9 illustrates how this attribute defines the connections in the motion graph.

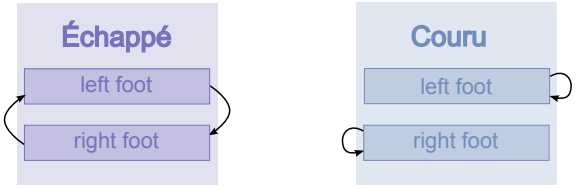


Fig. 5.9: Motion graphs for steps with (*Échappé*) and without (*Couru*) weight transfer.

We continued to analyze the structure of these steps by observing the motion at the boundary of each segment. We noticed that the most common beginning/finishing poses for ballet steps are the *Sous-sur* (rising up) and the *Plié* (bending of the knees) and these were in fact the only two beginning/finishing poses that were present in the captured movements. Since we can only have graph connections when the motion segment boundaries are similar,

and since the *Sous-sur* and the *Plié* are very distinct, we conclude that we can only connect two segments if the finishing pose of the first one is the same as the beginning pose of the second one.

Hence, to increase the connectivity between the different steps we asked the dancer to perform each motion four times varying the beginning and finishing poses. Of course, not every step can be that easily altered (e.g., the *Couru* can be performed starting and finishing with either pose; however, the *Échappé* must end on a *Plié*). Therefore, although we were able to create a very connected motion graph, the possible combinations of steps was still limited.

Lastly, the choreographer suggested a dance for the chosen musical piece that only involved these four ballet steps and we synthesize an animation of the resulting motion. To combine the dance steps we run the A* algorithm described in Section 3.4.1. Figure 5.10 illustrates the sequence of steps determined by the choreographer and Figure 5.11 shows the state machine that represents the connections between the step clusters.

Fig. 5.10: Sequence of dance steps in the timeline.

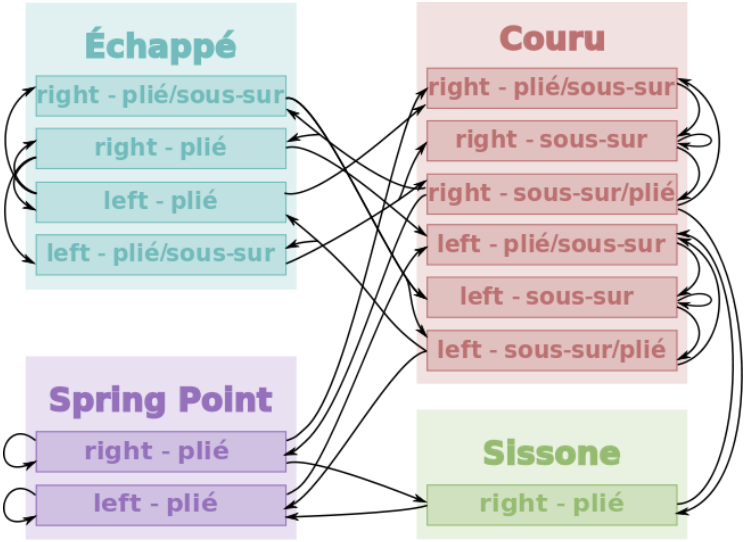


Fig. 5.11: Connections between the step clusters.

We observed that, since we made every effort to capture all the different variation of each motion, allowing the strongest possible graph connectivity, we were able to synthesize any feasible combination of steps (i.e., any combination that can be performed by a human dancer).

Chapter 6

Conclusions

In this work, we suggested an authoring and collaboration platform for dance shows, discussed the relevance of this research to dance design, studied the technical aspects related to both dance and choreography specification and synthesis, developed tools that were sufficient to demonstrate the proposed concepts, and illustrated the applications of our method with experiments performed with dancers and musicians.

To conclude, we will discuss the achieved results and future directions. We will make this discussion for the three major aspects of our research: dance, group motion and the authoring platform.

6.1 Dance

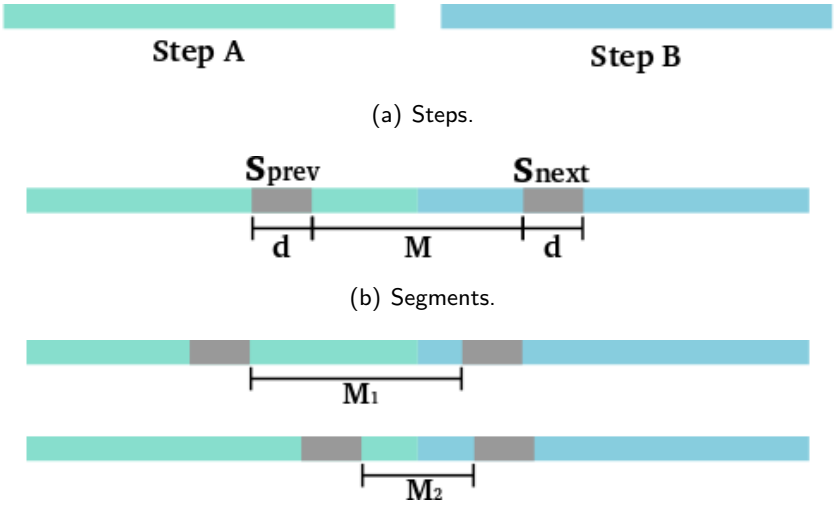
We suggested a structured measure-synchronous motion graph for dance synthesis. This technique proved to be very effective because it guarantees synchronism with music and, therefore, allowed synthesis of motions that maintain the rhythmic structure of the songs. This method, however, has the disadvantage of strongly limiting the connections in the graph.

A first consideration is that we only allow transitions to be made at exactly the last frame of each measure. This is not very effective because, since transitions are made with interpolations of two blocks of frames, it is possible that they would be smoother at different frames. Hence, a simple modification that allows creating connections in an interval around the last measure frame could highly improve the performance of our method while preserving the synchronism we desire.

Another consideration related to graph connectivity is the number of structural variations that are necessary to permit sequences of dance steps. Consider, for example, the ballet steps described in Section 5.4. A ballerina can easily perform a *Couru* followed by a *Sissone* and a *Couru* followed by an *Échappé* and, as far as she perceives it, the *Couru* motion does not changed at all. However, if we analyze the two motions, we see that the first *Couru* finishes in a *Sous-sur*, as the ballerina continues to perform a step on tip-toe, while the second one

ends with a *Plié*. Hence, we must ask the dancer to perform the *Couru* step twice and store both motions as structural variations. This is undesirable, not only because it is extra work for the dancers, but also because it is quite unnatural[1]. Synthesizing these variations without capturing the motion twice is, of course, not a trivial problem and solutions can range from data driven methods to physically based techniques.

An idea for solving this problem by exploring MoCap data is shown in Figure 6.1. Let steps A and B, shown in Figure 6.1(a), be two segments we wish to connect, let d be the size of the window of frames used to create transitions (see Section 3.1.3), and let M be such that $M \leq (N/2 - d)$, where N is the number of frames per measure. The idea is to search the database for segments of size $M+2d$, such that the first and the last d frames are, respectively, similar to segments s_{prev} and s_{next}, illustrated in Figure 6.1(b). Notice that the only restriction is that s_{prev} belongs to step A and thats$_{next}$ belongs to step B, see Figure 6.1(c). Therefore, a search should be made for many different positions of s_{prev} and sizes of M. We can see that this is a very computationally expensive procedure, since we must test the entire database several times. However, this fact may not come as a very strong disadvantage because, in typical scenarios, these calculations can be executed a priori (i.e., as soon as the dancers add new motions to the database and before the choreographer actually starts to design the dance).



(a) Steps.

(b) Segments.

(c) Different possibilities.

Fig. 6.1: Illustration of method for creating new transitions.

In addition to allowing synthesis of motions that preserve the music's rhythmic structure, segmenting the MoCap data into measures allowed us to explore several editing tools which were fully described in Section 3.3. The results presented in Chapter 5, show that these techniques were indeed effective for manipulating motion data.

[1]We actually had a very interesting experience with Raquel Leão while we were capturing these two movements, because she insisted that she was doing the same step and we had to show her that she was performing, in fact, two different motions.

An aspect that we believe deserves further investigation is motion variation. In this work, we elaborated signal processing tools for altering dance, which in many ways can be related to very early research in this area that involves adding noise [26]. More recent works use statistical analysis and machine learning to create spatial and temporal variations of motions data from a small collection of examples [19, 23]. A different, but also very popular approach, is learning and transferring styles between motions [15, 6]. Variation and style are different because in the first case the motions are quite similar, while, in the second, mood or behavior can be modified creating very distinct movements. We believe that both style and variations would be interesting to simulate in dance show scenarios, in which the movements are not only different because of the natural asynchronism, but also because some dancers may be shyer or more graceful than others.

6.2 Group Motions

In this work, we developed tools for specifying and synthesizing the motion of groups of characters as they move on stage creating formations and following trajectories. Our discussions on dance analysis allowed us to propose declarative specification methods, that proved to be effective in our experiments. In the future, it would be interesting to improve these mechanisms with additional feedback from dancers, musicians and choreographers.

We also argued that, since we are able to represent and specify groups of dancers on the computer, we can create novel computational tools for dance design, taking advantage of several related works in crowd planning and simulation. We illustrated this idea using behavioral animation methods to control group motions on stage. Recently, we have learned that experiments of flocking behavior for dance have also been carried out lately by other researchers yielding interesting results [22].

There are, of course, several other computational techniques that can be easily applied to our framework, but which we were unable to explore due to time limitations. In terms of specifying formations, we proposed a sketch based method that allows choreographers to determine shape, density and patterns. Extensions of this approach would include studying different algorithms for distributing dancers inside shapes [12].

For motion based on boundary conditions, we believe that although our matching and collision control methods were quite effective, it would also be valid to investigate different mechanisms for transforming between two formations and compare them with our method. Related work on this area includes methods based on spectral analysis of clusters of individuals, which try to preserve adjacency relationships among the characters during transitions [33].

Finally, with regard to synthesizing new motions from initial conditions and evolution rules, previous research is extremely rich. In addition to other methods based on autonomous agents, recent approaches for crowd simulation propose modeling global flow [35] as well as extracting

information from videos of real human crowds [20]. In addition to exploring procedural methods for design, such mechanisms can also be used for editing group motions [18].

A final aspect of our platform that was very successful is the hierarchical structure used for group segmentation. Further work on this area would involve analyzing dependencies between dancers of different subgroups. Such mechanisms would allow, for example, formations to be restructured and paths to be reshaped in order to avoid collisions.

6.3 The Authoring Platform

The main objective of this work is to propose a novel platform for dance design that integrates the creative elements that compose dance shows and stimulates collaborations between dancers, musicians and choreographers. To this end, it was necessary not only to explore computer animation tools for individual and group motion synthesis, but also to study the artists' process of creation in order to suggest a natural authoring environment.

Understanding dance and dance design was quite challenging, first because it is a field quite distant from ours, and second because the resources are somewhat limited. Dance has no standard formalization or notation and there is no technical reference that formally describes the process of dance content creation. In beginning this work, we made a strong effort to investigate the artistic elements we would reference, but most of the material we found came from web sites, observations made on existing shows, one-on-one conversations with artists, or our personal experience in the field.

More towards the end of this work, we had a close interaction with the dancer and choreographer Raquel Leão, who spent a month and a half with us discussing the environment, sharing ideas, and participating in experiments. We believe that this was a very rich experience, which truly allowed us to understand what artists are looking for in an authoring platform, and also to validate our analysis of dance elements. Hence, future directions of this work should definitely involve stronger interactions with different artists, as their feedback may allow us to refine and extend these ideas in many different ways.

In order to demonstrate the applications of the proposed platform, we implemented the essential functionalities which allowed us to validate the studied concepts and produce experiments with artists. It is important to mention that it was not our intention to fully develop a robust and integrated software. Hence, although we tried to polish the interface as much as possible, the different functionalities were not integrated (e.g., we had different programs for graph search, visualization, and behavioral animation techniques that used OpenSteer). In addition, some algorithms and several pre-processing routines were implemented in Matlab or manually.

We believe that this implementation was sufficient, since it has allowed us to demonstrate the proposed methods with examples. However, if there were no time limitations, it would

be interesting to develop a robust software which combines the various tools and runs in real time. It would also be interesting to design interfaces for music and individual dance input.

Finally, the proposed ideas should be taken to the next level, being applied to live scenarios in which conception and execution are done simultaneously, in a framework of instant feedback that fully integrates the collaborations of dancers, choreographers and musicians. As discussed in Chapter 2, we believe that this environment is much more than a tool to facilitate creation, but in fact, suggests new forms of artistic expressions, bridging the gap between performers and spectators. Expansions of the developed tools, additional experimentations, and contributions from artists can made these ideas evolve in directions that we can only start to imagine.

In short, we have suggested a novel platform that, although is very much applicable to traditional dance compositions, puts forward a whole new model for design that results from combining art and technology. We believe that, as advances in graphics technologies become increasingly robust and accessible, they will increasingly permeate different aspects of life and society and will be exceptionally influential in art, permitting not only qualitative changes, but also structural transformations of established paradigms.

We hope our work contributes to this revolution.

Appendix A

Motion Capture

Motion Capture (MoCap) is a technology that allows us to record human motion with sensors and to digitally map the motion to computer-generated creatures [3].

The applications of motion capture go far beyond animation and include biomedical analysis, surveillance, sports performance analysis and input mechanism for human-computer interaction. Each application has its own particular set of singularities and challenges.

In this Appendix, we will discuss motion capture of full body motion for character animation. We will introduce the basic concepts, detail the technique and describe how it was implemented at the VISGRAF Laboratory.

A.1 The Technique

Motion Capture systems can be divided into three different categories [1]: inside-in (sensors and sources located on the body), inside-out (sensors located on the body and sources outside) and outside-in (sources located on the body and sensors outside).

An example of an inside-in system is an electromechanical suit, where sensors are attached to to the performer's body measuring the actual joints inside the body. The advantages of this method are the absence of occlusions (all sensors are always "visible") and the portability of the suits. Nevertheless the actor's movement are constrained by the armature.

In electromagnetic (inside-out) systems, electromagnetic sensors, placed on joints of the moving object, measure their orientation and position with respect to an electromagnetic field generated by a transmitter. This method also directly collects positions and orientations of the sensors and does not have to consider occlusion problems. The drawbacks of this technique relate to the range and accuracy of the magnetic field as well as the constraint of movement by cables.

Optical systems are inside-out systems which use data captured from image sensors to triangulate the 3D position of a subject between one or more calibrated cameras. Data acquisition is traditionally implemented using special retro-reflexive markers attached to an actor

and infrared cameras. However, more recent systems are able to generate accurate data by tracking surface features identified dynamically for each particular subject. This is called the marker-less approach.

Optical techniques are widely used since they allow large performance areas (depending on the number of cameras) and performers are not seriously constrained by markers. Nevertheless, orientation information is not directly generated and therefore extensive post-processing needs to be done in order to reconstruct the three dimensional motion.

The three major tasks that need to be undertaken are [24]: markers have to be identified and correlated in the 2D images, 3D locations have to be constructed from the 2D locations, and 3D marker locations need to be constrained to the model being capture (e.g., human skeleton).

The first step is to find the markers at each frame and track them over the video sequence. The latter can be quite difficult because markers often overlap, change position relative to one another and are occluded. In addition, noise can arise from the physical system (markers may move relative to their initial positions) or the sampling process. Hence, three major problems occur when tracking the markers and may need user interventions to be resolved: missing data in the presence of occlusions, swapped paths between two markers that pass within a small distance of each other, and noise.

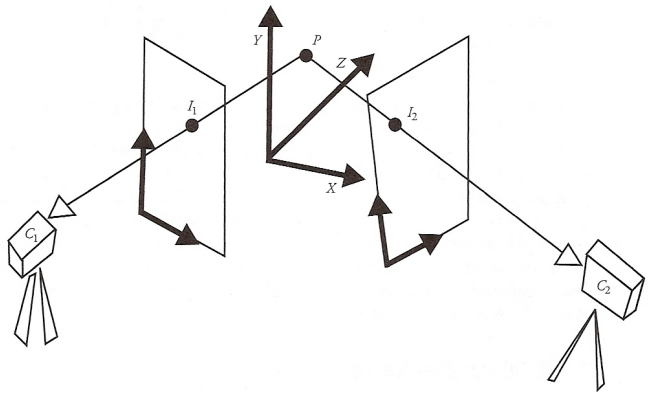


Fig. A.1: Two-camera view of a point. Extracted from [24].

The second step is to reconstruct the 3D points from the 2D trajectories. For this purpose, cameras need to calibrated , i.e., the position, orientation and intrinsic properties of the cameras need to be known. This can be done by recording a number of image points whose locations are known. With calibrated cameras, the three dimensional coordinates of each marker can be reconstructed from at least two different views (the more orthogonal the views the better), as shown in Figure A.1.

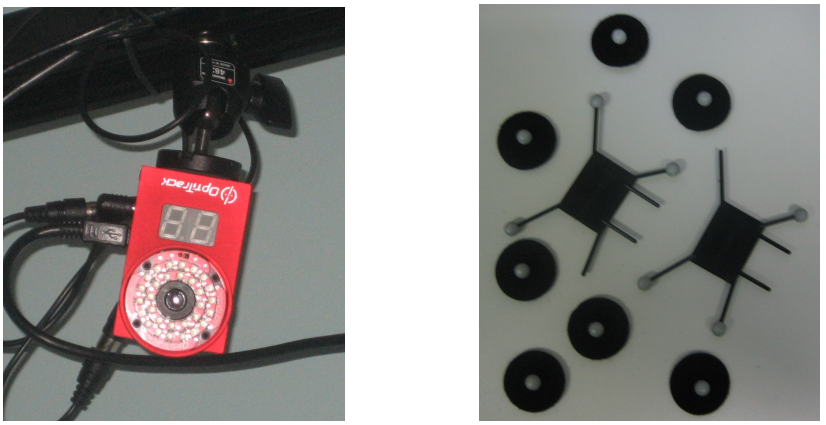
Finally, the 3D marker positions need to be transformed into the motion parameters of a kinematic skeleton model. For this, it is crucial to place the markers in adequate positions. The markers cannot be located exactly on the joint firstly because they are placed on the surface

of the skin and second because they have to be set in positions where they will not move according to the performance. This represents a problem because although distances between consecutive joints are always the same, distances between markers may vary. Therefore, in order to locate the joint relative to the marker, we need not only the position, but also the orientation.

To solve these problems, most standard formats require placing three non-linear markers on each rigid body part instead of one in each joint [9]. These locations are then used to determine the position and orientation of each limb and the skeleton configuration is determined while the tracked subject performs an initialization pose (T-pose). Since noise is usually added to the data, directly using the calculated joint positions will probably result in varying bone lengths. To get around this problem, many systems calculate joint rotations and use a skeletal hierarchy to reconstruct the pose of the articulated body.

A.2 MoCap at Visgraf

There are currently several comercial MoCap System available. During the course of this project, motion capture was done in the VISGRAF Laboratory using OPTITRACK, `NaturalPoint`'s optical MoCap System. The infrared cameras (see Figure A.2.b) are sensitive to the retroreflexive markers (see Figure A.2.b) which are placed in the performer's body.



(a) OPTITRACK camera (b) Retroreflexive markers

Fig. A.2: MoCap setup.

As with traditional animation and many other arts, MoCap is actually composed of a number of phases[8]:

- studio set-up,
- calibration of capture area,
- performance and capture of movement,

- clean-up of data, and
- post-processing of data.

The studio was set up with ten cameras, as shown in Figure A.3. Tracking a large number of performers or expanding the capture area is accomplished by the addition of more cameras. Since each marker must be“seen" by at least two cameras, a greater number of cameras diminishes the possibility of occlusions.

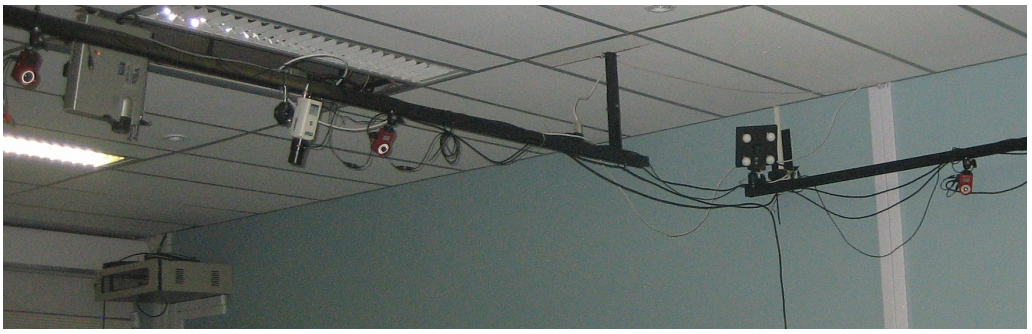


Fig. A.3: OPTITRACK cameras set up at the Visgraf Laboratory.

Camera calibration was done with the help of OPTITRACK's software, ARENA. This software also specifies the position of the markers in the performer's body (see Figure A.4).

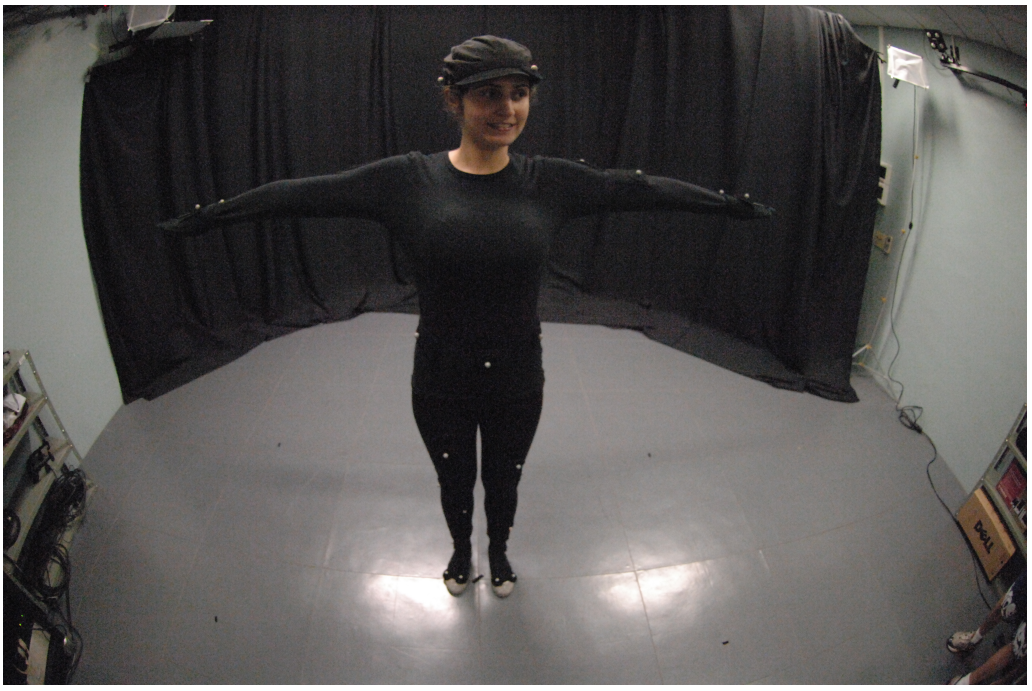


Fig. A.4: Dancer performing at the VISGRAF Laboratory

The ARENA software (see Figure A.5) was also used to process the acquired data. The first processing step is to trajectorize the data. This procedure takes the 2D data from each

individual camera and changes it to a fully 3D path of each individual marker. After this procedure the software allows for post capture editing (such as filling gaps, fixing swaps and smoothing tracks) and exporting a file in a BVH or C3D format.

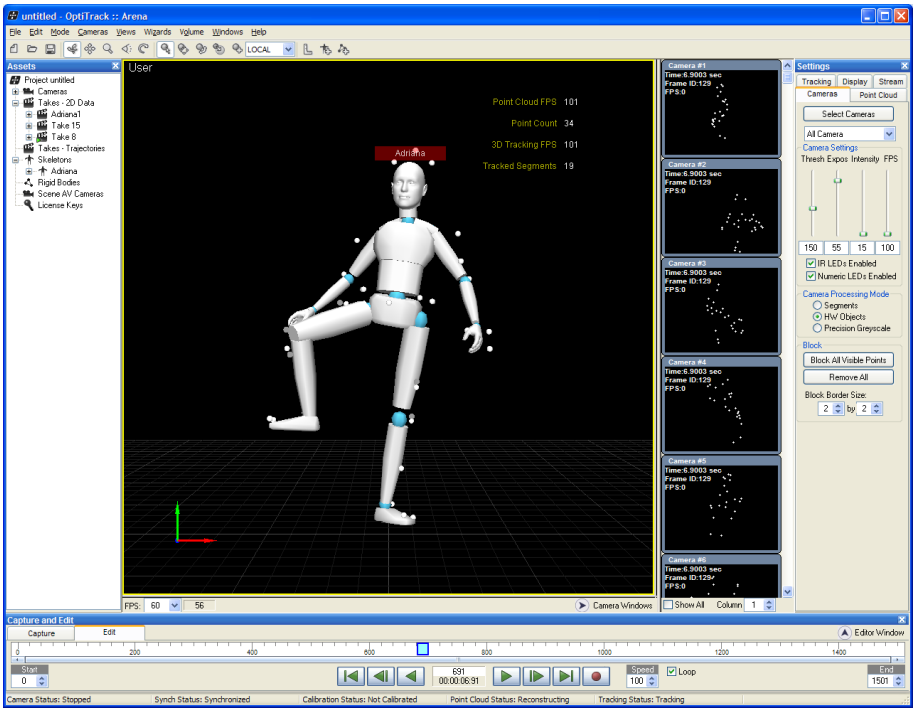


Fig. A.5: The ARENA software.

Appendix B

BVH File Format

In this Appendix, we will describe the BVH file format by means of a simple example. Consider the following BVH file:

```
HIERARCHY
ROOT A
{
    OFFSET 0.000000 0.000000 0.000000
    CHANNELS 6 Xposition Yposition Zposition Zrotation Xrotation Yrotation
    JOINT B
    {
        OFFSET 10.000000 0.000000 0.000000
        CHANNELS 3 Zrotation Xrotation Yrotation
        End Site
        {
            OFFSET 0.000000 -20.000000 0.000000
        }
    }
}
MOTION
Frames:    3
Frame Time:    0.040000
30.0    80.0    20.0    0.0     0.0    0.0     0.0    0.0    0.0
30.0    80.0    20.0    60.0    0.0    0.0     0.0    0.0    0.0
30.0    80.0    20.0    60.0    0.0    0.0   -30.0    0.0    0.0
```

The first part of the file describes the skeleton, which has a root node A, a joint B and and ending node C, as illustrated in Figure B.1. Notice that the the distance between the nodes

are given by the offsets described in the file. In this example $O_A = [0\ 0\ 0]$, $O_B = [10\ 0\ 0]$, and $O_C = [0\ -20\ 0]$.

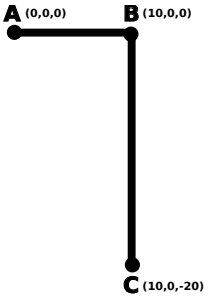


Fig. B.1: Articulated body.

This part of the file also details the channels associated to each node and therefore indicates the size of the motion data and what is the information of each entry of the motion vector. In this case, node A has 6 channels and node B, 3 channels. This means that the motion data (second part of the file) will be a series of vectors of size 9, each corresponding to a single frame.

Consider, for example, the third frame (third line of the file after `MOTION`). The first three components are the translation of node A ($T_A = [30\ 80\ 20]$), the second three are the rotations associated to this node, R_A, and the last three are the rotations associated to node B, R_B. Notice that the order of rotations are given in the first part of the file. In this example we first rotate 60^o around the z axis, than 0^o around the x axis, and, finally, the 0^o around the y axis. Therefore, $R = R_y R_x R_z$.

Finally, we calculate the position of each node using the following equations:

$$\begin{cases} A = O_A + T_A \\ B = A + O_b R_A \\ C = B + O_c R_B R_A \end{cases}$$

Figure B.2 illustrates the arm position in frames 2 and 3.

(a) Frame 2 (b) Frame 3

Fig. B.2: Articulated body motion.

Bibliography

[1] Charlotte Belland, James W. Davis, Michael Gleicher, and Barbara Helfer. *Motion Capture: Pipeline, Applications, and Use*. SIGGRAPH'02 Course Notes 28, SIGGRAPH-ACM publication, San Antonio, Texas, USA, July 2002.

[2] Matthew Brand and Aaron Hertzmann. Style machines. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 183–192, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[3] Chris Bregler. Motion capture technology for entertainment [in the spotlight]. *Signal Processing Magazine, IEEE*, 24(6):160 –158, November 2007.

[4] Armin Bruderlin and Lance Williams. Motion signal processing. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 97–104, New York, NY, USA, 1995. ACM.

[5] ChoreoPro. Dance designer. `http://www.choreopro.com`.

[6] Marco da Silva, Yeuhi Abe, and Jovan Popović. Interactive simulation of stylized human locomotion. In *ACM SIGGRAPH 2008 papers*, SIGGRAPH '08, pages 82:1–82:10, New York, NY, USA, 2008. ACM.

[7] Reinhard Diestel. *Graph Theory*. Fourth electronic edition, 2010.

[8] Maureen Furniss. *Motion Capture*. MIT Communications Forum, http://web.mit.edu/comm-forum/papers/furniss.html.

[9] Margaret S. Geroch. Motion capture for the rest of us. *J. Comput. Small Coll.*, 19(3):157–164, 2004.

[10] Michael Gleicher. Animation from observation: Motion capture and motion editing. *SIGGRAPH Comput. Graph.*, 33(4):51–54, 2000.

[11] Jonas Gomes and Luiz Velho. *Fundamentos da Computação Gráfica*. IMPA, 2003.

[12] Qin Gu and Zhigang Deng. Formation sketching: An approach to stylize groups in crowd simulation. *Proceedings of Graphics Interface (GI)*, May 2011.

[13] Jessica Hodgins. Animating human motion. *Scientific American*, 278(3), March 1998.

[14] Jessica Hodgins and Zoran Popovic. *Animating Humans by Combining Simulation and Motion Capture*. SIGGRAPH'00 Course Notes 33, SIGGRAPH-ACM publication, New Orleans, Louisiana, USA, July 2000.

[15] Eugene Hsu, Kari Pulli, and Jovan Popović. Style translation for human motion. *ACM Trans. Graph.*, 24:1082–1089, July 2005.

[16] Lucas Kovar and Michael Gleicher. Automated extraction and parameterization of motions in large data sets. *ACM Trans. Graph.*, 23(3):559–568, 2004.

[17] Lucas Kovar, Michael Gleicher, and Frédéric Pighin. Motion graphs. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 473–482, New York, NY, USA, 2002. ACM.

[18] Taesoo Kwon, Kang Hoon Lee, Jehee Lee, and Shigeo Takahashi. Group motion editing. In *ACM SIGGRAPH 2008 papers*, SIGGRAPH '08, pages 80:1–80:8, New York, NY, USA, 2008. ACM.

[19] Manfred Lau, Ziv Bar-Joseph, and James Kuffner. Modeling spatial and temporal variation in motion data. In *ACM SIGGRAPH Asia 2009 papers*, SIGGRAPH Asia '09, pages 171:1–171:10, New York, NY, USA, 2009. ACM.

[20] Kang Hoon Lee, Myung Geol Choi, Qyoun Hong, and Jehee Lee. Group behavior from video: a data-driven approach to crowd simulation. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA '07, pages 109–118, Aire-la-Ville, Switzerland, Switzerland, 2007. Eurographics Association.

[21] Yan Li, Tianshu Wang, and Heung-Yeung Shum. Motion texture: a two-level statistical model for character motion synthesis. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 465–472, New York, NY, USA, 2002. ACM.

[22] Susan Marshall and Naomi Leonard. Flocklogic. `http://www.princeton.edu/~flocklogic/`.

[23] Jianyuan Min, Yen-Lin Chen, and Jinxiang Chai. Interactive generation of human animation with deformable motion models. *ACM Trans. Graph.*, 29:9:1–9:12, December 2009.

[24] Rick Parent. *Computer animation: algorithms and techniques*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2002.

[25] Ken Perlin. Path planning. http://mrl.nyu.edu/~perlin/experiments/path/.

[26] Ken Perlin. Real time responsive animation with personality. *IEEE Transactions on Visualization and Computer Graphics*, 1:5–15, March 1995.

[27] Craig Reynolds. Opensteer. http://opensteer.sourceforge.net/.

[28] Craig Reynolds. Steering behaviors for autonomous characters, 1999.

[29] Craig W. Reynolds. Flocks, herds, and schools: A distributed behavioral model. In *Computer Graphics*, pages 25–34, 1987.

[30] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, second edition, 2003.

[31] Adriana Schulz, Marcelo Cicconet, and Luiz Velho. Motion scoring. In *ACM SIGGRAPH 2010 Posters*, SIGGRAPH '10, pages 13:1–13:1, New York, NY, USA, 2010. ACM.

[32] Adriana Schulz and Velho Luiz. Rotations and interpolations. Technical Report. http://www.impa.br/~aschulz/anim/rotations.pdf, 2010.

[33] Shigeo Takahashi, Kenichi Yoshida, Taesoo Kwon, Kang Hoon Lee, Jehee Lee, and Sung Yong Shin. Spectral-based group formation control. *Comput. Graph. Forum*, 28(2):639–648, 2009.

[34] Lorenzo Torresani, Peggy Hackney, and Chris Bregler. Learning motion style synthesis from perceptual observations. In *Proc. of Neural Information Processing Systems (NIPS)*, 2007.

[35] Adrien Treuille, Seth Cooper, and Zoran Popović. Continuum crowds. *ACM Trans. Graph.*, 25:1160–1168, July 2006.

[36] Jue Wang, Steven M. Drucker, Maneesh Agrawala, and Michael F. Cohen. The cartoon animation filter. *ACM Trans. Graph.*, 25:1169–1173, July 2006.

[37] Andrew Witkin and Zoran Popovic. Motion warping. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 105–108, New York, NY, USA, 1995. ACM.