VISULASATION OF BACK MOVEMENT USING THE MOTEK D-FLOW SOFTWARE AND THE VICON MOTION CAPTURE SYSTEM

DAVID JOHN HENRY ALLAN

This thesis is submitted in partial fulfilment of the requirements for the Degree of MSc in Biomedical Engineering.

Bioengineering Unit
University of Strathclyde
Glasgow

October 2011

Declaration of Authenticity and Author's Rights

‘This thesis is the result of the author’s original research. It has been composed by the author and has not been previously submitted for examination which has led to the award of a degree.’

Signed: Date:

Acknowledgements

I would like to take this time to first thank Prof. Philip J. Rowe for his time and support throughout the course of this MSc project.

I would also like to thank Lindsay Millar for her continued support and always being there if help is needed wither or not she could help she has provided an insight or an idea that has made this project be a lot easier that it could have been.

Additionally Gwenllian Ffulr deserves a spot with her help in getting all the equipment setup in the beginning and showing how to get all the cameras talking to each other and how to use the software which was not as easy as one may think.

Also to John McMenemy for providing the brief and inspiration for the development of this application.

And finally I would like to thank all the Biomedical engineering family and staff for all their ongoing support and help throughout this last year. Without them this year would not have been nearly as fun or gone as smoothly as it has and to them I have the greatest thanks of all to give.

Special mention goes out to my incredibly bad taste in music as well as Red Bull as without it I wouldn't have got this done.

Abstract

Background: Low back pain (LBP) is one of the world's most common medical conditions with an estimated 60% of all adults experiencing it within their lifetime. Those that go on to develop chronic LBP start to develop guarded movements and a negative outlook so motion capture technology is being looked at as a way to promote better unguarded movement within this group of suffers.

Aims: The aim of this project is to create an application that uses motion capture technology that can be used in the treatment of LBP in a clinical setting that requires little to no train to operate whereby it provides visual feedback to the patient based on their position by comparing their current position to that of a recording.

Methods: The application was developed in Motek D-Flow, Motek Medical, Amsterdam Netherlands, in conjunction with a 8-camera VICON system, VICON, Oxford, United Kingdom. This was achieved using a series of scripting modules written in Lua to recognise a set of 5 clusters each with 4 markers placed on key anatomical structures on both the spine and legs and then compare that to a previous recording by comparing the distances between the clusters in both the live and recorded situations.

Results: The results show that it was possible to create an application where by it compares the differences between a live and a recorded position and is able to provide visual feedback based off the outcome with a traffic light based system to show how many of the clusters are in the correct place.

Table of Contents

List of Tables

List of Figures

Chapter 1 –Introduction and Clinical Rational

1.1 Introduction

Low back pain (LBP) is one of the most common and frequent medical reasons for absences from work whereby at least 60% of adults will experience some form of LBP within their lifetime (Walsh et al 1992) with an estimated 2.8million working days being lost in the UK because of it (HSE 2013/2014)[1]. Its estimated that 80%-90% of those that experience episodes of LBP will have recovered within six weeks (Waddell et al 1987) with an average recovery time of 12.3 days with those working in environments which contain a large volume of manual handling being most at risk e.g. construction, agriculture, postal services (MacFarlane et al 1997). With men between 34-44 presenting the highest risk (HSE 2013/2014)[1].of those that do suffer from LBP a large number still suffer and are affected for long periods of time (Croft&Dunn, 2004).

1.1.1 Mechanics of injury

A clinical review by Adams et al 2004 whereby they extensively reviewed nineteen other studies looking at the mechanics of injury in the lumber spine identified four key areas in which cause injury.

1. Compression
2. Bending
3. Axial torsion
4. Bending and compression

From the literature the common cause of these four mechanics is in that most of them occur as a result of repeated movements and loading in the case of compression this has been classified by micro-fractures and healing trabeculae being found in most cadaveric vertebral bodies. Bending works in a similar manner with people moving from full extension to flexion this causes the muscles that limit back movement to lose their ability to protect the back by losing their protective reflex of which this leads to damages in the apophyseal joints or the joint capsules.

Axial rotation can cause the vertebral ligaments be substantially stretched and cause damage to the intervertebral discs.

[1] http://www.hse.gov.uk/statistics/causdis/musculoskeletal/msd.pdf

If bending and compression are combined at the same time such as when somebody lifts something heavy such as weight this is what causes prolapse in the discs of which this can happen over only one loading cycle where by with the bending moment or the compressive force exceed their normal limits but this is enough to cause damage to the spine. Those that continue to apply the load in this same way can result in expulsion of the nucleus pulposus.

This is just a very short description of some of the mechanisms that can cause back pain but there are so many different mechanisms in play when it comes to looking at the spine that trying to quantify all of the failure mechanisms that may occur this will take years to accomplish and the chances at understanding all the mechanisms is small as the spine is such a complicated structure

1.1.2 Treatment of low back pain

One of the issues that occurs when looking at lower back pain is that due to the complexity of the spine there are many potential different possibilities that could be causing the pain and so as such it can be difficult to choose the correct course of treatment of which there are many. These range from the non-invasive physiotherapy, osteopaths, manual therapy to the more invasive spinal surgery (van Middelkoop et al 2010). The most common of these treatments is that of exercise therapy where by the patient is set a set of movements that is designed to restore normal musculoskeletal function Haden et al 2005 shows that for those that suffer from chronic LBP (pain >12weeks) that general exercise therapy is slightly effective at decreasing pain and whilst also increasing their general function whilst those suffering from acute back pain experienced no difference in pain levels when preforming the exercises. One of the other main treatments of LBP is that of spinal manipulative therapy (SMT) but there is evidence to support that this method has a small short-term effect on pain when compared to other methods (Rubinstein et al 2011, van Middelkoop et al 2010) other standard methods of treatment and prevention include core strengthening to stabilize the spine as well as massage therapy to loosen muscles. There are also pharmacological approaches using painkillers, opiates, non-steroid anti-inflammatory (NSAIDs) as well as antidepressants.

The current guidelines (Koes et al 2001, Bouwmeester et al 2009) that are proposed to people suffering from acute LBP are to maintain active, avoid bed rest and if need be prescribed with paracetamol/acetaminophen as well as non-steroid anti-inflammatory.

There has also been a rise in the number of cases which people are presented with chronic LBP where by it has been noted that those with symptoms of chronic LBP have a negative relationship with pain (pain catastrophizing) as well as a fear of re/injury (kinesiophobia) (Picavet et al 2002). This fear of moving and causing pain lead to avoidance of movements that may produce pain which leads to the withdrawal from work and leisure and causes hypervigilance, disuse and even depression. With the disuse leading to the weakening of the muscles that will lead to more serious problems as such when dealing with patients with chronic LBP it is important not only to treat the cause but to also treat the psychological issues that occur as well.

1.2 Literature Review

1.2.1 Introduction

With the spine being such as complicated structure it can be difficult to find the root of a patients back pain with it sometimes being best to gain an idea as to how the spine is currently moving as well as its range of motion. This can be measured using a variety of methods for example the three main methods of back pain diagnosis are X-Ray, CT, MRI but other methods can be used as well. In this section each of these methods will be discussed as to how they could be used in the visualisation of the back, if at all and how these can be used in rehabilitation as a way of providing visual feedback.

1.2.2 X-Ray

The X-Ray is arguably one of the oldest methods of measuring the spine as well as one of the most used methods in terms of diagnosis whereby images in the anteroposterior and lateral views allow clinicians to detect any changes in spinal shape, alignment, disc and veritable body height and if there is any degeneration of the spine in terms of bone density and architecture (Jarvik et al 2002). One of the issues with X-Rays it that they can’t see soft tissue as it is absorbs the X-Rays.

Flexion- extension X-Rays are a commonly used method to identify if there are problems caused by abnormalities in the intervertebral discs (Taylor et al 2007). Even though it is one of the most common methods of viewing the these abnormalities there are limiting factors that can lead to misinterpretation due to varying quality of the radiographs, reproduability, missing measurement standards and differences in clinical practice as to how to diagnose (Pitkänen et al 1994, Taylor et al 2007) and as such computer assisted technology has been developed to help reduce the amount of error between clinicians.

In order to be able to determine the range of motion (ROM) of the spine what is known as functional X-Rays must be taken where by a pair of X-Rays are taken at two maximum end positions e.g. full flexion and full extension (Schulze et al 2011). New software has been developed where by it takes these functional X-Rays and calculates the ROM of the spine which has proved to be an accurate way of predicting overall spinal ROM (Schulze et al 2011).

This technology could be applied to a model of the spine to show what would happen when a certain movement is preformed and therefore help to create a visualisation so that a patient can see what is happening to their spine but as a tool for rehabilitation by providing visual feedback using this technology would just take too long as the patient would have to go on a waiting list in order to gain the images needed to recreate the model by this time the pain that they are experiencing may have passed and rehabilitation is no longer needed.

1.2.3 CT Scan

Computed tomography or CT is another way in which the spine can be imaged by using X-Rays to generate cross sectional images of the spine in the axial direction but these can be reformed to look at the sagittal and coronal planes. In order to image the spine cross sectional cuts are made about ~3-5mm in thickness. Unlike the X-Ray which can only see bone CT scans can image herniated disks, muscles, tissue, tendons, ligaments and blood vessels[2].

One of the uses of CT scanning is in the 3D reconstruction of scoliotic spines Pomero et al 2004 shows that it is possible to recreate the spinal structure of a scoliotic spine by using a

[2] http://www.mayoclinic.org/diseases-conditions/back-pain/basics/tests-diagnosis/con-20020797

new 3D reconstruction method in comparison to the traditional method where by lots of little cuts ~1mm where made which would subject patients to a large radiation dosage.

By taking this technique and applying it this could have its applications in a rehabilitation sense where by the complete 3D model could be generated and input into a motion capture programme whereby patients movements would then be translated onto a screen so that they could see exactly what their spine is doing when moving though this method does also suffer from the same problems as the X-Rays where by waiting times to get appointment can be very long especially in a none emergency case so perhaps a generalised model could be created to substitute.

1.2.4 MRI

Magnetic resonance imaging (MRI) provides high–resolution imaging but because of the orientation that a patient is required to be in this alters the spinal orientation of the patient. In this case most MRI machines require a patient to be lying supine inside of a bore this causes a reorientation of gravity acting on the spin and as many LBP patients suffer little to no pain when lying down this poses a problem in accurately imaging them as the lower lordosis has changed position and this may be one of the areas of which is causing the pain. To combat this some studies have applied axial compressive loads (Kimura et al 2001, Wisleder et al 2001).

There is however other MRI machines which are called Open MRI which allow a patient to be sitting or standing when within this poses a better opportunity to look at the spine when it is in its natural orientation in comparison to forcing it with compressive force which may cause more damage to a patient.

Simons et al 2013 have developed a vertebral reconstruction method whereby they are able to quickly and accurately remodel the spine by using an open MRI so as to best be able to evaluate the lumbar spine when in its natural orientation and allow a greater deal of movement and no radiation.

This as a potentially greater use in rehabilitation as it will allow a patient to be placed into a position the unloads the area that cause pain and a comparison can be made between the two positions and as the patient won't suffer radiation a greater amount of treatment could be administered. In terms of visual feedback the same process that is outlined above where by the images could be imported into a motion capture program and used to show real time

what the spine is doing. The downside to this is MRI machines are expensive to purchase and so there is a limited number of them that are available and so will reduce the number of patient that will have access to them normally and additionally due to their limited number the waiting list again can be quite long in order to use them.

1.2.5 Summary

In summary there are three main diagnostic techniques that are used when it comes to imaging the spine but they are all subject to their draw backs with the main one being that of waiting times. So an easy way in which visualisation of spinal movement must be found that doesn't rely on these large pieces of machinery so as to make it more accessible for rehabilitation.

1.3 Clinical Rational

One of the most important thing in treating low back pain is in the timing of treatment; even though 80%-90% of LBP patients are acute (<6 weeks) it's the remaining 10%-20% where the main issues in terms of dealing with the pain occurs. As a person with LBP continues to suffer they begin to move into the sub-acute low back pain category (6-12 weeks) and this is where the guarding movements start to develop as well as the negative outlook in terms of dealing with their pain, a result this is where you want to treat people to try and prevent them from becoming chronic LBP suffers an at the same time so that they don't develop these bad guarding movements that have detrimental effects later on. In speaking with physiotherapist John McMenemy, one issue he raised is that people suffering from LBP tend to be lacking the self-motivation to continue with physiotherapy and so as a result rehabilitation takes longer than it should he also raised the issue that they have no body awareness and so do not preform the exercises correctly. One of the ways that has been looked at is in the using of motion capture technology to be able to try and treat those that currently suffer from chronic LBP by increasing their activity levels and using visual feedback to ensure that they are preforming the movements correctly to try and break their habits of guarded moving.

1.2.1 Motion Capture approach to treating chronic pain

Within the last decade there has been a rapid increase in the number of gaming systems that are using motion capture technology such as the Xbox, Microsoft, WA, USA, PlayStation,

Sony, Minato, Japan as well as the Nintendo Wii, Nintendo, Kyoto, Japan with each possessing their own unique way of providing feedback with the two biggest being the Xbox Kinect, Microsoft, WA, USA and the Nintendo Wii balance board, Nintendo, Kyoto, Japan.

These systems were first made to promote gamers to become more active by exercising through these games and so the term "exergame" has been coined to describe this genre, examples of which include EA Sport Active II, EA Sports, 2010, and Wii Fit, Nintendo, 2007. As such there has been a growth of research looking into how these games can be used clinically (Kharrazi et al 2012). One of the ways in which these games have been adapted into a clinical use is in rehabilitation of people with chronic low back pain as it has been proven that those that maintain their activity levels have reduced levels of pain (Aung et al) but as discussed above those that suffer from chronic LBP also have psychological issues that go along with the pain and lead to guarded movement to try and prevent said pain. An attempt to solve this is being conducted by the University College London (UCL) called the Emo-Pain project[3], which is being developed as a multi-faceted virtual coaching system of chronic LBP rehabilitation this works through facial recognition, EMG and motion capture to detect their emotional state and muscle activities to see if they can detect when a person is in pain.

Another study by Jansen-Kosterink et al 2013 also looks at trying to treat people with chronic low back pain this time using the "PlayMancer" project (FP7-ICT-215839-2007) where by 10 participant with either neck/shoulder or low back pain were asked to participate in 4 gaming sessions whereby they would play 3 minigames each with their own goal as to help with rehabilitation. The games were controlled using motion capture using a 36 marker reflective suit and eight infrared cameras (IOtracker). The overall results for this were good with the participants finding the games motivating and providing a distraction from their pain that allowed them to better execute the exercises that the minigames were asking. This shows that there is an opportunity to use these games for rehabilitation purposes but one of the problems is that both of these two studies must be completed in a lab of which very few people suffering will have access to and as such games that are usable at home but provide the same good responses and outcomes need to be developed.

[3] www.emo-pain.ac.uk

Studies have been conducted where by the Microsoft Kinect cameras have been compared to that of the VICON camera systems, VICON, Oxford, United Kingdom (Bailey et al 2012, Galna et al 2014, and Fern'ndez-Baena et al 2012). These studies show that there is a clinical use for the Kinect but one of the issues that was raised by Bailey et al was that there is a great deal of noise involved with the Kinect cameras due to depth mapping used in how it finds objects within its capture volume and when compared to that of the VICON is of obvious poorer quality though they are trying the use of Butterworth filters too smooth out the images so as to make them of more comparable quality. This is additionally backed up in Galna whereby they show that the Kinect lacks the ability to accurately detect smaller movements and is best suited for larger movements.

1.4 Summary

Through the limited literature found where by these exergames have been used in the treatment of chronic LBP they have shown that there is a great deal of scope of which theses can be used in but as it stands there are still great limitations in their use with either patients having to go to specialist departments to be participants in studies or the technology doesn't have the same accuracy as these specialist departments has. So the proposed project is to develop an application using motion capture technology, to help in the rehabilitation of LBP patients, using a portable motion capture system similar to those used in these specialist labs but can be used by physiotherapists as a first line of treatment to try and prevent these patients turning chronic.

Chapter 2 – Methodology

2.1 Introduction

The concept of this project came from a meeting with John McMenemy, Physiotherapist, and was to develop an application using Motek D-Flow, Motek Medical, Netherlands, written using the coding language Lua, which is easy to use by physiotherapists that requires little to no training and helps patients to increase body awareness though the use of motion capture with the initial concept of the application allowing the physiotherapist to place a patient suffering from back pain into a position that unloads the soft tissue that is causing the pain where they take a recording of the patient in that position and using visual feedback in the form of a block avatar and target based system to get the patient themselves back into that position. Additional information such as hip angles, pelvic tilt and spinal shape were also ideally looked for.

Due to time constraints it was deemed that this was overly ambitious and as such a smaller scale bench test, proof of concept was developed which uses automatic recognition of clusters placed on the thighs, sacrum, the L1-T12 spinal level to show the mid back and a cluster on the T1-C7 spinal level for the upper back to form targets at which a patient is encouraged to hit though visual feedback based off how far they are from the unloaded position.

Within this chapter the design and development process that was applied through the timescale of this thesis project shall be discussed focusing on the each module within the application (Figure 1)

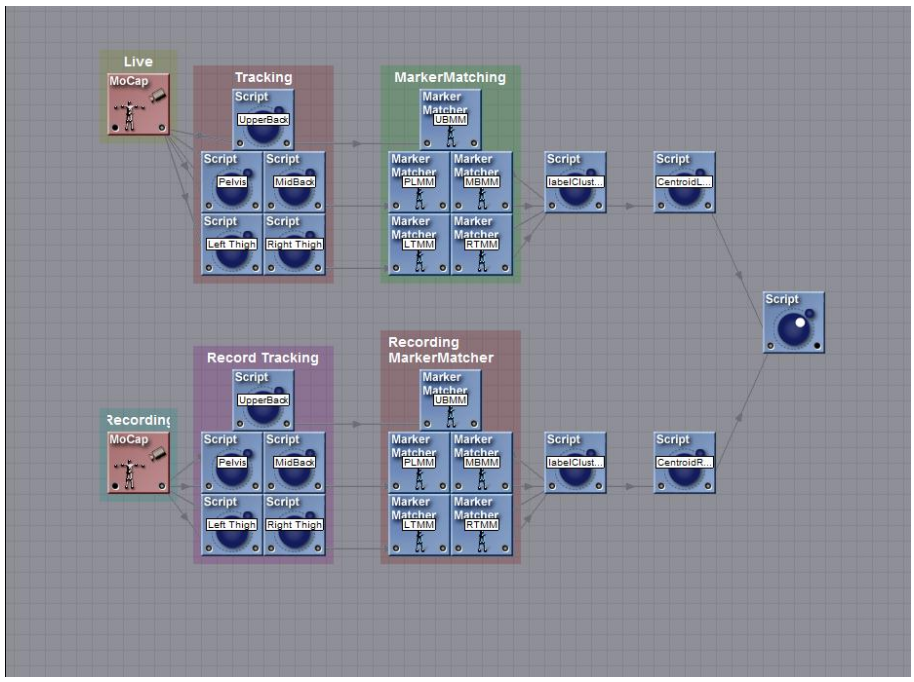


Figure 1 Screenshot from Motek D-Flow showing the completed application

Each section will include a description of the main purpose of script module as well as how the code in each section works and finally any deign issues that occurred with individual and overall limitations of the application being discussed late on in Chapter 4.

2.2 Camera Configuration and Setup

The camera system being used of this project is a VICON Boneta 10, VICON, Oxford, United Kingdom, based system consisting of 8 cameras, which runs in conjunction with VICON Nexus, VICON, Oxford, United Kingdom, to track and locate the markers in the capture volume. Cameras are mounted using mounting brackets onto a scaffolding frame as seen in Figure 2.

In the beginning of the application development process the cameras were set up as in Figure 2 with 4 cameras in the front to capture the thigh markers and 4 cameras to capture the back and pelvis markers both using the same arrangement.

Figure 2 Initial camera setup used for both the front and back arrangements

Upon testing the system and looking at the consistency of which markers were being tracked it was observed that the configuration of the cameras at the back of the participant were adequately positioned and provided good tracking in both static and dynamic situations with little to no dropout of markers. On the other hand the frontal cameras proved to be less reliable in their tracking this was due to the angle of which the cameras were looking down upon the clusters. This problem was especially observed in the cameras mounted on the top rail it was surmised at the time that they were finding it difficult to differentiate the different markers on the clusters and as such were only treating the cluster as if they had 2/3 markers each. It was not as much of a problem with those mounted on the middle rail but this still occurred. This was confirmed when looking at the individual camera views and looking at the marker outlines within Nexus.

The initial attempt at solving this problem was to reduce the number of cameras that the system needs to locate a marker position from 3 cameras down to 2 cameras therefore reducing the reliance on the top two cameras, this helped to partially solve the problem to some extent, as it reduced the amount of marker drop outs, but it did not completely eliminate the problem and so another solution had to be found.

So it was proposed that moving the 4 front cameras would help to reduce the issue and see if that made a difference in terms of decreasing dropout. The back cameras were kept in the

same positions as they worked adequately but the front facing cameras were moved into the positions shown in Figure 3

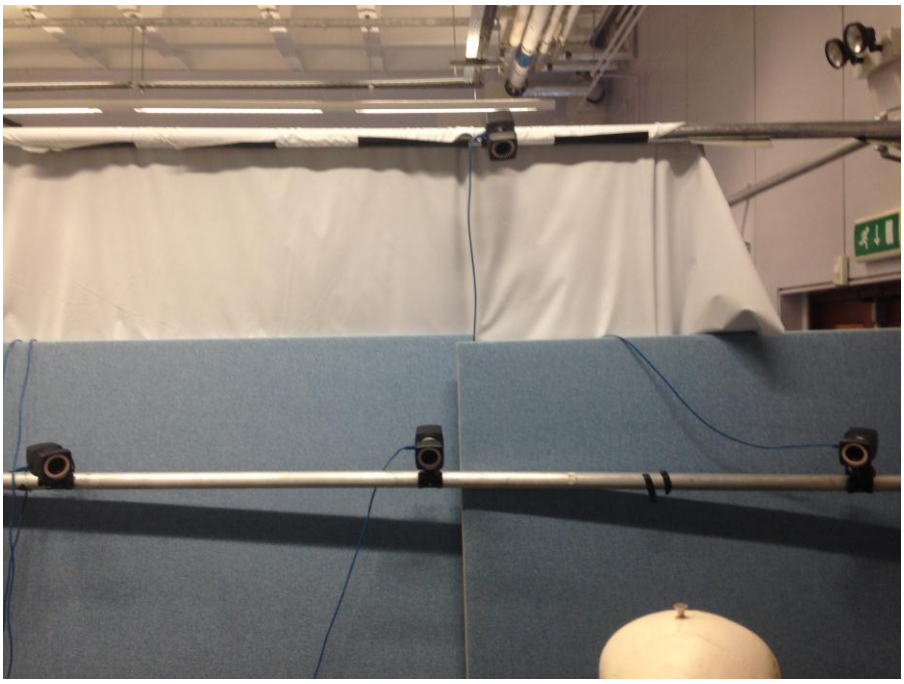


Figure 3 Modified camera setup used for the capture of the front markers

The configuration of only using 2 cameras to locate a marker position was kept the same as that worked initially and with the new camera positions the drop out of markers in the front was significantly reduced when compared to the original camera positions. This is difficult parameter to quantify as the system is not always running but from observation it was seen to be a reduction.

2.3 Cluster Generation

The five clusters that were used within the development of the application are made of a thermoplastic material; this allowed the cluster bases, of which the markers attach to, to be formed to the shape that was required to provide a comfortable fit to the participants. The thermoplastic comes in large flat sheets and this was cut using a set of sheers to make it more manageable, the clusters for the thighs, pelvis, mid back and upper back where then marked out with the dimensions, where they were then cut out using a small hack saw.

In order to be able to attach the clusters for the thighs, pelvis and mid back to the participants slots where cut into the now cut out cluster bases. The initial idea was to use elastic strapping such as in those that are used in heart rate monitors but upon investigation this proved to be both too expensive as well as difficult to source so and alternative solution was proposed. In this new method a long strip of elastic was purchased along with stick on velcro, the elastic was cut to size and then velcro was applied a full assembly can be found in Figure 4.

Figure 4 Assembly of elastic and velcro used to attach cluster to participants

This method proved to be an acceptable solution in the short term but one of the main problems that was observed was the adhesive of the velcro was not strong enough to maintain contact with the elastic and as a result this caused the clusters to fall off the participant and in some cases even fly off, but as the clusters were being used for short periods of time it was an acceptable hindrance though this did increase the amount of time taken to conduct some of the testing. An ideal solution that could be used would be something such as this Figure 5[4].

[4] http://hdsupplysolutions.com/wcsstore/ThdsMroUs/product/fm/large/12/128110_L.jpg

Figure 5 Ideal method of attaching the clusters to participants

With the thigh clusters in order to molding them to a suitable shape to fit the thighs comfortably they were placed into an oven that was set to its maximum temperature ~300C and the material was allowed to become completely soft where they were then draped over a preexisting thigh cluster and allowed to cool completely to set the new shape.

For the upper back cluster due to its location on the C7 joint (Figure 6) it was deemed unsafe to apply a band of elastic around a participants neck so the solution to this is to attach the cluster to the participants back/neck using medical grade double sided tape as to avoid irritation and potential other adverse reactions.

Figure 6 Anatomical position of the upper back cluster illustrating how using elastic to attach the cluster is not appropriate

The cluster used had a natural curvature to it and as such made a comfortable fit for a short amount of time it was used for. A potential improvement that could be made to this would to uses some sort of soft foam as to allow a more comfortable fit for a greater periods time, though this could provide problems with attaching the clusters due to them not having a solid back to adhere to and may also increase the movement of the cluster due to it wobbling caused by the extension.

The markers used for the clusters were standard 14mm markers (Figure 7) though initially smaller 5mm markers were intended to be used for the upper back and mid back clusters as to help to reduce the size and improve the accuracy of which they can be placed onto the correct spinal level.

Figure 7 14mm reflective marker

When trying these smaller markers a few problems occurred mainly a) not enough markers for both clusters and b) markers dropping out when there was too much movement.

As a solution to the first problem a combination of both 5mm and 14mm makers were used with two 5mm and two 14mm being used on each cluster it was observed though that during stationary positions that the models worked as intended with the clusters being easily identified though there were instances if the markers of different sizes where placed too close together that VICON Nexus would confuse itself and ignore the smaller maker all together. This causes the model to fail, as in order to create the cluster 4 markers are needed. As a result of this it was decided that only 5mm markers would be used on the upper back cluster to reduce its size, and 14mm on the mid back cluster. This lead to problem b) being witnessed.

Problem b) was mainly witnessed when only 5mm markers were used on the upper back cluster though it also happens to the other clusters using only the 14mm markers but not to the same extent. The upper back cluster worked using the smaller markers but moving too quickly or too much caused the one or more of the markers to drop out causing the creation of the cluster to fail in the application though this did not cause any real damage in the beginning of the development process this became a problem later on with how the cluster

and the markers are labeled and as such the decision was made to increase the size of the cluster and to use 14mm markers instead to reduce the amount of drop outs that occurred.

The clusters created for bench testing the application and its creation processed fulfilled their role though some improvements can be made to make then more robust especially when it comes to attaching then to a patient. Please see Appendix I for images of all five completed clusters with corresponding marker placement.

2.4 Cluster Tracking

With the clusters created and cameras setup the first step of the application development process was to record each individual cluster to find the coordinates the 4 markers on the cluster as these coordinates are used to identify the individual clusters from within the capture volume. The initial steps to this is to first create a calibration file which includes the number of markers per cluster as well as the X, Y, Z coordinates of the markers which make up the clusters. To do this a pre-existing application developed by Professor Philip Rowe of the University of Strathclyde was used and can be found as Appendix I. The application works by taking the position of the markers in the current frame and the markers position in the previous frame and taking the average of the 2 these values and as such produce a set of coordinates for all 4 different markers on each cluster. This can be seen in Table 1

Table 1: Table 1 showing an example of the X, Y, and Z coordinates of 4 cluster markers using a test layout

0.0065386355854571	0.021027226001024	-0.00053236965322867
-0.028414577245712	**0.021022409200668**	**-0.0051563042216003**
0.025300938636065	**0.020421786233783**	**-0.022766890004277**
-0.063294805586338	**0.019407499581575**	**-0.033360552042723**

Once the marker coordinates have been created for all five clusters, the files are then used with in the main application, in the cluster tracker modules (see Appendix III for code). Each cluster has its own individual script module as this makes coding the module significantly easier. Within the script the coordinates of the 4 markers, for example Table 1, are loaded in where by the script calculates the distances between all the markers on the cluster and orders them in ascending order for the first frame and takes a sum of the ordered length, there is also a tolerance level set both of which the values are important later on.

After the initialization of the first frame the script then goes on to updating for each new frame whereby it searches through and eliminates all the missing markers, this is in case the number of markers available has been set too high. The maximum number of markers that are used for this current setup is 20 but within the code it says that there are 40 markers available this has been done in case there are future wishes to include additional individual markers included with in the model in order to find additional information for example the knee joint center or the center of the pelvis. These missing markers are determined by taking the absolute value of their coordinates and if the sum of these is equal to zero they are then given extreme coordinates to move them out of the visualization area.

The next step is to calculate the distances between all the remaining markers, this is done in a similar manner as in the initialization of the first frame whereby they are then ordered descendingly and ranked to find the markers closest neighbors. A sum of the 4 closest markers is then taken and this is compared to the sum taken from the first frame and a difference is taken, if the difference is less than the tolerance level set in the beginning then the 4 markers that are closest to those of the cluster are then placed in the corresponding positions as to where they are on the cluster. If the difference is more than the tolerance then the cluster disappears, the same will happen if one of the cluster markers is obscured and this poses a problem when recording a movement though a lot of the initial problems that were encountered with this we solved using the solutions discussed in section 2.2.

Very little was done in terms of changing the code that was written by Prof Rowe the main thing that was edited was the tolerance levels for identifying the clusters. To find an adequate and workable tolerance level a static test was first conducted which involved placing all 5 clusters on a participant and taking a recording of them standing still which was played back on a loop and the tolerance level of each cluster adjusted till they all showed up and exhibited no interference with the other clusters. This proved to be an issue with two of the cluster namely the mid back cluster and the pelvis cluster the proposed issue was that the clusters were too similar in marker placement and so the scripts could not accurately differentiate between the two even with increasing the tolerance level, and so the pelvis clusters configuration was changed. This solved the problem within a static situation and so a dynamic test was conducted which was recorded where by the participant was asked to preform small movements such as extending and flexing their back, shallow squats as well as bending laterally on both sides. This allowed greater fine tuning from the recording of the

tolerance levels, though it is important to note that if the tolerance level was set too tight ~2mm (0.002) then if the participant moved too quickly then the clusters would drop out as the application could not run quick enough, though this was caused by a limitation within the computing power of the computer being used and not of the code itself.

One of the other problems that occurred during this development phase was when a marker fell off the clusters as they were only being held in place using medical grade double-sided tape. Though this only happened once or twice throughout the entire application development process it proved to be a time consuming process to correct as either the marker had to be placed back into the correct position or a new calibration file had to be created which then involved retesting the tolerance levels. This could be easily solved if the markers were being used long term in those positions by drilling into the back of the clusters and screwing the markers onto mounts so as to prevent them from falling off but as the markers were only being used for a short amount of time it was deemed unnecessary.

2.5 Cluster Marker Labelling

With the clusters now being recognised within the scripts and application as a whole the next step to be developed was to be able to label and number the markers on each cluster for use in further steps. To do this two pieces of code were written, the first being the Labelling script module (see Appendix IV) and a function called labelling002 (see Appendix V); initial versions of these have been previously developed within the department and were provided by Lindsay Millar, PhD student, University of Strathclyde, Glasgow.

The main Labelling script is set up so that all of the 20 markers coordinates, on the 5 clusters, are input into the same script module; unlike before when each cluster had its own individual module. The outputs of the cluster tracking modules are arranged so that first 12 input channels are for the upper back cluster then mid back, pelvis, left thigh and finally the right thigh totalling 60 input channels. These are then sorted back into their groups for each cluster where by the 12 input channels that correspond to the cluster are condensed down into a 4 marker coordinate systems, of which are used for the labelling002 function.

Before any changes to the original version of the labelling002 function were made the order in which D-Flow numbers the markers on each cluster had to be first determined this was done using the marker matcher modules to find out the assigned marker numbers Figure 8.

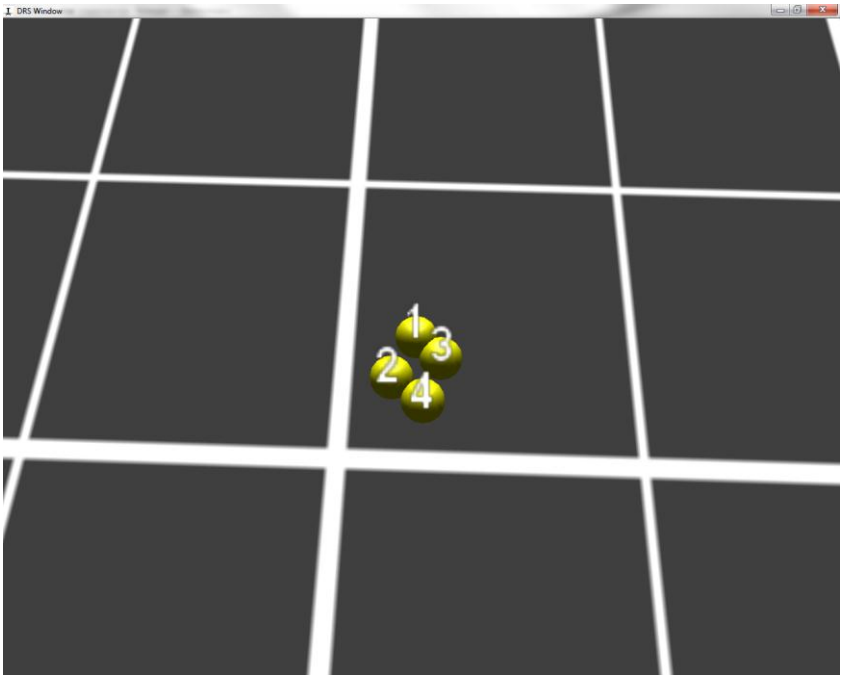


Figure 8 Markers showing the numbering assigned through D-Dlow

This is an important step as the order that the markers come into the application may not be the same every time and so the input channel numbers of subsequent scripts may vary and will provide widely different answers to the problem. By knowing the number that D-Flow is assigning a marker through the marker matcher it is possible to check that the module and function are operating correctly and that the same output number of the Labelling script is applied every time, this is where the labelling002 function comes in.

For each cluster when the four marker coordinates are inputted into the Labelling module they are fed into the labelling002 function where it completes a series of calculations in which it determines the Euclidean distances between each for the markers on the cluster where it then creates an array of these values from shortest to longest (Table 2) and with what markers make up these distances the is order and the markers are what is used to label each of the markers. With this information it is then possible to draw a diagram for example the upper back cluster Figure 9.

Table 2: Array of distances between two markers arranged in descending order from shortest to longest.

Distance	Marker 1	Marker 2
0.023282008757854	1	3
0.026890593714869	2	4
0.0323361287521	3	4
0.03372795408043	1	2
0.036451413805633	2	3
0.045839858732595	1	4

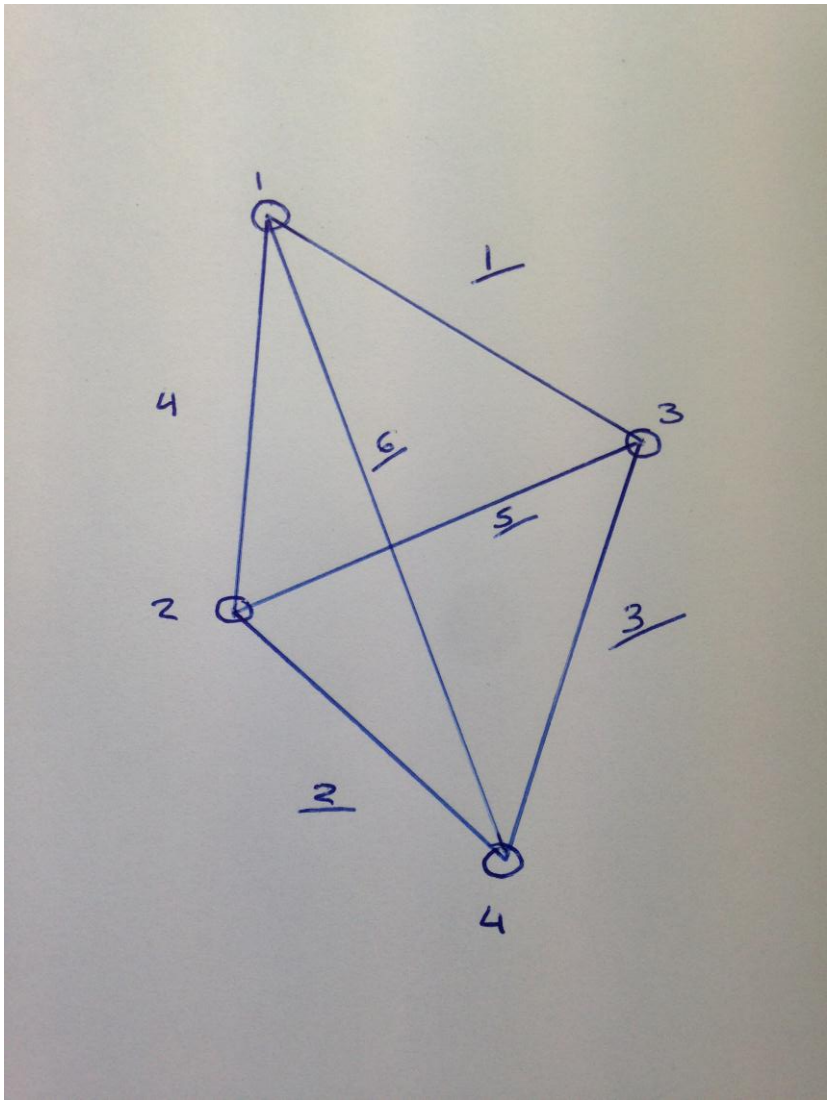


Figure 9 Illustration showing the arrangement of the distances. Underlined numbers are the lengths corresponding to Table 2

For the purposes of this application it is desired that the markers are numbered from top to bottom and as can be seen in Figure 9 this is not the case and so this is where the next part of the labelling002 code solves the issue. In this instance the marker that D-Flow has labelled number 1 is what the application wants as marker number 1. To solve this the code looks for a marker to start with, of which, the starting marker must be part of the shortest distance. As marker 1 is part of the shortest distance it has been chosen as the starting marker though marker 3 in the application labelling system could also be used. With the starting marker now found the code looks at the other marker that makes up the shortest distance and sets this, as the second marker in this example the other marker that makes up the shortest distance is application marker number 3. With two markers now found and assigned marker numbers the other two markers can now be found by using the line that joins the starting marker to the marker that is needed i.e. from Figure 9 it can be seen that what is desired as marker 3 is at the end of line 4 and so in the code it is set so that marker 3 is the other marker at the end of line 4 starting at marker 1. By process of elimination and observation marker 4 is at the end of line 6 and is inputted into the code in a similar manner. All the coding for this example cluster and its numbering can be seen in Appendix IV in the function labelUBMarkers(segmentMarkers). With the marker order now correct for the application the same process can then be applied to the remaining cluster using the same process each with its own function as displayed above. The order is then output back into the main Labelling script where by it outputs the coordinates of each marker onto the correct output channel.

This was one of the most time consuming sections to get initially working due to it being written by someone else with the main reason not realising that the first marker has to be part of the shortest distance though once this issue was addressed it was very easy to go through and correct the mistakes and replicate for additional clusters.

2.6 Cluster centroid calculation

The initial idea for this section of the application was to have the clusters be able to calculate an axis system for each individual cluster to be able to define a definite plane for each of the major spinal level unfortunately this proved to be problematic with the code being able to calculate a three axis system for the clusters but not in the correct orientation or position. After spending a considerable amount of time trying to sort this issue out and with the

allocated time to complete the project running out it was decided to forget this section and instead go for an easier alternative where by the centre of each cluster would be calculated and this used instead.

With the markers labelled through the labelling module it was possible to easily calculate the centre of each cluster where by very simple piece of code was written (see Appendix VI) where it takes, for example, the upper back markers, and finds the maximum X value of the 4 markers and then does the same for the minimum X values and taking the average of the two by adding them together and dividing by 2; the same process is the repeated for the Y and Z coordinates. This allows an imaginary marker to be created at the coordinates of the centre of the cluster markers of which, this new marker is then used as a reference point in order to calculate distances between clusters for the next stage of the application.

2.7 Recording and Comparison

To have something to compare the live feed to, a section of the application must be created that calculates the distances between clusters in the position that the physiotherapist places the patient in in order to unload the soft tissues. This is done quite simply by creating a copy of the live section of the code the main difference between being that instead of the motion capture module outputting a live feed directly from NEXUS it instead plays a recorded file, on loop, of the patient in the unloaded position. This then runs through the same processes as described in the previous sections to output marker centroids for the recorded markers.

To take a recording is a very straightforward process whereby in the motion capture module there is a record button. After assigning the save location through the module the record button can be pressed and D-Flow will record the marker positions in the capture volume and output this as a text file that contains all the X, Y, and Z coordinates for all 20 makers. For this application at this moment of time it is deemed that shorter recording times are more stable in the application and cause fewer crashes of which will be discussed later in Chapter 4. Changing what the motion capture module is reading form Live to File can then play the file back

With the recording and live sections of the application now complete and outputting a set of centroid coordinates it is now possible to compare the two sections together, which leads to the final module of this application the code can be found in Appendix VII.

This module will also provide the visual feedback for the application and so the objects that are used for this are first created of which each of the main points has 4 different markers white, red, yellow and green each representing a different tolerance level which will defined later on in the code.

The live marker positions are pulled into the code and are placed with a new marker at the centroid location of the cluster markers derived earlier. With these now inputted it is possible to calculate the differences between the recording and the live feed. Initially in the development process it was decided that in order to determine the differences between the live and recorded positions the easiest way would be to calculate the distances from the origin to the centroid markers; but it was pointed out in a meeting with Professor Rowe that this would mean the patient had to stand in exactly the same position every time that the movement or an adjustment wanted to be made, ultimately this would make the application more time consuming to use which would decrease its feasibility in a physiotherapy situation due to the physiotherapist spending more time getting the patient into the right position instead of helping them treat the patient.

With this in mind it was suggested that instead of using the origin of the capture volume that instead the pelvis is treated as the origin and that distances are calculated from there. To calculate the distances between the pelvis and the clusters the following formula was used:

$$|Distance| = \sqrt{(X_1 - X_2)^2 + (Y_1 - Y_2)^2 + (Z_1 - Z_2)^2}$$

Equation 1 Equation used in the calculation of distances

Where by X_1 is the X coordinate of the pelvis cluster centroid and X_2 is the X coordinate of the cluster that the distance is being calculated to. The only difference is in the upper back where by the distance is calculated from the mid back cluster instead of the pelvis where by X_1 is the upper back and X_2 is for the mid back. The same principle is applied of the Y and Z coordinates. This applied both to the live and recorded sections as can be seen in Appendix VII.

It is now possible to be able to determine the difference between the corresponding sections by simple subtraction and then squaring and square rooting to get rid of any negatives. The number calculated then goes into a series of logic statements that are used to provide the visual feedback to the patient for example if the total difference between the live and recorded distances from the pelvis to the left thigh is greater than 1 then the marker at the centre of the left thigh would appear white. As the patient moves their left thigh closer to the recorded distance the marker at the centre of the left thigh will change from white to red to yellow and then green each with its own tolerance level this is done by saying that when difference is within that range send the other 3 coloured markers off to extreme coordinates and place the new coloured one at the centre of the cluster. At the same time as the individual clusters are changing the pelvis has its own individual visual feedback system.

The pelvis has its own marker placed at its centre but is much larger to make it more visible for visual feedback. When one of the clusters reaches with in its green tolerance level it defines a variable called VFB (visual feedback) followed by what cluster it is in relation to. Taking the previous example, once the left thigh is in its correct position the script will output a variable with a value of 1 called VFBLT where by these variables for the 4 clusters are then summed together. With each cluster that gets into the correct position the larger marker at the centre of the pelvis will also change colour with white being 1 cluster, then red, yellow with green being the final colour when all 4 clusters are in their correct position. This system provides some rudimentary visual feedback for both the patient and physiotherapist with room to improve as the application is developed further.

Chapter 3 - Results and Validation

3.1 Introduction

In order to test and validate the application is working as intended three tests where proposed. The first a static bench test where by the clusters are placed on a stool within the capture volume to test the application as a whole to ensure that it runs, first as just live mode and then introduce a recording. The second being a static test with the clusters placed on a skeleton to ensure that the application works when the clusters are placed into their anatomical positions, as most of the testing that had been conducted so far was using the bench test method as in the first test, as well as to again test the tolerance levels of the cluster recognition modules as their orientation has changed and the final test where by the skeleton

is placed into a set random position and a recording is taken to test how well the application tracks movement as well as to test the visual feedback system.

3.2 Test 1: Bench test

With test 1 the clusters were placed on a stool as in Figure 10

Figure 10 Configuration of clusters used in the bench test

The reason for this configuration was to place each cluster in its relative position to where it would be placed on the body. With the application running in just live mode it can be seen in Figure 11 that each of the clusters shows up and in its correct position and with the correct marker configuration.

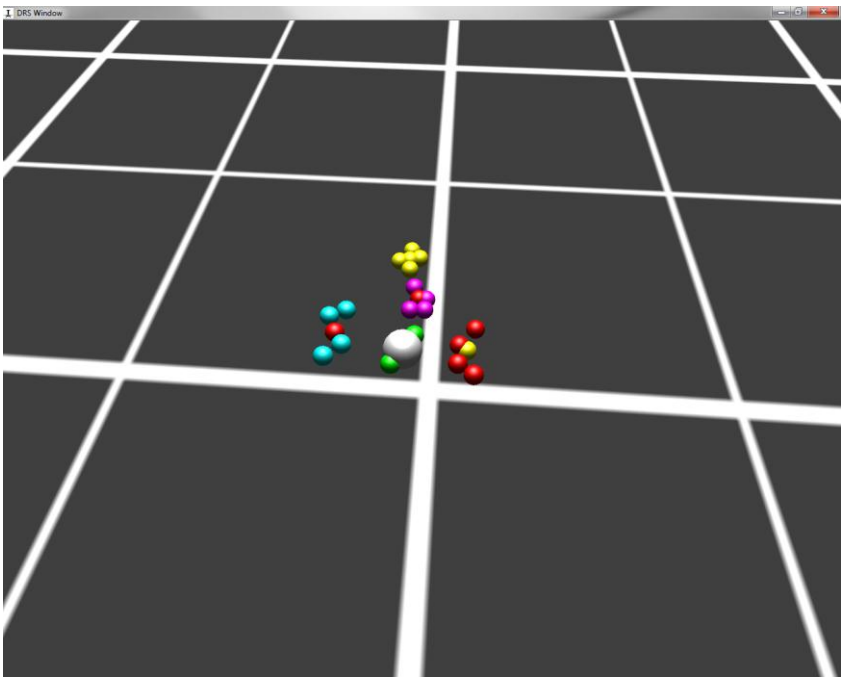


Figure 11 Image showing the cluster being recognised in the capture volume by the code

One of the things that was observed with this setup was a conflict between the pelvis cluster and the mid back cluster where by the cluster flickered in and out and so the tolerance of the pelvis cluster was decreased from 0.04 to 0.03 this eliminated the issue. A recording was then taken and that section of the application started and as Figure 12 shows each of the centre makers is green indicating that the application is working as intended.

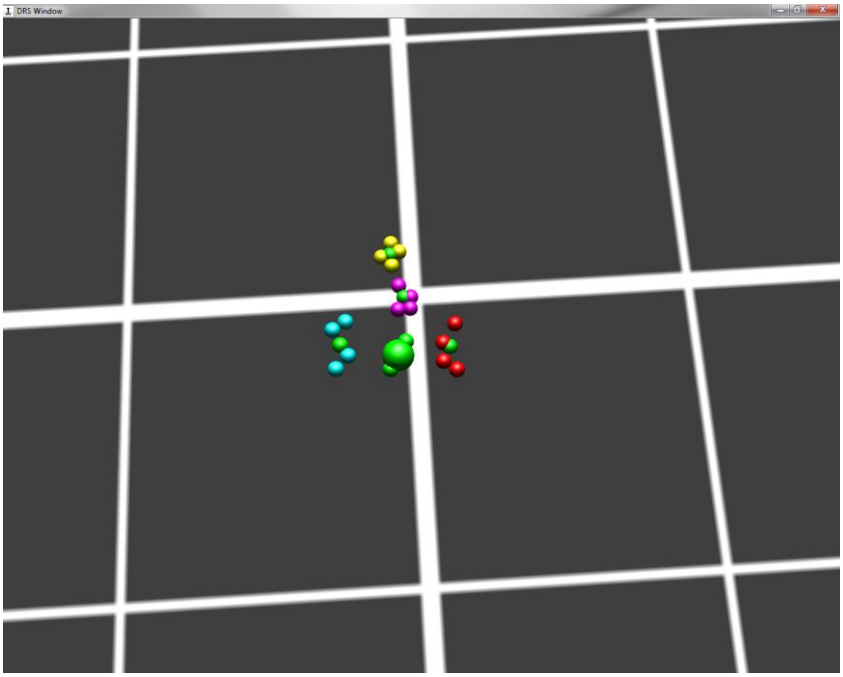


Figure 12 Comparison between the recorded and live feeds the green centres indicating that the cluster is in the correct position and that the application is working as intended

After the initial testing it was decided to quickly test the visual feedback system by moving a cluster one at a time first to check the main feedback system in terms of the pelvis and then each individual cluster. Figure 13 show the pelvis when only 2 clusters are in their correct position and the result is as to be expected with the centre of the pelvis indicating red.

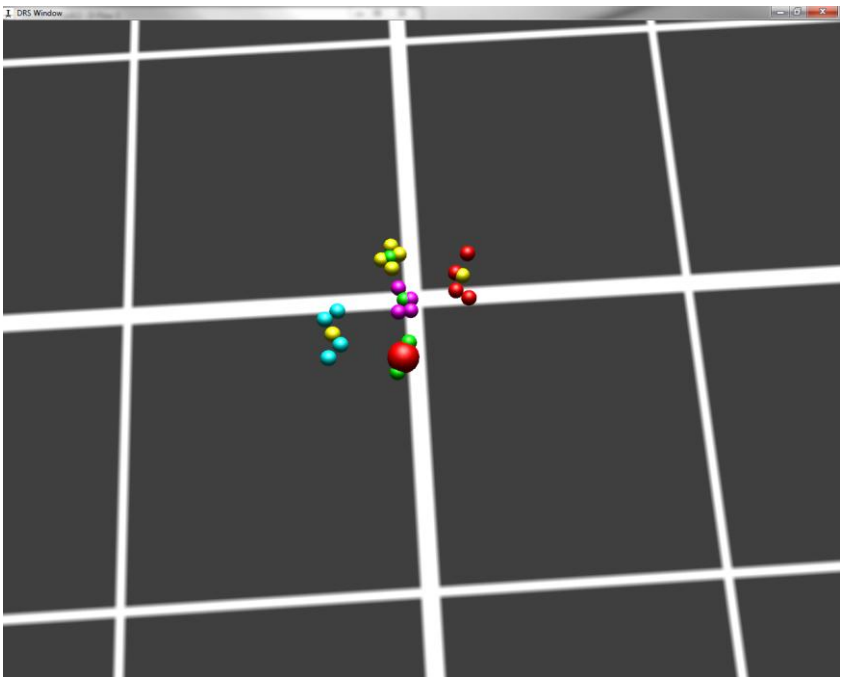


Figure 13 Imaging showing that as the thighs have now been moved out of their correct positions that the visual feedback responds according to the code

From these results it can now be said that the application works but so far only in a bench test situation where the clusters are held in a stable environment with little to no outside interference and as such the next set is to test it on a skeleton to check that the application works when the clusters are positioned anatomically.

3.3 Test 2: Static test on skeleton

The clusters were placed as close to the anatomical positions stated in Section 2.1 with socks used to pad out the thighs and pelvis to give them a more realistic shape and additionally to help them stay attached to the skeleton as can be seen in Figure 14 and Figure 15

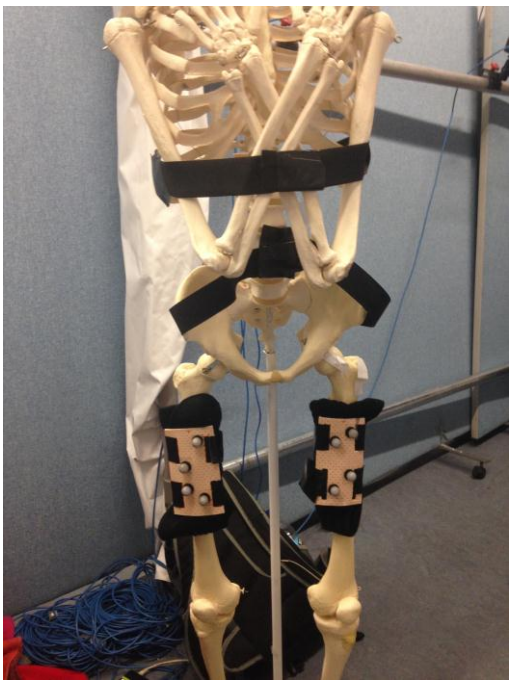


Figure 14 Frontal view showing thigh clusters

Figure 15 Rear view showing upper back, mid back and pelvis clusters

Using the same procedure as above and starting it off in live mode the following image in Figure 16 was captured.

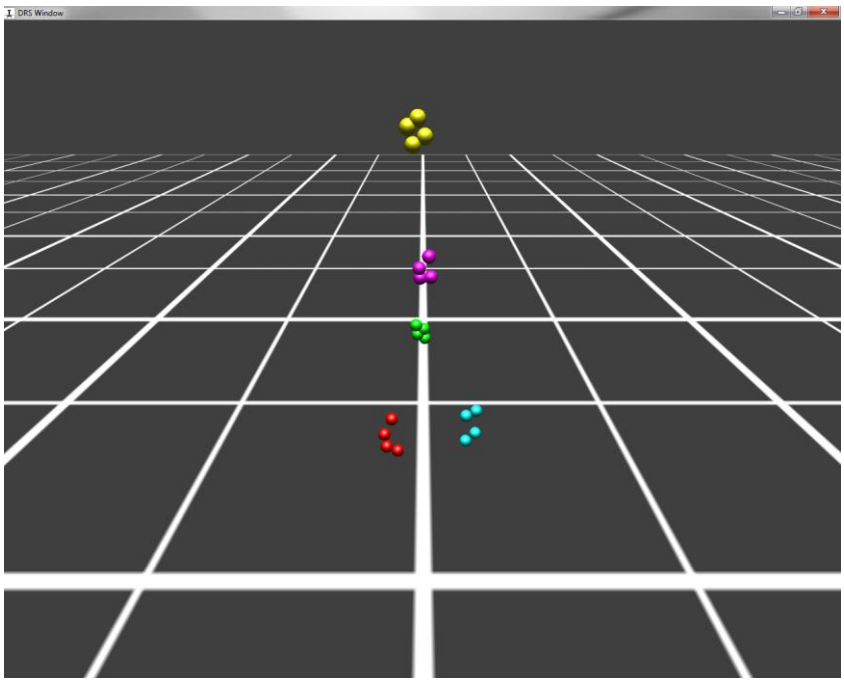


Figure 16 Image showing cluster being recognised when placed on the skeleton

This was the first time the clusters had been placed on the skeleton as it had been unavailable but the figure clearly shows that the clusters are being tracked correctly even with a gentle oscillation applied to the skeleton, by gently rocking the shoulder, to check that it still works when moving as when the clusters are placed on to a patient they are not going to be able to stay exactly still like a skeleton so generating a little movement to simulate the kinds that would be exhibited on a patient this helps to validate the application.

At this moment within the testing the application is working as intended but this is with the skeletons hands taped up as seen in Figure 14. This is not a natural position that patients may find themselves in and so the hands of the skeleton were released and allowed to fall to the side. One of the things that was observed is that when the oscillation was again applied that the thigh cluster sometimes dropped out; this being caused by the hands of the skeleton obscuring the markers as they move back and forth and therefore causing the cluster to drop out. This issue can to some extend be ignored as a patient will be able to control what there are doing and so will stop their arms moving around as much as those of the skeleton but

care must be taken and the patient made aware that this is a potential downfall area and so care must be taken.

Using the same recording as was used in the static bench test to check that application still worked in its comparison Figure 17 was observed.

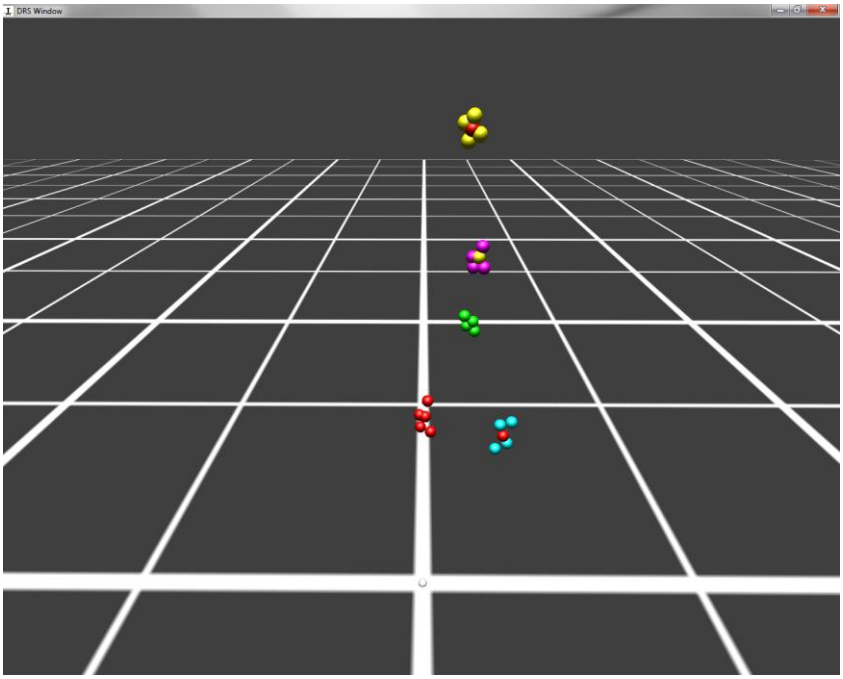


Figure 17 Showing visual feeback of clusters based off the recording used in the bench test

Whereby it can be seen that all bar the mid back cluster are indicating red meaning that they are a significant distance away from that of the recording as is hoped the application would do. The mid back cluster is not surprising in the fact that it is yellow as the distance between the pelvis and the mid back cluster is the smallest and as such it may be advisable to alter the tolerance levels for these clusters.

3.4 Test 3: Dynamic testing on skeleton

With the markers being kept in the same place and the hands of the skeleton still lose it was then time to perform some dynamic testing of the application to do this a recording is first taken with the skeleton in a new position. The skeleton is naturally slumped slightly forward so a correction was made where by the skeleton was straightened up at which a recording was taken in this straightened position, the marker positions of which can be seen in Figure 18

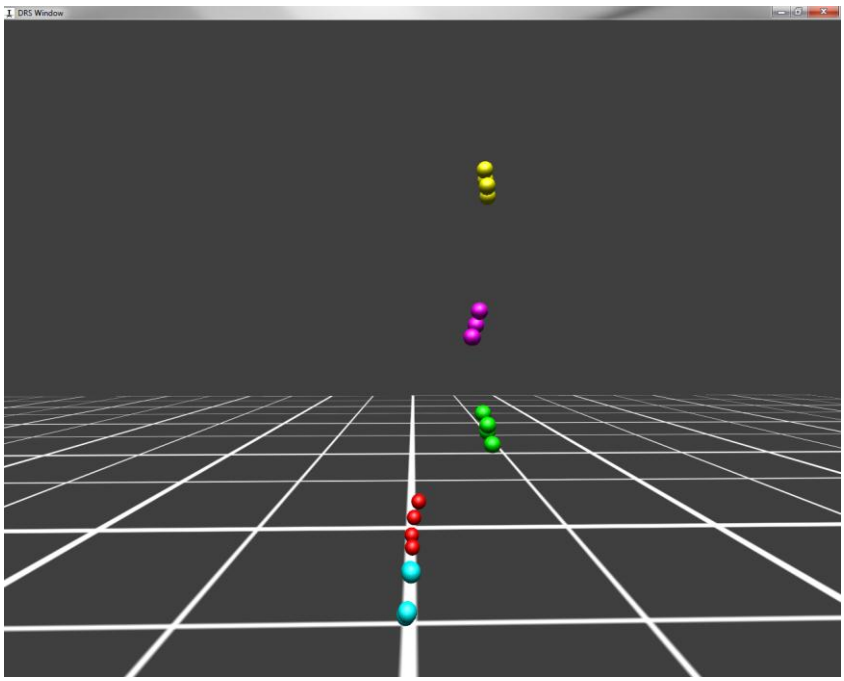


Figure 18 Showing the natural position of the skeleton with clusters attached

Figure 19 shows the difference between the natural and corrected positions.

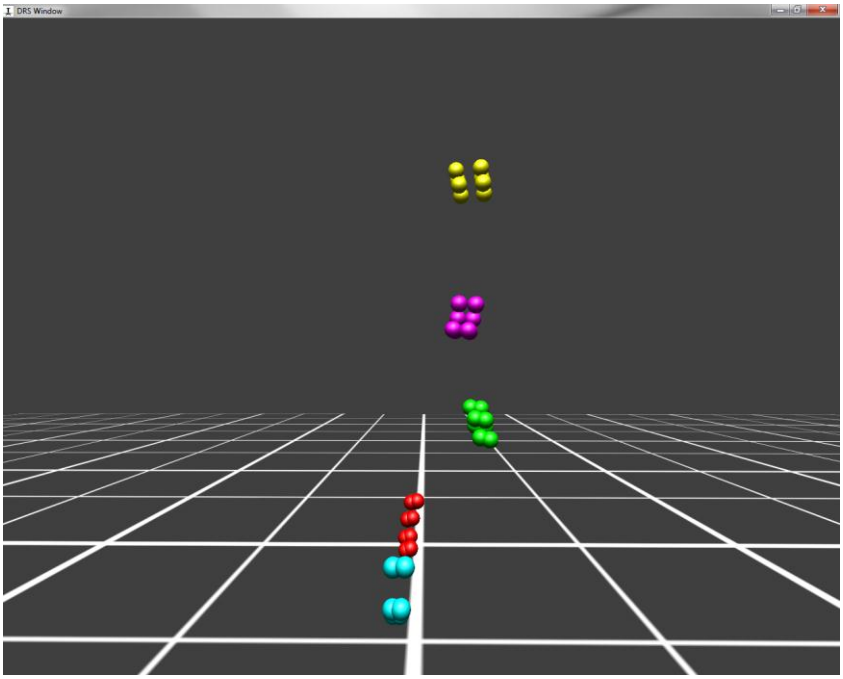


Figure 19 Showing the difference between natural and corrected positions

As it can be seen from Figure 19 by straightening up the skeleton it has moved the upper back cluster posteriorly with little to no change in height. When the clusters are in their corrected positions all the clusters are green which is what it hoped for as in Figure 20

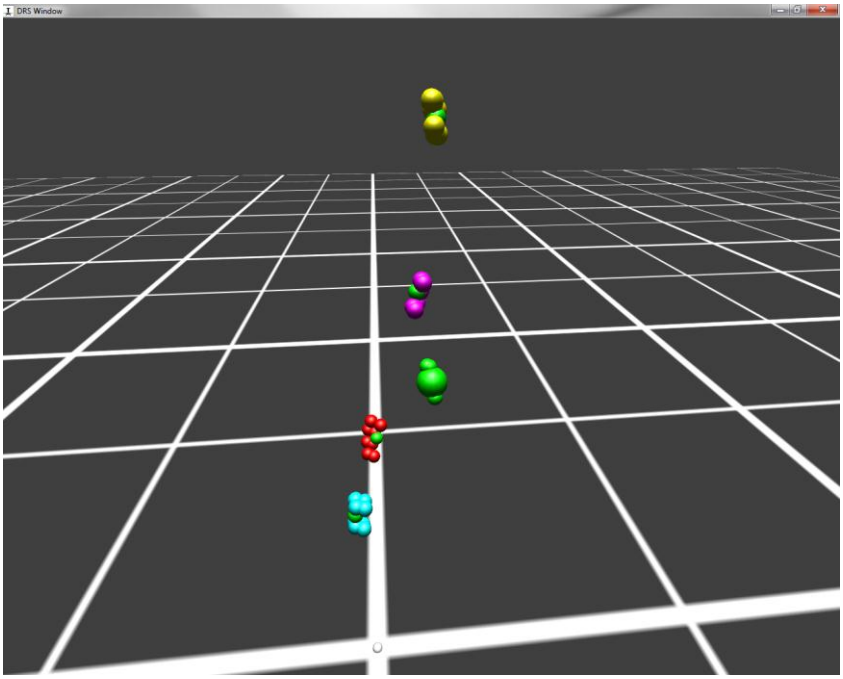


Figure 20 Showing visual feedback based off new recording

There is a problem with this though as when the skeleton was released into its original resting position all the indicators are still green. Looking at the outputs of the code the reason for this is even though the clusters have moved positions the distances between the clusters has stayed relatively unchanged. The tolerance levels could be reduced to fix this but as some of the changes in length were in the thousandths this would make to system incredibly sensitive.

So in order to better test the dynamic capabilities of the application the clusters were placed on to a participant in the same configuration as that of the skeleton; where they were then asked to stand up as tall as they could and a recording was taken. They were then asked to relax and slump their shoulders this is done to simulate when a physiotherapist gets a patient to repeat a movement/correction. With the visual feedback turned on they were then asked to get back into that tall position with the application now able to directly differentiate between the relaxed position and the corrected one.

Comparing the results of the two dynamic test situations and comparing the outcomes it could be proposed that the reason why the skeleton didn't work quite as well as the participant trial is because of the how rigid and relatively imposable the skeleton is and as such the range of movements that it can perform are significantly less.

Taking all the results from all three tests and looking back at the purpose of this application and what it aims to achieve it can be said that at this moment of time that the application works as intended and provides valid results and outcomes that show that it has potential in being able to be used as a rehabilitation tool in a clinical situation. Ideally the next steps in testing the application would to have a trained physiotherapist come and try the application and give feedback as to the features at that moment of time they like and areas of which they feel that could be improved.

Chapter 4 – Discussion, Limitations and Future Developments

4.1 Introduction

In this chapter will discuss the relevance of the application in terms of both a biomechanical as well as a clinical situation and how potential other applications of this nature will help to bridge the gap between the clinicians and biomechanists. It will also discuss the limitations that were found during the development of the application and how these affect the application, being shortly followed by the future developments that can be implemented to better the application and solve some of the problems outlined in the limitations.

4.2 Discussion

The overall development process of the application was relatively straightforward but there was a large learning curve in the beginning when it came to initially learning how to use the software, especially in setting up the cameras to be able to able to get both good recognition of the markers through VICON Nexus but at the same time allowing the cameras to have enough field of view so as to generate as big of a capture volume as possible. D-Flow as well has its issues with setup in particular getting the marker outputs of Nexus into D-Flow so that it can recognise their positions and the actual coding of the application is a potential stumbling block when it comes to being able to fix issues as unless the person has a good understanding of the base language in this case Lua then any changes that they may make can cause other issues further down the line.

For those that are not particularly good with computer this possess an issue as for them being able to troubleshoot an issue, such as if the cameras were to stop working, as the process is not straight forward and this is where potential differences between clinicians and biomechanists comes into effect. Biomechanists working with motion capture technology will find it easier to solve these problems as they spend more time using these systems. Contrast this to clinicians whereby they may only have a few training sessions to get familiar to the technology if something goes wrong they are less likely to able to use the system if something breaks and as such may less likely to use the technology and this is one of the areas in which progress needs to be made in order to make this technology more intuitive and easier to use for people to use in everyday life and in the case of clinicians introduce it more into the rehabilitation of patients.

There is also the issue in where by clinicians and biomechanists don't really talk to each other. There is a plethora of biomechanical studies looking at anywhere from spinal biomechanics to that of the foot but clinicians feel there is a lot of big conclusions drawn from little thing and as such tend to ignore it and so a better link between the two groups needs to be found where by clinicians go to biomechanists with questions and problems and the biomechanists then apply their knowledge to help to solve the problem.

When looking at the application that was developed as part of this project it in of itself is fairly unique in the way in which it aims to approach the problem of applying this biomechanical technology in terms of motion capture and software to a clinical situation. The amount of papers that could be found that actually used this technology as part of the rehabilitation was relatively quite small where instead the vast majority used it as an evaluative tool to look at areas such as kinematics and kinetics pre and post intervention. This was also evident at the recent 25th Congress of International Society of Biomechanics (ISB), Glasgow, 2015 where by searching in the abstract book[5] for the term "visual feedback" only one out of the nine papers was related to its usage in rehabilitation. This shows that there is large gap in where this technology could be applied that would help in reducing the amount of time that a person requires treatment, which will help to reduce waiting times, and additionally in term of LBP potentially reduce the amount of people that suffer from chronic LBP.

Motion capture technology is expensive with the camera system being used in this project costing upwards of £10000 but steps have been made to make motion capture technology more accessible as seen in Section 1.2.1 with the usage of the Xbox Kinect not only is this a cheaper alternative costing ~£100 but it is also significantly smaller in terms of overall size in comparison to the multi camera system. One company that has made use of this has been Motek ForceLink, Amsterdam, Netherlands, whereby they have developed a gait retraining system called the ReGait[6] where by using the Kinect it is able to detect changes in step length, height and frequency this shows that there are attempts to get this cheaper alternative technology into clinical settings. The main issue with these smaller scale systems is that they lack the same resolution and accuracy that these bigger systems have and in some ways this limits their ability to be used due to the fact that they cannot recognise smaller movements so

[5] https://dl.dropboxusercontent.com/u/4465273/ISB_2015_Abstract_Book_Final.pdf
[6] http://www.motekforcelink.com/product/regait/

to be able to really see them being used main stream more work is going to have to be done first. Additionally the Kinect can be said to only see in 2.5D where by it can see both height and width but even though it has depth perception it can't quantifiably tell how far back something goes and as such systems need to be created whereby they can utilise two Kinects to be able to quantify this last dimension.

4.3 Limitations

One of the biggest limitations of the application is in how the markers are labelled within the code. The problem is due to how the code is written whereby it runs linearly and is based off a hierarchy. What this means is if for example if one of the markers on cannot be found on the mid back the whole application fails at the point at which it tries to label that marker and stops working this poses a huge problem when it comes to dynamic movements, especially recording as if there is a lost marker in the recording the application will always fail at that point this is why it is important to point out when a patient is wearing the clusters to watch where they put their hands as in testing this was the main reason for most of the failures. Care must also be taken with where the physiotherapist stands as if they stand directly in front or behind a patient this will obscure the markers and so to stop the application from constantly failing they would have to stand to the side. This in some cases may not be the best way to treat a patient and so would negate the effectiveness of the application.

Correcting this problem would be both very difficult as well as time consuming so within timeframe that was given for the development of the application it is a limitation that has to be worked around that it why in Section 2.7 it is said that short recording lengths work better this is because they reduce the amount of time that a marker can go missing resulting in failures. This ultimately is the main problem with the application that prevents it running smoothly as every time the application stops it must be restarted and with the current setup involves having to go back to the computer and press the play button in D-Flow, which is less than desirable.

One of the other issues with this module is that all five clusters must be in the capture volume, which can limit the applications flexibility. The current work around for this is to place the clusters that are needed in their correct positions and place the remaining ones at the edge of the capture volume where they can still be picked up by the cameras. The problem with this is that it will alter how the visual feedback works for example if only the thigh and pelvis clusters are being used the upper and mid back clusters would be placed

away. As these two clusters are always going to be in the same position they will output green markers causing the starting pelvis indicator to start off at red instead of blank.

One of the other main limitations of the application is the amount of computing power that is required to run it smoothly. When nothing is running D-Flow runs at a comfortable 300Hz as soon as the application is activated it slows down to 18hz this is not ideal as it slows how fast the application can run. This may also be a factor in why short recording work better than long ones as if the clusters are moving too fast the application wont is able to catch up with them causing a drop. The easy solution to this problem is to get a more powerful computer but that may not be an option especially when the camera system may be getting used by multiple people and could be running off a laptop.

The final limitation is in how the correct and current positions are compared this is evident in Section 3.4 where by a small change is not accurately represented and as such is not very good at identifying small changes. This could potential limit the usage in some populations in which these smaller but important adjustments are made for example in consultation with John McMenemy he described a patient that was a former military personal and as such when standing up straight he would naturally stand to attention which forced his hips to posteriorly tilt. In this situation the correction was small whereby the patient was asked to relax and allow their hips to drop and tilt anteriorly; if the application was to be to used in this instance due to its lack of sensitivity it most likely would not be able to pick up the change in relation to the thighs and mid back clusters.

4.4 Future Developments

The first thing that that would have to be addressed is how to get the labelling module to work when it loses a marker and so that the application still runs. One of the suggested approaches to this is to create a temporary variable whereby it stores the locations of the previous frame and with this information if one of the markers drops out it will use the previous frames cluster location till a frame in which it can find all four of the clusters markers can be found. Another alternative is to separate each cluster into a logic loop where by it states that if any of the markers of that cluster return a nil value then to set that clusters markers to extreme coordinates this may potentially be the easier option but at the same time the least useful as a complete cluster is lost rather than just lagging behind or in a slightly incorrect position as would be the case in the first solution. Once this main problem has been

sorted and the application running smoothly improvements can be made to start including additional information.

As stated in Section 2.1 one of the pieces of information that would ideally be looked for in the application is the pelvic angle. Currently only the location of the pelvis is found, by including how to calculate the pelvic angle the pelvic tilt of a patient is able to be found which is one of the corrections that a physiotherapist can make to help to treat back pain. To do this a relationship can be formed with the pelvic cluster where by using four additional markers located at the LASIS, RASIS, LPSIS and RPSIS and a pointer that is used to label these new markers. This allows the anatomical centre of the pelvis to be found in addition the location of these four new markers allows the orientation of the pelvis to be quantified.

This leads onto the next section whereby introducing these new markers will require the visual feedback system to be altered where it takes into account these new markers the easiest way to do this would be to relate them to the pelvis cluster using the same method as described above where by the distance is taken form each if these new pelvis makers to the centre of the pelvis. An additional element that may be added is in the form of an arrow that points out from the anatomical centre of the hip to indicate the pelvic tilt and potentially introduce a similar traffic light based system that relates the angle of the pelvis to the global axis system with an adjustment parameter that allows a physiotherapist to dictated how much pelvic tilt they want the patient to experience with a tolerance either side of 5 degrees which can be reduced as the patient get better at controlling where their hips are and as they gain better body awareness.

The final improvement that need to be made is the creation of an easy to use graphical interface (GUI) at the moment in order to show or hide markers the code has to be manually changed so that it turns the visual element of the clusters off even though this is an easy thing to do it is both time consuming and if the wrong section is changed then it could cause the entire application to fail. The ideal GUI would include the ability to switch off the live and recorded cluster from appearing which stops the visualisation window form getting crowded with markers, the ability to only run set parts of the application so a play button for the live part and a play button for the recorded and finally an easy way to record and play back the recorded data to the patient.

With these adjustments to the application it should be able to fulfil more of the initial brief provided by John McMenemy and should also at the same time make it more user friendly to other physiotherapists where they will require minimal training in order to use the application quickly and effectively.

Chapter 5 – Conclusion

In order to see more motion capture technology being used within a clinical setting more steps must be taken to make this technology more user-friendly and intuitive as well as affordable but steps have been taken to improve this. Looking back to the objective that were set for the project in Section 2.1 the author feels that these have been meet though there are obviously still some of the components that were asked for still missing in particular the spinal curvature and a robust avatar that accurately portrays a patient rather than just a cluster of markers on the screen which initially mean nothing to a patient until it is explained to them. The application development went overall smoothly with a few limitations coming into effect where development time took longer than expected in particular the labelling and axis generation which set the development back a week and mean that some the of sections such as the GUI had to be unfortunately put aside for when there is more time. Overall the application works well taking into account some of its limitations but it gives out accurate and usable information that can easily be built upon if the project is taken up again. In conclusion the project was a success with an application being developed that could be used within a clinical situation.

Bibliography

References

Adams, M. (2004). Biomechanics of back pain. *Acupuncture in Medicine*, 22(4), pp.178-188.

Aung, H. (2015). Automatic Recognition of Protective Behaviour in Chronic Pain Rehabilitation. Trento, Italy.

Bailey, S. and Bodenheimer, B. (2012). A comparison of motion capture data recorded from a Vicon system and a Microsoft Kinect sensor. *Proceedings of the ACM Symposium on Applied Perception - SAP '12*.

Bouwmeester, W., van Enst, A. and van Tulder, M. (2009). Quality of Low Back Pain Guidelines Improved.*Spine*, 34(23), pp.2562-2567.

Dunn, K. (2004). Epidemiology and natural history of low back pain. *Europa Medicophysica*, (40), pp.9-13.

Fern'ndez-Baena, A., Susin, A. and Lligadas, X. (2012). Biomechanical Validation of Upper-Body and Lower-Body Joint Movements of Kinect Motion Capture Data for Rehabilitation Treatments. *2012 Fourth International Conference on Intelligent Networking and Collaborative Systems*.

Galna, B., Barry, G., Jackson, D., Mhiripiri, D., Olivier, P. and Rochester, L. (2014). Accuracy of the Microsoft Kinect sensor for measuring movement in people with Parkinson's disease. *Gait & Posture*, 39(4), pp.1062-1068.

Hayden, J. (1996). Exercise therapy for treatment of non-specific low back pain. *Cochrane Database of Systematic Reviews*.

Jansen-Kosterink, S., Huis in ’t Veld, R., Schönauer, C., Kaufmann, H., Hermens, H. and Vollenbroek-Hutten, M. (2013). A Serious Exergame for Patients Suffering from Chronic Musculoskeletal Back and Neck Pain: A Pilot Study. *Games for Health Journal*, 2(5), pp.299-307.

Kharrazi, H., Lu, A., Gharghabi, F. and Coleman, W. (2012). A Scoping Review of Health Game Research: Past, Present, and Future. *Games for Health Journal*, 1(2), pp.153-164.

Kimura, S., Steinbach, G., Watenpaugh, D. and Hargens, A. (2001). Lumbar Spine Disc Height and Curvature Responses to an Axial Load Generated by a Compression Device Compatible with Magnetic Resonance Imaging. *Spine*, 26(23), pp.2596-2600.

Kochen, M., Blozik, E., Scherer, M. and Chenot, J. (2009). Imaging for low-back pain. *The Lancet*, 373(9662), pp.436-437.

Koes, B., van Tulder, M., Ostelo, R., Kim Burton, A. and Waddell, G. (2001). Clinical Guidelines for the Management of Low Back Pain in Primary Care. *Spine*, 26(22), pp.2504-2513.

Macfarlane, G., Thomas, E., Papageorgiou, A., Croft, P., Jayson, M. and Silman, A. (1997). Employment and Physical Work Activities as Predictors of Future Low Back Pain. *Spine*, 22(10), pp.1143-1149.

Michael, A. (2004). Biomechanics of back pain. *Acupuncture in Medicine*, 22(4), pp.178-188.

Picavet, H. (2002). Pain Catastrophizing and Kinesiophobia: Predictors of Chronic Low Back Pain. *American Journal of Epidemiology*, 156(11), pp.1028-1034.

Pitkänen, M. and Manninen, H. (1994). Sidebending versus flexion-extension radiographs in lumbar spinal instability. *Clinical Radiology*, 49(2), pp.109-114.

Pomero, V., Mitton, D., Laporte, S., de Guise, J. and Skalli, W. (2004). Fast accurate stereoradiographic 3D-reconstruction of the spine using a combined geometric and statistic model. *Clinical Biomechanics*, 19(3), pp.240-247.

Rubinstein, S., van Middelkoop, M., Assendelft, W., de Boer, M. and van Tulder, M. (2011). Spinal Manipulative Therapy for Chronic Low-Back Pain. *Spine*, 36(13), pp.E825-E846.

Schulze, M., Trautwein, F., Vordemvenne, T., Raschke, M. and Heuer, F. (2011). A method to perform spinal motion analysis from functional X-ray images. *Journal of Biomechanics*, 44(9), pp.1740-1746.

Simons, C., Cobb, L. and Davidson, B. (2013). A Fast, Accurate, and Reliable Reconstruction Method of the Lumbar Spine Vertebrae Using Positional MRI. *Annals of Biomedical Engineering*, 42(4), pp.833-842.

Taylor, M., Hipp, J., Gertzbein, S., Gopinath, S. and Reitman, C. (2007). Observer agreement in assessing flexion-extension X-rays of the cervical spine, with and without the use of quantitative measurements of intervertebral motion. *The Spine Journal*, 7(6), pp.654-658.

van Middelkoop, M., Rubinstein, S., Kuijpers, T., Verhagen, A., Ostelo, R., Koes, B. and van Tulder, M. (2010). A systematic review on the effectiveness of physical and rehabilitation interventions for chronic non-specific low back pain. *European Spine Journal*, 20(1), pp.19-39.

van Middelkoop, M., Rubinstein, S., Verhagen, A., Ostelo, R., Koes, B. and van Tulder, M. (2010). Exercise therapy for chronic nonspecific low-back pain. *Best Practice & Research Clinical Rheumatology*, 24(2), pp.193-204.

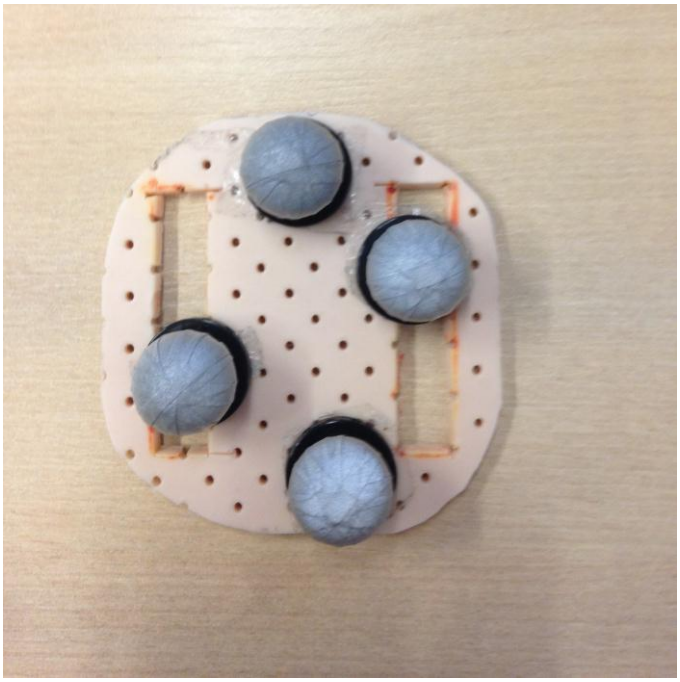
WADDELL, G. (1987). 1987 Volvo Award in Clinical Sciences: A New Clinical Model for the Treatment of Low-Back Pain. *Spine*, 12(7), pp.632-644.

Walsh, K., Cruddas, M. and Coggon, D. (1992). Low back pain in eight areas of Britain. *Journal of Epidemiology & Community Health*, 46(3), pp.227-230.

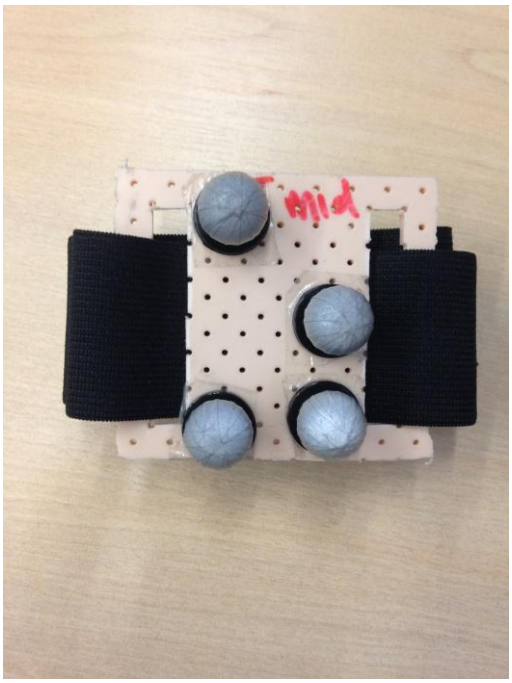
Wisleder, D., Werner, S., Kraemer, W., Fleck, S. and Zatsiorsky, V. (2001). A Method to Study Lumbar Spine Response to Axial Compression During Magnetic Resonance Imaging. *Spine*, 26(18), pp.E416-E420.

Appendices

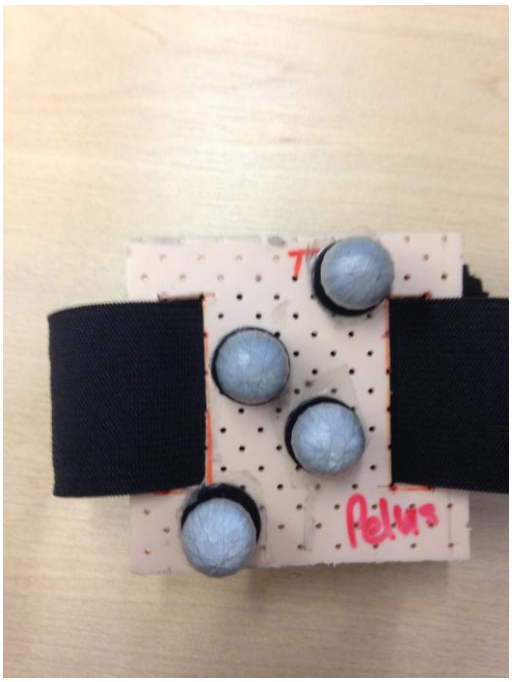
Appendix I: Complete clusters



Appendix I. 1 Upper back cluster

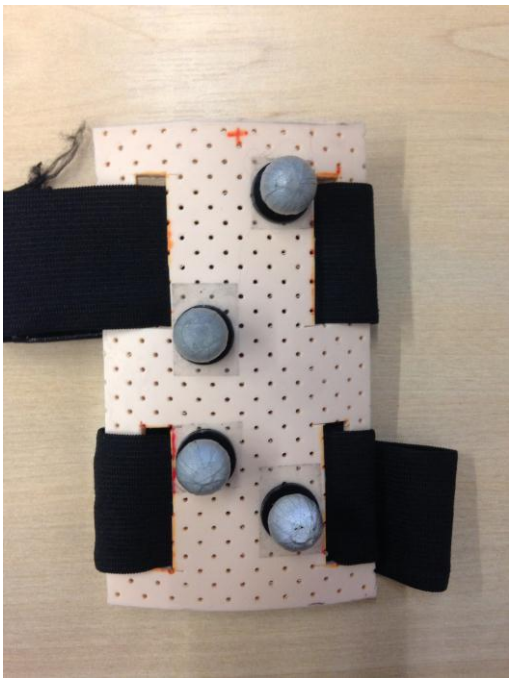


Appendix I. 2 Mid back cluster



Appendix I. 3 Pelvis cluster

Appendix I. 4 Left thigh cluster



Appendix I. 5 Right thigh cluster

Appendix II: Calibration cluster marker

```
-- Initialization of all (not local) variables
ini = ini or 0
input ={} or 0
aver={} or 0

-- Function definitions

--Initialization code

if ini == 0 then
-- initialization code here
markers=5
channels=markers*3
n=1
ini = 1
end

aver[0]=markers

for i = 1 , channels do
        aver[i]=inputs.get("Input"..i)
        end

-- Script update (all parts below are part of the script update)

n=n+1

for i = 1, channels do
        val1=aver[i]
        val2=inputs.get("Input"..i)
        aver[i]=((val1*(n-1))+val2)/n
        end
print ("frames recorded ", n)

-- Event handling
-- Input based variables
-- Application logic
-- Set Output


for i = 0 , channels do

outputs.set("Output"..i,aver[i])

end
```

Appendix III: Cluster tracking

```
o=o or 0

-- 1) Initialization of all (not local) variables
ini = ini or 0

length=length or {}
ordlen=ordlen or {}
c=c or {}

fc=fc or {}
flength=flength or {}
fordlen=fordlen or {}
fnear=fnear or {}

cmsum=cmsum or {}
cm1=cm1 or {}
cm2=cm2 or {}
cm3=cm3 or {}
cm4=cm4 or {}


markers=markers or {}

tolerance=0.03 -- tolerance is currently set to 3cm for all segments, this is too high but atm it helps tracking
markerson=1

outputs.setchannels("UB1x", "UB1y", "UB1z",
                                        "UB2x", "UB2y", "UB2z",
                                        "UB3x", "UB3y", "UB3z",
                                        "UB4x", "UB4y", "UB4z")

-- 3)Initialization code first frame ini==0
if ini == 0 then

        for j=1 , 4 do
                markers[j]=object.create("Sphere","Yellow")
                markers[j]:setscaling(0.03,0.03,0.03)
                markers[j]:setposition(-999,-999,-999)
        end

        local allinputs={}
                for i=1 ,122 do
                        allinputs[i]="input"..i
                end
        inputs.setchannels(unpack(allinputs))
```

```
	local alloutputs={}
		for i=1 ,120 do
			alloutputs[i]="output"..i
		end
	outputs.setchannels(unpack(alloutputs))

	sum=999

--
--load in cluster
--

	infname='J:\\David\\Clusters\\testupperBack002.txt'
	io.input(infname)
	nummarkers=io.read("*number")
	numcoords=nummarkers*3
	for i = 1 , numcoords do
		c[i]=io.read("*number")
	end

-- calculate inter marker lengths
	count=0
	for i=1 ,nummarkers-1 do
		for j=i+1, nummarkers do
			count=count+1
			length[count]=(c[((i-1)*3)+1]-c[((j-1)*3)+1])^2
			length[count]=length[count]+(c[((i-1)*3)+2]-c[((j-1)*3)+2])^2
			length[count]=length[count]+(c[((i-1)*3)+3]-c[((j-1)*3)+3])^2
			length[count]=length[count]^0.5
		end
	end

-- order lengths assending
	for i=1 , count do
		ordlen[i]=9999
		rank=0
		for j=1 , count do
			if length[j]<=ordlen[i] then
				ordlen[i]=length[j]
				rank=j
			end
		end
	length[rank]=1000
	end

	sum=ordlen[1]+ordlen[2]+ordlen[3]+ordlen[4]+ordlen[5]+ordlen[6]

	ini=1
```

```
end



------------------------------------------------------
-- 4) Script update on each frame
------------------------------------------------------

best=0
diff=9999
fnummarkers=40
fchannels=fnummarkers*3

--get marker data
for i = 1 , fchannels do
	fc[i]=inputs.get("input"..i)
end

--eliminate missing markers

for i=1, fnummarkers do
	if (fc[((i-1)*3)+1]+fc[((i-1)*3)+2]+fc[((i-1)*3)+3])==0 then
		fc[((i-1)*3)+1]=9999
		fc[((i-1)*3)+2]=9999
		fc[((i-1)*3)+3]=9999
	end
end

-- calculate inter marker lengths for each marker
-- in turn with each other marker

for k=1,fnummarkers do

	fnear[1] =k
--find the lengths to the other markers
	for i=1 ,fnummarkers do
		flength[i]=(fc[((k-1)*3)+1]-fc[((i-1)*3)+1])^2
		flength[i]=flength[i]+(fc[((k-1)*3)+2]-fc[((i-1)*3)+2])^2
		flength[i]=flength[i]+(fc[((k-1)*3)+3]-fc[((i-1)*3)+3])^2
		flength[i]=flength[i]^0.5
--if the same marker then make length large
		if (i==k) then flength[i]=9999
		end
	end


--order lengths to find closeest neighbours
	for i=2 , 4 do
```

```
                fordlen[i]=9999
                rank=9999
                for j=1 , fnummarkers do
                        if flength[j]<=fordlen[i] then
                        fordlen[i]=flength[j]
                        rank=j
                end
        end
        fnear[i]=rank
        flength[rank]=9999
end

-- calculate sum of lengths for that combination

count=0
fsum=0
        for i=1 ,4 do
                for j=i+1, 4 do
                        count=count+1
                        flen=(fc[((fnear[i]-1)*3)+1]-fc[((fnear[j]-1)*3)+1])^2
                        flen=flen+(fc[((fnear[i]-1)*3)+2]-fc[((fnear[j]-1)*3)+2])^2
                        flen=flen+(fc[((fnear[i]-1)*3)+3]-fc[((fnear[j]-1)*3)+3])^2
                        flen=flen^0.5
                        fsum=fsum+flen
                end
        end

        if (fsum==0) then
                fsum=999
        end

        cmsum[k]=fsum
        cm1[k]=fnear[1]
        cm2[k]=fnear[2]
        cm3[k]=fnear[3]
        cm4[k]=fnear[4]


        if ((((cmsum[k]-sum)^2)^0.5)<diff) then
                diff=(((cmsum[k]-sum)^2)^0.5)
                best=k
        end
end


if diff<=tolerance then

        if markerson==0 then
```

```
                markers[1]:setposition(-999,-999,-999)
                markers[2]:setposition(-999,-999,-999)
                markers[3]:setposition(-999,-999,-999)
                markers[4]:setposition(-999,-999,-999)

        else
                markers[1]:setposition(fc[((cm1[best]-1)*3)+1],fc[((cm1[best]-
1)*3)+2],fc[((cm1[best]-1)*3)+3])
                markers[2]:setposition(fc[((cm2[best]-1)*3)+1],fc[((cm2[best]-
1)*3)+2],fc[((cm2[best]-1)*3)+3])
                markers[3]:setposition(fc[((cm3[best]-1)*3)+1],fc[((cm3[best]-
1)*3)+2],fc[((cm3[best]-1)*3)+3])
                markers[4]:setposition(fc[((cm4[best]-1)*3)+1],fc[((cm4[best]-
1)*3)+2],fc[((cm4[best]-1)*3)+3])
        end
else
        markers[1]:setposition(-999,-999,-999)
        markers[2]:setposition(-999,-999,-999)
        markers[3]:setposition(-999,-999,-999)
        markers[4]:setposition(-999,-999,-999)
end

-- Set Output

if diff<=tolerance then
        outputs.set(1,fc[((cm1[best]-1)*3)+1])
        outputs.set(2,fc[((cm1[best]-1)*3)+2])
        outputs.set(3,fc[((cm1[best]-1)*3)+3])
        outputs.set(4,fc[((cm2[best]-1)*3)+1])
        outputs.set(5,fc[((cm2[best]-1)*3)+2])
        outputs.set(6,fc[((cm2[best]-1)*3)+3])
        outputs.set(7,fc[((cm3[best]-1)*3)+1])
        outputs.set(8,fc[((cm3[best]-1)*3)+2])
        outputs.set(9,fc[((cm3[best]-1)*3)+3])
        outputs.set(10,fc[((cm4[best]-1)*3)+1])
        outputs.set(11,fc[((cm4[best]-1)*3)+2])
        outputs.set(12,fc[((cm4[best]-1)*3)+3])

else
        outputs.set(1,-999)
        outputs.set(2,-999)
        outputs.set(3,-999)
        outputs.set(4,-999)
        outputs.set(5,-999)
        outputs.set(6,-999)
        outputs.set(7,-999)
        outputs.set(8,-999)
        outputs.set(9,-999)
```

```
        outputs.set(10,-999)
        outputs.set(11,-999)
        outputs.set(12,-999)

end
```

Appendix IV: Cluster labelling

```
require "labelling002"

-- Initialisation of variables
ini = ini or 0
markers = markers or {}
allinputs = allinputs or {}
nrInputs = 60

outputs.setchannels("UBM1x", "UBM1y", "UBM1z",
                                                    "UBM2x",        "UBM2y",
"UBM2z",
                                                    "UBM3x",    "UBM3y",
"UBM3z",
                                                    "UBM4x",        "UBM4y",
"UBM4z",
                                                    "MBM1x",        "MBM1y",
"MBM1z",
                                                    "MBM2x",        "MBM2y",
"MBM2z",
                                                    "MBM3x",    "MBM3y",
"MBM3z",
                                                    "MBM4x",        "MBM4y",
"MBM4z",
                                                    "PLM1x", "PLM1y", "PLM1z",
                                                    "PLM2x", "PLM2y", "PLM2z",
                                                    "PLM3x",    "PLM3y",
"PLM3z",
                                                    "PLM4x", "PLM4y", "PLM4z",
                                                    "LTM1x", "LTM1y", "LTM1z",
                                                    "LTM2x", "LTM2y", "LTM2z",
                                                    "LTM3x",    "LTM3y",
"LTM3z",
                                                    "LTM4x", "LTM4y", "LTM4z",
                                                    "RTM1x", "RTM1y", "RTM1z",
                                                    "RTM2x", "RTM2y", "RTM2z",
                                                    "RTM3x",    "RTM3y",
"RTM3z",
                                                    "RTM4x",        "RTM4y",
"RTM4z")
```

```
-- Initialisation code
if ini == 0 then
	for i = 1, nrInputs do
		allinputs[i] = "Channel"..i
	end
	inputs.setchannels(unpack(allinputs))
	ini = 1
end

i = 1
j = 1
while i < #allinputs do
	markers[j] = {}
	markers[j]["x"] = inputs.get("Channel"..i)
	markers[j]["y"] = inputs.get("Channel"..i+1)
	markers[j]["z"] = inputs.get("Channel"..i+2)
	i = i+3
	j = j+1
end

upperBack = {}
midBack = {}
pelvis = {}
leftThigh = {}
rightThigh = {}

for i = 1, 4 do
	upperBack[i] = i
end

for i = 1, 4 do
	midBack[i] = i+4
end

for i =1, 4 do
	pelvis[i] = i+8
end

for i =1, 4 do
	leftThigh[i] = i+12
end

for i =1, 4 do
	rightThigh[i] = i+16
end

UBmarkers = labelUBMarkers(upperBack)
MBmarkers = labelMBMarkers(midBack)
```

```
PLmarkers = labelPLMarkers(pelvis)
LTmarkers = labelLTMarkers(leftThigh)
RTmarkers = labelRTMarkers(rightThigh)


outputs.set("UBM1x", markers[UBmarkers[1]]["x"])
outputs.set("UBM1y", markers[UBmarkers[1]]["y"])
outputs.set("UBM1z", markers[UBmarkers[1]]["z"])
outputs.set("UBM2x", markers[UBmarkers[2]]["x"])
outputs.set("UBM2y", markers[UBmarkers[2]]["y"])
outputs.set("UBM2z", markers[UBmarkers[2]]["z"])
outputs.set("UBM3x", markers[UBmarkers[3]]["x"])
outputs.set("UBM3y", markers[UBmarkers[3]]["y"])
outputs.set("UBM3z", markers[UBmarkers[3]]["z"])
outputs.set("UBM4x", markers[UBmarkers[4]]["x"])
outputs.set("UBM4y", markers[UBmarkers[4]]["y"])
outputs.set("UBM4z", markers[UBmarkers[4]]["z"])
outputs.set("MBM1x", markers[MBmarkers[1]]["x"])
outputs.set("MBM1y", markers[MBmarkers[1]]["y"])
outputs.set("MBM1z", markers[MBmarkers[1]]["z"])
outputs.set("MBM2x", markers[MBmarkers[2]]["x"])
outputs.set("MBM2y", markers[MBmarkers[2]]["y"])
outputs.set("MBM2z", markers[MBmarkers[2]]["z"])
outputs.set("MBM3x", markers[MBmarkers[3]]["x"])
outputs.set("MBM3y", markers[MBmarkers[3]]["y"])
outputs.set("MBM3z", markers[MBmarkers[3]]["z"])
outputs.set("MBM4x", markers[MBmarkers[4]]["x"])
outputs.set("MBM4y", markers[MBmarkers[4]]["y"])
outputs.set("MBM4z", markers[MBmarkers[4]]["z"])
outputs.set("PLM1x", markers[PLmarkers[1]]["x"])
outputs.set("PLM1y", markers[PLmarkers[1]]["y"])
outputs.set("PLM1z", markers[PLmarkers[1]]["z"])
outputs.set("PLM2x", markers[PLmarkers[2]]["x"])
outputs.set("PLM2y", markers[PLmarkers[2]]["y"])
outputs.set("PLM2z", markers[PLmarkers[2]]["z"])
outputs.set("PLM3x", markers[PLmarkers[3]]["x"])
outputs.set("PLM3y", markers[PLmarkers[3]]["y"])
outputs.set("PLM3z", markers[PLmarkers[3]]["z"])
outputs.set("PLM4x", markers[PLmarkers[4]]["x"])
outputs.set("PLM4y", markers[PLmarkers[4]]["y"])
outputs.set("PLM4z", markers[PLmarkers[4]]["z"])
outputs.set("LTM1x", markers[LTmarkers[1]]["x"])
outputs.set("LTM1y", markers[LTmarkers[1]]["y"])
outputs.set("LTM1z", markers[LTmarkers[1]]["z"])
outputs.set("LTM2x", markers[LTmarkers[2]]["x"])
outputs.set("LTM2y", markers[LTmarkers[2]]["y"])
outputs.set("LTM2z", markers[LTmarkers[2]]["z"])
outputs.set("LTM3x", markers[LTmarkers[3]]["x"])
```

```
outputs.set("LTM3y", markers[LTmarkers[3]]["y"])
outputs.set("LTM3z", markers[LTmarkers[3]]["z"])
outputs.set("LTM4x", markers[LTmarkers[4]]["x"])
outputs.set("LTM4y", markers[LTmarkers[4]]["y"])
outputs.set("LTM4z", markers[LTmarkers[4]]["z"])
outputs.set("RTM1x", markers[RTmarkers[1]]["x"])
outputs.set("RTM1y", markers[RTmarkers[1]]["y"])
outputs.set("RTM1z", markers[RTmarkers[1]]["z"])
outputs.set("RTM2x", markers[RTmarkers[2]]["x"])
outputs.set("RTM2y", markers[RTmarkers[2]]["y"])
outputs.set("RTM2z", markers[RTmarkers[2]]["z"])
outputs.set("RTM3x", markers[RTmarkers[3]]["x"])
outputs.set("RTM3y", markers[RTmarkers[3]]["y"])
outputs.set("RTM3z", markers[RTmarkers[3]]["z"])
outputs.set("RTM4x", markers[RTmarkers[4]]["x"])
outputs.set("RTM4y", markers[RTmarkers[4]]["y"])
outputs.set("RTM4z", markers[RTmarkers[4]]["z"])
```

Appendix V: Labelling002

```
function getEuclideanDistance(p1, p2)
                        local euclideanDistance = (((p1["x"] - p2["x"])^2) + ((p1["y"]
- p2["y"])^2) + ((p1["z"] - p2["z"])^2))^0.5
        return euclideanDistance
end



function sortArrayAscending(A)
        for i = 1, #A do
                for j = i+1, #A do
                        if A[i][1] > A[j][1] then
                                temp = A[j]
                                A[j] = A[i]
                                A[i] = temp
                        end
                end
        end
        return A
end

function createEDarray (B)
        local A = {}
        A[1] ={getEuclideanDistance(markers[B[1]], markers[B[2]]), B[1], B[2]}
        A[2] ={getEuclideanDistance(markers[B[1]], markers[B[3]]), B[1], B[3]}
        A[3] ={getEuclideanDistance(markers[B[1]], markers[B[4]]), B[1], B[4]}
```

```
        A[4] ={getEuclideanDistance(markers[B[2]], markers[B[3]]), B[2], B[3]}
        A[5] ={getEuclideanDistance(markers[B[2]], markers[B[4]]), B[2], B[4]}
        A[6] ={getEuclideanDistance(markers[B[3]], markers[B[4]]), B[3], B[4]}
return A
end

function findFirstMarker(A, p1, p2)
        if A[1][2] == A[p1][2] or A[1][2] == A[p1][3] or A[1][2] == A[p2][2] or
A[1][2] == A[p2][3] then
                return A[1][2]
        elseif
                A[1][3] == A[p1][2] or A[1][3] == A[p1][3] or A[1][3] == A[p2][2]
or A[1][3] == A[p2][3] then
                return A[1][3]
        end
end

function findSecondMarker(A, p1)
        if p1 ~= 9999 then
                if A[1][2] == p1 then
                        return A[1][3]
                elseif
                        A[1][3] == p1 then
                        return A[1][2]
                end
        end
end

function findOtherMarker(A, p1, p2)
        if p2 ~= 9999 then
                if A[p1][2] == p2 then
                        return A[p1][3]
                elseif
                        A[p1][3] == p2 then
                        return A[p1][2]
                end
        end
end

function labelLTMarkers(segmentMarkers)
                --for i = 1, #segmentMarkers do
                        --if segmentMarkers[i] <= #markers then
                                local segmentEDarray = {}
                                segmentEDarray = createEDarray(segmentMarkers)
                                sortArrayAscending(segmentEDarray)
                                --for i = 1, #segmentEDarray do
                                        --print(unpack(segmentEDarray[i]))
                                --end
```

```
                        local markerName = {}
                        markerName[1] = findFirstMarker(segmentEDarray, 4,
6)
                        if markerName[1] == nil then
                                markerName[1] = 9999
                        end
                        markerName[2]                                   =
findSecondMarker(segmentEDarray, markerName[1])
                        if markerName[3] == nil then
                                markerName[3] = 9999
                        end
                        markerName[3]  =  findOtherMarker(segmentEDarray,
4, markerName[1])
                        if markerName[2] == nil then
                                markerName[2] = 9999
                        end
                        markerName[4]  =  findOtherMarker(segmentEDarray,
6, markerName[1])
                        if markerName[4] == nil then
                                markerName[4] = 9999
                        end
                --end
        --end
return markerName
end

function labelRTMarkers(segmentMarkers)
        segmentEDarray = {}
        segmentEDarray = createEDarray(segmentMarkers)
        segmentEDarray = sortArrayAscending(segmentEDarray)
        --for i = 1, #segmentEDarray do
                --print(unpack(segmentEDarray[i]))
        --end
        markerName = {}
        markerName[3]  =  findFirstMarker(segmentEDarray,  2,  5)   --First
marker must be part of the shortest length between markers--
                if markerName[3] == nil then --markerName is the number of
the marker that is being looked at--
                        markerName[3] = 9999
                end
        markerName[4]          =          findSecondMarker(segmentEDarray,
markerName[3])
                if markerName[4] == nil then
                        markerName[4] = 9999
                end
        markerName[1]       =       findOtherMarker(segmentEDarray,       5,
markerName[3])
                if markerName[1] == nil then
```

```
                markerName[1] = 9999
            end
        markerName[2] = findOtherMarker(segmentEDarray, 2,
markerName[3])
            if markerName[2] == nil then
                markerName[2] = 9999
            end
return markerName
end

function labelPLMarkers(segmentMarkers)
        segmentEDarray = {}
        segmentEDarray = createEDarray(segmentMarkers)
        segmentEDarray = sortArrayAscending(segmentEDarray)
        --for i = 1, #segmentEDarray do
            --print(unpack(segmentEDarray[i]))
        --end
        markerName = {}
        markerName[3] = findFirstMarker(segmentEDarray, 4, 3)
            if markerName[4] == nil then
                markerName[4] = 9999
            end
        markerName[2] = findSecondMarker(segmentEDarray,
markerName[3])
            if markerName[2] == nil then
                markerName[2] = 9999
            end
        markerName[1] = findOtherMarker(segmentEDarray, 4,
markerName[3])
            if markerName[1] == nil then
                markerName[1] = 9999
            end
        markerName[4] = findOtherMarker(segmentEDarray, 3,
markerName[3])
            if markerName[4] == nil then
                markerName[4] = 9999
            end
return markerName
end

function labelMBMarkers(segmentMarkers)
        segmentEDarray = {}
        segmentEDarray = createEDarray(segmentMarkers)
        segmentEDarray = sortArrayAscending(segmentEDarray)
        --for i = 1, #segmentEDarray do
            --print(unpack(segmentEDarray[i]))
        --end
        markerName = {}
```

```
		markerName[3] = findFirstMarker(segmentEDarray, 6, 2)
			if markerName[3] == nil then
				markerName[3] = 9999
			end
		markerName[4] = findSecondMarker(segmentEDarray, markerName[3])
			if markerName[4] == nil then
				markerName[4] = 9999
			end
		markerName[1] = findOtherMarker(segmentEDarray, 6, markerName[3])
			if markerName[1] == nil then
				markerName[1] = 9999
			end
		markerName[2] = findOtherMarker(segmentEDarray, 2, markerName[3])
			if markerName[2] == nil then
				markerName[2] = 9999
			end
return markerName
end

function labelUBMarkers(segmentMarkers)
		segmentEDarray = {}
		segmentEDarray = createEDarray(segmentMarkers)
		segmentEDarray = sortArrayAscending(segmentEDarray)
		--for i = 1, #segmentEDarray do
			--print(unpack(segmentEDarray[i]))
		--end
		markerName = {}
		markerName[1] = findFirstMarker(segmentEDarray, 4, 6)
			if markerName[1] == nil then
				markerName[1] = 9999
			end
		markerName[2] = findSecondMarker(segmentEDarray, markerName[1])
			if markerName[2] == nil then
				markerName[2] = 9999
			end
		markerName[3] = findOtherMarker(segmentEDarray, 4, markerName[1])
			if markerName[3] == nil then
				markerName[3] = 9999
			end
		markerName[4] = findOtherMarker(segmentEDarray, 6, markerName[1])
			if markerName[4] == nil then
				markerName[4] = 9999
```

```
                    end
return markerName
end
```

Appendix VI: Cluster centroid

```
--Initilisation of varables
ini = ini or 0
allinputs = allinputs or {}
input = input or {}
shapes = shapes or {"Sphere", "Cube", "Cylinder", "Cone"}
colours = colours or {"Red", "Green", "Blue", "White", "Gray", "White", "Cyan"}
marker = marker or {}

outputs.setchannels("UBX", "UBY", "UBZ",
                                    "MBX", "MBY", "MBZ",
                                    "PLX", "PLY", "PLZ",
                                    "LTX", "LTY", "LTZ",
                                    "RTX", "RTY", "RTZ")

--Initilisation Code
if ini == 0 then

for i = 1, 60 do
        allinputs[i] = "Channel"..i
end

inputs.setchannels(unpack(allinputs))

ini = 1
end

for i = 1, 60 do
        input[i] = inputs.get("Channel"..i)
end

---UB Cluster---

UBXmax = math.max(input[1], input[4], input[7], input[10])
UBXmin = math.min(input[1], input[4], input[7], input[10])
UBX = (UBXmax + UBXmin)/2
print(UBXmax, UBXmin, UBX)

UBYmax = math.max(input[2], input[5], input[8], input[11])
UBYmin = math.min(input[2], input[5], input[8], input[11])
UBY = (UBYmax + UBYmin)/2
print(UBYmax, UBYmin, UBY)
```

```
UBZmax = math.max(input[3], input[6], input[9], input[12])
UBZmin = math.min(input[3], input[6], input[9], input[12])
UBZ = (UBZmax + UBZmin)/2
print(UBZmax, UBZmin, UBZ)

outputs.set("UBX", UBX)
outputs.set("UBY", UBY)
outputs.set("UBZ", UBZ)

---MB Cluster---

MBXmax = math.max(input[13], input[16], input[19], input[22])
MBXmin = math.min(input[13], input[16], input[19], input[22])
MBX = (MBXmax + MBXmin)/2
print(MBXmax, MBXmin, MBX)

MBYmax = math.max(input[14], input[17], input[20], input[23])
MBYmin = math.min(input[14], input[17], input[20], input[23])
MBY = (MBYmax + MBYmin)/2
print(MBYmax, MBYmin, MBY)

MBZmax = math.max(input[15], input[18], input[21], input[24])
MBZmin = math.min(input[15], input[18], input[21], input[24])
MBZ = (MBZmax + MBZmin)/2
print(MBZmax, MBZmin, MBZ)

outputs.set("MBX", MBX)
outputs.set("MBY", MBY)
outputs.set("MBZ", MBZ)



---PL Cluster---

PLXmax = math.max(input[25], input[28], input[31], input[34])
PLXmin = math.min(input[25], input[28], input[31], input[34])
PLX = (PLXmax + PLXmin)/2
print(PLXmax, PLXmin, PLX)

PLYmax = math.max(input[26], input[29], input[32], input[35])
PLYmin = math.min(input[26], input[29], input[32], input[35])
PLY = (PLYmax + PLYmin)/2
print(PLYmax, PLYmin, PLY)

PLZmax = math.max(input[27], input[30], input[33], input[36])
PLZmin = math.min(input[27], input[30], input[33], input[36])
PLZ = (PLZmax + PLZmin)/2
```

```
print(PLZmax, PLZmin, PLZ)

outputs.set("PLX", PLX)
outputs.set("PLY", PLY)
outputs.set("PLZ", PLZ)



---LT Cluster---

LTXmax = math.max(input[37], input[40], input[43], input[46])
LTXmin = math.min(input[37], input[40], input[43], input[46])
LTX = (LTXmax + LTXmin)/2
print(LTXmax, LTXmin, LTX)

LTYmax = math.max(input[38], input[41], input[44], input[47])
LTYmin = math.min(input[38], input[41], input[44], input[47])
LTY = (LTYmax + LTYmin)/2
print(LTYmax, LTYmin, LTY)

LTZmax = math.max(input[39], input[42], input[45], input[48])
LTZmin = math.min(input[39], input[42], input[45], input[48])
LTZ = (LTZmax + LTZmin)/2
print(LTZmax, LTZmin, LTZ)

outputs.set("LTX", LTX)
outputs.set("LTY", LTY)
outputs.set("LTZ", LTZ)



---RT Cluster---

RTXmax = math.max(input[49], input[52], input[55], input[58])
RTXmin = math.min(input[49], input[52], input[55], input[58])
RTX = (RTXmax + RTXmin)/2
print(RTXmax, RTXmin, RTX)

RTYmax = math.max(input[50], input[53], input[56], input[59])
RTYmin = math.min(input[50], input[53], input[56], input[59])
RTY = (RTYmax + RTYmin)/2
print(RTYmax, RTYmin, RTY)

RTZmax = math.max(input[51], input[54], input[57], input[60])
RTZmin = math.min(input[51], input[54], input[57], input[60])
RTZ = (RTZmax + RTZmin)/2
print(RTZmax, RTZmin, RTZ)
```

```
outputs.set("RTX", RTX)
outputs.set("RTY", RTY)
outputs.set("RTZ", RTZ)
```

Appendix VII: Cluster comparison

```
--Initilisation of varables
ini = ini or 0
allinputs = allinputs or {}
input = input or {}
shapes = shapes or {"Sphere", "Cube", "Cylinder", "Cone"}
colours = colours or {"Red", "Green", "Blue", "White", "Gray", "White", "Cyan", "Yellow"}
marker = marker or {}

outputs.setchannels("UpperBack", "MidBack", "Pelvis", "LeftThigh", "RightThight")

markerson = 1

--Initilisation Code
if ini == 0 then

for i = 1, 30 do
        allinputs[i] = "Channel"..i
end

inputs.setchannels(unpack(allinputs))

--Joint centres
object.create(shapes[1], colours[6]):setscaling(0.03, 0.03, 0.03) --1 PLLT
object.create(shapes[1], colours[1]):setscaling(0.03, 0.03, 0.03)   --2 Red
object.create(shapes[1], colours[8]):setscaling(0.03, 0.03, 0.03)   --3 Yellow
object.create(shapes[1], colours[2]):setscaling(0.03, 0.03, 0.03)   --4 Green

object.create(shapes[1], colours[6]):setscaling(0.03, 0.03, 0.03) --5 PLRT
object.create(shapes[1], colours[1]):setscaling(0.03, 0.03, 0.03)   --6 Red
object.create(shapes[1], colours[8]):setscaling(0.03, 0.03, 0.03)   --7 Yellow
object.create(shapes[1], colours[2]):setscaling(0.03, 0.03, 0.03)   --8 Green

object.create(shapes[1], colours[6]):setscaling(0.03, 0.03, 0.03) --9 PLMB
object.create(shapes[1], colours[1]):setscaling(0.03, 0.03, 0.03)   --10 Red
object.create(shapes[1], colours[8]):setscaling(0.03, 0.03, 0.03)   --11 Yellow
object.create(shapes[1], colours[2]):setscaling(0.03, 0.03, 0.03)   --12 Green

object.create(shapes[1], colours[6]):setscaling(0.03, 0.03, 0.03) --13 MBUB
object.create(shapes[1], colours[1]):setscaling(0.03, 0.03, 0.03)   --14 Red
object.create(shapes[1], colours[8]):setscaling(0.03, 0.03, 0.03)   --15 Yellow
object.create(shapes[1], colours[2]):setscaling(0.03, 0.03, 0.03)   --16 Green

object.create(shapes[1], colours[6]):setscaling(0.06, 0.06, 0.06) --17 PL
object.create(shapes[1], colours[1]):setscaling(0.06, 0.06, 0.06)   --18 Red
object.create(shapes[1], colours[8]):setscaling(0.06, 0.06, 0.06)   --19 Yellow
```

```
object.create(shapes[1], colours[2]):setscaling(0.06, 0.06, 0.06)   --20 Green

ini = 1
end

for i = 1, 30 do
        input[i] = inputs.get("Channel"..i)
end

---Live Marker Positions---

--PLLT--

objects.get(1):setposition(input[25],input[26],input[27])

--PLRT

objects.get(5):setposition(input[28],input[29],input[30])

--PLMB

objects.get(9):setposition(input[19],input[20],input[21])

--MBUB--

objects.get(13):setposition(input[16],input[17],input[18])

--Pelvis--

objects.get(17):setposition(input[22],input[23],input[24])

----Recording Distances----

RPLLTabs = ((input[7]-input[10])^2 + (input[8]-input[11])^2 + (input[9]-input[12])^2)^0.5
RPLRTabs = ((input[7]-input[13])^2 + (input[8]-input[14])^2 + (input[9]-input[15])^2)^0.5
RPLMBabs = ((input[7]-input[4])^2 + (input[8]-input[5])^2 + (input[9]-input[6])^2)^0.5
RMBUBabs = ((input[1]-input[4])^2 + (input[2]-input[5])^2 + (input[3]-input[6])^2)^0.5

--print(RUBabs, RMBabs, RPLabs, RLTabs, RRTabs)

----Live Absolute Values----

LPLLTabs = ((input[22]-input[25])^2 + (input[23]-input[26])^2 + (input[24]-
input[27])^2)^0.5
LPLRTabs = ((input[22]-input[28])^2 + (input[23]-input[29])^2 + (input[24]-
input[30])^2)^0.5
LPLMBabs = ((input[22]-input[19])^2 + (input[23]-input[20])^2 + (input[24]-
input[21])^2)^0.5
LMBUBabs = ((input[16]-input[19])^2 + (input[17]-input[20])^2 + (input[18]-
input[21])^2)^0.5


--print(LUBabs, LMBabs, LPLabs, LLTabs, LRTabs)
```

```
----Compare the two together----

TPLLTabs = ((RPLLTabs - LPLLTabs)^2)^0.5
TPLRTabs = ((RPLRTabs - LPLRTabs)^2)^0.5
TPLMBabs = ((RPLMBabs - LPLMBabs)^2)^0.5
TMBUBabs = ((RMBUBabs - LMBUBabs)^2)^0.5

if markerson== 0 then
	for i = 1,20 do
		objects.get(i):setposition(-999,-999,-999)
	end
	else


---Pelvis to Left thigh---

if TPLLTabs <= 1 then -- White
	do objects.get(1):setposition(input[25], input[26], input[27])
		objects.get(2):setposition(-999,-999,-999)
		objects.get(3):setposition(-999,-999,-999)
		objects.get(4):setposition(-999,-999,-999)
VFBLT = 0
		if TPLLTabs <= 0.3 then -- Red
			do objects.get(1):setposition(-999,-999,-999)
				objects.get(2):setposition(input[25], input[26], input[27])
				objects.get(3):setposition(-999,-999,-999)
				objects.get(4):setposition(-999,-999,-999)
	VFBLT = 0
				if TPLLTabs <= 0.1 then --Yellow
					do objects.get(1):setposition(-999,-999,-999)
						objects.get(2):setposition(-999,-999,-999)
						objects.get(3):setposition(input[25],
input[26], input[27])
						objects.get(4):setposition(-999,-999,-999)
		VFBLT = 0
						if TPLLTabs <= 0.025 then --Green
							do  objects.get(1):setposition(-999,-
999,-999)
								objects.get(2):setposition(-
999,-999,-999)
								objects.get(3):setposition(-
999,-999,-999)

	objects.get(4):setposition(input[25], input[26], input[27])
								VFBLT = 1
							end
						end
					end
				end
			end
		end
	end
```

```
end

--Peliva to Right thigh---

if TPLRTabs <= 1 then -- White
	do objects.get(5):setposition(input[28], input[29], input[30])
		objects.get(6):setposition(-999,-999,-999)
		objects.get(7):setposition(-999,-999,-999)
		objects.get(8):setposition(-999,-999,-999)
VFBRT = 0
		if TPLRTabs <= 0.3 then -- Red
			do objects.get(5):setposition(-999,-999,-999)
				objects.get(6):setposition(input[28], input[29], input[30])
				objects.get(7):setposition(-999,-999,-999)
				objects.get(8):setposition(-999,-999,-999)
		VFBRT = 0
				if TPLRTabs <= 0.1 then --Yellow
					do objects.get(5):setposition(-999,-999,-999)
						objects.get(6):setposition(-999,-999,-999)
						objects.get(7):setposition(input[28],
input[29], input[30])
						objects.get(8):setposition(-999,-999,-999)
					VFBRT = 0
						if TPLRTabs <= 0.025 then --Green
							do  objects.get(5):setposition(-999,-
999,-999)
								objects.get(6):setposition(-
999,-999,-999)
								objects.get(7):setposition(-
999,-999,-999)

	objects.get(8):setposition(input[28], input[29], input[30])
								VFBRT = 1
							end
						end
					end
				end
			end
		end
	end
end

---Pelvis to Mid Back---

if TPLMBabs <= 1 then -- White
	do objects.get(9):setposition(input[19], input[20], input[21])
		objects.get(10):setposition(-999,-999,-999)
		objects.get(11):setposition(-999,-999,-999)
		objects.get(12):setposition(-999,-999,-999)
VFBMB = 0
		if TPLMBabs <= 0.3 then -- Red
			do objects.get(9):setposition(-999,-999,-999)
				objects.get(10):setposition(input[19], input[20], input[21])
```

```
                                objects.get(11):setposition(-999,-999,-999)
                                objects.get(12):setposition(-999,-999,-999)
                        VFBMB = 0
                                if TPLMBabs <= 0.1 then --Yellow
                                        do objects.get(9):setposition(-999,-999,-999)
                                                objects.get(10):setposition(-999,-999,-999)
                                                objects.get(11):setposition(input[19],
input[20], input[21])
                                                objects.get(12):setposition(-999,-999,-999)
                                VFBMB = 0
                                                if TPLMBabs <= 0.025 then --Green
                                                        do  objects.get(9):setposition(-999,-
999,-999)

        objects.get(10):setposition(-999,-999,-999)

        objects.get(11):setposition(-999,-999,-999)

        objects.get(12):setposition(input[19], input[20], input[21])
                                                                VFBMB = 1
                                                end
                                        end
                                end
                        end
                end
        end
end

---Mid Back to Upper Back---

if TMBUBabs <= 1 then -- White
        do objects.get(13):setposition(input[16], input[17], input[18])
                objects.get(14):setposition(-999,-999,-999)
                objects.get(15):setposition(-999,-999,-999)
                objects.get(16):setposition(-999,-999,-999)
VFBUB = 0
                if TMBUBabs <= 0.3 then -- Red
                        do objects.get(13):setposition(-999,-999,-999)
                                objects.get(14):setposition(input[16], input[17], input[18])
                                objects.get(15):setposition(-999,-999,-999)
                                objects.get(16):setposition(-999,-999,-999)
                VFBUB = 0
                                if TMBUBabs <= 0.1 then --Yellow
                                        do objects.get(13):setposition(-999,-999,-999)
                                                objects.get(14):setposition(-999,-999,-999)
                                                objects.get(15):setposition(input[16],
input[17], input[18])
                                                objects.get(16):setposition(-999,-999,-999)
                                 VFBUB = 0
                                                if TMBUBabs <= 0.025 then --Green
                                                        do      objects.get(13):setposition(-
999,-999,-999)
```

```
        objects.get(14):setposition(-999,-999,-999)

        objects.get(15):setposition(-999,-999,-999)

        objects.get(16):setposition(input[16], input[17], input[18])
                                                                VFBUB = 1
                                                            end
                                                        end
                                                    end
                                                end
                                            end
                                        end
                                    end
                                end
                            end
                        end
end

---Visual feedback---

VFB = (VFBUB + VFBMB + VFBRT + VFBLT)

if VFB == 0 then -- Nothing
        do objects.get(17):setposition(-999,-999,-999)
                objects.get(18):setposition(-999,-999,-999)
                objects.get(19):setposition(-999,-999,-999)
                objects.get(20):setposition(-999,-999,-999)
        end
end

if VFB == 1 then -- White
        do objects.get(17):setposition(input[22], input[23], input[24])
                objects.get(18):setposition(-999,-999,-999)
                objects.get(19):setposition(-999,-999,-999)
                objects.get(20):setposition(-999,-999,-999)
        end
end

if VFB == 2 then -- Red
        do objects.get(17):setposition(-999,-999,-999)
                objects.get(18):setposition(input[22], input[23], input[24])
                objects.get(19):setposition(-999,-999,-999)
                objects.get(20):setposition(-999,-999,-999)
        end
end

if VFB == 3 then --Yellow
        do objects.get(17):setposition(-999,-999,-999)
                objects.get(18):setposition(-999,-999,-999)
                objects.get(19):setposition(input[22], input[23], input[24])
                objects.get(20):setposition(-999,-999,-999)
        end
end

if VFB == 4 then --Green
```

```
        do objects.get(17):setposition(-999,-999,-999)
                objects.get(18):setposition(-999,-999,-999)
                objects.get(19):setposition(-999,-999,-999)
                objects.get(20):setposition(input[22], input[23], input[24])
        end
end

end
```