Action recognition in motion-capture data

Thomas A. Grønneløv*

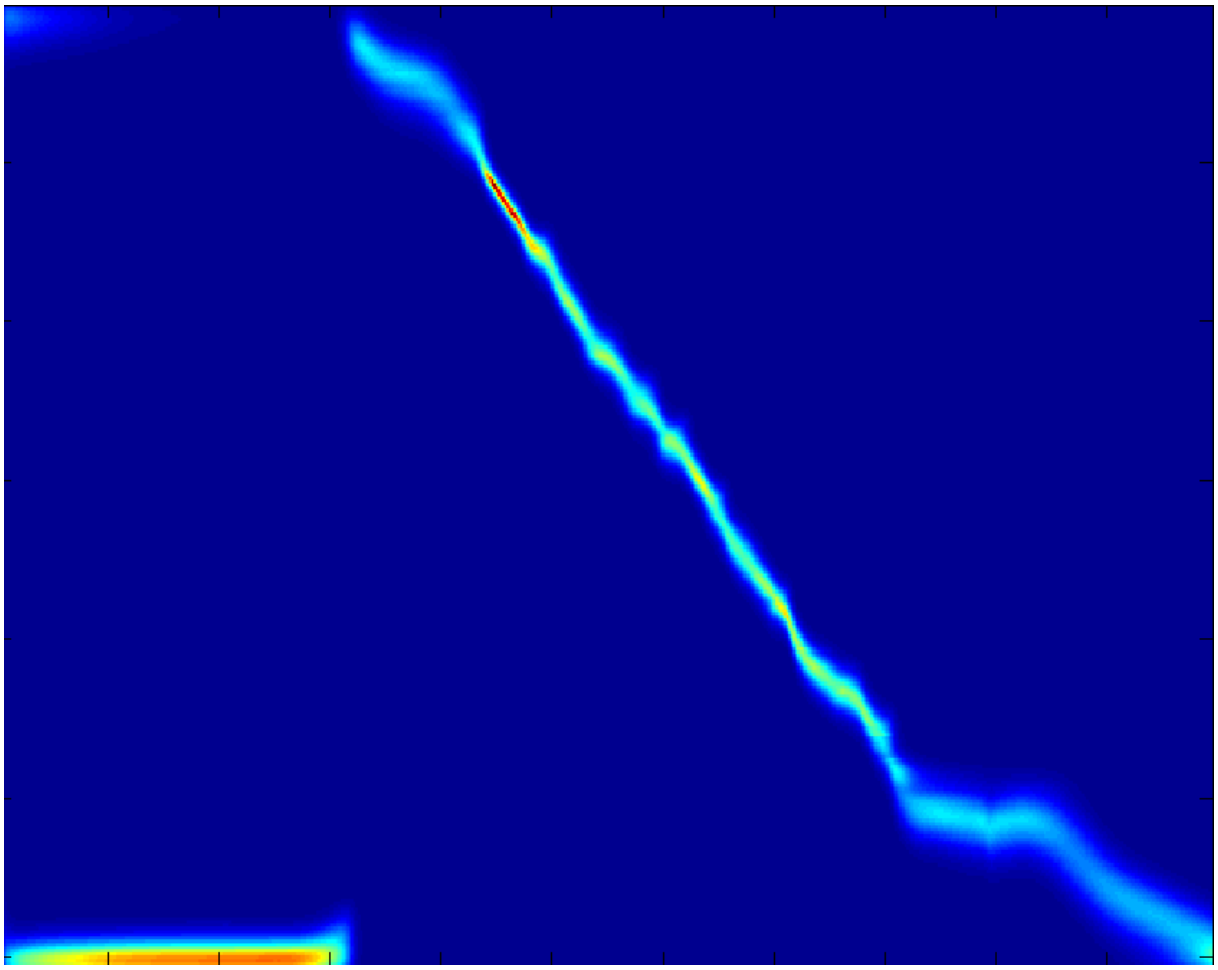


Figure 1: Probability matrix of frame matching

Abstract

Motion capture data is generally context less measurements of skeletal data taken at discrete time intervals. It is not linked with any understanding of what motions the data represent. In this project we aim to design an automated system which can compare new, previously unseen, motion capture data with existing known motion templates, and both classify the new data and compare it with the template. This will allow the system to act as a virtual trainer, guiding an individual to perform actions which best mimics the templates. The templates will in this situation be the target motion describing an exercise or technique.

We present a probabilistic approach to motion capture action recognition, comparison and guidance to best emulate a template and show how this can give good estimates about the intent of a given action.

*e-mail:Tag@Greenleaf.Dk

Contents

1 Introduction

In physical therapy we generally have an exercise which is aimed at mobilizing the joints, tendons and muscles through predefined motions and stretches in a predefined order[Wai et al. 2010]. This motion can be described verbally, through graphical illustrations or it can be demonstrated by a person experienced in the correct form of the exercises. Commonly the exercise is demonstrated and then the patient performs the same exercise while an expert observes and helps identify errors in how the exercise is performed and gives hints as to how the patient can correct these errors. If, however, an expert is not available, the patient receives no feedback and may end up performing the exercises in a less than optimal way. The exercise may even end up doing more harm than good.

In this project we investigate a method by which we compare a patients movements with previously recorded movements representing a correct exercise. Both movements will be in the form of Motion Capture data. This comparison will allow the system to give some feedback to the patient describing if the exercise is correct or if it needs to be adjusted. If adjustments are required, the system should give hints to the patient allowing him to correct the movements and perform the exercise in a more correct way.

Apart from serving as a guide to the correct form, it can be used as a form of training entertainment game in which the patient scores point when doing the exercise properly. For a stretching exercise the optimal stretch could be increased slowly from exercise to exercise to constantly change the target and keep the patient motivated in doing otherwise repetitive and boring exercises. This seems like a reasonable expectation considering the multitude of home fitness games available on the market today.

It should be pointed out that this system in no way aims to compete with an actual instructor that can observer in great detail while actually manipulating the patients joints to correct an intermediate pose. We aim only at improving on the current state of having no feedback what so ever when an instructor is not available.

The methods described in this paper will be usable to other forms of training as well, but with our primary focus on physical therapy, we place a greater emphasis on going through the exercise motions rather than arriving at some optimal end position. The path itself is the important part.

In this paper we will decouple the registration of motion from the analysis of that motion. While motion capture is briefly described it is not the subject of this text. This should be a general method which could be used with any pre-recorded motion in the correct format. The focus will be on comparing recorded motion capture data with previously recorded template data.

1.1 Data

Some of the figures represent a lot of data and therefore some details become quite small and hard to see. The figures are therefore presented in their original form in the archived version of this paper. The MOCAP data being used is available in the same archive, along with an electronic version of the paper itself.

`WWW.Greenleaf.Dk/TAG/projects/MOCAP/MOCAP.zip`

1.2 The reader

The reader is expected to have a basic understanding of probability theory and its notation as well as some simple statistics. Some linear algebra is used and should be understood at an intermediate level.

The relevant topics in probability theory is covered in texts such as [Russell and Norvig 2003] or [Bishop 2007]. The revelant linear algebra is coverd in texts such as [Messer].

1.3 Terminology

- End-effector, the end of a joint-bone chain in a skeleton. Usually the hands, feet and head.
- Actor, person being recorded for motion capture.
- Patient, person using the system to be guided in an exercise.
- Pose, joint rotation configuration of a skeleton.
- Frame, a single pose recording from a motion capture data file.
- Motion, a series of poses describing how a skeleton is moving.
- Action, a motion with a purpose.
- Template, a Motion Capture recoding of an action which is used to define ”the correct way”.
- Path, the best understanding of how the *intentions* in a new exercise matches the template over time.
- State velocity, how many states a path advances through a Markov model every exercise frame.
- Euler angle space, a $\Re^3$ vector space spanned by the three Euler angles a joint can rotate about. A point in this space represents an orientation.
- Spatial space, the usual 3D space that we all exist in and move about in.

2 Motion capture

Motion capture, hereafter termed MOCAP, in its broadest definition is the process of recording motion. Most common is recording the skeletal movement of humans and animals as well as the inanimate objects they interact with. The capture process measures the complex motion and transform it into a simpler lower dimensional dataset which contains the essence of the motion. This is commonly the 3D positions over time of certain points of interest on the body or object.

2.1 Past and present of Motion capture

In the following MOCAP is taken to mean the motion of a human body and not the broader meaning of capturing any motion. The person performing the recorded actions is called an actor. We briefly outline the past and present of motion capture in the following.

2.1.1 Very early history - 1876

What is probably the earliest example of MOCAP [Kitagawa and Windsor 2008] is from 1876 where Eadweard Muybridge was asked to prove or disprove that a galloping horse would sometimes leave the ground with all four feet simultaneously. At the time there existed only still image photography, so in order to create a high speed recording of a running horse, he placed a sequence of cameras along the path of the horse and has each camera rigged to go off as the horse went by. This resulted in a sequence of still images which turned out to show that the four feet did in fact all leave the ground at one time. Following his success with recording the running horse, he went on to use the technique on various other animals and publish a book with the images, which was quite new ting at the time[History].

The next development came when flash photography allowed a single still-image camera to emulate film recording[Precinemahistory]. With this technique, motion is recorded with one continuous exposure in a dark environment. At a set frequency, a flash would go off and illuminate the scene. The resulting image is a number of exposures overlaid on the same photography, but separated in time by

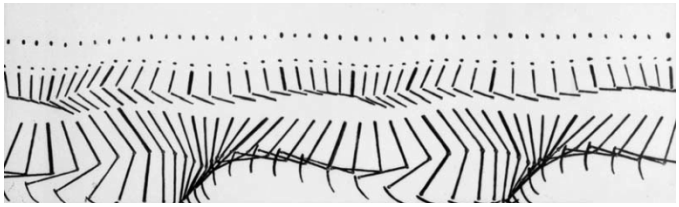


Figure 2: Flash photography as early as 1886. Shown here in negative colors. Notice how the shape of a walking human can be recognized.

the flash intervals. Figure 2 shows a flash recording from 1886 by Etienne-Jules Marey. The actor was dressed in a black suit with white stripes along arms and legs - quite similar to more modern versions of MOCAP, but simpler in that it does not require the camera to directly record movies. The single recorded image comprised of all the sample point images. The image with the overlaid exposures captures the essence of the motion, and you can easily observe, for example, the knee angle during a step, or, knowing the flash frequency, the velocity of the motion.

It was not until 1915 that MOCAP was used for generating graphics, and not for pure motion analysis. Rotoscoping was invented by Max Fleischer[Precinemahistory] and using this technique an ordinary film is recorded. This is then project to a drawing surface one image at a time. Artists can now easily draw animations on top of the previously recorded motion and when done with one image the film advances to the next one. This makes it easy to match the motion and generate animations which look more realistic than pure drawing. It was used in a number of smaller movies by Fleischer Studios before being used in the feature length film Show White by the Disney Studios in 1937[Kitagawa and Windsor 2008].

For a comprehensive review of the history of MOCAP we refer to [Kitagawa and Windsor 2008].

2.1.2 Recent history - 1980

In 1985 a more contemporary form of MOCAP was seen in the commercial ”Brilliance” which featured a computer generated human like robot that moved like a human being. The MOCAP data used in the animation was generated by painting bright dots on a human actor at 18 joints and recording this from a number of different views. Not by having a large number of cameras, but by rotating the actor on a turntable[Kitagawa and Windsor 2008].

The more modern advances in MOCAP were driven by the needs of the military and medical researchers and their different needs have created a number of very different methods of recording human motion. We will briefly mention a few of the more noteworthy ones, most of which are still in use today in a revised form.

One of the most direct forms of measurement consists of an exoskeleton which is strapped onto the actor and directly measure joint rotations[Menache 1999]. This could be as simple as a potentiometer connected to the suit joint which followed along when the actor moved. The suit has the benefit that it give fairly accurate measurements in a very easy to use format. It also does not suffer from the occlusion problems which affect vision based MOCAP, but at the same time the suit is clumsy and it affects the very motion that it should record. Further more, the actors global position and orientation is not measured directly but can only, to a certain extent, be inferred through joint movements over time. Some systems do have accelerometers and gyroscopes, but they also suffer from accumulation of errors.

Another from of active MOCAP is based on electromagnetic sensors which are placed on the joints of the actor[Kitagawa and Windsor 2008]. The motion is recorded inside an electromagnetic field which lets the sensors measure both position and orientation within the field. While this method is less invasive, the actor does still have to be ”wired up” with electrical sensors on the joints and wires running up and down the body. The size of the field also limits the the motion, and the field is sensible to outside influences such as metal objects.

Both the exoskeleton and the magnetic method has

the strength that the data is generally clean and ready to be used. This allows for real-time uses such as remotely controlling a robotic arm and hand, or detecting the orientation of a fighter pilots helmet to directly change the visuals on the helmet mounted display in response to where he looks.

A variation of the electromagnetic sensors is acoustic[Menache 1999]. Here the actor has sound-pulse emitters placed on the joints and microphones are placed surrounding the scene. Based on the change in frequency, velocity can be calculated and based on the delay of the arrival time of the pulses at the various microphones, the position can be calculated. This system is generally not very accurate and it is easily affected by outside noise.

2.1.3 Contemporary methods

Both the electromagnetic and the acoustic methods require the actor to perform in a very limited scene where either the electromagnetic field or the microphones are placed. This scene is further restricted by requiring no metal objects influencing the field or disturbing sounds. To remedy these problems, accelerometers and gyroscopes have also been used for recording. Here the actor joints are fitted with small sensors detecting acceleration and orientation and based on these measurements, the joint motion over time can be calculated. While this method can be used in any environment, it suffers from an accumulation of measurement errors since all motion is relative to previous motion.

The currently most common method[Moeslund et al. 2006] consists of passive reflective markers placed on the actor. The motion is then simultaneously recorded from a large number of viewpoints by high-speed cameras. Based on the location of the markers in the different views, their 3D position can be calculated using triangulation and optimization methods such as Linear Least Squares[Jens Michael Carstensen]. This method is not very invasive since the suit with markers does not limit motion, but it is susceptible to occlusion problems where an object, or the actor himself, blocks view to one or more markers from a number of viewpoints. This can to some extent be remedied by the fact that the markers themselves are inexpensive, so a larger number can be used than is the case with active markers. There is such a thing as too many markers though. As the number of markers grow, the chance of ambiguity between two or more markers grow. Another problem is that only position is recorded. Joint and bone rotations need to be inferred based on markers located on top of flesh, skin and clothes, requiring even more markers.

While the requirement that the actor should wear a special suit with reflective markers does not influence the motion itself, it is sometimes still to ask too much. If you want to analyze surveillance videos of some action, possibly a crime, you can not expect the actor to wear any special clothing. Also if the MOCAP is part of a home entertainment system or if the action to be recorded is at an actual sporting event, then it is generally a severe limitation to require special clothing. An active field of research is therefore the Marker less MOCAP[Hauberg et al.], which is normally vision-based. Here there are no specific requirements apart from what an ordinary human observer would have: the actor should be visible and not move faster than what the eye/camera can detect. If the entire body is clothed in similar colors, then ambiguities may arise. This is true for an automated system as well as for a human observer.

2.1.4 Visual marker based MOCAP

While our focus is not on acquiring motion capture data, but on using it, we need to be aware of the various ways in which our data may have become actually corrupt. We also need to understand the limitations in accuracy since no method will generate 100% accurate measurements in all situations. Since most current MOCAP data is created using the vision based method of reflective markers, this is the method we look at in the following.

2.2 Data lifecycle

Marker based MOCAP requires a number of cameras surrounding the actor in order to maintain an unblocked line of sight to all markers from two or more cameras. Additionally, the accuracy of the triangulation increases with the number of cameras observing the same marker. This presents a slight problem through, in that the cameras need to be calibrated. The MOCAP system needs to know where each camera is relative to the other cameras with a high degree of accuracy, and the system needs to know the internal projective characteristics[Jens Michael Carstensen] of the cameras. This is generally done once and for all when setting up the equipment in a studio, but if it is not done perfectly, it may become a source of measurement errors. Another source of errors is the fact that even with a large number of cameras, a marker may become hidden from all views and it will then not be tracked before it again becomes visible to at least two cameras.

The markers are generally placed on a skin tight suit, but for obvious reasons the markers cannot be placed inside the joints. Instead they are placed close to the joint and the actual joint location is often [Kitagawa and Windsor 2008] calculated as the center point of two markers on either side of the joint. The markers may be slightly offset and each may be triangulated slightly wrong. This represents another source of possible error.

When the MOCAP recording is completed, the result is a sequence of 3D point clouds, each representing the spatial positions of the markers at discrete points in time - the frames. One marker may represent the side of the left knee and another the right elbow, but it is a human person who is responsible for connecting the marker points to the skeletal understanding of the points. This is another process which may induce errors. Perhaps not in the initial bonding, the so called rigging, but then later when ambiguities arise in the marker tracking and the system becomes confused about which point in the current frame was what point in the previous frame. When such situations arise, the human operator needs to sort out the confusion.

After recording, rigging and data clearing, the resulting motion captured skeleton is available, but it may contain some errors.

An entirely automated system will have even greater potential for generating incorrect data and this is something that should be considered in any system analyzing this data.

2.3 Motion capture data format

We will work with data in the BioVision Hierarchy (BVH) format[Biovision]. This format is based around an initially defined skeleton with joints and bones followed by a sequence of data describing the rotation over time of each individual joint as well as possible translation of the joints. An overview of the format will be presented here since it is the foundation of later pose comparison.

2.3.1 Skeletal information

A skeleton is based around a root joint which is often chosen to be the center of the pelvis. At this root joint other joints are connected through implicit bones in a parent child relationship. The bones are implicit since only the child joints position offset from the parent joint is given. Each child can in turn have zero, one or multiple child joints of its own. This way the entire skeleton is described in hierarchical fashion from the root joint out to the last child joint. The end of the joint chain is marked by an end-effector which generally represents the toes the fingers and the head.

The BVH description of the skeleton used in this project is shown in section C and a rendered version of the skeleton in its initial pose is shown in Figure 3.

In a skeleton, every joint has its own location in spatial space as well as its own orientation. Both are described in the parent joint's coordinate system. Initially all joints have zero rotation, and their

coordinate axis coincide with the axis of the world. Since a child is described in the parent joints coordinate system, when a joint is rotated, every child joint will be seen to rotate around the parent in the world frame of reference, and the child's orientation will change. In the parent joint's frame of reference, the child joint is ofcourse not changing. As an example, rotating in the shoulder joint will move and rotate the elbow joint in the world. This will in turn causes the wrist and hand to move, as seen in Figure 4. In the figure, the red lines represent the y-axis for the joints. The left-side arm is not rotated and the y-axis still coincides with the world y-axis, while the right-side arm has been rotated around the z-axis (into the image) in the shoulder joint. This has caused the shoulders coordinate system to change in the world and its children have moved accordingly - in the world.

This means that each joint is treated individually and that for any given joint we can look at *its* rotations without having to consider the rotation of other joints. It will remain fixed in the frame of reference of its parent - in which it is described. This is very much the same way we naturally look at a human body. If only the shoulder joint rotates, then the entire arm rotates but there is no actual joint rotation in the elbow or wrist.

To look at the details of the format we see the first "joint" being defined, the root, is called "hips". Actually it is none of the hip joints, but rather the center of the pelvis in this skeleton. The offset is the location of the joint in the world frame. The channel information states that this joint over time is described by 6 scalars which follow in a later section. They are translation x,y,z followed by rotation described in Euler angles over the z,y and x axis. The root is generally the only "joint" which translates over time and this is to describe actual movement through spatial space. Other joints will usually only rotate and not translate.

```
HIERARCHY
ROOT Hips
{
OFFSET 0.00000 0.00000 0.00000
```

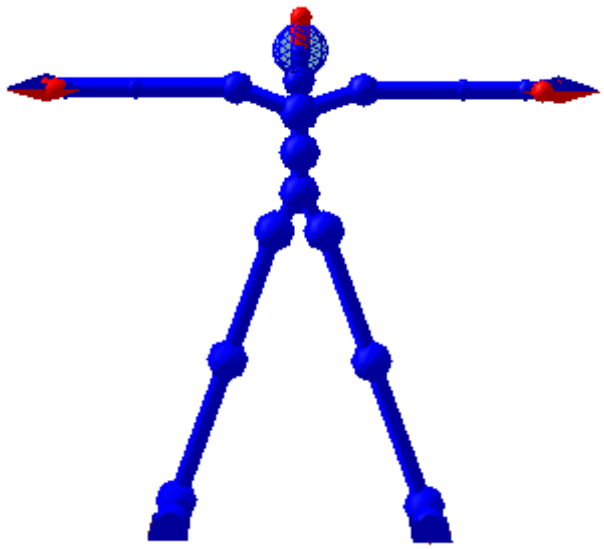


Figure 3: Initial pose skeleton with the hip as root joint and colored orange. Observe how the limbs are separated to make it easier to recognize the individual joints in a MOCAP recordings first frames.

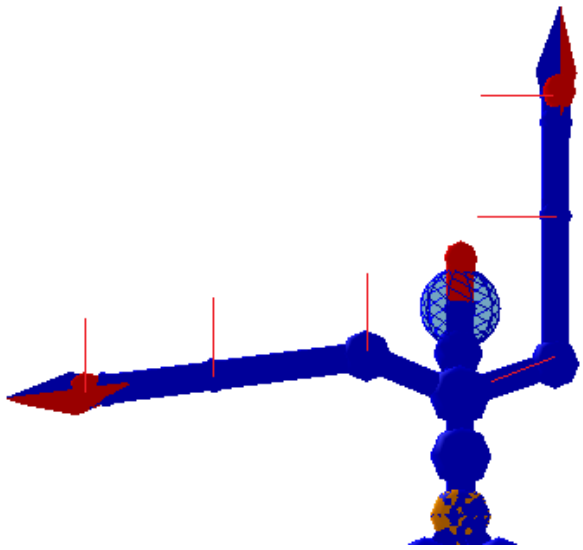


Figure 4: Skeleton rotating its shoulder joint and thereby moving the entire arm. It is seen that the child joints coordinate systems rotate with the parent.

```
CHANNELS 6 Xposition Yposition Zposition
Zrotation Yrotation Xrotation
```

Following this root joint we see LHipJoint being defined. Is has an offset of [0,0,0] which means that it is located exactly on top of its parent, the root. This may initially seem strange, but actually this is not what we would consider the hip joint, rather it is still part of the rigid pelvis and its purpose is to make a coherent skeleton which has one root and bones going out from it. Its rotation over time is defined by 3 Euler angles z,y and x. Actually this is part of the pelvis and therefore remains fixed over all motion capture samples.

```
JOINT LHipJoint
{
 OFFSET 0 0 0
 CHANNELS 3 Zrotation Yrotation Xrotation
```

The first rotating joint outside the root is LeftUpLeg which is essentially the left hip joint and the thigh bone connected to it. It is offset [1.28,-1.83,1.04] from its parent, the part of the pelvis coinciding with the root. Its rotation over time is also described by three Euler angles.

```
  JOINT LeftUpLeg
  {
    OFFSET 1.28858 -1.83292 1.03970
CHANNELS 3 Zrotation Yrotation Xrotation
```

From here on, the skeleton in section C continues on in a hierarchical fashion from parent to child in the same way as just described.

2.3.2 Motion

Following the skeleton definition is a section of motion data. Every joint has stated how many channels it uses and which parameter (translation or rotation) they control. Every frame of motion consists of a sequence of numbers located on the same line in the data file. The first number in the first line defines the first channel of the skeleton for the first frame. The second number defines the second channel and so on. For our current skeleton the following data line would mean that the x,y,z position of the root in the first frame was approximately [10.6,18,-30.4] and that it should be rotated [-20.2, -85.3, 23.2] degrees around the z, the y and the x-axis.

The data is 10.5735 17.9984 -30.3778 -20.1666 -85.3060 23.2228 .....

What is worth noting is that the skeletons position in spatial space is defined by the translation of the root and that each individual joint's rotation is defined by its own three Euler angles. It should further be noted that this skeletal description does not have any constraints. Any joint can rotate in any direction - as far as the skeleton is concerned.

Text book joint rotations The rotation of the lower leg in the knee joint is measured relative to its parent, the thigh bone. While the three Euler angels may initially seem unintuitive, this rotation relative to a parent is a common way of measuring joint flexibility. As an example the flexibility table in [Kurz 1994] lists the knee flexion as 130 degrees, its extension as 15 degrees and its internal rotation as 10 degrees. All rotations are relative to the thigh bone with the lower leg initially aligned with the thigh bone. The only thing complicating matters is that the thigh bone does not itself coincide with its coordinate system. The thighbone is not aligned with for example the y-axis. This is, however, a simple matter of transforming its coordinate system into the more intuitive bone-aligned system. This is illustrated in Figure 5 where the right side axis are the ones we have and the left side axis are the ones we would find more natural.

3 Physical therapy

For physical therapy the relevant measures of an exercise is primarily the rotations of joints, either individual joints or more commonly a combination of rotations. The exercise aims to mechanically mobilizing the joint through motion, strengthen the muscles through work and stimulate the nerves

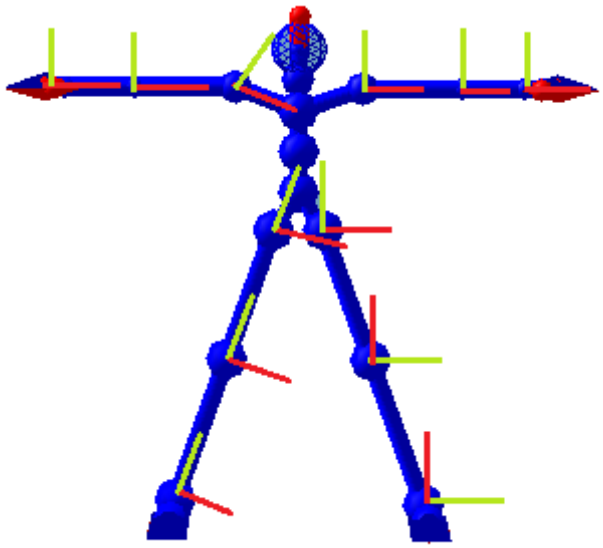


Figure 5: X and Y axis for the main limb joints. Observe how the joint coordinate systems do generally not coincide with the natural rotation axis.

through signaling[Wai et al. 2010]. An example of a very simple exercise could be training the knee joint by sitting on a chair with upper leg horizontal and then slowly raising the foot by stretching in the knee. Weights could be added to the ankles to put more emphasis on the muscles as well. The measure of correctness depends only on the rotation of the knee. If the range of motion is large enough and if the angular acceleration is slow enough to not show an overstretch of the knee by jerking to a halt at the end of the movement, then the exercise is performed well. The measure of correctness can therefore be calculated from only one joint's angular position and angular acceleration. As an example of a more complex exercise we take the split in which the initial position is standing and the feet then slowly slides to the side until the desired stretch is reached. The rotation of both femur[1] parts of the hip joints relative to the pelvis is the relevant elements in this exercise.

Joints relevant for comparison we will not include a full skeleton comparison in anything but very complex and very specific actions. The actions requiring full skeleton comparison could be gymnastic exercises which need to match a full body template.

[1]Thigh bone

3.1 Types of feedback

An instructor can either observe the entire exercise once or twice and then give general feedback based on all he has learned[Wai et al. 2010], or the feedback can be real-time so it is given as soon as a discrepancy between exercise and template is seen. The first method will be more correct since interpretation of what is done can be based on all information, while the real-time feedback looks only at the past and present.

We will focus primarily on instant feedback since this seems to be the more usable of the two methods. The proposed system can, however, easily be adopted to give feedback based on the full observation of the exercise.

4 Template acquisition and test data

We have currently not considered what should be recorded and with whom. In this section we will look at what a proper template should look like and how it could be generated.

There are a few problems with acquiring a recording which properly describes a correct exercise for a patient. If a trainer demonstrates the exercise while being recorded, this should result in a perfect template for later action comparison. If, however, the trainer is a tall thin man and the patient using the system is a short overweight woman then the joints will obviously not end up in the same relative positions in spatial space. For this reason we will drop joint position comparisons.

It should be noted that there are situations when joint rotations are not descriptive enough. If an

exercise consists of hands or feet touching each other, then it is in fact the end-effectors relative position which are relevant and not so much if the elbow or knee is bent to this or that extent. Such exercises do exist, but in this project we have chosen to focus on the more common exercises which deal with moving certain joints through certain motions in a certain order.

Another problem, which is complex to handle, is that of differently built persons performing the same exercise. A jumping action performed by a heavy and by a light person will not result in the same bend in the knees when landing. The heavier person will tend to absorb the impact over a longer period of time, resulting in more bent knees. There exist methods for "motion retargeting"[Hecker et al. 2008] which attempt to change the MOCAP data so it appears that the recorded person is heavier, taller or otherwise differently built. We note the existence of such methods for further work, but we will not look into it in this project. Instead we will assume that the difference in build is either not very large or is not affecting the motion in a too dramatic way. For non-ballistic exercises the latter assumption will generally always hold.

Finally a trainer may be more flexible than the patient. If a flexible trainer demonstrates an exercise which will give him a stretch, then we cannot assume that the inflexible patient will duplicate the joint rotations of the trainer. It would in fact be more likely to cause damage if he over stretches to match the template. In such a situation we consider two options. One is to scale the joint rotations so a rotation of 90 degrees could be relaxed to for example 70 degrees while the other is more robust and general. We could simply make a recording of the patient performing the exercise, to the best of his abilities, while guided by an instructor. When the instructor judges that the exercise has been performed correctly, the patient can use this as a guideline for exercises performed without the instructor being present.

In the rest of this paper we will assume that we already have the template and need to compare a new recording with it. We will further assume that the template shows the patient , under instruction, doing the exercise correctly. How this is done in practice and how the problem changes without those assumptions, that is left as future work.

4.1 The test dataset and its limitations for testing

This project is only a small part of a larger research project dealing with automated physical therapy guiding. Another part deals with performing non-intrusive marker-less motion capture. The marker-less MOCAP is not yet a completed project, and for that reason it could not deliver data for testing. This means that the data used in this project has come from other sources. Sources which do not deal with physical therapy exercises. This is sources which have created the data in advance so that we cannot make accurate testing of our record, analyze,feedback methods.

We have selected a small number of relevant samples of human motion types which are available in more than one version. As an example we have four recordings of different, but similar, persons turning cartwheels, we have a set of yoga stretches as well as a martial arts jumping kick, ordinary straight walking and a gymnastic tumble ending in a partial split position[2]. We find this set of data to be sufficient to perform an initial validation of our methods of action recognition and comparison, but recognize that the testing could have been more thorough if we had had easier access to data of our own choosing.

5 Action recognition

Much work has gone into the subject of human action recognition. It is very useful to have automated systems recognize human actions. In au-

[2]The motions can be downloaded as shown in section 1.1

tomated surveillance applications, it will let the system realize when certain interesting, illegal, actions are taking place and alert a human operator. In human-computer interaction it lets the human communicate using gestures. The basis for the recognition has commonly been raw video from one or from several angles. Not nearly as much work has been done in the field of action recognition based on motion capture data, though the field of motion analysis have used MOCAP to a larger extent.

Action recognition has previously tended to be based on 2D imaging where the unclassified recordings are matched against 2D templates, but with no attention to true 3D body motion. The purpose has been to recognize certain actions from monocular camera recordings and not to analyze or recognize detailed skeletal motion. While some [Weinland et al. 2007] have worked with 3D action recognition the above mentioned 2D analysis is by far the most common. For a review of past action recognition we refer the reader to [Aggarwal and Cai 1997] and for more recent developments [Wang 2003].

In order to recognize an action we must first establish a way to compare two poses. Given a measure of similarity for two individual poses, this must be extended to compare two motions which are each a sequence of poses.

We deem two motions to be similar if the joint rotations and relative accelerations follow in the same order and with the same magnitude. If the absolute speed at which the joints rotate are different, then the motion is still similar - only faster. MOCAP data is frame-based with each frame containing information about joint orientations at one point in time, but by looking at neighboring frames, the velocity and acceleration can be estimated through finite difference.

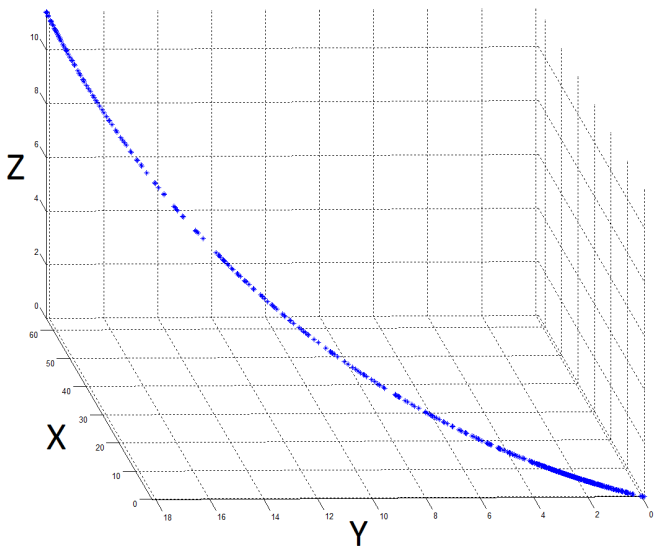


Figure 6: Euler angles for knee during walk. It is seen that motion is strictly along a a single curved path in Euler angle space. The path has a clearly dominant direction (rotation axis)

5.1 Initial motion analysis

Having access to a BVH motion capture file, we logically have the information required to recognize basic motions since the entire skeletal motion is, in fact, defined. Animating a stick figure based on the skeleton and motion would also allow us to easily identify most simple motions. The question is how we best analyze the actual motion in an automated way.

If we look at a single joint, the left knee, during a walking motion, we see Figure 6 that the rotation is not confined to a single Euler angle even though the knee is generally restricted to a single degree of freedom. It cannot rotate to the sides and it cannot twist much either. Actually the knee does primarily rotate around a single axis, but this axis does not coincide with either of the world axis which is clearly see in Figure 5.

If we look closer at the rotation angles and calculate the principal components of the rotations over time, we see that the three eigenvalues of the covariance matrix of the rotation data are 0.046, 0.6335 and 367.7888. The largest is almost 600

times larger than the second largest. In other words, it is reasonable to say that the knee only has one rotational axis which is given by the largest eigenvector [0.9465, 0.2812, 0.1585]. This rotation axis is relative to the world axis as well as to the parent. This is true because in the initial skeletal configuration all joints align with the world axis and the parent child rotational relationship was defined in the world frame where all axis coincided.

We therefore, in the MOCAP data, have three angles describing the knee rotation, though one angle could do it. It would seem reasonable to reproject the thee Euler angles onto the principal axis of rotation so we could have a single scalar describing the rotation.

If we, however, look at other joints such as the hip Figure 6, we find the eigenvalues to be 5.2411, 12.4070 and 174.5390 which, as we could expect, indicate three degrees of rotational freedom, though one axis is the more dominant. Looking at primary eigenvector we see that is [-0.9206, -0.1760, 0.3486] which is close to the global x-axis. Again this could be expected from observing Figure 3 and noting that the global x-axis is pointing to the right of the image, and knowing that they are from a walking motion.

The conclusion is that while we could project the rotation in the knee joint, and the elbow as well, onto a single axis of rotation and make the comparison of two motions easier, most joints will still be more complex to describe and thus compare.

5.1.1 Comparing actual joints

Taking the simple example of comparing the left knee of two different motion captured walks, we begin to see the complexity. We have reprojected the three Euler angles of the knees onto the principal rotation angle and plotted the resulting sequence of rotations over time for both knees. This is seen in Figure 8. It is easily seen that both motions go through the same rotation of the knee with approximately 60 degrees between maximal and minimal flexion. We also note that the period is

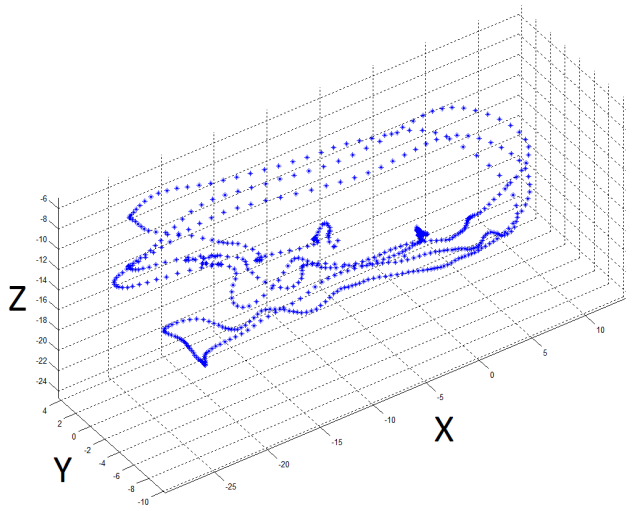


Figure 7: Euler angles for hip during walk. We see that the joint has a large degree of freedom and will move throughout an entire 3D volume in Euler angle space.

143 frames and that one motion starts ahead of the other. If all that the exercise required was for the knee to go through a certain rotation, then we could easily say if the motion matched the exercise. This is, however, something the patient could see just as well, and exercises are generally more complex than that.

We could perhaps, based on only the left knee, and using a visual inspection, conclude that the motions are similar, but we would have to consider how temporal translation influences similarity as well as temporal scaling. What if one motion starts 1s before the other but is only half the speed of the other? If we expand on the comparison and consider not only one knee but rather two hips and two knees for each motion, totaling eight joints compared two and two, we would have a better description of a walking motion. This is shown in Figure 9 where two walking motions each have both knees and both hips plotted in Euler angle space. One motion is drawn using dots and the other using solid lines. It is evident that the motions are similar. The knees move along the same Euler angle space path since the knee only has one degree of freedom and the hips move in the same neigh-

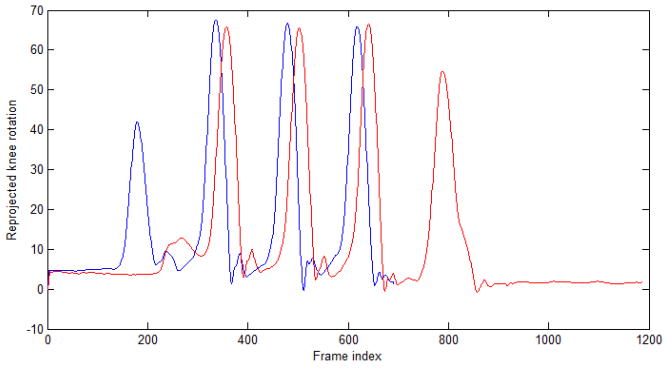


Figure 8: Two knees have been reprojected to their principal axis of rotation and comparison is now much simpler

borhood. What is not obvious from this plot is the timing of the motions.

We cannot easily see if one motion is faster than the other or if one pauses and starts up again, or perhaps the hip joints match at times when the knees do not and vice verse. Additionally we only compare rotation over time here and *not velocity and acceleration*. You could argue that by plotting the discrete angles over time we implicitly have velocity and acceleration information, or that we could perform a finite difference and obtain those values that way and plot them separately, but in our view this would only complicate matters further. There exist methods to compare paths in 3D, but is not a trivial task, and as joints are added the complexity quickly grows. Methods such as manifold learning are described in further detail in [Izenman 2008].

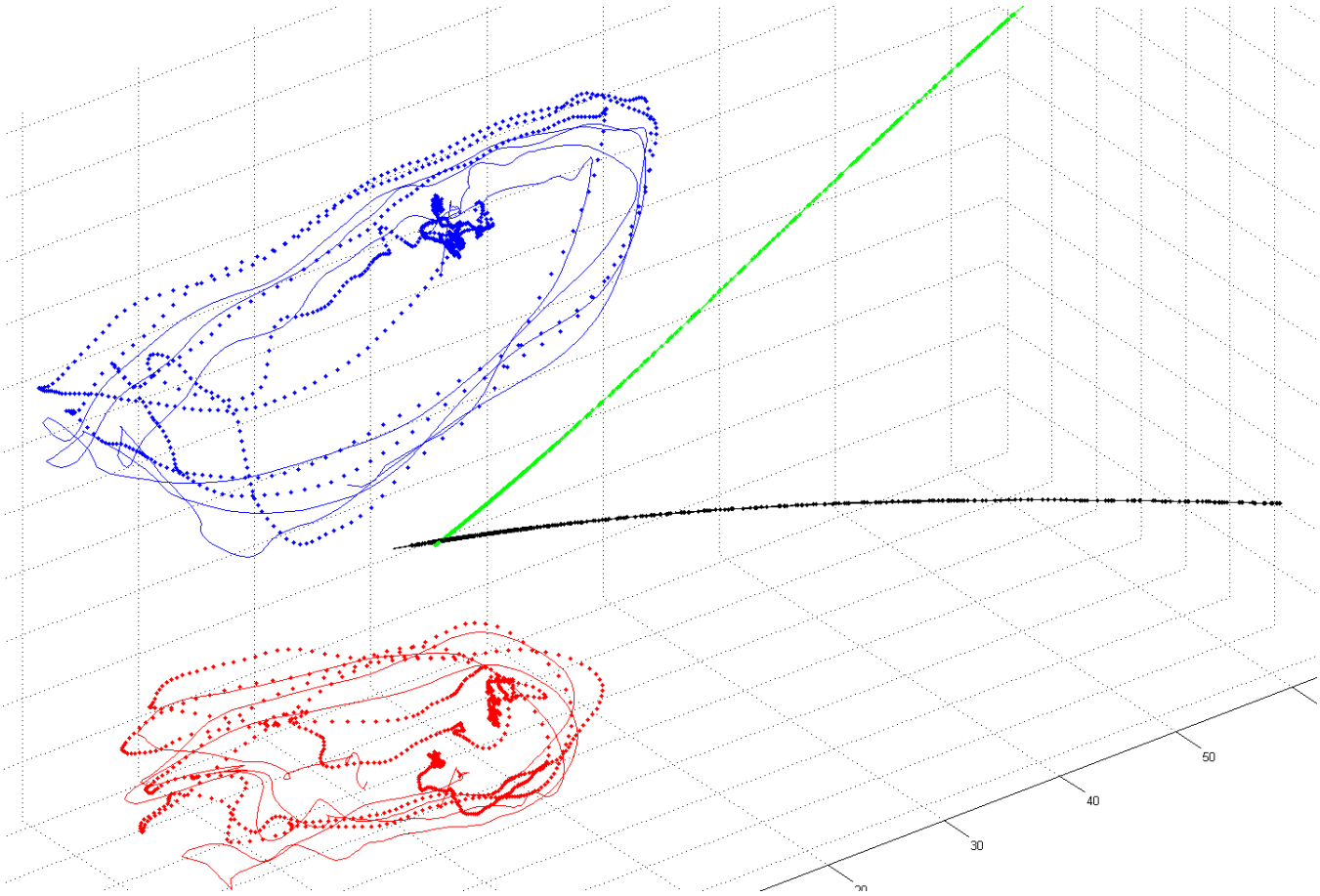


Figure 9: Euler angle space plot of two motions. Left and right knees and hips for each motion are shown together. One motion is drawn with dots and the other with solid lines. Observe how the left and right sides are offset by a simple translation. Also notice how the knees have a perfect match between dots and lines, the two actions, while the hips do not, due to their larger degree of freedom. Note that the right and left knees rotate around slightly different axis due to how they are offset in the skeleton.

5.2 Motion matching by paths

Having realized that the joint rotations over time do in fact convey very useful information for action recognition we will attempt to develop a method for actually carrying out a joint rotation based comparison. This will be based around first recognizing exactly what part of a template a given sequence of newly recorded MOCAP data is meant to emulate, and secondly comparing the attempted motion with the correct motion in the template.

The primary use of motion capture is in the field of animation and here there exist a large knowledge base around MOCAP here. A common problem in animation is a character that is animated using one MOCAP dataset and then has to be animated using another dataset. The initial animation might be a walking motion and then the character should change to use a new set of MOCAP data describing a running motion. The naive solution is to simply switch to the running motion when the character runs and otherwise use the walk. This would work, and has been used extensively in computer games previously [Ménardais et al. 2004], but it gives a discontinuity in the motion when the change is carried out, and this does not look natural.

There exist a method called ”motion blending” [Kovar and Gleicher 2003] which is aimed at solving this problem. Rather than changing from one animation to another over a single frame, the change is smoothed out over several frames, thus giving the illusion of a single animation starting with walking and then transitioning into a run. The question is now which frames to blend. If we look at all the frames in the walk and all the frames in the run it is obvious that the least abrupt blend will be between frames that are alike. It is also obvious that if we blend a pair of frames then next we should blend another pair which are close to the first pair and which are forward in time. If they are not close then we would again get a discontinuity and if they are not always forward in time then one or both animations could suddenly start running backwards. This means we need not to locate isolated pairs of somewhat similar frames but rather the best sequence of pairs forward in time without large jumps.

Considering what this method actually does, it seems it may help solve our problem as well. Proper motion blending solves the following problems for us. It

- Quantifies difference between a pair of poses.
- Locates best path through all frames forward in time for pairs of poses from two different motions.

The best path in time contains the best frame to frame match between two motions. If the motions are identical then this path will be a straight line and contain frame pairs which have zero difference as seen in Figure 10 and Figure 14. If the motions are very dissimilar the path will contain frame pairs which are very different. There will be no ”good” path. Finally, for each such best match along the path, we can compare the two frames and evaluate exactly in what aspect they are different and by how much. This is action recognition and comparison.

5.2.1 Time warp curves

The path through frames and time is called a time warp curve. A time warp curve is one of three components in a registration curve as described in [Kovar and Gleicher 2003]. A registration curve consists of a time warp curve, an alignment curve, describing global offsets, and constraints such as particular positions where the end-effectors of the skeleton should be located at particular times. The entire registration curve is designed to make smooth transitions between different motion captured actions for use in animation. To make the transitions seamless, it is required that blending between actions is done at locations which are not too dissimilar based on the above mentioned three measures. Our measure of frame dissimilarity is detailed in section 5.2.2.

We are not concerned with actually performing smooth blending and animation, so a full registra-

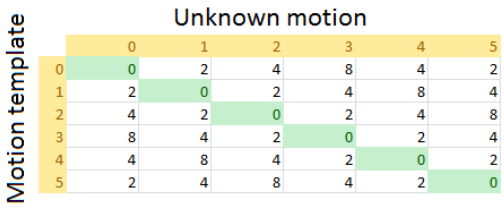
tion curve is not required for our purpose. In recognizing a particular action, the absolute positions of the skeletal joints in spatial space are unimportant since only their relative positions, described through joint angles, are significant. For this reason we will not use alignment curves. We also do not have constraints placing end-effectors at particular global positions at particular times and we can therefore also skip that part. A physical therapy exercise is defined by how joints move over time and not how certain limbs may touch the external world. This is described in more detail in section 3 and section 5.2.2.

That leaves us with time warp curves which tells us how two motions best match over time. We briefly present time warp paths here, as it is used in animation, but it should be noted that we will not implement the paths exactly as described. It is however what we strongly base our solution on.

A time warp curve can be described as a minimal cost path through a $n\,x\,m$ difference matrix containing the all to all pairwise frame differences as seen first in Figure 10. Each position in the matrix contains a scalar describing the difference between one pair of frames in two MOCAP sequences. The upper left corner represents the start of time for both MOCAP sequences, and time advances down and right from that point.

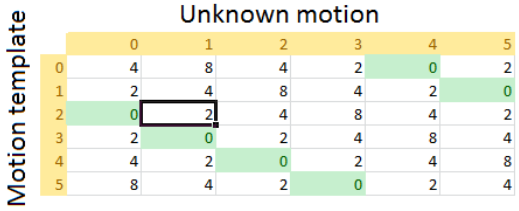
The time warp curve must advance continuously forward through time. It is therefore a requirement that the curve is weakly monotone in both frame sequences. It may never go backwards in time (from a higher to a lower frame number) but it may remain at the same frame number for one of the actions for a short while as seen in Figure 12, since this happens when the motion speeds are not absolutely identical. It should also not be broken and jump over a frame column or frame row since this is then not a continuous motion.

To make things more clear we now show four examples of difference matrices. First two for identical actions with and without a start time offset, then two similar actions with different duration and finally two non similar actions.



	0	1	2	3	4	5
0	0	2	4	8	4	2
1	2	0	2	4	8	4
2	4	2	0	2	4	8
3	8	4	2	0	2	4
4	4	8	4	2	0	2
5	2	4	8	4	2	0

Figure 10: Difference matrix and time warp curve for two identical actions starting at identical times. Observe the diagonal with zero values due to the zero difference between identical actions.



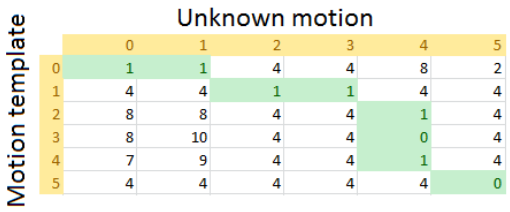
	0	1	2	3	4	5
0	4	8	4	2	0	2
1	2	4	8	4	2	0
2	0	2	4	8	4	2
3	2	0	2	4	8	4
4	4	2	0	2	4	8
5	8	4	2	0	2	4

Figure 11: Difference matrix and time warp curve for two identical actions starting at different times. We see a zero diagonal which has been offset two frames in the template, due to the temporal shift of the actions.

In Figure 10 a $6\,x\,6$ difference matrix is shown for two identical actions. Each action consists of six frames and the first column is the difference between frame 0 in the ”unknown” action and every frame in the template action. Since the two actions are identical, the difference between the two number-zero frames is zero and vice verse for number-one all the way to the number-five frames. At the same time we see that the frames zero and one is not identical which makes sense since they represent the same motion at two different times.

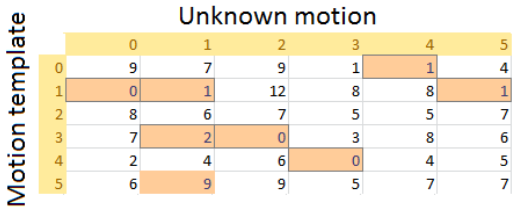
In Figure 11 we have the same two actions but with the difference that one is starting two frames after the other. This means that now the first pair of identical frames are found at frame zero and two. Note that the path reached the bottom row and then wraps to the top. This is still forward in time when using a cyclic template.

In Figure 12 we see two similar actions which do not keep same speed through the motion. Both frames 0 and 1 in the recording match frame 0



	0	1	2	3	4	5
0	1	1	4	4	8	2
1	4	4	1	1	4	4
2	8	8	4	4	1	4
3	8	10	4	4	0	4
4	7	9	4	4	1	4
5	4	4	4	4	4	0

Figure 12: Difference matrix and time warp curve for two similar actions starting at the same time but with different speed through the motion. Observe how there is a continuous path, but it changes its direction along its length.



	0	1	2	3	4	5
0	9	7	9	1	1	4
1	0	1	12	8	8	1
2	8	6	7	5	5	7
3	7	2	0	3	8	6
4	2	4	6	0	4	5
5	6	9	9	5	7	7

Figure 13: Difference matrix and time warp curve for two different actions. We see that there are no clear path through the matrix.

in the template with low dissimilarity and then frames 2 and 3 in the recording match frame 1 in the template. This indicates that the recorded action goes through the same poses as the template, but at a much lower speed. Next frame 4 in the recorded motion matches frames 2,3 and 4 in the template now indicating that the recorded action is faster than the template. In real life action recognition this is likely to happen. Not every step of a walk will have the exact same duration and speed through all individual poses.

The final example shows an attempted time warp curve for two actions which are not alike, though they do each contain poses which are found in the other action as well. Those posed are however not found in a sequence forward in time.

5.2.2 Pose/frame similarity

To calculate the difference matrix, we need a measure of frame dissimilarity. In [Kovar and Gle-

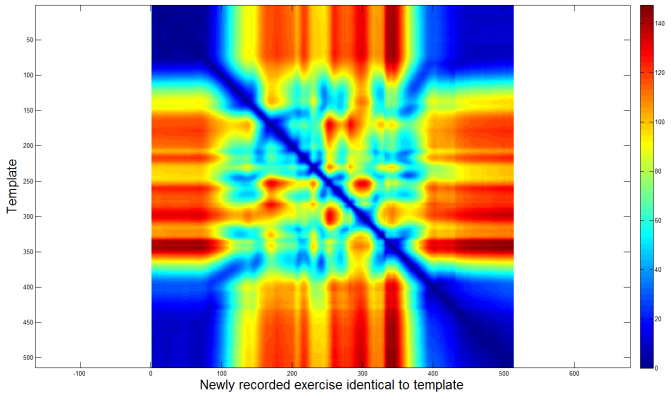


Figure 14: A motion compared with itself. There is a perfect match along the diagonal and symetry arround the diagonal.

icher 2003] the joint angles and bone lengths of the skeleton are used to calculate the individual joints positions relative to the skeleton root (the pelvis). This is commonly done using forward kinematics as seen in (1), where $T_i^n(\phi_i)$ is the transformation from the coordinate system of bone i to the coordinate system of bone n. ϕ_i represents the joint rotation in joint i. The resulting transformation of any bone system, as seen in the world system O is then given by (1).

$$T_n^O = \prod_{i=1}^{n} T_i^{i-1}(\phi_i) \tag{1}$$

This can be done for each frame along with its two neighbors on, either side, and the total number of transformed joint positions form a point cloud in spatial space. This is done for both frames which are to be compared. The error, dissimilarity, is the the sum of squared distances between the points representing each joint in one frame and the points representing each joint in the other frame.

This gives a measure of the spatial joint distances, by applying (1) to each joint in each frame and summing the difference. When the neighboring frames joint positions are included as well, we implicitly have the difference in velocity and acceleration. This gives a measure of difference in not

only current joint positions, but in joint velocities as well. In effect this is an implicit finite difference method.

This is how it is done in [Kovar and Gleicher 2003], but we desire a comparison method which can generally compare two poses from two different MOCAP recordings using two different actors. The method just outlined, which is used to build the time warp curves, is rooted solid in the world of animation. An animated character which should transition from one motion to another will not change his physical appearance and only the motion will change - not the actor performing the motion. For this reason they can make a reasonable comparison based only on the sum of spatial differences for the joints in the two frames. This will obviously run into problems if not all bone lengths are the same.

As stated previously in section 3, many physical therapy exercises are slow movements which try to move certain joints through specific rotations in a certain order. This means that velocity and acceleration is not required for a first implementation of a virtual trainer system to make sense. In this project we therefore limit ourselves to joint rotations and disregard the first and second time derivative of those rotations.

For this reason we opt for the alternative pose difference measure, which is to use the joint angles. As described in section 5.1, the joint angles do accurately describe the motion for a physical therapy exercise. It does so without being affected by different lengths of bones.

We can still illustrate the frame difference based on point clouds (using Euler angle difference), but the points will not be in spatial space but rather in Euler angle space as seen in Figure 15. The red lines in the figure connects identical joints in the two frames. This red line distance is the Euclidian distance in Euler angle space.

We define a measure of frame difference as (2) where α is the first frame and β is the other frame. Both frames consist of a series of joints indexed by

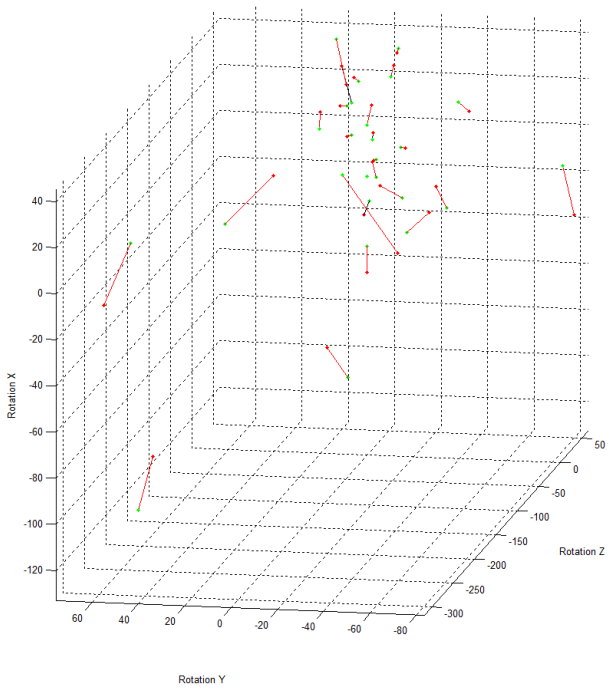


Figure 15: Two frames rendered as point clouds in Euler angle space. Red lines connect points from the same joint in the two frames. Observe how the frame dissimilarity can be related to the length of the lines.

$i = 1..N$ and named respectively α_i and β_i.

$$|\alpha - \beta| = \sum_{i}^{N} |\alpha_i - \beta_i| \tag{2}$$

While the frame difference is the sum of joint differences, we now need to decide which measure of joint different we could use. We consider two possible versions. The first is the Euclidean joint distance in Euler angle space, as shown in Figure 15 while the other is based on quaternions. The MOCAP data in a BVH file has its joint rotations defined as Euler angles and this makes this comparison both simple and quick but at the same time Euler angles suffer the problem that you may have a very large angle difference for rotations which end up in close to the same orientation.

Consider a 90 degree rotation around the y-axis followed by a 90 degree rotation around the x-axis. Doing this for the right arm in the configuration seen in Figure 5 would place the arm pointing up with palms facing backwards. If we instead rotated the same arm -90 degrees around the z-axis we would get the arm pointing up with palms facing away from the body. The two configurations are not identical, though both may actually be equally awkward, but does this difference justify an angular difference of $\sqrt{90^2 + 90^2 + (-90)^2} = 155.88$? The answer is probably no, but in our actual testing this has never shown itself to be a problem. This is likely due to the fact that actual motions do not jump from one configuration to another very different one in one step. In the example with raising the arms, an actual recording would have a large number of intermediate poses between initial configuration and final pose. Those poses would point to a difference between template and exercise early on.

The Euclidean Euler angle difference, or the 2-norm of the difference, between two $\Re^3$ joints A and B is defined as

$$|A - B|_2 = \sqrt{\sum_{i=1}^{3} (A_i - B_i)^2} \tag{3}$$

The quaternion difference requires us to convert from Euler angles to quaternions first. After conversion we have two $\Re^4$ quaternions QA and QB describing the two joint rotations. The difference is then the dot product of the two quaternions[Kuipers 2002].

$$|Q1 - Q2| = Q1 \cdot Q2 \tag{4}$$

$$|Q1 - Q2| = Q1_1 Q2_1 + Q1_2 Q2_2 + Q1_3 Q2_3 + Q1_4 Q2_4 \tag{5}$$

The quaternion dot product tells us about the difference between the two, but it does not return an angle which is easily understood. The dot product of two quaternions can be converted into an angular difference[Kuipers 2002]. This angle is the shortest rotation between the two quaternions around some axis. This axis is irrelevant for our use. The example with the lifted arm from before would, using quaternions, report an angle difference of only 90 degrees.

$$|Q_1 - Q_2|_{angle} = 2 \arccos(Q_1 \cdot Q_2) \tag{6}$$

If we compare the two difference measures in Figure 16, we see that the results are quite similar. The quaternion based method result in larger values, but the observable matrix patterns are virtually identical. If we divide the quaternion result with the Euler result, we see that the difference is not as simple as a constant scaling, but is is close. The differences range from around 100 to 140 with the highest differences located as localized hot spots. It is currently unclear exactly what causes these isolated areas of higher difference, but it seems likely that is is the difference seen in the arm example. There Euler angle difference was 156 degrees and quaternion difference was 90 degrees. In some situations the two measures are different and in other situations they will report the same difference value.

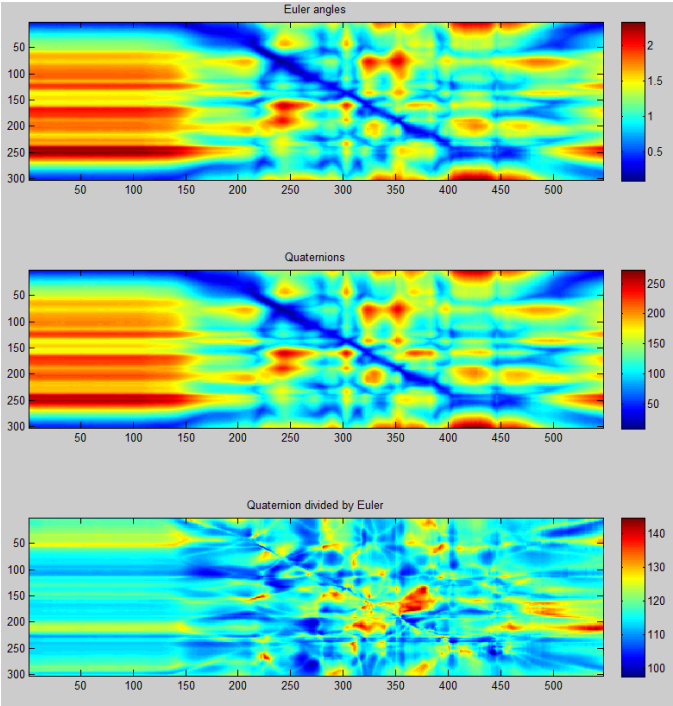


Figure 16: Comparing the Euler angle difference with the quaternion difference. We see in the first two plots that there is no clear difference in the pattern between using Quaternions and Euler angles but the last plot show that there is in fact a difference when dividing the two.

Prioritized joint comparison The skeletons we use in this project have 31 joints each and by summing over the joint rotation differences in a pose comparison, we assign equal weight to every joint. This is not desirable. If we have an exercise which focuses on the legs, then it does not make sense to compare two poses by looking at the arms. It does generally not matter if the left arm is bent in the elbow while training the legs. Additionally we may want to place greater emphasis on correct motion of the knee than on the hip. For this reason we add a weight vector W to the difference measure which changes the total Eulerian pose comparison over N joints to

$$|\alpha - \beta| = \sum_{i}^{N} W_i |\alpha_i - \beta_i| \tag{7}$$

where α and β are again the frames and α_i and β_i are the i'th joints in the two frames.

6 Timewarp paths

Given a difference matrix, we need to find the best matching path through the matrix. This path should be defined in such a way that *every* frame in the new recording matches *one* frame in the template. This should be interpreted as that for every recorded frame we need to decide which frame in the template we have a most likely match with. We do, however, not need to match every frame in the template with a newly recorded frame. We are trying to understand what the patient is trying to do to, not looking through his motions to recognize a particular part of the exercise.

To reiterate, a path is an attempted recognition. It is the systems best guess at when the patient does what. It may say "here in frame 200 you do this stretch which the template demonstrated in frame 180. Later in frame 250 you attempt to do what is demonstrated in the template frame 260". The point is that the path is the best guess at how the two actions match and when what is attempted.

Following this, we can start analyzing how the system interpreted the patients exercise.

If this best path is too costly, then we consider the actions too different to be a match. With a full matrix calculated, a path has to have a certain minimal length where it does not go back in time and where it is continuous. We will then start doing pathfinding from the first exercise frame which is located at the left column in the matrix in the previous examples. The target for the pathfinding will be the last frame of the exercise, which is the rightmost column in the matrix.

We need a method which is insensitive to when the motions start. We may have a template of a walking motion starting with the left foot front, while the unclassified motion starts being observed with right foot front. We may alternatively have a patient who stands around and waits a little before actually starting the exercise. The method should also not be overly sensitive to different motion speeds, since walking is still walking when the joints go through the same states only at a slightly higher speed.

6.1 Dynamic programming paths

In animation the data is clean. There is no extra random data from a character waiting to start an exercise. From the first to the last frame, the data is meaningful and relevant. The data is also free from MOCAP errors since it has been processed by MOCAP specialists and later by animators to ensure that is has no flaws. This makes finding the path somewhat easy.

We look for a path starting at a frame pair consisting of the first exercise frame and some template frame. From here we look for the cheapest, in terms of least frame difference, path to the a frame pair consisting of the last exercise frame and some template frame. The path should, as mentioned previously, also be forward in time and have a reasonable slope. This can be solved in linear time using dynamic programing [Cormen et al. 2001].

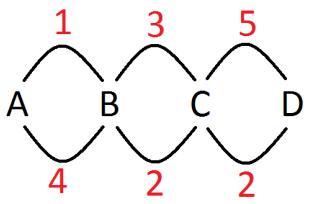


Figure 17: A simple pathfinding problem. Find the cheapest path from A to D by moving along either the upper or lower sub path. Notice that the number of possible paths grows exponentially for every extra link in the chain.

In dynamic programming we try to solve a complex problem by breaking it down to its individual simpler subproblems. We then solve the subproblems from an end and eventually arrive at the solution to the complex problem. This is the way this is solved in animation time warp paths, which is described in [Kovar and Gleicher 2003].

If we look at Figure 17 we see a pathfinding problem where we seek the cheapest path from A to D. A connects to B through an upper and a lower connection and B connects to C which connects to D. There are two paths to B and two from B to C and finally two from C to D. That is a total of $2^3 = 8$ paths we should check to find the cheapest one. This is an $O(2^n)$ problem and for large n it is very computational costly. For a typical difference matrix we may have 900 columns with each 400 rows. That gives us $400^{900} = 7.14E + 2341$ possible paths. This is obviously not doable by brute force.

Looking again at Figure 17 we observe that the cheapest path from A to D is obviously the cheapest path from A to C followed by the cheapest path from C to D. This is the low path. We then observe that the cheapest path from B to C is the, still unknown, cheapest path from A to B followed by the cheapest path from B to C which is clearly the low path. Finally we observe that the cheapest path from A to B is the high path. This means that we could just find the path from A to B and then from B to C and finally to D along the cheapest step each time. This method requires only three

comparisons of two numbers each time. It is $O(n)$ and for the matrix mentioned before we not have 900 comparisons among 400 numbers. This is $900400 = 360000$ operations, which is not a problem.

$$
\begin{aligned}
P(A,D) &= P(A,C) + min(C,D) \\
P(A,D) &= P(A,B) + min(B,C) + min(C,D) \\
P(A,D) &= min(A,B) + min(B,C) + min(C,D) \\
P(A,D) &= high, low, low \qquad (8)
\end{aligned}
$$

6.1.1 Problems for the method

As mentioned before, this is easy for clean entirely meaningful data used in animation. It is, however, not quite as easy for MOCAP matching of noisy and potentially meaningless data.

Sporadic jumps in difference As an example, imaging a difference matrix with a nice clear pathway of low difference from left to right Figure 18 which is suddenly blocked by a thin vertical region of very high difference values. If the pathfinding method is limited to moving from left to right and stay on a horizontal or downwards course, then it will turn when reaching the high difference region and it will move down and around the obstacle even though the ”natural” thing would have been to power through the thin region and stay on track. It will find the path of lowest difference, but it will not realize that the high difference region was probably due to a data error.

Late start Another problem is when the exercise does not start right away. This gives a number of columns in the difference matrix which have some potentially low frame difference though the data is essentially irrelevant. This will be able to affect how the path is traced when the exercise actually does start. In Figure 19 we see such an example. There is an initial irrelevant period of somewhat decent frame matches before the exercise starts. The actual exercise start is at the bluish vertical line. Because of the initial somewhat decent frame matches, the path tends to stay high and continue along the yellow region rather than jump down low and pick up on the very good match at the green line. The path taken will be through the yellow region and then the wide yellow line. The path that should have been taken would start at the wide green line and then follow it to the end.

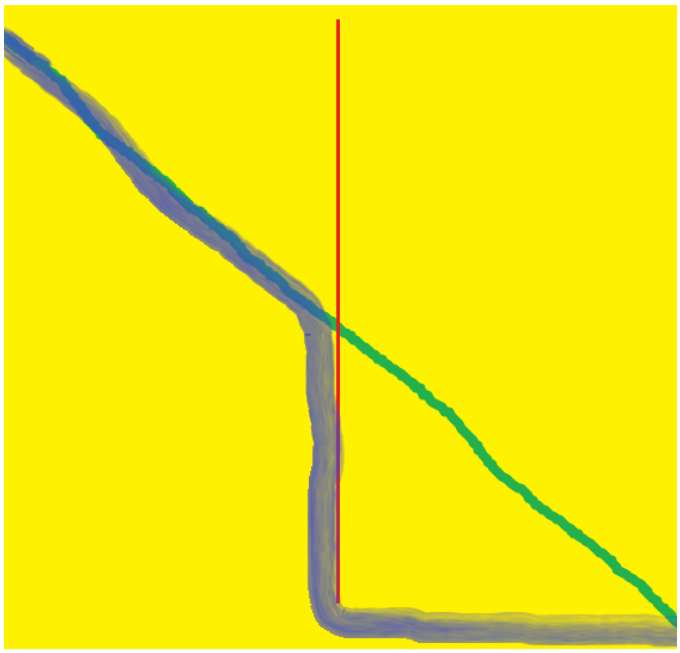


Figure 18: Path blocked by thin region with very high frame difference. This causes the pathfinder to break off and find a cheapest, but useless path.

Late data influences early path We need to have all the frames of the exercise before we can start to find the overall best path. If we have columns arriving one at a time as the patients exercise is recorded, we will have a dynamically built difference matrix. In the dynamic programming method, as outlined above, we have that $P(0..t) \neq P(0..t+1)$ compared over the interval $[0..t]$. It was influenced by data arriving at a later time and changed. This could mean that at some earlier time we interpreted the path as being at one location and later, when new data arrives, we may rethink and

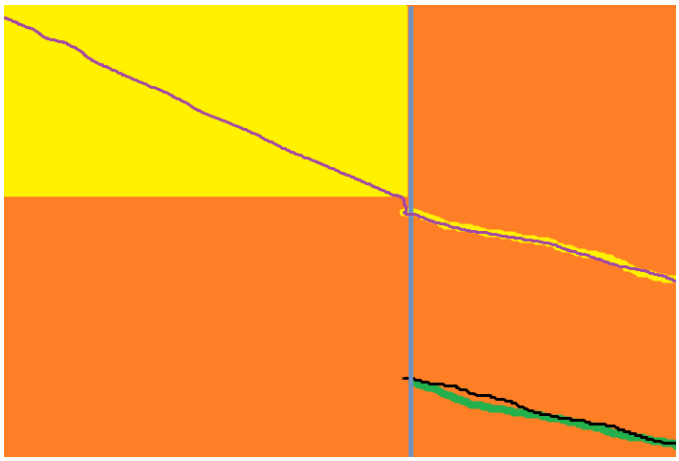


Figure 19: Irrelevant data with a fairly low frame difference spoils the path through the actual exercise. The true path should start after the vertical line and ignore data to the left of this position.

change the path back in time. While this change of path based on extra data is desired for analyzing a whole difference matrix, it is counter intuitive for a matrix being built, analyzed and providing feedback one frame at a time.

Method is left wanting To sum it up, the direct and deterministic method will indeed find the cheapest path, but sometimes this is not really what we want. We want the path to power through small data errors and we want it to keep its options open when we have large regions with a somewhat decent, but not really good, match. Initially we attempted fixes for these shortcomings and did improve on the original method, but the problems persisted. We attempted to begin tracing paths whenever the difference was below some threshold and continued these paths as long as the difference was below some other higher threshold. This gave us a series of sub paths which we then attempted to join with limited success. We also added a preference for moving in a down right direction even though the difference might be higher there. The solutions were still not reliable. If data error was just high enough, it would still stop up and if frame differences varied enough, it was impossible to find good values for the low and high threshold for when a path should start and when it should stop. Further more, the more we hacked the original dynamic programming solution with penalties for this and preferences for that, we realized that we were moving farther and farther away from a solution we could analyze and understand. For that reason, we dropped this path and started looking at the probabilistic method instead.

6.2 Probabilistic reasoning

We base our understanding of motion similarity and comparison on time warp paths, but for our purpose we have problems which do not arise in animation. We use the *concept* of time warp paths to *understand* the problem and find a solution but we do not use the paths exactly as described in [Kovar and Gleicher 2003]. Instead we use probabilistic reasoning. We do not know exactly where the path goes, but we can provide probabilities. This lets us give a running best estimate and it lets us include a measure of confidence.

If an instructor observes a patient who is supposed to do a certain exercise, then the instructor will initially look for when the exercise seems to start, and he will not comment on the initial movements getting into position. When the exercise does in fact start, he will expect a match between template and exercise which goes forward in time and which looks the same to a certain degree. If there is a sudden brief large error (the patient quickly scratching himself or otherwise doing a quick movement, or in our case a MOCAP error) the instructor will perhaps comment on the brief error but he will not look all perplex and suddenly not know what is going on when the patient continues the exercise. Rather he will understand where they were and based on that, he will realize where the exercise continues. If the patient is to flex his knee but he during the bend, for some reason, briefly stretches it a few degrees and then continues flexing, then again this is to be expected and it is something an instructor would not be confused by, though he may or may not make a comment. If the patient

goes through an exercise, but half way through we has to quickly relocate his body to be more comfortable, and then backs a few seconds up starts the exercise at that earlier point, then again this should not confuse an instructor, and it should not confuse an automated system.

In other words, we do not have pure and perfect animation data. We have MOCAP data which has not been cleaned up, so it may hold brief large errors. We also do not have an actor which we can ensure will go through exact movement in exact order and always forward in time and we do not know an exact starting time of the exercise.

To recap; by a path we mean our best understanding of how the exercise frames matches the template frames. A path going through frame A in the exercise and frame B in the template simply means that we believe frame A was meant to emulate frame B in the template. This does not mean that A was in fact the best emulation of B. It only means that we believe this is what the patient tried to do and that this is what we should comment on. We base the belief on our understanding of how a pose changes into another pose over time as well as on our observations regarding the frame matches.

The probability of a frame pair having the path run through it depends on

- Does the pair have a low pose difference?
- Do we believe the path just passed through a frame pair in the immediate neighborhood?

This means that a path will accept more and more adversity the longer it has been moving on through good frame matches, since it will tend to grow in confidence.

At one end of the scale we could show just the best exercise-template match as in Figure 20. At the other end we have probabilities where the pose match is just one factor and not the whole picture. In Figure 21 we see the most probable path. To make the illustration more clear, only the path probabilities are shown. We note that the minimal pose difference is a bad measure in itself. The

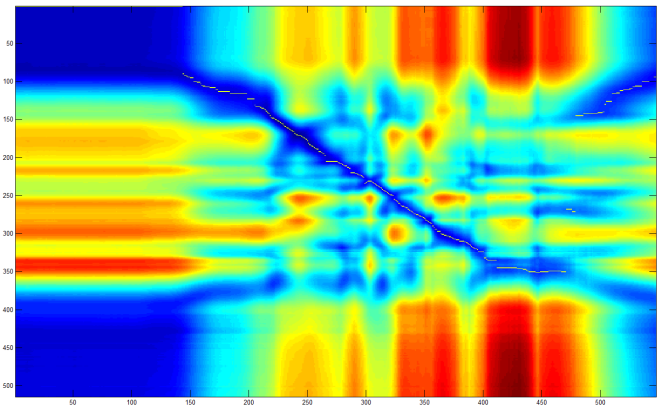


Figure 20: Comparing exercise with template and showing the best match for every exercise frame. This is just the minimal pose difference. Observe how the path jumps up and down and is in no way continous.

path jumps back and forward in time and it is in no way a continuous path. This is comparable to a very indecisive instructor who just does not know how to relate past and present. The probabilistic path on the other hand is blurry (uncertain) initially when the exercise has not started and then it focuses on the actually matching frames before becoming confused at the end when the exercise stops.

6.2.1 Most probable path

We will be using a method known as a Hidden Markov Model HMM, as detailed in this section. The model, in relation to our project is explained, but for a thorough explanation of HMM we refer the reader to texts such as [Russell and Norvig 2003] and [Bishop 2007].

A HMM considers the development of probabilities over time. What is seen in Figure 20 can be considered the highest path probability if we look *only* at one point in time, at a time, while in Figure 21 we see the effect of looking at probabilities both now and before as we move forward in time.

The H in HMM, from hidden, indicates that the state variable which we are really interested in, is

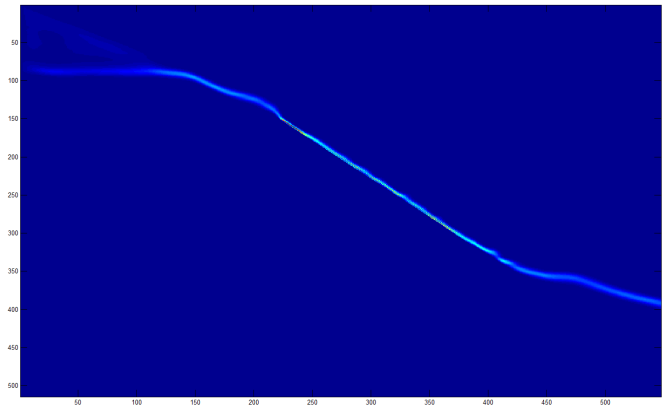


Figure 21: Probability of path based on pose similarity and neighbor, backwards in time, probability. Notice how this path is continuous, though slightly blurred at times.

hidden. We do not *know* where the path is, but we try to find a best estimate. This is based on observations, sometimes referred to as evidence, and on transition probabilities. This model is advanced in time, where one time step is equal to advancing one frame in the exercise. Again, remember that we start with only one exercise frame and build on this, one frame at a time.

State vector The interesting state over time is the path where exercise and template matches. This is called X in the following. This state X will be encoded as a frame index into the template. If the path is along the diagonal, we have a perfect matching comparison, where we see X_t, for $t = 1..n$, will be $[1,2,3..n]$ where n is the last frame. The subscript t indicates that this is specifically the state at time t and not at some other time.

Transition matrix The matrix T describes how probable it is that the hidden state X changes from one state to another. The probability is described by a $n\,x\,n$ transition matrix where n is the number of states. If an entry a,b in this matrix is 0.5, it means that the probability of going from state a to state b is 50%. The columns of this matrix sum to 1 and all entries are greater than or equal to zero. We will get back to how this should be defined for our purpose.

Evidence Evidence is termed E in the following. We do not observe the hidden state X, but we observe something which gives us clues about the state. This is the evidence E. As in a court of law, evidence does not actually tell us if a person is guilty of a crime or not. It merely points to what is most likely. In our case, the evidence is the frame pair matches. The better the match, the more probable it is that they are *the* match which the path passes through. In the model we do, however, not look at how likely it is that the path goes through this state at the current time, instead we turn this on its head and look at how likely our evidence is given a certain state. We do not look at probability of X given E $P(X_t|E_t)$ but at probability of seeing the evidence we saw given X $P(E_t|X_t)$. This may at first seem strange, but it is the normal way of doing it and it makes perfect sense as we show shortly. Note that generally if we do not use evidence, then, as time passes, we would only see a smoothing of the initial probability whenever T is not favoring any particular state over the others. If we do not observe the unknown, indirectly, through evidence then the uncertainty will simply grow over time till we know nothing.

Probability vector The probability of the system being in any one of its possible states is described by the probability vector. For a n state system, this vector has length n. The sum of its entries will be 1 and all elements will be greater than or equal to zero, since they describe probabilities. This vector is written as $P(X_t)$ for the probability vector of the state at time t.

Initial probability This method calculates current probabilities based on previous probabilities, but initially there is only the current. The initial probability, the prior probability, $P(X_0)$ is based only on the frame pair differences. The more different, the less likely it is that the path goes

through here. Other things could be taken into consideration, such as the fact that a path would be more likely to start in the start of the template than in the end, but we consider only frame pair differences for now.

6.2.2 Markov assumption

Naturally the probability vector at any one point in time depends on all that went before it. This could mean that for a proper estimation of $P(X_t)$ we would need to consider an ever increasing number of prior evidence as t grows. We would also have to consider a larger and larger number of conditional transition matrices the more previous states the current depend on.

This can be avoided by using two assumptions.

The first is that the probabilities are defined by a stationary process. The process of change in probabilities does not itself change. The system is the same but the probabilities in the system do change. In the context of our project this would mean that if it is initially more likely that a state of $X_t = n$ leads to $X_{t+1} = n+1$ than it is that it leads to $X_{t+1} = n-1$, then that is true for the entirety of the analysis. The mechanism does not change though $P(X_t)$ does.

The second assumption is that the state only depends on a finite set of previous states. Rather than having X_t depend on all X_n where $0 \leq n \leq t-1$ we say that it depends on a finite number of previous states so that X_t depend on all X_{t-n} where $1 \leq n \leq \infty$. This is generally not exactly true. Everything in the past impacts the present, but you could argue that by looking at the immediate present, which in turn looks at its immediate present, we do look all the way back through the chain of immediate presents. If $P(X_t)$ depends only on $P(X_{t-1})$ this is considered a first order Markov process. If it depends on states further back it will be second order, third order and so on.

In our method we limit ourselves to a first order Markov process.

6.2.3 Defining prior probability

We define prior probability (9) as the reciprocal error multiplied with the reciprocal sum of reciprocal differences. This results in a probability vector which sums to 1. The length of this vector is n, which is the number of frames in the template we matches with. $difference_n$ then represents the difference between the first exercise frame and template frame n. We do this since initially we have no information to guide us in choosing a path, except the frame differences; and the probability of a match is inversely proportional to the difference.

$$P_0 = \begin{bmatrix} 1/difference_1 \\ 1/difference_2 \\ \vdots \\ 1/difference_n \end{bmatrix} \frac{1}{\sum_{i=1}^{n} \frac{1}{difference_i}} \tag{9}$$

6.2.4 Defining probability transition

Finding the best transition matrix is not a simple task. It requires in depth analysis of actual optimal paths or a very detailed understanding of the underlying problem. For this project we have defined a matrix (10) which is based on our understanding of the problem. It works, but it could most likely be improved as detailed in section 9.

We base the matrix on three observations.

- We can go from any state to any other state.
- It is most likely that we we go to a state which is close to the current one.
- Time advances so it is more likely that we go to a new state which is lower (row-wise) than one that is higher.

Based on this, we device a transition matrix with columns which sum to 1. It has larger probability values close to the current state and non-zero elements in all other states. Additionally it has the center of mass of the neighbor probabilities shifted one state (row) down. The shift is written as α

in the following. The shift is to include our expectation of forward movement in time at approximately the same speed as in the template. It should be noted that this way an exercise can pause and it can even go back in time. This without being prevented from this by zero probabilities. If we wish to prevent the system from recognizing paths that behave like this, we would simply ensure a zero probability for such behavior. We do, however, not wish to prevent it since we want to recognize the intent of the patient and then consider how this intent matches the template.

We implement this by writing a 1-dimensional Gaussian distribution into each column of the matrix. The standard deviation is so that the bulk of the probabilities are within the closest 7 states in either direction. We use $\sigma^2 = 5$. We place the mean of the Gaussian just below the main diagonal of T. For column 10 the mean is at row 11, meaning that the highest probability points towards the next state forward in time. The only thing left to do, after writing the shifted Gaussian, is to ensure all columns sum to 1. A Gaussian is infinite, so its integral is only 1 for an infinite domain. For this reason the values should be scaled upwards, but the primary reason for normalization is that the border states near the edge of the matrix will have a sizable portion of the Gaussian outside the matrix. In row 1 almost half of the Gaussian is lost outside the matrix since there is no probability of going from state 1 to the nonexistent state -1,-2 etc. By scaling the columns, those border cases will improve their valid probabilities. A transition matrix is shown graphically in Figure 22 for a $100\,x\,100$ matrix using a forward shift in time of 10 states. Using 10 states is to make the effect more clear. It is not something which makes sense in itself since it implies that the exercise runs ten times faster than the template. Note that the slope of the band of high probability in the matrix is still 45 degrees - only it is offset 10 elements so state 1 leads to state 11 which leads to state 21 etc.

In (10) r means row and c means column. α was previously defined to be the expected forward shift in state taken every time step. The factor before

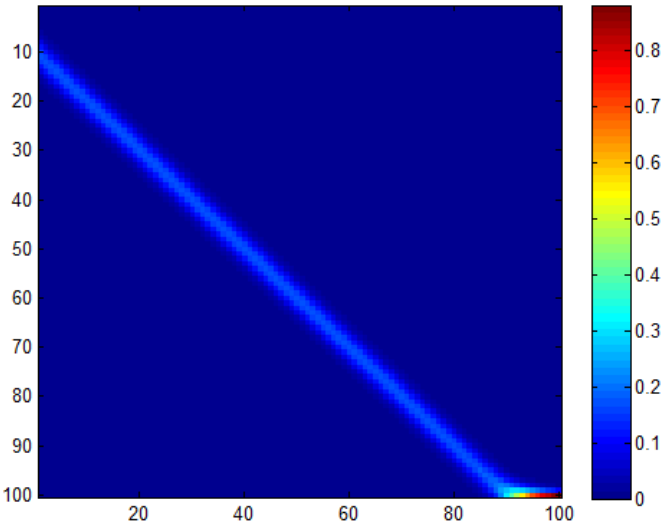


Figure 22: Color coded transition matrix with a forward state shift of 10 to better show the effect. The shape is a Gaussian in the columns with its mean shifted one row for every column.

the matrix element is to ensure normalization of the 1D column Gaussians, so that the integral over all rows will be 1. It is strictly not necessary here since we will normalize the columns afterwards to deal with the problem of lost Gaussian density over the edge of the domain. It is only included for completeness.

$$T = \frac{1}{\sigma\sqrt{2\pi}} \begin{bmatrix} e^{-\frac{(1-1-\alpha)^2}{2\sigma^2}} & \dots & e^{-\frac{(r-c-\alpha)^2}{2\sigma^2}} \\ \vdots & \ddots & \vdots \\ e^{-\frac{(r-1-\alpha)^2}{2\sigma^2}} & \dots & e^{-\frac{(r-c-\alpha)^2}{2\sigma^2}} \end{bmatrix} \tag{10}$$

Defining evidence If the frame difference is large then that makes it less likely that the path passes through that frame pair. We define the probability of observing evidence given a certain state as $P(E|X) = \frac{1}{difference}$ For $state^t_{10}$ (path passing through row 10 at time t) with a difference matrix entry at $10,t$ of D, this gives us $P(E_{10}|X_{10}) = \frac{1}{D_{10}}$.

Doing this for all states results in an evidence vector n elements long for n states. We can then easily

define E based on D as (11). Our method has to ensure that the sum of elements in E is 1. While one could argue that evidence should be able to push the total probability low, if the evidence does not match any state, this would be wrong. However unlikely all states might be, one of them *is* the correct one, so the sum of all probabilities should be one to indicate this. If not, we would have states not described by our system. We do this by multiplying D with the reciprocal sum of its elements.

$$\frac{1}{E} = D\frac{1}{\sum_{i=1}^{n} D_i} \tag{11}$$

Calculating with HMM A Matlab implementation of the following methods is shown in section B, for the highest level of the calculations. The basic support methods are not included.

Given the prior probability $P(X_0)$, the current probability $P(X_t)$, the transition matrix T and the evidence E_t we are ready to calculate the next probability $P(X_{t+1})$. Note that multiplying a transition matrix with a probability vector results in a (new) probability vector $P_2 = TP_1$.

$$P(X_{t+1}) = P(X_0)\prod_{i=1}^{t} P(X_i|X_{i-1})P(E_i|X_i) \tag{12}$$

In order to advance out probability matrix, given that we use a first order Markov model, we therefore make the simple matrix multiplication

$$P(X_{t+1}) = T\,P(X_t)P(E_{t+1}|X_{t+1}) \tag{13}$$

After this calculation, the probability vector P may need to be rescaled so its elements sum to 1. If we imaging a two state system with $P_1 = [0.1, 0.9]^T$ and $P(E|X) = [0.9, 0.1]^T$ then multiplying the two will result in $P_2 = [0.09, 0.09]^T$ which does not sum to 1. The first state was very improbable, but the evidence supported it. The other state was very likely, but the evidence did not support it. After scaling we have $P_2 = [0.5, 0.5]$.

These calculations result in a path as seen in Figure 24 and other.

6.2.5 Best path in hindsight

As for the dynamic programming solution to best path, we can make a probabilistic calculation that looks at all information, after the exercise is completed. This could give us the best path through the entire difference matrix.

There is a commonly used algorithm known as the Viterbi algorithm[Forney 1973] which given a chain of observations will tell us the most likely change of states. We have the pose difference observations over the entire exercise and based on them we can find the states, which are the template frames that match the exercise frames. We will not describe the algorithm here, but only mention it for completeness and refer the reader to texts such as [Forney 1973], [Russell and Norvig 2003] or for easier reference [Wikipedia].

While the method could give us the best match, we are currently not considering recognition in hindsight, and for real-time feedback this method has the same problems with a possible change of mind when new data arrives. We need to understand the exercise right now and not when it is completed.

6.2.6 Learning probabilities

In section 6.2.4 we argued for a certain form of transition matrix. The matrix was designed by hand based on our understanding of the problem. As an alternative, we could have let the system observe a number of best paths and then, based on those, design a better transition matrix. This would mean that we needed access to those best paths, which we do not have. A human could conceivably be tasked with plotting "best" paths through a difference matrix and the system could then learn how the states (row locations of the path) changed

over time. A perhaps more reasonable alternative would be to let the system look at a large set of difference matrices - not exercise frame by exercise frame but rather the entire comparison as a whole, as mentioned in section 3.1 and section 6.1. This would let the system get a better chance of finding the truly best path over all time and learn from this[Welton and Ades 2005].

It is, however, not a path we have explored in this project. We will refer this to future work.

6.2.7 Final paths

Having a probability density over the entire difference matrix lets us choose the actual path. The path consisting of a sequence of template-exercise frame pairs with one such match for each exercise frame. Using the probability density we find the path simply as maximal likelihood. For each exercise frame we select the template frame with highest probability in the probability matrix P. Such a path through a difference matrix is seen in Figure 23.

$$Path = [1..n, \arg\max_{row} P(row, 1..n)] \qquad (14)$$

The path is the [row,column] pairs of indexes into template and exercise. The columns are a sequence of 1 to n for n exercise frames, while the row if the maximal column value for column 1..n in P

6.3 Confidence level

The binary path is generally a very good guess, but it does not in itself convey any information about when the path is based on the best guess among many almost equally good guesses and when it is based on a guess bordering absolute certainty. The point is that the path holds very little relevant information when it is based on a low probability guess and it holds much information when it is based on a high probability guess. Further more, this uncertainly can be taken into account when giving feedback. If the path is very fuzzy then the match is generally bad. Either the exercise has not started yet or the exercise was poorly executed. An entirely fuzzy path indicates a bad exercise attempt.

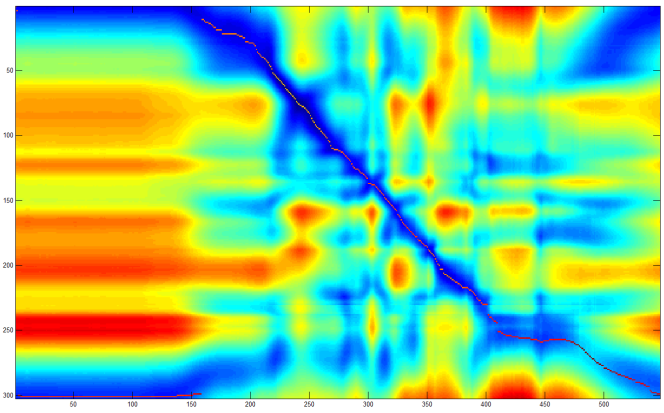


Figure 23: Binary maximal likelihood path found through a tornado kick.

The path probabilities seen in Figure 24 show a path which starts out fuzzy and ends fuzzy but is quite certain in the central region. This is from the roll and flip exercise which truly has a bad template-exercise match. This confidence can be formalized in two general ways. We write $C(f)$ as confidence in path at frame f in the exercise. P will be the path probability matrix seen in Figure 24 and $P(f)$ is the column vector at column f in P. In (15) we simple look at the maximal probability value for any given column (exercise frame). In (16) we choose to not look at how high a probability the estimated path has, but rather at the variance of the probabilities. A high variance indicates that our probabilities are spread out. One might expect the two to give the same confidence indication since a high variance will also cause a smoothing of the probabilities which results in a lower max value, but this is only really true if the probability distribution over a column is Gaussian. If, on the other hand, we have two widely separated points with a probability of 0.5 each, then one is chosen as the path and the max value returns 0.5. This seems to indicate a high degree of certainty. If we

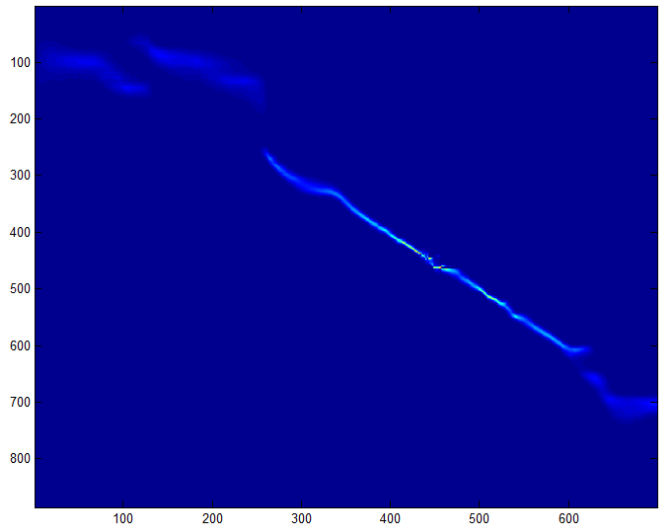


Figure 24: Path with an initial and a final part which is quite uncertain.

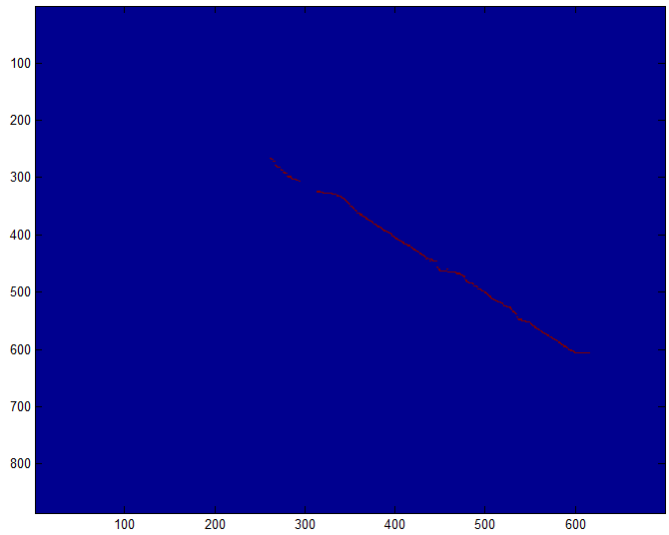


Figure 25: A binary path which has elements ignored when they are based on probabilities below 0.05. Notice how the path is not defined for every column (exercise frame) due to the initial and final bad match between template and exercise.

instead look at the variance, then we will see that the probability density is quite spread out. We will then get a high variance. This means that (15) will return a high confidence while (16) will return a low confidence.

The example is an extreme, but the point still stands: (15) may return the true P-value at the path, but (16) better describes how the probabilities are spread. For this reason we opt for using the variance measure (16).

$$C(f) = max(P(f)) \tag{15}$$

$$C(f) = \frac{1}{var(P(f))} \tag{16}$$

We could change the calculated path by ignoring parts with a low confidence. In Figure 25 we see such a path. Only the parts we are sure about are included. The actual limit on confidence should likely be found through experimentation depending on the use. When such a path is used for exercise feedback, we could chose to not give feedback while the path does not exist, or we could actually inform the patient about the mismatch.

6.4 Recognizing a particular action

If we build one large template which is the concatenation of a number of different actions, then we should see a strong probability match through *one* of the actions. Since each action will be located between a certain start and end row in the resulting difference matrix, we could simply see what interval the path is primarily contained within.

In Figure 26 we see such a recognition. It is not easy to see in the image, but the probabilities are not isolated to the kicking action initially. The initial probabilities are spread between all three actions which is clearer in the plot of difference down through the columns representing exercise time t=1, t=10 and t=50 in Figure 27. Quickly after starting the probabilities grow inside the correct action, while it fades in the others. We can conclude that at t=50 we are quite confident that the match is along row 971. The other candidate with half the probability of the first match is row 923 and both are located in the rows of the kick action. It is clear that the blue plot at t=50 has de-

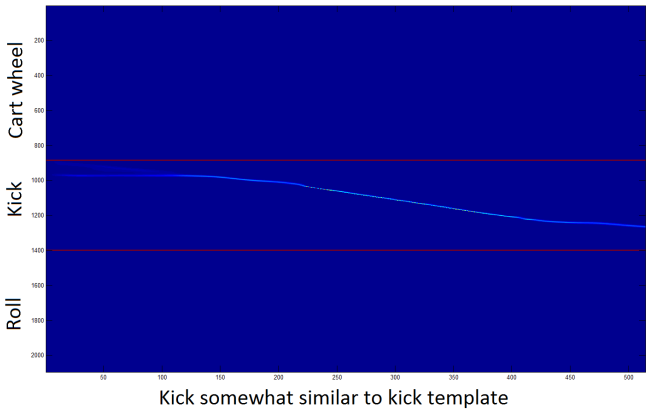


Figure 26: One particular action recognized among three candidates. We see that the path probability is contained inside a single action.

cided on the kick with an almost zero probability of any match outside that action.

The same plot is shown for a recognition of a cart wheel exercise in Figure 28. The template for this exercise is located first in the concatenated template and even initially only the true template has significant probabilities of being matched. As time progresses this probability just increases.

As an alternative, we compare a jumping action to the same three templates. This time the action does not match any of the templates, and as we see in Figure 29, there is no clear path. It is blurry and jumps up and down. The conclusion would be that this action did not match any of the templates.

7 Action comparison

When we have a path through frames and time, we have a best guess at what the patient was trying to mimic. Every frame in the patients exercise will have a maximal probability of matching a specific frame in the template. To give the most reasonable feedback, we should therefore compare what the patient did with what he tried to do.

We do this by continuously comparing the currently recorded exercise frame with the matched

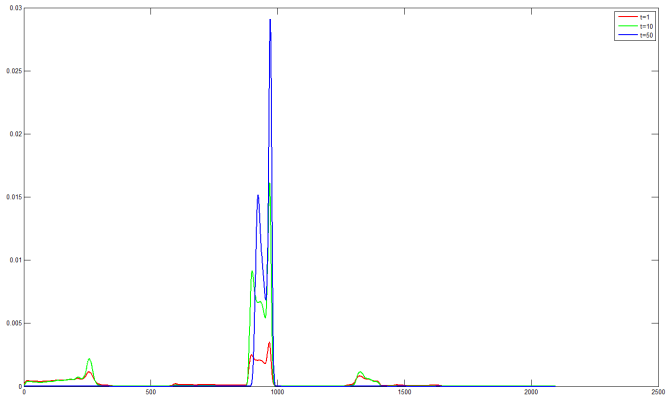


Figure 27: First 10 columns of the path probability. It is seen that all three candidate actions do have a non zero probability, but that the probability quickly becomes contained entirely inside one action

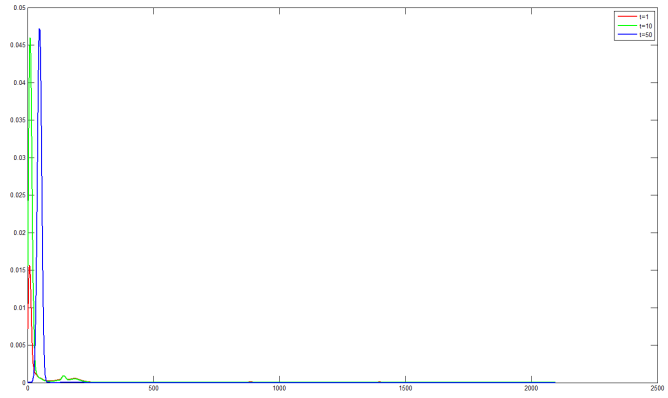


Figure 28: Recognizing a cart wheel exercise which is located early on in the concatenated template. The probabilities are increasing and concentrated for the later frames.

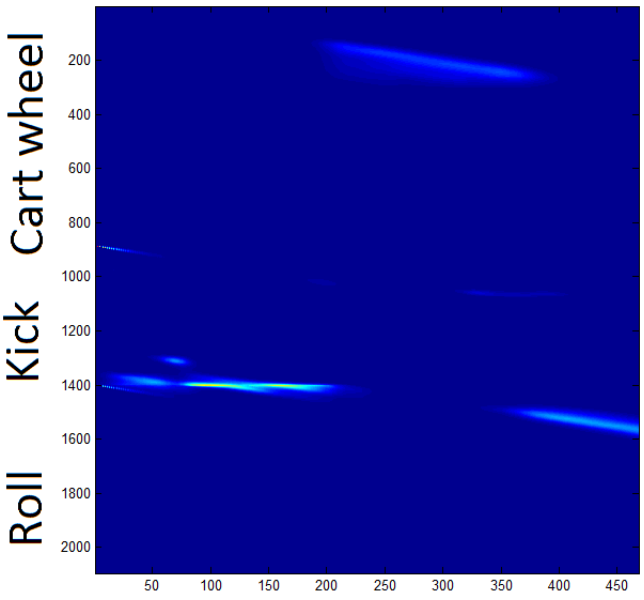


Figure 29: Attempting to match an action to three templates which are all different. We see that the path probability this time jumps among the different templates and that no clear path is found.

template frame. If the difference is low, we do not correct the small errors but rather give the patient a green light for continuing the exercise. If, on the other hand, there is a larger discrepancy between what is attempted and what is observed by the system, we will inform the patient that the exercise is not correct. Given that we know which frame in the exercise that is most likely attempting to match which frame in the template, we can make a joint by joint comparison and give feedback informing the patient if, as an example, the left knee is bent too much or if the right elbow is bent too little.

We subtract the two frames to get the pose difference measure from section 5.2.2 in Euler angle space.

In order to describe this difference in human understandable terms, we need to name the joint rotations so it is clear for this system what Euler angle rotation corresponds to ”rotating the hip outwards” etc. The most reasonable solution is to let the system observe, through a MOCAP recording, an actor moving a joint in a particular way and tell the system the name of this motion. This is much due to the observations in section 5.1 where we observed that even a simple bending of the knee seemed complex in Euler angle space.

We do currently not have access to MOCAP of that type, so we resort to using the initial pose from Figure 5 defined in section C. By observing the skeleton definition we can make a passable estimate of the joint rotation axis.

8 Results

We present some of the results from our test of the system.

The corner stone of our comparisons is the time warped alignment of two actions. Without this, the pose comparisons will not be possible. We first show that time warps do in fact let us align actions so that the sequence of poses match well. This is demonstrated in section 8.1

Following this, we show some examples of actual calculations of best paths using such a time warp in section 8.2. Eventually we have to drop probabilities and choose exactly which template frames should be matched with which exercise frames. Such final paths are shown for four different action comparisons.

It is important that the system will not easily be thrown off course, so we have considered different ways to confuse it by manipulating with the difference matrix and we show that while large scale manipulations will confuse the system, it is able to, to a large extent, realize its confusion and to get back on track as soon as the manipulation stops. This we show in section 8.3.

Finally, the system should compare specific frames and quantify the changes needed to transform one into the other. This is shown in section 8.4, where three ”wrong” poses are attempted corrected.

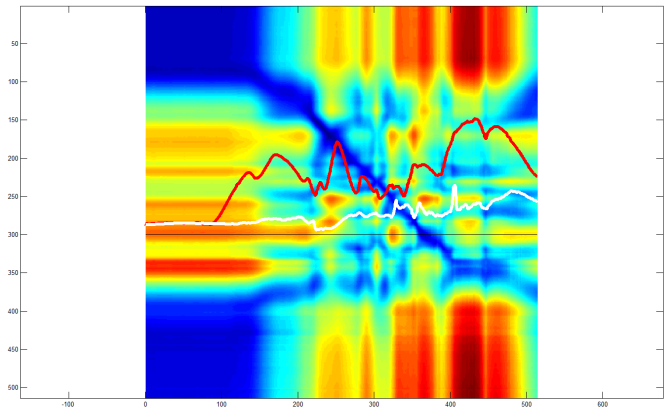


Figure 30: A difference matrix with plots of time warped path error in white and default time alignment error in red. Notice how the un-warped path shows a larger error than the warped path.

8.1 Effect of time warp

By warping time for either the template or the recorded exercise, we obtain a better alignment of the two actions than if we do not warp time. In Figure 30 we see a difference matrix and on top it has a red plot showing a frame by frame comparison as well as in white a comparison with the template time warped to match the change in starting and ending time of the exercise as well as the difference in technique execution. It is seen that the warped path does in fact give a better match over all the frames. It should be noted that the two actions we tested warping with here do start nearly at the same time and the actions are quite similar in execution speed. For this reason, the effect is not greater, but still it is pronounced.

In Figure 31 we see a path probability which is based on an action aligned with the template. This results in a matrix with maximal likelihood for the path running through the matrix diagonal. Both actions now also take the same number of frames to complete. When comparing this to other, non warped, path probabilities, such as Figure 21, the effect is very clear.

The true test of warping is to compare actual rendered skeleton frames with and without time warping. The effect of using time warping is seen in

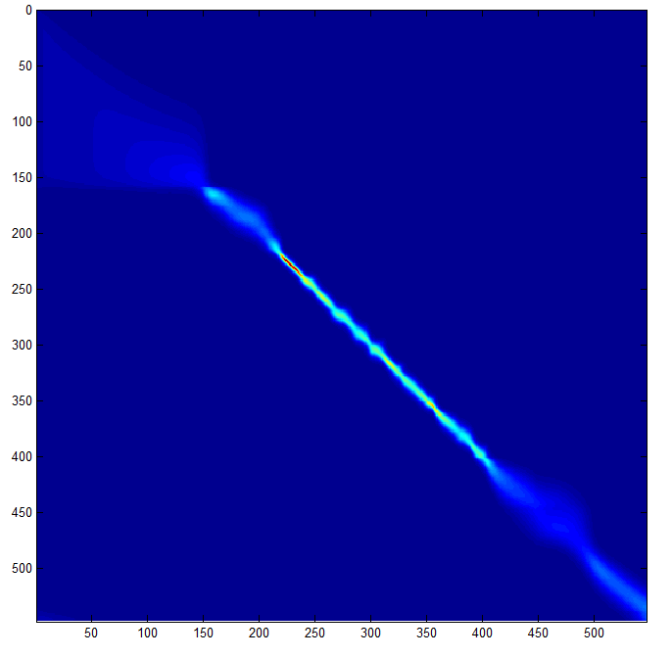


Figure 31: A path probability after aligning the template and the exercise using time warping. We see that the maximal probability is now located along the diagonal of the probability matrix.

Figure 32 and Figure 34 while the unwarped frame to frame comparison is seen in Figure 33 and Figure 35. We see that by warping time, we can find the good match which we do not initially have. We also see that the frames are still not identical, which is not to be expected since we do not morph the pose, only the time at which it occurs in the action.

8.2 Best path

We have tested the found paths with a number of exercises. Of those tests we show the paths for four comparisons.

In Figure 36 we see an almost unbroken path through the entire exercise. Initially the difference matrix has low difference both high and low and given that the path tends to move downwards, it starts out running horizontally along the bottom. As soon as evidence suggests otherwise, the path jumps up and locks on to the ”true” path which is

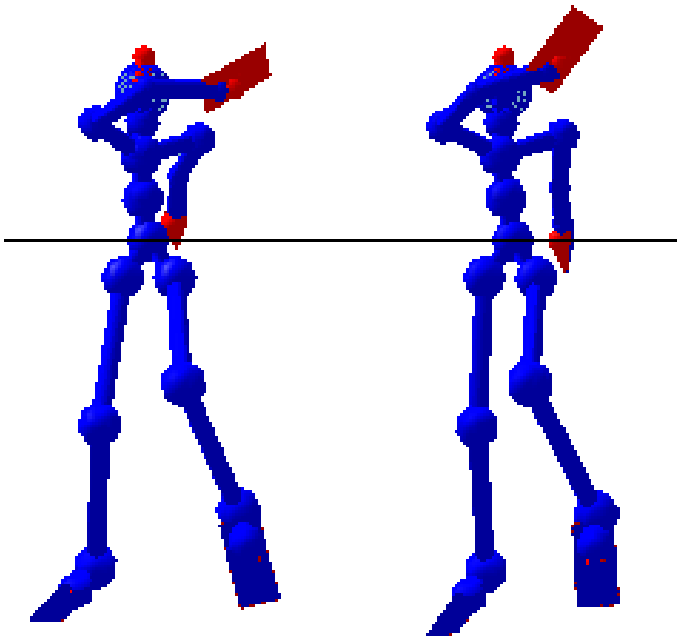


Figure 32: Frame 250 with time warp matching. Observe that the two poses are very similar.

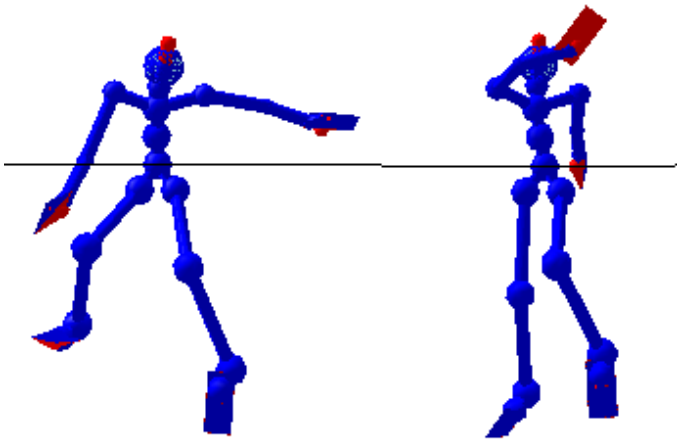


Figure 33: Frame 250 without time warp matching. Observe that the two poses are very different without alignment.

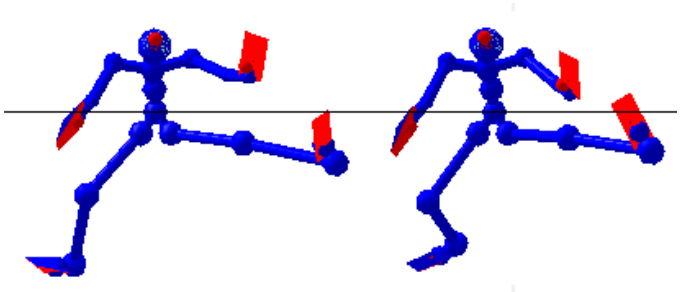


Figure 34: Frame 350 with time warp matching

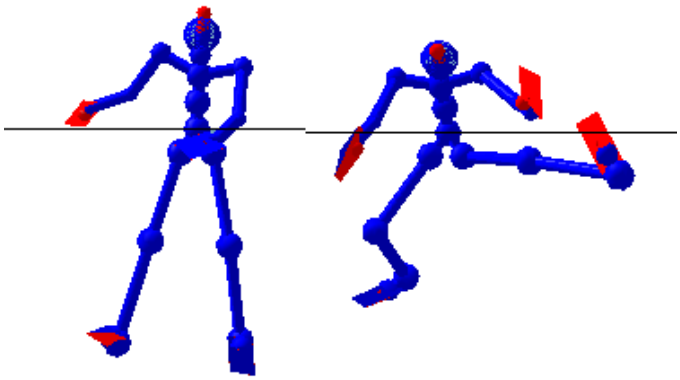


Figure 35: Frame 350 without time warp matching

followed to the end of the exercise. The initial period, where the path wrongly follows the bottom, is the same period where the exercise has not really started yet. The actor waits to start, so the system cannot, and should not, recognize anything here.

In Figure 37 we see the same initial dead time when the exercise has not started. When the exercise actually starts, the path finds a good match and follows it during the duration of the exercise.

The cart wheel exercise is seen in Figure 38. This is a MOCAP which has data error with a joint suddenly flipping. This results in the artifacts seen in the image, but again we see the path being a little uncertain in the initial waiting period and then it locks on to the correct match. In the idle period in the end, the path again jumps a little around.

The exercise consisting of a forward roll followed by a split is seen in Figure 39. Again there are data errors, but the path is found. There is an initial idle period, but incidentally it results in a narrow hor-

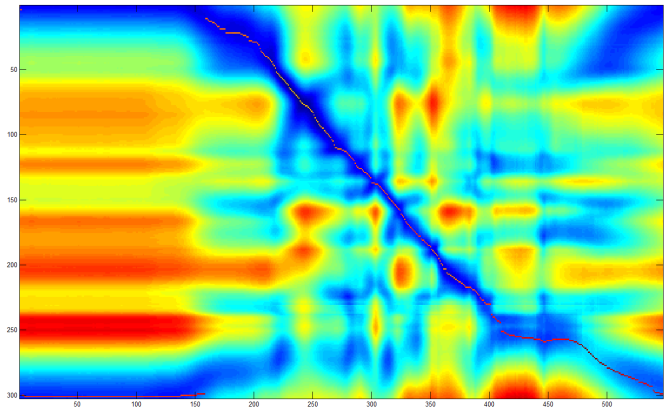


Figure 36: Path found through a tornado kick. Notice how the final path follows the valleys in the difference-landscape.

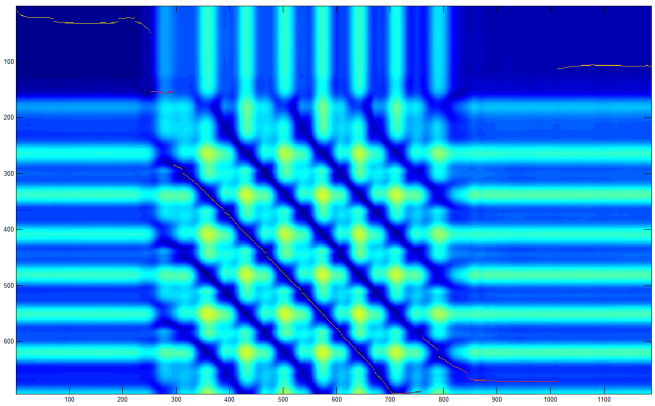


Figure 37: Path through a periodic walk. Observe the repetitive pattern of the walk. This is caused by the actor taking several steps which are all very similar. Still a good path is found.

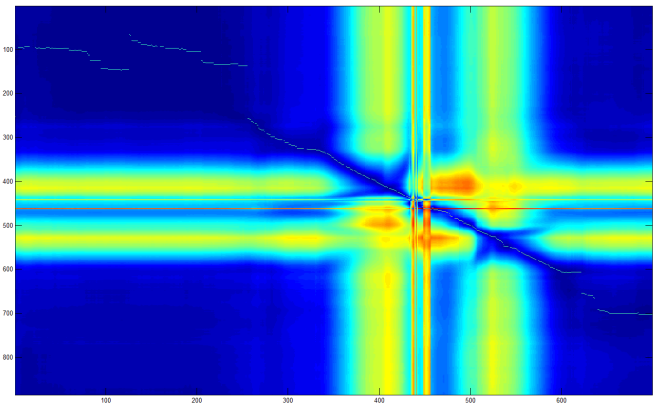


Figure 38: Path through cart wheel exercise. There are data errors, as seen from the cross pattern slightly to the right of the center. Still we see that the path does not diverge from the course.

izontal valley in the difference matrix which the path correctly locks on to. The two motions are not that similar, but the path still tends to follow the natural line through the difference matrix.

8.3 Robustness

The strong side of a probabilistic method is its robustness. It may guess wrong, but given the information it has, it guesses as well as can be expected.

In Figure 40 a path is found through a difference matrix which was changed drastically. The region from row 250 column 1 to row 300 column 250 was changed to have a very low pose difference. The path starts out with equal probability for row 250 to row 300 in the first column, but given our expectation that paths generally go forward in time, the path collects itself in the lower part of this low difference region as time goes on. When the artificial region stops, the path advances forward in time looking at actual neighboring pose differences, but it grows more and more confused since nothing really matches well. Eventually the uncertainty is so great that other matches back in the template is the best guess at a match and the path jumps back up (back in template time) and gets back on the actual best path match which is

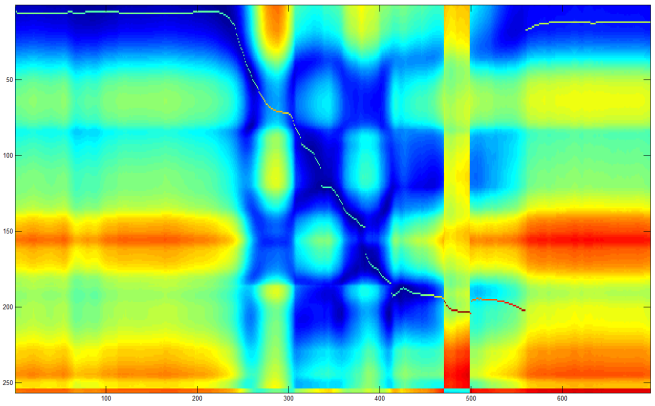


Figure 39: Path through a roll followed by a split. Observe that a path is found, but that it breaks off and jumps back in template time at the end when the template and exercise become very different. We also see data error, in form of a vertical bar slightly right of the center, but this does not affect the path much.

then followed.

We here see that a large region with corrupt data will confuse the path, but it seems to *realize* that is is confused. When the corruption eventually stops, the path doesn't take too long to get back on track. It does so by jumping back in time which is not illegal in this probability based method - only less probable than moving forward.

8.3.1 Path resulting from different state velocities

We compared paths resulting from different expectations of movement over states when building the transition matrix. As expected, the difference between expecting a state velocity of 0 and 1 and 2 was not large, but it was there none the less. We see figures of he three versions in Figure 44, Figure 45 and Figure 46. The difference is small, but looking carefully at critical sections, we notice that the state velocity of 1 hits the right path slightly better than with velocity 0 or 2. The difference is small, but in the first part of the path where the it turns downwards, it is still clear to see that a velocity of 1 hits the center of the difference valley

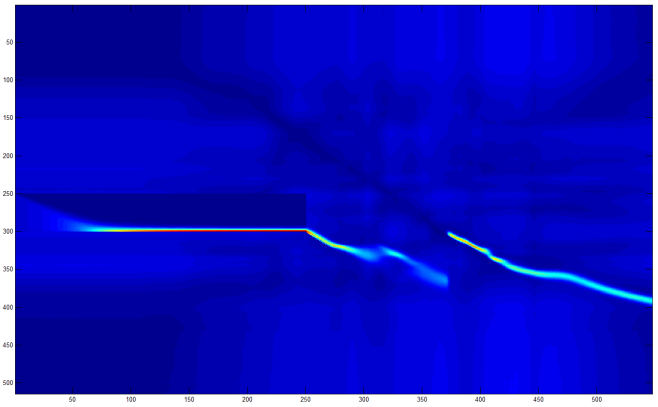


Figure 40: A difference matrix has artificially modified values (darker rectangle at left center), but still a good pat is found. The difference values were scaled down for visualization so the path probabilities would be more clear.

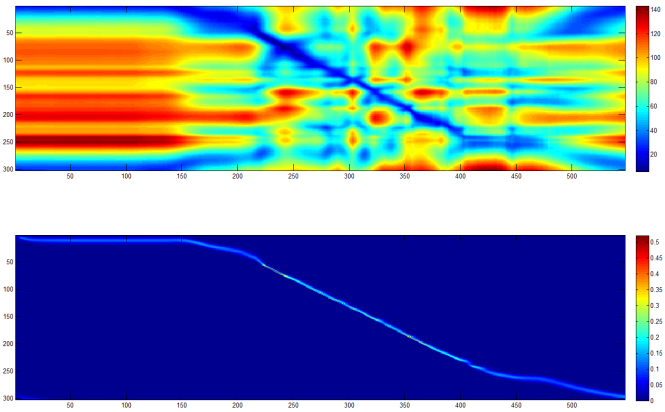


Figure 41: The path is seen to start in a blurred form since there are no particularly good frame matches early on. As it then enters a region with a clear match it contracts to forn a distinct line with high probability. As the exercise ends, the strong match is again lost and the path again becomes blurred.

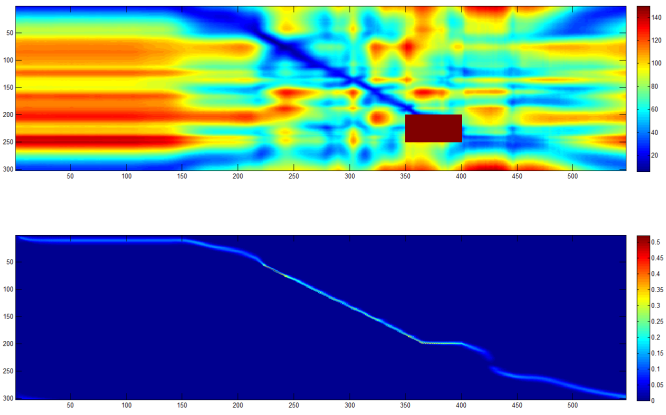


Figure 42: A HMM based path through sabotaged difference matrix. The path is seen to hit the sabotaged area on the top and then bounce along the top trying both to move down right and to stay out of the block of low probability match

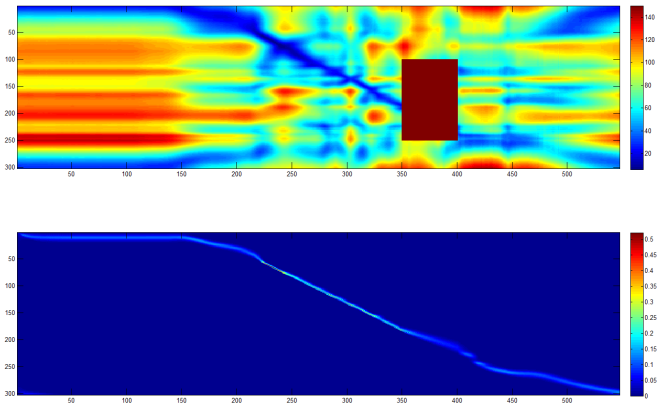


Figure 43: A HMM based path through sabotaged difference matrix. The path is seen to hit the sabotaged area on the side. Being unlikely to move back in time, it takes the direct path through the area, but the probability is seen to diffuse outwards while moving down right since there are no particularly good frame matches to keep it compact

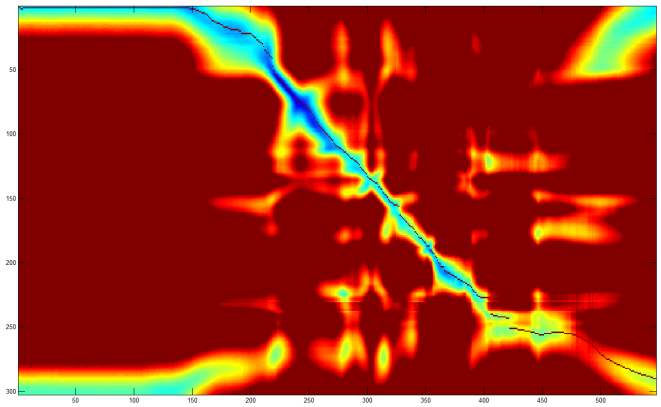


Figure 44: Path with state velocity 0. We see that the path tends to turn too slowly and hit the left (upper) side of the valley a little bit.

better. This is generally true over the entire length of the path.

As another test, we calculate the sum of all element in the element wise multiplication of the path (matrix with 1 at path and 0 elsewhere) and the difference matrix. If the path is successful in finding the best matches, the difference sum should be low -assuming there are no serious data errors in the comparisons.

Using the same example as we already have seen in this section, we find the sums to be as follows. It should be noted that the best path is not *required* to go through lower differences. If that was the case, we could simply have made the path seek out the lowest differences. It is however reassuring that using the same method, but having different state velocity expectations, will show a lowest cost path using our actual expectation of state velocity equal to 1.

Pose difference along path	
Forward shift	Sum of path difference * 1000
0	14.65
1	12.06
2	19.25
5	63.23
10	64.44

8.4 Action correction

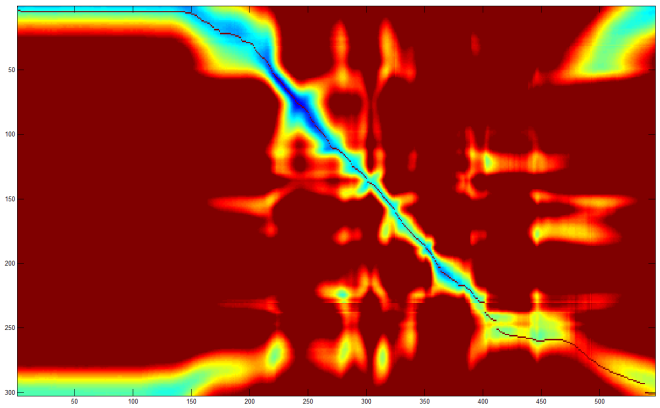


Figure 45: Path with state velocity 1. Observe how this path seems to better center in the valley than the path with lower and higher velocities.

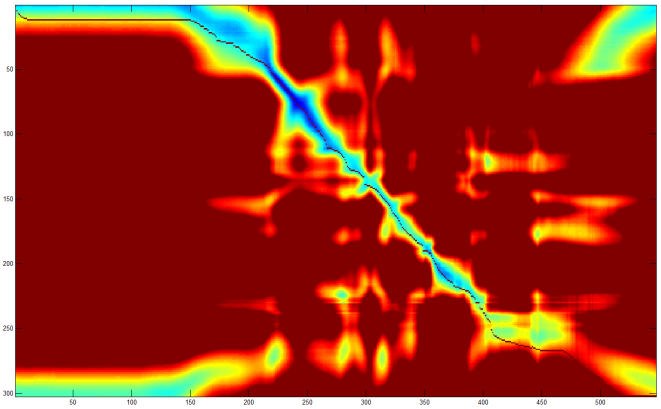


Figure 46: Path with state velocity 2. Notice how the path turns too much to the right (down) relative to the valley center.

The initial pose, which we compare with, is seen in Figure 47. This is the pose which represents the ”correct” form while the other poses are large ”errors”. Note that this template is close to the initial skeleton configuration, so most angles in the template are zero. We make comparisons only on the joint(s) that we know have flexed. The other joints, in these examples, have no difference and need not be compared. In an actual automated comparison, all joints should be compared and when one or more joints have a large enough difference, we will investigate what that difference is and how it should be corrected.

As mentioned previously, we do not have access to live MOCAP recordings and instead we have to base our testing on pre-recorded data. This means that we cannot currently demonstrate a person performing an exercise, being corrected and then continuing the exercise a more correct way. We therefore demonstrate the comparison and feedback on poses which are somewhat similar, but not quite.

We do not demonstrate the feedback method on a time warp aligned pair of action since it will generally be hard to clearly tell if the feedback is valid or not, when the errors are small. Using more different poses will let us show that the method is actually working. We do not compare how the body is translated and rotated in spatial space, but only how the joints outside the root moves. It is irrelevant where the body is located in spatial space and in our illustrations we rotate the body to best show the differences in pose.

In the following three examples, we use $T = [rx, ry, rz]$ to describe the correct template Euler angles of the relevant joint(s) while $E = [rx, ry, rz]$ is the corresponding exercise Euler angles for the joint(s). Δ is the angle that the joint should be rotated to correct the faulty pose. We base the feedback on superficial observation of the skeleton definition in section C due to the reasons explained in section 7.

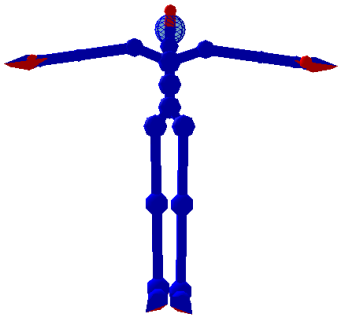


Figure 47: Natural pose used for comparison

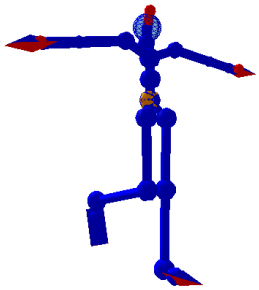


Figure 49: Flexing right knee

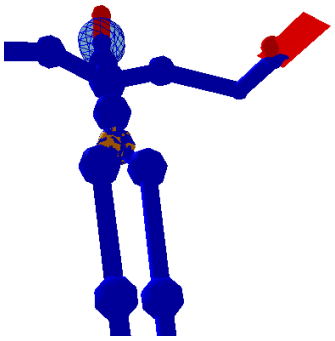


Figure 48: Flexing left elbow

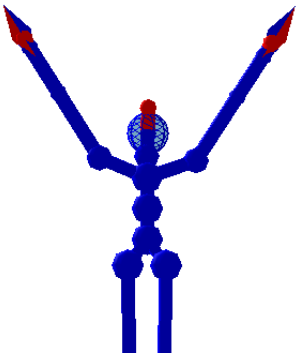


Figure 50: Raising both arms

8.4.1 Flexing left elbow

$T = [0.0, 0.0, 0.0]$
$E = [0.2, -73.3, 1.1]$
$\Delta = [-0.2, 73.3, -1.1]$

Observing the skeleton definition, we see that a positive y-axis rotation for the left arm will stretch the arm, so the user feedback should be ”Stretch your left arm 73 degrees”.

8.4.2 Flexing right knee

$T = [0, 0, 0]$
$E = [77.10, -12.90, -17.10]$
$\Delta = [-77.10, 12.90, 17.10]$

Based on the skeleton definition, we see that a bending of either knee is primarily[3] done by a positive rotation around the x-axis. This means that the feedback should be ”stretch right knee 77 degrees”.

8.4.3 Raising both arms

$T_{right} = [0.0, 0.0, 8.0]$
$T_{left} = [0.0, 0.0, -8.0]$
$E_{right} = [0.2, 1.3, 57.3]$
$E_{left} = [-0.1, 0.3, -56.8]$
$\Delta_{right} = [-0.2, -1.3, -49.3]$
$\Delta_{left} = [0.1, -0.3, 48.8]$

We observe in the initial skeleton definition that a positive rotation around the z-axis will raise the left arm to the side and a negative rotation will raise the right arm. This means that the feedback will now be ”Lower right arm 49 degrees and lower left arm 48 degrees”.

[3] As seen in section 5.1 the axis of rotation does not perfectly match the x-axis.

9 Conclusion and future work

The system can find reasonable interpretations of what might have been the intent of the MOCAP actor in relation to a template exercise. Based on this most likely intent, it can perform detail comparisons between what was intended and what was performed.

Currently, due to the lack of relevant MOCAP data, the system will not report this difference in easily understandable terms, but rather in Euler angle differences. Having access to more configurable MOCAP data should allow the system to be easily extended to provide human understandable feedback, but the whole concept of easily understandable feedback may be a more complex problem which could be investigated in a future HCI-project.

It should be investigated how the system can better learn a transition matrix based on actual observations, rather than having a predefined Gaussian-based matrix as the one used now. While Markov model learning is a subject that has been investigated in depth, the learning part of this system may be more complicated given that there is no easy access to optimal paths to learn from.

10 Acknowledgements

We used BVHacker from `http://davedub.co.uk/bvhacke` to do basic visualization of the poses from the motion capture data.

The motion capture data used in testing was obtained from the CMU Graphics Lab Motion Capture Database available at `http://mocap.cs.cmu.edu`

References

AGGARWAL, J. K., AND CAI, Q. 1997. Human motion analysis: a review. 90–102.

BIOVISION. `http://www.cs.wisc.edu/graphics/Courses/cs-838-1999/Jeff/BVH.html`.

BISHOP, C. M. 2007. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer.

CORMEN, T. H., LEISERSON, C. E., RIVEST, R. L., AND STEIN, C. 2001. *Introduction to Algorithms*. MIT Press.

FORNEY, G. D. 1973. The viterbi algorithm. *Proceedings of the IEEE 61*, 3, 268–278.

HAUBERG, S., JENSEN, B. R., ENGELL-NØRREGÅRD, M. P., ERLEBEN, K., AND PEDERSEN, K. S. Dense marker-less three dimensional motion capture.

HECKER, C., RAABE, B., ENSLOW, R. W., DEWEESE, J., MAYNARD, J., AND VAN PROOIJEN, K. 2008. Real-time motion retargeting to highly varied user-created morphologies. In *Proceedings of ACM SIGGRAPH '08*.

HISTORY, M. A. Eadweard muybridge's photography of motion. `http://americanhistory.si.edu/muybridge`.

IZENMAN, A. J. 2008. *Modern Multivariate Statistical Techniques : Regression, Classification, and Manifold Learning*. Springer Texts in Statistics. Springer New York.

JENS MICHAEL CARSTENSEN, P. *Image analysis, vision and computer graphics*.

KITAGAWA, M., AND WINDSOR, B. 2008. *MOCAP for artists - Workflow and techniques for motion capture*. Focal Press.

KOVAR, L., AND GLEICHER, M. 2003. Flexible automatic motion blending with registration curves. In *SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 214–224.

KUIPERS, J. B. 2002. *Quaternions and Rotation Sequences: A Primer with Applications to Orbits, Aerospace and Virtual Reality*. Princeton University Press.

KURZ, T. 1994. *Stretching scientifically*. Stadion publishing.

MENACHE, A. 1999. *Understanding Motion Capture for Computer Animation and Video Games*, 1st ed. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

MÉNARDAIS, S., KULPA, R., MULTON, F., AND ARNALDI, B. 2004. Synchronization for dynamic blending of motions. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, SCA '04, 325–335.

MESSER, R. *Linear Algebra*. Harper Collins College Publishers.

MOESLUND, T. B., HILTON, A., AND KRÜGER, V. 2006. A survey of advances in vision-based human motion capture and analysis. *Comput. Vis. Image Underst. 104* (November), 90–126.

PRECINEMAHISTORY. The history of the discovery of cinematography. `http://www.precinemahistory.net`.

RUSSELL, S. J., AND NORVIG, P. 2003. *Artificial Intelligence: A Modern Approach*. Pearson Education.

TIMES, N. Y. Tracking an athletes moves. http://www.nytimes.com/interactive/2010/10/03/sports/20101003-mocap.html.

WAI, A. A. P., BISWAS, J., FOOK, F. S., KENNETH, L. J., PANDA, S. K., AND YAP, P. 2010. Development of holistic physical therapy management system using multimodal sensor network. In *Proceedings of the 3rd International Conference on PErvasive Technologies Related to Assistive Environments*, ACM, New York, NY, USA, PETRA '10.

WANG, L. 2003. Recent developments in human motion analysis. *Pattern Recognition 36*, 3 (March), 585–601.

WEINLAND, D., BOYER, E., AND RONFARD, R., 2007. Action recognition from arbitrary views using 3d exemplars. `http://perception.inrialpes.fr/Publications/2007/WBR07`.

WELTON, N. J., AND ADES, A. E. 2005. Estimation of Markov Chain Transition Probabilities and Rates from Fully and Partially Observed Data: Uncertainty Propagation, Evidence Synthesis, and Model Calibration. *Medical Decision Making 25*, 6, 633–645.

WIKIPEDIA. The viterbi algorithm. `http://en.wikipedia.org/wiki/Viterbi_algorithm`.

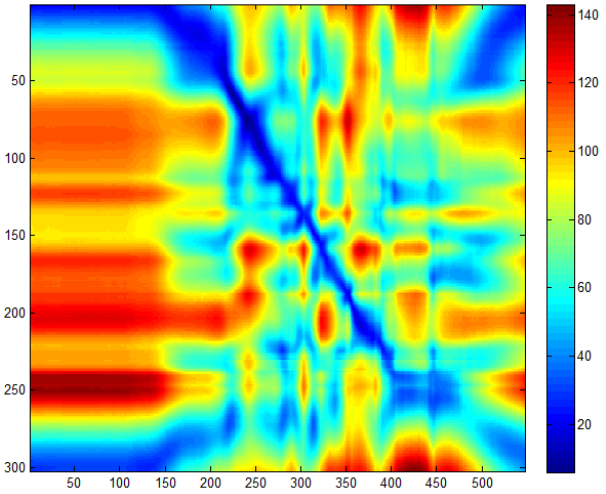


Figure 51: Difference matrix

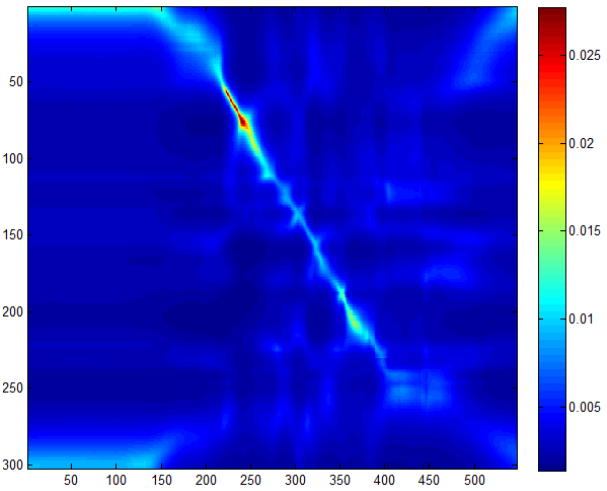


Figure 52: Evidence matrix

A Example difference, evidence and path probability matrices

We show the steps from initial difference calculation to final path. The difference matrix and the derived evidence matrix as well as the probability matrix of the path and the final binary path are shown.

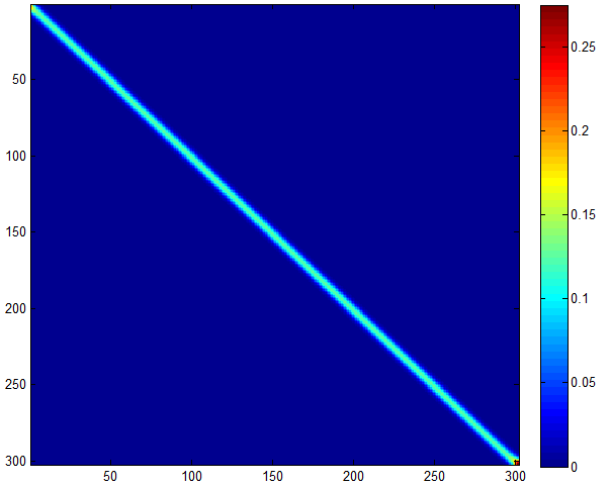


Figure 53: transition matrix

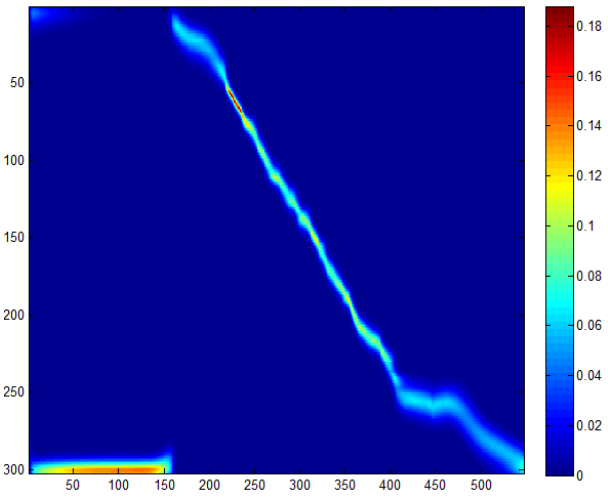


Figure 54: Probability matrix

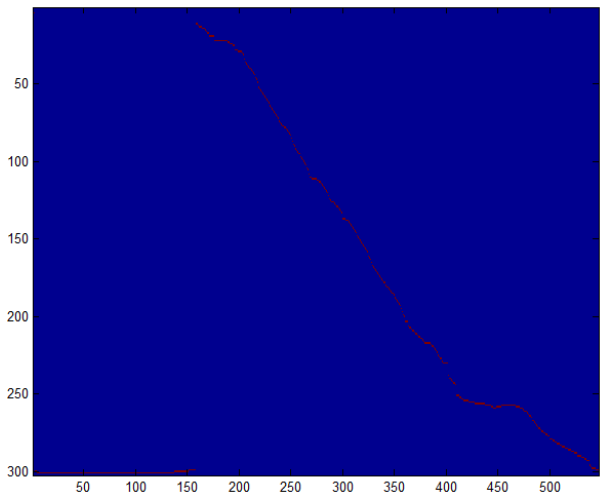


Figure 55: Binary path

B Matlab calculation of path

The final Matlab code calculating the path is shown below. We do not show the support functions in this paper, but from our description of the choices made, the logic should be easy to follow.

```
%read in the template and the exercise
templateAng = bvhFrames('e:\template.bvh');
unknownAng = bvhFrames('e:\exercise.bvh');

%create the transition matrix
T=transitionMatrix(size(templateAng,1),10,1);

%calculate difference matrix
differenceMatrix = bvhDifference(templateAng, unknownAng);

%calculate evidence matrix
E_matrix = ones(size(differenceMatrix))./differenceMatrix;

%column normalize difference matrix for easier evidence
E_matrix = normalizeColumns(E_matrix);

%define prior probability based only on initial evidence
PX0 = E_matrix(:,1);
PX_old=PX0;

%define matrix holding total probabilities and write prior probabilities
P = zeros(size(templateAng,1),size(unknownAng,1));
P(:,1)=PX0;

%run through all exercise frames, one by one, and calculate path probability
for t=2:size(unknownAng,1)
    PX_new = T*PX_old;
    PX_new = PX_new.*E_matrix(:,t);
    PX_new = normalizeColumns(PX_new);
    P(:,t)=PX_new;
    PX_old=PX_new;
end;

%define empty path
path = zeros(size(P));

%define path to be maximal likelyhood for each exercide frame
for c=1:size(P,2)
    r = find(P(:,c)==max(P(:,c)));
    path(r,c)=1;
end;
```

C BioVision Hierachical skeleton

```
HIERARCHY
ROOT Hips
{
    OFFSET 0.00000 0.00000 0.00000
    CHANNELS 6 Xposition Yposition Zposition Zrotation Yrotation Xrotation
    JOINT LHipJoint
    {
        OFFSET 0 0 0
        CHANNELS 3 Zrotation Yrotation Xrotation
        JOINT LeftUpLeg
        {
            OFFSET 1.28858 -1.83292 1.03970
            CHANNELS 3 Zrotation Yrotation Xrotation
            JOINT LeftLeg
            {
                OFFSET 2.47846 -6.80952 0.00000
                CHANNELS 3 Zrotation Yrotation Xrotation
                JOINT LeftFoot
                {
                    OFFSET 2.75274 -7.56311 0.00000
                    CHANNELS 3 Zrotation Yrotation Xrotation
                    JOINT LeftToeBase
                    {
                        OFFSET 0.12848 -0.35301 1.68223
                        CHANNELS 3 Zrotation Yrotation Xrotation
                        End Site
                        {
                            OFFSET 0.00000 -0.00000 0.87311
                        }
                    }
                }
            }
        }
    }
    JOINT RHipJoint
    {
        OFFSET 0 0 0
        CHANNELS 3 Zrotation Yrotation Xrotation
        JOINT RightUpLeg
        {
            OFFSET -1.31183 -1.83292 1.03970
            CHANNELS 3 Zrotation Yrotation Xrotation
            JOINT RightLeg
```

```
        {
            OFFSET -2.44740 -6.72417 0.00000
            CHANNELS 3 Zrotation Yrotation Xrotation
            JOINT RightFoot
            {
                OFFSET -2.67343 -7.34519 0.00000
                CHANNELS 3 Zrotation Yrotation Xrotation
                JOINT RightToeBase
                {
                    OFFSET -0.16914 -0.46471 1.77626
                    CHANNELS 3 Zrotation Yrotation Xrotation
                    End Site
                    {
                        OFFSET -0.00000 -0.00000 0.92777
                    }
                }
            }
        }
    }
}
JOINT LowerBack
{
    OFFSET 0 0 0
    CHANNELS 3 Zrotation Yrotation Xrotation
    JOINT Spine
    {
        OFFSET 0.01249 2.14252 0.03842
        CHANNELS 3 Zrotation Yrotation Xrotation
        JOINT Spine1
        {
            OFFSET -0.02957 2.14199 -0.03751
            CHANNELS 3 Zrotation Yrotation Xrotation
            JOINT Neck
            {
                OFFSET 0 0 0
                CHANNELS 3 Zrotation Yrotation Xrotation
                JOINT Neck1
                {
                    OFFSET -0.01678 1.65709 -0.05476
                    CHANNELS 3 Zrotation Yrotation Xrotation
                    JOINT Head
                    {
                        OFFSET 0.07912 1.58331 -0.11843
                        CHANNELS 3 Zrotation Yrotation Xrotation
                        End Site
                        {
```

```
                OFFSET 0.03298 1.73940 -0.04271
            }
        }
    }
}
JOINT LeftShoulder
{
    OFFSET 0 0 0
    CHANNELS 3 Zrotation Yrotation Xrotation
    JOINT LeftArm
    {
        OFFSET 3.39556 1.16496 0.08673
        CHANNELS 3 Zrotation Yrotation Xrotation
        JOINT LeftForeArm
        {
            OFFSET 5.23910 -0.00000 -0.00000
            CHANNELS 3 Zrotation Yrotation Xrotation
            JOINT LeftHand
            {
                OFFSET 3.31467 -0.00000 0.00000
                CHANNELS 3 Zrotation Yrotation Xrotation
                JOINT LeftFingerBase
                {
                    OFFSET 0 0 0
                    CHANNELS 3 Zrotation Yrotation Xrotation
                    JOINT LeftHandIndex1
                    {
                        OFFSET 0.96590 -0.00000 0.00000
                        CHANNELS 3 Zrotation Yrotation Xrotation
                        End Site
                        {
                            OFFSET 0.77873 -0.00000 0.00000
                        }
                    }
                }
                JOINT LThumb
                {
                    OFFSET 0 0 0
                    CHANNELS 3 Zrotation Yrotation Xrotation
                    End Site
                    {
                        OFFSET 0.79062 -0.00000 0.79062
                    }
                }
            }
        }
```

```
            }
        }
        JOINT RightShoulder
        {
            OFFSET 0 0 0
            CHANNELS 3 Zrotation Yrotation Xrotation
            JOINT RightArm
            {
                OFFSET -3.27193 1.31331 0.33940
                CHANNELS 3 Zrotation Yrotation Xrotation
                JOINT RightForeArm
                {
                    OFFSET -5.30252 -0.00000 0.00000
                    CHANNELS 3 Zrotation Yrotation Xrotation
                    JOINT RightHand
                    {
                        OFFSET -3.67875 -0.00000 0.00000
                        CHANNELS 3 Zrotation Yrotation Xrotation
                        JOINT RightFingerBase
                        {
                            OFFSET 0 0 0
                            CHANNELS 3 Zrotation Yrotation Xrotation
                            JOINT RightHandIndex1
                            {
                                OFFSET -0.59481 -0.00000 0.00000
                                CHANNELS 3 Zrotation Yrotation Xrotation
                                End Site
                                {
                                    OFFSET -0.47955 -0.00000 0.00000
                                }
                            }
                        }
                        JOINT RThumb
                        {
                            OFFSET 0 0 0
                            CHANNELS 3 Zrotation Yrotation Xrotation
                            End Site
                            {
                                OFFSET -0.48687 -0.00000 0.48687
                            }
                        }
                    }
                }
            }
        }
    }
```

```
    }
}
}
```