School of Mathematics and Systems Engineering

Reports from MSI - Rapporter från MSI

Construction of a Motion Capture System

Jonas Lindequist & Daniel Lönnblom

Sep
2004

MSI
Växjö University
SE-351 95 VÄXJÖ

Report 04087
ISSN 1650-2647
ISRN VXU/MSI/DA/E/--04087/--SE

Abstract

Motion capture is the process of capturing movements from real life into a computer. Existing motion capture systems are often very expensive and require advanced hardware that makes the process complex. This thesis will answer the following question: is it possible to create an optical motion capture system using only a single low cost DV-camera (Digital Video Camera), that still will produce accurate motion capture data? To answer this question and construct our motion capture system we need to complete these following steps:

- Create a usable film sequence.
- Analyze the sequence.
- Create motion capture data.
- Apply the motion capture data for 3D character and analyze the outcome.

The method chosen for this thesis is constructive research. In short terms it is the study of whether we can or cannot build a new artifact. The following theoretic tools were used in the process of creating a motion capture system: Color theory, RGB, Connected component labeling, Skeletons in 3D animation, Calculating angels using trigonometry, .x files and Quaternions.

We have found that an optical motion capture system is very complex and it is hard to produce as a low budget system. Our attempt did not live up to our expectations. The idea with using only one DV camera was to simplify the system since it would require no calibration or syncronisation. It would also make the system cost efficient and more available to the general public. The single camera solution unfortunatly created a number of problems in our system. Our system does however work with less complex movements. It can produce motion capture data that is accurate enough to be used in low budget games. It is also cost effective compared to other systems on the market. The system has a very easy setup and does not need any calibration in addition to the init position.

Keywords: Motion capture system, image analysis, marker tracking, character animation and video capturing.

Index

List of figures

1. Problem introduction

Motion capture is the process of capturing movements from real life into a computer, most often in XYZ-coordinates. Usually humans are the actors in motion capture but virtually any movement is possible to record, like animals or objects. The movement can be post processed or applied to a 3D-character to make it mimic the movements of the actor, see figure 1.1. [1, 2]

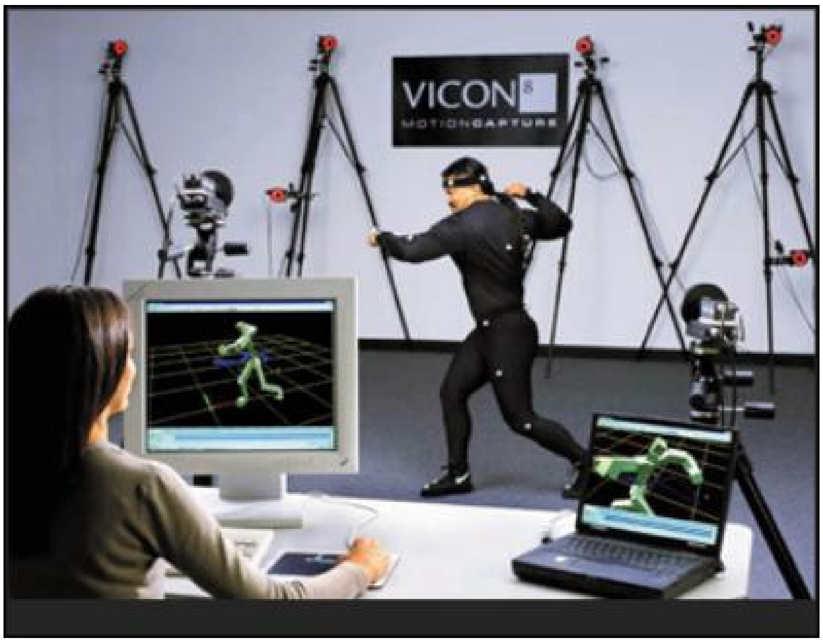


Figure 1.1 Capturing of a motion

1.1 Background

Before anything else, we would like to explain more about different available motion capture systems and how they are used today.

1.1.1 Motion capture systems

There are a few different ways of capturing a motion, the most commonly used methods are optical motion capture, electromagnetic systems and electromechanical systems.

Optical motion capture uses cameras to record an actor performing movements. The amount of cameras used differ between the systems but most systems use 4 cameras or more. [1] The cameras used, are often very high performance cameras capturing up to 2000 frame per second. When using a very high capturing rate, the movement from a frame to the next will be very small. The actor wears a suit of markers that later will be scanned for in the video clip. The markers are made of some material intended to stand out from the surroundings, often some reflecting material or LED:s. The cameras are placed at different angles towards the actor and a sequence is filmed which afterwards can be analyzed and the positions of the markers stored in a computer. The different inputs from the cameras are then combined and processed, and a three-dimensional representation is produced. This output is known as motion capture data or motion data and can be analyzed or applied to a 3D-character. [2] The pros of using an optical system are that it is capable of recording the most complex motions with the least deterioration in quality. It also allows the actor to move unhindered, without any suit or other measurement devices attached to the body. The problems with optical motion capture are when the markers get occluded. Also the processing time is very high and the data produced by an optical motion capture must often be adjusted and cleaned up and this combined with the demanding calculations makes it a non real time system. An Optical motion capture system often costs between $100,000 and $250,000. [1, 2]

In an electromagnetic system, the actor wears a suit with magnetic sensors attach to it. The systems then produce an electromagnetic field, using a magnetic field generator. The sensors return both their position and rotation and due to that only 10-20 markers are needed to record the whole body. Advantages with the Electromagnetic system are that the data is clean and the processing time is very low. I.e. it can be used in real-time applications like television or VR. The disadvantage is the sensitivity of the sensors that easily can pick up electromagnetic noise as well as the presence of ferrous metals. Also, some equipment is required and can be disturbing for some movements. The magnetic sensors are often linked to a computer using cables, and that will restrict both movements and range. An electromagnetic system costs between $5,000 and $150,000

Electromechanical systems let the actor wear a mechanical suit that is composed of potentiometers measuring the joint angles and extensions. The system’s processing-time is very fast and there is no electromagnetic interference. The disadvantages are that the suit is encumbering and heavy. Also the mechanical components get worn out by time resulting in less accurate data. An electromechanical system cost between $5,000 and $10,000. [1]

1.1.2 Motion capture markets

Motion capture has a wide field of usage. Although motion capture most often is associated with the entertainment industry, such as movies and videogames, there are several other usage areas. Examples are in medicine, sports, industry and military.

In Medicine motion capture is used to perform gait analysis, these are studies of human motion. The advantage of using motion capture is that you separate the different mechanisms used when for example performing a walk cycle, thereby it is simple to detect abnormalities and changes. Motion capture can also be a useful help when creating ergonomic workstations or motions.

On the sports market motion capture is used as a analyzer of athletes in order to improve their performances. The most common sport using motion capture is Golf. The swing of a golfer can be captured using motion capture and then analyzed or compared to a professional golfers swing. The advantage gained from motion capture compared to video capture is that you get a three dimensional view from motion capture.

Motion capture can be very useful in different industries because it can quickly measure small fast movements and therefore be used as a process controller. It can also be used to analyze vibrations on equipment or vehicles or as a tool in virtual industrial design.

Motion capture can also be used in the military to contribute to the training of military personnel. Military field exercises, virtual instructors, and role-playing games are just some of the possibilities that the Motion capture system can provide.

However, the entertainment market is the market that has made motion capture a common term. Almost all videogames containing human motion uses motion capture to accurately generate animated characters. It is a cost effective method that produces realistic motions compared to manually animate the character. Television and feature films use motion capture mostly to create digital stuntmen, these are animated character that perform stunts that are impossible for a live actor. Motion capture can also be used to create cartoons and give the cartoon characters realistic movements in a effective way.[3]

1.2 Problem discussion

As we have described above there already exist several motion capture systems that are used in many different fields where the entertainment sector is the largest. However, a drawback with existing motion capture systems are that they tend to be very expensive and require advanced hardware that makes the process complex. The question we raise is if it is possible to develop a cheap, simplified version of a motion capture system. With price and simplicity in mind we believe an optical motion capture system is best fitted since it uses to us accessible equipment to a reasonable cost.

The main reason why existing optical motion capture systems are so expensive is because of their choice of hardware. Generally, expensive high performance cameras with very high frame rate are used. We will examine whether or not it is sufficient to use only a single DV camera to record a three dimensional motion. To do this we will use a low end DV camera with the ability to capture only 25 frames per second, compared to other systems on the market that uses cameras with up to 2000 frames per second. [5] [6]

Another drawback with existing optical motion capture systems, except the high costs, is the complicated setup of the capturing environment that is needed. This is usually so complex that you need a special studio for all the hardware setup and careful calibration. To keep costs down and simplicity high our solution also must include an easy set up that can be done in any dim room rigged with a DV camera. To make this possible our system will use fluorescent markers that reflect the light of a UV light tube. [7]

Due to the camera restrictions we also need to examine the best placements for the markers. They cannot be too many because they then will get mixed up, and they cannot be too few or we will loose vital information about the body movements. Normally only one color, white, is used on the markers. A way to make it easier to keep the markers separated would be to use more colors and give markers close to each other different colors.

All together our idea is to create an optical motion capture system using only a single DV camera and different colored markers in an ordinary room equipped with an UV light tube.

There are still four main problems that need to be solved before we have reached our goal. The first is to create a usable film sequence. We must produce a film sequence from which we can clearly distinguish the markers from each other. The sequence must be free from noise and background objects that might interfere with our markers. Many motions performed by a human actor have a high speed, such as running or jumping. The film sequence must be able to represent these motions without any long gaps between the different frames and with as little motion blurring as possible. Here we must also consider how we should place the markers, what colors to use and where to place the different colors. Due to the fact that there are only a few different colors that reflect the light with enough brightness and different colors can reflect the same color, most colors must be used more then once. [8]

The second problem to manage is the analyzing of sequences. We need to find the markers in a frame and when we have found the initial markers we need to track them throughout the sequence. Moreover, we must find a way to handle occluded and disappearing markers. For the analyzing to be as efficient as possible since processing a complete motion is a time consuming task, we need to implement efficient algorithms.

Thirdly, we must create motion data. When all marker positions have been found, we have to calculate how to rotate the skeleton to match the markers. We must also consider how to distinguish between rotations in and out from the camera view 2D plane.

The last issue involves using data for 3D-characters. To be able to apply the motion data to a 3D-character we must first convert all rotations into quaternions. We also have to understand the .x format, the file format used in Microsoft DirectX, in order to add our key frame motion data into it. [9]

1.3 Problem

Is it possible to create an optical motion capture system using only a single low cost DV-camera, that still will produce motion capture data with an errormargin of less than 10%, meaning that the motion performed by the actor will be mimiced by the 3d character within the errormargin stated above.

To answer this question and construct our motion capture system we need to complete these following steps:

- Create a usable film sequence.
- Analyze the sequence.
- Create motion capture data.
- Apply the motion capture data for 3D-character and analyze the outcome.

Where analyzing a sequence and creating motion capture data will be the two biggest steps and these are the steps that we primary will focus on. Finally, we will test our solutions and the complete system.

1.4 Limitations

We will construct a prototype for an optical motion capture solution. The goal is not a complete professional system but more of a simplified solution that can be developed into a fully working solution. The reasons for these limitations are the time avaliable for the project and the budget we have to work with.

2 Method

The purpose with this chapter is to clarify how we carried out this project. The method chosen for this thesis is constructive research as described in Järvinen. [10]

2.1 Constructive research

In short terms this is a study of whether we can or cannot build a new artifact. Järvinen names this constructive research. You can divide research into basic and applied research. In basic research the main purpose is to observe what already exists, for example in theory-testing research, in case studies or quantitative surveys. Applied research on the other hand uses the results from basic research and applies this to an object to achieve a new artifact or to reach a desired final state. Constructive research is a kind of applied research except that in constructive research the final product can be a prototype or a simple plan. The most typical use of constructive research is to build a new innovation and this process is based on existing knowledge and the addition of new technical advancements.

This study matches Järvinen's description of the constructive research very well. Our purpose is to create a new artifact with help from existing theories and systems combined with new ideas. Our goal with this research is a prototype of how motion capture can be achieved at less cost.

2.1.1 The building process

When using constructive research you divide the process of creating a product into three steps.

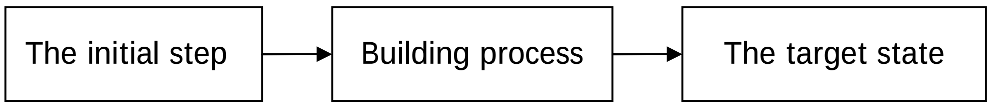


Figure 2.1 The building process

The initial state describes old artifacts or research that the building process will be based on. The initial state can be modified to gain performance and or new solutions. Our initial state is based on existing optical motion capture systems as described in chapter one.

The building process is to implement the desired changes from the initial state to reach the target state if the target state is known. If the target state is unknown there are two ways to go. Firstly you specify the target state and then try to implement measures to achieve this state. The other way is to perform both the target seeking and the implementation in parallel. The figure below describes these two different ways.

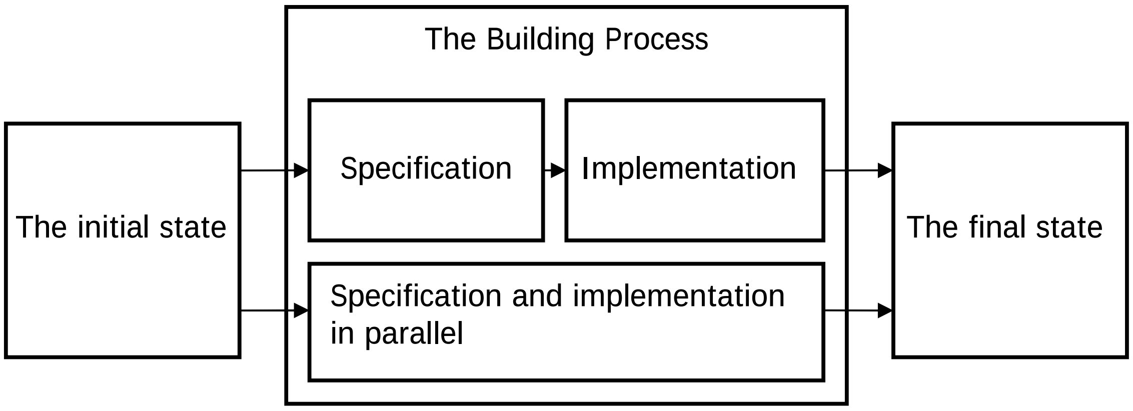


Figure 2.2 Different paths of the building process

2.1.2 Specification process

This process goal is to produce a model of the target. The model contains information requirements. To determine the information requirements one uses one or more of these four strategies.

- Asking the end user
- Deriving from an existing information system
- Synthesis from characteristics of the utilizing system
- Discovering from experimentation with an evolving information system

2.1.3 Implementation process

The research question for the implementation process can be formulated as follows: Having the certain initial and target states and particular resources, how can we build an artifact satisfying the given specifications? There are different alternatives to solving this "how" question. A researcher can apply two different heuristics, problem reduction and state-transition.

In problem reduction the researcher divides the problem into sub-problems, which are then divided in sub-sub-problems until a solution is found to the sub-problem. To perform problem reduction there are two principles that can be applied, breadth first or depth-sort. The breadth first principle divides a problem into sub-problems and tries to solve all the sub-problems at the highest level first If a sub-problem can't be solved it continues to the next lower level, and so on. In the depth first principle the researcher first seeks the most difficult sub-problems and tries to solve it. After that problem is solved the next most difficult sub-problem is found and solved. If the problem is not solvable the construction process stops.

In the state-transition heuristics the original problem definition (the initial state) is transformed to the target state in a sequence of transitions. The transitions must be reported and if there are too many transitions the least important ones can be excluded.

2.1.4 Parallel specification and implementation processes

The reason why an artifact is specified and implemented in parallel is often because it is difficult to predict a target state that has never existed. By using a mix of specification and implementation the target state can be outlined during the building process. There is no need to have exact knowledge of the target state in the initial state. By using the parallel specification and implementation process the building process can be evolved by solving one problem at a time and then assemble it into a final target state.

2.1.5 Our choice of method

We have chosen to use constructive research with Parallel specification and implementation process for this thesis. We used the parallel approach since we had a target state but it wasn't completely defined at the beginning of the building process. The target state evolved during the implementation process

For the specification process we will use these two strategies

- Deriving from an existing information system
- Discovering from experimentation with an evolving information system

We derive our artifact from old research and technologies. Our purpose is then to evolve this artifact into the target state by experimenting with new technology.

For the implementation we use problem reduction with the breadth first principle, the main problem is divided into sub-problems and solved at the highest level possible. By using this method we can solve the main problem in an efficient way and during the building process we can evolve and revise the target state.

2.2 Use of theory

Here we would like to comment on our use of the theories presented in this thesis. With theory we mean the tools we have used that are already created by others. Since this is constructive research we have no intentions to test our theories or see how well they matches reality. Instead we use them as tools in our construction process. This means that we accept the theories as correct, and this is possible due to the amount of evaluations already made and the high level of acceptance they already have.

2.3 Scientific quality of this thesis

Through out the work, we have strived for correctness, by checking and re-checking that all the facts we present are right. Järvinen describes some desirable attributes for a paper that beside correctness include significance, innovation, interest, timeliness, accessibility, readability and style. [10] We find our work *significant* since a good solution would reduce costs dramatically for motion capture system users. It is *innovative* since it is a constructive research and hopefully you find it as *interesting* as we do. Moreover, the results will be *timely* because there are many markets for motion capture systems today as described in chapter one. Finally, we have made the report easy to understand and well structured so it will be *readable* and *stylish*.

With Järvinen’s words in mind and by carefully describing our methods for this research, the theories we build it upon and the results we maintain, we believe to have secured a high degree of scientific quality in this thesis.

3 Theory

In order to show where the different theories are used we have divided them into four main categories, based on the four steps in the problem presented in the first chapter.
This is how we have used the theories:

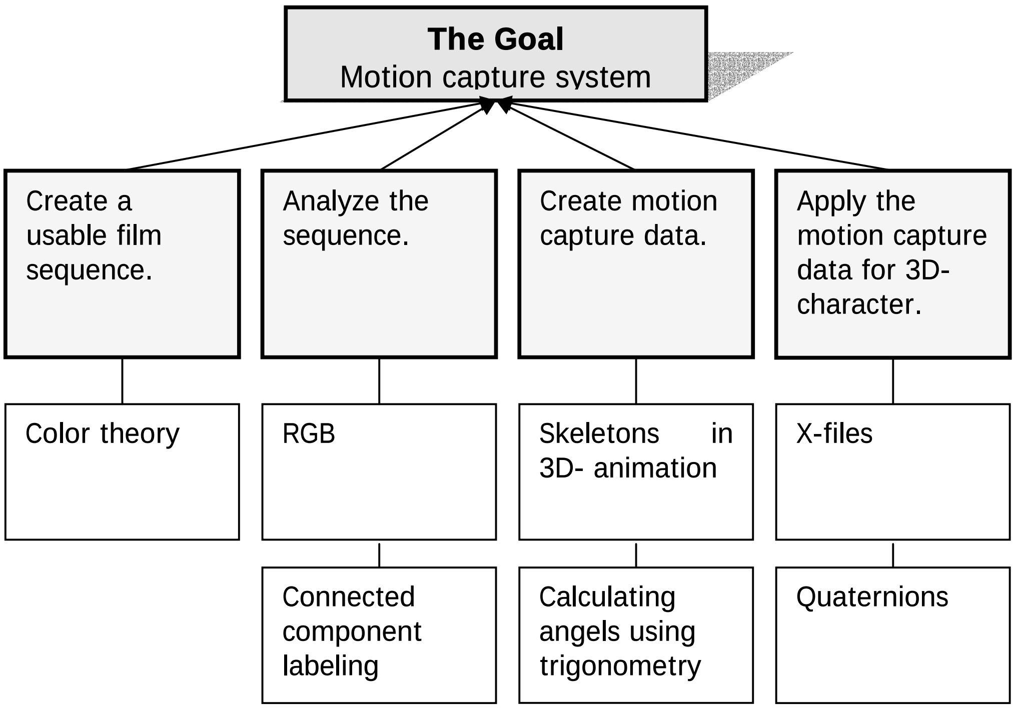

The Goal
Motion capture system
Create a usable film sequence.
Analyze the sequence.
Create motion capture data.
Apply the motion capture data for 3D-character.
Color theory
RGB
Skeletons in 3D- animation
X-files
Connected component labeling
Calculating angels using trigonometry
Quaternions

Figure 3.1 Theory structure

Through out the rest of this chapter these theories presented above will be explained in more detail.

3.1 Color Theory

Our eyes can see visible light in a spectrum ranging from red through orange, yellow, green, blue and violet. Above violet is ultraviolet light, which we cannot see.

A phosphor is any substance that emits visible light in response to some sort of radiation. In other words, a phosphor converts the energy in the radiation into visible light. So a fluorescent paint on a fluorescent poster contains a phosphor that converts UV radiation into a specific color of visible light. Normal colors simply reflect light, but a fluorescent color absorbs the radiation and re-emits it in the visible spectrum so it looks much brighter than a normal color. Different colors can emit the same light when exposed to UV light, this means that we can only use a limited number of colors. In example colors like white and blue tend to look the same as they both reflect white light [11].

3.2 RGB

The RGB system uses the primary colors Red, Green and Blue. These colors mixed together create the color white. Each primary color can contain 256 different shades which gives the RGB system a total of 256x256x256 = ~16 millions different color combinations. Every color combination has a complementary color that is the opposite hue (color value) of the original color. To distinguish colors from each other the best way is to use the three primary colors and their complementary colors. These colors will have the optimal distance to each other. [12]

3.3 Connected component labeling

This method scans an image line by line and gathers the intensity values from every pixel. The pixels are then grouped together based on their pixel connectivity. Pixel connectivity means that a pixel and its surrounding pixels of the same intensity are combined in to a single component. [13]

When classifying components you use a decision theoretic approach to the image. Classification algorithm employs two phases, training and testing. The training phase gathers characteristic properties from the image. These characteristics are saved as independent classes for the testing phase. The testing phase analyzes an image and then compares it to the trained classes descriptions. When a component of the image matches a class it is classified as a component and the component is saved as an independent object. [13]

3.4 Skeletons in 3D-animation

A skeleton is a hierarchy of joints and limbs. Combined they define a skeleton, in this thesis very similar to a human skeleton. When a skeleton is created joints are placed at similar positions as real joints on a human body, although you often reduce the amount of joints to the ones necessary for the animation. The joints are connected to each other with limbs (also called bones).

All movements the skeleton can make are based on rotations. When you move a joint, it is possible that the joints higher up in the hierarchy (parents) must be rotated to mach the movement. For example if you drag in the hand-joint, it is probable that the elbow-joint also must move (rotate) since all the limbs length are non-changeable.

Since it is only necessary to store data to control the skeletons joints, instead of all vertices in a mesh, the memory-storage is minimized. Also the joints do not need to be saved in every frame, only in special key frames. The key frames can occur on different frames for different joints.

When applying a mesh (3D-character) to the skeleton, each vertex in the mesh is associated with one or more limbs and gets a relative position. The vertex then follows its limb as the limb moves. In the cases where a vertex is associate with more then one limb, the limbs are weighted and the vertex will follow both limbs according to there weights.
This technique is called skinning or vertex blending.

It is very efficient to store an animation as a skeleton because a skeleton will work with many different meshes. This means that you can reuse a skeleton’s movements for different purposes. [14]

3.5 Calculating angels using trigonometry

These are the formulas for calculating the angles in the 3D-space, knowing only the coordinates of the parent-point and the child-point and the initial length of the line between them. [15]

First, calculate the angle in the 2D-plane, facing the camera.

R1 = **arctan**(dy/dx)	if dx > 0
R1 = **arctan**(dy/dx) + 180	if dx < 0
R1 = 90	if dx = 0 and dy > 0
R1 = -90	if dx = 0 and dy < 0

Where

dx = the child x-coordinate minus the parent x-coordinate.
dy = the child y-coordinate minus the parent y-coordinate.
R1= the angle in the 2D-plane.

Second, calculate the rotation out from the 2D-plane (depth-rotation).

R2 = **arccos**(cL / L)

Where

L = the initial length between the two points.
cL = the current length between the two points.
R2 = the rotation, out from the 2D-plane.

This statement is only valid for cL <= L and L != 0.

3.6 .x files

Microsoft DirectX uses a file format called .x for modeling and animation. The .x format is a flexible file format containing mesh information, bone structures and animation sets. The file format is supported by the DirectX API and can easily be implemented in C++ code using the DirectX API.

.x files are structured using templates, the ones used in this thesis are header, mesh, frame and animations sets. The header contains description of the file version and other optional information.

The mesh contains four elements. The first is the number of vertices building the mesh, the second is a list with the vertices and their x, y and z coordinates. Thirdly are the number of faces and the last element is a list of the faces and which vertices each face is built from.

The frame is the template for bone structuring in .x files. The frames give you information on how to build the skeleton for animation of a mesh. The mesh is attached to a frame by creating a frame transform matrix that is then used to transform the mesh and thereby create an animation.

Animation set templates are used to structure the animation data, the structure uses four elements of information. Element one is the name of the frame that is connected to the animation set. Animation key is the second element and defines if the animation contains rotations, transformations or scaling. These can be combined to achieve the final animation. The third element tells us how many key frames the animation will use and the last element contains animation data for every key frame defined in the third element.

.x files can be written in ASCII which makes the format easy to parse and to write.
To parse an .x file you read the file and extract the templates. Then you parse the information from every template that is contained inside brackets. This way you can create a complete .x files structure in a computers' memory and you can then modify this information according to your needs. When you have parsed an .x file and modified it you have to write it back to a file. The process is similar to the parsing you simply gather the templates and write them back in the same order they where parsed. The modifications will remain and a new .x file created. [9]

3.7 Quaternions

Quaternions are a way to store angles in 3D-space and solves angular problems like the Gimbal-lock, when two axis of rotation align. [19]

Quaternions are four-dimensional numbers that can be written as

$$a + bi + cj + dk$$

where a, b, c and d are real numbers and i, j and k are the unique quaternions. For i, j and k the following rules apply

$$i^2 = j^2 = k^2 = ijk = -1.$$

The multiplication with quaternions is associative but not commutative. [14]

4 Practical solution

In this chapter we will explain our solution to the four steps our motion capture system is built upon. We will start by explaining how a suitable film sequence can be created. Then follows how the analyzing of a sequence has been performed. After that is a description of how we transformed the obtained data into angles and last how we applied those angles to a 3D-character.

4.1 Create a usable film sequence

To create a usable film sequence the most important factors are the equipment used and the environment in which we produce the video sequences.

4.1.1 Markers

As markers we use carbon paper stripes with fluorescent colors in different colors. This proved to be a good idea since they reflect different colors very intense. As described in our color theory we are limited to use a small number of colors. The colors that worked together were red, green, blue, pink and yellow. When yellow was exposed to UV light it emitted a green light but this worked since green emits a much darker shade of green. Since the frame rate of the cameras in use is very low, the position of each marker can change drastically from one frame to the next. Therefore the markers cannot be too many or too close to each other. If they were, they constantly would be mixed up and the errors would increase by a multitude.

Our approach uses seven markers, see figute 4.1, this is enough to be able to calculate all major joints of the human body but still not to many to increase complexity to an unnecessary level.

The markers placements should be the hands, elbows, shoulders, hip, knees and feet. We could increase the number of markers but the extra markers wouldn't serve any purpose since they do not represent rotating limbs [2, 18]

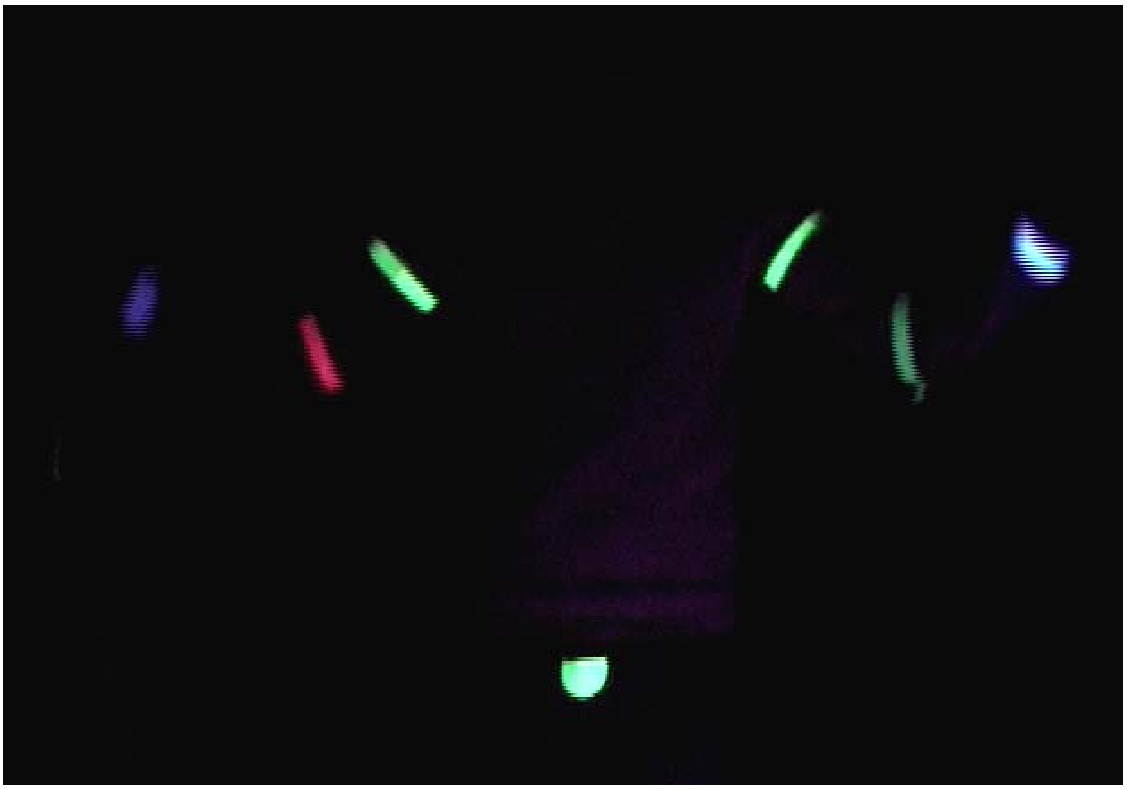


Figure 4.1 Markers

4.1.2 Video Setup

To capture a sequence on video we used a Canon DV MV400 Camera with the ability to capture 25 frames per second (standard PAL Format) in a resolution of 720x576 pixels. The camera was placed on a tripod to achieve a steady image. [6]

4.1.3 Environment

To create the surroundings we dimmed the room enough so the fluorescent markers would be easily distinguished and the actor still would be able to see what he performed.

4.1.4 Integrating video into the application

For integrations DirectShow is used. DirectShow is a standardized Microsoft Win32 API to access video devices from an application. The DirectShow is an interface that creates a streaming media layer on top of your application, this makes it easy to show and access video streams directly from your application.

DirectShow uses filters and graphs to process multimedia data. A typical graph network of filters works as follow. [16]

- A source filter reads the data from the media file.
- A parse filter extracts video and audio data and sends this to the decoder.

- The data is decompressed.
- A render filter takes the uncompressed data and draws it.

4.1.5 Grabbing individual frames from video

To extract a frame from a video sequence we need to access the video frame by frame and we need the ability to stop the video at any given time. DirectShow supports a filter for grabbing frames called SampleGrabber. We simply add the SampleGrabber filter to the stream and when we render the stream each frame will be collected and saved to a bitmap. The bitmap can then be processed within the application. [17]

4.2 Analyzing a sequence

To be able to analyze a film sequence we need a marker-tracking algorithm. In our theory we have described how to find which pixels are markers and which are not. The process is called connected component labeling. In this chapter we will explain the algorithm we created to be able to find the markers in the film, and to follow them as they move. This includes tracking of the markers, locating markers in the first frame, explanation of the initial position, and error detection and solutions. Also, we will explain some errors that may occur and how to take care of them.

4.2.1 Tracking the markers during the sequence

From the initialization we have gathered information on how many markers are present on the body, the colors of each marker and the position.

For each of the following frames the following steps need to be performed. First we need to estimate the new marker position. Then we need to scan for the markers at the estimated position. Last, if a pixel with approved color has been found, we need to make sure that the pixel belongs to a cluster and that it is the right one based on its color.

To estimate where the marker most likely will be positioned in the current frame we lookup the position values of the markers positions at the two previous frames. Then calculate the estimated directional vector using

directionalVector[(*dx1-dx2*) , (*dy1-dy2*)]

and apply the directional vector to the position of the previous frame.

previousFrame + directionalVector = estimatedPosition

This will give an approximation of where the scan should begin.

In the second step where we scan for the markers, we start a scan at the estimated position. The scan is performed by creating a bounding box which size is increased for each scan iteration, see figure 4.1.

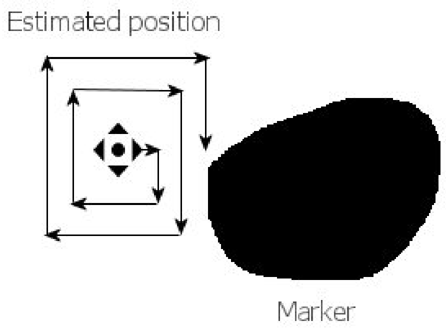


Figure 4.2 Marker tracking algorithm

For each iteration the scan performs a scanline inside the bounding box limits and searches for pixel values above the background threshold. If an approved pixel (above threshold) is found the scan is stopped and a new marker is found. This does not necessary mean that the right marker has been found and we apply the cluster check algorithm.

When the scanline algorithm has found a pixel we have to determine if the pixel is part of a marker or if its noise. This is done by creating a bounding box centered at the found pixel and then the box is scanned for more approved pixels. It enough pixels are found to meet our minimum threshold, the cluster is considered to be a marker, if not, the pixel is considered to be noise. To find out if marker that has been found is the right one, we apply the color comparison algorithm.

A new marker has been found where it is predicted to be, but to determine if it really is the right one we apply a color comparison algorithm. The color comparison algorithm works as follows. It compares the color value associated with the marker with the color value from the currently found pixel. If the colors are the same (within a certain margin), we can conclude that we have got the right marker. If they differ, the marker is not the right one and we continue the scan for markers.

Code:

```
For each frame{
     For each point{
          Retrieve offset from last position;
          Start scan at last position + offset;
          Scan using a growing rectangle;
          If pixel above threshold found{
               If pixel is part of cluster {
                    If color equals last point{
                         Save point to list;
                    }
               }
               else
```

```
                    Continue to grow scan rectangle;
          }
     }
}
```

4.2.2 Locate markers in the first frame

To initialize the tracker we need to have the markers in a starting position, see figure 4.3. In this position all markers must be visible and the background noise must not exceed the threshold value for the background. The first frame is analyzed using a scanline algorithm, see figure 4.2, where every pixel in the frame is examined and the RGB-values compared to the threshold-value. At least one of the three color components (red, green and blue) must excide the threshold-value.

When a pixel with enough brightness is found, a second algorithm will search a rectangle area below the pixel and save every approved pixel there as a cluster. That is, the marker is represented as a cluster of pixels. The top-left and bottom-right of the approved pixels in the rectangle are saved as the bounding box of the cluster, and are also used to locate the center of the cluster. If the number of approved pixels in the rectangle below is less then the threshold for the minimum number of pixels for a cluster, the pixel will be discarded as noise. Here, the RGB-value for the cluster is also stored for later to make it easier to distinguish the clusters from each other. [13]

The marker (cluster) is then added to a list of found markers, this is done to exclude this rectangle from the scanline algorithm during the rest of the scan.

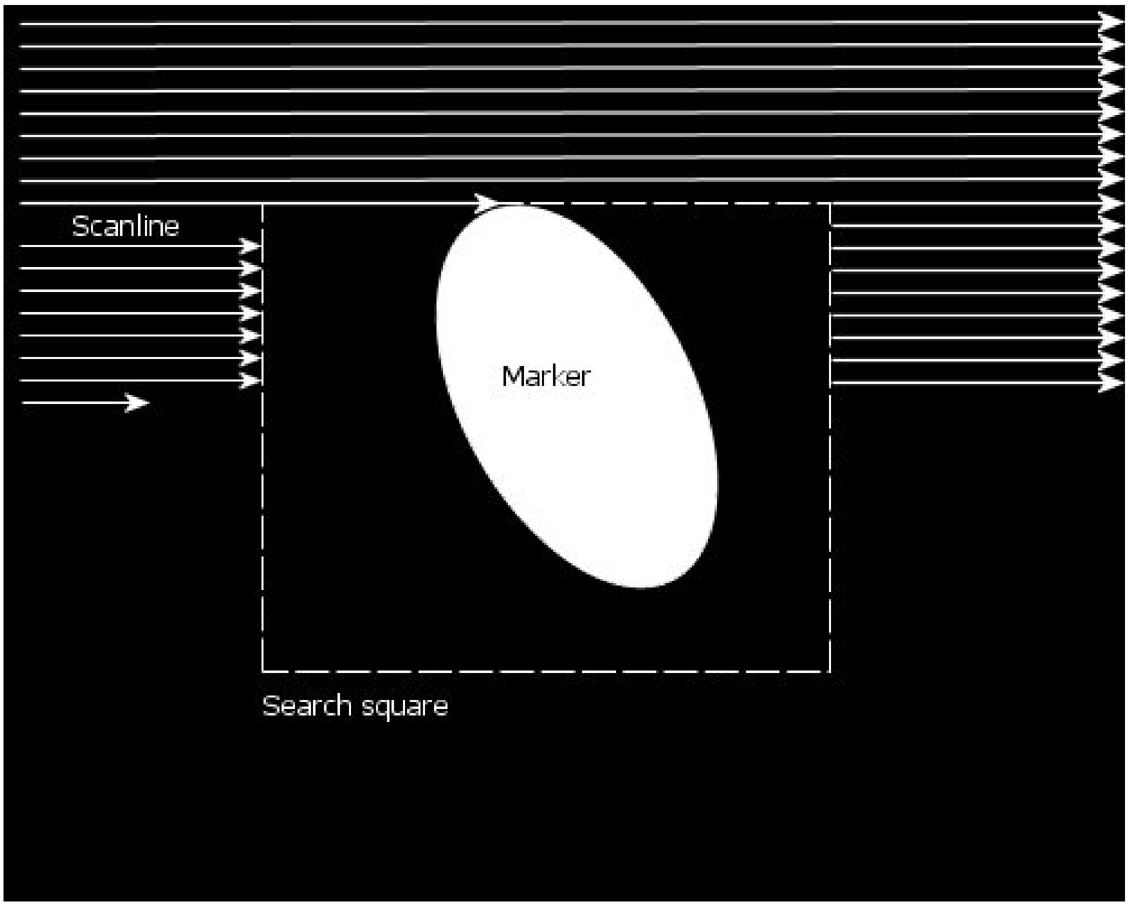


Figure 4.3 Marker scanning

```
Code:

For (all pixels in image){
     If(currentPixel above threshold){
          If cluster of pixels is found{
               Calculate midpoint of cluster;
               Add point to list of found points;
          }
     }
}
```

4.2.3 Initial position

When all markers have been found for the first time they must be analyzed and mapped to a skeleton-joint. This is done by using an algorithm that takes a marker's position and based on the surrounding markers determine what joint it represents. This requires that the actor to be in the initiation position, see figure 4.3. The initiation position is necessary for two reasons. One, we must obtain the full length of each limb, to be able to calculate the depth-rotations. Second, if the actor's position would have been arbitrary, the mapping of the marker and limb relationships must have been done manually.

This initiation is made at the first frame to be able to show a simple stick figure moving during the rest of the frames. This stick figure is a help to understand the markers in a frame as they otherwise may be hard to understand, and makes it easier to detect mistakes.

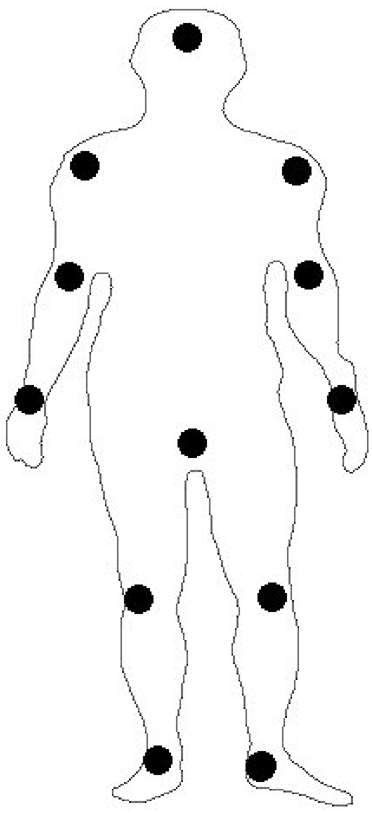


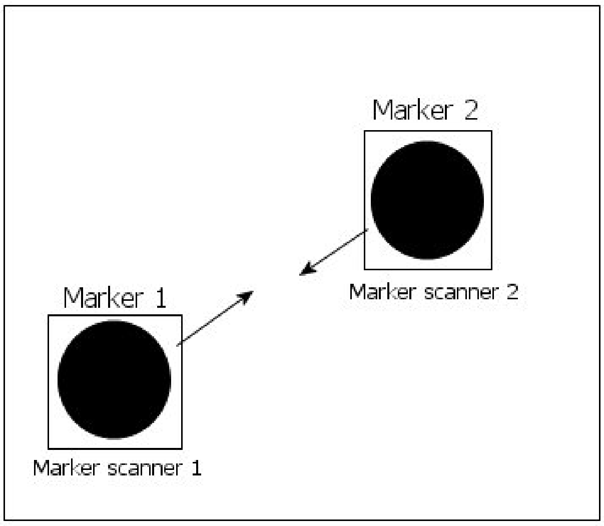
Figure 4.4 The initial position

4.2.4 Error correction

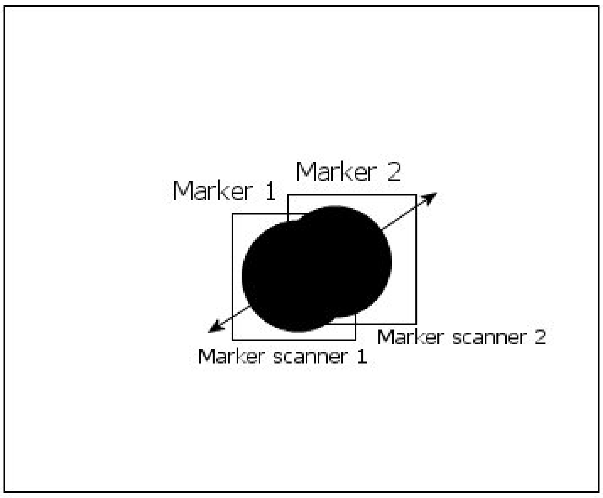
When tracking markers in a sequence of frames you need to consider these different failure scenarios. The marker is occluded by another marker, the marker disappears due to lighting problems and the marker moves to far between two frames

When the marker is occluded by another marker the tracking algorithm must keep the two overlapping points separated. During the frames the markers are overlapped you simply track the two points as one using the Scan for marker algorithm, but when one marker leaves the area you need to determine which marker has left the area. To solve this you use a combination of two algorithms. The first algorithm determines the color of the two markers that overlap. If these colors differ from each other you can easily determine which marker has left the area by simply checking what color the marker had before it entered the overlapped area.

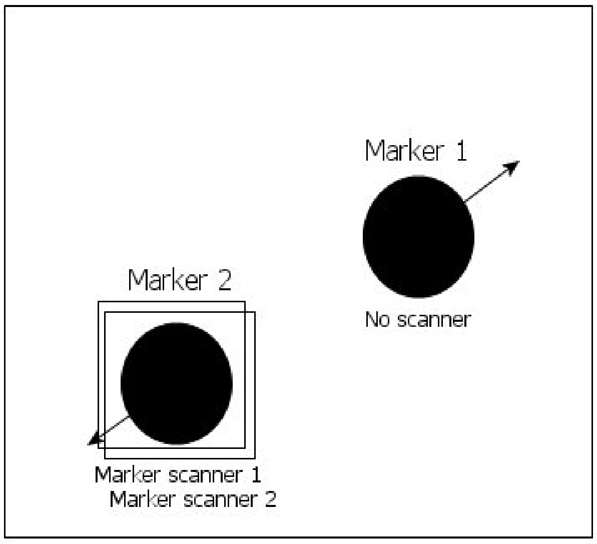
If the markers both have the same color you have to use different algorithms to determine the outcome. Markers with the same color on different joints will be present on our actor. This is because of the availability of only six different fluorescent colors that are distinct enough to separate. Two markers at different joints using the same color can produce an error if they are close to each other. When a marker is detected it is based on the distance from the last position and the color of the marker. If two white markers are close to each other the markers can either detect its correct next position or it can find the other markers position, see figure 4.4.



a) Two markers are moving towards each other, both have the same color.



b) The markers cross each others paths and the scan rectangles intersect.



c) The markers leave the joint area but the scan rectangles are now both following the second marker.

Figure 4.5 Marker occlusion

This will not produce an error until the two markers start moving in different directions then the marker detection algorithm might start tracking the wrong marker. From here there are two possible scenarios. The two different marker detectors follow the same marker. This scenario can be corrected using this algorithm:

1. Check if two marker detectors track the same marker.
2. Check that the marker tracked is moving.
3. If two detectors tracks the marker for more than n frames stop the tracking.

When the tracking is stopped the user must manually step back to the frame where the tracker lost the correct marker and place the tracker at the correct marker. Then the tracking is continued from that frame and the tracking will be correct.

The second scenario is a little more complicated. If two markers of the same color passes through each other the tracker does not know which marker is which when they leave the shared area. This can be solved using this algorithm:

1. Check the direction each marker had before entering
2. Use this direction to predict where each marker should be at the given time

When the marker disappears due to lighting problem the reason is often that the marker is tilted in a certain angle that doesn’t reflect the light properly and the marker becomes to dark. To handle this we let the Scan for markers algorithm (1.2.2) simply stay put at the position where the marker was last seen, and wait for the marker to appear again. When the marker again is viable the tracking will resume. Another problem that can appear when a marker disappears for a longer period of time is that the marker’s position may have changed drastically, and now will appear outside the range of the Scan for marker algorithm.

The marker can sometimes move large distances between two frames resulting in that the algorithm concludes that the marker has disappeared and stays in the same area waiting for the marker to appear again. The marker will continue to move outside of the trackers scan area and the marker is lost. This has to be handled manually and the program will stop if a marker has disappeared for more than 10 frames. From here the user has to step back to where the tracker lost the marker and the manually point the tracker to where the marker will be in the next frame. This is a rare scenario and will only occur in sequences with bad frame rates.

Below the pseudo-code for the error detection and correction algorithms are presented

```
If marker is not found within the scan rectangle {
     Set offset to 0;
     Goto next frame;
     If marker is not found in x frames{
          Output error and activate manual correction;
     }
}
```

```
If a marker with wrong color is found within scan rectangle
{
     Set offset to 0;
     Goto next frame;
     If marker is not found in x frames{
          Output error and activate manual correction;
     }
}
If two markers are found within the same scan rectangle {
     If colors are equal {
          Output an error and activate manual correction;
     }
     else {
          Save the marker with correct color to list;
     }
}
```

4.3 Creating motion capture data

4.3.1 Calculating the rotations

When all positions of the markers are found, the rotations of all limbs of the skeleton must be calculated. This is done by iteration trough the skeleton and for each limb using trigonometry formulas. The coordinates used in the formulas are always the limbs parent’s and its child’s. For example the lower leg’s parent is the knee, and the ankle is its child. Obviously, it is impossible from just one set of 2D-coodinates to know if the depth-rotation is positive or negative, that is if the limb is rotated towards or away from the camera. But the rotation is estimated for each limb and can later easily be changed; the advantage however, is that only one camera is needed. The Estimation of each joint is based on the human body’s DOF, degrees of freedom.

When all limb-rotations have been calculated, it is necessary for all limbs to subtract its parent’s rotation. This must be done because all limbs are at first calculated independently of their parents, but they must be in relative rotations.

```
Code:

For each limb
     Calculate its rotation based on the markers;
     Calculate its depth rotation based on its lenght
     compared to its initial length;

For each childnode
     Subtract parents rotation;
```

4.3.2 Creating key frames from motion capture data

It is not sufficient to save all the calculated rotations to control the skeleton, due to memory-reasons. Therefore this program will produce key frames, whenever it is suitable. Since the .x format use linear interpolation between the key frames, a key-frame will be created at the start and at the end of a constant acceleration.

All rotations and key-frames are presented in a graph when all calculations are finished. It is here possible for the user to modify any key or add/remove a key if suitable.

```
Code:

For all frames
     For all limbs rotations{
          If local maximum or minimum is found
          {
               Add frame as keyframe;
          }
     }
     Create a line between all adjacent keyframes.
     For all limbs rotations{

          If the differense between the current keyframe-
          line and the actual rotation > maxValue {
               Add new keyframe;
               Create new keyframe line;
               Start for the beginning of the frames;
          }
     }
```

4.4 Apply data to 3D-character

4.4.1 Converting angels to quaternions

Before it is possible to apply any rotations to the skeleton it is necessary to convert all rotations from common Euler angles to quaternions. [19] This is done for two reasons. First, to avoid the problem of Gimbal lock. [19] Second, quaternions are the accepted way of storing angels in most major animation-formats, including the .x file format, which is the result of this program.

4.4.2 Apply motion data to 3D-character

Motion data is applied to a 3D-character by modifying an existing skeleton. The skeleton can be exported from several professional 3D-packages like Alias Maya or Discreet 3DSMax. The skeleton must include the joints you want to modify and they must be exported using the jointnames defined for the markers in the application. When a movement has been recorded and the angles extracted the data is applied to the exported skeleton using parsing and writing as described in chapter 3.6. To modify the skeleton we add the key frames and rotations for each joint according to the motion data to the .x file. The resulting .x file can then be used with any package supporting the .x file format.

5 Testing

We have performed several tests of our motion capture solution. These test will be describe, both conditions and results, in this chapter.

For the testing of our solution we have created a prototype for the whole system. The differences between the test-prototype and the full-scale solution are the following: We are only using the upper body of the actor and therefore only need seven markers instead of twelve as marked in the initial position.

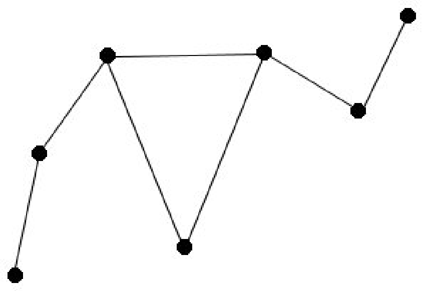


Figure 5.1 Upper body markers

We also removed all dynamic film improvement tools like the ability to change brightness, contrast and so on. We performed all testing using both real-life videos and test sequences created in Maya. The test sequences created in Maya were to look like a real-life sequence under optimal circumstances. The sequence was animated to look like real human movement and was then rendered. The test sequence from Maya produces the exact angels and those could be compared with the ones generated by the program. This gave us the possibility to calculate error margins.

5.1 Marker testing

We have experimented a lot with different markers and done several tests with each new setup. Our goal was to find a marker setup that could produce different colors for each joint. This posed to be a larger problem than expected. Our first approach was to use diodes emitting light in different colors, this worked in a complete dark room but we still couldn’t produce more than 4 colors that could easily be differed from each other. Here we also needed to be close to a power source that limited both range and mobility.

The second approach was to use colored carbon paper and cut out markers to place at the joints, this worked in daylight but the colors couldn’t always be separated from each other. We then extended the carbon paper method and used fluorescent carbon paper and cutout stripes that could be attached around arms and legs and then dimmed the entire room and placed a UV-light in front of the actor. This proved to be very successful and we could now separate the colors from each another. It would be better to increase the number of UV-lights so that the light reflects the markers from all angels of the actor. One UV light was sufficient for our purpose since it covers the most vital angles.

Using stripes also improved the problem of the markers disappearing at certain angles because the stripe was attached around the entire limb.

5.2 Occlusion

For this test we created two types of video sequences. One real-life video sequence where the markers occlude each other several times during the clip and one video sequence was created using Maya. The real-life sequence passed the test when the markers covering each other were of different colors but when two similar colors overlap the marker detection algorithm could not separate the two and we had to manually correct this error. The video sequence created using Maya passed the entire test, this was due to the fact that we could create all the markers in different colors, thereby the marker detection algorithm didn't loose track of any markers since it could distinguish the colors.

5.3 Markers velocity

The real-time sequence lost the markers when the markers move more than about 200 pixels in either direction. The computer generated video sequence also lost track of the marker when the velocity was more than 200 pixels in either direction. The sequence had to be manually corrected after loosing the markers but the data produce was correct.

5.4 Markers disappearing

When a marker disappeared the algorithm stopped and waited for the marker to come back, this happened two times during a 750 frames video sequence. In the real-life sequence the algorithm lost track of the marker one time due to that marker moved too far away when invisible. The other time the marker was lost due to that the algorithm found another marker with similar color when the real marker was invisible. The same scenario happened to the computer-generated sequence. The sequences could be corrected using the manual correction function and then the data was produced successfully.

5.5 Rotation accuracy

Our test sequences show that the angels in the 2D-plane facing the camera are calculated with between 2-15% margin for error compared to the real angels. It is not possible for us to get a better result with the equipment used because the markers need to be a certain size to be able to be located and the center of the marker may differ from the center of the real joint.

The depth-rotations are harder to get accurate. Our tests show that we calculate these angels with a 10-30% margin of error. This is because the actor is not showing 100% of the limbs length in the initial position. They also often have a slightly different center than the real joint. Besides the margin of error, the angle might be totally reversed. There is no way for us to know if the angel is towards or from the camera using only one camera. We can however estimate the angles for some limbs due to the human anatomy to get more than 50% accuracy. If the angle is reversed, it is easy in our solution to just reverse it back to get a correct result but this has to be done manually in the post-processing process.

5.6 Time measurements

The most time-consuming part of our solution is to extract a frame from the movie. This time increases linear to the number of frames. The step of finding the markers in the frames is a small part compared to the frame extraction and also increase linear to the number of markers.

When the algorithm has found all the markers and saved their positions in every frame we start the post-processing. This involves calculating the rotations between the key frames for every joint. This process is linear to the number of key frames and markers and therefore is time-consuming in large video sequences. The final step is to create the .x file, this is done in the post-processing and this function too is linear to the number of key frames and joints. To sum the time measurements the most time-consuming function still is the frame extraction and the other functions only occupy a fraction of the total processing time.

5.7 Summery of testing

This chapter contains the restrictions that has been made either by us, or have become apparent during the testing of our solution.

5.7.1 Number of cameras

We had some ideas about the number of cameras that would be suitable for our solution. Our first idea was to record not only from the front but also from one side. The purpose of the second (side) camera would have been to determine if the rotation of a limb was negative or positive, that is, moving forward or backward. However, we found that the body covered many of the markers too often and therefore the camera did not work as planed. One solution is to add a third camera, recording from opposite direction of the second one. But adding a third camera also adds very much to the complexity, and our goal all along has been to keep it as simple as possible so it can be used outside a studio, and without calibration that would have been necessary with three cameras.

We also considered triangulation, but since triangulation requires at least two cameras that can locate the marker, and many more are often used, this solution was discarded. Triangulation also requires careful calibration that our system does without.

5.7.2 One camera restriction

Due to the fact that this is a one-camera system, the actor will have some limitations of his movements. Movement that involve full rotations cannot be done because the body will cover many of the markers for a long time and during that time move them a significant distance. And therefore the markers would disappear never to be found again.

To solve this problem more cameras are needed, to record the actor from many angels at the same time. Also, if you consider any limb to be a vector, than the roll-rotation cannot be performed. To be able to calculate the roll-rotation, many more markers must be added and still it would not be very accurate with only one camera, and therefore it has no place in our solution.

6 Conclusions

In this thesis we wanted to create a version of a optical motion capture, that do not require high price, high performance cameras and hardware. We wanted to see if it would be possible to build a system with a cheap, easy accessible camera, and a standard home computer. We have found that an optical motion capture system is very complex and it is hard to produce as a low budget system. Our attempt did not live up to our expectations in terms of accuracy and usability. The idea with using only one DV camera was to simplify the system since it would require no calibration or synchronization. It would also make the system cost efficient and more available to the general public. The single camera solution unfortunately created a number of problems in our system.

6.1 Results

The different test performed on the optical motion capture system show that the system is usable but contains several flaws which makes it unsuitable for usage in professional environments. The major flaw is the occlusion factor. Since we are only using a single camera we cannot know where occluded or invisible markers are or what direction they are moving in. We therefore have to use the manual correction to be able to processes an entire movie clip. Another flaw is the 2-dimensional effect we get from using only one camera. We can only estimate in what direction a movement is performed when the movement is around the z-axis. This gives us a somewhat over 50% chance of making a correct estimation. However, if incorrect it could easily be corrected using the graph editor and therefore does not pose a big problem.

Here follows a discussion about the actor’s mobility. Presumed that the environment is optimal, it still is not possible for the actor to rotate because this will occlude several markers at the same time. To be able to rotate the actor the manual correction will become massive and fail the purpose with the motion capture system. Due to the camera restrictions the actor cannot move too fast or the marker will be lost and must be manually corrected. All these movement limitations of course limit the usage.

Our system does however work with less complex movements. It can produce motion capture data that is accurate enough to be used in low budget games. It is also very cheap compared to other systems on the market. The system has a very easy setup and does not need any calibration in addition to the init position.

The first step of the problem was to create a film sequence. We found that best way of creating the “marker suit” was to use fluorescent colors and a UV light. The important things about the markers are that they should not disappear unless in perfect angle. It is necessary when working with a single camera that markers that may often overlap are in different colors. The room should be dim and the camera placed on a tripod to get a steady picture with nothing to be seen but the markers. The sequence produced showed sufficient colors; we could find the markers and separate them from each other. The idea of using colored markers made it a lot easier to keep the markers separated. Without the fluorescent colors, the number of errors and need for manual correction would increase by a multitude. However, the use of only a single camera and the low frame rate resulted in the movement limitations described above.

The second step of the problem, to analyze the film sequence, was the most difficult and extensive. Here we had to find a way to separate the markers from each other and estimate their position as they move. We had to discard noise and conclude that it was the right marker, based on its color and estimated position. We found that our algorithm could track the markers accurately as long as the markers were clearly visible and separated. If they got occluded or moved too fast the algorithm was not able to continue the tracing process and there had to be a manual correction. However, if there were several markers occluded at the same time or a marker disappeared for a longer period of time, the algorithm was not able to continue and produce accurate data.

Step three was to transform the marker positions into usable motion capture data. We here created an algorithm that calculated how the actor's limbs were rotated in the 3D-space based on the marker positions. This was done using trigonometry. We also had to choose what frames to be stored in key frames to minimize the memory consumption without losing information about the movement. To calculate the angles and to create key frames worked out successfully. The problem in this step was mainly the depth rotations. Once again the occlusion and the 2-dimensional effect because of the single camera caused difficulties

The last step was to apply the motion capture data to a 3D-character. We needed to create a skeleton with the same joints as the markers on the actor. We also needed to convert all angels into quaternions because this is the angle format used in .x files and in many other motion capture data formats. When we applied the data to the 3D-character it mimicked the movements correctly and the skeleton joints matched the joints of the actor. The conversion from the angles to quaternions was successful as well.

6.2 Future development

If we were to build a complete motion capture system there are some things that we would want to improve.

- First and most important we would want a better camera, with a higher frame rate and a higher resolution. With the higher fram rate we could much easier follow the markers as they move, because their movement from one frame to the next would decrease. With a higher resolution the accuracy would increse since we would have a higher precision in the marker tracker algorithm.

- We also desire more then one camera so that we could track the markers in 3D-space with a higher level of correctness, the higher corectness would be achieved because the markers would always be visible from some of the cameras and the occlusion problem would be eliminated. Increasing the number of cameras would also increase the complexity of the system and require extensive calibration.

- Finally we would use markers with a wider range of colors so that every joint could have a marker in an individual color. This would make it easy to separate the markers from each other and thereby produce far less errors.

References

[1] Ruiz K, *Experimental Motion Capture Systems Development*,
< http://www.rpi.edu/~ruiz/research/research2/krmkmocap.htm>, 2004-05-11

[2] Menache A, *Understanding Motion Capture for Computer Animations and Video Games*, Academic Press, 2000

[3] Motion Analysis Corporation. Santa Rosa, California
< http://www.motionanalysis.com/>, 2004-04-23

[4] nCubic Corp,
<http://www.ncubic.com/home/motion/motioncapture_eng.jsp>, 2004-03-10

[5] Photron USA, Inc, San Diego,
<http://www.photron.com/cameras.cfm>, 2004-05-11

[6] Canon Inc. Europe,
<http: //www.canon.se> 2004-06-01

[7] Shannon Luminous Materials, Inc, *Technical - Black Light Ultraviolet Light*,
<http://www.blacklite.com/Technical/black_lights_ultraviolet_lights.htm>, 2004-05-11

[8] Synthgen, LLC, *Table of fluorecent Dye*,
<http://www.synthegen.com/products/fluorescent/table.lasso>, 2004-05-11

[9] XBDEV.NET, *DirectX 3D File Format*,
<http://www.xbdev.net/3dformats/x/xfileformat.php>, 2004-05-11

[10] Järvinen P, *On research Methods*, Opinpajan Kirja, Tampere,2001

[11] HowStuffWorks Inc, *How does a black light work?*,
<http://electronics.howstuffworks.com/question59.htm>, 2004-05-11

[12] Boumphrey F, *The RGB color wheel*,
<http://www.hypermedic.com/color/theory_rgbw.php>, 2004-05-11

[13] Fisher. B, Perkins. S, Walker. S and Wolfart. E, *Connected Components Labeling*,
<http://www.cee.hw.ac.uk/hipr/html/label.html>, 2004-05-11

[14] Watt, A, Policarpo. F, *3D- games, animation and advanced real-time rendering, vol 2*
Addison-Wesley, 2003

[15] Hill. F, Computer *Graphics using OpenGL 2nd Edition*, Prentice Hall, 2001

[16] Geraint Davies Consulting Ltd, *DirectShow Filters,*
<http://www.gdcl.co.uk/dshow_dev.htm>, 2004-05-11

[17] Axelsson M, *Extracting bitmaps from movies using DirectShow*,
<http://www.codeproject.com/audio/framegrabber.asp>, 2004-05-11

[18] Carneige Mellon Entertainment Technology Center, *Marker Placement Guide,*
<http://www.etc.cmu.edu/projects/mastermotion/Documents/markerPlacementGuide.doc
>, 2004-05-11

[19] Brown M, *Euler & Quats,*
<http://www.anticz.com/eularqua.htm>, 2004-06-23