Chapter 1

Introduction

Many computer applications prominently feature virtual human characters. For example, in the entertainment industry, virtual humans appear in movies as digital body doubles to perform superhuman feats, and video games often depict a virtual human figure interacting with an environment. Virtual human characters also appear frequently in training simulations to supplement on-the-job training or in-class learning in a wide range of industries: the military uses training simulations to help train soldiers in combat, athletes use simulations to improve their game, construction workers can practice using fork lifts and bulldozers in a simulation, and there is talk now of using training simulations to better teach medical students bed-side manner.

It is often important that the virtual human figures in these applications appear lifelike to help create a sense of immersion. While there are a number of different characteristics that contribute to a character's believability, one important aspect is the way a character moves. Unfortunately, human motion is a complicated process driven by a person's subconscious goals. Bones, muscles, and other tissues work together to make a human's body perform a specific task. Creating animations for a computer character that mimic these complicated processes is a challenge. This challenge is further complicated by the fact that people are good at recognizing when a motion looks fake, even if they cannot pinpoint the exact reason why. This dissertation is concerned with the automated

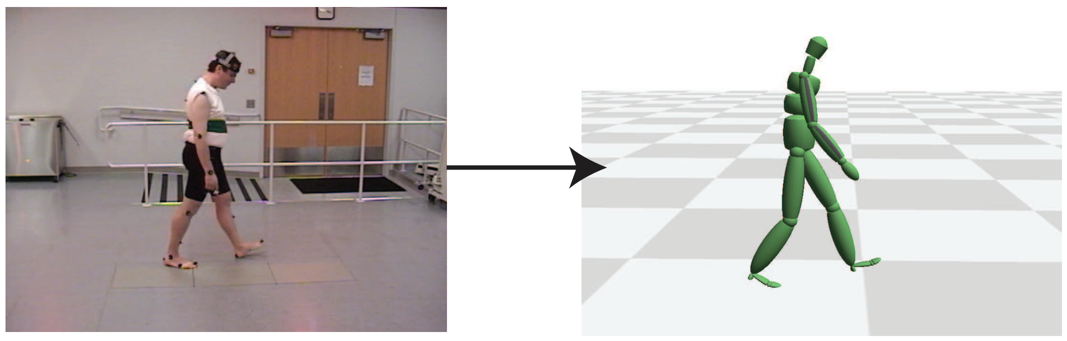


Figure 1.1 Motion capture is the process by which observed human motion is translated into a digital character motion.

synthesis of quality human motion for *interactive applications*, or applications like training simulations and video games where a virtual human character is directed through an environment by a user.[1]

Traditionally, talented and patient artists tackled the challenge of authoring believable animations for human characters for movies and video games. More recently, these artists have been aided by the introduction of *motion capture*. Motion capture is a set of techniques that use sensors to observe and record the detailed motion of an object or set of objects (see Figure 1.1). While the quality of the recorded motion and the process of capturing that motion vary from one motion capture technology to another, motion capture is capable of producing accurate representations of the way a person moves [Men00].

Motion capture is a potentially useful technique for creating believable human motion as it produces animations with all of the subtle nuances of real motion. The problem with motion capture data is that it is only directly useful for reproducing a specific motion. For example, the motion of a person reaching toward the front of a shelf is different in small but perceivable ways from the motion of that same person reaching toward the back of that same shelf; modifying a motion by hand in order to capture all of these subtle changes can be difficult.

[1]For the purposes of this dissertation, I am concerned with synthesizing motion for the character that is under the direct control of the user. However, the techniques discussed in this document can also be applied to non-player characters (NPCs) found in an interactive application.

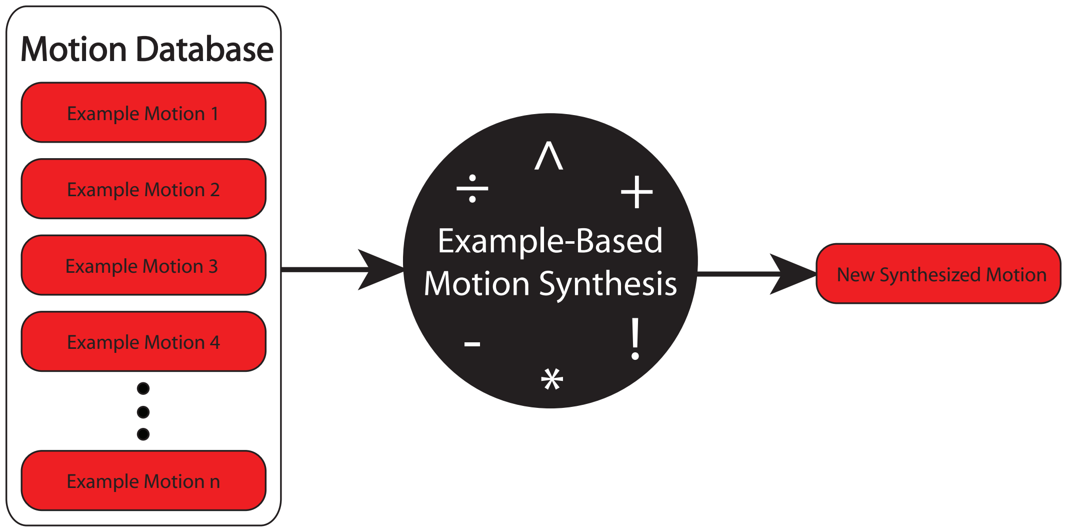


Figure 1.2 Example-based motion synthesis uses a database of example motions to synthesize new ones.

Recent research has focused on ways to improve the utility of motion capture data. The goal of much of this research is to use a database of motion-captured examples to generate new motions that appear as if they were created in the same way as the originals. These techniques are called *example-based motion synthesis methods* (see Figure 1.2). Research in example-based motion synthesis has greatly increased the usefulness of motion capture data by providing new ways to create believable human motion.

While current research has provided new techniques for reusing motion capture data, creating believable human motion is still a problem in many domains. In particular, interactive applications place additional constraints on the way human motion is generated. At runtime, interactive applications must be able to quickly generate motions that not only look realistic but also accurately meet user requests. At the same time, in order to encourage experimentation and allow quality motion generation for interactive applications with a small budget, the process of authoring human character motion should not be overly time consuming.

Unfortunately, current approaches to authoring and synthesizing human motion fall short of meeting all of these constraints at once. Video game companies spend hundreds of man hours carefully hand-tailoring motions that allow efficient motion generation with low latency but sacrifice visual realism. Alternatively, example-based motion generation techniques provide automated authoring of visually realistic motions, but these methods require too much synthesis time and respond too slowly for use in interactive applications. Automated authoring methods that could provide all of the characteristics described above at the same time would not only improve the quality of human motion in high-budget video games and simulations but would also allow low-budget interactive applications to benefit from the ability to generate realistic human motion. *My goal is to increase the utility of example-based motion generation methods for interactive applications by supplying insight into techniques that provide efficiency, low latency, high accuracy, quality, and automated authoring.* For a more detailed explanation of each of these characteristics, see Section 1.1.

There are two distinct but related motion synthesis subproblems. The first is the problem of generating individual motions, or *motion clips*, that meet specified constraints. For instance, an application might need to synthesize a motion clip where a character walks at a specified curvature and looks in a specified direction. The second problem is how to produce long, continuous *motion streams*. For example, an application might need to synthesize a motion where a character walks at a specified curvature, *then* picks up an item from a shelf, *then* skips away, and so on. Some motion synthesis techniques aim to solve both problems while others focus on only one. This dissertation addresses both. ***It is my thesis that example-based techniques that decouple motion parameters to synthesize motion clips and that use highly-structured control mechanisms for transitioning between these individual clips can provide automated authoring of high-quality, controllable human motion streams for interactive applications.***

This thesis is based on two observations, each of which corresponds to one of the two motion synthesis subproblems. First, any sufficiently complex interactive application needs to generate segments of motion with many different constraints or *parameters*. For example, one application may need to generate motions of a human walking and running at different speeds and curvatures while performing some action with their upper body and gazing at different locations in an environment. It is important that we be able to decouple these parameters during motion synthesis, allowing motions that meet a subset of the parameters to be generated independently and then combined into one composite motion. Unfortunately, most example-based motion synthesis techniques generate an entire frame, or all the degrees of freedom (DOFs), of motion at once. Without decoupling, we face a combinatorial explosion in the number of motions that need to be captured. We would need an example of a character performing every possible combination of controllable parameters. Decoupling allows motions for specific parameters to be captured independently. Similarly, by decoupling, it is possible to produce motions with an unexpected combination of parameters. For example, I may not expect that I will need to be able to control both the location that a character reaches to as well as the direction a character looks simultaneously. Decoupling would allow me to produce motions that control both characteristics without needing to return to a motion capture studio to capture additional data that exhibits variations in both facing direction and reach location at the same time. And, finally, decoupling lends itself well to parallel processing. *Methods that* *allow the decoupling of motion parameters during synthesis can improve the accuracy, synthesis time, and storage requirements of generated motions*.

The second observation is that human characters in interactive applications often need to be able to perform many different actions. For example, a character might be able to run, walk, jump, punch, climb ladders, duck, swim, sit down, etc. A number of example motions would be needed for each of these actions in order to synthesize accurate human motion using an example-based approach. It is important that an interactive application be able to find specific motions within this large number of examples in order to accurately synthesize new human motion quickly at runtime and to produce good transitions between the generated motion clips. Structured character control mechanisms facilitate fast decision-making by simplifying the problem through organization. And because structure can be used to organize large numbers of motion clips, it allows the control of many different types of motion (e.g., running, walking, punching, and jumping), as well as a lot of variation within a single motion type (e.g., a character might be able to punch to many more locations in space). *Highly-structured motion representation models allow for fast and accurate motion control without extensively sacrificing quality*.

In order to support my thesis, I present algorithms that decouple some of the most common and important motion parameters and introduce a highly-structured motion transition representation that allows easy interactive control because of its structure. In particular, I make the following specific technical contributions:

1. **Runtime Method for Splicing Upper-Body Actions with Locomotion (Chapter 4).** Interactive applications often divide the problem of character planning into two pieces: what action the character is performing and how/where the character is moving. I provide a method that allows motion synthesis to also *decouple* character action from locomotion, allowing *automated authoring* of *quality* motions that can be *synthesized quickly* and *with a significantly decreased number of required example motions for accurate character control*.

2. **Runtime Method for Adjusting the Gaze Direction of a Character (Chapter 5).** While many have recognized the importance of head facing direction, or *gaze*, for understanding a

person's motivations and for exhibiting realism in virtual characters, few have studied ways to edit gaze in a realistic way. I provide a method for *altering gaze at runtime independent of other motion parameters* by using examples from a motion capture database to build a low-dimensional representation of gaze that is based on scientific findings related to the way humans move. The method allows *quality* motions to be *authored in an automated way without extensive storage requirements*.

3. **Method for Automated Authoring of Parametric Motion Graphs (Chapter 6).** I provide a simple, *automated* method for building parametric motion graphs, or *highly-structured*, graph-based control mechanisms that combine parametric synthesis to gain *accuracy and flexibility* and synthesis by concatenation to produce *quality transitions* between motions.

4. **Runtime Method for Using Parametric Motion Graphs (Chapter 6).** I show how to use parametric motion graphs at runtime to produce *accurate, high-quality, responsive* motion streams at *interactive speeds*. Because parametric motion graphs are *highly-structured*, *little authoring effort* is needed to use a graph for interactive character control.

With these contributions I show the utility of decoupling parameters during motion clip generation by providing distinct decoupling solutions to two different but important motion generation problems. I also illustrate the usefulness of highly structured control mechanisms for synthesizing controllable motion streams from motion clips by showing how parametric motion graphs provide all of the required properties of interactive applications because of their structure.

The remainder of this chapter provides a more detailed description of the difficulties involved with synthesizing human motion for interactive applications, as well as an overview of the techniques presented in the rest of the document.

1.1 Problem Overview

Because of the complexities of the human body and our sensitivity to human motion quality, synthesizing quality human motion under any set of circumstances is difficult. But the problem becomes even more difficult when placed within the confines of an interactive application. In particular, interactive applications need methods that have each of the following characteristics:

1. **Efficient Synthesis** Any method for generating human motion at runtime in an interactive application must be able to synthesize each frame of motion at consistent, interactive speeds. A user should never see a dip in frame rate because it took longer than normal to generate a frame of motion.

2. **Efficient Data Storage** Because interactive applications must meet data storage limitations, it is important that representations used for human motion generation use little storage space.

3. **Low Latency or Response Time** At runtime, interactive applications must provide quick feedback to the user; the time between when a user requests an action and when the system actually renders the motion associated with that action should be predictable. A user should always know whether a character has “heard” his or her request.

4. **Accurate Motion Generation** Since a user issues commands in interactive applications, it is important that the motions synthesized at runtime in response to those commands closely meet the requests. For instance, if a user commands a character to punch at a target, the motion displayed by the system should show the character punching at the target as closely as possible.

5. **Visual Quality** Ideally, any motion generated for an interactive application should appear as if it were generated in the same way as the original example motions. While the quality of a motion is partly a subjective characteristic, some aspects of quality can be evaluated. For instance, it is important that generated motions be continuous. Motions cannot look as if they are segmented in time or across body parts. Distracting artifacts, such as knee-popping and jitters, should be avoided to add to the appearance of realism.

6. **Automated Authoring** To help promote easy experimentation and save important man-hours, any method for authoring human motion for an interactive application should not

	Example-Based Research	**In Practice**
Efficient Synthesis	No	Yes
Efficient Storage	No	Yes
Low Latency	No	Yes
Motion Accuracy	Yes	No
Visual Quality	Yes	No
Automated Authoring	Yes	No

Table 1.1 Table comparing existing motion synthesis methods against the required characteristics of interactive applications.

require extensive setup or hand-tweaking. Ideally, the computer should help alleviate any tedious tasks through automation.

Existing methods for synthesizing human motion fall short of meeting all of the requirements of interactive applications. These existing methods can generally be divided into two categories: methods used in practice in video games or training simulations and example-based motion synthesis methods developed primarily at academic institutions. In general, the methods used in practice are efficient but fail to meet quality requirements, while the existing example-based synthesis techniques produce high-quality results but are often too inefficient to be used at runtime. Table 1.1 summarizes these generalizations. Chapter 2 goes into more detail on how existing example-based methods and methods used in practice fail to meet the needs of interactive applications.

The goal of my work is to provide insight into how to develop example-based synthesis methods that are efficient enough to be used in interactive applications but can still synthesize the high-quality, accurate motions characteristic of these techniques.

1.2 Technical Solutions Overview

This section describes in detail the individual technical problems I have addressed with my work and how I have solved these problems.

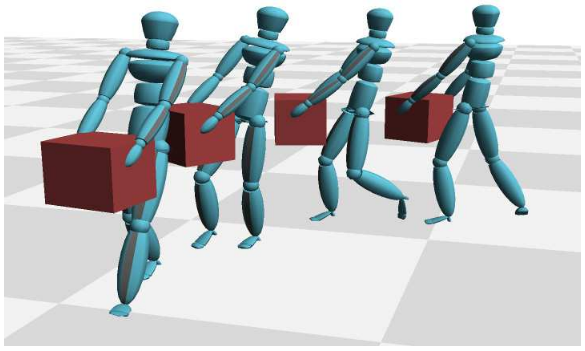


Figure 1.3 A motion generated by splicing the upper body of a person carrying a heavy box with the lower body of a person walking in a curve.

1.2.1 Motion Splicing

Often the problem of character control is divided into locomotion and action. For example, a character can carry a box, punch, carry a suitcase, or wave, independent of whether it is walking straight, walking in a curve, walking up some stairs, jogging, or just moving from one foot to the other. Interactive applications that allow many different actions and kinds of locomotion must be able to generate motions for every possible *combination* of locomotion and action. Synthesizing these motions using current example-based methods requires capturing examples of the character performing all possible combinations. If n locomotion examples are needed and there are m actions that the character can perform, nm examples would need to be collected all together. It would be attractive if a character's upper-body action and lower-body locomotion could be captured independently, requiring only $n+m$ examples. However, this independent treatment requires a method for *splicing* the separate pieces together. For example, to create a motion of someone stepping up onto a platform while carrying a cup, an application could splice the lower body of a character stepping up onto a platform with the upper body of someone carrying a cup.

I have developed a simple and efficient technique for synthesizing high-fidelity motions by attaching, or *splicing*, the upper-body action of one motion example to the lower-body locomotion of another. Existing splicing algorithms do little more than copy degrees of freedom (DOFs) from one motion onto another. This naïve DOF replacement can produce unrealistic results because it ignores both physical and stylistic correlations between various joints in the body. My approach uses spatial and temporal relationships found within the example motions to retain the overall posture of the upper-body action while adding secondary motion details appropriate to the timing and configuration of the lower body. By decoupling upper-body action from lower-body locomotion, my motion synthesis technique allows example motions to be captured independently and later combined to create new, natural-looking motions.

To illustrate my splicing technique, I give a concrete example. Imagine that we have motions of a character walking in a curved path and of a character walking in a straight line while carrying a heavy box. Using motion splicing, one can generate a motion of the character carrying a heavy box down a curved path. This motion will display many essential characteristics of the original box-carrying motion. For instance, the arms will stay the same distance apart throughout the motion, and the upper body will lean back slightly because of the weight of the box. The motion will also exhibit characteristics that are a direct result of the correlations between the upper body and lower body. For example, the shoulders will rotate and bob slightly with the stride of the lower body, and the entire motion will lean in towards the turn. In particular, note that splicing produces motions with correct physical effects without explicitly modeling physics; these physical effects are transferred because the original example motions encode them. In contrast, naïve DOF replacement produces a motion where the upper body leans forward and wobbles unpredictably in relation to the lower body.

1.2.2 Gaze Control

Humans often convey their intentions through subtle cues before performing actions. In particular, the head of a person often moves to fixate on a point before acting. I call these head cues *gaze*, and they are a natural way to convey goals. Unfortunately, virtual human characters often lack gaze cues. Sometimes video game developers do adjust a character's head facing in order to convey characteristics like "aiming" direction, but the motions employed often look unrealistic as they are based on simple inverse kinematics methods, such as just twisting the character's neck.

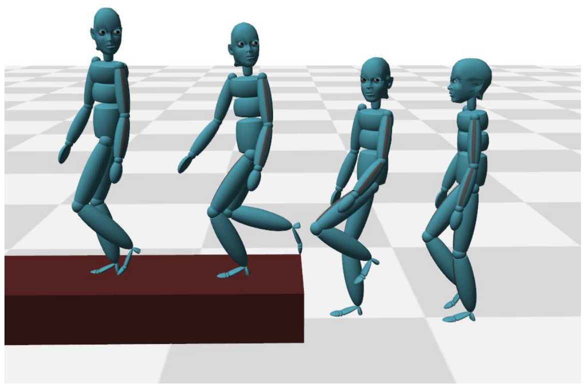


Figure 1.4 A motion generated by applying my gaze control model to adjust this character’s gaze towards the left.

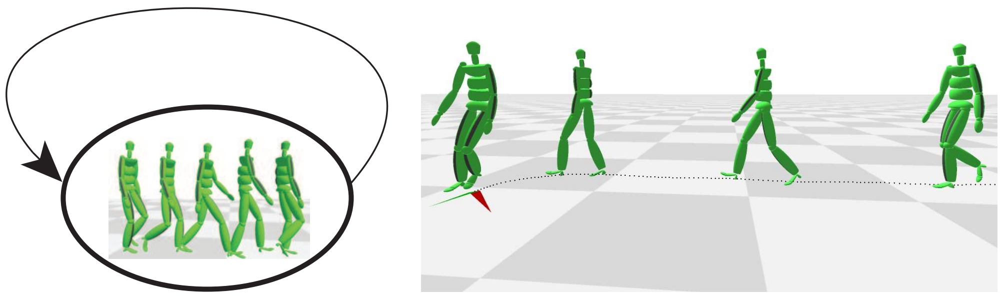
Furthermore, the way an individual adjusts their gaze depends heavily on their personal movement style. It is not sufficient to find a single way to adjust gaze for every character.

I approach this problem from an example-based perspective. If one could determine how to adjust the gaze of a character using motion-captured examples, it would be possible to generate accurate, quality motion that includes gaze cues. I represent the gaze style of a character using a low-dimensional representation of gaze change motions that is informed by existing biological research on how humans adjust their gaze. The model uses raw motion capture data of a person changing their gaze to build a parametric gaze map that can be used at runtime to quickly control the gaze of a character.

Interactive gaze control can be used to create a character that reacts faster to its environment. For example, a character could look at interesting objects as they get near them and look along the direction of travel when moving, all without appearing unnatural. One nice characteristic of interactive characters with controllable gaze cues is that the motion of the head can lead the motion of the character. For instance, when a character is standing still and a user requests that the character turn to the right, a realistic motion takes time to begin moving the character in the requested direction, which can cause a user to be unsure of whether the character has received the request. On the other hand, if the character begins turning his/her head towards the right, the user knows that the character is reacting. The head motion lends a sense of responsiveness to the character because it mimics how humans react in similar situations. See Figure 1.4 for an example of my gaze model being used to make a digital character look towards the left. Unlike the results garnered using a simple head twisting technique, my results exhibit natural motion characteristics such as overshoot and asymmetry.

1.2.3 Parametric Motion Graphs

For interactive applications, it is not only important to quickly generate a motion that meets specified constraints but also to transition between different motions rapidly and smoothly. One downside to using an example-based motion synthesis method is that for any complex application it would require a large number of example motions in order to synthesize all possible user-requested



(a) (b)

Figure 1.5 (a) A parametric motion graph for walking through an environment with smooth turns. (b) An interactively controlled character using the walking graph. The character is in the process of turning around in order to walk in the user requested travel direction depicted by the arrow on the ground.

streams of motion. But an interactive application must be able to find and transition between motion clips quickly in order to minimize latency. So, the problem faced by example-based motion synthesis researchers is one of organization. How can we represent all possible transitions between example motions in a highly-structured way in order to facilitate fast decision-making while not sacrificing the quality, accuracy, or ease of authoring gained by using an example-based synthesis approach?

To tackle this problem, I have developed a highly structured, example-based motion synthesis data structure called a *parametric motion graph*. A parametric motion graph describes possible ways to generate seamless streams of motion by concatenating short motion clips generated through parametric synthesis. Parametric synthesis allows accurate generation of any motion from an entire space of motions, parameterized by a continuously valued parameter. For example, parametric synthesis can generate motions of a person picking up an item from any location on a shelf. While neither seamless motion concatenation nor parametric synthesis is a new idea, by combining both synthesis techniques, parametric motion graphs can provide accurate control through parametric synthesis and can generate long streams of high-fidelity motion without visible seams using linear blend transitions.

To illustrate the utility of parametric motion graphs, consider a concrete example. I can create a character that can be directed through an environment with continuous steering controls. Using an existing blending-based parametric synthesis method, I first build a parametric motion space of a person walking at different curvatures for two steps. Next, I quickly build a parametric motion graph from this motion space using the algorithm presented in this thesis. The resulting graph contains a single node, representing the parametric walking motion space, and a single edge that starts and ends at this node (see Figure 1.5a). This edge describes how to transition from the end of a generated walking clip to any generated walking clip in a subspace of the motion space. This simple structure organizes many motions in a way that allows efficient character control at run-time. By translating a user's desired travel direction into desired curvature requests, the parametric motion graph can be used to synthesize a continuous stream of walking motion that reacts to a user's commands. This stream of motion will be smooth, will run at interactive rates, and will only contain high-fidelity transitions between clips of motion. Figure 1.5b shows a screen capture of an interactive walking character using parametric motion graphs. Unlike other techniques that can create interactive walking characters, my technique requires little authoring effort, is capable of accurate motion generation, and works with a wide range of different motions. For instance, once I have an interactive walking character, it is easy to create a character that locomotes by running or cartwheeling simply by building a parametric motion space of running or cartwheeling motions, also parameterized by curvature.