UNIVERSITY OF STRATHCLYDE

FACULTY OF ENGINEERING

DEPARTMENT OF BIOMEDICAL ENGINEERING

MASTERS THESIS

TITLE:

DEVELOPING D-FLOW APPLICATIONS FOR MEASURING PARAMETERS AND VISUALISING GEOMETRY OF SPINE OF PATIENTS WITH SCOLIOSIS

Project Supervisor: *Professor Phillip Rowe*

SUBMITTED BY:

Student: *Edwin Andione Khundi*

Registration Number: *201486963*

SUBMITTED AS PARTIAL FULFILLMENT OF MASTERS OF BIOMEDICAL ENGINEERING

DUE DATE: September, 2015.

COPYRIGHT

ACKNOWLEDGEMENTS

I would like to thank my supervisor, Professor Phil Rowe, for his advice and support during the entire project period.

I would like also to thank Sun Jang for her pilot work on the protocol of using 40 markers on when capturing coordinates of markers on the body of patient with scoliosis.

I also give thanks to Malawi University of Science and Technology for sponsoring support and the biomedical engineering department staff at University of Strathclyde for their project support when approached.

Besides I extend my gratitude by thanking my friends, mentors and family for motivational words and different pieces of advice.

DEDICATION

Dedicated to all patients with scoliosis and spine related injuries or diseases!

Also dedicated to Dr. Moses Chinyama, Professor Elizabeth Molyneux, my project supervisor, family and friends

TABLE OF CONTENTS

Contents

ABSTRACT

The knowledge of geometry of spine is quite useful in treatment of various diseases of the spine such as scoliosis. Clinicians usually obtain geometry of spine or vertebrae column using radiology although other imaging techniques such as computed tomography, ultrasound, nuclear imaging and magnetic resonance imaging (MRI) are also used. The main limitation of radiography is that it is two dimensional while spine geometry of scoliosis is three dimensional. Thus the main aim of this project was to develop a D-flow based application that will calculate and visualise three dimensional shape of the spine using motion capture system. Three applications namely; pointing wand calibration, marker dependent and marker independent applications, were developed in D-flow software during the project. The applications were then tested by visualising geometry of skeleton and S- shaped cable. The marker dependent application was tested by attaching passive Infrared markers to the skeleton while for the marker independent no marker was attached to the skeleton. For the marker dependent application, the markers attached on skeleton were labelled in a particular sequence by pointing the wand close to a marker and pressing button of phidget. After labelling all markers the software visualised the spine automatically. For the marker independent application the geometry of the spine was obtained by pointing the wand at each anatomical land mark of vertebrae and pressing button of phidget to capture coordinates of that position. The protocol of labelling vertebrae for marker independent starts from cervical region up to sacral region. The coordinates of the labelled points were used to calculate different parameters and reconstruct a visualisation of the spine. The numerical results of parameters such as angle parameters from both applications and also from manual measurements were analysed and compared using statistical software Minitab. Consequently there was no much difference between values of parameters from both developed visualising applications. Hence both could accurately visualise the geometry of spine and calculate parameters defining curvature of the spine. In conclusion the developed applications were easy to use and besides they could accurately measure the parameters and visualise innovatively the shape of the spine as required.

Chapter 1: INTRODUCTION AND PROJECT'S AIMS AND OBJECTIVES

1.1 INTRODUCTION

There are many patients with spinal column deformities such as scoliosis or injuries. Statistically about 2-3% of entire population is affected by scoliosis disease[1]. Thus this paper explores the development and testing of user friendly software applications that can facilitate measurements of both three dimensional and two dimensional parameters of geometry of spine while providing a three dimensional visual feedback as a way of facilitating diagnosis and treatment of patients with spinal column deformities such as scoliosis.

Scoliosis or injuries distorts the geometry of the spine and causes interference of vertebrae column and internal organs. Spine geometry or curvature is a quite useful indication to determine severity of spinal deformity-related diseases. The geometry of the spine is usually assessed by clinicians using common imaging technologies such as fluoroscopy (X-ray). The major limitations of fluoroscopy include: inability to produce three dimensional parameters defining geometry of the spine and also there is a risk of cancer in subjects due to its ionising radiations. There are other alternatives such as use of Magnetic resonance imaging and computed tomography however the pieces of equipment involved in such imaging modalities are so expensive. Besides the space where the patient is paced is small such that dynamic measurements are not usually carried out with such technologies. Recently prices of motion capture system has been reduced due to competition and other production factors. Hence motion capture system can be the better alternative of obtaining 3-D parameters and visualisations since it can be economical. Unfortunately it is not easy to label markers for customised models and also it requires some programing skills in order to analyse the data correctly.

Thus the main purpose of the project is to develop and test D-Flow based software applications that integrates motion capture system in order to measure three dimensional parameters that defines geometry or curvature of the spine of a subject. The applications are also supposed to show visualisation on how curved the spine is so that the user of the software get a vivid visual feedback.

Secondly, the key issues that were considered included improving labelling of markers, and providing the required visual feedback for both static and dynamic application when measuring the geometry or when trying to correct the geometry of the patient's spine. Another criterion that was considered include developing user friendly applications that can be used without need of programming language skills. Moreover another issue is to measure the parameters defining the curvature more accurately.

Furthermore the other research aims that were integrated in the project included trying to visualise the spine geometry without attaching markers on the body of the subject while still using motion capture system to capture the 3D shape of the spine. Besides, the project explored the similarity of the parameters obtained from manual measurements.

1.2 AIMS AND OBJECTIVES

There are many aims and objectives that were considered in this project. Some of the main aims include: developing an innovative applications that can measure 3D parameters and visualise geometry of spine using motion capture system; testing the developed applications and validating the results obtained from the applications. On the hand the main objectives include: developing application that depend on use marker attachement on the body of the subject; developing application that does not require marker attachement on the body of the subject and making sure that the developed applications are accurate, user friendly and stable.

Firstly the aim of developing applications that can measure 3D parameters and visualise geometry of spine using motion capture system tries to eliminate the problem of lack of virtual reality based visual feedback and 3D parameters to the user in many methods of diagnosing spine related diesese such as scoliosis. For instance radiography is usually used which is limited only to two dimensional images that are non interactive. Interactive visualisation based on virtual reality can assist the user to correct the curvature of the spine more easily and accurately when trying to bend the spine of the subject or when trying to find points to apply corrective forces on the body of the patient with scoliosis. In addition the visual feedback is more useful than just data set of numerical parameters.

Besides, the milestones that were set to be achieved in this project included: developing user friendly and accurate applications that can depend or not depend on attaching markers on the body of the subject when measuring and visualising the geometry of the spine, testing the applications and comparing the similarity of the numerical parameters to those parameters measured manually. The consideration of trying to develop an application that does not depend on attaching markers on subject was realised because some patients may not have other problems with marker attachment on their bodies. Also the other milestone was to try labelling of markers or points on the subject using pointer and touch sensor. Thus the user can only interact much with the pointer and touch sensor when using the application without need to try connecting the markers in the application software.

In conclusion the project's aim are so ambitious in trying to develop new application of measuring and visualising the 3D geometry of the spine of the subject while eliminating marker or point labelling issues of motion capture intergrated application softwares.

Chapter 2: LITERATURE REVIEW

2.1 ANATOMY OF SPINE

The vertebral column consists of 26 bones (24 vertebrae, sacrum and coccyx)[2] joined together by ligaments. The sacrum consists of 5 fused bones and the coccyx has 4 fused bones hence sometimes the spinal column may be described as a structure with 33 bones. The vertebral column has cylindrical hole in which there is the spinal cord running through it. The spinal column provides support and bears weight of the head, neck and trunk. Geometrically the spinal column is a curved structure. The main curves of the spine are cervical, thoracic, lumbar and sacral curves. Figure 2.1 below shows the structure of spinal column, the spine.

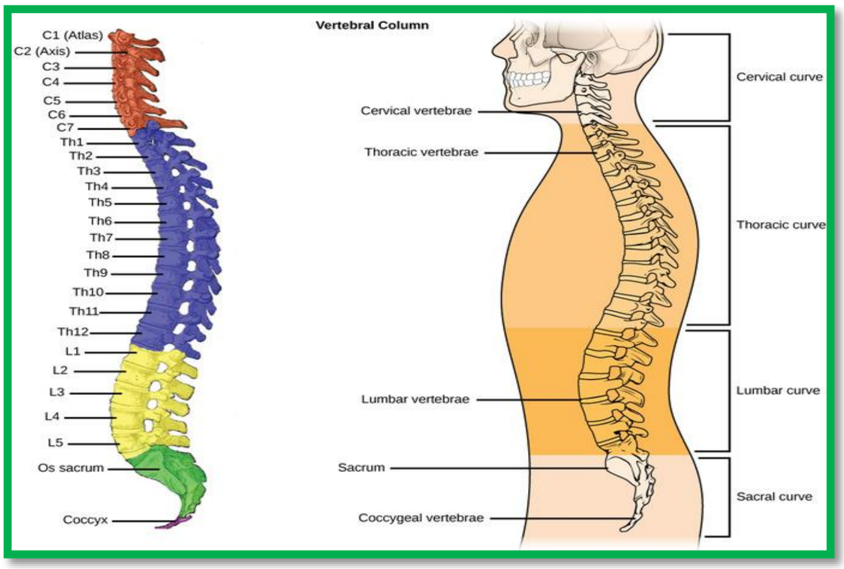


FIGURE 2. 1: Structure of human Vertebral column [3]

The spine curves can sometimes be abnormal either due to diseases, pregnancy, injury and other unknown causes. Some of the abnormal curvatures of the spine include kyphosis, lordosis and scoliosis. Kyphosis refers to humpback or abnormal posterior curvature of the spine while lordosis is the abnormal anterior curvature of lumbar spine causing buttocks and abdomen to protrude abnormally. Besides scoliosis is the abnormal lateral deformity of the spine.

2.2 SCOLIOSIS AND OTHER DISEASES OF THE SPINE

Although scoliosis is usually defined as abnormal lateral curvature of spine, it is a three dimensional deformity of the spine where there is a coexistence of lateral, anterior and posterior curvatures of the spine. The level or severity of scoliosis is defined by some parameters such as spinal curvature[4] and angle of trunk rotation (ATR). Figure 2.2 shows a straight and curved spines. Some of the causes of scoliosis include congenital, neuromuscular and osteopathic causes but mostly the causes in many patients are unknown. The type of scoliosis with unknown causes is referred to as idiopathic scoliosis. Patients with scoliosis are treated through one of the following treatments: surgery, wearing orthoses or braces in order to collect their condition and physiotherapy. When casting braces for the patients clinicians mostly use two dimensional X-ray images to determine the point where to apply 3 or 4 point forces in order to correct the curvature of the patients' spine. This is challenging since the curve is 3D in nature. Different scientists have tried to visualise the curve using more advanced imaging modalities such as magnetic resonance (MR), Computed Tomography and Ultrasound. Most of these advanced imaging modalities are expensive and sometimes not easy to apply with much flexibility in practice. There are also concerns on side effects of too much radiation exposure during imaging when using x-ray and computed tomography imaging modalities.

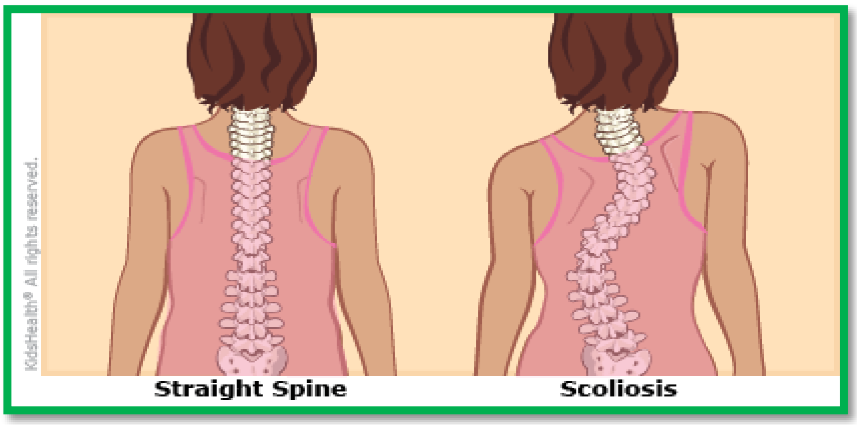


FIGURE 2. 2: Straight spine and spine with scoliosis [5]

2.3 IMAGING MODALITIES AVALABLE FOR SCOLIOSIS PATIENTS

There are different imaging modalities that are used to visualise shape of the spine of scoliosis patients. Some of them include radiography, magnetic resonance imaging (MRI) and computed tomography (CT) scanning. All the imaging modalities have pros and cons however the cheaper and most widely used among all is radiography or fluoroscopy.

2.3.1 Radiography of spine

This involves using radiation to show or visualise internal tissues, in this case the spine. Figure 2.3 below shows a typical radiograph.

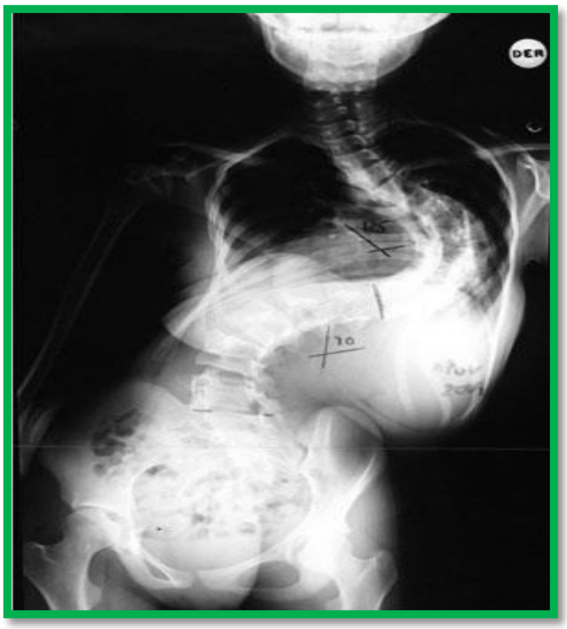


FIGURE 2. 3.1: radiograph of spine of a patient with scoliosis[6]

Advantages of Radiography

- It is cheap or easily affordable, non-invasive and quick to use
- Enable real internal examinations of the real bones and structures

Limitations

- It is two dimensional
- It can cause cancer

2.3.2 MRI scanning

Magnetic Resonance Imaging involves the use of magnetic fields and radio waves in order to create images of internal structures such as bones. This imaging technique can be used to evaluate the curvature of the spine and also reconstruct the 3D deformity of the spine. Figure 2.3.2 below illustrates an example of images obtained from MRI scanning.

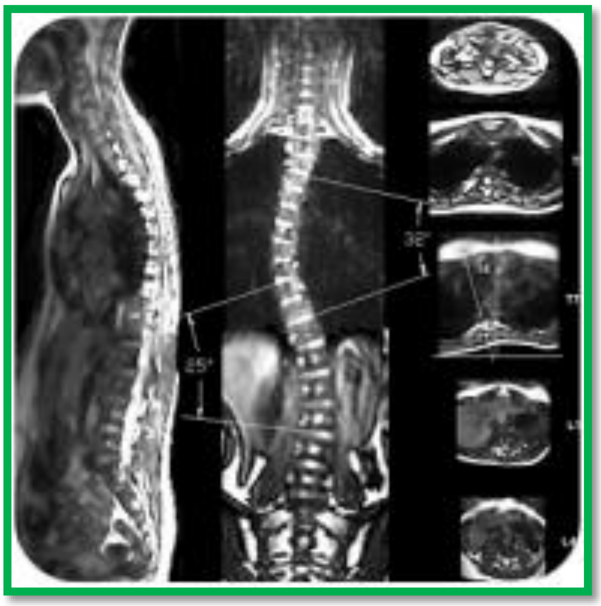


FIGURE 2.3.2 : a typical MRI image of spine with abnormal curvature due to scoliosis[7]

Advantages of MRI

- It is non-invasive and painless
- It does not have ionising radiation as in radiography, CT scanning and nuclear imaging
- Enable real internal examinations of the real bones and structures
- It can create 3D images

Limitations

- It is noisy
- It expensive
- Can cause patients to experience some claustrophobic effects
- Cannot be used in some conditions where the patient has some implants such as pace makers.

2.3.3 Computed Tomography

This is an imaging technique that uses X-Ray just like radiography but together with a computer to reconstruct a more detailed 3D image of the internal features of body such as bones. It is a specialised x-ray based imaging technique. Figure 2.3.3 below shows an example of Computed tomography (CT) scan. CT scans can be used to determine parameters of the spine of a patient with scoliosis. Some of the parameters that are determined from CT scans include: deformity and curvature of vertebrae and spine respectively.

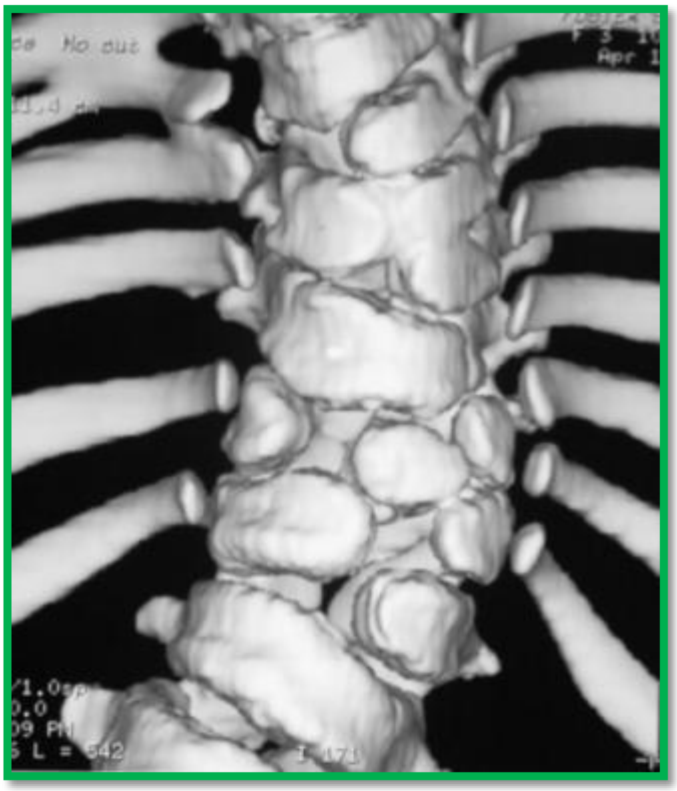


FIGURE 2.3.3: CT scan of the spine[8]

Advantages

- More detailed information: 3D information
- Non-invasive and pain-less.

Disadvantages

- X-ray radiation used in this imaging technique can cause cancer.

2.3.4 Ultrasonography

This is a diagnostic imaging technique of internal features of human body using a machine that uses principle of sound waves with high frequency above audible limit for humans. The technique though limited can be used to visualise some deformity of the spine and surrounding tissues. Figure 2.3.4 below, shows an image of spine obtained through 3D ultrasound imaging technique.

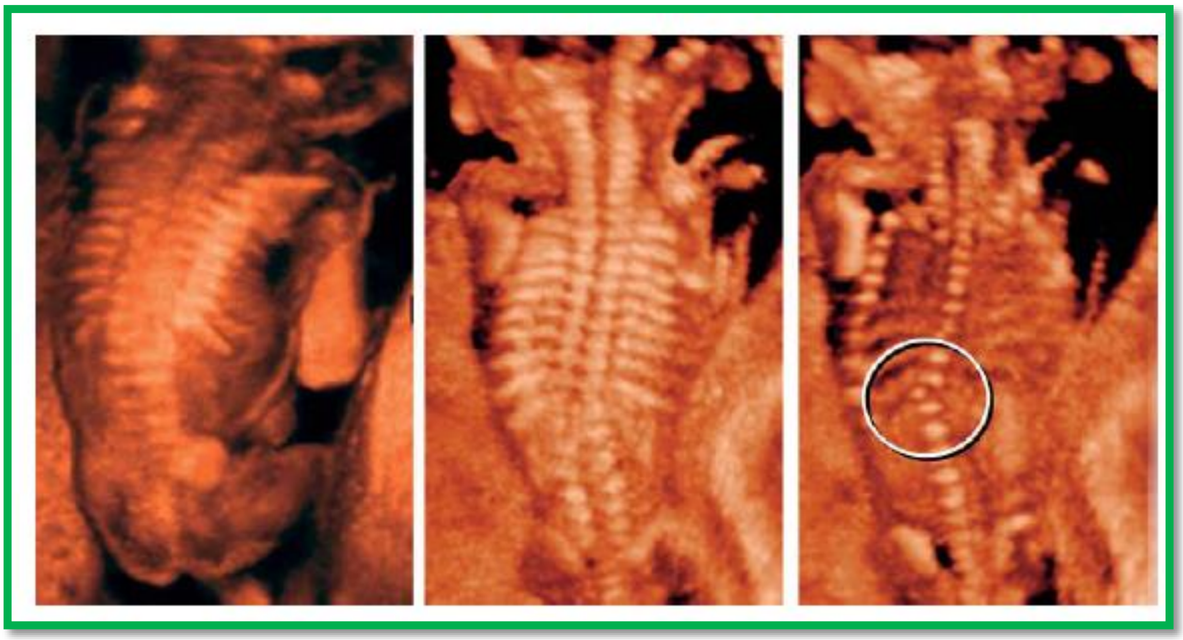


Figure 2.3.4: scoliosis of 20 week old foetus detected by 3D ultrasound[9]

Advantages

- Painless and non-invasive.
- Has no ionising radiation
- It can be used to obtain 3D reconstruction of part of the spine

Disadvantages

- It is not easy to use it.
- It is very limited and quality of images is not very good.

2.3.5 Nuclear imaging

It is an imaging technique that uses small doses of radiative substances in order to determine or diagnose severity or level of disease or deformity even in early stages of disease or deformity development [10]. This imaging technique may involve being injected, swallowing or inhaling a radioactive material that can traced and there after reconstruct the image of the part where the material is present. This imaging technique is not routinely used to diagnose scoliosis however it can be used to evaluate painful scoliosis[8]. Figure 2.3.5 shows the image obtained from nuclear imaging.

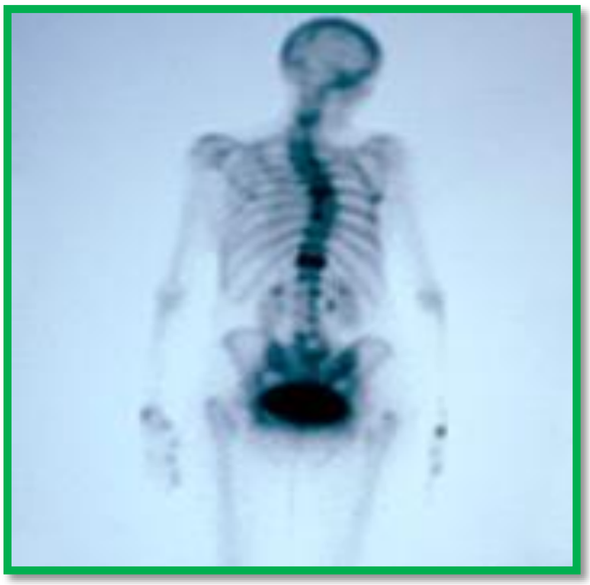


Figure 2.3.5: Scoliosis visualised by nuclear imaging[11]

Advantages

- It is non-invasive and painless
- Shows performance and more details of internal features
- Can diagnose wide range of disorders and diseases

Limitations

- Ionising radiation used as tracing material can cause cancer
- The scanner causes claustrophobic effects in some patients

2.3.6 Motion Capture system

The motion capture system is a system used to capture trajectory or coordinates of a moving body or body segment. Motion capture systems are commonly used in gait and body segment analysis. Some of motion capture systems uses infrared cameras, markers and computer with special software while others uses inertial sensors. One of the popular motion capture system is Vicon motion capture system which uses Vicon Nexus software to analyse coordinates of markers. The latest version of this software Nexus 2.1.1 has some built in visual effects that enables the user to see the movement of body segments. However the visualisation indicate the trunk as one rigid segment. It has also modelling features however they require skills to be used appropriately. In other words it is easier for an engineer to use them than a clinician. Besides the Vicon motion capture software can be used in conjunction with Motek D-Flow software which is a quite useful modelling software. Figure 2.3.6 shows a visualisation model of human body using Vicon Nexus software and motion capture system.

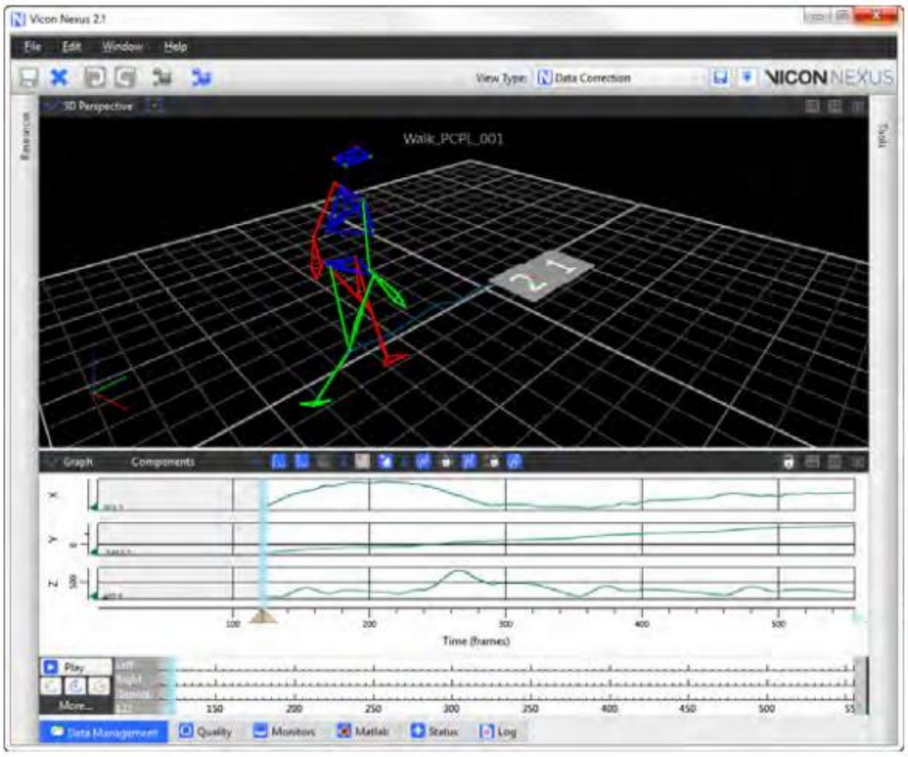


Figure 2.3.6: visualisation of human body using Vicon Software[12]

Advantages

- No radiation
- 3D information of a point or segment of the body

Limitations

- The software still is not yet that easy to use by many
- Cannot visualise internal features

2.4 TREATMENTS OF SCOLIOSIS

There are different treatment procedures for scoliosis patients. Some of them include bracing, surgery, physiotherapy, and hybrid procedures.

2.4.1 Bracing

Bracing is a primary method used to treat scoliosis, relieve pain and support the part with serious injury. There is evidence that bracing correct moderate idiopathic scoliosis[13]. Braces are mostly made by casting however there are also off the shelf braces which are made by special manufacturing companies. Casting of braces for patients with Scoliosis in hospitals is carried out following different recommended procedures and also by following The International Scientific Society on Scoliosis Orthopaedic and Rehabilitation Treatment (SOSORT) guidelines. According to SOSORT guidelines it is pointed out that “bracing is recommended to treat patients with curves above 20 ± 5° Cobb, still growing, and demonstrated progression of deformity or elevated risk of worsening, unless otherwise justified in the opinion of a clinician specialized in conservative treatment of spinal deformities" [14].

There are different types of braces such as Thoraco-Lumbo-Sacral-Orthosis (TLSO), Cervico-Thoraco-Lumbo-Sacral-Orthosis (known as a Milwaukee brace), and Charleston Bending Brace. In constructing braces the orthostist uses images from radiography, CT scanning or any other imaging modality available in order to determine three or four points where to apply correcting forces. Figure 2.9 below shows different types of braces.

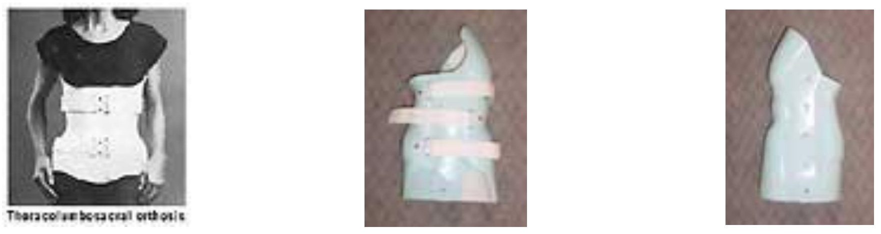

Thoracolumbosacral orthosis
Thoraco-Lumbo-Sacral-Orthosis
Cervico-Thoraco-Lumbo-Sacral-Orthosis
Charleston Bending Brace

Figure 2. 1: Different types of braces[15]

2.4.2 Surgery

This involves carrying out invasive surgical procedure to implant supporting biocompatible materials such as synthetic material or their own bones from other places in order to correct the curve and stabilize the shape of the spine of the patients. It is quick to achieve correction of the curve using surgical procedure however it still remain dangerous and can result to many problems such as nerve damage, stroke or even death of the patient[16].

2.4.3 Physiotherapy

This involve carrying out physical exercises in order to correct the curve of the spine. It may be used in conjunction with bracing treatment or after the patient has recovered from surgical treatment procedure.

2.5 STUDIES RELATED TO DERTEMINING GEOMETRY OF SPINE BASED ON MOTION CAPTURE SYSTEM

There are many studies in literature that have either developed models of the spine or applications facilitating determining numerical data describing geometry of the spine. Some of the studies include: “Realistic model of spine geometry in human skeleton in Vicon system” by Dlugosz et .al [17], “Non-invasive approach towards the in vivo estimation of 3D inter-vertebral movements: methods and preliminary results” by Cerveri et. al [18] and “the use of motion analysis technology as an alternative means of assessing spinal deformity in patients with adolescent idiopathic scoliosis” by Matthew J. Solomito[19]. This section briefly explores these studies and their limitations.

Firstly the study titled Realistic model of spine geometry in human skeleton in Vicon system by Dlugosz et .al focussed on developing a model in Vicon Python softoware. The model can then be inserted into already existing models in Vicon Software.The 3D visualisations in this paper were created by 3D scaning and use of 3D graphics software. The model was then exported from Blender software[17]. The advantages of this Realistic model of spine geometry is that it gives a better visualisation of spine segment. This study is limited such that it does not necessarily provide numerical calculations of parameter that can assist in defining the geometry of the spine numerically. Beside the model is embedded in already existing models which of course when used have labelling and complexity issues.

Secondly, the study titled “Non-invasive approach towards the in vivo estimation of 3D inter-vertebral movements: methods and preliminary results” by Cerveri et. al was about designing and using a model to obtain robust estimations of the vertebral rotations for the duration of torso movements from skin-surface markers[18]. The major limitations of this study is that the focuss was on lower spine region and intervertebral rotations. However it gives an insight of non invasive approach when studying the spine and also how to attach markers on the spine. Figure 2.5.1 below shows how markers were attached on the subject.

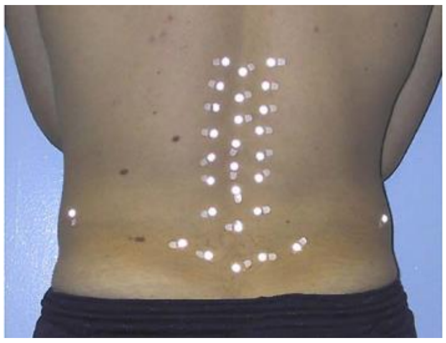


Figure 2.5.1: marker attachement on lower spine[18]

Furthermore another study on spine was by Matthew J. Solomito. This study was titled “the use of motion analysis technology as an alternative means of assessing spinal deformity in patients with adolescent idiopathic scoliosis” and focussed on developing a Matlab based script to facilitate calculation of different parameters of the spine. The major limitation of this study is that the developed application did not use virtual reality 3D models of the spine and also the results obtained are numerical only.

In conclusion there are many studies that have been carried out in order to try to use motion capture systems when obtaining parameters of geometry of human spine. The major limitations of many studies include: use of application that solely depend on attaching markers on subject; use of application that only produce numerical data sets or models and unable to eliminate labelling issues associated with application softwares of motion capture systems.

Chapter 3: CONCEPTS

3.1 FIRST CONCEPT

The first concept was to develop an application that can visualise the spine of a subject without attaching markers on body of subject and just use a pointer and touch sensor to locate different points of the spine.

In this concept the user is supposed to point a landmark where there is a vertebra and then press a button of a touch sensor in order to capture the coordinates of that point. The coordinates will then be used for calculations and visualisation of different vertebrae.

Thus the user will only use the pointer by moving it along the spine while pressing button of touch sensor in order to visualize the spine.

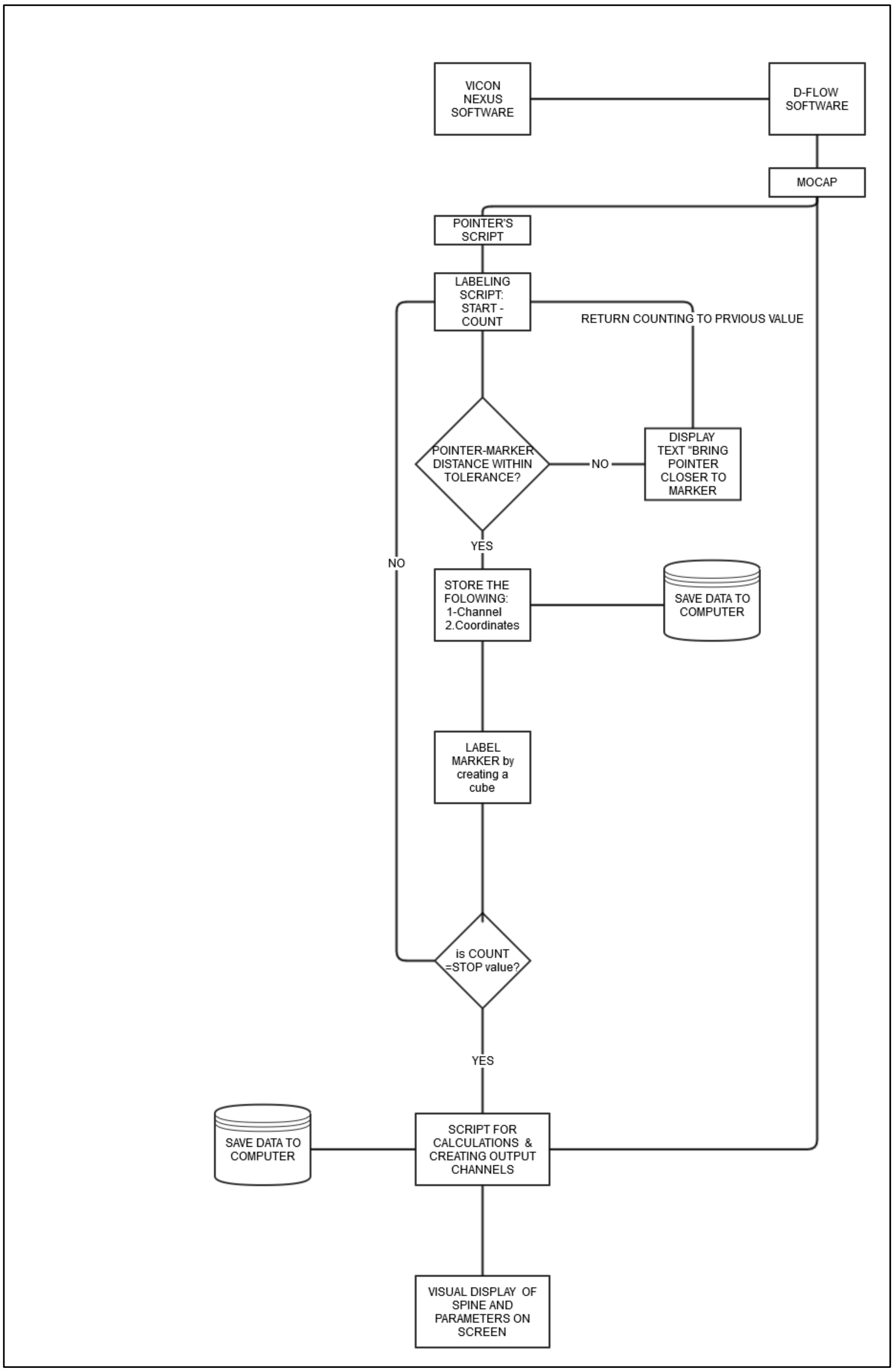
3.2 SECOND CONCEPT

The second concept was to develop an application that depends on attaching markers on the body of the subject and also using pointer and touch sensor to label those markers.

As shown in flowchart figure 3.2.1 on the next page, the application is supposed to operate in D-flow software and linked to Vicon Nexus. Once motion capture system is calibrated and cameras masked using Vicon Nexus software, the application can be opened using D-flow software.

A phiget and pointer (wand) are supposed to be used in order to label markers or capture marker's position. The captured marker then changes colour from any colour to blue and shape from sphere to cube. If the position of the marker is not captured, conditions written in code of the software are supposed to prevent capturing nil values. Nil values causes clash of the program during calculations. If the marker's channels defining position are not captured the software is supposed to warn the user to repeat marker labelling, by displaying the message "bring pointer closer to marker." If the marker is labelled, the shape of the marker will change from a sphere to a cube and the labelling process will continue until all markers are labelled. This visualization of changing markers shape is done by creating a cube object on the position where there is the labelled marker. The labelling process will continue up to the point where all 40 markers have been labelled.

The labelling process is carried out by a script that identifies and stores the marker’s channel index numbers. The channels numbers are used in another script that streams coordinates and carry out visualization of the spine and calculations of different parameters.



VICON NEXUS SOFTWARE
D-FLOW SOFTWARE
MOCAP
POINTER'S SCRIPT
LABELING SCRIPT: START - COUNT
RETURN COUNTING TO PRVIOUS VALUE
POINTER-MARKER DISTANCE WITHIN TOLERANCE?
NO
DISPLAY TEXT "BRING POINTER CLOSER TO MARKER
YES
NO
STORE THE FOLOWING: 1-Channel 2.Coordinates
SAVE DATA TO COMPUTER
LABEL MARKER by creating a cube
is COUNT =STOP value?
YES
SAVE DATA TO COMPUTER
SCRIPT FOR CALCULATIONS & CREATING OUTPUT CHANNELS
VISUAL DISPLAY OF SPINE AND PARAMETERS ON SCREEN
13

Figure 3.2. 1: Flow diagram of the Application based on concept 2

Chapter 4: METHODOLOGY

4.1 APPARATUS AND MATERIALS

The apparatuses that were used during the development of the application includes: Vicon's motion capture system using Bonita cameras, computer with D-flow and Vicon Nexus software. Besides a touch sensor and touch sensor interface kit were used as a Phiget. A phiget is an electronic system controlled by computer[20] and that can be connected to a graphical user interface to control performance of different script statements. A pointing wand consisting of three attached permanent markers and one temporary marker just used for calibration of the end tip of the pointing wand. More over a skeleton and cable were used as subjects in order to mimic different human vertebrae column's geometries. The following figure shows some of the apparatus used during development of the applications for visualising the geometry of the spine.

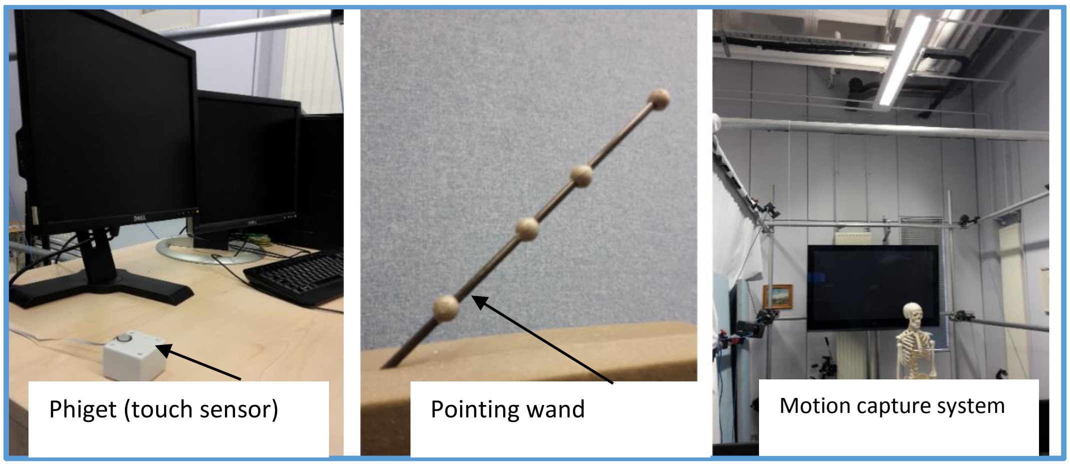


Figure 4.1: apparatus or devices, subject and systems that were used

4.2 SOFTWARE THAT WAS USED FOR DEVELOPMENT OF THE APPLICATIONS

The applications were developed in D-flow software since it's easy to work in D-flow. There were three applications that were developed namely wand calibration application, marker dependent application and marker independent application.

The pointing wand calibration is used to calibrate the pointing wand coordinates if the wand is being used for the first time. The application stores details or major parameters describing geometry of the wand and store the parameters in a special file that is linked to both marker dependent and marker independent applications.

The marker dependent application was developed such that it can be used to labelling forty markers attached on the subject's body and determine position of the spine. Once the markers are labelled properly the application could be able to track the geometry and movement of the spine of the subject. Details on how the applications were developed are outlined in concept development section of this report.

Furthermore the marker independent application was developed for visualising static geometry of the subject.

More details on how the application were developed using D-Flow software are outlined in the concept development section.

4.3 MARKERS ATTACHMENT ONTO SUBJECT

The markers during testing of application were attached on a lab skeleton as shown in figures 4.3.1 to 4.3.3 using double sided tape. The reason behind attaching markers on the skeleton is that the application will be tested easily at each level of development. The application was not be tested on patients.

The number of markers that was used during application development is 40. Some markers are just to show position of some parts of the body while some are more meaningful and useful for calculation and visualisation of the parameters required from scoliosis patients. As shown in figures 4.3.1 to 4.3.3, the markers were named after position where they were attached, for instance marker at C4C5 vertebrae was named as C4C5 marker. The protocol of Based on this protocol implemented in the coding of the application, the markers are always supposed to be attached as follows shown. The chosen markers are so critical in calculation of required parameters from the patients with scoliosis.

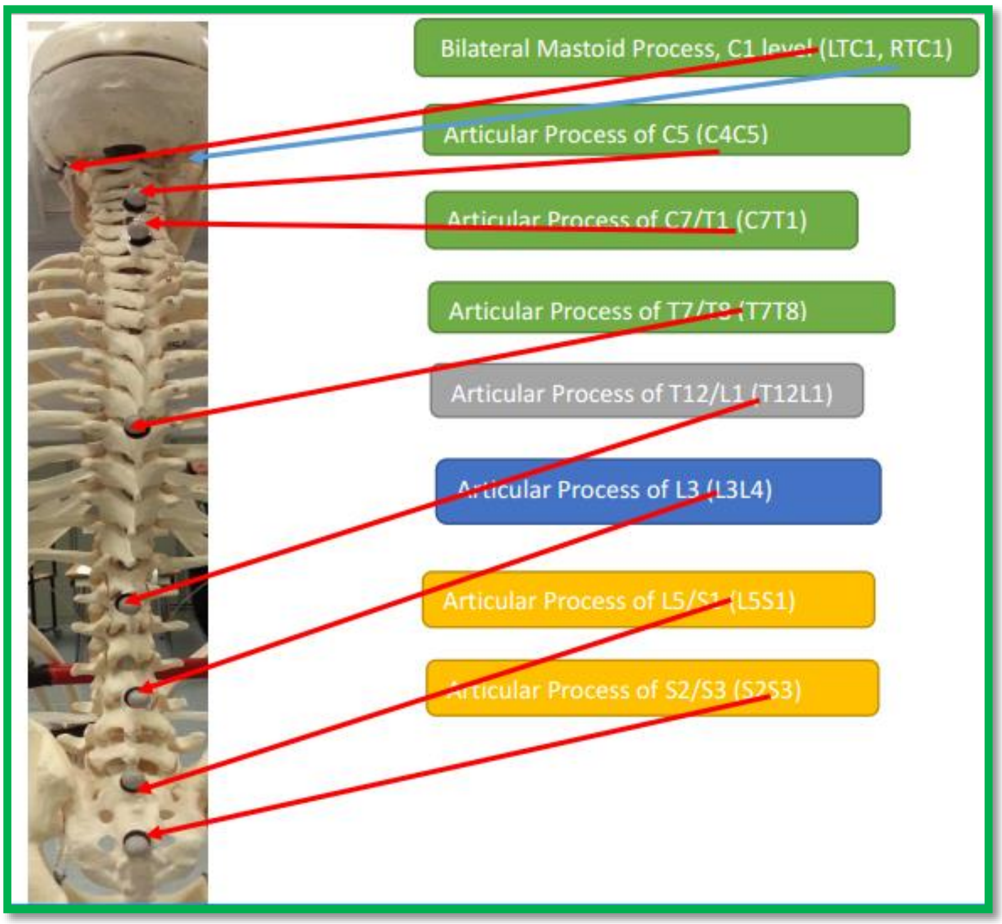

Bilateral Mastoid Process, C1 level (LTC1, RTC1)
Articular Process of C5 (C4C5)
Articular Process of C7/T1 (C7T1)
Articular Process of T7/T8 (T7T8)
Articular Process of T12/L1 (T12L1)
Articular Process of L3 (L3L4)
Articular Process of L5/S1 (L5S1)
Articular Process of S2/S3 (S2S3)

Figure 4.3.1: Names of Markers along spine

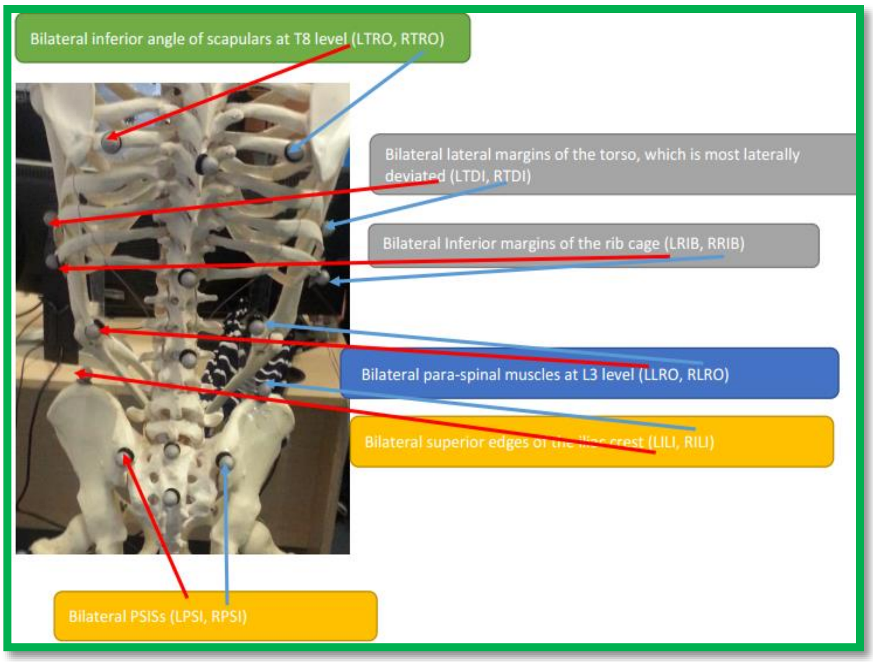

Bilateral inferior angle of scapulars at T8 level (LTRO, RTRO)
Bilateral lateral margins of the torso, which is most laterally deviated (LTDI, RTDI)
Bilateral Inferior margins of the rib cage (LRIB, RRIB)
Bilateral para-spinal muscles at L3 level (LLRO, RLRO)
Bilateral superior edges of the iliac crest (LILI, RILI)
Bilateral PSISs (LPSI, RPSI)

Figure 4.3.2: : Name of markers close to those markers on the spine

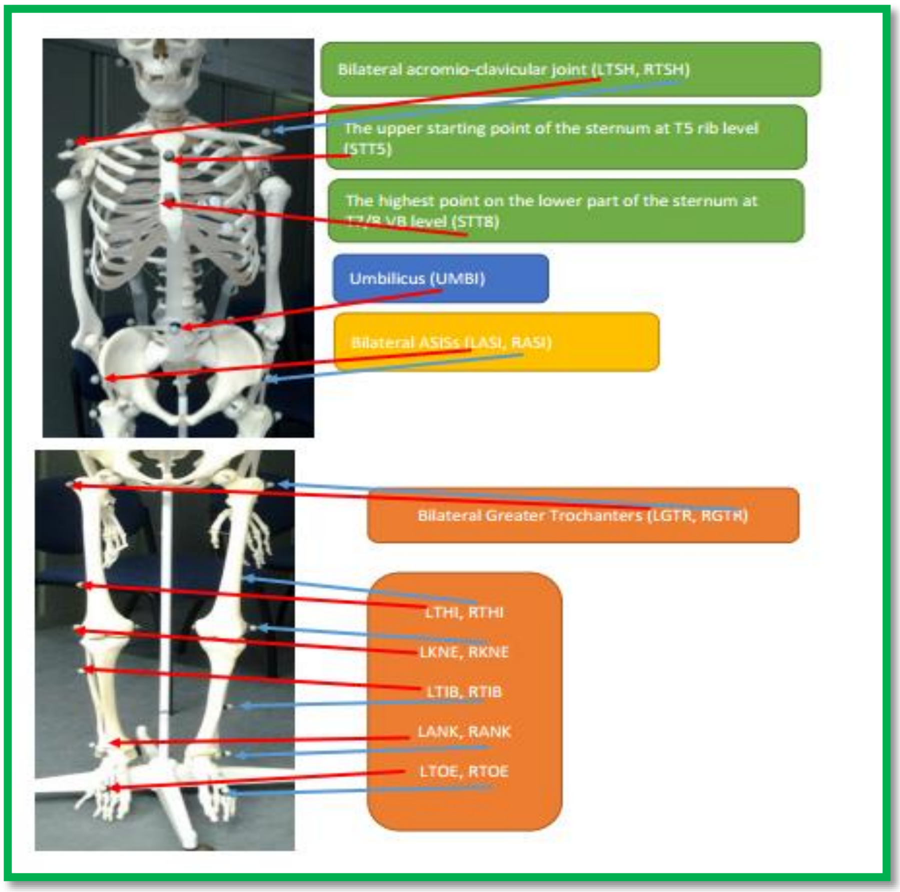

Bilateral acromio-clavicular joint (LTSH, RTSH)
The upper starting point of the sternum at T5 rib level (STT5)
The highest point on the lower part of the sternum at T7/8 VB level (STT8)
Umbilicus (UMBI)
Bilateral ASISs (LASI, RASI)
Bilateral Greater Trochanters (LGTR, RGTR)
LTHI, RTHI
LKNE, RKNE
LTIB, RTIB
LANK, RANK
LTOE, RTOE

Figure 4.3.3: Markers on the outermost peripheral of the body, front and feet

4.4 LABELLING OF THE MARKERS OR POINTS

4.4.1 Labelling in marker dependent application

The process of labelling was carried out using pointing wand and touch sensor. The labelling script was also developed such that the labelling process should follow a particular pattern as shown in figure 4.4.1 below. The labelling was developed in order to facilitate memory of how to label markers without remembering index numbers of markers as shown in table 4.4.1.

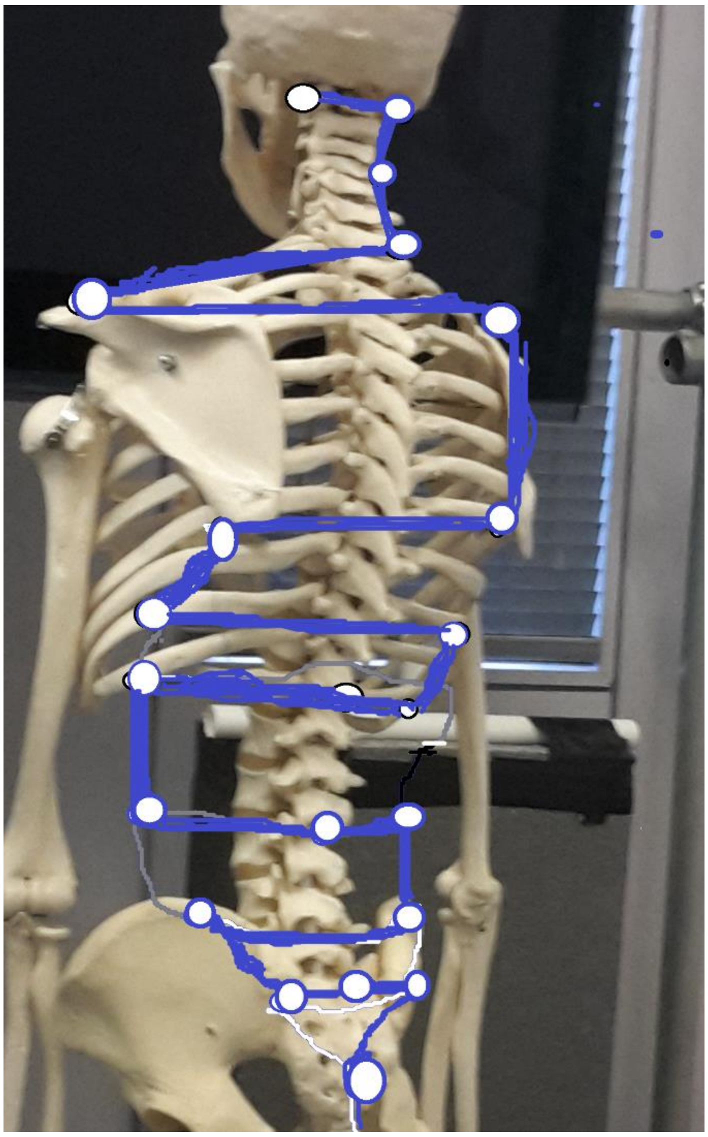


Figure 4.4.1: :Pattern of labelling to facilitate easier and quicker labelling

Thus if the application is used, the first marker to be labelled should always be left marker at C1 vertebra (LTC1). The table below shows index numbers of the markers that were used in this protocol that uses forty markers.

Table 4.4.1: index numbers of the markers representing order of the labelling process

Index number	**Marker name**
1	LTC1
2	RTC1
3	C4C5
4	C7T1
5	LTSH
6	RTSH
7	RTRO
8	T7T8
9	LTRO
10	LTDI
11	RTDI
12	RRIB
13	T12L1
14	LRIB
15	LLRO
16	L3L4
17	RLRO
18	RILI
19	LILI
20	LPSI
21	L5S1
22	RPSI
23	S2S3
24	LASI
25	RASI
26	RGTR
27	LGTR
28	LTHI
29	RTHI
30	RKNE
31	LKNE
32	LTIB
33	RTIB
34	RANK
35	LANK
36	LTOE
37	RTOE
38	STT5
39	STT8
40	UMBI

4.4.2 Labelling in marker independent application

The labelling in the marker independent application was carried out using point wand and touch sensor (phiget). The labelling script was developed such that the labelling process should be carried out from C1 to S2. The table below summarises the labelling points and their names.

Table 4.4.2: order of labelling points corresponding to vertebrae, for the marker independent application

Points or index number	Point Name
1	C1
2	C2
3	C3
4	C4
5	C5
6	C6
7	C7
8	T1
9	T2
10	T3
11	T4
12	T5
13	T6
14	T7
15	T8
16	T9
17	T10
18	T11
19	T12
20	L1
21	L2
22	L3
23	L4
24	L5
25	S1
26	S2

4.5 CALCULATION OF PARAMETERS

The calculation of parameters are based on markers' coordinates. The way motion capture system shows the coordinates is that the first channel is x coordinate, followed by channel with y coordinate and then z-coordinate channel. Thus the coordinates of x are identified by formula $x = 3(i - 1) + 1$ while y coordinate formula is $y = 3i(i - 1) + 2$ and that of z-coordinate is $z = 3i(i - 1) + 3$ where "i" is the number representing counting of channels and "i" starts from 1 to total number of channels. The coordinates can then be used to calculate distances and angles using Pythagoras theorem and trigonometric functions respectively.

The parameters that were calculated and their guiding equations are shown in Calculation Section of this document.

4.6 VISUALISATION OF PARAMETERS

The spine column was visualised used objects with shape similar to that of different vertebrae. The objects were drawn using Google Sketchup software and were exported to format (.scene) which is compatible with D-flow software. In order to export the three dimension objects from Google Sketchup a software called Orge converter was installed which then creates a command “Export to ogre” on the interface of Google Sketchup software.

The drawn and exported object were then added as scene objects on the interface of D-Flow software where their position channels were linked to inputs channels representing vertebrae from the calculation and visualisation script module.

There were 26 objects that were used to represent each vertebra of the spine that is from C1to Coccyx.

4.7 ANALYSIS OF DATA: VALIDATION, TESTING APPLICATIONS AND COMPARISON OF PARAMETERS

The applications were tested by visualising a spine column of a laboratory skeleton and another s-shaped cable placed on the skeleton as shown in the results section. The values of different parameters were recorded. The performances of the software were noted based on frequency or frame rate.

Besides manual measurement were used as gold standard to compare with the values of parameters from the applications. The following manual instruments were used: a goniometer and protractor for measuring angles of segments, Measuring tape or rule for measuring linear distances or perimeters.

Using Minitab Software, the parameters obtained by the developed Applications were analysed in order to obtain descriptive statistics of the data set of parameters and then compared to those values of the parameters obtained from manual measurements. The comparisons of similarity of the two measurements enabled validation of correctness or accuracy of the recorded measurements from the developed applications.

Chapter 5: CONCEPT DEVELOPMENT

5.1 POINTING WAND CALIBRATION APPLICATION

The pointing wand calibration application which was developed in D-Flow software is used to record dimensions between four markers attached to a pointing wand. The pointing wand is pointer used to locate and capture channels of a marker attached on a subject or used just to capture coordinates of different points on a body of the subject. The pointing wand calibration application stores average dimensions between the markers and stores them in a text (.txt) document. The code of this application is shown in appendix 1. Figure 5.1 (a), (b) and (c) below shows the flow diagram of this application, module arrangement in D-flow and the pointing wand respectively.

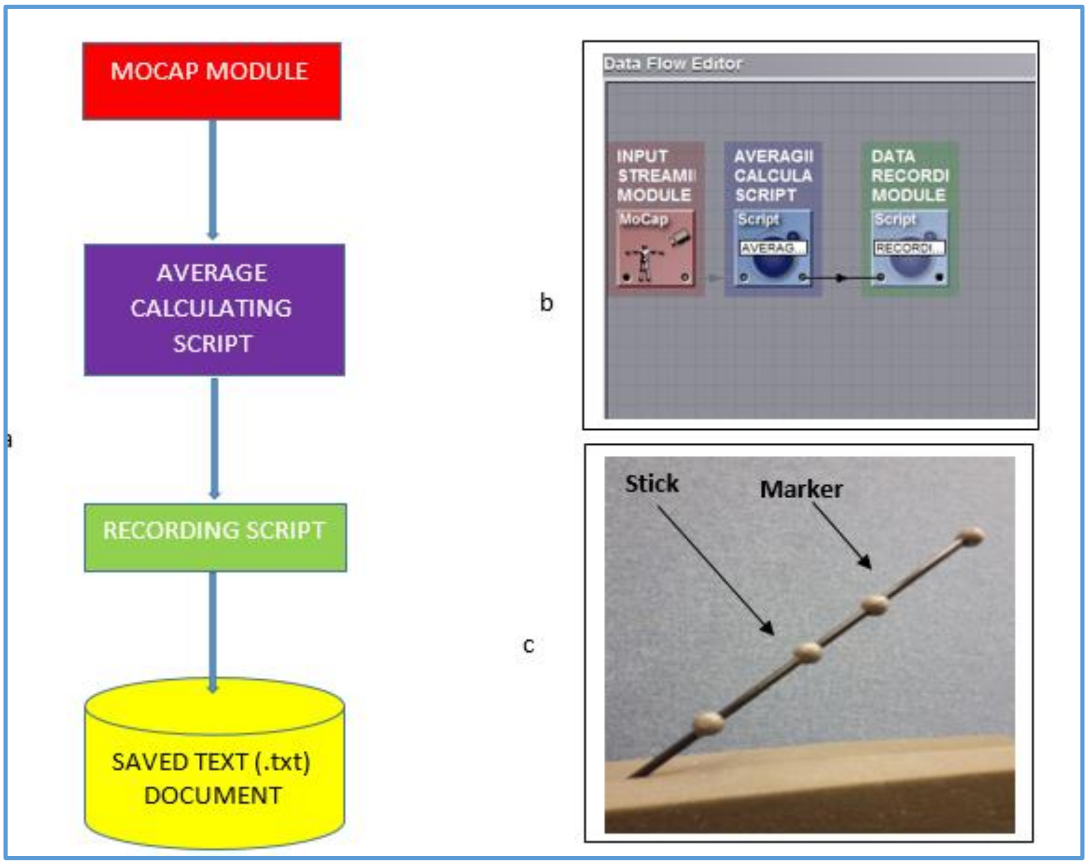


Figure 5. 1: (a) the flow diagram of pointing wand calibration application, module arrangement in D-flow and the pointing wand (b) module arrangement in D-Flow (c) pointer or wand

Firstly the MOCAP module is used to stream data, coordinates of four markers, from motion capture system. This module is readily available in D-flow application and cannot be edited.

Secondly the average calculating script is used to calculate average coordinates of the four markers attached on the pointing stick or wand in "n" numbers of frames recorded.

The average-calculating script was written in a SCRIPT module available in D-Flow application.

The average calculating script is then linked to recording script. The recording script was written such that the average coordinates of markers are recorded and saved in computer. The path of the directory where the text file containing average data is stored is specified in this script.

Finally in order to calibrate the pointing wand coordinates in text file, the wand is supposed to be placed in the capture volume and then clicking play button of the pointing wand calibration application. This was done and the pointing wand calibration application was able to store the average coordinates in the specified directory folder of the computer.

5.2 MARKER DEPENDENT APPLICATION

The main use of this application is to label markers attached on a subject, calculate required parameters and thereafter visualize the shape of the spine of the subject. The marker dependent application was developed in D-Flow as well. The maximum number of markers that must be attached on the subject's body is forty. The developed codes of this application is shown in appendix section. Figure 5.2.1 and 5.2.2 on the next pages shows the flow diagram of this application and modules arrangement in D-Flow respectively.

Firstly the inputs of this application are MOCAP module and PHIDGET module. The MOCAP module is used to stream data, coordinates of forty markers attached to subject and three markers on the pointing wand, from motion capture system while the PHIDGET module receives data from PHIDGET device and its interface. When the button of the PHIDGET is pressed the PHIDGET channel shows numeric greater than 0 and less or equal to 1 otherwise when the button of the PHIDGET is not pressed the channel of PHIDGET indicates zero value. These two input modules MOCAP and PHIDGET were connected to different scripts modules in order to label markers and their after calculate different parameters and visualize position of the spine.

Secondly the MOCAP module was connected to the pointing wand detection module. The pointing wand detection module is a script that streams data from MOCAP module and identifies markers of the on the wand and thereafter create a visual cubic marker at the tip of the pointing wand. This script also creates output of the same inputs as the inputs from MOCAP module so that when this script is connected to other modules, the streamed data from MOCAP channels should be available for those connected modules. The pointing wand detection module script reads the average coordinates stored in the pointing wand calibration text file and calculates the average length between the calibrated markers. Once the dimensions of between the markers is calculated the module then compares the average distances to distances between all markers. The script does so by calculating the distances of between adjacent markers using all coordinates from channels of MOCAP module identifies the pointing wand by comparing to best suitable combination with a tolerance of 0.002m. When the condition of best matching distance is fulfilled the module creates coordinates of a visual marker and visualizes the visual marker in DRS window using “ Object.create” and “setposition” functions.

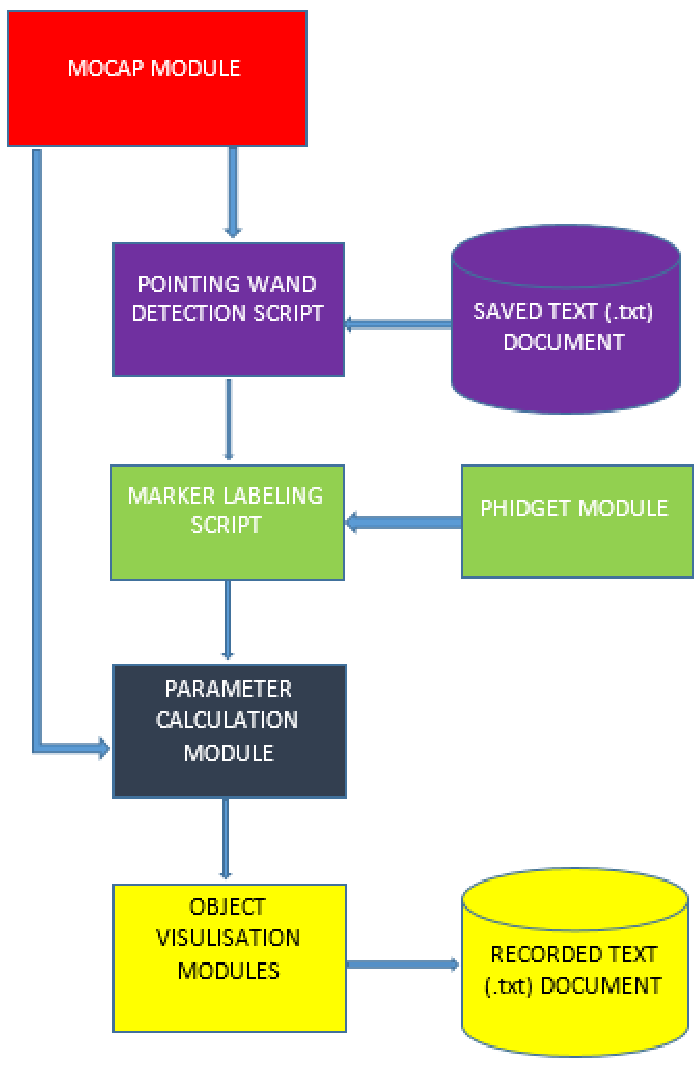

MOCAP MODULE
POINTING WAND DETECTION SCRIPT
SAVED TEXT (.txt) DOCUMENT
MARKER LABELING SCRIPT
PHIDGET MODULE
PARAMETER CALCULATION MODULE
OBJECT VISULISATION MODULES
RECORDED TEXT (.txt) DOCUMENT

Figure 5. 2.1: flow diagram of markers dependent application

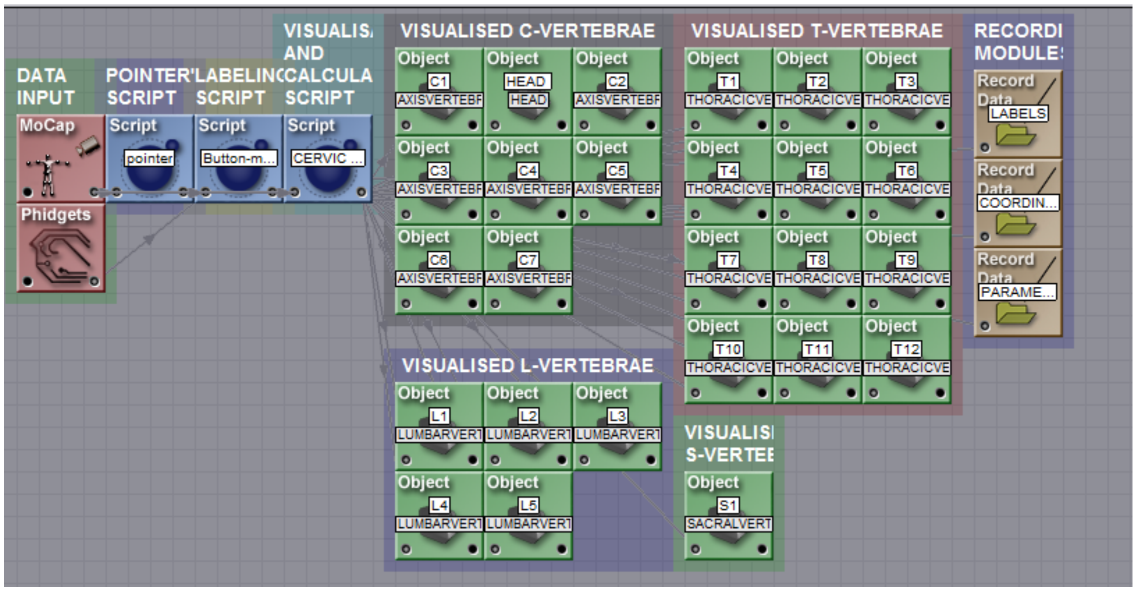


Figure 5.2.2: Module arrangement for the marker dependent application

Furthermore when the pointing wand has been detected by pointing wand detection script, the labelling script waits for PHIDGET input. The labelling script was written such that if the PHIDGET button is pressed but the wand is away from a marker the script should not capture nil value and should reset counting of iteration loop as shown in figure 5.2.3 on the next page. Besides if the pointing wand is close to the marker within a distance of $\pm 0.002m$ and the button of the PHIDGET is pressed the labelling script captures and stores index numbers of three channels representing that marker. Besides other variables were developed that can stream data from labelled channels. The streamed data is thereafter used in calculation and visualization of the spine.

Finally another module for recording the required output parameters was added in order to save the required data for referencing purposes.

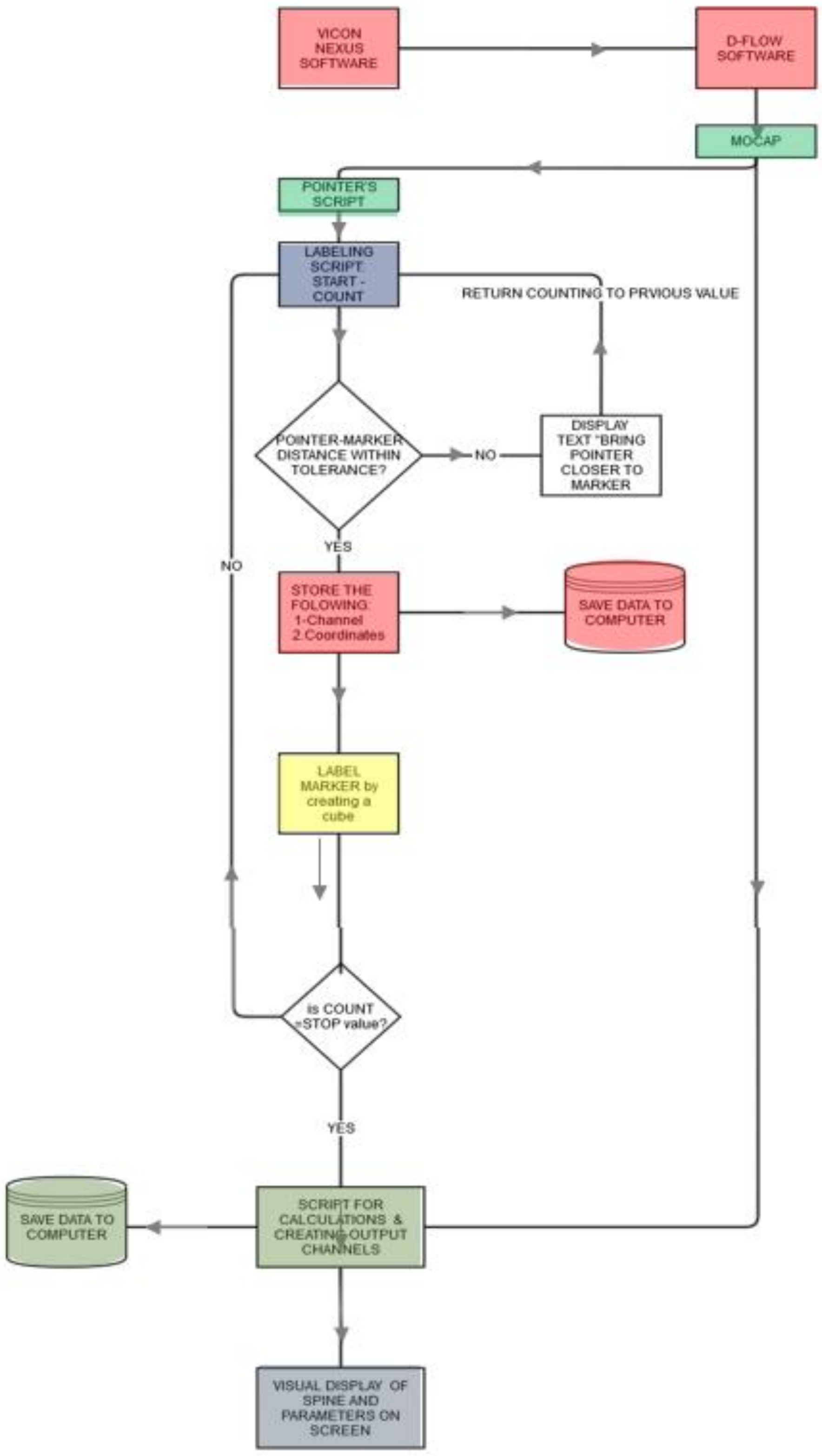

VICON NEXUS SOFTWARE
D-FLOW SOFTWARE
MOCAP
POINTER'S SCRIPT
LABELING SCRIPT: START - COUNT
RETURN COUNTING TO PRVIOUS VALUE
POINTER-MARKER DISTANCE WITHIN TOLERANCE?
NO
DISPLAY TEXT "BRING POINTER CLOSER TO MARKER
YES
NO
STORE THE FOLOWING: 1-Channel 2.Coordinates
SAVE DATA TO COMPUTER
LABEL MARKER by creating a cube
is COUNT =STOP value?
YES
SAVE DATA TO COMPUTER
SCRIPT FOR CALCULATIONS & CREATING OUTPUT CHANNELS
VISUAL DISPLAY OF SPINE AND PARAMETERS ON SCREEN

Figure 5.2.3: iteration loops for the marker dependent application

Steps of how to use the marker dependent application

1. Attach markers using protocol or marker attachment procedure explained in methodology section.
2. Open nexus software and calibrate the camera if the cameras are not calibrated.
3. Open the developed D-flow based application and make sure all markers on the subject can be seen in DRS window
4. Click play or start button on the application
5. Move wand in the capture volume and check if the visual cubic marker can be seen at the tip of the pointing wand.
6. If the green visual cubic marker can be seen, as shown in figure 5.2.4 below, you can start labelling markers starting from the LTC1 marker and following the pattern shown in labelling procedure as explained in the methodology section. Point the wand to the marker and make sure the tip of the wand is close enough such that the visual cubic marker engulfs the marker. When the cubic visual marker has covered the marker click the button on touch sensor. If the visual marker was close enough to the marker that was being labelled, the shape of the marker changes from sphere to blue cubic marker

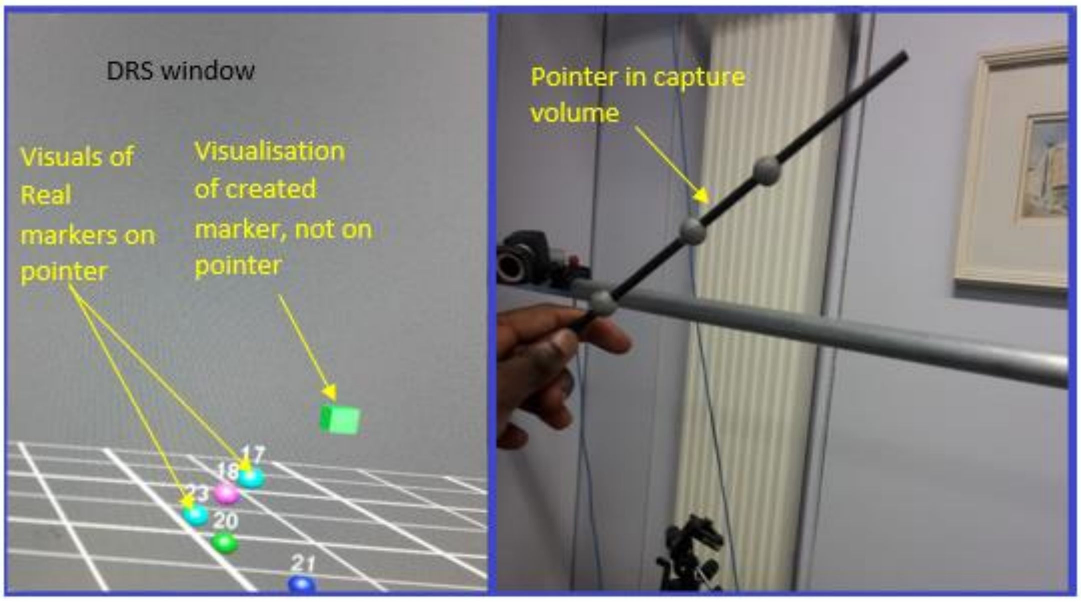

DRS window
Visuals of Real markers on pointer
Visualisation of created marker, not on pointer
17
18
23
20
21
Pointer in capture volume

Figure 5.2.4: the pointing wand in capture volume being visualised on the distributed rendering system (*DRS) window*

7. Continue labelling and once labelling is completed, the software automatically calculates parameters and it also shows visualization of the spine in the distributed rendering system (DRS) window. Thus if labelling has been done correctly you should be able to see the visualization of the spine

Advantages

1. Calculation and visualization of 3D Parameters which are more useful than 2D parameters
2. Able to track the positions, thus capable of visualizing the subject as it moves or as parts of the subject moves. This can be used for determining the correct points to put pads that can exert forces for correcting the spine.
3. Provide better, quicker, visualization of the shape
4. Computerized data, easy to handle, revise and refer to whenever necessary
5. No radiation

Limitation

1. Inability to show inner interference of vertebrae onto other internal organs

5.3 MARKER INDEPENDENT APPLICATION

This application was developed in D-Flow as well with the purpose of reconstructing the geometry of the spine without marker attachment on the subject but only by using the pointing wand and the PHIDGET. However this application cannot be used to track the movement of the subject. This section demystifies how this application works and how it was developed. The codes for this application is indicated in appendix section. Figures 5.3.1 and 5.3.2 show the flow diagram module arrangement and the position capturing loops of this marker-less application respectively.

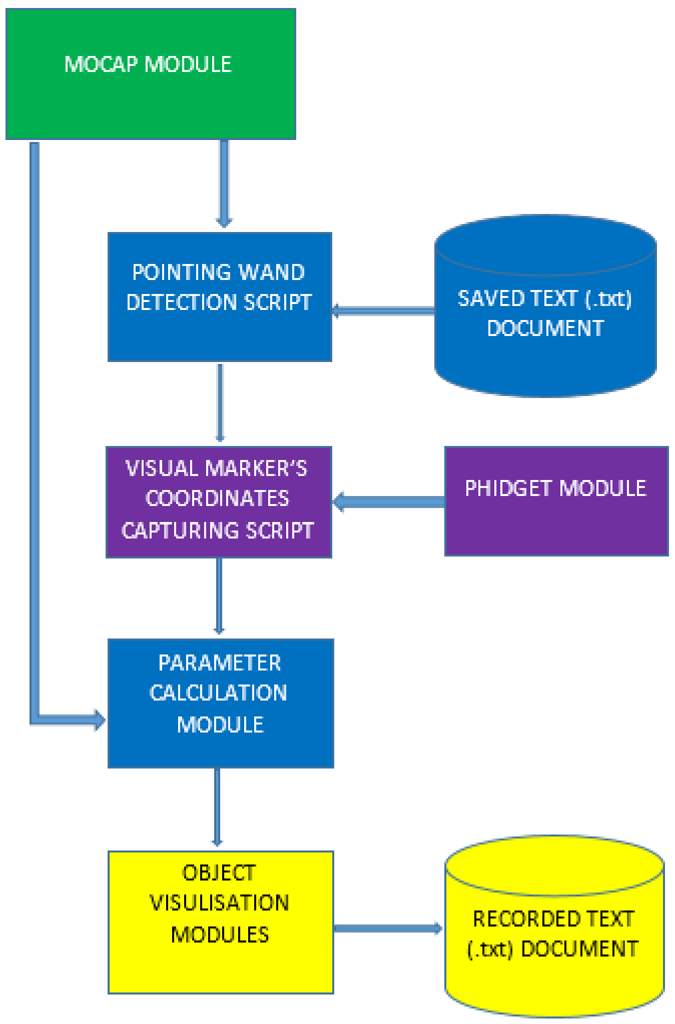

MOCAP MODULE
POINTING WAND DETECTION SCRIPT
SAVED TEXT (.txt) DOCUMENT
VISUAL MARKER'S COORDINATES CAPTURING SCRIPT
PHIDGET MODULE
PARAMETER CALCULATION MODULE
OBJECT VISULISATION MODULES
RECORDED TEXT (.txt) DOCUMENT

Figure 5.3.1: Flow diagram for the marker independent application

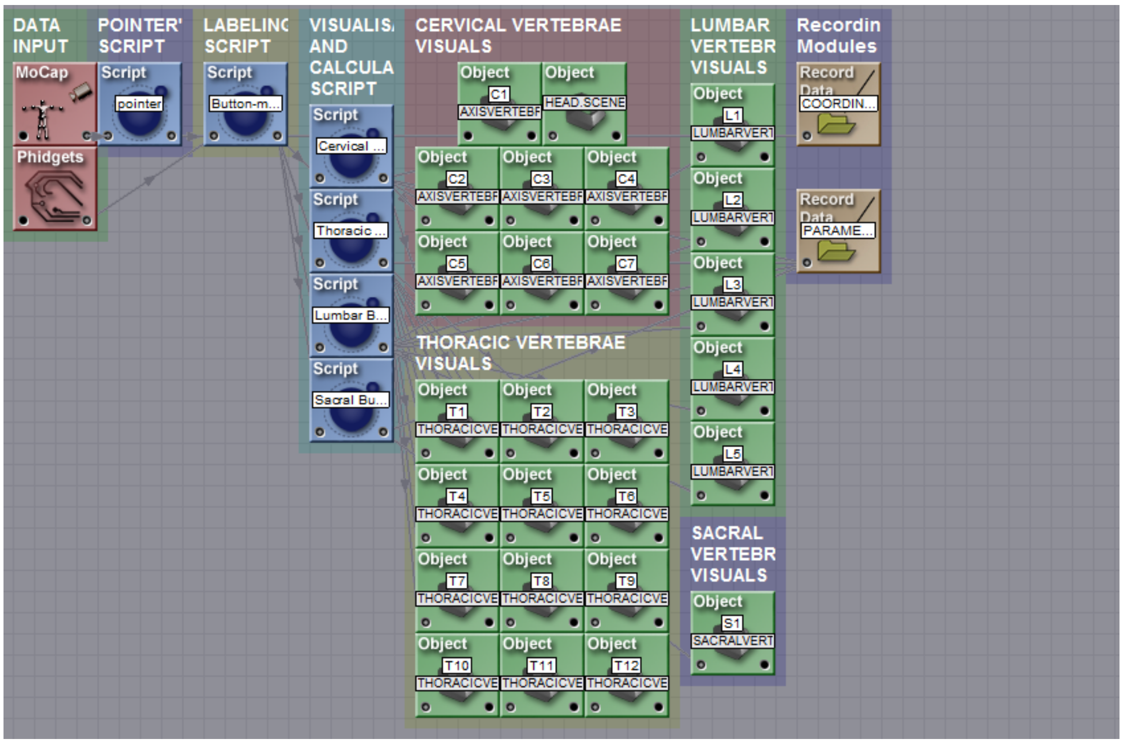


Figure 5.3.2: Modules arrangement for the marker independent application

In the first place, the development of marker independent application code was similar to that used for marker dependent application. The main deference is that in the marker independent application there is no marker labelling script, instead there is visual marker's coordinates capturing script used to capture coordinates of the tip of the pointing wand. Thus similar to the marker dependent application; MOCAP, PHIDGET, Script and record file modules were used in the marker independent application for calculation parameters and visualizing geometry of the back of subject.

The MOCAP was used to stream data of three markers on the wand while the PHIDGET module was used to receive data from touch sensor.

Besides the script for pointing wand detection was used similarly to that code in the marker dependent application for detection wand. Generally what this pointing wand detection script does is just comparing stored dimensions from wand calibration text file to the distance between markers of pointing wand in the capture volume. If the dimensions match the script creates a visual marker at the tip of the pointing wand. It is the visual marker which is used to tract the shape of the spine in this application.

Moreover the script for capturing the coordinate of the tip of the wand was developed. This script does is that it captures coordinates of the tip of the wand when the button of the touch sensor is placed. The coordinates are stored in different variables starting from cervical vertebra C1 to coccyx abbreviated as S2 in the code. The visual marker's coordinates capturing also creates a cubic object at a point where the coordinates has been captured. The creation of a cubic object is an indication or feedback that shows that the coordinates of that particular point have been captured. This script generally is uses a sandwich of conditional statements, simple statements for storing variables and "for" loop statements.

After the developing module for capturing coordinates the next module for calculating parameters was also developed. And this application reads inputs containing coordinates and make calculation of different parameters such as angles and distances. The calculated parameters and the coordinates of different points are main outputs of this script. To achieve these tasks the parameter calculation script module consists of oops and also both simple and conditional statements.

Furthermore modules of different objects created in Google Sketchup software were added and linked to correct channels of coordinates and angle parameters in order to visualize their position and orientation respectively.

Lastly the recording modules were added in order to record the required parameters and coordinates for further reference.

Steps of how to use the marker independent application

1. Open nexus software and calibrate the camera if the cameras are not calibrated.
2. Open the developed D-flow based application and make sure all markers on the subject can be seen in DRS window
3. Calibrate the wand using wand calibration application. This involves placing the wand in capture volume in a static position and placing play button on the interface of wand calibration application.
4. Open marker independent application and click play or start button on the application
5. Move wand in the capture volume and check if the visual cubic marker can be seen at the tip of the pointing wand.
6. If the visual cubic marker can be seen, you can start capturing vertebrae landmarks starting from point related to C1 to C7, then T1 –T12 and one at center of sacrum (S1) and lastly at the coccyx abbreviated as S2 in the algorithm.
7. You should be able to see a visualization of the spine and head in the DRS window or screen. If so click record button in order to record the coordinates and parameters

Advantages

1. Calculation and visualization of 3D Parameters which are more useful than 2D parameters
2. Provide better, quicker, visualization of the shape
3. Computerized data, easy to handle, revise and refer to whenever necessary
4. No radiation

Limitations

1. Inability to show inner interference of vertebrae onto other internal organs
2. Unable to track the positions, thus not capable of visualizing the subject as it moves or as parts of the subject moves.

Chapter 6: CALCULATIONS OF VARIABLES (PARAMETERS)

The applications can calculate values of different variables based on convention arithmetic theories some which are outlined below. The main aim for both applications is to calculate and visualise variables describing the geometry of the spine column of a subject. Thus the main focus is not position of the spine of the subject in the capture volume but the spine's geometry of the subject based on relative distances between adjacent points or markers. Some of the main theories used during calculation of different variables includes: Pythagoras theorem and trigonometric functions such as cosine rule, sine rule and tangent functions.

Theories of Calculating of distances, angles, rotation angle of segment and mid-point

Firstly linear distances between two points with known coordinates were calculated using Pythagoras theorem. For instance, as shown below, distance between point A and B is calculated as follows:

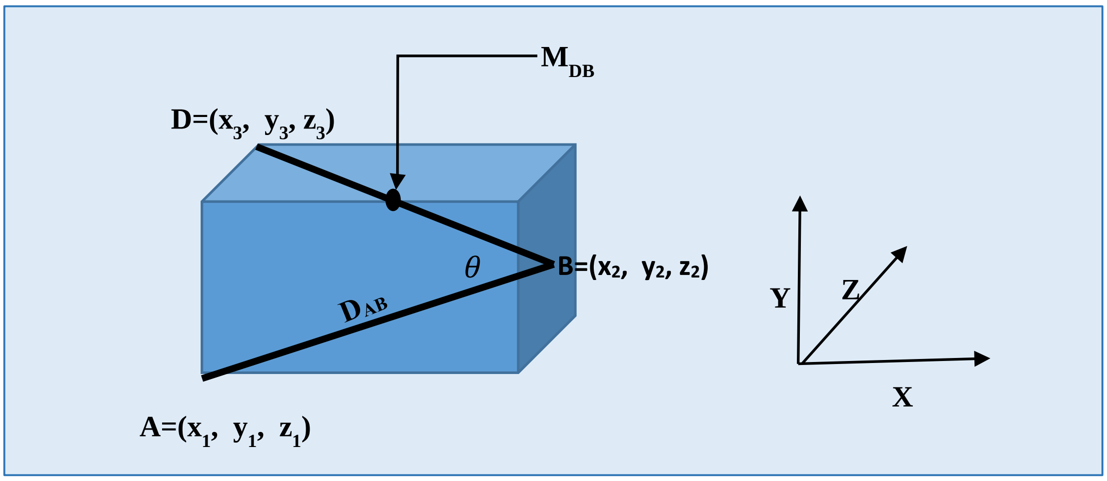
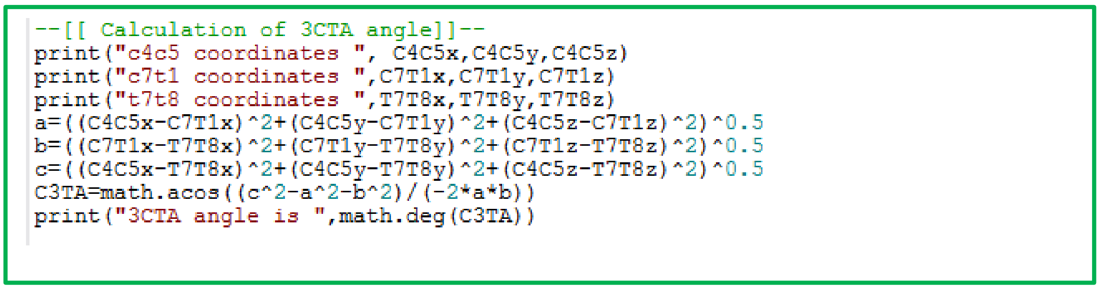


Figure 6. 1.: illustration of three points in 3D planes

- Thus distance between A and B, $\boldsymbol{D_{AB}} = \left((\boldsymbol{x_2} - \boldsymbol{x_1})^2 + (\boldsymbol{y_2} - \boldsymbol{y_1})^2 + (\boldsymbol{z_2} - \boldsymbol{z_1})^2\right)^{0.5}$
- Midpoint between point D and B $\boldsymbol{M_{DB}} = \left[\frac{(x_2+x_1)}{2}, \frac{(y_2+y_1)}{2}, \frac{(z_2+z_1)}{2}\right]$
- Applying Cosine rule , $a^2 = b^2 + c^2 - 2\,bc\,(Cos\ A)$, to find θ:

$$\theta = \cos^{-1}\left[\frac{D_{AD}^2 - D_{AB}^2 - D_{BD}^2}{-2D_{AB}D_{BD}}\right] \text{ where } D_{AB}, D_{BD} \text{ and } D_{AD} \text{ represent distances between A and B, B and D and A and D respectively}$$

- For example: in the application codes, calculations of distance "a" between C4C5 and C7T1 markers and angle between C4C5, C7T1 and T7T8 markers between was written as follows:



```
--[[ Calculation of 3CTA angle]]--
print("c4c5 coordinates ", C4C5x,C4C5y,C4C5z)
print("c7t1 coordinates ",C7T1x,C7T1y,C7T1z)
print("t7t8 coordinates ",T7T8x,T7T8y,T7T8z)
a=((C4C5x-C7T1x)^2+(C4C5y-C7T1y)^2+(C4C5z-C7T1z)^2)^0.5
b=((C7T1x-T7T8x)^2+(C7T1y-T7T8y)^2+(C7T1z-T7T8z)^2)^0.5
c=((C4C5x-T7T8x)^2+(C4C5y-T7T8y)^2+(C4C5z-T7T8z)^2)^0.5
C3TA=math.acos((c^2-a^2-b^2)/(-2*a*b))
print("3CTA angle is ",math.deg(C3TA))
```

Figure 6.1: illustration of distance and angle calculation in the code of marker dependent application

The algorithms for calculating different parameters in marker dependent and marker independent applications were somehow different. The section below indicates algorithm for calculating different parameters in the two applications.

Parameters’ calculations on cervical region

Cervical angles (CEA)

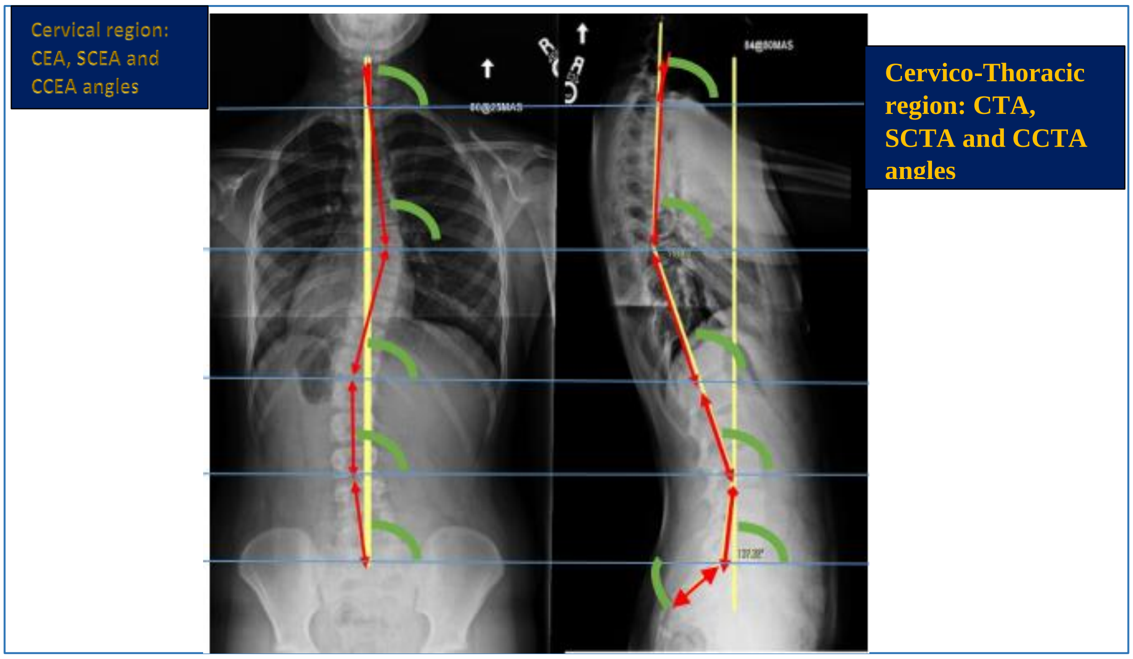
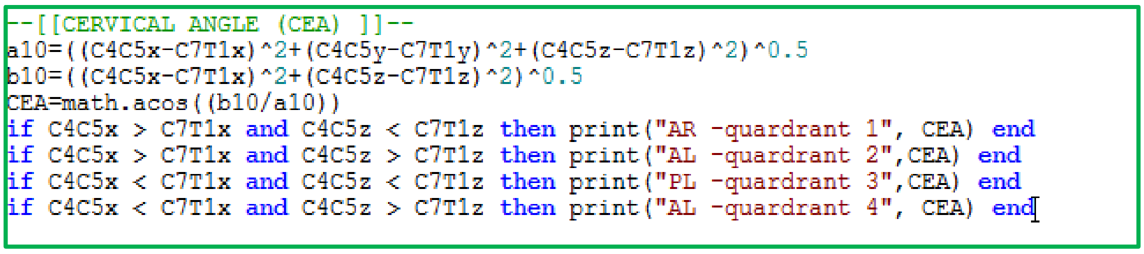

Cervical region: CEA, SCEA and CCEA angles
Cervico-Thoracic region: CTA, SCTA and CCTA angles

Figure 6. 2: X-ray image with cervical region being highlighted

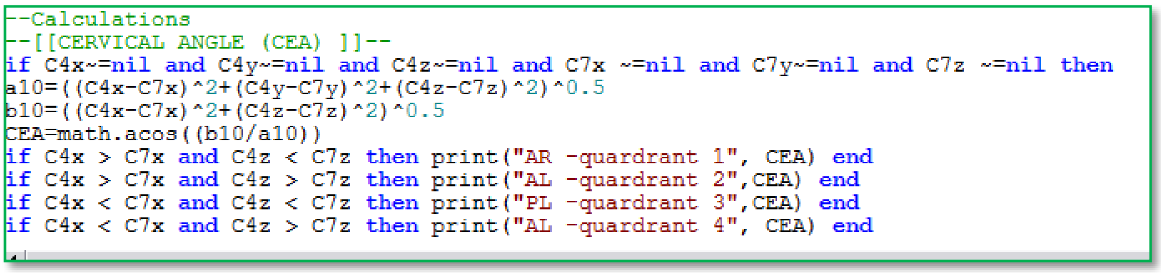
- Three dimension cervical angle C3EA
  This is the actual angle of the cervical vertebrae in three dimensions. In the marker dependent application the following code depicts how the C3EA was calculated. C4C5 and C7T1 markers’ coordinates were used to calculate cervical angles.



```
--[[CERVICAL ANGLE (CEA) ]]--
a10=((C4C5x-C7T1x)^2+(C4C5y-C7T1y)^2+(C4C5z-C7T1z)^2)^0.5
b10=((C4C5x-C7T1x)^2+(C4C5z-C7T1z)^2)^0.5
CEA=math.acos((b10/a10))
if C4C5x > C7T1x and C4C5z < C7T1z then print("AR -quardrant 1", CEA) end
if C4C5x > C7T1x and C4C5z > C7T1z then print("AL -quardrant 2",CEA) end
if C4C5x < C7T1x and C4C5z < C7T1z then print("PL -quardrant 3",CEA) end
if C4C5x < C7T1x and C4C5z > C7T1z then print("AL -quardrant 4", CEA) end
```

Figure 6. 3: script for CEA angle for marker dependent application

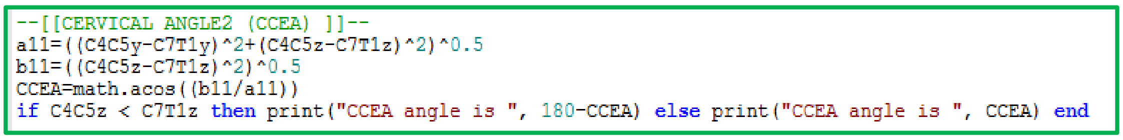
In the marker independent application the C4 and C7 points were used to calculate to calculate cervical angles.



```
--Calculations
--[[CERVICAL ANGLE (CEA) ]]--
if C4x~=nil and C4y~=nil and C4z~=nil and C7x ~=nil and C7y~=nil and C7z ~=nil then
a10=((C4x-C7x)^2+(C4y-C7y)^2+(C4z-C7z)^2)^0.5
b10=((C4x-C7x)^2+(C4z-C7z)^2)^0.5
CEA=math.acos((b10/a10))
if C4x > C7x and C4z < C7z then print("AR -quardrant 1", CEA) end
if C4x > C7x and C4z > C7z then print("AL -quardrant 2",CEA) end
if C4x < C7x and C4z < C7z then print("PL -quardrant 3",CEA) end
if C4x < C7x and C4z > C7z then print("AL -quardrant 4", CEA) end
```

Figure 6. 4 Marker independent application code for calculating CEA

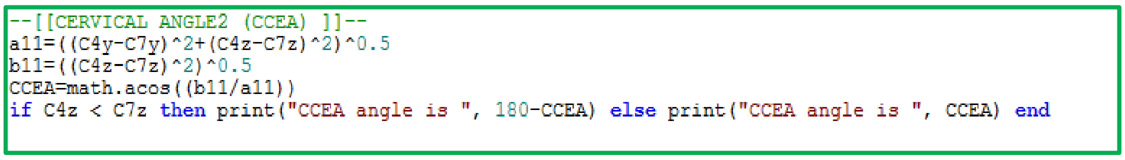
- Two dimension cervical angle in coronal plane CCEA
  This is the angle of curvature of the cervical vertebrae as it appears in coronal plane. This angle was calculated using two dimensions (y, z) of C4C5 and C7T1 markers' coordinates in markers dependent application.



```
--[[CERVICAL ANGLE2 (CCEA) ]]--
a11=((C4C5y-C7T1y)^2+(C4C5z-C7T1z)^2)^0.5
b11=((C4C5z-C7T1z)^2)^0.5
CCEA=math.acos((b11/a11))
if C4C5z < C7T1z then print("CCEA angle is ", 180-CCEA) else print("CCEA angle is ", CCEA) end
```

Figure 6. 5: CCEA for marker dependent app

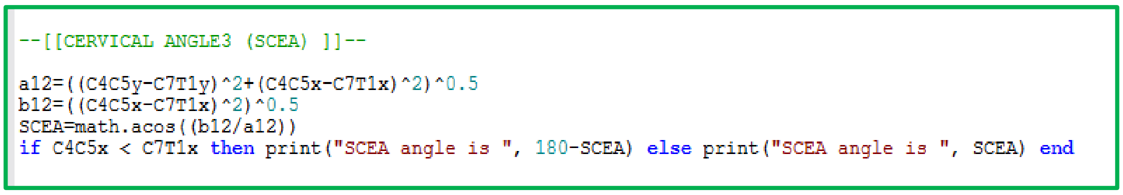
CCEA in markers independent application was calculated using two dimensions of C4 and C7 points.



```
--[[CERVICAL ANGLE2 (CCEA) ]]--
a11=((C4y-C7y)^2+(C4z-C7z)^2)^0.5
b11=((C4z-C7z)^2)^0.5
CCEA=math.acos((b11/a11))
if C4z < C7z then print("CCEA angle is ", 180-CCEA) else print("CCEA angle is ", CCEA) end
```

Figure 6. 6: CCEA for marker independent app

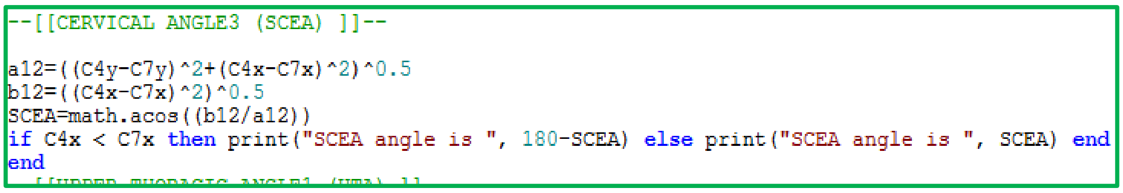
- Two dimension cervical angle in sagittal plane SCEA
  This is the angle of curvature of the cervical vertebrae as it appears in sagittal plane. This angle was calculated using two dimensions (x, y) of C4C5 and C7T1 markers' coordinates in markers dependent application as shown below.



```
--[[CERVICAL ANGLE3 (SCEA) ]]--

a12=((C4C5y-C7T1y)^2+(C4C5x-C7T1x)^2)^0.5
b12=((C4C5x-C7T1x)^2)^0.5
SCEA=math.acos((b12/a12))
if C4C5x < C7T1x then print("SCEA angle is ", 180-SCEA) else print("SCEA angle is ", SCEA) end
```

Figure 6. 7: SCEA for marker dependent app

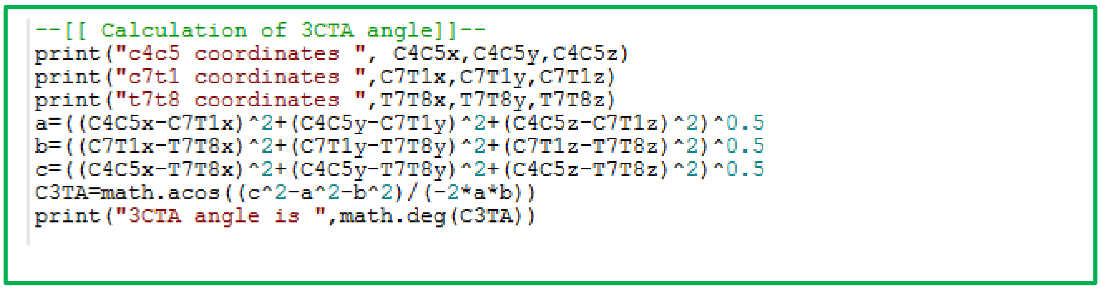
SCEA in markers independent application was calculated using two dimensions(x,y) of C4 and C7 points as follows:



```
--[[CERVICAL ANGLE3 (SCEA) ]]--

a12=((C4y-C7y)^2+(C4x-C7x)^2)^0.5
b12=((C4x-C7x)^2)^0.5
SCEA=math.acos((b12/a12))
if C4x < C7x then print("SCEA angle is ", 180-SCEA) else print("SCEA angle is ", SCEA) end
end
```

Figure 6. 8: SCEA for marker independent app

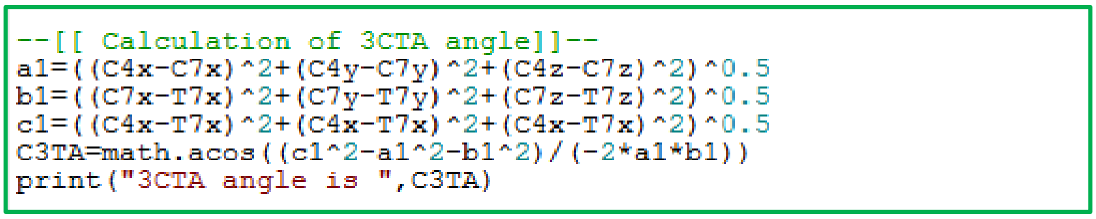
- Three dimensional Cervico-Thoracic angle 3CTA.
  This is the actual angle of curvature of spine at cervico-thoracic region calculated using three dimensions. In markers' dependent application, this angle was calculated using the following algorithm.



```
--[[ Calculation of 3CTA angle]]--
print("c4c5 coordinates ", C4C5x,C4C5y,C4C5z)
print("c7t1 coordinates ",C7T1x,C7T1y,C7T1z)
print("t7t8 coordinates ",T7T8x,T7T8y,T7T8z)
a=((C4C5x-C7T1x)^2+(C4C5y-C7T1y)^2+(C4C5z-C7T1z)^2)^0.5
b=((C7T1x-T7T8x)^2+(C7T1y-T7T8y)^2+(C7T1z-T7T8z)^2)^0.5
c=((C4C5x-T7T8x)^2+(C4C5y-T7T8y)^2+(C4C5z-T7T8z)^2)^0.5
C3TA=math.acos((c^2-a^2-b^2)/(-2*a*b))
print("3CTA angle is ",math.deg(C3TA))
```

Figure 6. 9: CTA for marker dependent app

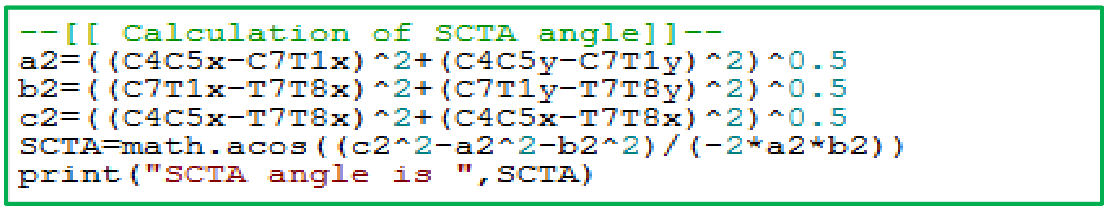
  CTA angle was calculated using the following algorithm in the marker independent application.



```
--[[ Calculation of 3CTA angle]]--
a1=((C4x-C7x)^2+(C4y-C7y)^2+(C4z-C7z)^2)^0.5
b1=((C7x-T7x)^2+(C7y-T7y)^2+(C7z-T7z)^2)^0.5
c1=((C4x-T7x)^2+(C4x-T7x)^2+(C4x-T7x)^2)^0.5
C3TA=math.acos((c1^2-a1^2-b1^2)/(-2*a1*b1))
print("3CTA angle is ",C3TA)
```

Figure 6. 10: CTA for marker independent app

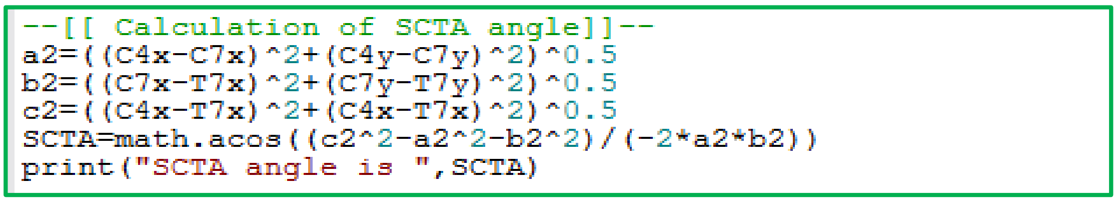
- Two dimensional sagittal cervico-thoracic angle SCTA
  This is the angle of curvature of spine at cervico-thoracic region as it appears in sagittal plane calculated using two dimensions (x, y). In markers' dependent application, this angle was calculated using the following script.



```
--[[ Calculation of SCTA angle]]--
a2=((C4C5x-C7T1x)^2+(C4C5y-C7T1y)^2)^0.5
b2=((C7T1x-T7T8x)^2+(C7T1y-T7T8y)^2)^0.5
c2=((C4C5x-T7T8x)^2+(C4C5x-T7T8x)^2)^0.5
SCTA=math.acos((c2^2-a2^2-b2^2)/(-2*a2*b2))
print("SCTA angle is ",SCTA)
```

Figure 6. 11: SCTA for marker dependent app

  The following script illustrates how SCTA angle was calculated in the marker independent application.



```
--[[ Calculation of SCTA angle]]--
a2=((C4x-C7x)^2+(C4y-C7y)^2)^0.5
b2=((C7x-T7x)^2+(C7y-T7y)^2)^0.5
c2=((C4x-T7x)^2+(C4x-T7x)^2)^0.5
SCTA=math.acos((c2^2-a2^2-b2^2)/(-2*a2*b2))
print("SCTA angle is ",SCTA)
```

Figure 6.12: SCTA for marker independent app

Parameters’ calculations on thoracic region

Upper thoracic angles

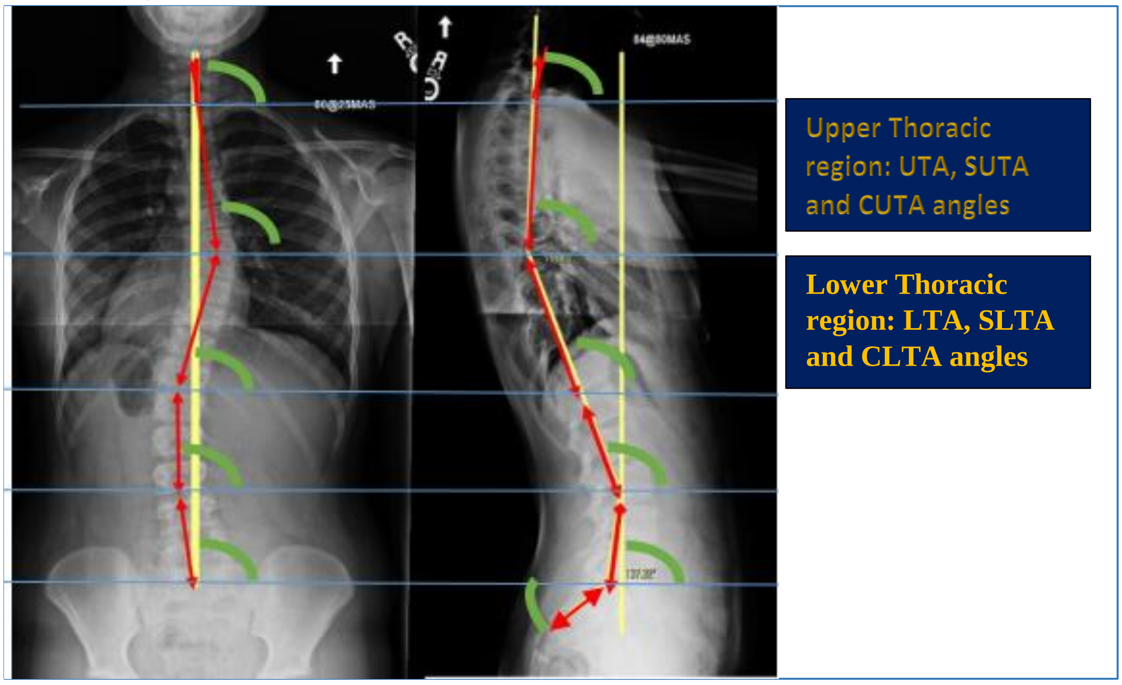
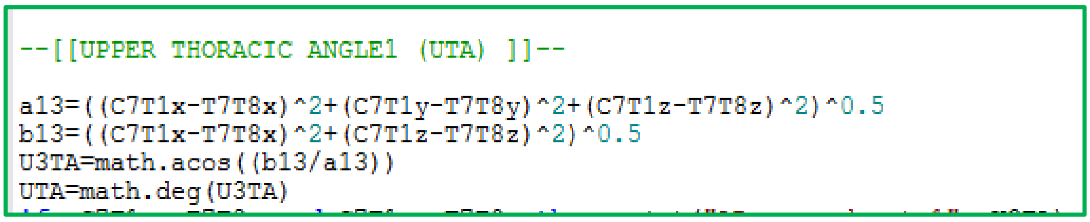

Upper Thoracic region: UTA, SUTA and CUTA angles
Lower Thoracic region: LTA, SLTA and CLTA angles

Figure 6. 13: Figure CC 10: X-ray image highlighting Thoracic region

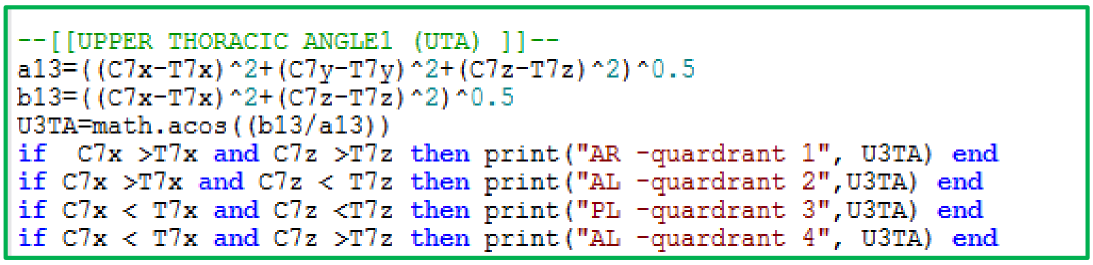
- Three dimension upper thoracic angle U3TA
  This is the actual angle of curvature of spine at the upper thoracic region calculated using three dimensions (x, y, z). In markers’ dependent application, this angle was calculated using coordinates of C7T1 and T7T8 markers. The following algorithm summarises how to calculate this angle



```
--[[UPPER THORACIC ANGLE1 (UTA) ]]--

a13=((C7T1x-T7T8x)^2+(C7T1y-T7T8y)^2+(C7T1z-T7T8z)^2)^0.5
b13=((C7T1x-T7T8x)^2+(C7T1z-T7T8z)^2)^0.5
U3TA=math.acos((b13/a13))
UTA=math.deg(U3TA)
```

Figure 6. 14: UTA for marker dependent app

In the markers’ independent application the following algorithm to calculating U3TA was developed.

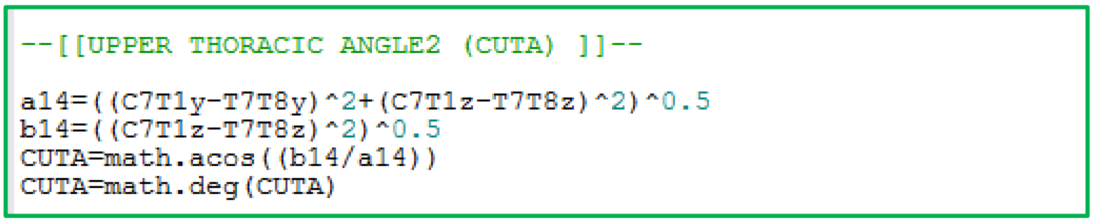


```
--[[UPPER THORACIC ANGLE1 (UTA) ]]--
a13=((C7x-T7x)^2+(C7y-T7y)^2+(C7z-T7z)^2)^0.5
b13=((C7x-T7x)^2+(C7z-T7z)^2)^0.5
U3TA=math.acos((b13/a13))
if  C7x >T7x and C7z >T7z then print("AR -quardrant 1", U3TA) end
if C7x >T7x and C7z < T7z then print("AL -quardrant 2",U3TA) end
if C7x < T7x and C7z <T7z then print("PL -quardrant 3",U3TA) end
if C7x < T7x and C7z >T7z then print("AL -quardrant 4", U3TA) end
```

Figure 6. 15:UTA for marker independent app

- Two dimension upper thoracic angle in coronal plane CUTA

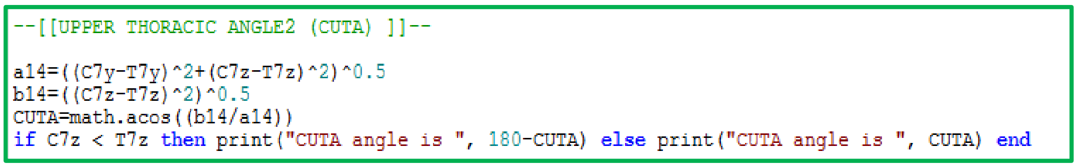
This is the angle of curvature of spine at the upper thoracic region calculated using two dimensions (z, y) as viewed in coronal plane. In markers' dependent application, this angle was calculated using coordinates of C7T1 and T7T8 markers. The following procedure summarises how to calculate this angle in the markers dependent application



```
--[[UPPER THORACIC ANGLE2 (CUTA) ]]--

a14=((C7T1y-T7T8y)^2+(C7T1z-T7T8z)^2)^0.5
b14=((C7T1z-T7T8z)^2)^0.5
CUTA=math.acos((b14/a14))
CUTA=math.deg(CUTA)
```

Figure 6. 16: CUTA for marker dependent app

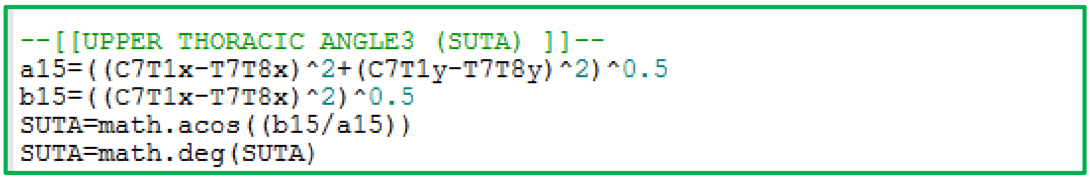
The following procedure was developed to calculate CUTA in the markers' independent application.



```
--[[UPPER THORACIC ANGLE2 (CUTA) ]]--

a14=((C7y-T7y)^2+(C7z-T7z)^2)^0.5
b14=((C7z-T7z)^2)^0.5
CUTA=math.acos((b14/a14))
if C7z < T7z then print("CUTA angle is ", 180-CUTA) else print("CUTA angle is ", CUTA) end
```

Figure 6. 17: CUTA for marker independent app

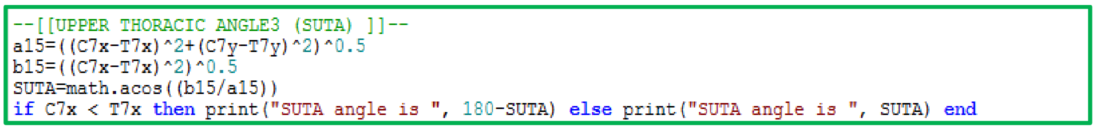
- Two dimension upper thoracic angle in sagittal plane SUTA
  Sagittal upper thoracic angle (SUTA) is the angle of curvature of spine at the upper thoracic region calculated using two dimensions (z, y) as viewed in sagittal plane. In markers' dependent application, this angle was calculated using coordinates of C7T1 and T7T8 markers. The script below summarises how to calculate this angle in the markers dependent application.



```
--[[UPPER THORACIC ANGLE3 (SUTA) ]]--
a15=((C7T1x-T7T8x)^2+(C7T1y-T7T8y)^2)^0.5
b15=((C7T1x-T7T8x)^2)^0.5
SUTA=math.acos((b15/a15))
SUTA=math.deg(SUTA)
```

Figure 6.18: SUTA for marker dependent app

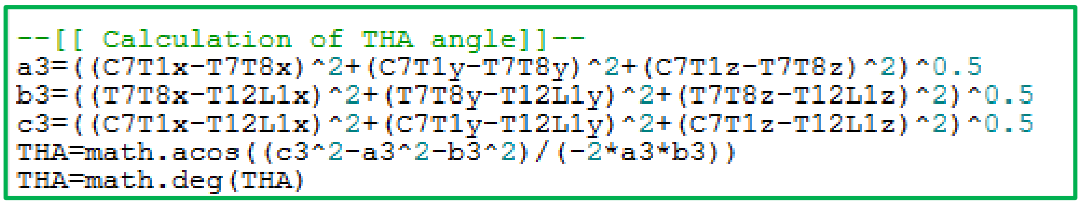
The script below, was developed to calculate SUTA in the markers' independent application.



```
--[[UPPER THORACIC ANGLE3 (SUTA) ]]--
a15=((C7x-T7x)^2+(C7y-T7y)^2)^0.5
b15=((C7x-T7x)^2)^0.5
SUTA=math.acos((b15/a15))
if C7x < T7x then print("SUTA angle is ", 180-SUTA) else print("SUTA angle is ", SUTA) end
```

Figure 6. 19: SUTA for marker independent app

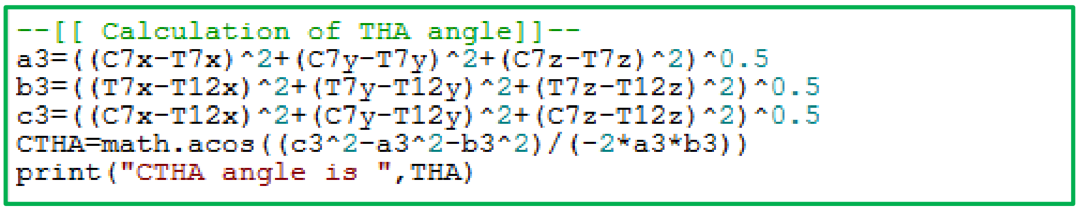
- Three dimensional Kyphosis thoracic angle 3THA
  Three dimensional kyphosis thoracic angle of curvature is the actual posterior curvature angle of the thoracic section of the vertebrae column. In the script of markers dependent application THA angle is calculated as follows:



```
--[[ Calculation of THA angle]]--
a3=((C7T1x-T7T8x)^2+(C7T1y-T7T8y)^2+(C7T1z-T7T8z)^2)^0.5
b3=((T7T8x-T12L1x)^2+(T7T8y-T12L1y)^2+(T7T8z-T12L1z)^2)^0.5
c3=((C7T1x-T12L1x)^2+(C7T1y-T12L1y)^2+(C7T1z-T12L1z)^2)^0.5
THA=math.acos((c3^2-a3^2-b3^2)/(-2*a3*b3))
THA=math.deg(THA)
```

Figure 6.20: THA for marker dependent

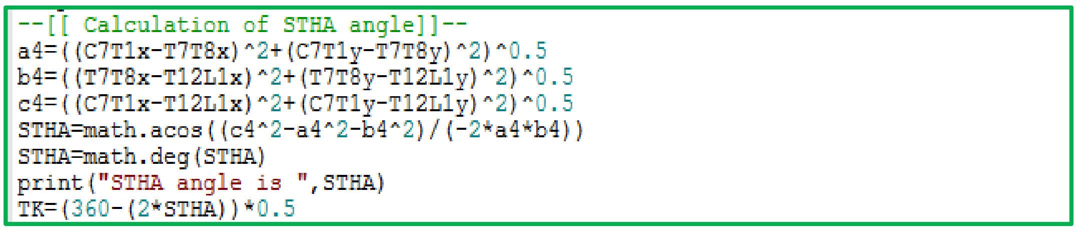
Besides in the script of markers independent application THA angle is calculated as follows:



```
--[[ Calculation of THA angle]]--
a3=((C7x-T7x)^2+(C7y-T7y)^2+(C7z-T7z)^2)^0.5
b3=((T7x-T12x)^2+(T7y-T12y)^2+(T7z-T12z)^2)^0.5
c3=((C7x-T12x)^2+(C7y-T12y)^2+(C7z-T12z)^2)^0.5
CTHA=math.acos((c3^2-a3^2-b3^2)/(-2*a3*b3))
print("CTHA angle is ",THA)
```

Figure 6. 21: THA for marker independent app

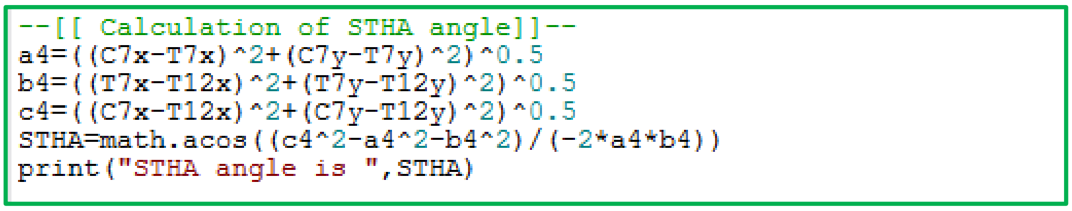
- Two dimensional sagittal Kyphosis thoracic angle STHA
  Two dimensional sagittal kyphosis thoracic angle (STHA) of curvature is the posterior curvature angle of the thoracic section as viewed from sagittal plane of the vertebrae column. In the script of markers dependent application STHA angle is calculated as follows:



```
--[[ Calculation of STHA angle]]--
a4=((C7T1x-T7T8x)^2+(C7T1y-T7T8y)^2)^0.5
b4=((T7T8x-T12L1x)^2+(T7T8y-T12L1y)^2)^0.5
c4=((C7T1x-T12L1x)^2+(C7T1y-T12L1y)^2)^0.5
STHA=math.acos((c4^2-a4^2-b4^2)/(-2*a4*b4))
STHA=math.deg(STHA)
print("STHA angle is ",STHA)
TK=(360-(2*STHA))*0.5
```

Figure 6. 22: STHA for marker dependent app

Moreover for markers independent application STHA angle is calculated as shown in the script below:

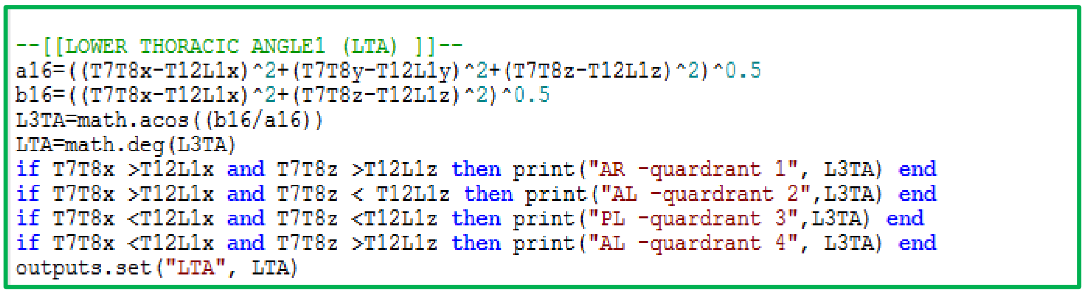


```
--[[ Calculation of STHA angle]]--
a4=((C7x-T7x)^2+(C7y-T7y)^2)^0.5
b4=((T7x-T12x)^2+(T7y-T12y)^2)^0.5
c4=((C7x-T12x)^2+(C7y-T12y)^2)^0.5
STHA=math.acos((c4^2-a4^2-b4^2)/(-2*a4*b4))
print("STHA angle is ",STHA)
```

Figure 6. 23: STHA for marker independent app

Lower thoracic angles

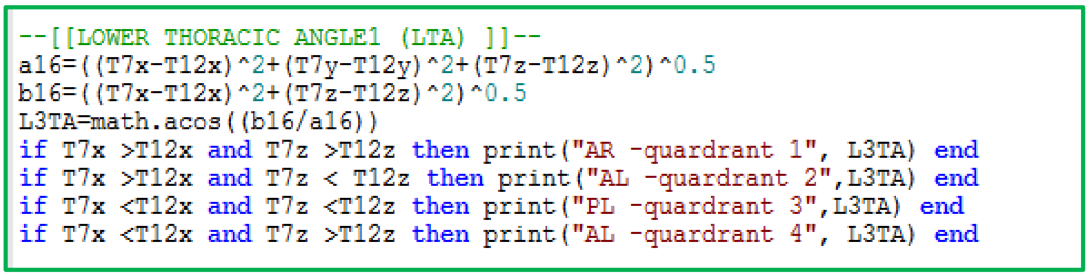
- Three dimension lower thoracic angle 3LTA
  Three dimensional lower thoracic angle is the actual angle of curvature of the spine at the lower thoracic region. This angle is calculated using three dimensional coordinates of T7T8 and T12L1 markers in the markers dependent application while in markers independent application the angle is calculated using three dimensional coordinates of T7 and T12 points. The script below outlines the procedure of calculating LTA angle in markers dependent application.



```
--[[LOWER THORACIC ANGLE1 (LTA) ]]--
a16=((T7T8x-T12L1x)^2+(T7T8y-T12L1y)^2+(T7T8z-T12L1z)^2)^0.5
b16=((T7T8x-T12L1x)^2+(T7T8z-T12L1z)^2)^0.5
L3TA=math.acos((b16/a16))
LTA=math.deg(L3TA)
if T7T8x >T12L1x and T7T8z >T12L1z then print("AR -quardrant 1", L3TA) end
if T7T8x >T12L1x and T7T8z < T12L1z then print("AL -quardrant 2",L3TA) end
if T7T8x <T12L1x and T7T8z <T12L1z then print("PL -quardrant 3",L3TA) end
if T7T8x <T12L1x and T7T8z >T12L1z then print("AL -quardrant 4", L3TA) end
outputs.set("LTA", LTA)
```

Figure 6. 24: LTA for marker dependent app

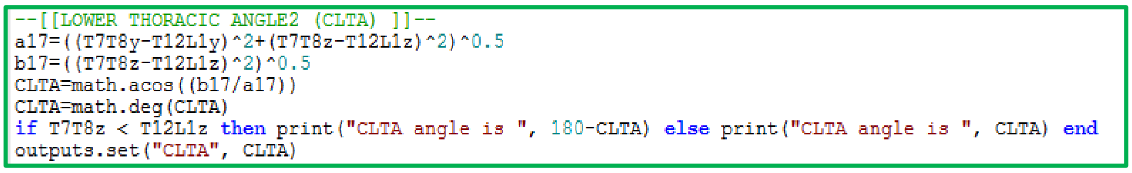
The script below shows how to calculate the LTA angle in markers independent application.



```
--[[LOWER THORACIC ANGLE1 (LTA) ]]--
a16=((T7x-T12x)^2+(T7y-T12y)^2+(T7z-T12z)^2)^0.5
b16=((T7x-T12x)^2+(T7z-T12z)^2)^0.5
L3TA=math.acos((b16/a16))
if T7x >T12x and T7z >T12z then print("AR -quardrant 1", L3TA) end
if T7x >T12x and T7z < T12z then print("AL -quardrant 2",L3TA) end
if T7x <T12x and T7z <T12z then print("PL -quardrant 3",L3TA) end
if T7x <T12x and T7z >T12z then print("AL -quardrant 4", L3TA) end
```

Figure 6. 25: LTA for marker independent

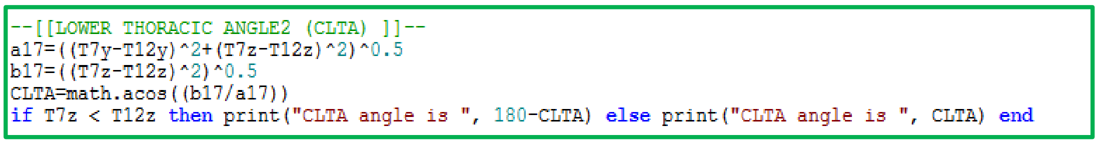
- Two dimension lower thoracic angle in coronal plane CLTA
  Two dimensional lower thoracic angle is the angle of curvature of the spine at the lower thoracic region as viewed in coronal plane. This angle is calculated using two dimensional coordinates (y , z) of T7T8 and T12L1 markers in the markers dependent application while in markers independent application the angle is calculated using two dimensional coordinates of T7 and T12 points. The algorithm of calculating CLTA angle in markers dependent application is shown below.



```
--[[LOWER THORACIC ANGLE2 (CLTA) ]]--
a17=((T7T8y-T12L1y)^2+(T7T8z-T12L1z)^2)^0.5
b17=((T7T8z-T12L1z)^2)^0.5
CLTA=math.acos((b17/a17))
CLTA=math.deg(CLTA)
if T7T8z < T12L1z then print("CLTA angle is ", 180-CLTA) else print("CLTA angle is ", CLTA) end
outputs.set("CLTA", CLTA)
```

Figure 6. 26: CLTA for marker dependent app

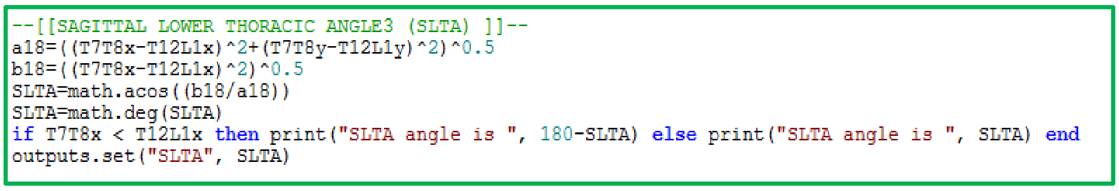
The script of how to calculate the CLTA angle in markers independent application is illustrated below.



```
--[[LOWER THORACIC ANGLE2 (CLTA) ]]--
a17=((T7y-T12y)^2+(T7z-T12z)^2)^0.5
b17=((T7z-T12z)^2)^0.5
CLTA=math.acos((b17/a17))
if T7z < T12z then print("CLTA angle is ", 180-CLTA) else print("CLTA angle is ", CLTA) end
```

Figure 6. 27: CLTA for marker independent app

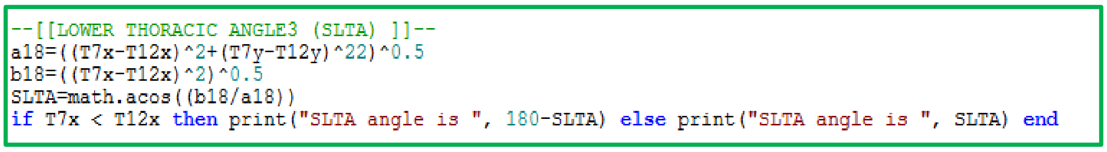
- Two dimension lower thoracic angle in sagittal plane SLTA
  Two dimensional lower thoracic angle is the angle of curvature of the spine at the lower thoracic region as viewed in sagittal plane. This angle is calculated using two dimensional coordinates (x,y) of T7T8 and T12L1 markers in the markers dependent application while in markers independent application the angle is calculated using two dimensional coordinates of T7 and T12 points. The following algorithm shows the procedure of calculating SLTA angle in markers dependent application.



```
--[[SAGITTAL LOWER THORACIC ANGLE3 (SLTA) ]]--
a18=((T7T8x-T12L1x)^2+(T7T8y-T12L1y)^2)^0.5
b18=((T7T8x-T12L1x)^2)^0.5
SLTA=math.acos((b18/a18))
SLTA=math.deg(SLTA)
if T7T8x < T12L1x then print("SLTA angle is ", 180-SLTA) else print("SLTA angle is ", SLTA) end
outputs.set("SLTA", SLTA)
```

Figure 6. 28: SLTA for marker dependent app

The following script illustrates the calculation of SLTA angle in markers independent application.

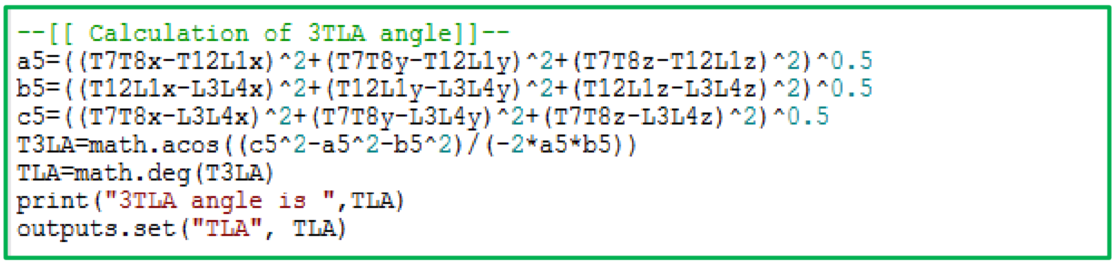


```
--[[LOWER THORACIC ANGLE3 (SLTA) ]]--
a18=((T7x-T12x)^2+(T7y-T12y)^22)^0.5
b18=((T7x-T12x)^2)^0.5
SLTA=math.acos((b18/a18))
if T7x < T12x then print("SLTA angle is ", 180-SLTA) else print("SLTA angle is ", SLTA) end
```

Figure 6. 29: SLTA for marker independent app

- Three dimensional Thoraco-lumbar angle 3TLA

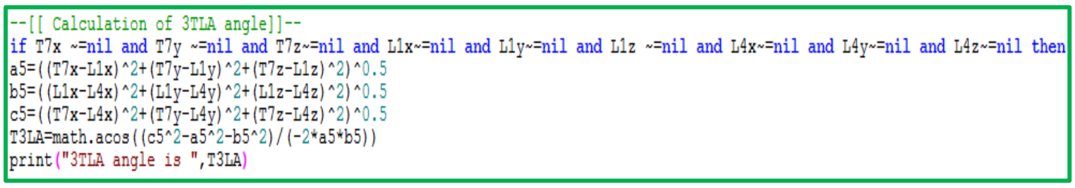
  Thoraco-lumbar angle is an actual angle of curvature of the thoracic-lumbar region calculated using three dimensional coordinates (x,y and z). In the markers dependent application the TLA angle was calculated using T7T8, L1L2 and L3L4 markers' coordinates while in the markers dependent application as shown below.



```
--[[ Calculation of 3TLA angle]]--
a5=((T7T8x-T12L1x)^2+(T7T8y-T12L1y)^2+(T7T8z-T12L1z)^2)^0.5
b5=((T12L1x-L3L4x)^2+(T12L1y-L3L4y)^2+(T12L1z-L3L4z)^2)^0.5
c5=((T7T8x-L3L4x)^2+(T7T8y-L3L4y)^2+(T7T8z-L3L4z)^2)^0.5
T3LA=math.acos((c5^2-a5^2-b5^2)/(-2*a5*b5))
TLA=math.deg(T3LA)
print("3TLA angle is ",TLA)
outputs.set("TLA", TLA)
```

Figure 6. 30:TLA for marker dependent app

On the other hand the following algorithm shows how to calculate the 3TLA angle in the markers' independent application.

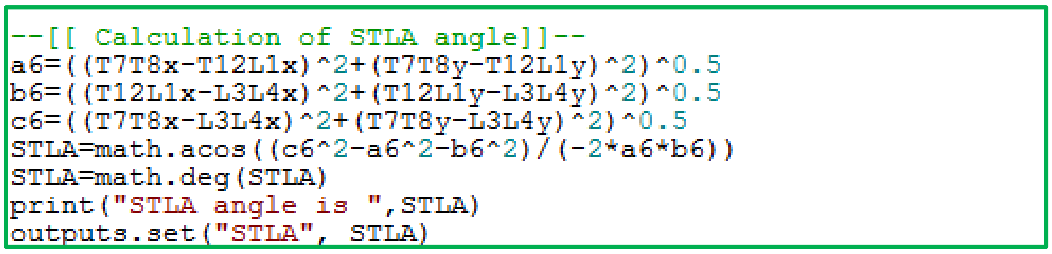


```
--[[ Calculation of 3TLA angle]]--
if T7x ~=nil and T7y ~=nil and T7z~=nil and L1x~=nil and L1y~=nil and L1z ~=nil and L4x~=nil and L4y~=nil and L4z~=nil then
a5=((T7x-L1x)^2+(T7y-L1y)^2+(T7z-L1z)^2)^0.5
b5=((L1x-L4x)^2+(L1y-L4y)^2+(L1z-L4z)^2)^0.5
c5=((T7x-L4x)^2+(T7y-L4y)^2+(T7z-L4z)^2)^0.5
T3LA=math.acos((c5^2-a5^2-b5^2)/(-2*a5*b5))
print("3TLA angle is ",T3LA)
```

Figure 6.31: TLA for marker independent app

- Two dimensional sagittal plane Thoraco-lumbar angle STLA

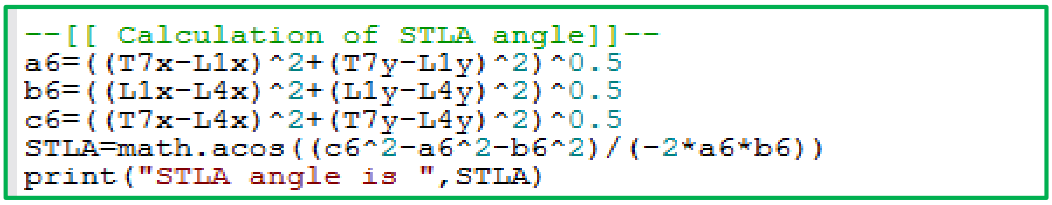
  Sagittal Thoraco-lumbar angle is an angle of curvature of the thoracic-lumbar region as viewed in sagittal plane and calculated using two dimensional coordinates (x,y). In the markers dependent application the STLA angle was calculated using T7T8, L1L2 and L3L4 markers' coordinates while in the markers dependent application as shown below.



```
--[[ Calculation of STLA angle]]--
a6=((T7T8x-T12L1x)^2+(T7T8y-T12L1y)^2)^0.5
b6=((T12L1x-L3L4x)^2+(T12L1y-L3L4y)^2)^0.5
c6=((T7T8x-L3L4x)^2+(T7T8y-L3L4y)^2)^0.5
STLA=math.acos((c6^2-a6^2-b6^2)/(-2*a6*b6))
STLA=math.deg(STLA)
print("STLA angle is ",STLA)
outputs.set("STLA", STLA)
```

Figure 6. 32: STLA for marker dependent app

Besides the following script shows how to calculate the STLA angle in the markers' independent application.



```
--[[ Calculation of STLA angle]]--
a6=((T7x-L1x)^2+(T7y-L1y)^2)^0.5
b6=((L1x-L4x)^2+(L1y-L4y)^2)^0.5
c6=((T7x-L4x)^2+(T7y-L4y)^2)^0.5
STLA=math.acos((c6^2-a6^2-b6^2)/(-2*a6*b6))
print("STLA angle is ",STLA)
```

Figure 6. 33: STLA for marker independent app

Parameters' calculations on lumbar region

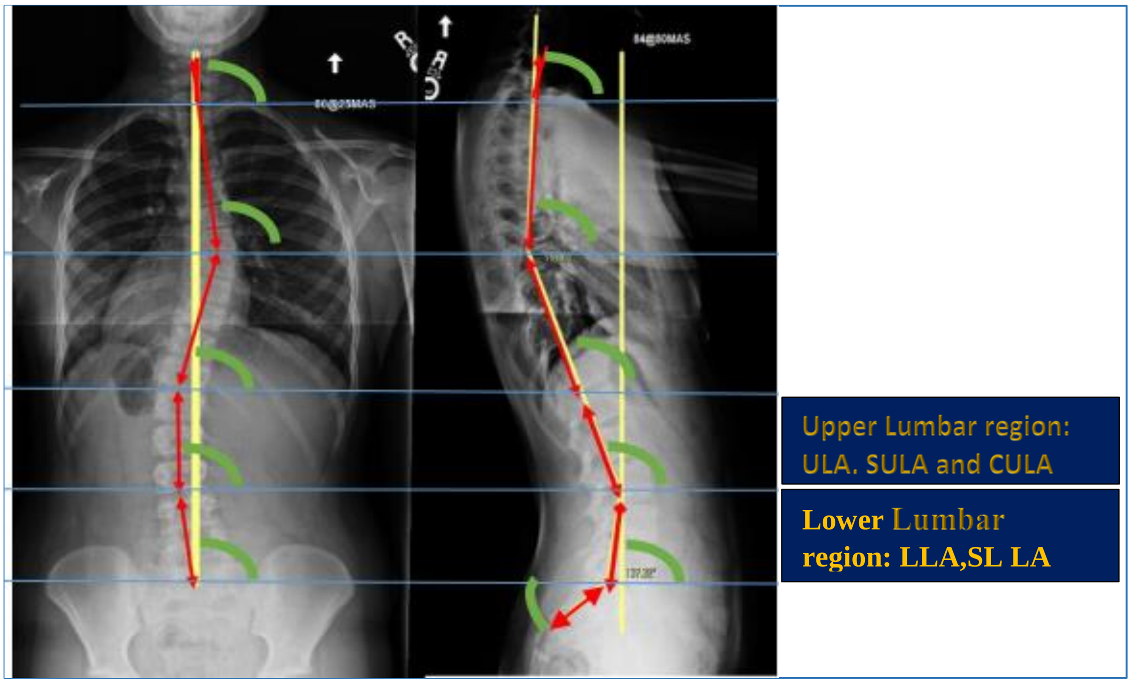
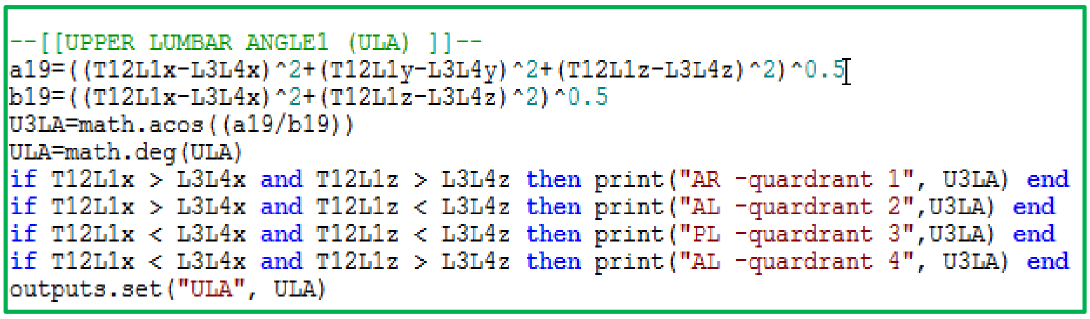

Upper Lumbar region: ULA. SULA and CULA
Lower Lumbar region: LLA,SL LA

Figure 6.34: X-ray image highlighting lumbar region

Upper lumbar angles

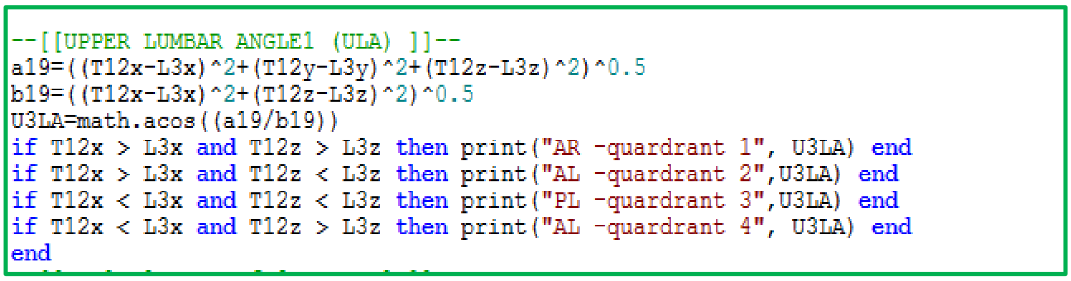
- Three dimension upper lumbar angle 3ULA
  This is the actual angle of curvature on the upper lumbar region of the vertebrae column calculated using three dimensional coordinates (x, y and z). The following script depict how the ULA angle is calculated in markers dependent application.



```
--[[UPPER LUMBAR ANGLE1 (ULA) ]]--
a19=((T12L1x-L3L4x)^2+(T12L1y-L3L4y)^2+(T12L1z-L3L4z)^2)^0.5
b19=((T12L1x-L3L4x)^2+(T12L1z-L3L4z)^2)^0.5
U3LA=math.acos((a19/b19))
ULA=math.deg(ULA)
if T12L1x > L3L4x and T12L1z > L3L4z then print("AR -quardrant 1", U3LA) end
if T12L1x > L3L4x and T12L1z < L3L4z then print("AL -quardrant 2",U3LA) end
if T12L1x < L3L4x and T12L1z < L3L4z then print("PL -quardrant 3",U3LA) end
if T12L1x < L3L4x and T12L1z > L3L4z then print("AL -quardrant 4", U3LA) end
outputs.set("ULA", ULA)
```

Figure 6. 35: ULA for marker dependent app

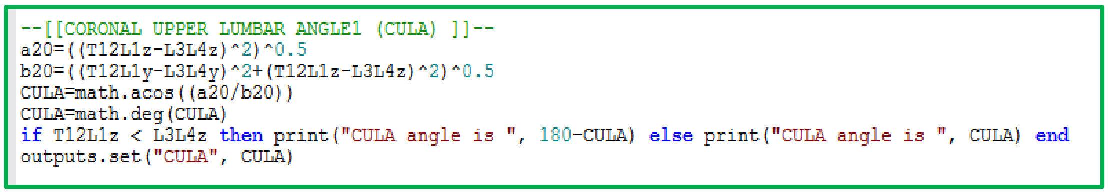
Besides the ULA angle is calculated as follows in the markers independent application.



```
--[[UPPER LUMBAR ANGLE1 (ULA) ]]--
a19=((T12x-L3x)^2+(T12y-L3y)^2+(T12z-L3z)^2)^0.5
b19=((T12x-L3x)^2+(T12z-L3z)^2)^0.5
U3LA=math.acos((a19/b19))
if T12x > L3x and T12z > L3z then print("AR -quardrant 1", U3LA) end
if T12x > L3x and T12z < L3z then print("AL -quardrant 2",U3LA) end
if T12x < L3x and T12z < L3z then print("PL -quardrant 3",U3LA) end
if T12x < L3x and T12z > L3z then print("AL -quardrant 4", U3LA) end
end
```

Figure 6. 36: ULA for marker independent app

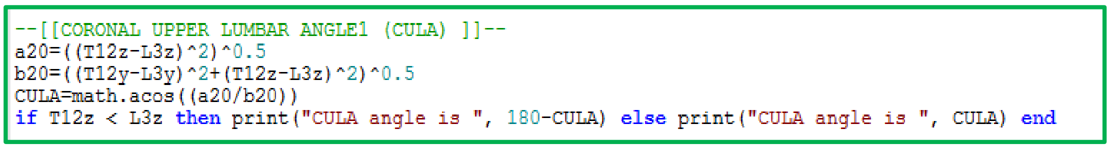
- Two dimension upper lumbar angle in coronal plane CULA
  This is the angle of curvature on the upper lumbar region of the vertebrae column as viewed in coronal plane and calculated using two dimensional coordinates (y and z). The following script shows the equations that were developed to guide the calculation of CULA angle in the markers dependent application.



```
--[[CORONAL UPPER LUMBAR ANGLE1 (CULA) ]]--
a20=((T12L1z-L3L4z)^2)^0.5
b20=((T12L1y-L3L4y)^2+(T12L1z-L3L4z)^2)^0.5
CULA=math.acos((a20/b20))
CULA=math.deg(CULA)
if T12L1z < L3L4z then print("CULA angle is ", 180-CULA) else print("CULA angle is ", CULA) end
outputs.set("CULA", CULA)
```

Figure 6. 37: CULA for marker dependent app

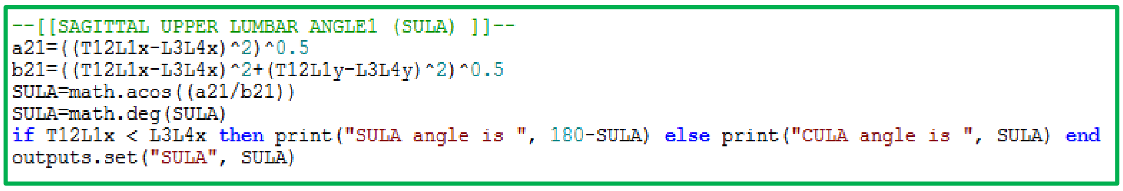
However for the markers independent application, the following equations shown in the following script for calculating CULA angle was developed.



```
--[[CORONAL UPPER LUMBAR ANGLE1 (CULA) ]]--
a20=((T12z-L3z)^2)^0.5
b20=((T12y-L3y)^2+(T12z-L3z)^2)^0.5
CULA=math.acos((a20/b20))
if T12z < L3z then print("CULA angle is ", 180-CULA) else print("CULA angle is ", CULA) end
```

Figure 6. 38: CULA for marker independent app

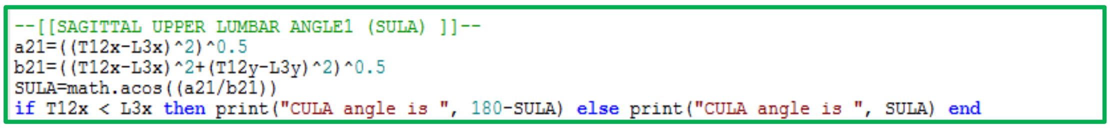
- Two dimension upper lumbar angle in sagittal plane SULA
  This is the angle of curvature on the upper lumbar region of the vertebrae column as viewed in sagittal plane and calculated using three dimensional coordinates (x, y). The following script shows how to calculate SULA angle in the markers dependent application.



```
--[[SAGITTAL UPPER LUMBAR ANGLE1 (SULA) ]]--
a21=((T12L1x-L3L4x)^2)^0.5
b21=((T12L1x-L3L4x)^2+(T12L1y-L3L4y)^2)^0.5
SULA=math.acos((a21/b21))
SULA=math.deg(SULA)
if T12L1x < L3L4x then print("SULA angle is ", 180-SULA) else print("CULA angle is ", SULA) end
outputs.set("SULA", SULA)
```

Figure 6.39: SULA for marker dependent app

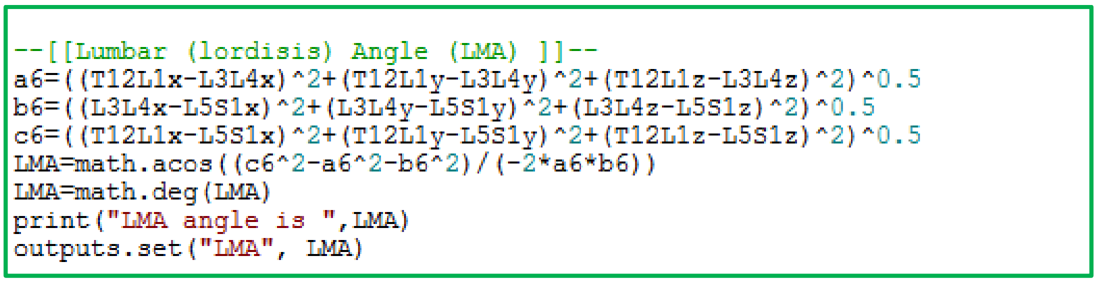
Furthermore, following script shows how to calculate SULA angle was developed.



```
--[[SAGITTAL UPPER LUMBAR ANGLE1 (SULA) ]]--
a21=((T12x-L3x)^2)^0.5
b21=((T12x-L3x)^2+(T12y-L3y)^2)^0.5
SULA=math.acos((a21/b21))
if T12x < L3x then print("CULA angle is ", 180-SULA) else print("CULA angle is ", SULA) end
```

Figure 6.40: SULA for marker independent app

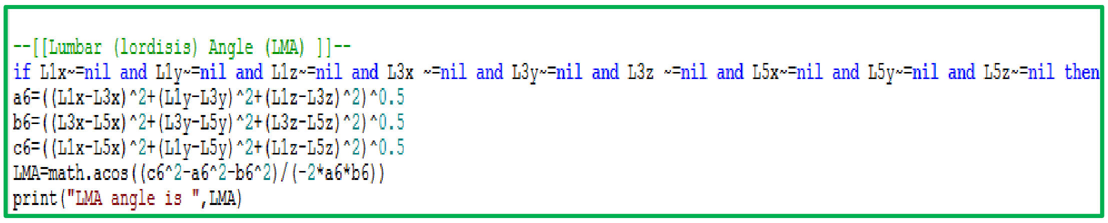
- Three dimensional Lordosis lumbar angle 3LMA
  This the actual angle of curvature of the spine at the Lumbar region and is calculated using three dimensional coordinates (x,y,z). In the markers dependent application LMA angle was calculated as follows:



```
--[[Lumbar (lordisis) Angle (LMA) ]]--
a6=((T12L1x-L3L4x)^2+(T12L1y-L3L4y)^2+(T12L1z-L3L4z)^2)^0.5
b6=((L3L4x-L5S1x)^2+(L3L4y-L5S1y)^2+(L3L4z-L5S1z)^2)^0.5
c6=((T12L1x-L5S1x)^2+(T12L1y-L5S1y)^2+(T12L1z-L5S1z)^2)^0.5
LMA=math.acos((c6^2-a6^2-b6^2)/(-2*a6*b6))
LMA=math.deg(LMA)
print("LMA angle is ",LMA)
outputs.set("LMA", LMA)
```

Figure 6. 41: LMA for marker dependent app

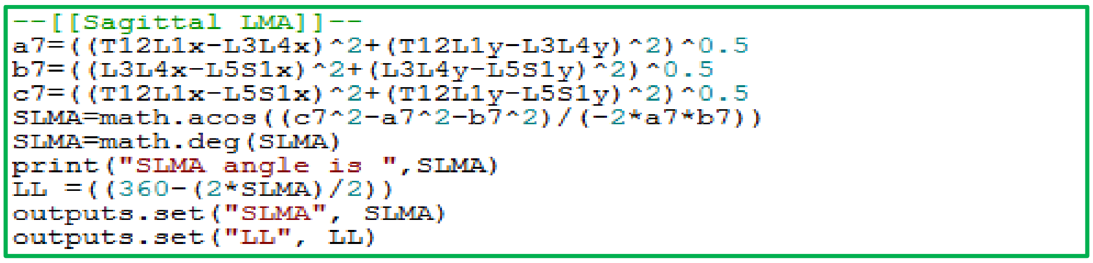
For the markers independent application, the following script was developed to calculate LMA angle.



```
--[[Lumbar (lordisis) Angle (LMA) ]]--
if L1x~=nil and L1y~=nil and L1z~=nil and L3x ~=nil and L3y~=nil and L3z ~=nil and L5x~=nil and L5y~=nil and L5z~=nil then
a6=((L1x-L3x)^2+(L1y-L3y)^2+(L1z-L3z)^2)^0.5
b6=((L3x-L5x)^2+(L3y-L5y)^2+(L3z-L5z)^2)^0.5
c6=((L1x-L5x)^2+(L1y-L5y)^2+(L1z-L5z)^2)^0.5
LMA=math.acos((c6^2-a6^2-b6^2)/(-2*a6*b6))
print("LMA angle is ",LMA)
```

Figure 6.42: LMA for marker independent app

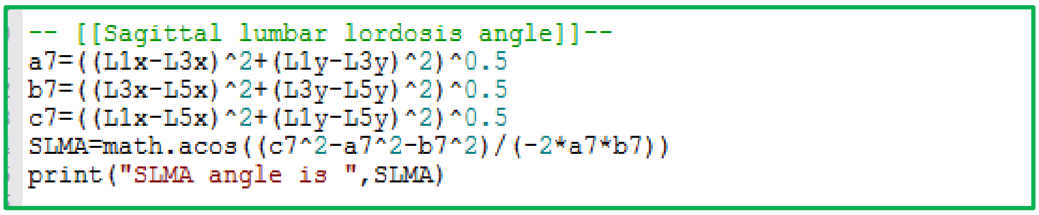
- Two dimensional sagittal plane Lordosis lumbar angle SLMA
  This the angle of curvature of the spine at the Lumbar region as viewed in sagittal plane and calculated using two dimensional coordinates (x,y). The main equations used to calculate SLMA angle in markers dependent application are shown in the following script.



```
--[[Sagittal LMA]]--
a7=((T12L1x-L3L4x)^2+(T12L1y-L3L4y)^2)^0.5
b7=((L3L4x-L5S1x)^2+(L3L4y-L5S1y)^2)^0.5
c7=((T12L1x-L5S1x)^2+(T12L1y-L5S1y)^2)^0.5
SLMA=math.acos((c7^2-a7^2-b7^2)/(-2*a7*b7))
SLMA=math.deg(SLMA)
print("SLMA angle is ",SLMA)
LL =((360-(2*SLMA)/2))
outputs.set("SLMA", SLMA)
outputs.set("LL", LL)
```

Figure 6. 43: SLMA for marker dependent app

For the markers independent application the following script highlights the governing equations for calculating SLMA angle.

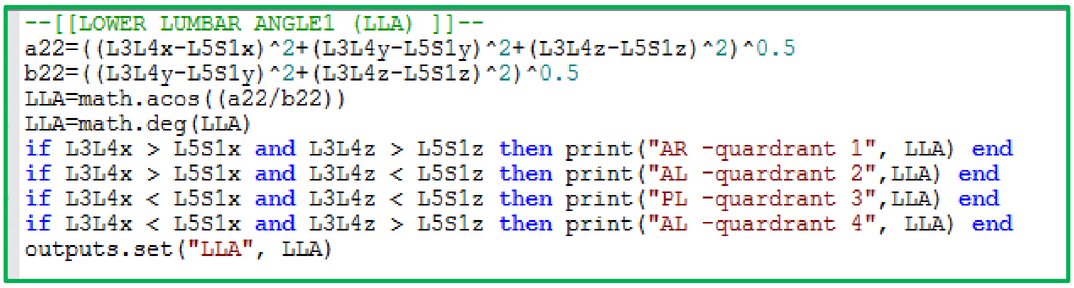


```
-- [[Sagittal lumbar lordosis angle]]--
a7=((L1x-L3x)^2+(L1y-L3y)^2)^0.5
b7=((L3x-L5x)^2+(L3y-L5y)^2)^0.5
c7=((L1x-L5x)^2+(L1y-L5y)^2)^0.5
SLMA=math.acos((c7^2-a7^2-b7^2)/(-2*a7*b7))
print("SLMA angle is ",SLMA)
```

Figure 6. 44: SLMA for marker independent app

Lower lumbar angles

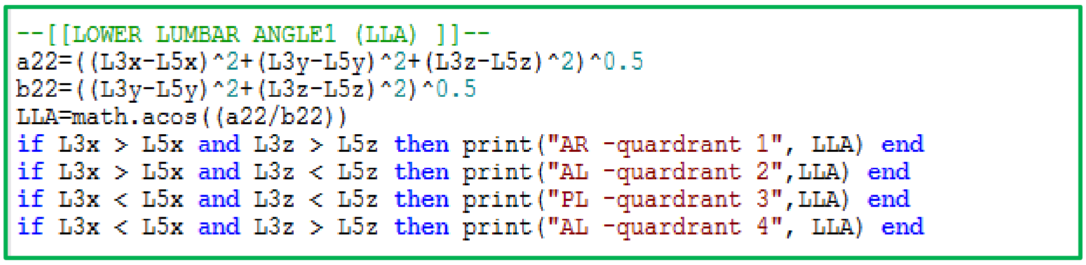
- Three dimension lower lumbar angle 3LLA
  This is the actual angle of curvature at the lower lumbar region calculated using three dimensional coordinates (x, y and z). In the script of the markers dependent application the LLA angle was calculated using coordinates of L3L4 And L5S1 markers while in the markers independent application the LLA angle was calculated by L3 and L5 Points. The following script indicates how to calculate LLA angle in the markers dependent application.



```
--[[LOWER LUMBAR ANGLE1 (LLA) ]]--
a22=((L3L4x-L5S1x)^2+(L3L4y-L5S1y)^2+(L3L4z-L5S1z)^2)^0.5
b22=((L3L4y-L5S1y)^2+(L3L4z-L5S1z)^2)^0.5
LLA=math.acos((a22/b22))
LLA=math.deg(LLA)
if L3L4x > L5S1x and L3L4z > L5S1z then print("AR -quardrant 1", LLA) end
if L3L4x > L5S1x and L3L4z < L5S1z then print("AL -quardrant 2",LLA) end
if L3L4x < L5S1x and L3L4z < L5S1z then print("PL -quardrant 3",LLA) end
if L3L4x < L5S1x and L3L4z > L5S1z then print("AL -quardrant 4", LLA) end
outputs.set("LLA", LLA)
```

Figure 6. 45: LLA for marker dependent app

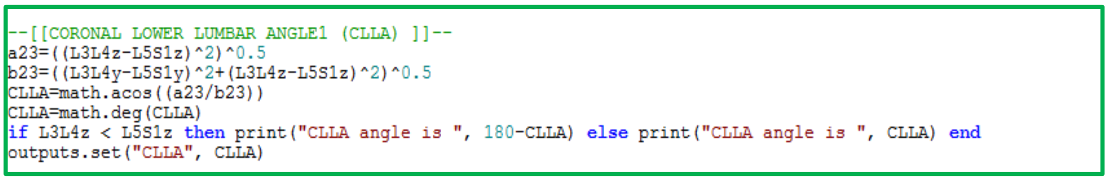
The following script indicates how to calculate LLA angle in the markers independent application.



```
--[[LOWER LUMBAR ANGLE1 (LLA) ]]--
a22=((L3x-L5x)^2+(L3y-L5y)^2+(L3z-L5z)^2)^0.5
b22=((L3y-L5y)^2+(L3z-L5z)^2)^0.5
LLA=math.acos((a22/b22))
if L3x > L5x and L3z > L5z then print("AR -quardrant 1", LLA) end
if L3x > L5x and L3z < L5z then print("AL -quardrant 2",LLA) end
if L3x < L5x and L3z < L5z then print("PL -quardrant 3",LLA) end
if L3x < L5x and L3z > L5z then print("AL -quardrant 4", LLA) end
```

Figure 6. 46: LLA for marker independent app

- Two dimension lower lumbar angle in coronal plane CLLA
  This is the angle of curvature at the lower lumbar region as viewed in coronal plane and calculated using two dimensional coordinates (y and z). The following extract script shows the equations guiding calculation of CLLA angle in the markers dependent application.



```
--[[CORONAL LOWER LUMBAR ANGLE1 (CLLA) ]]--
a23=((L3L4z-L5S1z)^2)^0.5
b23=((L3L4y-L5S1y)^2+(L3L4z-L5S1z)^2)^0.5
CLLA=math.acos((a23/b23))
CLLA=math.deg(CLLA)
if L3L4z < L5S1z then print("CLLA angle is ", 180-CLLA) else print("CLLA angle is ", CLLA) end
outputs.set("CLLA", CLLA)
```

Figure 6. 47: CLLA for marker dependent app

The following script indicates how to calculate CLLA angle in the markers independent application.

```
--[[CORONAL LOWER LUMBAR ANGLE1 (CLLA) ]]--
a23=((L3z-L5z)^2)^0.5
b23=((L3y-L5y)^2+(L3z-L5z)^2)^0.5
CLLA=math.acos((a23/b23))
if L3z < L5z then print("CLLA angle is ", 180-CLLA) else print("CLLA angle is ", CLLA) end
```

Figure 6. 48: CLLA for marker independent app

- Two dimension lower lumbar angle in sagittal plane SLLA
  This is the angle of curvature at the lower lumbar region as viewed in sagittal plane and calculated using two dimensional coordinates (x,y). The script below shows the algorithm of calculating of SLLA angle in the markers dependent application.

```
--[[SAGITTAL LOWER LUMBAR ANGLE1 (SLLA) ]]--
a24=((L3L4x-L5S1x)^2)^0.5
b24=((L3L4x-L5S1x)^2+(L3L4y-L5S1y)^2)^0.5
SLLA=math.acos((a24/b24))
SLLA=math.deg(SLLA)
if L3L4x < L5S1x then print("CULA angle is ", 180-CULA) else print("CULA angle is ", SLLA) end
outputs.set("SLLA", SLLA)
```

Figure 6.49: SLLA for marker dependent app

The following script is for SLLA angle calculation in the markers independent application.

```
--[[SAGITTAL LOWER LUMBAR ANGLE1 (SLLA) ]]--
a24=((L3x-L5x)^2)^0.5
b24=((L3x-L5x)^2+(L3y-L5y)^2)^0.5
SLLA=math.acos((a24/b24))
if L3x < L5x then print("CULA angle is ", 180-CULA) else print("CULA angle is ", SLLA) end
```

Figure 6.50: CULA for marker independent app

Parameters' calculations on sacral region

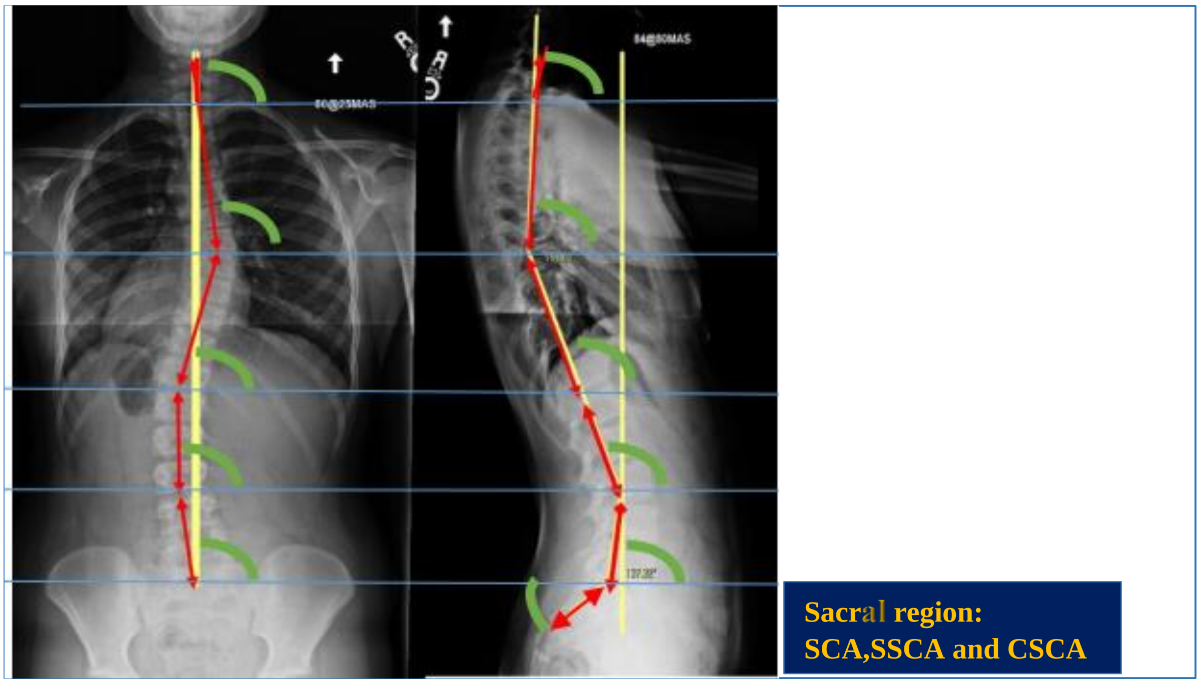
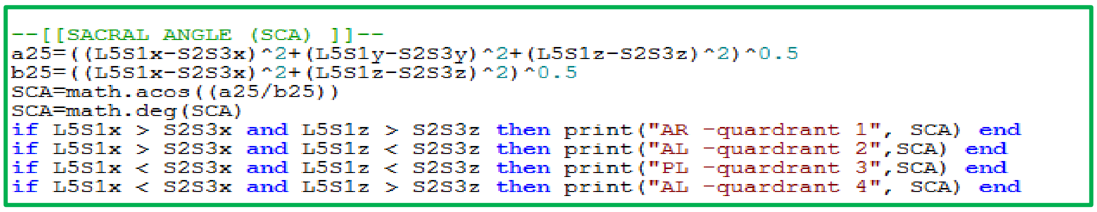


Figure 6. 51: X-ray image with a highlight of sacral region

Sacral angles (SCA)

- Three dimension sacral angle 3SCA

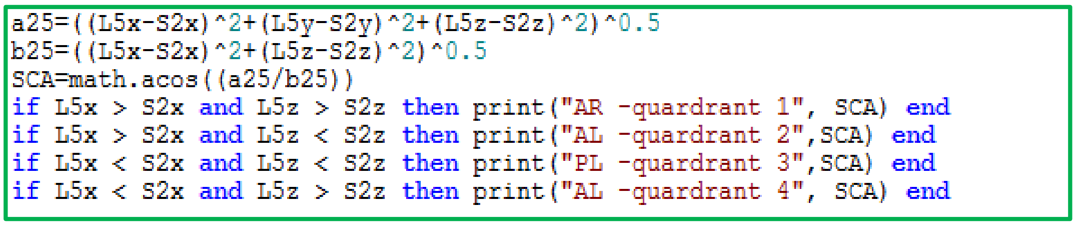
Sacral angle (SCA) is the actual angle of the spine at the sacral region calculated using three dimensional coordinates (x, y z). For the markers dependent application the angle is calculated using coordinates of L5S1 and S2S3 markers. The following script extracted from markers dependent script shows how SCA is calculated.



```
--[[SACRAL ANGLE (SCA) ]]--
a25=((L5S1x-S2S3x)^2+(L5S1y-S2S3y)^2+(L5S1z-S2S3z)^2)^0.5
b25=((L5S1x-S2S3x)^2+(L5S1z-S2S3z)^2)^0.5
SCA=math.acos((a25/b25))
SCA=math.deg(SCA)
if L5S1x > S2S3x and L5S1z > S2S3z then print("AR -quardrant 1", SCA) end
if L5S1x > S2S3x and L5S1z < S2S3z then print("AL -quardrant 2",SCA) end
if L5S1x < S2S3x and L5S1z < S2S3z then print("PL -quardrant 3",SCA) end
if L5S1x < S2S3x and L5S1z > S2S3z then print("AL -quardrant 4", SCA) end
```

Figure 6.52: SCA for marker dependent app

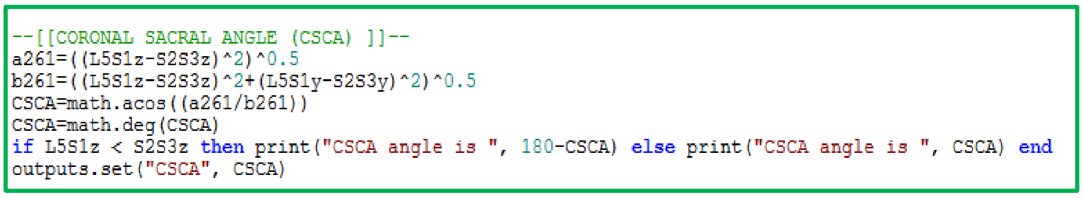
However in the markers independent application SCA is calculated as follows:



```
a25=((L5x-S2x)^2+(L5y-S2y)^2+(L5z-S2z)^2)^0.5
b25=((L5x-S2x)^2+(L5z-S2z)^2)^0.5
SCA=math.acos((a25/b25))
if L5x > S2x and L5z > S2z then print("AR -quardrant 1", SCA) end
if L5x > S2x and L5z < S2z then print("AL -quardrant 2",SCA) end
if L5x < S2x and L5z < S2z then print("PL -quardrant 3",SCA) end
if L5x < S2x and L5z > S2z then print("AL -quardrant 4", SCA) end
```

Figure 6.53: SCA for marker independent app

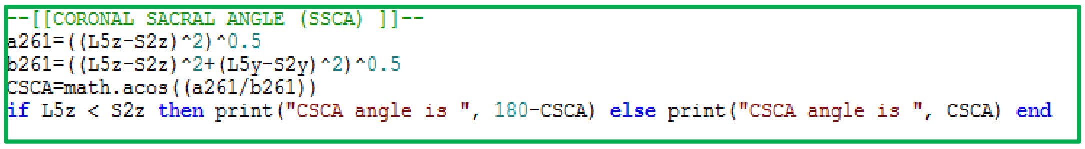
- Two dimension sacral angle in coronal plane CSCA
  Coronal sacral angle (SCA) is the angle of the spine at the sacral region as viewed in coronal plane and calculated using two dimensional coordinates (y, z). For the markers dependent application CSCA angle is calculated using coordinates of L5S1 and S2S3 markers. The following script extracted from markers dependent script shows how CSCA is calculated.



```
--[[CORONAL SACRAL ANGLE (CSCA) ]]--
a261=((L5S1z-S2S3z)^2)^0.5
b261=((L5S1z-S2S3z)^2+(L5S1y-S2S3y)^2)^0.5
CSCA=math.acos((a261/b261))
CSCA=math.deg(CSCA)
if L5S1z < S2S3z then print("CSCA angle is ", 180-CSCA) else print("CSCA angle is ", CSCA) end
outputs.set("CSCA", CSCA)
```

Figure 6. 54: CSCA for marker dependent app

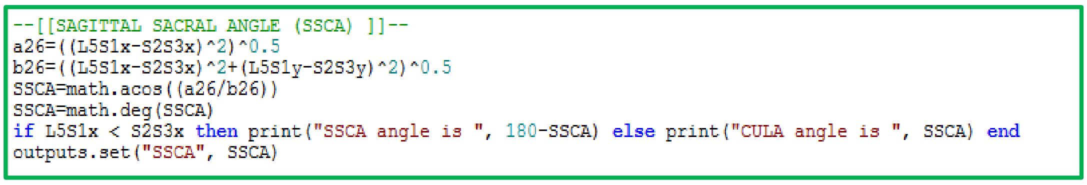
The CSCA angle in the markers independent application was calculated as follows:



```
--[[CORONAL SACRAL ANGLE (SSCA) ]]--
a261=((L5z-S2z)^2)^0.5
b261=((L5z-S2z)^2+(L5y-S2y)^2)^0.5
CSCA=math.acos((a261/b261))
if L5z < S2z then print("CSCA angle is ", 180-CSCA) else print("CSCA angle is ", CSCA) end
```

Figure 6.55: CSCA for marker independent app

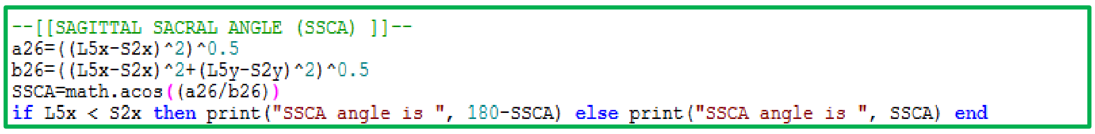
- Two dimension sacral angle in sagittal plane SSCA
  Sagittal sacral angle (SSCA) is the angle of the spine at the sacral region as viewed in sagittal plane and calculated using two dimensional coordinates (x,y). L5S1 and S2S3 Markers' coordinates were used for calculating SSCA in a markers dependent application. The following script shows the procedure of calculating SSCA angle in markers dependent script.



```
--[[SAGITTAL SACRAL ANGLE (SSCA) ]]--
a26=((L5S1x-S2S3x)^2)^0.5
b26=((L5S1x-S2S3x)^2+(L5S1y-S2S3y)^2)^0.5
SSCA=math.acos((a26/b26))
SSCA=math.deg(SSCA)
if L5S1x < S2S3x then print("SSCA angle is ", 180-SSCA) else print("CULA angle is ", SSCA) end
outputs.set("SSCA", SSCA)
```

Figure 6. 56: SSCA for marker dependent app

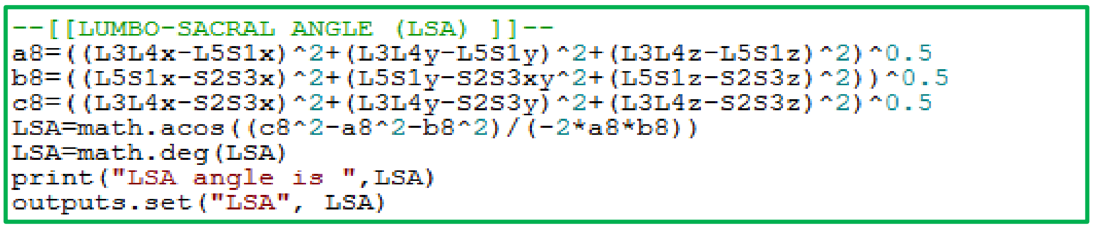
The SSCA angle in the markers independent application was calculated as follows:



```
--[[SAGITTAL SACRAL ANGLE (SSCA) ]]--
a26=((L5x-S2x)^2)^0.5
b26=((L5x-S2x)^2+(L5y-S2y)^2)^0.5
SSCA=math.acos((a26/b26))
if L5x < S2x then print("SSCA angle is ", 180-SSCA) else print("SSCA angle is ", SSCA) end
```

Figure 6. 57: SSCA for marker independent app

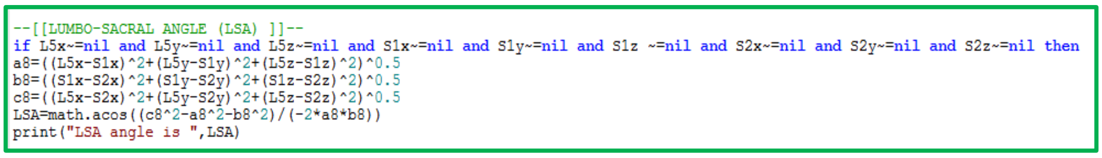
- Three dimension Lumbo-sacral angle 3LSA
  This is the actual angle of curvature at the lumbo-sacral region of the spine calculated using three dimension coordinates of markers and points in the markers dependent application and markers independent application respectively.
  The following scrpt indicates how to calculate LSA angle in the markers dependent application.



```
--[[LUMBO-SACRAL ANGLE (LSA) ]]--
a8=((L3L4x-L5S1x)^2+(L3L4y-L5S1y)^2+(L3L4z-L5S1z)^2)^0.5
b8=((L5S1x-S2S3x)^2+(L5S1y-S2S3xy^2+(L5S1z-S2S3z)^2))^0.5
c8=((L3L4x-S2S3x)^2+(L3L4y-S2S3y)^2+(L3L4z-S2S3z)^2)^0.5
LSA=math.acos((c8^2-a8^2-b8^2)/(-2*a8*b8))
LSA=math.deg(LSA)
print("LSA angle is ",LSA)
outputs.set("LSA", LSA)
```

Figure 6. 58: LSA for marker dependent app

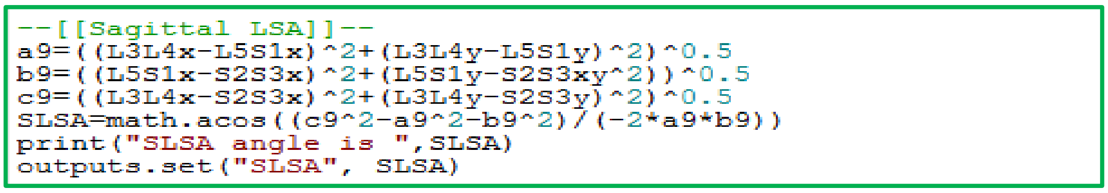
LSA angle in markers independent application was calculated using the following algorithm.



```
--[[LUMBO-SACRAL ANGLE (LSA) ]]--
if L5x~=nil and L5y~=nil and L5z~=nil and S1x~=nil and S1y~=nil and S1z ~=nil and S2x~=nil and S2y~=nil and S2z~=nil then
a8=((L5x-S1x)^2+(L5y-S1y)^2+(L5z-S1z)^2)^0.5
b8=((S1x-S2x)^2+(S1y-S2y)^2+(S1z-S2z)^2)^0.5
c8=((L5x-S2x)^2+(L5y-S2y)^2+(L5z-S2z)^2)^0.5
LSA=math.acos((c8^2-a8^2-b8^2)/(-2*a8*b8))
print("LSA angle is ",LSA)
```

Figure 6. 59: LSA for marker independent app

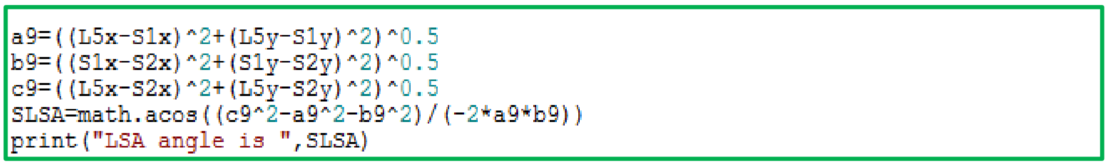
- Two dimension Lumbo-sacral angle in sagittal plane SLSA
  This is the angle of curvature at the lumbo-sacral region of the spine as viewed in sagittal plane and calculated using two dimensional coordinates (y, z) of markers and points in the markers dependent application and markers independent application respectively.
  The following script indicates how to calculate SLSA angle in the markers dependent application.



```
--[[Sagittal LSA]]--
a9=((L3L4x-L5S1x)^2+(L3L4y-L5S1y)^2)^0.5
b9=((L5S1x-S2S3x)^2+(L5S1y-S2S3xy^2))^0.5
c9=((L3L4x-S2S3x)^2+(L3L4y-S2S3y)^2)^0.5
SLSA=math.acos((c9^2-a9^2-b9^2)/(-2*a9*b9))
print("SLSA angle is ",SLSA)
outputs.set("SLSA", SLSA)
```

Figure 6.60: SLSA for marker dependent app

SLSA angle in markers independent application was calculated using the following algorithm.

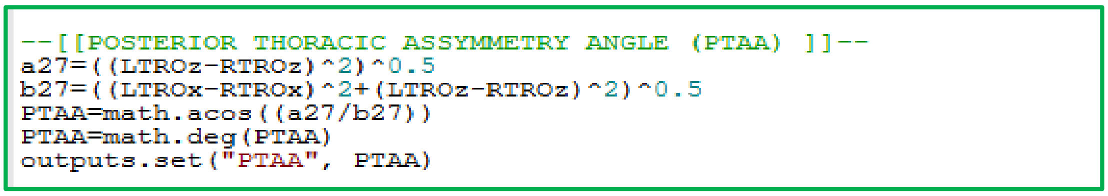


```
a9=((L5x-S1x)^2+(L5y-S1y)^2)^0.5
b9=((S1x-S2x)^2+(S1y-S2y)^2)^0.5
c9=((L5x-S2x)^2+(L5y-S2y)^2)^0.5
SLSA=math.acos((c9^2-a9^2-b9^2)/(-2*a9*b9))
print("LSA angle is ",SLSA)
```

Figure CC 61: SLSA for marker independent app

Other calculations

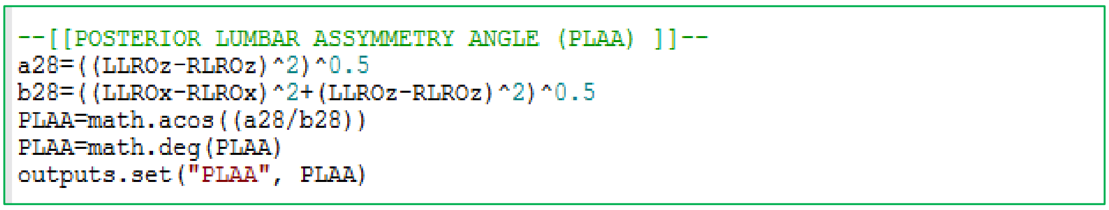
- Posterior thoracic asymmetry angle (angle of trunk rotation) (PTAA)
  This is the angle that show the rotation of the thoracic region of the trunk. The algorithm of calculating this angle is shown below.



```
--[[POSTERIOR THORACIC ASSYMMETRY ANGLE (PTAA) ]]--
a27=((LTROz-RTROz)^2)^0.5
b27=((LTROx-RTROx)^2+(LTROz-RTROz)^2)^0.5
PTAA=math.acos((a27/b27))
PTAA=math.deg(PTAA)
outputs.set("PTAA", PTAA)
```

Figure 6.62: PTAA for marker dependent app

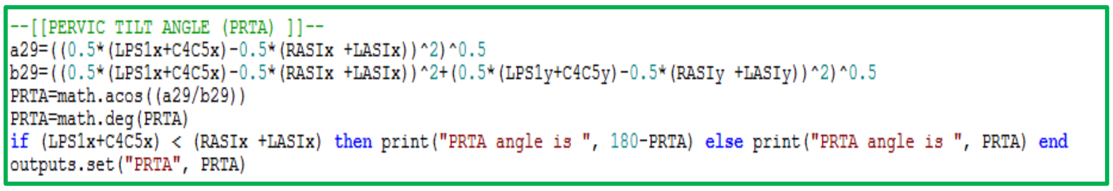
- Posterior lumbar asymmetry angle (PLAA)
  This is the angle that show the rotation of the lumbar region of the trunk. The algorithm of calculating this angle is shown below.



```
--[[POSTERIOR LUMBAR ASSYMMETRY ANGLE (PLAA) ]]--
a28=((LLROz-RLROz)^2)^0.5
b28=((LLROx-RLROx)^2+(LLROz-RLROz)^2)^0.5
PLAA=math.acos((a28/b28))
PLAA=math.deg(PLAA)
outputs.set("PLAA", PLAA)
```

Figure 6. 63: PLAA for marker dependent app

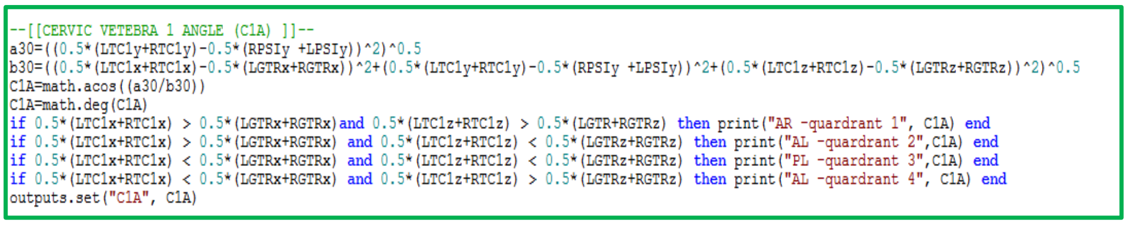
- Pelvic Tilt angle in sagittal plane (SPTA)
  This is the angle that show the tilted the pelvis is using posterior and anterior base points of the pelvis. The algorithm of calculating this angle is shown below.



```
--[[PERVIC TILT ANGLE (PRTA) ]]--
a29=((0.5*(LPS1x+C4C5x)-0.5*(RASIx +LASIx))^2)^0.5
b29=((0.5*(LPS1x+C4C5x)-0.5*(RASIx +LASIx))^2+(0.5*(LPS1y+C4C5y)-0.5*(RASIy +LASIy))^2)^0.5
PRTA=math.acos((a29/b29))
PRTA=math.deg(PRTA)
if (LPS1x+C4C5x) < (RASIx +LASIx) then print("PRTA angle is ", 180-PRTA) else print("PRTA angle is ", PRTA) end
outputs.set("PRTA", PRTA)
```

Figure 6. 64: PRTA for marker dependent app

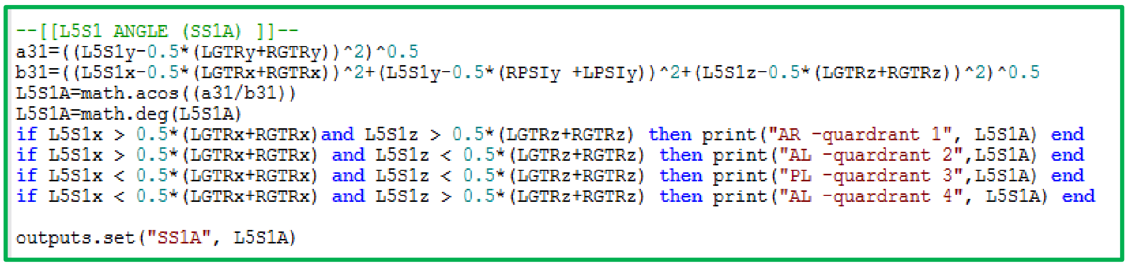
- Angle from centre of body at C1, C31A
  This is the actual angle showing how far the C1 vertebra is from the centre point of the body of the subject. This angle was calculated using three dimensional coordinates (x,y and z) as shown in the script below extracted from code of the makers dependent application.



```
--[[CERVIC VETEBRA 1 ANGLE (C1A) ]]--
a30=((0.5*(LTC1y+RTC1y)-0.5*(RPSIy +LPSIy))^2)^0.5
b30=((0.5*(LTC1x+RTC1x)-0.5*(LGTRx+RGTRx))^2+(0.5*(LTC1y+RTC1y)-0.5*(RPSIy +LPSIy))^2+(0.5*(LTC1z+RTC1z)-0.5*(LGTRz+RGTRz))^2)^0.5
C1A=math.acos((a30/b30))
C1A=math.deg(C1A)
if 0.5*(LTC1x+RTC1x) > 0.5*(LGTRx+RGTRx)and 0.5*(LTC1z+RTC1z) > 0.5*(LGTR+RGTRz) then print("AR -quardrant 1", C1A) end
if 0.5*(LTC1x+RTC1x) > 0.5*(LGTRx+RGTRx) and 0.5*(LTC1z+RTC1z) < 0.5*(LGTRz+RGTRz) then print("AL -quardrant 2",C1A) end
if 0.5*(LTC1x+RTC1x) < 0.5*(LGTRx+RGTRx) and 0.5*(LTC1z+RTC1z) < 0.5*(LGTRz+RGTRz) then print("PL -quardrant 3",C1A) end
if 0.5*(LTC1x+RTC1x) < 0.5*(LGTRx+RGTRx) and 0.5*(LTC1z+RTC1z) > 0.5*(LGTRz+RGTRz) then print("AL -quardrant 4", C1A) end
outputs.set("C1A", C1A)
```

Figure 6. 65: C1A for marker dependent app

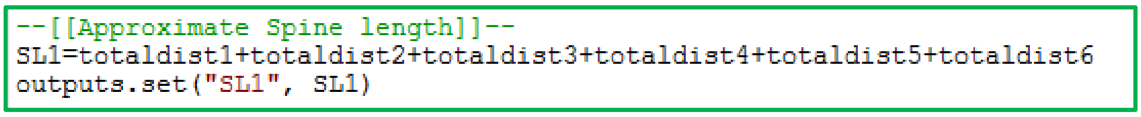
- Sagittal angle at S1, SS1A
  This is angle showing how far the C1 vertebra is from the centre point of the body of the subject as viewed in sagittal plane. This angle was calculated using two dimensional coordinates (x, y) as shown below in the script extracted from code of the makers dependent application.



```
--[[L5S1 ANGLE (SS1A) ]]--
a31=((L5S1y-0.5*(LGTRy+RGTRy))^2)^0.5
b31=((L5S1x-0.5*(LGTRx+RGTRx))^2+(L5S1y-0.5*(RPSIy +LPSIy))^2+(L5S1z-0.5*(LGTRz+RGTRz))^2)^0.5
L5S1A=math.acos((a31/b31))
L5S1A=math.deg(L5S1A)
if L5S1x > 0.5*(LGTRx+RGTRx)and L5S1z > 0.5*(LGTRz+RGTRz) then print("AR -quardrant 1", L5S1A) end
if L5S1x > 0.5*(LGTRx+RGTRx) and L5S1z < 0.5*(LGTRz+RGTRz) then print("AL -quardrant 2",L5S1A) end
if L5S1x < 0.5*(LGTRx+RGTRx) and L5S1z < 0.5*(LGTRz+RGTRz) then print("PL -quardrant 3",L5S1A) end
if L5S1x < 0.5*(LGTRx+RGTRx) and L5S1z > 0.5*(LGTRz+RGTRz) then print("AL -quardrant 4", L5S1A) end

outputs.set("SS1A", L5S1A)
```

Figure 6. 66: SS1A for marker dependent app

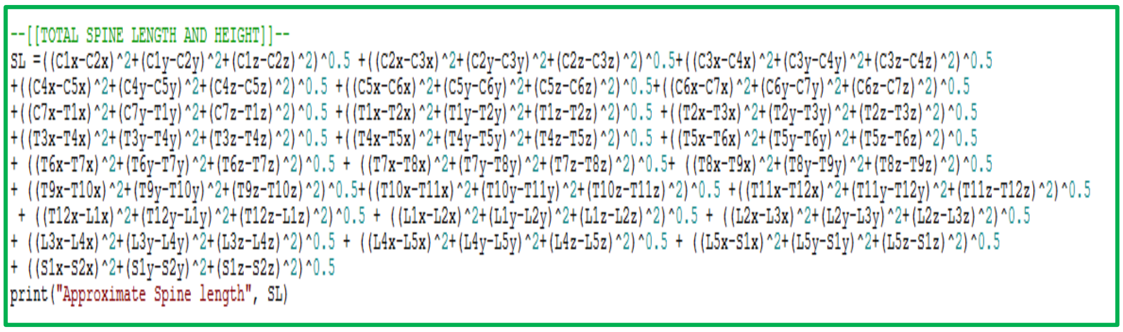
- Length of the spine, SL
  This parameter shows approximate length of the spine. It is a perimeter distance from C1 to coccyx region. The spine length parameter in markers dependent application was calculated as follows:



```
--[[Approximate Spine length]]--
SL1=totaldist1+totaldist2+totaldist3+totaldist4+totaldist5+totaldist6
outputs.set("SL1", SL1)
```

Figure 6. 67: Spine length for marker dependent app

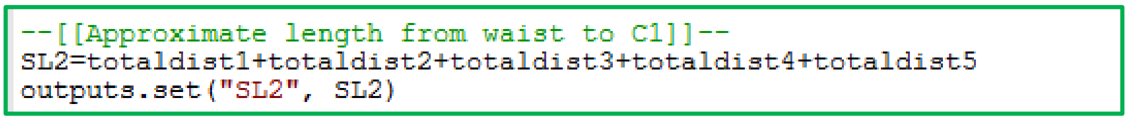
In markers independent application the spine length was calculated as follows.



```
--[[TOTAL SPINE LENGTH AND HEIGHT]]--
SL =((C1x-C2x)^2+(C1y-C2y)^2+(C1z-C2z)^2)^0.5 +((C2x-C3x)^2+(C2y-C3y)^2+(C2z-C3z)^2)^0.5+((C3x-C4x)^2+(C3y-C4y)^2+(C3z-C4z)^2)^0.5
+((C4x-C5x)^2+(C4y-C5y)^2+(C4z-C5z)^2)^0.5 +((C5x-C6x)^2+(C5y-C6y)^2+(C5z-C6z)^2)^0.5+((C6x-C7x)^2+(C6y-C7y)^2+(C6z-C7z)^2)^0.5
+((C7x-T1x)^2+(C7y-T1y)^2+(C7z-T1z)^2)^0.5 +((T1x-T2x)^2+(T1y-T2y)^2+(T1z-T2z)^2)^0.5 +((T2x-T3x)^2+(T2y-T3y)^2+(T2z-T3z)^2)^0.5
+((T3x-T4x)^2+(T3y-T4y)^2+(T3z-T4z)^2)^0.5 +((T4x-T5x)^2+(T4y-T5y)^2+(T4z-T5z)^2)^0.5 +((T5x-T6x)^2+(T5y-T6y)^2+(T5z-T6z)^2)^0.5
+ ((T6x-T7x)^2+(T6y-T7y)^2+(T6z-T7z)^2)^0.5 + ((T7x-T8x)^2+(T7y-T8y)^2+(T7z-T8z)^2)^0.5+ ((T8x-T9x)^2+(T8y-T9y)^2+(T8z-T9z)^2)^0.5
+ ((T9x-T10x)^2+(T9y-T10y)^2+(T9z-T10z)^2)^0.5+((T10x-T11x)^2+(T10y-T11y)^2+(T10z-T11z)^2)^0.5 +((T11x-T12x)^2+(T11y-T12y)^2+(T11z-T12z)^2)^0.5
 + ((T12x-L1x)^2+(T12y-L1y)^2+(T12z-L1z)^2)^0.5 + ((L1x-L2x)^2+(L1y-L2y)^2+(L1z-L2z)^2)^0.5 + ((L2x-L3x)^2+(L2y-L3y)^2+(L2z-L3z)^2)^0.5
+ ((L3x-L4x)^2+(L3y-L4y)^2+(L3z-L4z)^2)^0.5 + ((L4x-L5x)^2+(L4y-L5y)^2+(L4z-L5z)^2)^0.5 + ((L5x-S1x)^2+(L5y-S1y)^2+(L5z-S1z)^2)^0.5
+ ((S1x-S2x)^2+(S1y-S2y)^2+(S1z-S2z)^2)^0.5
print("Approximate Spine length", SL)
```

Figure 6.68: Spine length for marker independent app

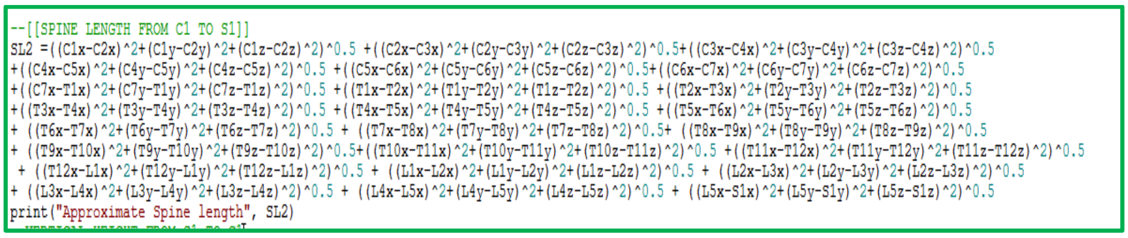
- Length from waist to C1
  This parameter depicts length of spine from waist or pelvis to the C1 vertebrae. The following script from markers dependent application estimates this length.



```
--[[Approximate length from waist to C1]]--
SL2=totaldist1+totaldist2+totaldist3+totaldist4+totaldist5
outputs.set("SL2", SL2)
```

Figure 6.69: spine length from waist for marker dependent app

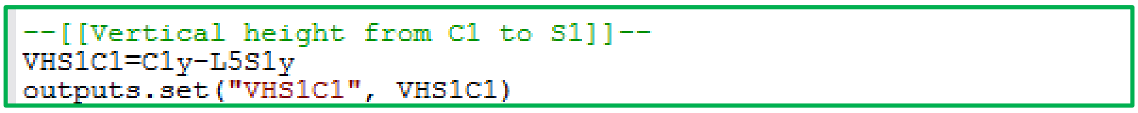
Length from waist to C1 in the markers independent application was estimated using the following equation:



```
--[[SPINE LENGTH FROM C1 TO S1]]
SL2 =((C1x-C2x)^2+(C1y-C2y)^2+(C1z-C2z)^2)^0.5 +((C2x-C3x)^2+(C2y-C3y)^2+(C2z-C3z)^2)^0.5+((C3x-C4x)^2+(C3y-C4y)^2+(C3z-C4z)^2)^0.5
+((C4x-C5x)^2+(C4y-C5y)^2+(C4z-C5z)^2)^0.5 +((C5x-C6x)^2+(C5y-C6y)^2+(C5z-C6z)^2)^0.5+((C6x-C7x)^2+(C6y-C7y)^2+(C6z-C7z)^2)^0.5
+((C7x-T1x)^2+(C7y-T1y)^2+(C7z-T1z)^2)^0.5 +((T1x-T2x)^2+(T1y-T2y)^2+(T1z-T2z)^2)^0.5 +((T2x-T3x)^2+(T2y-T3y)^2+(T2z-T3z)^2)^0.5
+((T3x-T4x)^2+(T3y-T4y)^2+(T3z-T4z)^2)^0.5 +((T4x-T5x)^2+(T4y-T5y)^2+(T4z-T5z)^2)^0.5 +((T5x-T6x)^2+(T5y-T6y)^2+(T5z-T6z)^2)^0.5
+ ((T6x-T7x)^2+(T6y-T7y)^2+(T6z-T7z)^2)^0.5 + ((T7x-T8x)^2+(T7y-T8y)^2+(T7z-T8z)^2)^0.5+ ((T8x-T9x)^2+(T8y-T9y)^2+(T8z-T9z)^2)^0.5
+ ((T9x-T10x)^2+(T9y-T10y)^2+(T9z-T10z)^2)^0.5+((T10x-T11x)^2+(T10y-T11y)^2+(T10z-T11z)^2)^0.5 +((T11x-T12x)^2+(T11y-T12y)^2+(T11z-T12z)^2)^0.5
 + ((T12x-L1x)^2+(T12y-L1y)^2+(T12z-L1z)^2)^0.5 + ((L1x-L2x)^2+(L1y-L2y)^2+(L1z-L2z)^2)^0.5 + ((L2x-L3x)^2+(L2y-L3y)^2+(L2z-L3z)^2)^0.5
+ ((L3x-L4x)^2+(L3y-L4y)^2+(L3z-L4z)^2)^0.5 + ((L4x-L5x)^2+(L4y-L5y)^2+(L4z-L5z)^2)^0.5 + ((L5x-S1x)^2+(L5y-S1y)^2+(L5z-S1z)^2)^0.5
print("Approximate Spine length", SL2)
```

Figure 6.70: spine length from waist for marker independent app

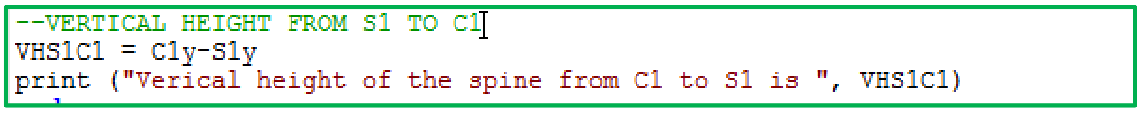
- Vertical Height of the spine from C1 to S1
  This is a variable that estimates the vertical height of the spine of the subject from C1 vertebra to S1 vertebra. The following Script highlight how to calculate the variable in markers dependent application.



```
--[[Vertical height from C1 to S1]]--
VHS1C1=C1y-L5S1y
outputs.set("VHS1C1", VHS1C1)
```

Figure 6.71: Vertical height form C1 to S1 for marker dependent app

In the markers independent application the vertical height was calculated using the following script.



```
--VERTICAL HEIGHT FROM S1 TO C1
VHS1C1 = C1y-S1y
print ("Verical height of the spine from C1 to S1 is ", VHS1C1)
```

Figure 6.72: Vertical height form C1 to S1 for marker independent app

Chapter 7: RESULTS

The two applications (marker dependent and marker independent applications) for visualising the spinal column were tested on two configurations of the spine. The first configuration was S-shaped in order to try to mimic shape of spine of patients with scoliosis while the second configuration was a normal geometry of the spine. The S-shaped configuration as shown in figure 7.1.1, was achieved by using a cable mounted at the back of the laboratory skeleton. On the other hand the normal configuration was achieved by tracking the shape of the spine laboratory as shown in figure

7.1 Markers independent application testing results

Performance

The performance was fantastic with a frame rate of 50Hz which is good for the static application. It was easy to use and could not crash despite being an application with huge script and many modules.

Subject's first geometry

The figure below represents the first S-shaped geometry that was visualised by both applications. The S-shaped curve is represented by the black cable in Figure 7.1.1. The labelling process started from C1 to C5 on the skeleton and then followed the locus of the black cable. The visualisations of spinal column from markers independent marker application are shown in figures 7. 1.2.

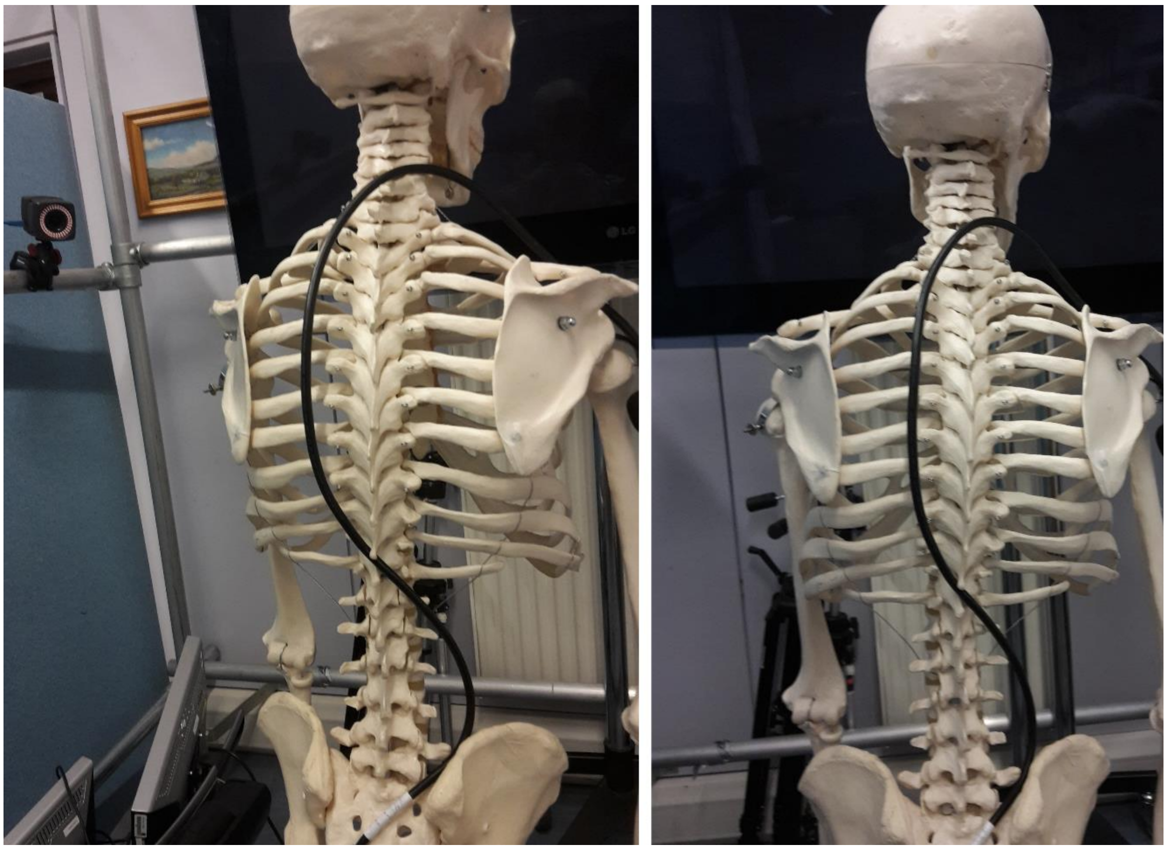


Figure 7.1.1: S-shaped cable at the back of the skeleton, mimicking geometry of patients with scoliosis

Visualisation for S-Shaped geometry

Below is the visualisation of S-shaped configuration by marker independent application.

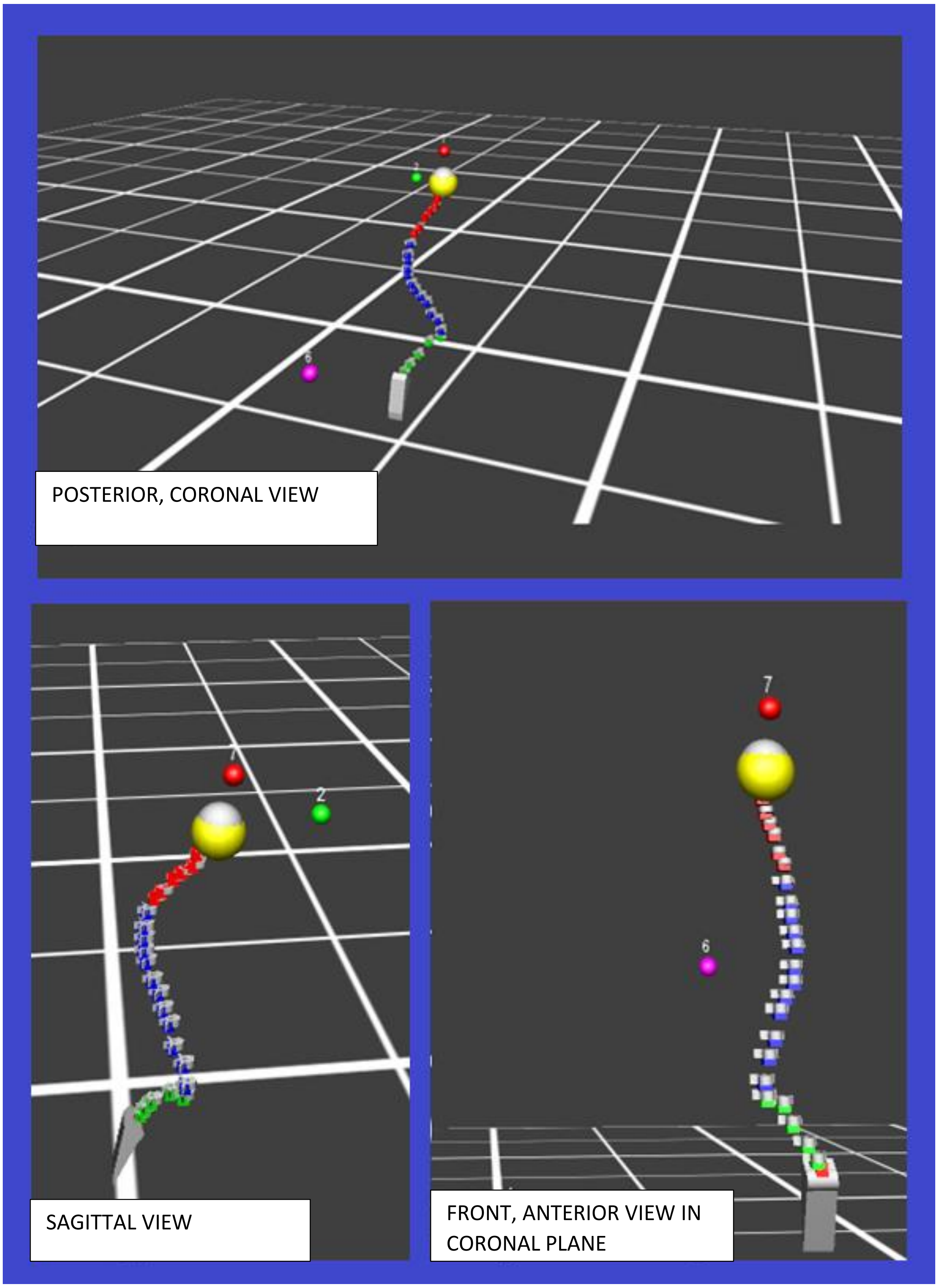


Figure 7.1.2: D-flow visualisation of s-shaped spine using marker independent application

Subject's Second geometry

The figure below shows the shape of the spine geometry of laboratory skeleton. This shape was used for the second test of visualising a normal shape of spine. Figure 7.1.3 below shows the geometry of the subject used to test the marker independent application. The same protocol of marker independent application for capturing points from C1 to coccyx was used.

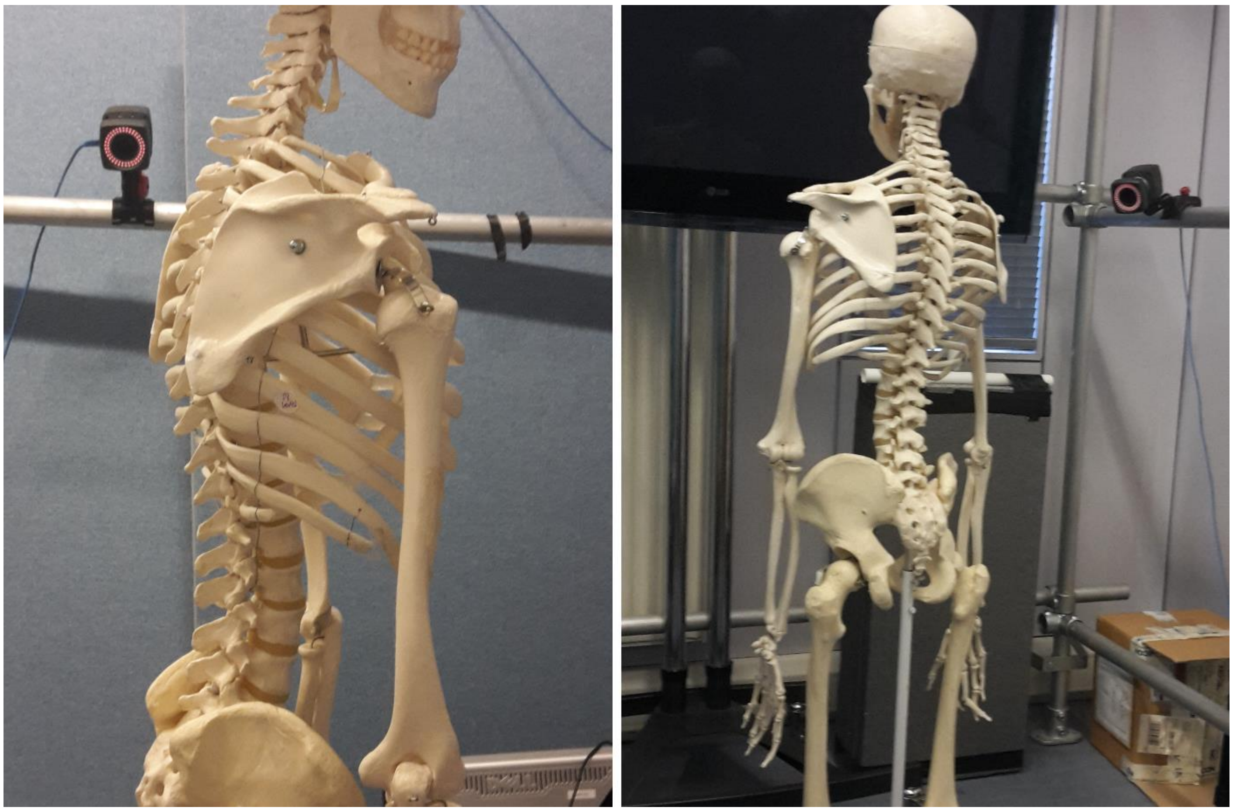


Figure 7.1.3: normal shape of the spine of a laboratory skeleton

Visualisation for normal geometry

The visualisation obtained after second testing of visualising normal geometry of lab skeleton is show in figure 7.1.4. The geometry of the spine in the visualisation model, as expected, was similar to that of the subject.

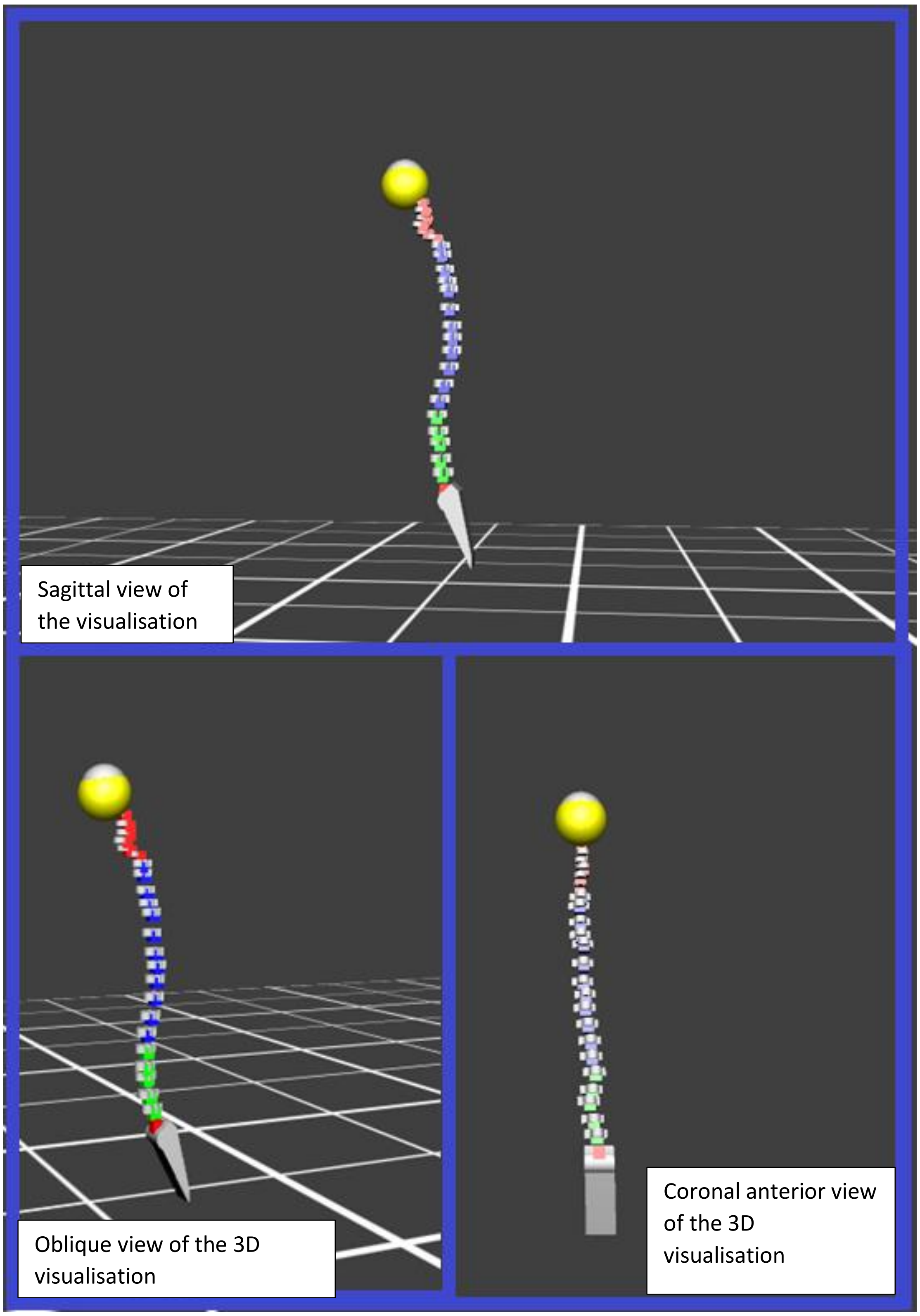

Sagittal view of the visualisation
Oblique view of the 3D visualisation
Coronal anterior view of the 3D visualisation

Figure 7.1. 4: Visualisation of the normal geometry of the spine using marker independent application

7.2 Markers dependent application testing results

This application was also tested by visualising two different configuration of markers attached to the skeleton. One configuration was for normal geometry of the spine while the other configuration was abnormal with a curvature in the lumbar region of the spine. The tests for this application was both static and dynamic. The dynamic test involved moving the subject in the capture volume.

Performance of the marker dependent application

The performance of the marker dependent application good as well with a frame rate of 30Hz which is good for both static and dynamic applications. It was easier to label the markers. For dynamic application the skeleton visualisation was able to move as the skeleton was being moved in the capture volume.

First Subject's geometry for the marker dependent application: abnormal geometry

The figure 7.2.1 below shows the abnormal geometry that was visualised by the marker dependent application. The red line indicates the locus of the spine. The visualisation from the application for this abnormal geometry is shown in figure 7.2.2.

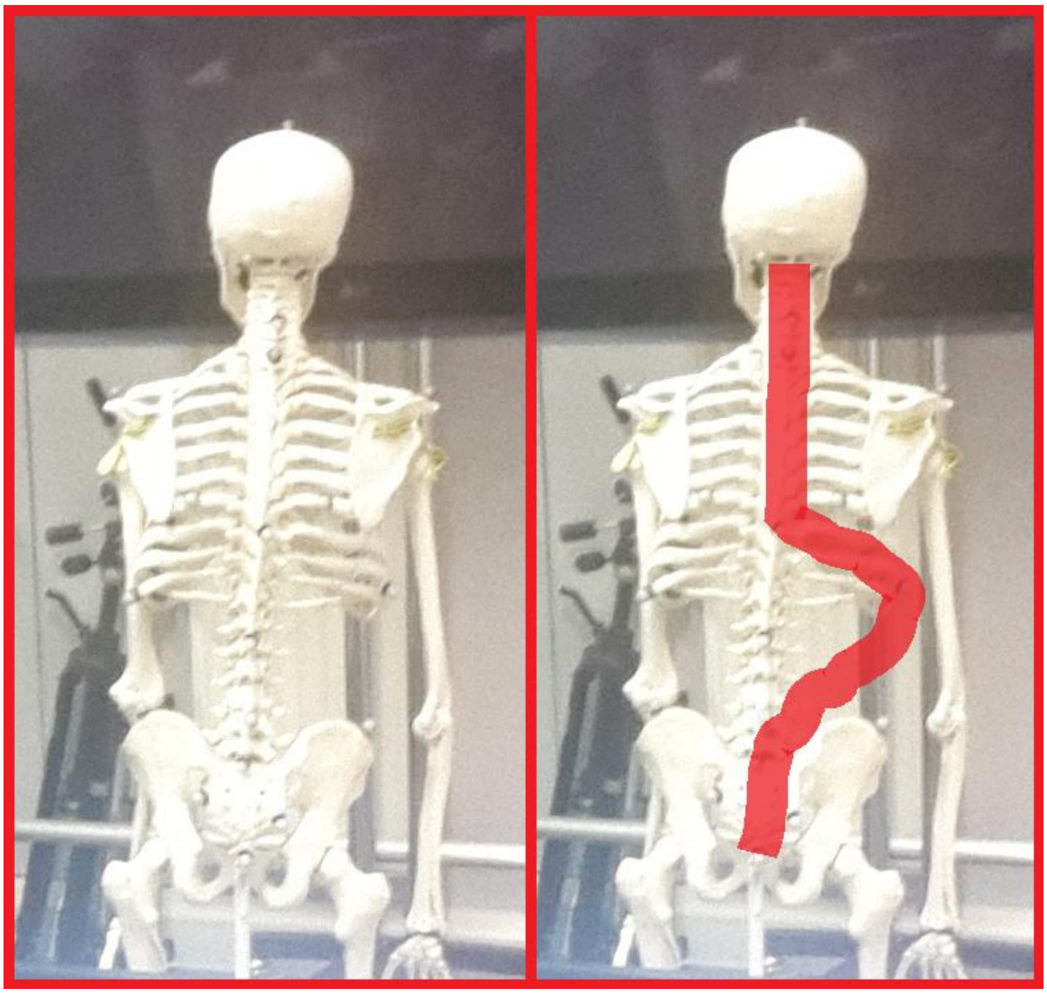


Figure 7.2.1: first configuration of markers. The red line shows the assumed locus of the spine

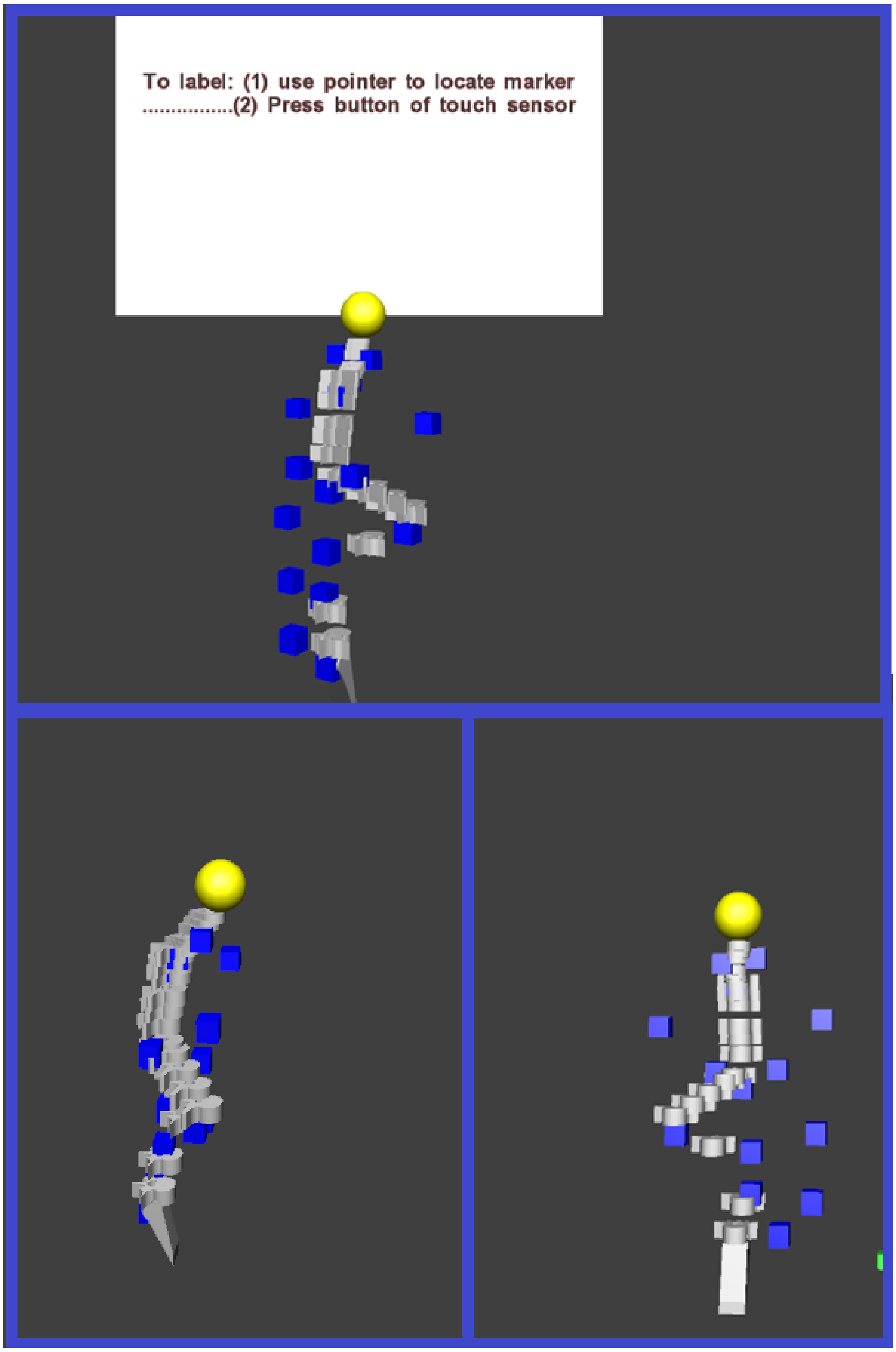

To label: (1) use pointer to locate marker
................(2) Press button of touch sensor

Figure 7.2.2: Visualisation of abnormal geometry by the markers dependent application

Second Subject’s geometry for the marker dependent application: Normal geometry

The figure 7.1.3 that shows a laboratory skeleton, shows the normal geometry that was used for the second test for marker dependent application. Figure 7.2.3 below shows the resultant visualisation after the second test of a normal spine of the lab skeleton.

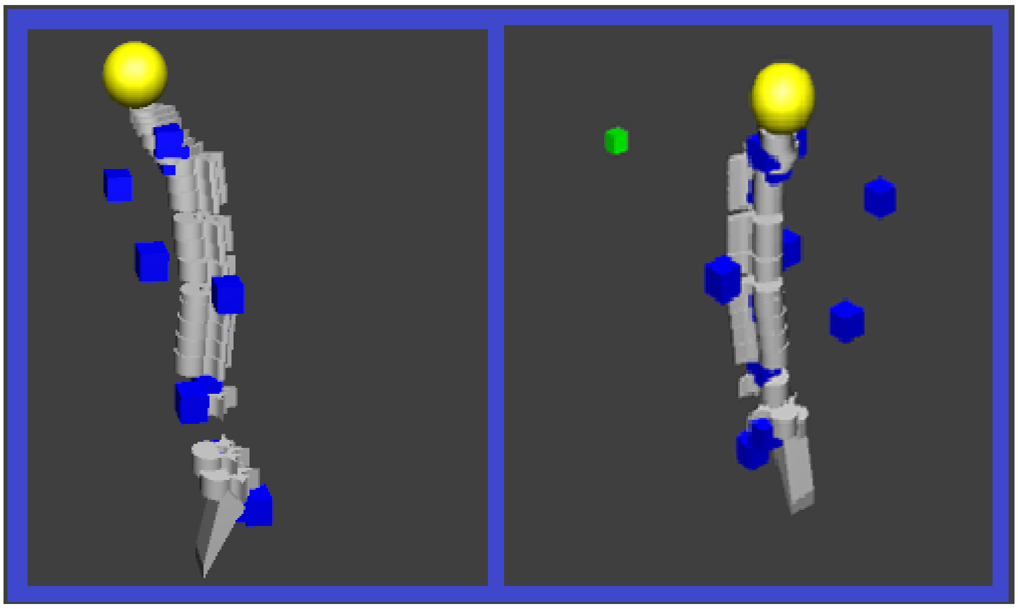


Figure 7.2.3: visualisation of normal spine of lab skeleton by marker dependent application

Numerical data from the testing

The following table R1 below shows an extract of some numerical values for coronal cervical angle (CCEA), upper thoracic angle (UTA) and sagittal cervical angle (SCEA) based on data obtained from the two applications. The data in table 7.2.1 is for the normal geometry of the skeleton subject.

Table 7.2.1: Example of some of numerical values of different the parameters. The data represents parameters for the normal geometry of the skeleton or second test for the two applications.

CCEA for marker dependent	CCEA for Marker Independent	UTA for marker Dependent	UTA for marker Independent	SCEA for marker dependent	SCEA for Marker Independent	SL2
89.320278	86.364648	261.997966	259.42347	119.892409	118.856702	0.557568
89.314739	86.364648	261.997177	259.42347	119.892409	118.856702	0.557568
89.304386	86.364648	261.99777	259.42347	119.892409	118.856702	0.557568
89.317174	86.364648	261.999392	259.42347	119.892409	118.856702	0.557568
89.303467	86.364648	261.996862	259.42347	119.892409	118.856702	0.557568
89.299952	86.364648	261.996456	259.42347	119.892409	118.856702	0.557568
89.312741	86.364648	261.994097	259.42347	119.892409	118.856702	0.557568
89.302907	86.364648	261.996836	259.42347	119.892409	118.856702	0.557568
89.293845	86.364648	261.998551	259.42347	119.892409	118.856702	0.557568
89.303981	86.364648	261.997516	259.42347	119.892409	118.856702	0.557568
89.30362	86.364648	262.000456	259.42347	119.892409	118.856702	0.557568
89.302319	86.364648	261.996506	259.42347	119.892409	118.856702	0.557568
89.301985	86.364648	261.997936	259.42347	119.892409	118.856702	0.557568
89.303279	86.364648	261.994873	259.42347	119.892409	118.856702	0.557568
89.29595	86.364648	261.998784	259.42347	119.892409	118.856702	0.557568
89.296752	86.364648	261.997049	259.42347	119.892409	118.856702	0.557568
89.288017	86.364648	261.996385	259.42347	119.892409	118.856702	0.557568

The descriptive statistics values for the different parameters were analysed using Minitab software. The table 7.2.2 below shows some of the descriptive statistics for coronal cervical angle (CCEA), upper thoracic angle (UTA) and sagittal cervical angle (SCEA) based on data obtained from the two applications.

Table 7.2. 2: Descriptive statistics at 95% confidence interval (CI for) some of measured parameters recorded within 6675 frames.

Variable	N	Mean	SE Mean	StDev	Minimum	Q1	Median
CCEA for marker dependent	6675	89.319°	0.000259	0.0211	89.271°	89.307	89.317°
CCEA for marker independent	6675	86.365°	0	0	86.365°	86.365	86.365°
UTA for marker dependent	6675	261.98°	0.000169	0.0138	261.96°	261.97	261.98°
UTA for marker independent	6675	259.42°	0	0	259.42°	259.42	259.42°
SCEA for marker dependent	6675	119.89°	0	0	119.89°	119.89	119.89°
SCEA for marker independent	6675	118.86°	0	0	118.86°	118.86	118.86°
Spine length from C1-S1	6675	0.557m	0	0	0.557m	0.557m	0.557m

Manual measurements

The following table R3 shows an extract manual measurements of some parameters.

Table 7.2. 3: Example of manual measurements of the selected variables

Variable name	value
Coronal cervical angle, CCEA	87°
Sagittal cervical angle, SCEA	117°
Upper thoracic angle, UTA	260°
Spine length from C1-S1	0.55m

Some of the selected angles were measured manually using goniometer. The figure below shows a location of SCEA angle and the lines being usually approximated as references to measure it.

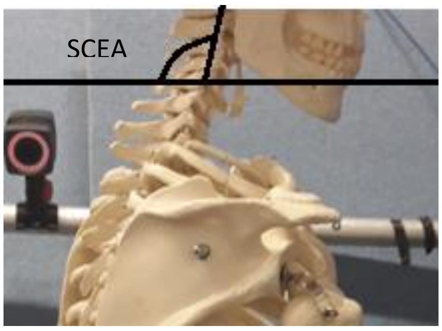


Figure 7.2.4: location of SCEA angle

On the other hand spine length was measured using a measuring tape. Thus the results were similar and they are meaningful.

Chapter 8: DISCUSSION

The development of the two applications for visualising spine geometry was a success as shown in the results section. The major criteria validating the success of the applications included: performance, similarity of visualisation geometry to shape of subject's spine, similarity of measured parameters by software to those measured manually, easy of labelling points or markers and minimum errors. Thus this section demystifies the success based on obtained results, advantages and limitations of the two applications. Moreover potential source of errors that should always be avoided when using the applications are also highlighted in this section.

Firstly as shown in results section, the performance of both applications was incredible with minimum frame rate frequency of 30 -50Hz. This frequency range is very suitable for both static and dynamic applications. Based on Nyquist theorem the minimum sampling frequency is supposed to be twice the frequency of signals. In the static application there is no movement while in dynamic applications for scoliosis patients interested in this project have low frequencies such that 30Hz is very suitable.

Secondly both applications were a success not only based on performance but also based on similarity of visualisations' geometry to the actual geometry of subject's spines. Both normal and abnormal geometries were used for to test each of the two applications. Both applications displayed geometry of normal spine similarly and the visualisations was also similar to the true geometry of the subject's spine. The application that does not require attaching markers on the body of the subject can only be used for static applications while the marker dependent application, which depends on attaching markers on the body of the subject, can be used for both static and dynamic applications.

Furthermore, the measured parameters were also correct and were validated by comparing them to manual measurements of the parameters using lab measurement tools such as goniometers and measuring tapes. As shown in table R2 of descriptive statistics for the selected variables, marker dependent application had a slightly larger mean for many variables than marker independent application. For instance the means of CCEA and SCEA angles from marker dependent application were 89.3° $\pm$0.021° and 119.8° $\pm$0 respectively. On the other hand the means of CCEA and SCEA angles from marker independent application were 86.6° $\pm$0 and 118.86° $\pm$0 respectively. Besides the manual measurements of CCEA and SCEA angles were 87°and 117° respectively. Thus the measurements were not very different. The cause of having larger values in marker dependent application is interpolation due to few markers on the spine. On the other hand the marker independent application had more points hence the values are more accurate.

Besides, the applications are very easy to use. The marker independent application has fewer labelling points, 26 to be specific, while marker dependent application has 40 points that must labelled correctly for successful visualisation and parameter calculation of spine's geometry. Thus from the tests it was quicker to complete visualising the geometry of subjects using the marker independent application than using marker dependent application.

More over the marker independent application had more points, twenty six in number, along the subject's spine as compared to 6 markers along the subject's spine for marker dependent application. So in the marker dependent application, the other points are determined by interpolation. The interpolation method used is the use of mid-point theory which uses average distance between two points. On the other hand, for the marker independent application there was no need of interpolation to determine points of vertebrae since all 26 points of the spine were used in the algorithm of this application. Hence the marker independent application produced more accurate visual geometries of spine than those from marker dependent application. The figure 8.1 shows the two method of determining points on spine for the two application.

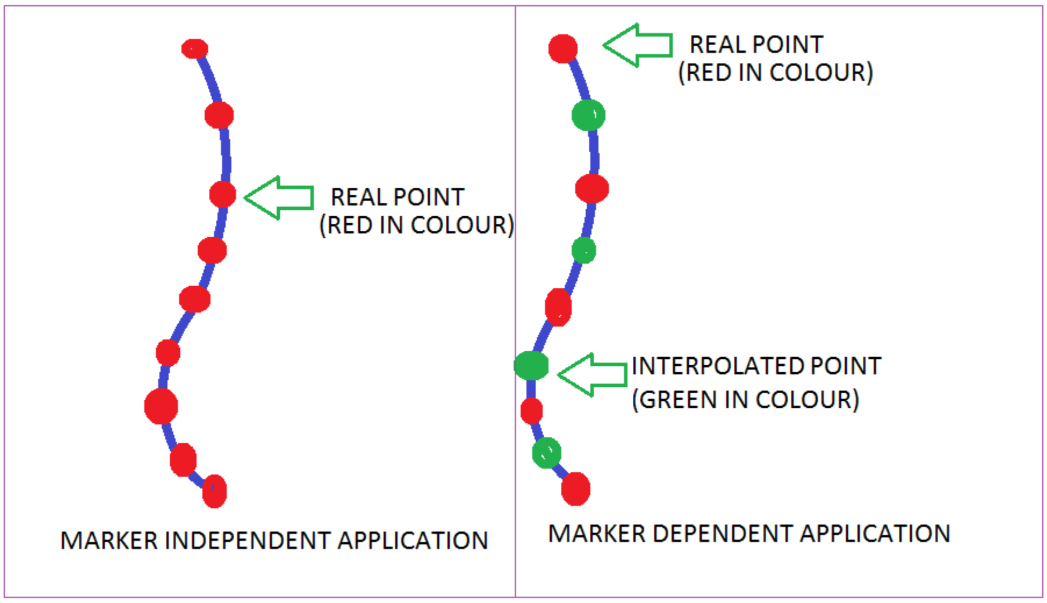


Figure 8.1: methods of locating a point on the spine using the two applications

Even though the application produced similar results, they have some different advantages and limitations. The application that does not require attaching markers on the spine of the subject is quicker to use and also applicable only to static applications of visualising geometry of the spine. On the other hand the marker dependent application is takes more time for labelling but can be used for both static and dynamic application.

In terms of potential sources of errors, the marker independent marker independent application has fewer potential sources as compare to marker dependent application. For marker independent application the major issue is that the user must label a correct landmark from vertebra cervical ,C1, and then moving downwards along the spine of the subject, otherwise if the user start labelling from sacral region upwards, both the measured parameters and visualisation will be wrong. Similarly for the marker dependent application, wrong labelling can result into meaningless parameters and visualisations. Another issue for that must always be corrected before using the marker dependent application is how markers are attached on the subject. Wrong orientation of markers or displacement of markers from their correct position can cause errors. Thus the markers must always be perpendicular to the position specified in the labelling method and also the marker be secured properly. Figure 8.2 below shows how marker attachment can affect the accuracy of the measured parameters and visualisation shape.

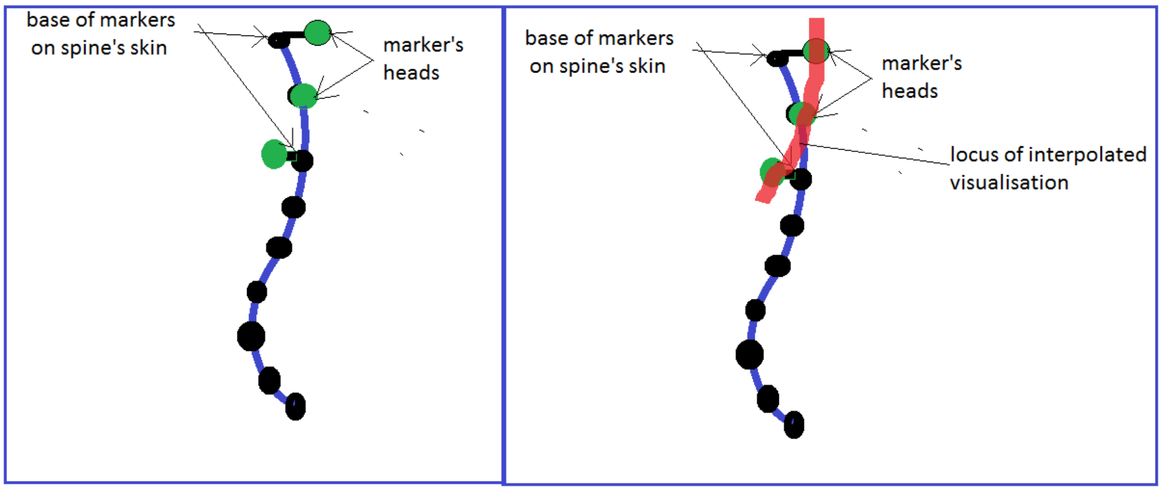

base of markers on spine's skin
marker's heads
base of markers on spine's skin
marker's heads
locus of interpolated visualisation

Figure 8. 2: wrong orientation of visualised locus based on loose or wrongly attached markers

The project's achievements enhances the knowledge of using motion capture system as both a diagnostic and treatment system for patients with spinal column deformities. The project also improves the knowledge of how to easily label markers attached on a subject by just using a pointer and a touch sensor. This method to label markers is a new idea. Besides the achievements improves the knowledge of how to measure geometry of the spine using motion capture system, even when attaching markers on the body of the subject would be impossible.

In conclusion the two applications were developed successfully and they were able to produce meaningful results in terms of numerical parameters and visualisations of the geometry of subject's spine. The results of parameters obtained from the applications were not very different from manual measurements hence it can be concluded that the measured parameters are valid. The applications records the numerical data in text files for future use like simulations and references.

Chapter 9: CONCLUSION AND RECOMMENDATIONS

The project aims of developing application to visualise and measure parameters of the geometry of the spine were successfully achieved. The use of D-Flow based application with added three dimensional objects drawn using Sketchup software, provided a good and user friendly way of visualising subject's spine and measuring parameters that can define the geometry of the spine. This section highlights on concluding points, limitation of the project and recommendation for future work so as to develop a much better and user friendly application based on the concepts of developed marker dependent and marker independent applications.

Firstly it can be concluded that it is possible to measure 3D parameters and visualise geometry of the spine using motion capture system as a way of determining curvature of spines of patients with scoliosis. Beside using pointer and touch sensor provides an innovative and easier way to label markers or points and hence this methods eliminates the barrier of failing to label markers correctly when using Soft wares that facilitates motion capture such as Vicon Nexus. In addition both methods of attaching markers and not attaching markers on the subject were successful in producing a meaningful visualisation of the spine. It can also be conclude that marker independent application was more accurate than marker dependent application because of the marker independent application had many points incorporated in the algorithm of the application. Hence for the marker dependent application, it can be more accurate to use more markers along the spine, for instance twenty six markers, each marker being placed above a corresponding vertebra. Many markers with more tracking equations incorporated in the algorithm that can eliminate some interpolation errors.

The parameters and visualisation can be helpful when determining a point where to apply corrective forces when making a brace for patients with scoliosis.

Despite the fact that the development and testing of the two applications, there are some limitations. One of the limitations of this project is that the testing was carried out on laboratory skeleton and not on real patients. Another limitation of the project is that it provides visualisation based on modelling, that cannot show interference of vertebrae and other internal organs. Besides the applications can only save data in text (.txt) format and cannot be saved into other formats such as pdf and excel. However the data can be copied from text files and pasted in Excel or Minitab for further analysis.

In conclusion the developed applications provides an innovative way of visualising the spine geometry. More especially the application that does not require attaching markers on the subject's spine provide a better alternative of measuring spine geometry on subjects or patients with some issues of attaching markers on their spine.

Chapter 10: REFERENCES

[1] National Scoliosis Foundation. http://www.scoliosis.org/info.php.

[2] Martini, N. a. B. (2014) Fundamentals of Anatomy & Physiology. In: Editor (ed.)^(eds.)|. *Book Title*|. Publisher|, City|.

[3] Boundless (2015) Human Axial Skeleton.

[4] center, U. o. M. M., Scoliosis. http://umm.edu/health/medical/reports/articles/scoliosis.

[5] Wellness, S. a., Scoliosis | Seattle Children's Hospital. http://www.seattlechildrens.org/kids-health/kids/health-problems/bones-muscles-and-joints/scoliosis/.

[6] SRS, Adolescent Idiopathic Scoliosis - Scoliosis Research Society (SRS). http://www.srs.org/patient_and_family/scoliosis/idiopathic/adolescents/.

[7] FONAR, Fonar Upright (TM) MRI. http://www.fonar.com/patient/.

[8] Medscape, Idiopathic Scoliosis Imaging: Overview, Radiography, Computed Tomography. http://emedicine.medscape.com/article/413157-overview.

[9] Ritsuko K Pooh, A. K., Jaypee Journals | Show Contents. http://www.jaypeejournals.com/eJournals/ShowText.aspx?ID=147&Type=FREE&TYP=TOP&IN=_eJournals/images/JPLOGO.gif&IID=18&isPDF=NO.

[10] (ACR), R. S. o. N. A. R. a. A. C. o. R. (2015) Nuclear Medicine, General.

[11] RiverRadiology, River Radiology - Nuclear Medicine. http://www.riverradiology.com/nuclear.html.

[12] Vicon, What's New in Vicon Nexus 2.1. https://www.google.co.uk/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&ved=0CCIQFjAAahUKEwiB0rLxp-fGAhXiFNsKHV11D44&url=http%3A%2F%2Fvicon.com%2Fdocumentation%2Fnexus%2Fv2.1%2Fdesktop%2FViconNexusWhatsNew.pdf&ei=mqerVYGdHOKp7Abd6r3wCA&usg=AFQjCNFI1T4LbtJPQI-V5qxgDSF8dLE2AA&sig2=xuUFBMANX4z4cFELvRmxNg.

[13] Zaina, F., J. C. De Mauroy, T. Grivas, M. T. Hresko, T. Kotwizki, T. Maruyama, N. Price, M. Rigo, L. Stikeleather, J. Wynne, and S. Negrini (2014) Bracing for scoliosis in 2014: state of the art. *Eur J Phys Rehabil Med*. 50: 93-110.

[14] Negrini, S., A. G. Aulisa, L. Aulisa, A. B. Circo, J. C. de Mauroy, J. Durmala, T. B. Grivas, P. Knott, T. Kotwicki, T. Maruyama, S. Minozzi, J. P. O'Brien, D. Papadopoulos, M. Rigo, C. H. Rivard, M. Romano, J. H. Wynne, M. Villagrasa, H. R. Weiss, and F. Zaina (2012) 2011 SOSORT guidelines: Orthopaedic and Rehabilitation treatment of idiopathic scoliosis during growth. pp. 3. *Scoliosis*, City.

[15] Spine-Health, Types of Scoliosis Braces. http://www.spine-health.com/conditions/scoliosis/types-scoliosis-braces.

[16] Health, S., Scoliosis Surgery: Potential Risks and Postoperative Care. http://www.spine-health.com/conditions/scoliosis/scoliosis-surgery-potential-risks-and-postoperative-care.

[17] Długosz, M. M., W. Chwała, P. Maciejasz, and W. Alda (2012) Realistic model of spine geometry in the human skeleton in the Vicon system.

[18] (2004) Non-invasive approach towards the in vivo estimation of 3D inter-vertebral movements: methods and preliminary results. 26: 841–853.

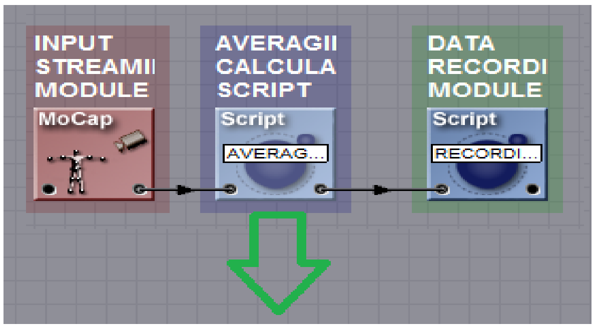
[19] Solomito, M. J., The Use of Motion Analysis T echnolog y as an Alternative Means of Assessing Spinal Deformity in P atients w ith Adolescent I diopathic Scoliosis. https://www.google.co.uk/url?sa=t&rct=j&q=&esrc=s&source=web&cd=2&ved=0CC0QFjABahUKEwiWpKfqj6XHAhUpCdsKHan3CL4&url=http%3A%2F%2Fdigitalcommons.uconn.edu%2Fcgi%2Fviewcontent.cgi%3Farticle%3D1100%26context%3Dgs_theses&ei=4A_MVZa0OKmS7Aap76PwCw&usg=AFQjCNFg5fPEXXcw0cXF1KesLOPDLa9I0Q&sig2=0qPyF3U5LBs2X8KTqQqR9Q.

[20] wikipedia, Phidget - Wikipedia, the free encyclopedia. https://en.wikipedia.org/wiki/Phidget.

Chapter 11: APPENDIX

POINTING WAND CALIBRATION SCRIPT

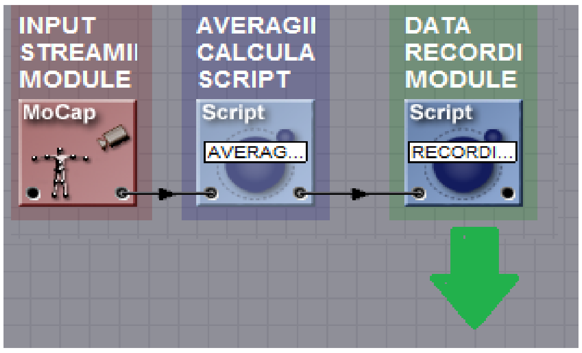
AVERAGING & CALCULATION SCRIPT



```
-- Initialization of all (not local) variables
ini = ini or 0
input =[17] or 0
aver={} or 0

-- Function definitions

--condition for calculating maximun number of channels

if ini == 0 then
markers=4
channels=markers*3
n=1
ini = 1
end

aver[0]=markers
--code for reading input data of channels
for i = 1 , channels do
aver[i]=inputs.get("Input"..i)
end

n=n+1 --increasing value of frame variable "n"

--marking avarage of the positon of the markers
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

--creating output channels to be used by linked module

for i = 0 , channels do

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

```
end
```

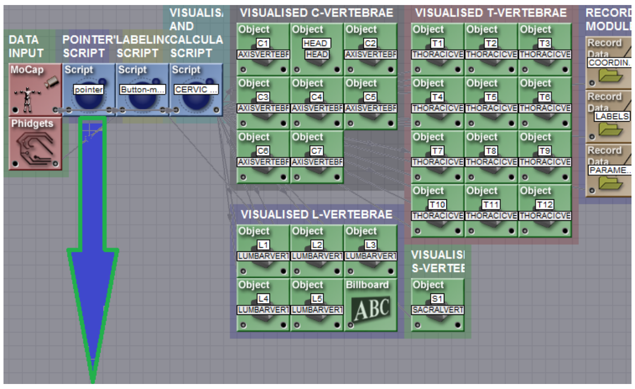
DATA RECORDING SCRIPT



```
-- initialisation of variables
ini = ini or 0
-- Conditions for executing recording of input data

if ini == 0 then

-- name of location where text document will be stored
outfname='C:\\Users\\edwin\\OneDrive\\Documents\\CAREN Resources\\Data\\envisage upper
limb\\edwinpointer.txt'

io.output(outfname)

--input get function used to stored input data in markers variable
markers=inputs.get("Input"..0)

--markers variable used to calculate maximum number of channels
channels=markers*3

--writing data input data into output file
io.write(" ")
io.write(inputs.get("Input"..0)," ")
for i = 1 , channels do
io.write(inputs.get("Input"..i)," ")
end
io.write("\n")
io.flush()

ini = 1
end



-- Script update (all parts below are part of the script update)
-- Event handling
-- Input based variables
-- Application logic
-- Set Output
```

MARKER DEPENDENT APPLICATION

POINTER SCRIPT



```
o=o or 0

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

-- 3)Initialization code first frame ini==0
if ini == 0 then
c=c or {}
m1=m1 or {}
m2=m2 or {}
t=t or {}
label = label or {}
nlabel=nlabel or {}
length=length or {}
flength=flength or {}
mcount=mcount or {}

pointers=pointers or {}
pmx=pmx or {}
pmy=pmy or {}
pmz=pmz or {}

local allinputs={}
local alloutputs={}
for i=1 ,150 do allinputs[i]="input"..i end
inputs.setchannels(unpack(allinputs))
local alloutputs={}
for i=1 ,153 do alloutputs[i]="output"..i end
outputs.setchannels(unpack(alloutputs))

--
--load in pointer
--

infname='C:\\CAREN Resources\\Data\\envisage upper limb\\edwinpointer.txt'
io.input(infname)
nummarkers=io.read("*number")
numlengths=nummarkers*3

--Creates pointer markers--
pointers[1]=object.create("Cube", "Green")
pointers[1]:setscaling(0.04,0.04,0.04)
pointers[1]:setposition(-999,-999,-999)
```

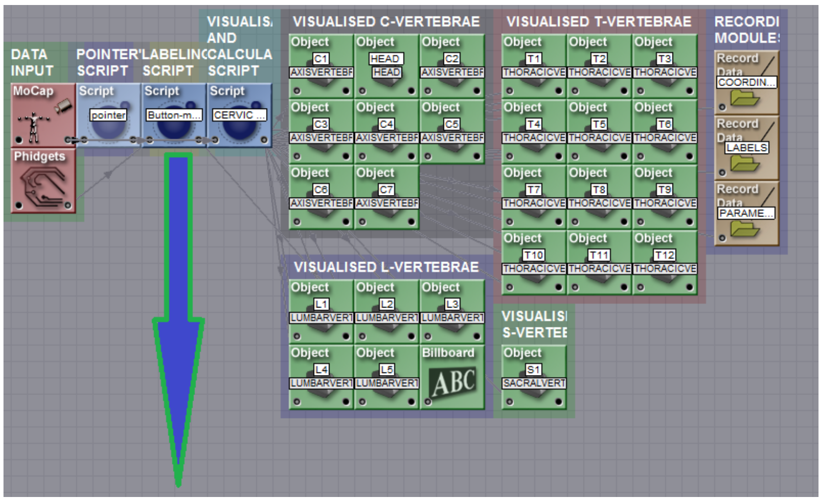
```
for i = 1 , numlengths do
c[i]=io.read("*number")
end
count=0
for i = 1 ,3 do
for j=i+1 , 4 do

--Calculating the lengths between pointer markers--
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
m1[count]=i
m2[count]=j
end
end
max=000
min=999
minlen=0
maxlen=0

for i=1 , count do
if length[i]<=min then
min=length[i]
minlen=i
end
if length[i]>=max then
max=length[i]
maxlen=i
end
end

for i=1, 4 do
label[i]=0
nlabel[i]=0
t[i]=0
end

--Calculating what markers give max/min lengths--
if m1[maxlen]==1 then t[1]=t[1]+1 end
if m1[maxlen]==2 then t[2]=t[2]+1 end
if m1[maxlen]==3 then t[3]=t[3]+1 end
if m1[maxlen]==4 then t[4]=t[4]+1 end
if m2[maxlen]==1 then t[1]=t[1]+1 end
if m2[maxlen]==2 then t[2]=t[2]+1 end
if m2[maxlen]==3 then t[3]=t[3]+1 end
if m2[maxlen]==4 then t[4]=t[4]+1 end
if m1[minlen]==1 then t[1]=t[1]+1 end
if m1[minlen]==2 then t[2]=t[2]+1 end
if m1[minlen]==3 then t[3]=t[3]+1 end
if m1[minlen]==4 then t[4]=t[4]+1 end
if m2[minlen]==1 then t[1]=t[1]+1 end
if m2[minlen]==2 then t[2]=t[2]+1 end
if m2[minlen]==3 then t[3]=t[3]+1 end
if m2[minlen]==4 then t[4]=t[4]+1 end



label[1]=0
```

```
if t[1]==2 then label[1]=1 end
if t[2]==2 then label[1]=2 end
if t[3]==2 then label[1]=3 end
if t[4]==2 then label[1]=4 end
nlabel[label[1]]=1
print("first marker is ",label[1])
t[1]=0
t[2]=0
t[3]=0
t[4]=0

if m2[minlen]==1 then t[1]=t[1]+1 end
if m2[minlen]==2 then t[2]=t[2]+1 end
if m2[minlen]==3 then t[3]=t[3]+1 end
if m2[minlen]==4 then t[4]=t[4]+1 end
if m1[minlen]==1 then t[1]=t[1]+1 end
if m1[minlen]==2 then t[2]=t[2]+1 end
if m1[minlen]==3 then t[3]=t[3]+1 end
if m1[minlen]==4 then t[4]=t[4]+1 end


t[label[1]]=0
label[2]=0
if t[1]==1 then label[2]=1 end
if t[2]==1 then label[2]=2 end
if t[3]==1 then label[2]=3 end
if t[4]==1 then label[2]=4 end
nlabel[label[2]]=1
print("second marker is ",label[2])
t[1]=0
t[2]=0
t[3]=0
t[4]=0

if m1[maxlen]==1 then t[1]=t[1]+1 end
if m1[maxlen]==2 then t[2]=t[2]+1 end
if m1[maxlen]==3 then t[3]=t[3]+1 end
if m1[maxlen]==4 then t[4]=t[4]+1 end
if m2[maxlen]==1 then t[1]=t[1]+1 end
if m2[maxlen]==2 then t[2]=t[2]+1 end
if m2[maxlen]==3 then t[3]=t[3]+1 end
if m2[maxlen]==4 then t[4]=t[4]+1 end


t[label[1]]=0
t[label[2]]=0
label[4]=0
if t[1]==1 then label[4]=1 end
if t[2]==1 then label[4]=2 end
if t[3]==1 then label[4]=3 end
if t[4]==1 then label[4]=4 end
nlabel[label[4]]=1
print("fourth marker is ",label[4])
if nlabel[1]==0 then label[3]=1 end
if nlabel[2]==0 then label[3]=2 end
if nlabel[3]==0 then label[3]=3 end
if nlabel[4]==0 then label[3]=4 end
print("third marker is ",label[3])
```

```
--Calculating the distances between markers 1-3 and 1&4--

flen1=(c[((label[1]-1)*3)+1]-c[((label[3]-1)*3)+1])^2
flen1=flen1+(c[((label[1]-1)*3)+2]-c[((label[3]-1)*3)+2])^2
flen1=flen1+(c[((label[1]-1)*3)+3]-c[((label[3]-1)*3)+3])^2
flen1=flen1^0.5
flen2=(c[((label[1]-1)*3)+1]-c[((label[2]-1)*3)+1])^2
flen2=flen2+(c[((label[1]-1)*3)+2]-c[((label[2]-1)*3)+2])^2
flen2=flen2+(c[((label[1]-1)*3)+3]-c[((label[2]-1)*3)+3])^2
flen2=flen2^0.5
flen3=(c[((label[2]-1)*3)+1]-c[((label[3]-1)*3)+1])^2
flen3=flen3+(c[((label[2]-1)*3)+2]-c[((label[3]-1)*3)+2])^2
flen3=flen3+(c[((label[2]-1)*3)+3]-c[((label[3]-1)*3)+3])^2
flen3=flen3^0.5
flen4=(c[((label[1]-1)*3)+1]-c[((label[4]-1)*3)+1])^2
flen4=flen4+(c[((label[1]-1)*3)+2]-c[((label[4]-1)*3)+2])^2
flen4=flen4+(c[((label[1]-1)*3)+3]-c[((label[4]-1)*3)+3])^2
flen4=flen4^0.5
ratio=flen4/flen1
print(flen4,flen1, ratio)

ini=1
end


for i=1,150 do
c[i]=inputs.get("input"..i)
outputs.set("output"..i,c[i])
end

for i=1,50 do
mcount[i]=0
end

for i=1,4 do
pmx[i]=999
pmy[i]=999
pmz[i]=999

end

for i = 1 ,49 do
for j=i+1 , 50 do
flength1=(c[((i-1)*3)+1]-c[((j-1)*3)+1])^2
flength1=flength1+(c[((i-1)*3)+2]-c[((j-1)*3)+2])^2
flength1=flength1+(c[((i-1)*3)+3]-c[((j-1)*3)+3])^2
flength1=flength1^0.5

if ((flen1-flength1)^2)^0.5 <=0.001 then
mcount[i]=mcount[i]+1
mcount[j]=mcount[j]+1
end
if  (((flen2-flength1)^2)^0.5) <=0.001 then
mcount[i]=mcount[i]+1
mcount[j]=mcount[j]+1
end
if ((flen3-flength1)^2)^0.5 <=0.001  then
mcount[i]=mcount[i]+1
mcount[j]=mcount[j]+1
end
```

```
end
end

pcount=1
for i=1,50 do
if mcount[i]>=2 then

pmx[pcount]=c[((i-1)*3)+1]
pmy[pcount]=c[((i-1)*3)+2]
pmz[pcount]=c[((i-1)*3)+3]
pcount=pcount+1
if pcount>=4 then pcount =3 end
end
end

if (pmx[1]^2+pmx[2]^2+pmx[3]^2)^0.5 > 0 and (pmx[1]^2+pmx[2]^2+pmx[3]^2)^0.5 < 999
then
pmx[4]=pmx[label[1]]+ratio*((pmx[label[3]])-(pmx[label[1]]))
pmy[4]=pmy[label[1]]+ratio*((pmy[label[3]])-(pmy[label[1]]))
pmz[4]=pmz[label[1]]+ratio*((pmz[label[3]])-(pmz[label[1]]))

if(pmx[4]^2+pmy[4]^2+pmz[4]^2)^0.5 >(pmx[3]^2+pmy[3]^2+pmz[3]^2)^0.5
then
pointers[1]:setposition(pmx[4],pmy[4],pmz[4])
end
if(pmx[4]^2+pmy[4]^2+pmz[4]^2)^0.5 <(pmx[3]^2+pmy[3]^2+pmz[3]^2)^0.5
then
pmx[4]=pmx[label[1]]+ratio*((pmx[label[1]])-(pmx[label[3]]))
pmy[4]=pmy[label[1]]+ratio*((pmy[label[1]])-(pmy[label[3]]))
pmz[4]=pmz[label[1]]+ratio*((pmz[label[1]])-(pmz[label[3]]))

pointers[1]:setposition(pmx[4],pmy[4],pmz[4])
end
end


outputs.set("output151",pmx[4])
outputs.set("output152",pmy[4])
outputs.set("output153",pmz[4])
```

LABELLING SCRIPT



```
bini =bini or 0
bini=0
if bini==0 then
ini=ini or 0
markers = markers or{}
allinputs = allinputs or {}
input = input or {}
indicator=indicator or {}

outputs.setchannels("LTC1x", "LTC1y", "LTC1z","RTC1x", "RTC1y", "RTC1z",
"C4C5x","C4C5y","C4C5z","C7T1x","C7T1y","C7T1z","LTSHx","LTSHy","LTSHz",
"RTSHx","RTSHy","RTSHz","RTROx","RTROy","RTROz","T7T8x","T7T8y","T7T8z",
"LTROx", "LTROy", "LTROz","LTDIx", "LTDIy", "LTDIz","RTDIx","RTDIy","RTDIz",
"RRIBx","RRIBy","RRIBz","T12L1x","T12L1y","T12L1z","LRIBx","LRIBy","LRIBz",
"LLROx","LLROy","LLROz","L3L4x","L3L4y","L3L4z","RLROx","RLROy","RLROz",
"RILIx", "RILIy", "RILIz","LILIx","LILIy","LILIz","LPS1x","LPS1y","LPS1z",
"L5S1x","L5S1y","L5S1z","RPSIx","RPSIy","RPSIz","S2S3x","S2S3y","S2S3z",
"LASIx","LASIy","LASIz","RASIx", "RASIy", "RASIz","RGTRx", "RGTRy", "RGTRz",
"LGTRx","LGTRy","LGTRz","LTHIx","LTHIy","LTHIz","RTHIx","RTHIy","RTHIz",
"RKNEx","RKNEy","RKNEz","LKNEx","LKNEy","LKNEz","LTIBx","LTIBy","LTIBz",
"RTIBx","RTIBy","RTIBz","RANKx","RANKy","RANKz","LANKx","LANKy","LANKz",
"LTOEx", "LTOEy", "LTOEz","RTOEx","RTOEy","RTOEz","STT5x","STT5y","STT5z",
"STT8x","STT8y","STT8z","UMBIx","UMBIy","UMBIz","Markernumber")


if ini==0 then
   for i = 1, 154 do
      allinputs[i] = "Channel"..i
   end

   inputs.setchannels(unpack(allinputs))

button1=0
button2=0
start=0
stop=0
count=0
value=-1
c=c or {}
input=input or {}
```

```
markername=markername or {}
markers = markers or{}
local allinputs={}

for i=1 ,154 do allinputs[i]="input"..i
end
inputs.setchannels(unpack(allinputs))
local alloutputs={}

if value== -1 then
    start=1
    stop=41
    count=1
    else
    start=value
    stop=value
    count=value
    end
ini=1
end

for i = 1, 154 do
    input[i] = inputs.get("input"..i)
end


if value~= -1 then
    start=value
    stop=value
    count=value
end

if value~=0 then

if count<=stop then
    button2=inputs.get("input154")
    if button2 == 0
    then button2=0
    else button2=1
    end

if button2==1 then
    if button1==0 then
        button1=1
        print("store data", count)

        minlength=999

            for i=1,154 do
            c[i]=inputs.get("input"..i)
            end
--finding marker nearest pointer--
        for i=1,50 do
            length=(c[((i-1)*3)+1]-c[151])^2
            length=length+(c[((i-1)*3)+2]-c[152])^2
            length=length+(c[((i-1)*3)+3]-c[153])^2
            length=length^0.5

                if length < minlength then
                    minlength = length
                    marker = i
                    end
                end
--[[if (input[1])^2< (input[4])^2
and (input[1])^2< (input[7])^2
and(input[1])^2<(input[9])^2
and (input[1])^2<(input[12])^2
and (input[1])^2<(input[15])^2
then
print("input1 is less than input 4, 7 and 9") else name =1
print("input1 is equal or greater than input 2")
end]]--
        if minlength < 0.03 then
            if printed ~= marker then
                print("marker number", marker)
                printed = marker
                outputs.set("Markernumber",marker)

                if marker <= 50 then
                        mmx = ((marker-1)*3)+1
                        mmy = ((marker-1)*3)+2
                        mmz = ((marker-1)*3)+3
                        end

print (input[mmx],input[mmy], input[mmz])
                end
            end
--[[Capturing marker coordinates]]--

if count==1 and input[mmx]~=nil and input[mmy]~=nil and input[mmy]~=nil
then
LTC1x=input[mmx]
LTC1y=input[mmy]
LTC1z=input[mmz]

aa=mmx
ab=mmy
ac=mmz

LTC1x=input[mmx]
LTC1y=input[mmy]
LTC1z=input[mmz]
print("LTC1 coordinates", LTC1x,LTC1y,LTC1z)
outputs.set("LTC1x", aa)
outputs.set("LTC1y", ab)
outputs.set("LTC1z", ac)
indicator[1]=object.create("Cube", "Blue")
indicator[1]:setscaling(0.04,0.04,0.04)
indicator[1]:setposition(LTC1x,LTC1y-0.02,LTC1z)
else if input[mmx]==nil and input[mmy]==nil and input[mmy]==nil
then count=1 value=1  print("point the wand closer to marker")end
end

if  count ==2
then
ad=mmx
ae=mmy
af=mmz

RTC1x=input[mmx]
```

```
RTC1y=input[mmy]
RTC1z=input[mmz]
print("RTC1 coordinates",RTC1x,RTC1y,RTC1z)
outputs.set("RTC1x", ad)
outputs.set("RTC1y", ae)
outputs.set("RTC1z", af)
indicator[2]=object.create("Cube", "Blue")
indicator[2]:setscaling(0.04,0.04,0.04)
indicator[2]:setposition(RTC1x,RTC1y-0.02,RTC1z)
end
if count ==3
then
ag=mmx
ah=mmy
ai=mmz
C4C5x=input[mmx]
C4C5y=input[mmy]
C4C5z=input[mmz]
print("C4C5 coordinates",C4C5x,C4C5y,C4C5z)
outputs.set("C4C5x",ag)
outputs.set("C4C5y", ah)
outputs.set("C4C5z", ai)
indicator[3]=object.create("Cube", "Blue")
indicator[3]:setscaling(0.04,0.04,0.04)
indicator[3]:setposition(C4C5x,C4C5y-0.02,C4C5z)
end
if count ==4
then
aj=mmx
ak=mmy
al=mmz
C7T1x=input[mmx]
C7T1y=input[mmy]
C7T1z=input[mmz]
print("C7T1 coordinates",C7T1x,C7T1y,C7T1z)
outputs.set("C7T1x",aj)
outputs.set("C7T1y", ak)
outputs.set("C7T1z", al)
indicator[4]=object.create("Cube", "Blue")
indicator[4]:setscaling(0.04,0.04,0.04)
indicator[4]:setposition(C7T1x,C7T1y-0.02,C7T1z)
end
if count==5 then
am=mmx
an=mmy
ao=mmz
LTSHx=input[mmx]
LTSHy=input[mmy]
LTSHz=input[mmz]
print("LTSH coordinates", LTSHx,LTSHy,LTSHz)
outputs.set("LTSHx",am)
outputs.set("LTSHy", an)
outputs.set("LTSHz", ao)
indicator[5]=object.create("Cube", "Blue")
indicator[5]:setscaling(0.04,0.04,0.04)
indicator[5]:setposition(LTSHx,LTSHy-0.02,LTSHz)
end
if count ==6
then
ap=mmx
```

```
aq=mmy
ar=mmz
RTSHx=input[mmx]
RTSHy=input[mmy]
RTSHz=input[mmz]
print("RTSH coordinates",RTSHx,RTSHy,RTSHz)
outputs.set("RTSHx",ap)
outputs.set("RTSHy", aq)
outputs.set("RTSHz", ar)
indicator[6]=object.create("Cube", "Blue")
indicator[6]:setscaling(0.04,0.04,0.04)
indicator[6]:setposition(RTSHx,RTSHy-0.02,RTSHz)
end
if count ==7
then
as=mmx
at=mmy
au=mmz
RTROx=input[mmx]
RTROy=input[mmy]
RTROz=input[mmz]
print("RTRO coordinates",RTROx,RTROy,RTROz)
outputs.set("RTROx",as)
outputs.set("RTROy", at)
outputs.set("RTROz", au)
indicator[7]=object.create("Cube", "Blue")
indicator[7]:setscaling(0.04,0.04,0.04)
indicator[7]:setposition(RTROx,RTROy-0.02,RTROz)
end
if count ==8
then
av=mmx
aw=mmy
ax=mmz
T7T8x=input[mmx]
T7T8y=input[mmy]
T7T8z=input[mmz]
print("T7T8 coordinates",T7T8x,T7T8y,T7T8z)
outputs.set("T7T8x",av)
outputs.set("T7T8y", aw)
outputs.set("T7T8z", ax)
indicator[8]=object.create("Cube", "Blue")
indicator[8]:setscaling(0.04,0.04,0.04)
indicator[8]:setposition(T7T8x,T7T8y-0.02,T7T8z)
end

if count==9 then
ay=mmx
az=mmy
ba=mmz
LTROx=input[mmx]
LTROy=input[mmy]
LTROz=input[mmz]
print("LTRO coordinates", LTROx,LTROy,LTROz)
outputs.set("LTROx", ay)
outputs.set("LTROy", az)
outputs.set("LTROz", ba)
indicator[9]=object.create("Cube", "Blue")
indicator[9]:setscaling(0.04,0.04,0.04)
indicator[9]:setposition(LTROx,LTROy-0.02,LTROz)
```

```
end
if count ==10
then
bb=mmx
bc=mmy
bd=mmz
LTDIx=input[mmx]
LTDIy=input[mmy]
LTDIz=input[mmz]
print("LTDI coordinates",LTDIx,LTDIy,LTDIz)
outputs.set("LTDIx", bb)
outputs.set("LTDIy", bc)
outputs.set("LTDIz", bd)
indicator[10]=object.create("Cube", "Blue")
indicator[10]:setscaling(0.04,0.04,0.04)
indicator[10]:setposition(LTDIx,LTDIy-0.02,LTDIz)
end

if count ==11
then
be=mmx
bf=mmy
bg=mmz
RTDIx=input[mmx]
RTDIy=input[mmy]
RTDIz=input[mmz]
print("RTDI coordinates",RTDIx,RTDIy,RTDIz)
outputs.set("RTDIx",be)
outputs.set("RTDIy", bf)
outputs.set("RTDIz", bg)
indicator[11]=object.create("Cube", "Blue")
indicator[11]:setscaling(0.04,0.04,0.04)
indicator[11]:setposition(LTDIx,LTDIy-0.02,LTDIz)
end
if count==12 then
bh=mmx
bi=mmy
bj=mmz
RRIBx=input[mmx]
RRIBy=input[mmy]
RRIBz=input[mmz]
print("RRIB coordinates", RRIBx,RRIBy,RRIBz)
outputs.set("RRIBx",bh)
outputs.set("RRIBy", bi)
outputs.set("RRIBz", bj)
indicator[12]=object.create("Cube", "Blue")
indicator[12]:setscaling(0.04,0.04,0.04)
indicator[12]:setposition(RRIBx,RRIBy-0.02,RRIBz)
end
if count ==13
then
bk=mmx
bl=mmy
bm=mmz
T12L1x=input[mmx]
T12L1y=input[mmy]
T12L1z=input[mmz]
print("T12L1 coordinates",T12L1x,T12L1y,T12L1z)
outputs.set("T12L1x",bk)
outputs.set("T12L1y", bl)
outputs.set("T12L1z", bm)
indicator[13]=object.create("Cube", "Blue")
indicator[13]:setscaling(0.04,0.04,0.04)
indicator[13]:setposition(T12L1x,T12L1y-0.02,T12L1z)
end
if count ==14
then
bn=mmx
bo=mmy
bp=mmz
LRIBx=input[mmx]
LRIBy=input[mmy]
LRIBz=input[mmz]
print("LRIB coordinates",LRIBx,LRIBy,LRIBz)
outputs.set("LRIBx",bn)
outputs.set("LRIBy", bo)
outputs.set("LRIBz", bp)
indicator[14]=object.create("Cube", "Blue")
indicator[14]:setscaling(0.04,0.04,0.04)
indicator[14]:setposition(LRIBx,LRIBy-0.02,LRIBz)
end
if count ==15
then
bq=mmx
br=mmy
bs=mmz
LLROx=input[mmx]
LLROy=input[mmy]
LLROz=input[mmz]
print("LLRO coordinates",LLROx,LLROy,LLROz)
outputs.set("LLROx",bq)
outputs.set("LLROy", br)
outputs.set("LLROz", bs)
indicator[15]=object.create("Cube", "Blue")
indicator[15]:setscaling(0.04,0.04,0.04)
indicator[15]:setposition(LLROx,LLROy-0.02,LLROz)

end
if count ==16
then
bt=mmx
bu=mmy
bv=mmz
L3L4x=input[mmx]
L3L4y=input[mmy]
L3L4z=input[mmz]
print("L3L4 coordinates",L3L4x,L3L4y,L3L4z)
outputs.set("L3L4x",bt)
outputs.set("L3L4y", bu)
outputs.set("L3L4z", bv)
indicator[16]=object.create("Cube", "Blue")
indicator[16]:setscaling(0.04,0.04,0.04)
indicator[16]:setposition(L3L4x,L3L4y-0.02,L3L4z)
end
if count==17 then
bw=mmx
bx=mmy
by=mmz
RLROx=input[mmx]
RLROy=input[mmy]
```

```
RLROz=input[mmz]
print("RLRO coordinates", RLROx,RLROy,RLROz)
outputs.set("RLROx", bw)
outputs.set("RLROy", bx)
outputs.set("RLROz", by)
indicator[17]=object.create("Cube", "Blue")
indicator[17]:setscaling(0.04,0.04,0.04)
indicator[17]:setposition(RLROx,RLROy-0.02,RLROz)
end
if count ==18
then
bz=mmx
ca=mmy
cb=mmz
RILIx=input[mmx]
RILIy=input[mmy]
RILIz=input[mmz]
print("RILI coordinates",RILIx,RILIy,RILIz)
outputs.set("LTDIx", bz)
outputs.set("LTDIy", ca)
outputs.set("LTDIz", cb)
indicator[18]=object.create("Cube", "Blue")
indicator[18]:setscaling(0.04,0.04,0.04)
indicator[18]:setposition(LTDIx,LTDIy-0.02,LTDIz)
end

if count ==19
then
cc=mmx
cd=mmy
ce=mmz
LILIx=input[mmx]
LILIy=input[mmy]
LILIz=input[mmz]
print("LILI coordinates",LILIx,LILIy,LILIz)
outputs.set("LILIx",cc)
outputs.set("LILIy", cd)
outputs.set("LILIz", ce)
indicator[19]=object.create("Cube", "Blue")
indicator[19]:setscaling(0.04,0.04,0.04)
indicator[19]:setposition(LILIx,LILIy-0.02,LILIz)
end
if count==20 then
cf=mmx
cg=mmy
ch=mmz
LPS1x=input[mmx]
LPS1y=input[mmy]
LPS1z=input[mmz]
print("LPS1 coordinates", LPS1x,LPS1y,LPS1z)
outputs.set("LPS1x",cf)
outputs.set("LPS1y", cg)
outputs.set("LPS1z", ch)
indicator[20]=object.create("Cube", "Blue")
indicator[20]:setscaling(0.04,0.04,0.04)
indicator[20]:setposition(LPS1x,LPS1y-0.02,LPS1z)
end
if count ==21
then
ci=mmx
```

```
cj=mmy
ck=mmz
L5S1x=input[mmx]
L5S1y=input[mmy]
L5S1z=input[mmz]
print("L5S1 coordinates",L5S1x,L5S1y,L5S1z)
outputs.set("L5S1x", ci)
outputs.set("L5S1y", cj)
outputs.set("L5S1z", ck)
indicator[21]=object.create("Cube", "Blue")
indicator[21]:setscaling(0.04,0.04,0.04)
indicator[21]:setposition(L5S1x,L5S1y-0.02,L5S1z)
end
if count ==22
then
cl=mmx
cm=mmy
cn=mmz
RPSIx=input[mmx]
RPSIy=input[mmy]
RPSIz=input[mmz]
print("RPSI coordinates",RPSIx,RPSIy,RPSIz)
outputs.set("RPSIx",cl)
outputs.set("RPSIy",cm)
outputs.set("RPSIz",cn)
indicator[22]=object.create("Cube", "Blue")
indicator[22]:setscaling(0.04,0.04,0.04)
indicator[22]:setposition(RPSIx,RPSIy-0.02,RPSIz)
end
if count ==23
then
co=mmx
cp=mmy
cq=mmz
S2S3x=input[mmx]
S2S3y=input[mmy]
S2S3z=input[mmz]
print("S2S3 coordinates",S2S3x,S2S3y,S2S3z)
outputs.set("S2S3x",co)
outputs.set("S2S3y",cp)
outputs.set("S2S3z",cq)
indicator[23]=object.create("Cube", "Blue")
indicator[23]:setscaling(0.04,0.04,0.04)
indicator[23]:setposition(S2S3x,S2S3y-0.02,S2S3z)
end
if count ==24
then
cr=mmx
cs=mmy
ct=mmz
LASIx=input[mmx]
LASIy=input[mmy]
LASIz=input[mmz]
print("LASI coordinates",LASIx,LASIy,LASIz)
outputs.set("LASIx",cr)
outputs.set("LASIy",cs)
outputs.set("LASIz",ct)
indicator[24]=object.create("Cube", "Blue")
indicator[24]:setscaling(0.04,0.04,0.04)
indicator[24]:setposition(LASIx,LASIy-0.02,LASIz)
```

```
end
if count ==25
then
cu=mmx
cv=mmy
cw=mmz
RASIx=input[mmx]
RASIy=input[mmy]
RASIz=input[mmz]
print("RASI coordinates",RASIx,RASIy,RASIz)
outputs.set("RASIx",cu)
outputs.set("RASIy",cv)
outputs.set("RASIz",cw)
indicator[25]=object.create("Cube", "Blue")
indicator[25]:setscaling(0.04,0.04,0.04)
indicator[25]:setposition(RASIx,RASIy-0.02,RASIz)
end
if count ==26
then
cx=mmx
cy=mmy
cz=mmz
RGTRx=input[mmx]
RGTRy=input[mmy]
RGTRz=input[mmz]
print("RGTR coordinates",RGTRx,RGTRy,RGTRz)
outputs.set("RGTRx",cx)
outputs.set("RGTRy",cy)
outputs.set("RGTRz",cz)
indicator[26]=object.create("Cube", "Blue")
indicator[26]:setscaling(0.04,0.04,0.04)
indicator[26]:setposition(RGTRx,RGTRy-0.02,RGTRz)
end
if count ==27
then
da=mmx
db=mmy
dc=mmz
LGTRx=input[mmx]
LGTRy=input[mmy]
LGTRz=input[mmz]
print("LGTR coordinates",LGTRx,LGTRy,LGTRz)
outputs.set("LGTRx",da)
outputs.set("LGTRy",db)
outputs.set("LGTRz",dc)
indicator[27]=object.create("Cube", "Blue")
indicator[27]:setscaling(0.04,0.04,0.04)
indicator[27]:setposition(LGTRx,LGTRy-0.02,LGTRz)
end
if count==28 then
dd=mmx
de=mmy
df=mmz
LTHIx=input[mmx]
LTHIy=input[mmy]
LTHIz=input[mmz]
print("LTHI coordinates", LTHIx,LTHIy,LTHIz)
outputs.set("LTHIx", dd)
outputs.set("LTHIy", de)
outputs.set("LTHIz", df)
```

```
indicator[28]=object.create("Cube", "Blue")
indicator[28]:setscaling(0.04,0.04,0.04)
indicator[28]:setposition(LTHIx,LTHIy-0.02,LTHIz)
end
if count ==29
then
dg=mmx
dh=mmy
di=mmz
RTHIx=input[mmx]
RTHIy=input[mmy]
RTHIz=input[mmz]
print("RTHI coordinates",RTHIx,RTHIy,RTHIz)
outputs.set("RTHIx", dg)
outputs.set("RTHIy", dh)
outputs.set("RTHIz", di)
indicator[29]=object.create("Cube", "Blue")
indicator[29]:setscaling(0.04,0.04,0.04)
indicator[29]:setposition(RTHIx,RTHIy-0.02,RTHIz)
end

if count ==30
then
dj=mmx
dk=mmy
dl=mmz
RKNEx=input[mmx]
RKNEy=input[mmy]
RKNEz=input[mmz]
print("RKNE coordinates",RKNEx,RKNEy,RKNEz)
outputs.set("RKNEx",dj)
outputs.set("RKNEy",dk)
outputs.set("RKNEz",dl)
indicator[30]=object.create("Cube", "Blue")
indicator[30]:setscaling(0.04,0.04,0.04)
indicator[30]:setposition(RKNEx,RKNEy-0.02,RKNEz)
end
if count==31 then
dm=mmx
dn=mmy
dp=mmz
LKNEx=input[mmx]
LKNEy=input[mmy]
LKNEz=input[mmz]
print("LKNE coordinates", LKNEx,LKNEy,LKNEz)
outputs.set("LKNEx",dm)
outputs.set("LKNEy",dn)
outputs.set("LKNEz",dp)
indicator[31]=object.create("Cube", "Blue")
indicator[31]:setscaling(0.04,0.04,0.04)
indicator[31]:setposition(LKNEx,LKNEy-0.02,LKNEz)
end
if count ==32
then
dq=mmx
dr=mmy
ds=mmz
LTIBx=input[mmx]
LTIBy=input[mmy]
LTIBz=input[mmz]
```

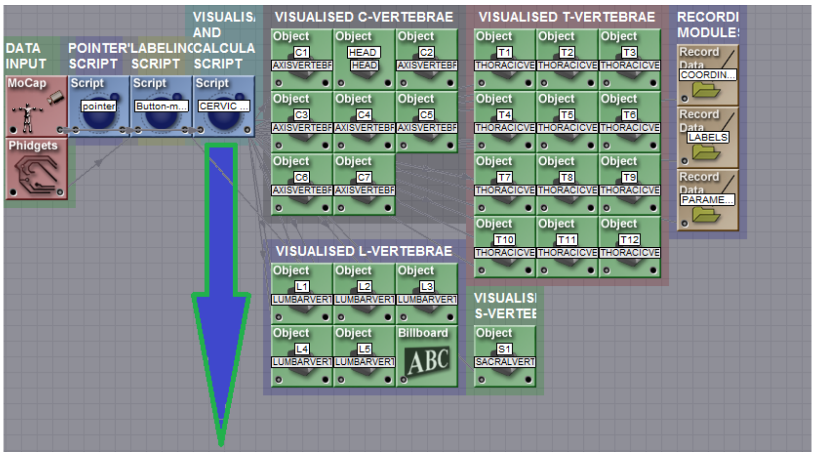
```
print("LTIB coordinates",LTIBx,LTIBy,LTIBz)
outputs.set("LTIBx",dq)
outputs.set("LTIBy",dr)
outputs.set("LTIBz",ds)
indicator[32]=object.create("Cube", "Blue")
indicator[32]:setscaling(0.04,0.04,0.04)
indicator[32]:setposition(LTIBx,LTIBy-0.02,LTIBz)
end
if count ==33
then
dt=mmx
du=mmy
dv=mmz
RTIBx=input[mmx]
RTIBy=input[mmy]
RTIBz=input[mmz]
print("RTIB coordinates",RTIBx,RTIBy,RTIBz)
outputs.set("RTIBx",dt)
outputs.set("RTIBy",du)
outputs.set("RTIBz",dv)
indicator[33]=object.create("Cube", "Blue")
indicator[33]:setscaling(0.04,0.04,0.04)
indicator[33]:setposition(RTIBx,RTIBy-0.02,RTIBz)
end
if count ==34
then
dw=mmx
dx=mmy
dy=mmz
RANKx=input[mmx]
RANKy=input[mmy]
RANKz=input[mmz]
print("RANK coordinates",RANKx,RANKy,RANKz)
outputs.set("RANKx",dw)
outputs.set("RANKy", dx)
outputs.set("RANKz", dy)
indicator[34]=object.create("Cube", "Blue")
indicator[34]:setscaling(0.04,0.04,0.04)
indicator[34]:setposition(RANKx,RANKy-0.02,RANKz)
end
if count==35 then
dz=mmx
ea=mmy
eb=mmz
LANKx=input[mmx]
LANKy=input[mmy]
LANKz=input[mmz]
print("LANK coordinates", LANKx,LANKy,LANKz)
outputs.set("LANKx", dz)
outputs.set("LANKy", ea)
outputs.set("LANKz", eb)
indicator[35]=object.create("Cube", "Blue")
indicator[35]:setscaling(0.04,0.04,0.04)
indicator[35]:setposition(LANKx,LANKy-0.02,LANKz)
end
if count ==36
then
ec=mmx
ed=mmy
ee=mmz
```

```
LTOEx=input[mmx]
LTOEy=input[mmy]
LTOEz=input[mmz]
print("LTOE coordinates",LTOEx,LTOEy,LTOEz)
outputs.set("LTOEx", ec)
outputs.set("LTOEy", ed)
outputs.set("LTOEz", ee)
indicator[36]=object.create("Cube", "Blue")
indicator[36]:setscaling(0.04,0.04,0.04)
indicator[36]:setposition(LTOEx,LTOEy-0.02,LTOEz)
end

if count ==37
then
ef=mmx
eg=mmy
eh=mmz
RTOEx=input[mmx]
RTOEy=input[mmy]
RTOEz=input[mmz]
print("RTOE coordinates",RTOEx,RTOEy,RTOEz)
outputs.set("RTOEx",ef)
outputs.set("RTOEy",eg)
outputs.set("RTOEz",eh)
indicator[37]=object.create("Cube", "Blue")
indicator[37]:setscaling(0.04,0.04,0.04)
indicator[37]:setposition(RTOEx,RTOEy-0.02,RTOEz)
end
if count==38 then
ei=mmx
ej=mmy
ek=mmz
STT5x=input[mmx]
STT5y=input[mmy]
STT5z=input[mmz]
print("STT5 coordinates", STT5x,STT5y,STT5z)
outputs.set("STT5x",ei)
outputs.set("STT5y",ej)
outputs.set("STT5z",ek)
indicator[38]=object.create("Cube", "Blue")
indicator[38]:setscaling(0.04,0.04,0.04)
indicator[38]:setposition(STT5x,STT5y-0.02,STT5z)
end
if count ==39
then
el=mmx
em=mmy
en=mmz
STT8x=input[mmx]
STT8y=input[mmy]
STT8z=input[mmz]
print("STT8 coordinates",STT8x,STT8y,STT8z)
outputs.set("STT8x",el)
outputs.set("STT8y",em)
outputs.set("STT8z",en)
indicator[39]=object.create("Cube", "Blue")
indicator[39]:setscaling(0.04,0.04,0.04)
indicator[39]:setposition(STT8x,STT8y-0.02,STT8z)
end
if count ==40 and input[mmx]~=nil and input[mmy]~=nil and input[mmy]~=nil
```

```
then
eo=mmx
ep=mmy
eq=mmz
bini=1
UMBIx=input[mmx]
UMBIy=input[mmy]
UMBIz=input[mmz]
print("UMBI coordinates",UMBIx,UMBIy,UMBIz)
outputs.set("UMBIx",eo)
outputs.set("UMBIy", ep)
outputs.set("UMBIz", eq)
indicator[40]=object.create("Cube", "Blue")
indicator[40]:setscaling(0.04,0.04,0.04)
indicator[40]:setposition(UMBIx,UMBIy-0.02,UMBIz)
indicator[1]:setposition(999,UMBIy-0.02,UMBIz)
indicator[2]:setposition(999,UMBIy-0.02,UMBIz)
indicator[3]:setposition(999,UMBIy-0.02,UMBIz)
indicator[4]:setposition(999,UMBIy-0.02,UMBIz)
indicator[5]:setposition(999,UMBIy-0.02,UMBIz)
indicator[6]:setposition(999,UMBIy-0.02,UMBIz)
indicator[7]:setposition(999,UMBIy-0.02,UMBIz)
indicator[8]:setposition(999,UMBIy-0.02,UMBIz)
indicator[9]:setposition(999,UMBIy-0.02,UMBIz)
indicator[10]:setposition(999,UMBIy-0.02,UMBIz)
indicator[11]:setposition(999,UMBIy-0.02,UMBIz)
indicator[12]:setposition(999,UMBIy-0.02,UMBIz)
indicator[13]:setposition(999,UMBIy-0.02,UMBIz)
indicator[14]:setposition(999,UMBIy-0.02,UMBIz)
indicator[15]:setposition(999,UMBIy-0.02,UMBIz)
indicator[16]:setposition(999,UMBIy-0.02,UMBIz)
indicator[17]:setposition(999,UMBIy-0.02,UMBIz)
indicator[18]:setposition(999,UMBIy-0.02,UMBIz)
indicator[19]:setposition(999,UMBIy-0.02,UMBIz)
indicator[20]:setposition(999,UMBIy-0.02,UMBIz)
indicator[21]:setposition(999,UMBIy-0.02,UMBIz)
indicator[22]:setposition(999,UMBIy-0.02,UMBIz)
indicator[23]:setposition(999,UMBIy-0.02,UMBIz)
indicator[24]:setposition(999,UMBIy-0.02,UMBIz)
indicator[25]:setposition(999,UMBIy-0.02,UMBIz)
indicator[26]:setposition(999,UMBIy-0.02,UMBIz)
indicator[27]:setposition(999,UMBIy-0.02,UMBIz)
indicator[28]:setposition(999,UMBIy-0.02,UMBIz)
indicator[29]:setposition(999,UMBIy-0.02,UMBIz)
indicator[30]:setposition(999,UMBIy-0.02,UMBIz)
indicator[31]:setposition(999,UMBIy-0.02,UMBIz)
indicator[32]:setposition(999,UMBIy-0.02,UMBIz)
indicator[33]:setposition(999,UMBIy-0.02,UMBIz)
indicator[34]:setposition(999,UMBIy-0.02,UMBIz)
indicator[35]:setposition(999,UMBIy-0.02,UMBIz)
indicator[36]:setposition(999,UMBIy-0.02,UMBIz)
indicator[37]:setposition(999,UMBIy-0.02,UMBIz)
indicator[38]:setposition(999,UMBIy-0.02,UMBIz)
indicator[39]:setposition(999,UMBIy-0.02,UMBIz)
indicator[40]:setposition(999,UMBIy-0.02,UMBIz)

end


count=count+1
```

```
        end
    end

    if button2==0 then
        if button1==1 then
            print("reset")
            button1=0
            end
        end
    end



end
end
```

VISUALISATION AND CALCULATION SCRIPT



```
-- Initilisation of variables
ini = ini or 0
allinputs = allinputs or {}
nrInputs = 270
markercoord=markercoord or {}
pmx=pmx or {}
pmy=pmy or {}
pmz=pmz or {}


-- Initialisation code
if ini == 0 then
    for i = 1, 270 do
        allinputs[i] = "input"..i

    end
    inputs.setchannels(unpack(allinputs))
 outputs.setchannels("axisx", "axisy", "axisz","rotaxisx", "rotaxisy", "rotaxisz", "Headx", "Heady",
"Headz","rotHeadx", "rotHeady", "rotHeadz","C1x", "C1y", "C1z","rotC1x", "rotC1y", "rotC1z", "C4C5x",
"C4C5y", "C4C5z", "C7T1x", "C7T1y", "C7T1z","T7T8x", "T7T8y", "T7T8z","C2x", "C2y", "C2z","rotC2x",
"rotC2y", "rotC2z",
"C3x", "C3y", "C3z","rotC3x", "rotC3y", "rotC3z", "C4x", "C4y", "C4z","rotC4x", "rotC4y", "rotC4z","C5x",
"C5y", "C5z","rotC5x", "rotC5y", "rotC5z", "C6x", "C6y", "C6z","rotC6x",
 "rotC6y", "rotC6z","C7x", "C7y", "C7z","rotC7x", "rotC7y", "rotC7z","T7T8x", "T7T8y", "T7T8z","T12L1x",
"T12L1y", "T12L1z","L5S1x", "L5S1y", "L5S1z","T1x", "T1y", "T1z","rotT1x", "rotT1y", "rotT1z", "T2x",
"T2y", "T2z","rotT2x", "rotT2y", "rotT2z",
"T3x", "T3y", "T3z","rotT3x", "rotT3y", "rotT3z", "T4x", "T4y", "T4z","rotT4x", "rotT4y", "rotT4z","T5x",
"T5y", "T5z","T6x", "T6y", "T6z", "T7x", "T7y", "T7z",
"T8x", "T8y", "T8z", "T9x", "T9y", "T9z","T10x", "T10y", "T10z","T11x", "T11y", "T11z","T12x", "T12y",
"T12z","L1x", "L1y", "L1z", "L2x", "L2y", "L2z",
"L3x", "L3y", "L3z", "L4x", "L4y", "L4z","L5x", "L5y", "L5z","C3EA", "CCEA", "SCEA","CTA","CCTA",
"SCTA","UTA", "CUTA", "SUTA","THA","STHA","TK", "LTA","CLTA","SLTA","TLA","CTLA",
"STLA","ULA", "CULA", "SULA","LMA","SLMA","LL", "LLA","CLLA","SLLA","SCA","CSCA",
"SSCA","LSA", "SLSA","PTAA","PLAA", "SPTA","C1A","SS1A","SL1", "SL2","LSA", "VHS1C1",
"rotVetz")


--[[Objects creation]]--

markercoord[1] =object.create("Cylinder", "Green")
markercoord[1]:setscaling(0.04,0.04,0.04)
markercoord[1]:setposition(-999,-999,-999)

ini = 1
end

for j=1,270 do
pmx[j]=999
pmy[j]=999
pmz[j]=999

end

-- script update
i = 1
j = 1
markers = {}
for i =1,270 do -- this creates an array of marker data, j = the marker number
   markers[i] = {}
   markers[i] = inputs.get("input"..i)
pmx=pmx or {}
pmy=pmy or {}
pmz=pmz or {}
pmx[1]=markers[1]
pmy[1]=markers[2]
pmz[1]=markers[3]
pmx[2]=markers[4]
pmy[2]=markers[5]
pmz[2]=markers[6]
       i = i+1


m=2

--[[Identifying channels]]--
aaa=markers[151]
LTC1x=markers[aaa]

aab=markers[152]
LTC1y=markers[aab]
```

```
aac=markers[153]
LTC1z=markers[aac]

aad=markers[154]
RTC1x=markers[aad]

aae=markers[155]
RTC1y=markers[aae]

aaf=markers[156]
RTC1z=markers[aaf]

if LTC1x ~= nil and RTC1x ~= nil then
Headx=0.5*(LTC1x + RTC1x) - 0.04 end
if LTC1y ~= nil and RTC1y ~= nil then
Heady=0.5*(LTC1y + RTC1y) + 0.02 end

if LTC1z ~= nil and RTC1z ~= nil then
Headz=0.5*(LTC1z + RTC1z)  end

aag=markers[157]
C4C5x=markers[aag]

aah=markers[158]
C4C5y=markers[aah]

aai=markers[159]
C4C5z=markers[aai]

aaj=markers[160]
C7T1x=markers[aaj]

aak=markers[161]
C7T1y=markers[aak]

aal=markers[162]
C7T1z=markers[aal]

aam=markers[163]
LTSHx=markers[aam]

aan=markers[164]
LTSHy=markers[aan]

aao=markers[165]
LTSHz=markers[aao]

aap=markers[166]
RTSHx=markers[aap]

aaq=markers[167]
RTSHy=markers[aaq]

aar=markers[168]
RTSHz=markers[aar]

aas=markers[169]
RTROx=markers[aas]

aat=markers[170]
```

```
RTROy=markers[aat]

aau=markers[171]
RTROz=markers[aau]

aav=markers[172]
T7T8x=markers[aav]

aaw=markers[173]
T7T8y=markers[aaw]

aax=markers[174]
T7T8z=markers[aax]

aay=markers[175]
LTROx=markers[aay]

aaz=markers[176]
LTROy=markers[aaz]

aba=markers[177]
LTROz=markers[aba]

abb=markers[178]
LTDIx=markers[abb]

abc=markers[179]
LTDIy=markers[abc]

abd=markers[180]
LTDIz=markers[abd]

abe=markers[181]
RTDIx=markers[abe]

abf=markers[182]
RTDIy=markers[abf]

abg=markers[183]
RTDIz=markers[abg]

abh=markers[184]
RRIBx=markers[abh]

abi=markers[185]
RRIBy=markers[abi]

abj=markers[186]
RRIBz=markers[abj]

abk=markers[187]
T12L1x=markers[abk]

abl=markers[188]
T12L1y=markers[abl]

abm=markers[189]
T12L1z=markers[abm]

abn=markers[190]
```

```
LRIBx=markers[abn]

abo=markers[191]
LRIBy=markers[abo]

abp=markers[192]
LRIBz=markers[abp]

abq=markers[193]
LLROx=markers[abq]

abr=markers[194]
LLROy=markers[abr]

abs=markers[195]
LLROz=markers[abs]

abt=markers[196]
L3L4x=markers[abt]

abu=markers[197]
L3L4y=markers[abu]

abv=markers[198]
L3L4z=markers[abv]

abw=markers[199]
RLROx=markers[abw]

abx=markers[200]
RLROy=markers[abx]

aby=markers[201]
RLROz=markers[aby]

abz=markers[202]
RILIx=markers[abz]

aca=markers[203]
RILIy=markers[aca]

acb=markers[204]
RILIz=markers[acb]

acc=markers[205]
LILIx=markers[acc]

acd=markers[206]
LILIy=markers[acd]

ace=markers[207]
LILIz=markers[ace]

acf=markers[208]
LPSIx=markers[acf]

acg=markers[209]
LPSIy=markers[acg]

ach=markers[210]
```

```
LPSIz=markers[ach]

aci=markers[211]
L5S1x=markers[aci]

acj=markers[212]
L5S1y=markers[acj]

ack=markers[213]
L5S1z=markers[ack]

acl=markers[214]
RPSIx=markers[acl]

acm=markers[215]
RPSIy=markers[acm]

acn=markers[216]
RPSIz=markers[acn]

aco=markers[217]
S2S3x=markers[aco]

acp=markers[218]
S2S3y=markers[acp]

acq=markers[219]
S2S3z=markers[acq]

acr=markers[220]
LASIx=markers[acr]

acs=markers[221]
LASIy=markers[acs]

act=markers[222]
LASIz=markers[act]

acu=markers[223]
RASIx=markers[acu]

acv=markers[224]
RASIy=markers[acv]

acw=markers[225]
RASIz=markers[acw]

acx=markers[226]
RGTRx=markers[acx]

acy=markers[227]
RGTRy=markers[acy]

acz=markers[228]
RGTRz=markers[acz]

ada=markers[229]
LGTRx=markers[ada]

adb=markers[230]
```

```
LGTRy=markers[adb]

adc=markers[231]
LGTRz=markers[adc]

add=markers[232]
LTHIx=markers[add]

ade=markers[233]
LTHIy=markers[ade]

adf=markers[234]
LTHIz=markers[adf]

adg=markers[235]
RTHIx=markers[adg]

adh=markers[236]
RTHIy=markers[adh]

adi=markers[237]
RTHIz=markers[adi]

adj=markers[238]
RKNEx=markers[adj]

adk=markers[239]
RKNEy=markers[adk]

adl=markers[240]
RKNEz=markers[adl]

adm=markers[241]
LKNEx=markers[adm]

adn=markers[242]
LKNEy=markers[adn]

ado=markers[243]
LKNEz=markers[ado]

adp=markers[244]
LTIBx=markers[adp]

adq=markers[245]
LTIBy=markers[adq]

adr=markers[246]
LTIBz=markers[adr]

ads=markers[247]
RTIBx=markers[ads]

adt=markers[248]
RTIBy=markers[adt]

adu=markers[249]
RTIBz=markers[adu]

adv=markers[250]
```

```
RANKx=markers[adv]

adw=markers[251]
RANKy=markers[adw]

adx=markers[252]
RANKz=markers[adx]

ady=markers[253]
LANKx=markers[ady]

adz=markers[254]
LANKy=markers[adz]

aea=markers[255]
LANKz=markers[aea]

aeb=markers[256]
LTOEx=markers[aeb]

aec=markers[257]
LTOEy=markers[aec]

aed=markers[258]
LTOEz=markers[aed]

aee=markers[259]
RTOEx=markers[aee]

aef=markers[260]
RTOEy=markers[aef]

aeg=markers[261]
RTOEz=markers[aeg]

aeh=markers[262]
STT5x=markers[aeh]

aei=markers[263]
STT5y=markers[aei]

aej=markers[264]
STT5z=markers[aej]

aek=markers[265]
STT8x=markers[aek]

ael=markers[266]
STT8y=markers[ael]

aem=markers[267]
STT8z=markers[aem]

aen=markers[268]
UMBIx=markers[aen]

aeo=markers[269]
UMBIy=markers[aeo]

aep=markers[270]
```

```
UMBIz=markers[aep]



end

--setting up outputs
if Headx== nil and Heady==nil and Headz==nil then

outputs.set("Headx", 999)
outputs.set("Heady",999)
outputs.set("Headz",999)
outputs.set("rotHeadx", 45)
outputs.set("rotHeady", 0)
outputs.set("rotHeadz", 0)
outputs.set("rotVetz", -90)
end
if
Headx~= nil and Heady~=nil and Headz~=nil then


outputs.set("Headx", Headx)
outputs.set("Heady", Heady+0.05)
outputs.set("Headz", Headz)
outputs.set("rotVetz", -90)
end
if LTC1x== nil and LTC1y==nil and LTC1z==nil and RTC1x== nil and RTC1y==nil and RTC1z==nil then
outputs.set("C1x", 999)
outputs.set("C1y",999)
outputs.set("C1z",999)
outputs.set("rotC1x", 45)
outputs.set("rotC1y", 0)
outputs.set("rotC1z", 0)
outputs.set("rotVetz", -90)
end
if
LTC1x~= nil and LTC1y~=nil and LTC1z~=nil and RTC1x~= nil and RTC1y~=nil and RTC1z~=nil then
C1x =0.5*(LTC1x + RTC1x)
C1y = 0.5*(LTC1y + RTC1y)
C1z = 0.5*(LTC1z + RTC1z)
outputs.set("C1x", C1x)
outputs.set("C1y", C1y)
outputs.set("C1z", C1z)
outputs.set("rotHeadx", 0)
outputs.set("rotHeady", 0)
outputs.set("rotHeadz",0)
outputs.set("rotVetz", -90)
end

if C4C5x== nil and C4C5y==nil and C4C5==nil then
outputs.set("C4C5x", 999)
outputs.set("C4C5y",999)
outputs.set("C4C5z",999)
outputs.set("C2x", 999)
outputs.set("C2y", 999)
outputs.set("C2z", 999)
outputs.set("C3x", 999)
outputs.set("C3y", 999)
outputs.set("C3z", 999)
outputs.set("rotVetz", -90)
```

```
end
if  C4C5x~= nil and C4C5y~=nil and C4C5z~=nil  then
outputs.set("C4C5x", C4C5x)
outputs.set("C4C5y", C4C5y)
outputs.set("C4C5z", C4C5z)
C3x= 0.5*(C1x + C4C5x)
C3y= 0.5*(C1y + C4C5y)
C3z= 0.5*(C1z + C4C5z)
C2x= 0.5*(C3x + C1x)
C2y= 0.5*(C3y + C1y)
C2z= 0.5*(C3z + C1z)

dist1= ((C1x-C4C5x)^2)^0.5
dist2= ((C1y-C4C5y)^2)^0.5
dist3= ((C1z-C4C5z)^2)^0.5
totaldist1= ((C1x-C4C5x)^2 + (C1y-C4C5y)^2 +(C1z-C4C5z)^2)^0.5
space = totaldist1/5
theta3x= math.deg(math.atan((C1y-C4C5y)/(C1z-C4C5z)))
theta3y= math.deg(math.atan((C1z-C4C5z)/(C1x-C4C5x)))
theta3z= math.deg(math.atan((C1y-C4C5y)/(C1x-C4C5x)))
outputs.set("C2x", C2x)
outputs.set("C2y", C2y)
outputs.set("C2z", C2z)
outputs.set("C3x", C3x)
outputs.set("C3y", C3y)
outputs.set("C3z", C3z)
outputs.set("rotVetz", -90)
end
if C7T1x== nil and C7T1y==nil and C7T1==nil then
outputs.set("C7T1x", 999)
outputs.set("C7T1y",999)
outputs.set("C7T1z",999)
outputs.set("C6x", 999)
outputs.set("C6y", 999)
outputs.set("C6z", 999)
outputs.set("C5x", 999)
outputs.set("C5y", 999)
outputs.set("C5z", 999)
outputs.set("rotVetz", -90)
end
if  C7T1x~= nil and C7T1y~=nil and C7T1z~=nil  then
outputs.set("C7T1x", C7T1x)
outputs.set("C7T1y", C7T1y)
outputs.set("C7T1z", C7T1z)
C6x= 0.5*(C7T1x + C4C5x)
C6y= 0.5*(C7T1y + C4C5y)
C6z= 0.5*(C7T1z + C4C5z)
C5x= 0.5*(C6x + C4C5x)
C5y= 0.5*(C6y + C4C5y)
C5z= 0.5*(C6z + C4C5z)

dist4= ((C4C5x-C7T1x)^2)^0.5
dist5= ((C4C5y-C7T1y)^2)^0.5
dist6= ((C4C5z-C7T1z)^2)^0.5
totaldist2= ((C4C5x-C7T1x)^2 + (C4C5y-C7T1y)^2 +(C4C5z-C7T1z)^2)^0.5
space2 = totaldist2/3
theta4x= math.deg(math.atan((C4C5y-C7T1y)/(C4C5z-C7T1z)))
theta4y= math.deg(math.atan((C4C5z-C7T1z)/(C4C5x-C7T1x)))
theta4z= math.deg(math.atan((C4C5y-C7T1y)/(C4C5x-C7T1x)))
outputs.set("C6x", C6x)
outputs.set("C6y", C6y)
outputs.set("C6z", C6z)
outputs.set("C5x", C5x)
outputs.set("C5y", C5y)
outputs.set("C5z", C5z)
outputs.set("rotVetz", -90)
end

if T7T8x== nil and T7T8y==nil and T7T8z==nil then
outputs.set("T7T8x", 999)
outputs.set("T7T8y", 999)
outputs.set("T7T8z", 999)
outputs.set("T4x", 999)
outputs.set("T4y", 999)
outputs.set("T4z", 999)
outputs.set("T5x", 999)
outputs.set("T5y", 999)
outputs.set("T5z", 999)
outputs.set("T6x", 999)
outputs.set("T6y", 999)
outputs.set("T6z", 999)
outputs.set("T3x", 999)
outputs.set("T3y", 999)
outputs.set("T3z", 999)
outputs.set("T2x", 999)
outputs.set("T2y", 999)
outputs.set("T2z", 999)
outputs.set("rotVetz", -90)
end
if  T7T8x ~= nil and T7T8y ~=nil and T7T8z ~=nil  then
outputs.set("T7T8x", T7T8x)
outputs.set("T7T8y", T7T8y)
outputs.set("T7T8z", T7T8z)
T4x= 0.5*(C7T1x + T7T8x)
T4y= 0.5*(C7T1y + T7T8y)
T4z= 0.5*(C7T1z + T7T8z)
T3x= 0.5*(C7T1x + T4x)
T3y= 0.5*(C7T1y + T4y)
T3z= 0.5*(C7T1z + T4z)
T2x= 0.5*(C7T1x + T3x)
T2y= 0.5*(C7T1y + T3y)
T2z= 0.5*(C7T1z + T3z)
T5x= 0.5*(T3x + T7T8x)
T5y= 0.5*(T3y + T7T8y)
T5z= 0.5*(T3z + T7T8z)
T6x= 0.5*(T5x + T7T8x)
T6y= 0.5*(T5y + T7T8y)
T6z= 0.5*(T5z + T7T8z)

dist7= ((T7T8x-C7T1x)^2)^0.5
dist8= ((T7T8y-C7T1y)^2)^0.5
dist9= ((T7T8z-C7T1z)^2)^0.5
totaldist3= ((T7T8x-C7T1x)^2 + (T7T8y-C7T1y)^2 +(T7T8z-C7T1z)^2)^0.5
space3 = totaldist3/8
theta5x= math.deg(math.atan((C7T1y-T7T8y)/(C7T1z-T7T8z)))
theta5y= math.deg(math.atan((C7T1z-T7T8z)/(C7T1x-T7T8x)))
theta5z= math.deg(math.atan((C7T1y-T7T8y)/(C7T1x-T7T8x)))
outputs.set("T4x", T4x)
outputs.set("T4y", T4y)
outputs.set("T4z", T4z)
outputs.set("T5x", T5x)
outputs.set("T5y", T5y)
outputs.set("T5z", T5z)
outputs.set("T6x", T6x)
outputs.set("T6y", T6y)
outputs.set("T6z", T6z)
outputs.set("T3x", T3x)
outputs.set("T3y", T3y)
outputs.set("T3z", T3z)
outputs.set("T2x", T2x)
outputs.set("T2y", T2y)
outputs.set("T2z", T2z)
outputs.set("rotVetz", -90)
end
if  T12L1x == nil and T12L1y ==nil and T12L1z ==nil  then
outputs.set("T12L1x", 999)
outputs.set("T12L1y", 999)
outputs.set("T12L1z", 999)
outputs.set("T10x", 999)
outputs.set("T10y", 999)
outputs.set("T10z", 999)
outputs.set("T11x", 999)
outputs.set("T11y", 999)
outputs.set("T11z", 999)
outputs.set("T9x", 999)
outputs.set("T9y", 999)
outputs.set("T9z", 999)
outputs.set("T8x", 999)
outputs.set("T8y", 999)
outputs.set("T8z", 999)
outputs.set("rotVetz", -90)
end
if  T12L1x ~= nil and T12L1y ~=nil and T12L1z ~=nil  then
outputs.set("T12L1x", T12L1x)
outputs.set("T12L1y", T12L1y)
outputs.set("T12L1z", T12L1z)
T10x= 0.5*(T12L1x + T7T8x)
T10y= 0.5*(T12L1y + T7T8y)
T10z= 0.5*(T12L1z + T7T8z)
T11x= 0.5*(T12L1x + T10x)
T11y= 0.5*(T12L1y + T10y)
T11z= 0.5*(T12L1z + T10z)
T9x= 0.5*(T7T8x + T10x)
T9y= 0.5*(T7T8y + T10y)
T9z= 0.5*(T7T8z + T10z)
T8x= 0.5*(T9x + T7T8x)
T8y= 0.5*(T9y + T7T8y)
T8z= 0.5*(T9z + T7T8z)

dist10= ((T7T8x-T12L1x)^2)^0.5
dist11= ((T7T8y-T12L1y)^2)^0.5
dist12= ((T7T8z-T12L1z)^2)^0.5
totaldist4= ((T7T8x-T12L1x)^2 + (T7T8y-C7T1y)^2 +(T7T8z-C7T1z)^2)^0.5
space4 = totaldist4/8
theta5x= math.deg(math.atan((T7T8y-T12L1y)/(T7T8z-T12L1z)))
theta5y= math.deg(math.atan((T7T8z-T12L1z)/(T7T8x-T12L1x)))
theta5z= math.deg(math.atan((T7T8y-T12L1y)/(T7T8x-T12L1x)))
outputs.set("T10x", T10x)
outputs.set("T10y", T10y)
outputs.set("T10z", T10z)
outputs.set("T11x", T11x)
outputs.set("T11y", T11y)
outputs.set("T11z", T11z)
outputs.set("T9x", T9x)
outputs.set("T9y", T9y)
outputs.set("T9z", T9z)
outputs.set("T8x", T8x)
outputs.set("T8y", T8y)
outputs.set("T8z", T8z)
outputs.set("rotVetz", -90)
end
if  L3L4x == nil and L3L4y ==nil and L3L4z ==nil  then
outputs.set("T12L1x", 999)
outputs.set("T12L1y", 999)
outputs.set("T12L1z", 999)
outputs.set("L2x", 999)
outputs.set("L2y", 999)
outputs.set("L2z", 999)
end
if  L3L4x ~= nil and L3L4y ~=nil and L3L4z ~=nil  then
outputs.set("T12L1x", T12L1x)
outputs.set("T12L1y", T12L1y)
outputs.set("T12L1z", T12L1z)
L2x= 0.5*(T12L1x + L3L4x)
L2y= 0.5*(T12L1y + L3L4y)
L2z= 0.5*(T12L1z + L3L4z)

dist13= ((T12L1x + L3L4x)^2)^0.5
dist14= ((T12L1y + L3L4y)^2)^0.5
dist15= ((T12L1z + L3L4z)^2)^0.5
totaldist5= ((T12L1x + L3L4x)^2 + (T12L1y + L3L4y)^2 +(T7T8z-C7T1z)^2)^0.5
space5 = totaldist5/3
theta5x= math.deg(math.atan((T12L1y + L3L4y)/(T12L1z + L3L4z)))
theta5y= math.deg(math.atan((T12L1z + L3L4z)/(T12L1x + L3L4x)))
theta5z= math.deg(math.atan((T12L1y + L3L4y)/(T12L1x + L3L4x)))
outputs.set("L2x", L2x)
outputs.set("L2y", L2y)
outputs.set("L2z", L2z)
outputs.set("rotVetz", -90)
end
if  L5S1x == nil and L5S1y ==nil and L5S1z ==nil  then
outputs.set("L5S1x", 999)
outputs.set("L5S1y", 999)
outputs.set("L5S1z", 999)
outputs.set("L4x", 999)
outputs.set("L4y", 999)
outputs.set("L4z", 999)
outputs.set("rotVetz", -90)
end
if  L5S1x ~= nil and L5S1y ~=nil and L5S1z ~=nil  then
outputs.set("L5S1x", L5S1x)
outputs.set("L5S1y", L5S1y)
outputs.set("L5S1z", L5S1z)
L4x= 0.5*(L5S1x + L3L4x)
L4y= 0.5*(L5S1y + L3L4y)
L4z= 0.5*(L5S1z + L3L4z)

dist16= ((L3L4x-L5S1x)^2)^0.5
dist17= ((L3L4y-L5S1y)^2)^0.5
dist18= ((L3L4z-L5S1z)^2)^0.5
```

```
totaldist6= ((L3L4x-L5S1x)^2 + (L3L4y-L5S1y)^2 +(L3L4z-L5S1z)^2)^0.5
space6 = totaldist5/3
theta6x= math.deg(math.atan((L3L4y-L5S1y)/(L3L4z-L5S1z)))
theta6y= math.deg(math.atan((L3L4z-L5S1z)/(L3L4x-L5S1x)))
theta6z= math.deg(math.atan((L3L4y-L5S1y)/(L3L4x-L5S1x)))
outputs.set("L4x", L4x)
outputs.set("L4y", L4y)
outputs.set("L4z", L4z)
outputs.set("rotVetz", -90)
end




--Calculations
--[[CERVICAL ANGLE (CEA) ]]--

if C4C5x==nil and C4C5y==nil and C4C5z==nil and C7T1x ==nil and C7T1y==nil and C7T1z ==nil and
T7T8x == nil and T7T8y == nil and T7T8z == nil then
outputs.set("C3EA", 0)
outputs.set("CCEA", 0)
outputs.set("SCEA", 0)
outputs.set("CTA", 0)
outputs.set("CCTA", 0)
outputs.set("SCTA",0)
word1="Edwin is my Hero"
print(word1)
else if C4C5x~=nil and C4C5y~=nil and C4C5z~=nil and C7T1x ~=nil and C7T1y~=nil and C7T1z ~=nil
and T7T8x ~= nil and T7T8y ~= nil and T7T8z ~= nil then
a10=((C4C5x-C7T1x)^2+(C4C5y-C7T1y)^2+(C4C5z-C7T1z)^2)^0.5
b10=((C4C5x-C7T1x)^2+(C4C5z-C7T1z)^2)^0.5
CEA=math.acos((b10/a10))
CEA=math.deg(CEA)
if C4C5x > C7T1x and C4C5z < C7T1z then print("AR -quardrant 1", CEA) end
if C4C5x > C7T1x and C4C5z > C7T1z then print("AL -quardrant 2",CEA) end
if C4C5x < C7T1x and C4C5z < C7T1z then print("PL -quardrant 3",CEA) end
if C4C5x < C7T1x and C4C5z > C7T1z then print("AL -quardrant 4", CEA) end

outputs.set("C3EA", CEA)

--[[CERVICAL ANGLE2 (CCEA) ]]--
a11=((C4C5y-C7T1y)^2+(C4C5z-C7T1z)^2)^0.5
b11=((C4C5z-C7T1z)^2)^0.5
CCEA=math.acos((b11/a11))
CCEA=math.deg(CCEA)

if C4C5z < C7T1z then print("CCEA angle is ", 180-CCEA) outputs.set("CCEA", 180-CCEA) else
print("CCEA angle is ", CCEA) outputs.set("CCEA", CCEA) end


--[[CERVICAL ANGLE3 (SCEA) ]]--

a12=((C4C5y-C7T1y)^2+(C4C5x-C7T1x)^2)^0.5
b12=((C4C5x-C7T1x)^2)^0.5
SCEA=math.acos((b12/a12))
SCEA=math.deg(SCEA)

if C4C5x < C7T1x then print("SCEA angle is ", 180-SCEA) outputs.set("SCEA", 180-SCEA) else
print("SCEA angle is ", SCEA)  outputs.set("SCEA", SCEA)end
```

```
--[[ Calculation of 3CTA angle]]--
a1=((C4C5x-C7T1x)^2+(C4C5y-C7T1y)^2+(C4C5z-C7T1z)^2)^0.5
b1=((C7T1x-T7T8x)^2+(C7T1y-T7T8y)^2+(C7T1z-T7T8z)^2)^0.5
c1=((C4C5x-T7T8x)^2+(C4C5x-T7T8x)^2+(C4C5x-T7T8x)^2)^0.5
C3TA=math.acos((c1^2-a1^2-b1^2)/(-2*a1*b1))
print("3CTA angle is ",C3TA)
C3TA=math.deg(C3TA)
outputs.set("CTA", C3TA)
--[[ Calculation of CCTA angle]]--
aa2=((C4C5x-C7T1x)^2+(C4C5y-C7T1y)^2)^0.5
ba2=((C7T1x-T7T8x)^2+(C7T1y-T7T8y)^2)^0.5
ca2=((C4C5x-T7T8x)^2+(C4C5x-T7T8x)^2)^0.5
CCTA=math.acos((ca2^2-aa2^2-ba2^2)/(-2*aa2*ba2))
print("CCTA angle is ",CCTA)
SCTA=math.deg(CCTA)
outputs.set("CCTA", CCTA)
--[[ Calculation of SCTA angle]]--
a2=((C4C5x-C7T1x)^2+(C4C5y-C7T1y)^2)^0.5
b2=((C7T1x-T7T8x)^2+(C7T1y-T7T8y)^2)^0.5
c2=((C4C5x-T7T8x)^2+(C4C5x-T7T8x)^2)^0.5
SCTA=math.acos((c2^2-a2^2-b2^2)/(-2*a2*b2))
print("SCTA angle is ",SCTA)
SCTA=math.deg(SCTA)
outputs.set("SCTA", SCTA)

--[[UPPER THORACIC ANGLE1 (UTA) ]]--

a13=((C7T1x-T7T8x)^2+(C7T1y-T7T8y)^2+(C7T1z-T7T8z)^2)^0.5
b13=((C7T1x-T7T8x)^2+(C7T1z-T7T8z)^2)^0.5
U3TA=math.acos((b13/a13))
UTA=math.deg(U3TA)
if  C7T1x >T7T8x and C7T1z >T7T8z then print("AR -quardrant 1", U3TA) outputs.set("UTA", UTA) end
if C7T1x >T7T8x and C7T1z < T7T8z then print("AL -quardrant 2",U3TA) outputs.set("UTA", 180-UTA)
end
if C7T1x < T7T8x and C7T1z <T7T8z then print("PL -quardrant 3",U3TA) outputs.set("UTA", 180+UTA)
end
if C7T1x < T7T8x and C7T1z >T7T8z then print("AL -quardrant 4", U3TA) outputs.set("UTA", 360-
UTA)end


--[[UPPER THORACIC ANGLE2 (CUTA) ]]--

a14=((C7T1y-T7T8y)^2+(C7T1z-T7T8z)^2)^0.5
b14=((C7T1z-T7T8z)^2)^0.5
CUTA=math.acos((b14/a14))
CUTA=math.deg(CUTA)

if C7T1z < T7T8z then print("CUTA angle is ", 180-CUTA) else print("CUTA angle is ", CUTA) end
outputs.set("CUTA", CUTA)

--[[UPPER THORACIC ANGLE3 (SUTA) ]]--
a15=((C7T1x-T7T8x)^2+(C7T1y-T7T8y)^2)^0.5
b15=((C7T1x-T7T8x)^2)^0.5
SUTA=math.acos((b15/a15))
SUTA=math.deg(SUTA)
if C7T1x < T7T8x then print("SUTA angle is ", 180-SUTA) outputs.set("SUTA", 180-SUTA) else
print("SUTA angle is ", SUTA) outputs.set("SUTA", SUTA) end

if T12L1x ~=nil and T12L1y ~=nil  then
--[[ Calculation of THA angle]]--
```

```
a3=((C7T1x-T7T8x)^2+(C7T1y-T7T8y)^2+(C7T1z-T7T8z)^2)^0.5
b3=((T7T8x-T12L1x)^2+(T7T8y-T12L1y)^2+(T7T8z-T12L1z)^2)^0.5
c3=((C7T1x-T12L1x)^2+(C7T1y-T12L1y)^2+(C7T1z-T12L1z)^2)^0.5
THA=math.acos((c3^2-a3^2-b3^2)/(-2*a3*b3))
THA=math.deg(THA)
print("THA angle is ",THA)
outputs.set("THA", THA)
--[[ Calculation of STHA angle]]--
a4=((C7T1x-T7T8x)^2+(C7T1y-T7T8y)^2)^0.5
b4=((T7T8x-T12L1x)^2+(T7T8y-T12L1y)^2)^0.5
c4=((C7T1x-T12L1x)^2+(C7T1y-T12L1y)^2)^0.5
STHA=math.acos((c4^2-a4^2-b4^2)/(-2*a4*b4))
STHA=math.deg(STHA)
print("STHA angle is ",STHA)
TK=(360-(2*STHA))*0.5
print("TK angle is ",TK)
outputs.set("STHA", STHA)
outputs.set("TK", TK)

--[[LOWER THORACIC ANGLE1 (LTA) ]]--
a16=((T7T8x-T12L1x)^2+(T7T8y-T12L1y)^2+(T7T8z-T12L1z)^2)^0.5
b16=((T7T8x-T12L1x)^2+(T7T8z-T12L1z)^2)^0.5
L3TA=math.acos((b16/a16))
LTA=math.deg(L3TA)
if T7T8x >T12L1x and T7T8z >T12L1z then print("AR -quardrant 1", L3TA) end
if T7T8x >T12L1x and T7T8z < T12L1z then print("AL -quardrant 2",L3TA) end
if T7T8x <T12L1x and T7T8z <T12L1z then print("PL -quardrant 3",L3TA) end
if T7T8x <T12L1x and T7T8z >T12L1z then print("AL -quardrant 4", L3TA) end
outputs.set("LTA", LTA)

--[[LOWER THORACIC ANGLE2 (CLTA) ]]--
a17=((T7T8y-T12L1y)^2+(T7T8z-T12L1z)^2)^0.5
b17=((T7T8z-T12L1z)^2)^0.5
CLTA=math.acos((b17/a17))
CLTA=math.deg(CLTA)
if T7T8z < T12L1z then print("CLTA angle is ", 180-CLTA) else print("CLTA angle is ", CLTA) end
outputs.set("CLTA", CLTA)

--[[SAGITTAL LOWER THORACIC ANGLE3 (SLTA) ]]--
a18=((T7T8x-T12L1x)^2+(T7T8y-T12L1y)^2)^0.5
b18=((T7T8x-T12L1x)^2)^0.5
SLTA=math.acos((b18/a18))
SLTA=math.deg(SLTA)
if T7T8x < T12L1x then print("SLTA angle is ", 180-SLTA) else print("SLTA angle is ", SLTA) end
outputs.set("SLTA", SLTA)

end
if L3L4x~=nil and L5S1x~=nil and S2S3x~=nil then
--[[ Calculation of 3TLA angle]]--
a5=((T7T8x-T12L1x)^2+(T7T8y-T12L1y)^2+(T7T8z-T12L1z)^2)^0.5
b5=((T12L1x-L3L4x)^2+(T12L1y-L3L4y)^2+(T12L1z-L3L4z)^2)^0.5
c5=((T7T8x-L3L4x)^2+(T7T8y-L3L4y)^2+(T7T8z-L3L4z)^2)^0.5
T3LA=math.acos((c5^2-a5^2-b5^2)/(-2*a5*b5))
TLA=math.deg(T3LA)
print("3TLA angle is ",TLA)
outputs.set("TLA", TLA)

--[[ Calculation of STLA angle]]--
a6=((T7T8x-T12L1x)^2+(T7T8y-T12L1y)^2)^0.5
b6=((T12L1x-L3L4x)^2+(T12L1y-L3L4y)^2)^0.5
```

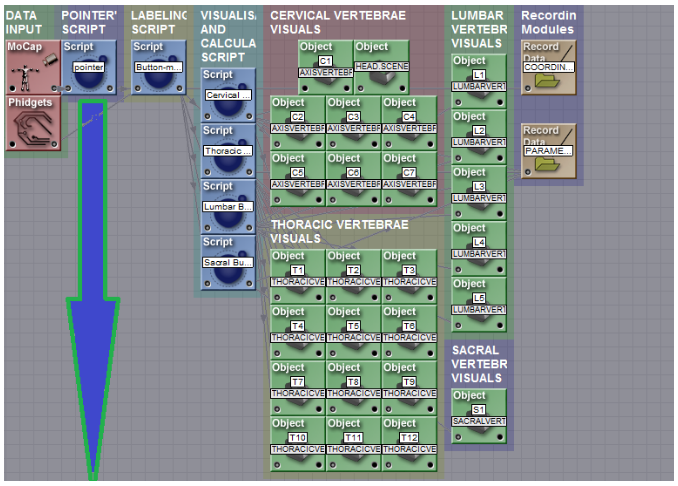
```
c6=((T7T8x-L3L4x)^2+(T7T8y-L3L4y)^2)^0.5
STLA=math.acos((c6^2-a6^2-b6^2)/(-2*a6*b6))
STLA=math.deg(STLA)
print("STLA angle is ",STLA)
outputs.set("STLA", STLA)
if T12L1x~=nil and T12L1y~=nil and L5S1x~=nil  and S2S3~=nil then
--[[UPPER LUMBAR ANGLE1 (ULA) ]]--
a19=((T12L1x-L3L4x)^2+(T12L1y-L3L4y)^2+(T12L1z-L3L4z)^2)^0.5
b19=((T12L1x-L3L4x)^2+(T12L1z-L3L4z)^2)^0.5
U3LA=math.acos((a19/b19))
ULA=math.deg(ULA)
if T12L1x > L3L4x and T12L1z > L3L4z then print("AR -quardrant 1", U3LA) end
if T12L1x > L3L4x and T12L1z < L3L4z then print("AL -quardrant 2",U3LA) end
if T12L1x < L3L4x and T12L1z < L3L4z then print("PL -quardrant 3",U3LA) end
if T12L1x < L3L4x and T12L1z > L3L4z then print("AL -quardrant 4", U3LA) end
outputs.set("ULA", ULA)

--[[CORONAL UPPER LUMBAR ANGLE1 (CULA) ]]--
a20=((T12L1z-L3L4z)^2)^0.5
b20=((T12L1y-L3L4y)^2+(T12L1z-L3L4z)^2)^0.5
CULA=math.acos((a20/b20))
CULA=math.deg(CULA)
if T12L1z < L3L4z then print("CULA angle is ", 180-CULA) else print("CULA angle is ", CULA) end
outputs.set("CULA", CULA)

--[[SAGITTAL UPPER LUMBAR ANGLE1 (SULA) ]]--
a21=((T12L1x-L3L4x)^2)^0.5
b21=((T12L1x-L3L4x)^2+(T12L1y-L3L4y)^2)^0.5
SULA=math.acos((a21/b21))
SULA=math.deg(SULA)
if T12L1x < L3L4x then print("SULA angle is ", 180-SULA) else print("CULA angle is ", SULA) end
outputs.set("SULA", SULA)

--[[Lumbar (lordisis) Angle (LMA) ]]--
a6=((T12L1x-L3L4x)^2+(T12L1y-L3L4y)^2+(T12L1z-L3L4z)^2)^0.5
b6=((L3L4x-L5S1x)^2+(L3L4y-L5S1y)^2+(L3L4z-L5S1z)^2)^0.5
c6=((T12L1x-L5S1x)^2+(T12L1y-L5S1y)^2+(T12L1z-L5S1z)^2)^0.5
LMA=math.acos((c6^2-a6^2-b6^2)/(-2*a6*b6))
LMA=math.deg(LMA)
print("LMA angle is ",LMA)
outputs.set("LMA", LMA)

--[[Sagittal LMA]]--
a7=((T12L1x-L3L4x)^2+(T12L1y-L3L4y)^2)^0.5
b7=((L3L4x-L5S1x)^2+(L3L4y-L5S1y)^2)^0.5
c7=((T12L1x-L5S1x)^2+(T12L1y-L5S1y)^2)^0.5
SLMA=math.acos((c7^2-a7^2-b7^2)/(-2*a7*b7))
SLMA=math.deg(SLMA)
print("SLMA angle is ",SLMA)
LL =((360-(2*SLMA)/2))
outputs.set("SLMA", SLMA)
outputs.set("LL", LL)

--[[LOWER LUMBAR ANGLE1 (LLA) ]]--
a22=((L3L4x-L5S1x)^2+(L3L4y-L5S1y)^2+(L3L4z-L5S1z)^2)^0.5
b22=((L3L4y-L5S1y)^2+(L3L4z-L5S1z)^2)^0.5
LLA=math.acos((a22/b22))
LLA=math.deg(LLA)
if L3L4x > L5S1x and L3L4z > L5S1z then print("AR -quardrant 1", LLA) end
if L3L4x > L5S1x and L3L4z < L5S1z then print("AL -quardrant 2",LLA) end
if L3L4x < L5S1x and L3L4z < L5S1z then print("PL -quardrant 3",LLA) end
if L3L4x < L5S1x and L3L4z > L5S1z then print("AL -quardrant 4", LLA) end
outputs.set("LLA", LLA)


--[[CORONAL LOWER LUMBAR ANGLE1 (CLLA) ]]--
a23=((L3L4z-L5S1z)^2)^0.5
b23=((L3L4y-L5S1y)^2+(L3L4z-L5S1z)^2)^0.5
CLLA=math.acos((a23/b23))
CLLA=math.deg(CLLA)
if L3L4z < L5S1z then print("CLLA angle is ", 180-CLLA) else print("CLLA angle is ", CLLA) end
outputs.set("CLLA", CLLA)

--[[SAGITTAL LOWER LUMBAR ANGLE1 (SLLA) ]]--
a24=((L3L4x-L5S1x)^2)^0.5
b24=((L3L4x-L5S1x)^2+(L3L4y-L5S1y)^2)^0.5
SLLA=math.acos((a24/b24))
SLLA=math.deg(SLLA)
if L3L4x < L5S1x then print("CULA angle is ", 180-CULA) else print("CULA angle is ", SLLA) end
outputs.set("SLLA", SLLA)

--[[SACRAL ANGLE (SCA) ]]--
a25=((L5S1x-S2S3x)^2+(L5S1y-S2S3y)^2+(L5S1z-S2S3z)^2)^0.5
b25=((L5S1x-S2S3x)^2+(L5S1z-S2S3z)^2)^0.5
SCA=math.acos((a25/b25))
SCA=math.deg(SCA)
if L5S1x > S2S3x and L5S1z > S2S3z then print("AR -quardrant 1", SCA) end
if L5S1x > S2S3x and L5S1z < S2S3z then print("AL -quardrant 2",SCA) end
if L5S1x < S2S3x and L5S1z < S2S3z then print("PL -quardrant 3",SCA) end
if L5S1x < S2S3x and L5S1z > S2S3z then print("AL -quardrant 4", SCA) end

outputs.set("SCA", SCA)

--[[CORONAL SACRAL ANGLE (CSCA) ]]--
a261=((L5S1z-S2S3z)^2)^0.5
b261=((L5S1z-S2S3z)^2+(L5S1y-S2S3y)^2)^0.5
CSCA=math.acos((a261/b261))
CSCA=math.deg(CSCA)
if L5S1z < S2S3z then print("CSCA angle is ", 180-CSCA) else print("CSCA angle is ", CSCA) end
outputs.set("CSCA", CSCA)

--[[SAGITTAL SACRAL ANGLE (SSCA) ]]--
a26=((L5S1x-S2S3x)^2)^0.5
b26=((L5S1x-S2S3x)^2+(L5S1y-S2S3y)^2)^0.5
SSCA=math.acos((a26/b26))
SSCA=math.deg(SSCA)
if L5S1x < S2S3x then print("SSCA angle is ", 180-SSCA) else print("CULA angle is ", SSCA) end
outputs.set("SSCA", SSCA)

--[[LUMBO-SACRAL ANGLE (LSA) ]]--
a8=((L3L4x-L5S1x)^2+(L3L4y-L5S1y)^2+(L3L4z-L5S1z)^2)^0.5
b8=((L5S1x-S2S3x)^2+(L5S1y-S2S3xy^2+(L5S1z-S2S3z)^2))^0.5
c8=((L3L4x-S2S3x)^2+(L3L4y-S2S3y)^2+(L3L4z-S2S3z)^2)^0.5
LSA=math.acos((c8^2-a8^2-b8^2)/(-2*a8*b8))
LSA=math.deg(LSA)
print("LSA angle is ",LSA)
outputs.set("LSA", LSA)

--[[Sagittal LSA]]--
a9=((L3L4x-L5S1x)^2+(L3L4y-L5S1y)^2)^0.5
```

```
b9=((L5S1x-S2S3x)^2+(L5S1y-S2S3xy^2))^0.5
c9=((L3L4x-S2S3x)^2+(L3L4y-S2S3y)^2)^0.5
SLSA=math.acos((c9^2-a9^2-b9^2)/(-2*a9*b9))
print("SLSA angle is ",SLSA)
outputs.set("SLSA", SLSA)


--[[POSTERIOR THORACIC ASSYMMETRY ANGLE (PTAA) ]]--
a27=((LTROz-RTROz)^2)^0.5
b27=((LTROx-RTROx)^2+(LTROz-RTROz)^2)^0.5
PTAA=math.acos((a27/b27))
PTAA=math.deg(PTAA)
outputs.set("PTAA", PTAA)


--[[POSTERIOR LUMBAR ASSYMMETRY ANGLE (PLAA) ]]--
a28=((LLROz-RLROz)^2)^0.5
b28=((LLROx-RLROx)^2+(LLROz-RLROz)^2)^0.5
PLAA=math.acos((a28/b28))
PLAA=math.deg(PLAA)
outputs.set("PLAA", PLAA)

--[[PERVIC TILT ANGLE (PRTA) ]]--
a29=((0.5*(LPS1x+C4C5x)-0.5*(RASIx +LASIx))^2)^0.5
b29=((0.5*(LPS1x+C4C5x)-0.5*(RASIx +LASIx))^2+(0.5*(LPS1y+C4C5y)-0.5*(RASIy +LASIy))^2)^0.5
PRTA=math.acos((a29/b29))
PRTA=math.deg(PRTA)
if (LPS1x+C4C5x) < (RASIx +LASIx) then print("PRTA angle is ", 180-PRTA) else print("PRTA angle is ",
PRTA) end
outputs.set("PRTA", PRTA)

--[[CERVIC VETEBRA 1 ANGLE (C1A) ]]--
a30=((0.5*(LTC1y+RTC1y)-0.5*(RPSIy +LPSIy))^2)^0.5
b30=((0.5*(LTC1x+RTC1x)-0.5*(LGTRx+RGTRx))^2+(0.5*(LTC1y+RTC1y)-0.5*(RPSIy
+LPSIy))^2+(0.5*(LTC1z+RTC1z)-0.5*(LGTRz+RGTRz))^2)^0.5
C1A=math.acos((a30/b30))
C1A=math.deg(C1A)
if 0.5*(LTC1x+RTC1x) > 0.5*(LGTRx+RGTRx)and 0.5*(LTC1z+RTC1z) > 0.5*(LGTR+RGTRz) then
print("AR -quardrant 1", C1A) end
if 0.5*(LTC1x+RTC1x) > 0.5*(LGTRx+RGTRx) and 0.5*(LTC1z+RTC1z) < 0.5*(LGTRz+RGTRz) then
print("AL -quardrant 2",C1A) end
if 0.5*(LTC1x+RTC1x) < 0.5*(LGTRx+RGTRx) and 0.5*(LTC1z+RTC1z) < 0.5*(LGTRz+RGTRz) then
print("PL -quardrant 3",C1A) end
if 0.5*(LTC1x+RTC1x) < 0.5*(LGTRx+RGTRx) and 0.5*(LTC1z+RTC1z) > 0.5*(LGTRz+RGTRz) then
print("AL -quardrant 4", C1A) end
outputs.set("C1A", C1A)

--[[L5S1 ANGLE (SS1A) ]]--
a31=((L5S1y-0.5*(LGTRy+RGTRy))^2)^0.5
b31=((L5S1x-0.5*(LGTRx+RGTRx))^2+(L5S1y-0.5*(RPSIy +LPSIy))^2+(L5S1z-
0.5*(LGTRz+RGTRz))^2)^0.5
L5S1A=math.acos((a31/b31))
L5S1A=math.deg(L5S1A)
if L5S1x > 0.5*(LGTRx+RGTRx)and L5S1z > 0.5*(LGTRz+RGTRz) then print("AR -quardrant 1", L5S1A)
end
if L5S1x > 0.5*(LGTRx+RGTRx) and L5S1z < 0.5*(LGTRz+RGTRz) then print("AL -quardrant 2",L5S1A)
end
if L5S1x < 0.5*(LGTRx+RGTRx) and L5S1z < 0.5*(LGTRz+RGTRz) then print("PL -quardrant 3",L5S1A)
end
```

```
if L5S1x < 0.5*(LGTRx+RGTRx) and L5S1z > 0.5*(LGTRz+RGTRz) then print("AL -quardrant 4", L5S1A)
end

outputs.set("SS1A", L5S1A)

--[[Approximate Spine length]]--
SL1=totaldist1+totaldist2+totaldist3+totaldist4+totaldist5+totaldist6
outputs.set("SL1", SL1)

--[[Approximate length from waist to C1]]--
SL2=totaldist1+totaldist2+totaldist3+totaldist4+totaldist5
outputs.set("SL2", SL2)

--[[Vertical height from C1 to S1]]--
VHS1C1=C1y-L5S1y
outputs.set("VHS1C1", VHS1C1)
end
end
end

end
```

MARKER INDEPENDENT APPLICATION

POINTER SCRIPT



```
o=o or 0

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

-- 3)Initialization code first frame ini==0
if ini == 0 then
c=c or {}
m1=m1 or {}
m2=m2 or {}
t=t or {}
label = label or {}
nlabel=nlabel or {}
length=length or {}
flength=flength or {}
mcount=mcount or {}

pointers=pointers or {}
pmx=pmx or {}
pmy=pmy or {}
pmz=pmz or {}

local allinputs={}
local alloutputs={}
for i=1 ,150 do allinputs[i]="input"..i end
inputs.setchannels(unpack(allinputs))
local alloutputs={}
for i=1 ,153 do alloutputs[i]="output"..i end
outputs.setchannels(unpack(alloutputs))

--
--load in pointer
--

infname='C:\\CAREN Resources\\Data\\envisage upper limb\\edwinpointer.txt'
io.input(infname)
nummarkers=io.read("*number")
numlengths=nummarkers*3

--Creates pointer markers--
pointers[1]=object.create("Cube", "Green")
```

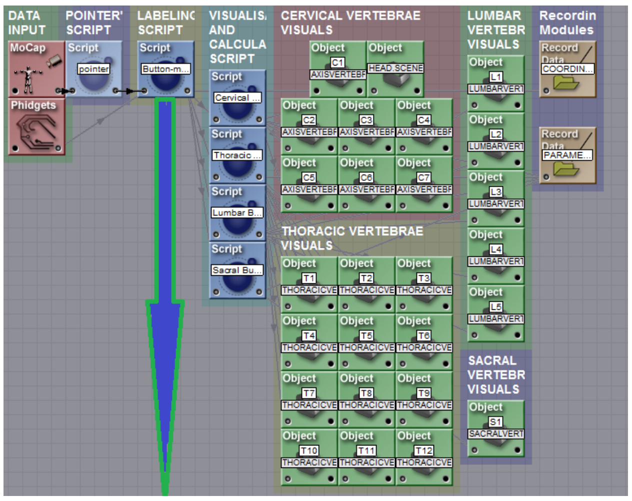
```
pointers[1]:setscaling(0.04,0.04,0.04)
pointers[1]:setposition(-999,-999,-999)

for i = 1 , numlengths do
c[i]=io.read("*number")
end
count=0
for i = 1 ,3 do
for j=i+1 , 4 do

--Calculating the lengths between pointer markers--
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
m1[count]=i
m2[count]=j
end
end
max=000
min=999
minlen=0
maxlen=0

for i=1 , count do
if length[i]<=min then
min=length[i]
minlen=i
end
if length[i]>=max then
max=length[i]
maxlen=i
end
end

for i=1, 4 do
label[i]=0
nlabel[i]=0
t[i]=0
end

--Calculating what markers give max/min lengths--
if m1[maxlen]==1 then t[1]=t[1]+1 end
if m1[maxlen]==2 then t[2]=t[2]+1 end
if m1[maxlen]==3 then t[3]=t[3]+1 end
if m1[maxlen]==4 then t[4]=t[4]+1 end
if m2[maxlen]==1 then t[1]=t[1]+1 end
if m2[maxlen]==2 then t[2]=t[2]+1 end
if m2[maxlen]==3 then t[3]=t[3]+1 end
if m2[maxlen]==4 then t[4]=t[4]+1 end
if m1[minlen]==1 then t[1]=t[1]+1 end
if m1[minlen]==2 then t[2]=t[2]+1 end
if m1[minlen]==3 then t[3]=t[3]+1 end
if m1[minlen]==4 then t[4]=t[4]+1 end
if m2[minlen]==1 then t[1]=t[1]+1 end
if m2[minlen]==2 then t[2]=t[2]+1 end
if m2[minlen]==3 then t[3]=t[3]+1 end
if m2[minlen]==4 then t[4]=t[4]+1 end
```

```
label[1]=0
if t[1]==2 then label[1]=1 end
if t[2]==2 then label[1]=2 end
if t[3]==2 then label[1]=3 end
if t[4]==2 then label[1]=4 end
nlabel[label[1]]=1
print("first marker is ",label[1])
t[1]=0
t[2]=0
t[3]=0
t[4]=0

if m2[minlen]==1 then t[1]=t[1]+1 end
if m2[minlen]==2 then t[2]=t[2]+1 end
if m2[minlen]==3 then t[3]=t[3]+1 end
if m2[minlen]==4 then t[4]=t[4]+1 end
if m1[minlen]==1 then t[1]=t[1]+1 end
if m1[minlen]==2 then t[2]=t[2]+1 end
if m1[minlen]==3 then t[3]=t[3]+1 end
if m1[minlen]==4 then t[4]=t[4]+1 end


t[label[1]]=0
label[2]=0
if t[1]==1 then label[2]=1 end
if t[2]==1 then label[2]=2 end
if t[3]==1 then label[2]=3 end
if t[4]==1 then label[2]=4 end
nlabel[label[2]]=1
print("second marker is ",label[2])
t[1]=0
t[2]=0
t[3]=0
t[4]=0

if m1[maxlen]==1 then t[1]=t[1]+1 end
if m1[maxlen]==2 then t[2]=t[2]+1 end
if m1[maxlen]==3 then t[3]=t[3]+1 end
if m1[maxlen]==4 then t[4]=t[4]+1 end
if m2[maxlen]==1 then t[1]=t[1]+1 end
if m2[maxlen]==2 then t[2]=t[2]+1 end
if m2[maxlen]==3 then t[3]=t[3]+1 end
if m2[maxlen]==4 then t[4]=t[4]+1 end


t[label[1]]=0
t[label[2]]=0
label[4]=0
if t[1]==1 then label[4]=1 end
if t[2]==1 then label[4]=2 end
if t[3]==1 then label[4]=3 end
if t[4]==1 then label[4]=4 end
nlabel[label[4]]=1
print("fourth marker is ",label[4])
if nlabel[1]==0 then label[3]=1 end
if nlabel[2]==0 then label[3]=2 end
if nlabel[3]==0 then label[3]=3 end
if nlabel[4]==0 then label[3]=4 end
```

```
print("third marker is ",label[3])


--Calculating the distances between markers 1-3 and 1&4--

flen1=(c[((label[1]-1)*3)+1]-c[((label[3]-1)*3)+1])^2
flen1=flen1+(c[((label[1]-1)*3)+2]-c[((label[3]-1)*3)+2])^2
flen1=flen1+(c[((label[1]-1)*3)+3]-c[((label[3]-1)*3)+3])^2
flen1=flen1^0.5
flen2=(c[((label[1]-1)*3)+1]-c[((label[2]-1)*3)+1])^2
flen2=flen2+(c[((label[1]-1)*3)+2]-c[((label[2]-1)*3)+2])^2
flen2=flen2+(c[((label[1]-1)*3)+3]-c[((label[2]-1)*3)+3])^2
flen2=flen2^0.5
flen3=(c[((label[2]-1)*3)+1]-c[((label[3]-1)*3)+1])^2
flen3=flen3+(c[((label[2]-1)*3)+2]-c[((label[3]-1)*3)+2])^2
flen3=flen3+(c[((label[2]-1)*3)+3]-c[((label[3]-1)*3)+3])^2
flen3=flen3^0.5
flen4=(c[((label[1]-1)*3)+1]-c[((label[4]-1)*3)+1])^2
flen4=flen4+(c[((label[1]-1)*3)+2]-c[((label[4]-1)*3)+2])^2
flen4=flen4+(c[((label[1]-1)*3)+3]-c[((label[4]-1)*3)+3])^2
flen4=flen4^0.5
ratio=flen4/flen1
print(flen4,flen1, ratio)

ini=1
end


for i=1,150 do
c[i]=inputs.get("input"..i)
outputs.set("output"..i,c[i])

end

for i=1,50 do
mcount[i]=0
end

for i=1,4 do
pmx[i]=999
pmy[i]=999
pmz[i]=999

end

for i = 1 ,49 do
for j=i+1 , 50 do
flength1=(c[((i-1)*3)+1]-c[((j-1)*3)+1])^2
flength1=flength1+(c[((i-1)*3)+2]-c[((j-1)*3)+2])^2
flength1=flength1+(c[((i-1)*3)+3]-c[((j-1)*3)+3])^2
flength1=flength1^0.5

if ((flen1-flength1)^2)^0.5 <=0.001 then
mcount[i]=mcount[i]+1
mcount[j]=mcount[j]+1
end
if  (((flen2-flength1)^2)^0.5) <=0.001 then
mcount[i]=mcount[i]+1
mcount[j]=mcount[j]+1
end
```

```
if ((flen3-flength1)^2)^0.5 <=0.001  then
mcount[i]=mcount[i]+1
mcount[j]=mcount[j]+1
end
end
end

pcount=1
for i=1,50 do
if mcount[i]>=2 then

pmx[pcount]=c[((i-1)*3)+1]
pmy[pcount]=c[((i-1)*3)+2]
pmz[pcount]=c[((i-1)*3)+3]
pcount=pcount+1
if pcount>=4 then pcount =3 end
end
end

if (pmx[1]^2+pmx[2]^2+pmx[3]^2)^0.5 > 0 and (pmx[1]^2+pmx[2]^2+pmx[3]^2)^0.5 < 999
then
pmx[4]=pmx[label[1]]+ratio*((pmx[label[3]])-(pmx[label[1]]))
pmy[4]=pmy[label[1]]+ratio*((pmy[label[3]])-(pmy[label[1]]))
pmz[4]=pmz[label[1]]+ratio*((pmz[label[3]])-(pmz[label[1]]))

if(pmx[4]^2+pmy[4]^2+pmz[4]^2)^0.5 >(pmx[3]^2+pmy[3]^2+pmz[3]^2)^0.5
then
pointers[1]:setposition(pmx[4],pmy[4],pmz[4])
end
if(pmx[4]^2+pmy[4]^2+pmz[4]^2)^0.5 <(pmx[3]^2+pmy[3]^2+pmz[3]^2)^0.5
then
pmx[4]=pmx[label[1]]+ratio*((pmx[label[1]])-(pmx[label[3]]))
pmy[4]=pmy[label[1]]+ratio*((pmy[label[1]])-(pmy[label[3]]))
pmz[4]=pmz[label[1]]+ratio*((pmz[label[1]])-(pmz[label[3]]))

pointers[1]:setposition(pmx[4],pmy[4],pmz[4])
end
end


outputs.set("output151",pmx[4])
outputs.set("output152",pmy[4])
outputs.set("output153",pmz[4])
```

LABELLING SCRIPT



```
--initialising the script and variables
bini =bini or 0
bini=0
if bini==0 then
ini=ini or 0
markers = markers or{}
allinputs = allinputs or {}
input = input or {}
indicator=indicator or {}
--creating or setting up output channels' names

outputs.setchannels("Headx","Heady","Headz","C1x","C1y","C1z","C2x","C2y","C2z","C3x","C3y","C3z","C4
x","C4y","C4z","C5x","C5y","C5z","C6x","C6y","C6z","C7x","C7y","C7z"
,"T1x","T1y","T1z","T2x","T2y","T2z","T3x","T3y","T3z","T4x","T4y","T4z","T5x","T5y","T5z","T6x","T6y
","T6z","T7x","T7y","T7z","T8x","T8y","T8z","T9x","T9y","T9z","T10x","T10y","T10z","T11x","T11y","T11
z","T12x","T12y","T12z",
"L1x","L1y","L1z","L2x","L2y","L2z","L3x","L3y","L3z","L4x","L4y","L4z","L5x","L5y","L5z","S1x","S1y",
"S1z","S2x","S2y","S2z")


if ini==0 then
    for i = 1, 154 do
        allinputs[i] = "Channel"..i
    end

    inputs.setchannels(unpack(allinputs))

--initialising variables
button1=0
button2=0
start=0
stop=0
count=0
value=-1
c=c or {}
input=input or {}
markername=markername or {}
markers = markers or{}
local allinputs={}

for i=1 ,154 do allinputs[i]="input"..i
```

```
end
inputs.setchannels(unpack(allinputs))
local alloutputs={}

--creating condition of the loop to run until the button is pressed 26 times
if value== -1 then
   start=1
   stop=26
   count=1
   else
   start=value
   stop=value
   count=value
   end
--initializing objects that will be set at differents points of visual marker's coordinates
coord=coord or  {}
coord[1]=object.create("Cube", "Red")
coord[1]:setscaling(0.02,0.02,0.02)
coord[1]:setposition(999,999,999)
coord[2]=object.create("Cube", "Red")
coord[2]:setscaling(0.02,0.02,0.02)
coord[2]:setposition(999,999,999)
coord[3]=object.create("Cube", "Red")
coord[3]:setscaling(0.02,0.02,0.02)
coord[3]:setposition(999,999,999)
coord[4]=object.create("Cube", "Red")
coord[4]:setscaling(0.02,0.02,0.02)
coord[4]:setposition(999,999,999)
coord[5]=object.create("Cube", "Red")
coord[5]:setscaling(0.02,0.02,0.02)
coord[5]:setposition(999,999,999)
coord[6]=object.create("Cube", "Red")
coord[6]:setscaling(0.02,0.02,0.02)
coord[6]:setposition(999,999,999)
coord[7]=object.create("Cube", "Red")
coord[7]:setscaling(0.02,0.02,0.02)
coord[7]:setposition(999,999,999)
coord[8]=object.create("Cube", "Blue")
coord[8]:setscaling(0.02,0.02,0.02)
coord[8]:setposition(999,999,999)
coord[9]=object.create("Cube", "Blue")
coord[9]:setscaling(0.02,0.02,0.02)
coord[9]:setposition(999,999,999)
coord[10]=object.create("Cube", "Blue")
coord[10]:setscaling(0.02,0.02,0.02)
coord[10]:setposition(999,999,999)
coord[11]=object.create("Cube", "Blue")
coord[11]:setscaling(0.02,0.02,0.02)
coord[11]:setposition(999,999,999)
coord[12]=object.create("Cube", "Blue")
coord[12]:setscaling(0.02,0.02,0.02)
coord[12]:setposition(999,999,999)
coord[13]=object.create("Cube", "Blue")
coord[13]:setscaling(0.02,0.02,0.02)
coord[13]:setposition(999,999,999)
coord[14]=object.create("Cube", "Blue")
coord[14]:setscaling(0.02,0.02,0.02)
coord[14]:setposition(999,999,999)
coord[15]=object.create("Cube", "Blue")
coord[15]:setscaling(0.02,0.02,0.02)
```

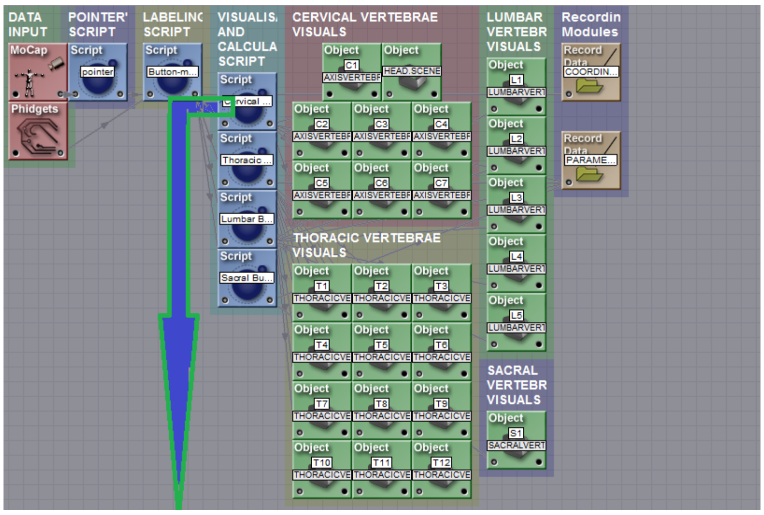
```
coord[15]:setposition(999,999,999)
coord[16]=object.create("Cube", "Blue")
coord[16]:setscaling(0.02,0.02,0.02)
coord[16]:setposition(999,999,999)
coord[17]=object.create("Cube", "Blue")
coord[17]:setscaling(0.02,0.02,0.02)
coord[17]:setposition(999,999,999)
coord[18]=object.create("Cube", "Blue")
coord[18]:setscaling(0.02,0.02,0.02)
coord[18]:setposition(999,999,999)
coord[19]=object.create("Cube", "Blue")
coord[19]:setscaling(0.02,0.02,0.02)
coord[19]:setposition(999,999,999)
coord[20]=object.create("Cube", "Green")
coord[20]:setscaling(0.02,0.02,0.02)
coord[20]:setposition(999,999,999)
coord[21]=object.create("Cube", "Green")
coord[21]:setscaling(0.02,0.02,0.02)
coord[21]:setposition(999,999,999)
coord[22]=object.create("Cube", "Green")
coord[22]:setscaling(0.02,0.02,0.02)
coord[22]:setposition(999,999,999)
coord[23]=object.create("Cube", "Green")
coord[23]:setscaling(0.02,0.02,0.02)
coord[23]:setposition(999,999,999)
coord[24]=object.create("Cube", "Green")
coord[24]:setscaling(0.02,0.02,0.02)
coord[24]:setposition(999,999,999)
coord[25]=object.create("Cube", "Red")
coord[25]:setscaling(0.02,0.02,0.02)
coord[25]:setposition(999,999,999)
coord[26]=object.create("Cube", "Red")
coord[26]:setscaling(0.02,0.02,0.02)
coord[26]:setposition(999,999,999)
coord[27]=object.create("Sphere", "Yellow")
coord[27]:setscaling(0.1,0.1,0.1)
coord[27]:setposition(999,999,999)
coord[28]=object.create("Cube", "Green")
coord[28]:setscaling(0.02,0.02,0.02)
coord[28]:setposition(999,999,999)
ini=1
end
--initialising array of inputs values
for i = 1, 154 do
   input[i] = inputs.get("input"..i)
end


if value~= -1 then
   start=value
   stop=value
   count=value
end

if value~=0 then

if count<=stop then
   button2=inputs.get("input154")
   if button2 == 0
   then button2=0
   else button2=1
   end

if button2==1 then
   mmx = 151
                    mmy = 152
                    mmz = 153
   if button1==0 then
      button1=1
--capturing coordinates of the virtual marker & setting up values of output channels --
   print("store data",input[1],input[mmx], count)

if count==1 and input[mmx] ~=999 and input[mmy] ~=999 and input[mmz] ~=999 then
C1x=input[mmx]
C1y=input[mmy]
C1z=input[mmz]
outputs.set("C1x",input[mmx])
outputs.set("C1y",input[mmy])
outputs.set("C1z",input[mmz])
outputs.set("Headx",input[mmx])
outputs.set("Heady",input[mmy]+0.28)
outputs.set("Headz",input[mmz])
coord[27]:setposition(input[mmx]-0.04,input[mmy]+0.04,input[mmz])
coord[1]:setposition(input[mmx],input[mmy],input[mmz])
count=2
end

if count==2 and input[mmx] ~=999 and input[mmy] ~=999 and input[mmz] ~=999 then
C2x=input[mmx]
C2y=input[mmy]
C2z=input[mmz]
outputs.set("C2x",input[mmx])
outputs.set("C2y",input[mmy])
outputs.set("C2z",input[mmz])
coord[2]:setposition(input[mmx],input[mmy],input[mmz])
end

if count==3 and input[mmx] ~=999 and input[mmy] ~=999 and input[mmz] ~=999 then
C3x=input[mmx]
C3y=input[mmy]
C3z=input[mmz]
outputs.set("C3x",input[mmx])
outputs.set("C3y",input[mmy])
outputs.set("C3z",input[mmz])
coord[3]:setposition(input[mmx],input[mmy],input[mmz])
end

if count==4 and input[mmx] ~=999 and input[mmy] ~=999 and input[mmz] ~=999 then
C4x=input[mmx]
C4y=input[mmy]
C4z=input[mmz]
outputs.set("C4x",input[mmx])
outputs.set("C4y",input[mmy])
outputs.set("C4z",input[mmz])
coord[4]:setposition(input[mmx],input[mmy],input[mmz])
end
if count==5 and input[mmx] ~=999 and input[mmy] ~=999 and input[mmz] ~=999 then
C5x=input[mmx]
C5y=input[mmy]
C5z=input[mmz]
```

```
outputs.set("C5x",input[mmx])
outputs.set("C5y",input[mmy])
outputs.set("C5z",input[mmz])
coord[5]:setposition(input[mmx],input[mmy],input[mmz])
end
if count==6 and input[mmx] ~=999 and input[mmy] ~=999 and input[mmz] ~=999 then
C6x=input[mmx]
C6y=input[mmy]
C6z=input[mmz]
outputs.set("C6x",input[mmx])
outputs.set("C6y",input[mmy])
outputs.set("C6z",input[mmz])
coord[6]:setposition(input[mmx],input[mmy],input[mmz])
end
if count==7 and input[mmx] ~=999 and input[mmy] ~=999 and input[mmz] ~=999 then
C7x=input[mmx]
C7y=input[mmy]
C7z=input[mmz]
outputs.set("C7x",input[mmx])
outputs.set("C7y",input[mmy])
outputs.set("C7z",input[mmz])
coord[7]:setposition(input[mmx],input[mmy],input[mmz])
end
if count==8 and input[mmx] ~=999 and input[mmy] ~=999 and input[mmz] ~=999 then
T1x=input[mmx]
T1y=input[mmy]
T1z=input[mmz]
outputs.set("T1x",input[mmx])
outputs.set("T1y",input[mmy])
outputs.set("T1z",input[mmz])
coord[8]:setposition(input[mmx],input[mmy],input[mmz])
end

if count==9 and input[mmx] ~=999 and input[mmy] ~=999 and input[mmz] ~=999 then
T2x=input[mmx]
T2y=input[mmy]
T2z=input[mmz]
outputs.set("T2x",input[mmx])
outputs.set("T2y",input[mmy])
outputs.set("T2z",input[mmz])
coord[9]:setposition(input[mmx],input[mmy],input[mmz])
end
if count==10 and input[mmx] ~=999 and input[mmy] ~=999 and input[mmz] ~=999 then
T3x=input[mmx]
T3y=input[mmy]
T3z=input[mmz]
outputs.set("T3x",input[mmx])
outputs.set("T3y",input[mmy])
outputs.set("T3z",input[mmz])
coord[10]:setposition(input[mmx],input[mmy],input[mmz])
end
if count==11 and input[mmx] ~=999 and input[mmy] ~=999 and input[mmz] ~=999 then
T4x=input[mmx]
T4y=input[mmy]
T4z=input[mmz]
outputs.set("T4x",input[mmx])
outputs.set("T4y",input[mmy])
outputs.set("T4z",input[mmz])
coord[11]:setposition(input[mmx],input[mmy],input[mmz])
end
```

```
if count==12 and input[mmx] ~=999 and input[mmy] ~=999 and input[mmz] ~=999 then
T5x=input[mmx]
T5y=input[mmy]
T5z=input[mmz]
outputs.set("T5x",input[mmx])
outputs.set("T5y",input[mmy])
outputs.set("T5z",input[mmz])
coord[12]:setposition(input[mmx],input[mmy],input[mmz])
end

if count==13 and input[mmx] ~=999 and input[mmy] ~=999 and input[mmz] ~=999 then
T6x=input[mmx]
T6y=input[mmy]
T6z=input[mmz]
outputs.set("T6x",input[mmx])
outputs.set("T6y",input[mmy])
outputs.set("T6z",input[mmz])
coord[13]:setposition(input[mmx],input[mmy],input[mmz])
end
if count==14 and input[mmx] ~=999 and input[mmy] ~=999 and input[mmz] ~=999 then
T7x=input[mmx]
T7y=input[mmy]
T7z=input[mmz]
outputs.set("T7x",input[mmx])
outputs.set("T7y",input[mmy])
outputs.set("T7z",input[mmz])
coord[14]:setposition(input[mmx],input[mmy],input[mmz])
end
if count==15 and input[mmx] ~=999 and input[mmy] ~=999 and input[mmz] ~=999 then
T8x=input[mmx]
T8y=input[mmy]
T8z=input[mmz]
outputs.set("T8x",input[mmx])
outputs.set("T8y",input[mmy])
outputs.set("T8z",input[mmz])
coord[15]:setposition(input[mmx],input[mmy],input[mmz])
end
if count==16 and input[mmx] ~=999 and input[mmy] ~=999 and input[mmz] ~=999 then
T9x=input[mmx]
T9y=input[mmy]
T9z=input[mmz]
outputs.set("T9x",input[mmx])
outputs.set("T9y",input[mmy])
outputs.set("T9z",input[mmz])
coord[16]:setposition(input[mmx],input[mmy],input[mmz])
end
if count==17 and input[mmx] ~=999 and input[mmy] ~=999 and input[mmz] ~=999 then
T10x=input[mmx]
T10y=input[mmy]
T10z=input[mmz]
outputs.set("T10x",input[mmx])
outputs.set("T10y",input[mmy])
outputs.set("T10z",input[mmz])
coord[17]:setposition(input[mmx],input[mmy],input[mmz])
end
if count==18 and input[mmx] ~=999 and input[mmy] ~=999 and input[mmz] ~=999 then
T11x=input[mmx]
T11y=input[mmy]
T11z=input[mmz]
outputs.set("T11x",input[mmx])
outputs.set("T11y",input[mmy])
outputs.set("T11z",input[mmz])
coord[18]:setposition(input[mmx],input[mmy],input[mmz])
end
if count==19  and input[mmx] ~=999 and input[mmy] ~=999 and input[mmz] ~=999 then
T12x=input[mmx]
T12y=input[mmy]
T12z=input[mmz]
outputs.set("T12x",input[mmx])
outputs.set("T12y",input[mmy])
outputs.set("T12z",input[mmz])
coord[19]:setposition(input[mmx],input[mmy],input[mmz])
end
if count==20 and input[mmx] ~=999 and input[mmy] ~=999 and input[mmz] ~=999 then
L1x=input[mmx]
L1y=input[mmy]
L1z=input[mmz]
outputs.set("L1x",input[mmx])
outputs.set("L1y",input[mmy])
outputs.set("L1z",input[mmz])
coord[20]:setposition(input[mmx],input[mmy],input[mmz])
end
if count==21 and input[mmx] ~=999 and input[mmy] ~=999 and input[mmz] ~=999 then
L2x=input[mmx]
L2y=input[mmy]
L2z=input[mmz]
outputs.set("L2x",input[mmx])
outputs.set("L2y",input[mmy])
outputs.set("L2z",input[mmz])

coord[21]:setposition(input[mmx],input[mmy],input[mmz])
end
if count==22 and input[mmx] ~=999 and input[mmy] ~=999 and input[mmz] ~=999 then
L3x=input[mmx]
L3y=input[mmy]
L3z=input[mmz]
outputs.set("L3x",input[mmx])
outputs.set("L3y",input[mmy])
outputs.set("L3z",input[mmz])
coord[22]:setposition(input[mmx],input[mmy],input[mmz])
end
if count==23 and input[mmx] ~=999 and input[mmy] ~=999 and input[mmz] ~=999 then
L4x=input[mmx]
L4y=input[mmy]
L4z=input[mmz]
outputs.set("L4x",input[mmx])
outputs.set("L4y",input[mmy])
outputs.set("L4z",input[mmz])
coord[23]:setposition(input[mmx],input[mmy],input[mmz])
end
if count==24 and input[mmx] == 999 and input[mmy] == 999 and input[mmz] ==999 then
count=24 value =24
print ("no wand inside capture volume")
end
if count==24 and input[mmx] ~=999 and input[mmy] ~=999 and input[mmz] ~=999 then
L5x=input[mmx]
L5y=input[mmy]
L5z=input[mmz]
outputs.set("L5x",input[mmx])
outputs.set("L5y",input[mmy])
outputs.set("L5z",input[mmz])
coord[24]:setposition(input[mmx],input[mmy],input[mmz])
end
if count==25 and input[mmx] ~=999 and input[mmy] ~=999 and input[mmz] ~=999 then
S1x=input[mmx]
S1y=input[mmy]
S1z=input[mmz]
outputs.set("S1x",input[mmx])
outputs.set("S1y",input[mmy])
outputs.set("S1z",input[mmz])
coord[25]:setposition(input[mmx],input[mmy],input[mmz])
end
if count==26 and input[mmx] ~=999 and input[mmy] ~=999 and input[mmz] ~=999 then
S2x=input[mmx]
S2y=input[mmy]
S2z=input[mmz]
outputs.set("S2x",input[mmx])
outputs.set("S2y",input[mmy])
outputs.set("S2z",input[mmz])
coord[26]:setposition(input[mmx],input[mmy],input[mmz])
end

    print("store data",mmx, count)
count=count+1
    end
end
end
end
end
--resetting values of button variables--

  if button2==0 then
    if button1==1 then
      print("reset")
      button1=0
      end
    end
```

VISUALISATION AND CALCULATION SCRIPT MODULES

CERVICAL SCRIPT MODULE



```
-- Initilisation of variables
ini = ini or 0
allinputs = allinputs or {}
nrInputs = 270
markercoord=markercoord or {}
pmx=pmx or {}
pmy=pmy or {}
pmz=pmz or {}


-- Initialisation code
if ini == 0 then
   for i = 1, 270 do
      allinputs[i] = "input"..i

   end
   inputs.setchannels(unpack(allinputs))

outputs.setchannels("marx", "mary", "marz","rotx", "roty", "rotz", "marsx", "marsy", "marsz","rotsx", "rotsy",
"rotsz","Headx","Heady","Headz","rotHeadx","rotHeady","rotHeadz","C1x","C1y","C1z","rotC1x","rotC1y","r
otC1z","C2x","C2y","C2z","rotC2x","rotC2y","rotC2z","C3x","C3y","C3z","rotC3x","rotC3y","rotC3z","C4x",
"C4y","C4z","rotC4x","rotC4y","rotC4z","C5x","C5y","C5z","rotC5x","rotC5y","rotC5z","C6x","C6y","C6z","
rotC6x","rotC6y","rotC6z","C7x","C7y","C7z","rotC7x","rotC7y","rotC7z"
,"objscalex","objscaley","objscalez","objscale2x","objscale2y","objscale2z","T1x","T1y","T1z","T2x","T2y","T
2z","T3x","T3y","T3z","T4x","T4y","T4z","T5x","T5y","T5z","T6x","T6y","T6z","T7x","T7y","T7z","T8x","
T8y","T8z","T9x","T9y","T9z","T10x","T10y","T10z","T11x","T11y","T11z","T12x","T12y","T12z",
"L1x","L1y","L1z","L2x","L2y","L2z","L3x","L3y","L3z","L4x","L4y","L4z","L5x","L5y","L5z","S1x","S1y",
"S1z","S2x","S2y","S2z", "C3EA","C3EA2","CCEA","SCEA","CTA","CCTA","SCTA")


--[[Objects creation]]--


ini = 1
end

for j=1,270 do
pmx[j]=999
pmy[j]=999
```

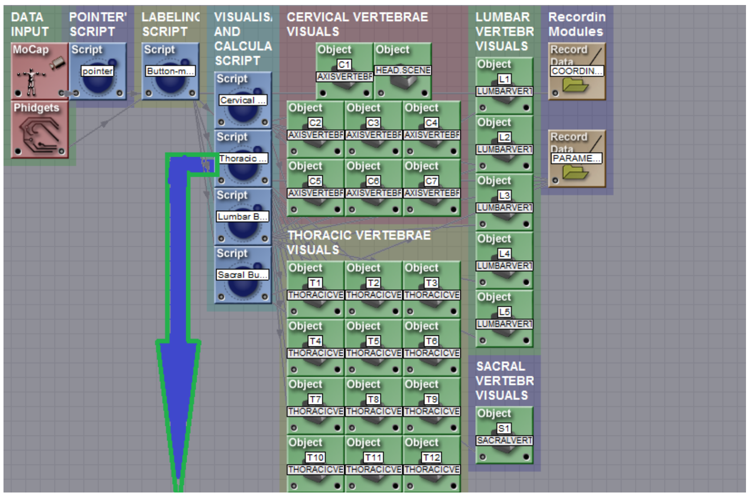
```
pmz[j]=999

end

-- script update
i = 1
j = 1
markers = {}
for i =1,270 do -- this creates an array of marker data, j = the marker number
    markers[i] = {}
    markers[i] = inputs.get("input"..i)
pmx=pmx or {}
pmy=pmy or {}
pmz=pmz or {}
pmx[1]=markers[1]
pmy[1]=markers[2]
pmz[1]=markers[3]
pmx[2]=markers[4]
pmy[2]=markers[5]
pmz[2]=markers[6]



m=2

--[[Identifying channels]]--
Headx=markers[1]
Heady=markers[2]
Headz=markers[3]
C1x=markers[4]
C1y=markers[5]
C1z=markers[6]
C2x=markers[7]
C2y=markers[8]
C2z=markers[9]
C3x=markers[10]
C3y=markers[11]
C3z=markers[12]
C4x=markers[13]
C4y=markers[14]
C4z=markers[15]
C5x=markers[16]
C5y=markers[17]
C5z=markers[18]
C6x=markers[19]
C6y=markers[20]
C6z=markers[21]
C7x=markers[22]
C7y=markers[23]
C7z=markers[24]
T1x=markers[25]
T1y=markers[26]
T1z=markers[27]
T2x=markers[28]
T2y=markers[29]
T2z=markers[30]
T3x=markers[31]
T3y=markers[32]
T3z=markers[33]
T4x=markers[34]
```

```
T4y=markers[35]
T4z=markers[36]
T5x=markers[37]
T5y=markers[38]
T5z=markers[39]
T6x=markers[40]
T6y=markers[41]
T6z=markers[42]
T7x=markers[43]
T7y=markers[44]
T7z=markers[45]
T8x=markers[46]
T8y=markers[47]
T8z=markers[48]
T9x=markers[49]
T9y=markers[50]
T9z=markers[51]
T10x=markers[52]
T10y=markers[53]
T10z=markers[54]
T11x=markers[55]
T11y=markers[56]
T11z=markers[57]
T12x=markers[58]
T12y=markers[59]
T12z=markers[60]
L1x=markers[61]
L1y=markers[62]
L1z=markers[63]
L2x=markers[64]
L2y=markers[65]
L2z=markers[66]
L3x=markers[67]
L3y=markers[68]
L3z=markers[69]
L4x=markers[70]
L4y=markers[71]
L4z=markers[72]
L5x=markers[73]
L5y=markers[74]
L5z=markers[75]
S1x=markers[76]
S1y=markers[77]
S1z=markers[78]
S2x=markers[79]
S2y=markers[80]
S2z=markers[81]
outputs.set("objscalex", 0.5)
outputs.set("objscaley", 0.5)
outputs.set("objscalez", 0.5)
outputs.set("objscale2x", 0.15)
outputs.set("objscale2y", 0.15)
outputs.set("objscale2z", 0.15)
i = i+1
end
--setting up outputs
if Headx== 0 and Heady==0 and Headz==0 then

outputs.set("Headx", 999)
outputs.set("Heady",999)
outputs.set("Headz",999)
outputs.set("rotHeadx", 45)
outputs.set("rotHeady", 0)
outputs.set("rotHeadz", 0)
end
if
Headx~= 0 and Heady~=0 and Headz~=0 then


outputs.set("Headx", Headx-0.041)
outputs.set("Heady", Heady-0.23)
outputs.set("Headz", Headz)
end
if C1x== 0 and C1y==0 and C1z==0 then
outputs.set("C1x", 999)
outputs.set("C1y",999)
outputs.set("C1z",999)
outputs.set("rotC1x", 45)
outputs.set("rotC1y", 0)
outputs.set("rotC1z", 0)
end
if
C1x~= 0 and C1y~=0 and C1z~=0 then
outputs.set("C1x", C1x)
outputs.set("C1y", C1y)
outputs.set("C1z", C1z)
thetax= (180*math.atan((Heady-C1y)/(Headz-C1z+0.00001)))/3.14
thetay= (180*math.atan((Headz-C1z)/(Headx-C1x+0.00001)))/3.14
thetaz= (180*math.atan((Heady-C1y)/(Headx-C1x+0.00001)))/3.14
print(thetax,thetay,thetaz)
outputs.set("rotHeadx", 90-thetaz)
outputs.set("rotHeady", thetay)
outputs.set("rotHeadz",90-thetax)
end
if C2x== 0 and C2y==0 and C2z==0 then
outputs.set("C2x", 999)
outputs.set("C2y",999)
outputs.set("C2z",999)
outputs.set("rotC2x", 45)
outputs.set("rotC2y", 0)
outputs.set("rotC2z", 0)
end

if C2x~= 0 and C2y~=0 and C2z~=0 then
outputs.set("C2x", C2x)
outputs.set("C2y", C2y)
outputs.set("C2z", C2z)
theta2x= (180*math.atan((C1y-C2y)/(C1z-C2z+0.00001)))/3.14
theta2y= (180*math.atan((C1z-C2z)/(C1x-C2x+0.00001)))/3.14
theta2z= (180*math.atan((C1y-C2y)/(C1x-C2x+0.00001)))/3.14

outputs.set("rotC1x", 0)
outputs.set("rotC1y",-90)
outputs.set("rotC1z",0)
end
if C3x== 0 and C3y==0 and C3z==0 then
outputs.set("C3x", 999)
outputs.set("C3y",999)
outputs.set("C3z",999)
outputs.set("rotC2x", 45)
outputs.set("rotC2y", 0)
outputs.set("rotC2z", 0)
end

if C3x~= 0 and C3y~=0 and C3z~=0 then
outputs.set("C3x", C3x)
outputs.set("C3y", C3y)
outputs.set("C3z", C3z)
theta3x= (180*math.atan((C1y-C2y)/(C1z-C2z+0.00001)))/3.14
theta3y= (180*math.atan((C1z-C2z)/(C1x-C2x+0.00001)))/3.14
theta3z= (180*math.atan((C1y-C2y)/(C1x-C2x+0.00001)))/3.14
outputs.set("rotC2x", 0)
outputs.set("rotC2y",-90)
outputs.set("rotC2z",0)
outputs.set("rotC3x", 0)
outputs.set("rotC3y",-90)
outputs.set("rotC3z",0)
end
if C4x== 0 and C4y==0 and C4z==0 then
outputs.set("C4x", 999)
outputs.set("C4y",999)
outputs.set("C4z",999)
outputs.set("rotC4x", 45)
outputs.set("rotC4y", 0)
outputs.set("rotC4z", 0)
end

if C4x~= 0 and C4y~=0 and C4z~=0 then
outputs.set("C4x", C4x)
outputs.set("C4y", C4y)
outputs.set("C4z", C4z)
theta3x= (180*math.atan((C1y-C2y)/(C1z-C2z+0.00001)))/3.14
theta3y= (180*math.atan((C1z-C2z)/(C1x-C2x+0.00001)))/3.14
theta3z= (180*math.atan((C1y-C2y)/(C1x-C2x+0.00001)))/3.14
outputs.set("rotC4x", 0)
outputs.set("rotC4y",-90)
outputs.set("rotC4z",0)

end
if C5x== 0 and C5y==0 and C5z==0 then
outputs.set("C5x", 999)
outputs.set("C5y",999)
outputs.set("C5z",999)
outputs.set("rotC5x", 45)
outputs.set("rotC5y", 0)
outputs.set("rotC5z", 0)
end
if C4x~= 0 and C4y~=0 and C4z~=0 then
outputs.set("C4x", C4x)
outputs.set("C4y", C4y)
outputs.set("C4z", C4z)
theta3x= (180*math.atan((C1y-C2y)/(C1z-C2z+0.00001)))/3.14
theta3y= (180*math.atan((C1z-C2z)/(C1x-C2x+0.00001)))/3.14
theta3z= (180*math.atan((C1y-C2y)/(C1x-C2x+0.00001)))/3.14
outputs.set("rotC4x", 0)
outputs.set("rotC4y",-90)
outputs.set("rotC4z",0)

end
if C5x== 0 and C5y==0 and C5z==0 then
```

```
outputs.set("C5x", 999)
outputs.set("C5y",999)
outputs.set("C5z",999)
outputs.set("rotC5x", 45)
outputs.set("rotC5y", 0)
outputs.set("rotC5z", 0)
end

if C5x~= 0 and C5y~=0 and C5z~=0 then
outputs.set("C5x", C5x)
outputs.set("C5y", C5y)
outputs.set("C5z", C5z)
theta3x= (180*math.atan((C1y-C2y)/(C1z-C2z+0.00001)))/3.14
theta3y= (180*math.atan((C1z-C2z)/(C1x-C2x+0.00001)))/3.14
theta3z= (180*math.atan((C1y-C2y)/(C1x-C2x+0.00001)))/3.14
outputs.set("rotC5x", 0)
outputs.set("rotC5y",-90)
outputs.set("rotC5z",0)

end
if C6x== 0 and C6y==0 and C6z==0 then
outputs.set("C6x", 999)
outputs.set("C6y",999)
outputs.set("C6z",999)
outputs.set("rotC6x", 45)
outputs.set("rotC6y", 0)
outputs.set("rotC6z", 0)
end

if C6x~= 0 and C6y~=0 and C6z~=0 then
outputs.set("C6x", C6x)
outputs.set("C6y", C6y)
outputs.set("C6z", C6z)
theta3x= (180*math.atan((C1y-C2y)/(C1z-C2z+0.00001)))/3.14
theta3y= (180*math.atan((C1z-C2z)/(C1x-C2x+0.00001)))/3.14
theta3z= (180*math.atan((C1y-C2y)/(C1x-C2x+0.00001)))/3.14
outputs.set("rotC6x", 0)
outputs.set("rotC6y",-90)
outputs.set("rotC6z",0)

end

if C7x== 0 and C7y==0 and C7z==0 then
outputs.set("C7x", 999)
outputs.set("C7y",999)
outputs.set("C7z",999)
outputs.set("rotC7x", 45)
outputs.set("rotC7y", 0)
outputs.set("rotC7z", 0)
end

if C7x~= 0 and C7y~=0 and C7z~=0 then
outputs.set("C7x", C7x)
outputs.set("C7y", C7y)
outputs.set("C7z", C7z)
theta3x= (180*math.atan((C1y-C2y)/(C1z-C2z+0.00001)))/3.14
theta3y= (180*math.atan((C1z-C2z)/(C1x-C2x+0.00001)))/3.14
theta3z= (180*math.atan((C1y-C2y)/(C1x-C2x+0.00001)))/3.14
outputs.set("rotC7x", 0)
outputs.set("rotC7y",-90)
outputs.set("rotC7z",0)

end




--Calculations
--[[CERVICAL ANGLE (CEA) ]]--

if C4x==0 and C4y==0 and C4z==0 and C7x ==0 and C7y==0 and C7z ==0 and T7x ==0 and T7y == 0 and
T7z ==0 then
outputs.set("C3EA", 0)
outputs.set("C3EA2", 0)
outputs.set("CCEA", 0)
outputs.set("SCEA", 0)
outputs.set("CTA", 0)
outputs.set("SCTA",0)
else if C4x~=0 and C4y~=0 and C4z~=0 and C7x ~=0 and C7y~=0 and C7z ~=0 and T7x ~= 0 and T7y ~= 0
and T7z ~= 0 then
a10=((C4x-C7x)^2+(C4y-C7y)^2+(C4z-C7z)^2)^0.5
b10=((C4x-C7x)^2+(C4z-C7z)^2)^0.5
CEA=math.acos((b10/a10))
a10=((C1x-C4x)^2+(C1y-C4y)^2+(C1z-C4z)^2)^0.5
b10=((C1x-C4x)^2+(C1z-C4z)^2)^0.5
CEA2=math.acos((b10/a10))
CEA2=math.deg(CEA2)

CEA=math.deg(CEA)
if C4x > C7x and C4z < C7z then print("AR -quardrant 1", CEA) outputs.set("C3EA", CEA) end
if C4x > C7x and C4z > C7z then print("AL -quardrant 2",CEA) outputs.set("C3EA", 180-CEA) end
if C4x < C7x and C4z < C7z then print("PL -quardrant 3",CEA) outputs.set("C3EA", 180+CEA) end
if C4x < C7x and C4z > C7z then print("AL -quardrant 4", CEA) outputs.set("C3EA", 360-CEA) end


--[[CERVICAL ANGLE2 (CCEA) ]]--
a11=((C4y-C7y)^2+(C4z-C7z)^2)^0.5
b11=((C4z-C7z)^2)^0.5
CCEA=math.acos((b11/a11))
CCEA=math.deg(CCEA)
if C4z < C7z then print("CCEA angle is ", 180-CCEA) outputs.set("CCEA", 180-CCEA) else print("CCEA
angle is ", CCEA) outputs.set("CCEA", CCEA) end


--[[CERVICAL ANGLE3 (SCEA) ]]--

a12=((C4y-C7y)^2+(C4x-C7x)^2)^0.5
b12=((C4x-C7x)^2)^0.5
SCEA=math.acos((b12/a12))
SCEA=math.deg(SCEA)
if C4x < C7x then print("SCEA angle is ", 180-SCEA) outputs.set("SCEA",180-SCEA)else print("SCEA
angle is ", SCEA) outputs.set("SCEA", SCEA) end

--[[ Calculation of 3CTA angle]]--
a1=((C4x-C7x)^2+(C4y-C7y)^2+(C4z-C7z)^2)^0.5
b1=((C7x-T7x)^2+(C7y-T7y)^2+(C7z-T7z)^2)^0.5
c1=((C4x-T7x)^2+(C4y-T7y)^2+(C4z-T7z)^2)^0.5
C3TA=math.acos((c1^2-a1^2-b1^2)/(-2*a1*b1))
print("3CTA angle is ",C3TA)
C3TA=math.deg(C3TA)
```

```
outputs.set("CTA", C3TA)
--[[ Calculation of CCTA angle]]--
a2=((C4z-C7z)^2+(C4y-C7y)^2)^0.5
b2=((C7z-T7z)^2+(C7y-T7y)^2)^0.5
c2=((C4z-T7z)^2+(C4y-T7y)^2)^0.5
CCTA=math.acos((c2^2-a2^2-b2^2)/(-2*a2*b2))
print("CCTA angle is ",CCTA)
SCTA=math.deg(CCTA)
outputs.set("CCTA", CCTA)
--[[ Calculation of SCTA angle]]--
a2=((C4x-C7x)^2+(C4y-C7y)^2)^0.5
b2=((C7x-T7x)^2+(C7y-T7y)^2)^0.5
c2=((C4x-T7x)^2+(C4y-T7y)^2)^0.5
SCTA=math.acos((c2^2-a2^2-b2^2)/(-2*a2*b2))
print("SCTA angle is ",SCTA)
SCTA=math.deg(SCTA)
outputs.set("SCTA", SCTA)



end

end
```

THORACIC MODULE



```
-- Initilisation of variables
ini = ini or 0
allinputs = allinputs or {}
nrInputs = 270
markercoord=markercoord or {}
pmx=pmx or {}
pmy=pmy or {}
pmz=pmz or {}


-- Initialisation code
if ini == 0 then
   for i = 1, 270 do
      allinputs[i] = "input"..i
```

```
    end
    inputs.setchannels(unpack(allinputs))

outputs.setchannels("marx", "mary", "marz","rotx", "roty", "rotz", "marsx", "marsy", "marsz","rotsx", "rotsy",
"rotsz","Headx","Heady","Headz","C1x","C1y","C1z","C2x","C2y","C2z","C3x","C3y","C3z","C4x","C4y","C
4z","C5x","C5y","C5z","C6x","C6y","C6z","C7x","C7y","C7z"
,"T1x","T1y","T1z","rotT1x","rotT1y","rotT1z","T2x","T2y","T2z","rotT2x","rotT2y","rotT2z","T3x","T3y","
T3z","rotT3x","rotT3y","rotT3z","T4x","T4y","T4z","rotT4x","rotT4y","rotT4z","T5x","T5y","T5z","rotT5x","
rotT5y","rotT5z","T6x","T6y","T6z","rotT6x","rotT6y","rotT6z","T7x","T7y","T7z","rotT7x","rotT7y","rotT7z
","T8x","T8y","T8z","rotT8x","rotT8y","rotT8z","T9x","T9y","T9z","rotT9x","rotT9y","rotT9z","T10x","T10y
","T10z","rotT10x","rotT10y","rotT10z","T11x","T11y","T11z","rotT11x","rotT11y","rotT11z","T12x","T12y"
,"T12z","rotT12x","rotT12y","rotT12z","objscalex","objscaley","objscalez",
"L1x","L1y","L1z","L2x","L2y","L2z","L3x","L3y","L3z","L4x","L4y","L4z","L5x","L5y","L5z","S1x","S1y",
"S1z","S2x","S2y","S2z","UTA", "CUTA","SUTA",
"THA","STHA","LTA","CLTA","SLTA","TLA","STLA")


--[[Objects creation]]--


ini = 1
end

for j=1,270 do
pmx[j]=999
pmy[j]=999
pmz[j]=999

end

-- script update
i = 1
j = 1
markers = {}
for i =1,270 do -- this creates an array of marker data, j = the marker number
    markers[i] = {}
    markers[i] = inputs.get("input"..i)
pmx=pmx or {}
pmy=pmy or {}
pmz=pmz or {}
pmx[1]=markers[1]
pmy[1]=markers[2]
pmz[1]=markers[3]
pmx[2]=markers[4]
pmy[2]=markers[5]
pmz[2]=markers[6]



m=2

--[[Identifying channels]]--
Headx=markers[1]
Heady=markers[2]
Headz=markers[3]
C1x=markers[4]
C1y=markers[5]
C1z=markers[6]
C2x=markers[7]
C2y=markers[8]
```

```
C2z=markers[9]
C3x=markers[10]
C3y=markers[11]
C3z=markers[12]
C4x=markers[13]
C4y=markers[14]
C4z=markers[15]
C5x=markers[16]
C5y=markers[17]
C5z=markers[18]
C6x=markers[19]
C6y=markers[20]
C6z=markers[21]
C7x=markers[22]
C7y=markers[23]
C7z=markers[24]
T1x=markers[25]
T1y=markers[26]
T1z=markers[27]
T2x=markers[28]
T2y=markers[29]
T2z=markers[30]
T3x=markers[31]
T3y=markers[32]
T3z=markers[33]
T4x=markers[34]
T4y=markers[35]
T4z=markers[36]
T5x=markers[37]
T5y=markers[38]
T5z=markers[39]
T6x=markers[40]
T6y=markers[41]
T6z=markers[42]
T7x=markers[43]
T7y=markers[44]
T7z=markers[45]
T8x=markers[46]
T8y=markers[47]
T8z=markers[48]
T9x=markers[49]
T9y=markers[50]
T9z=markers[51]
T10x=markers[52]
T10y=markers[53]
T10z=markers[54]
T11x=markers[55]
T11y=markers[56]
T11z=markers[57]
T12x=markers[58]
T12y=markers[59]
T12z=markers[60]
L1x=markers[61]
L1y=markers[62]
L1z=markers[63]
L2x=markers[64]
L2y=markers[65]
L2z=markers[66]
L3x=markers[67]
L3y=markers[68]
```

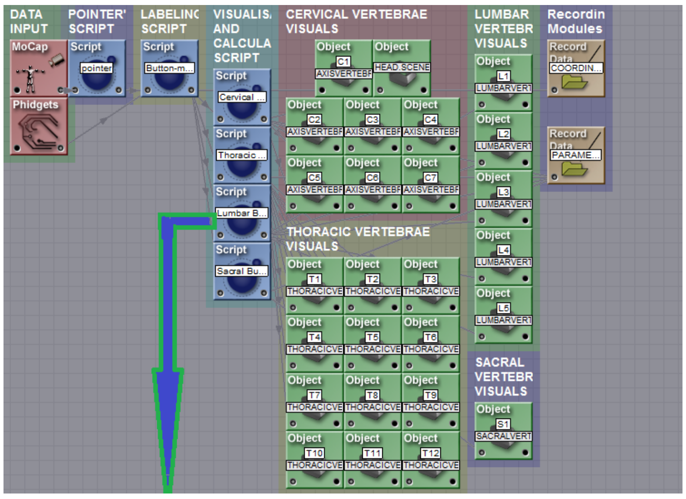
```
L3z=markers[69]
L4x=markers[70]
L4y=markers[71]
L4z=markers[72]
L5x=markers[73]
L5y=markers[74]
L5z=markers[75]
S1x=markers[76]
S1y=markers[77]
S1z=markers[78]
S2x=markers[79]
S2y=markers[80]
S2z=markers[81]
outputs.set("objscalex", 0.44)
outputs.set("objscaley", 0.44)
outputs.set("objscalez", 0.44)
i = i+1
end
--setting up outputs
if T1x== 0 and T1y==0 and T1z==0 then

outputs.set("T1x", 999)
outputs.set("T1y",999)
outputs.set("T1z",999)
outputs.set("rotT1x", 45)
outputs.set("rotT1y", 0)
outputs.set("rotT1z", 0)
end
if
T1x ~= 0 and T1y ~=0 and T1z ~=0 then
outputs.set("T1x", T1x)
outputs.set("T1y", T1y)
outputs.set("T1z", T1z)
outputs.set("rotT1x", 0)
outputs.set("rotT1y", -90)
outputs.set("rotT1z", 0)
end
if T2x== 0 and T2y==0 and T2z==0 then
outputs.set("T2x", 999)
outputs.set("T2y",999)
outputs.set("T2z",999)
outputs.set("rotT2x", 45)
outputs.set("rotT2y", 0)
outputs.set("rotT2z", 0)
end
if
T2x~= 0 and T2y~=0 and T2z~=0 then
outputs.set("T2x", T2x)
outputs.set("T2y", T2y)
outputs.set("T2z", T2z)
thetax= (180*math.atan((Heady-C1y)/(Headz-C1z+0.00001)))/3.14
thetay= (180*math.atan((Headz-C1z)/(Headx-C1x+0.00001)))/3.14
thetaz= (180*math.atan((Heady-C1y)/(Headx-C1x+0.00001)))/3.14
print(thetax,thetay,thetaz)
outputs.set("rotT2x", 0)
outputs.set("rotT2y", -90)
outputs.set("rotT2z",0)
end
if T3x== 0 and T3y==0 and T3z==0 then
outputs.set("T3x", 999)
outputs.set("T3y",999)
outputs.set("T3z",999)
outputs.set("rotT3x", 45)
outputs.set("rotT3y", 0)
outputs.set("rotT3z", 0)
end

if T3x~= 0 and T3y~=0 and T3z~=0 then
outputs.set("T3x", T3x)
outputs.set("T3y", T3y)
outputs.set("T3z", T3z)
theta2x= (180*math.atan((C1y-C2y)/(C1z-C2z+0.00001)))/3.14
theta2y= (180*math.atan((C1z-C2z)/(C1x-C2x+0.00001)))/3.14
theta2z= (180*math.atan((C1y-C2y)/(C1x-C2x+0.00001)))/3.14

outputs.set("rotT3x", 0)
outputs.set("rotT3y",-90)
outputs.set("rotT3z",0)
end
if T4x== 0 and T4y==0 and T4z==0 then
outputs.set("T4x", 999)
outputs.set("T4y",999)
outputs.set("T4z",999)
outputs.set("rotT4x", 45)
outputs.set("rotT4y", 0)
outputs.set("rotT4z", 0)
end

if T4x~= 0 and T4y~=0 and T4z~=0 then
outputs.set("T4x", T4x)
outputs.set("T4y", T4y)
outputs.set("T4z", T4z)
theta3x= (180*math.atan((C1y-C2y)/(C1z-C2z+0.00001)))/3.14
theta3y= (180*math.atan((C1z-C2z)/(C1x-C2x+0.00001)))/3.14
theta3z= (180*math.atan((C1y-C2y)/(C1x-C2x+0.00001)))/3.14
outputs.set("rotT4x", 0)
outputs.set("rotT4y",-90)
outputs.set("rotT4z",0)

end
if T5x== 0 and T5y==0 and T5z==0 then
outputs.set("T5x", 999)
outputs.set("T5y",999)
outputs.set("T5z",999)
outputs.set("rotT5x", 45)
outputs.set("rotT5y", 0)
outputs.set("rotT5z", 0)
end

if T5x~= 0 and T5y~=0 and T5z~=0 then
outputs.set("T5x", T5x)
outputs.set("T5y", T5y)
outputs.set("T5z", T5z)
theta3x= (180*math.atan((C1y-C2y)/(C1z-C2z+0.00001)))/3.14
theta3y= (180*math.atan((C1z-C2z)/(C1x-C2x+0.00001)))/3.14
theta3z= (180*math.atan((C1y-C2y)/(C1x-C2x+0.00001)))/3.14
outputs.set("rotT5x", 0)
outputs.set("rotT5y",-90)
outputs.set("rotT5z",0)
```

```
end
if T6x== 0 and T6y==0 and T6z==0 then
outputs.set("T6x", 999)
outputs.set("T6y",999)
outputs.set("T6z",999)
outputs.set("rotT6x", 45)
outputs.set("rotT6y", 0)
outputs.set("rotT6z", 0)
end
if T6x~= 0 and T6y~=0 and T6z~=0 then
outputs.set("T6x", T6x)
outputs.set("T6y", T6y)
outputs.set("T6z", T6z)
theta3x= (180*math.atan((C1y-C2y)/(C1z-C2z+0.00001)))/3.14
theta3y= (180*math.atan((C1z-C2z)/(C1x-C2x+0.00001)))/3.14
theta3z= (180*math.atan((C1y-C2y)/(C1x-C2x+0.00001)))/3.14
outputs.set("rotT6x", 0)
outputs.set("rotT6y",-90)
outputs.set("rotT6z",0)

end
if T7x== 0 and T7y==0 and T7z==0 then
outputs.set("T7x", 999)
outputs.set("T7y",999)
outputs.set("T7z",999)
outputs.set("rotT7x", 45)
outputs.set("rotT7y", 0)
outputs.set("rotT7z", 0)
end

if T7x~= 0 and T7y~=0 and T7z~=0 then
outputs.set("T7x", T7x)
outputs.set("T7y", T7y)
outputs.set("T7z", T7z)
theta3x= (180*math.atan((C1y-C2y)/(C1z-C2z+0.00001)))/3.14
theta3y= (180*math.atan((C1z-C2z)/(C1x-C2x+0.00001)))/3.14
theta3z= (180*math.atan((C1y-C2y)/(C1x-C2x+0.00001)))/3.14
outputs.set("rotT7x", 0)
outputs.set("rotT7y",-90)
outputs.set("rotT7z",0)

end
if T8x== 0 and T8y==0 and T8z==0 then
outputs.set("T8x", 999)
outputs.set("T8y",999)
outputs.set("T8z",999)
outputs.set("rotT8x", 45)
outputs.set("rotT8y", 0)
outputs.set("rotT8z", 0)
end

if T8x~= 0 and T8y~=0 and T8z~=0 then
outputs.set("T8x", T8x)
outputs.set("T8y", T8y)
outputs.set("T8z", T8z)
theta3x= (180*math.atan((C1y-C2y)/(C1z-C2z+0.00001)))/3.14
theta3y= (180*math.atan((C1z-C2z)/(C1x-C2x+0.00001)))/3.14
theta3z= (180*math.atan((C1y-C2y)/(C1x-C2x+0.00001)))/3.14
outputs.set("rotT8x", 0)
outputs.set("rotT8y",-90)
outputs.set("rotT8z",0)

end

if T9x== 0 and T9y==0 and T9z==0 then
outputs.set("T9x", 999)
outputs.set("T9y",999)
outputs.set("T9z",999)
outputs.set("rotT9x", 45)
outputs.set("rotT9y", 0)
outputs.set("rotT9z", 0)
end

if T9x~= 0 and T9y~=0 and T9z~=0 then
outputs.set("T9x", T9x)
outputs.set("T9y", T9y)
outputs.set("T9z", T9z)
theta3x= (180*math.atan((C1y-C2y)/(C1z-C2z+0.00001)))/3.14
theta3y= (180*math.atan((C1z-C2z)/(C1x-C2x+0.00001)))/3.14
theta3z= (180*math.atan((C1y-C2y)/(C1x-C2x+0.00001)))/3.14
outputs.set("rotT9x", 0)
outputs.set("rotT9y",-90)
outputs.set("rotT9z",0)

end
if T10x== 0 and T10y==0 and T10z==0 then
outputs.set("T10x", 999)
outputs.set("T10y",999)
outputs.set("T10z",999)
outputs.set("rotT10x", 45)
outputs.set("rotT10y", 0)
outputs.set("rotT10z", 0)
end

if T10x~= 0 and T10y~=0 and T10z~=0 then
outputs.set("T10x", T10x)
outputs.set("T10y", T10y)
outputs.set("T10z", T10z)
theta3x= (180*math.atan((C1y-C2y)/(C1z-C2z+0.00001)))/3.14
theta3y= (180*math.atan((C1z-C2z)/(C1x-C2x+0.00001)))/3.14
theta3z= (180*math.atan((C1y-C2y)/(C1x-C2x+0.00001)))/3.14
outputs.set("rotT10x", 0)
outputs.set("rotT10y",-90)
outputs.set("rotT10z",0)

end
if T11x== 0 and T11y==0 and T11z==0 then
outputs.set("T11x", 999)
outputs.set("T11y",999)
outputs.set("T11z",999)
outputs.set("rotT11x", 45)
outputs.set("rotT11y", 0)
outputs.set("rotT11z", 0)
end

if T11x~= 0 and T11y~=0 and T11z~=0 then
outputs.set("T11x", T11x)
outputs.set("T11y", T11y)
outputs.set("T11z", T11z)
theta3x= (180*math.atan((C1y-C2y)/(C1z-C2z+0.00001)))/3.14
```

```
theta3y= (180*math.atan((C1z-C2z)/(C1x-C2x+0.00001)))/3.14
theta3z= (180*math.atan((C1y-C2y)/(C1x-C2x+0.00001)))/3.14
outputs.set("rotT11x", 0)
outputs.set("rotT11y",-90)
outputs.set("rotT11z",0)

end
if T12x== 0 and T12y==0 and T12z==0 then
outputs.set("T12x", 999)
outputs.set("T12y",999)
outputs.set("T12z",999)
outputs.set("rotT10x", 45)
outputs.set("rotT10y", 0)
outputs.set("rotT10z", 0)
end

if T12x~= 0 and T12y~=0 and T12z~=0 then
outputs.set("T12x", T12x)
outputs.set("T12y", T12y)
outputs.set("T12z", T12z)
theta3x= (180*math.atan((C1y-C2y)/(C1z-C2z+0.00001)))/3.14
theta3y= (180*math.atan((C1z-C2z)/(C1x-C2x+0.00001)))/3.14
theta3z= (180*math.atan((C1y-C2y)/(C1x-C2x+0.00001)))/3.14
outputs.set("rotT12x", 0)
outputs.set("rotT12y",-90)
outputs.set("rotT12z",0)

end



--Calculations

--[[Caculations ]]---
if C7x ==0 and C7y ==0 and C7z==0 and T7x==0 and T7y==0 and T7z ==0 and T12x==0 and T12y==0
and T12z==0 and
L1x ==0 and L1y ==0 and L1z==0 and L3x==0 and L3y==0 and L3z ==0 and L4x==0 and L4y==0 and
L4z==0  and L5x==0 and L5y==0 and L5z==0 then
outputs.set("UTA", 0)
outputs.set("CUTA", 0)
outputs.set("SUTA", 0)
outputs.set("THA", 0)
outputs.set("STHA", 0)
outputs.set("LTA", 0)
outputs.set("CLTA", 0)
outputs.set("SLTA", 0)
outputs.set("TLA", 0)
outputs.set("STLA", 0)

else if C7x ~=0 and C7y ~=0 and C7z~=0 and T7x~=0 and T7y~=0 and T7z ~=0 and T12x~=0 and T12y~=0
and T12z~=0 and
L1x ~=0 and L1y ~=0 and L1z~=0 and L3x~=0 and L3y~=0 and L3z ~=0 and L4x~=0 and L4y~=0 and
L4z~=0  and L5x~=0 and L5y~=0 and L5z~=0 then

--[[UPPER THORACIC ANGLE1 (UTA) ]]--
a13=((C7x-T7x)^2+(C7y-T7y)^2+(C7z-T7z)^2)^0.5
b13=((C7x-T7x)^2+(C7z-T7z)^2)^0.5
U3TA=math.acos((b13/a13))
UTA=math.deg(U3TA)
if  C7x >T7x and C7z >T7z then print("AR -quardrant 1", U3TA) outputs.set("UTA", UTA)  end
if C7x >T7x and C7z < T7z then print("AL -quardrant 2",U3TA) outputs.set("UTA", 180-UTA)  end
if C7x < T7x and C7z <T7z then print("PL -quardrant 3",U3TA) outputs.set("UTA", 180+ UTA) end
if C7x < T7x and C7z >T7z then print("AL -quardrant 4", U3TA) outputs.set("UTA", 360-UTA) end

--[[UPPER THORACIC ANGLE2 (CUTA) ]]--

a14=((C7y-T7y)^2+(C7z-T7z)^2)^0.5
b14=((C7z-T7z)^2)^0.5
CUTA=math.acos((b14/a14))
CUTA=math.deg(CUTA)
if C7z < T7z then print("CUTA angle is ", 180-CUTA) outputs.set("CUTA", 180-CUTA) else print("CUTA
angle is ", CUTA)  outputs.set("CUTA",CUTA) end

--[[UPPER THORACIC ANGLE3 (SUTA) ]]--
a15=((C7x-T7x)^2+(C7y-T7y)^2)^0.5
b15=((C7x-T7x)^2)^0.5
SUTA=math.acos((b15/a15))
SUTA=math.deg(SUTA)
if C7x < T7x then print("SUTA angle is ", 180-SUTA) outputs.set("SUTA", 180-SUTA) else print("SUTA
angle is ", SUTA) outputs.set("SUTA", SUTA) end

--[[ Calculation of THA angle]]--
a3=((C7x-T7x)^2+(C7y-T7y)^2+(C7z-T7z)^2)^0.5
b3=((T7x-T12x)^2+(T7y-T12y)^2+(T7z-T12z)^2)^0.5
c3=((C7x-T12x)^2+(C7y-T12y)^2+(C7z-T12z)^2)^0.5
T3HA=math.acos((c3^2-a3^2-b3^2)/(-2*a3*b3))
THA=math.deg(T3HA)
print("THA angle is ",THA)
outputs.set("THA", THA)

--[[ Calculation of STHA angle]]--
a4=((C7x-T7x)^2+(C7y-T7y)^2)^0.5
b4=((T7x-T12x)^2+(T7y-T12y)^2)^0.5
c4=((C7x-T12x)^2+(C7y-T12y)^2)^0.5
STHA=math.acos((c4^2-a4^2-b4^2)/(-2*a4*b4))
STHA=math.deg(STHA)
print("STHA angle is ",STHA)
outputs.set("STHA", STHA)


--[[LOWER THORACIC ANGLE1 (LTA) ]]--
a16=((T7x-T12x)^2+(T7y-T12y)^2+(T7z-T12z)^2)^0.5
b16=((T7x-T12x)^2+(T7z-T12z)^2)^0.5
L3TA=math.acos((b16/a16))
LTA=math.deg(L3TA)
if T7x >T12x and T7z >T12z then print("AR -quardrant 1", L3TA) outputs.set("LTA", LTA) end
if T7x >T12x and T7z < T12z then print("AL -quardrant 2",L3TA) outputs.set("LTA", 180-LTA) end
if T7x <T12x and T7z <T12z then print("PL -quardrant 3",L3TA) outputs.set("LTA", 180+ LTA) end
if T7x <T12x and T7z >T12z then print("AL -quardrant 4", L3TA) outputs.set("LTA", 360-LTA) end

--[[LOWER THORACIC ANGLE2 (CLTA) ]]--
a17=((T7y-T12y)^2+(T7z-T12z)^2)^0.5
b17=((T7z-T12z)^2)^0.5
CLTA=math.acos((b17/a17))
CLTA=math.deg(CLTA)
if T7z < T12z then print("CLTA angle is ", 180-CLTA) outputs.set("CLTA", 180-CLTA) else print("CLTA
angle is ", CLTA) outputs.set("CLTA",CLTA) end

--[[LOWER THORACIC ANGLE3 (SLTA) ]]--
a18=((T7x-T12x)^2+(T7y-T12y)^22)^0.5
```

```
b18=((T7x-T12x)^2)^0.5
SLTA=math.acos((b18/a18))
SLTA=math.deg(SLTA)
if T7x < T12x then print("SLTA angle is ", 180-SLTA) outputs.set("SLTA", 180-SLTA)else print("SLTA
angle is ", SLTA) outputs.set("SLTA", SLTA)  end

--[[ Calculation of 3TLA angle]]--
if T7x ~=nil and T7y ~=nil and T7z~=nil and L1x~=nil and L1y~=nil and L1z ~=nil and L4x~=nil and
L4y~=nil and L4z~=nil then
a5=((T7x-L1x)^2+(T7y-L1y)^2+(T7z-L1z)^2)^0.5
b5=((L1x-L4x)^2+(L1y-L4y)^2+(L1z-L4z)^2)^0.5
c5=((T7x-L4x)^2+(T7y-L4y)^2+(T7z-L4z)^2)^0.5
T3LA=math.acos((c5^2-a5^2-b5^2)/(-2*a5*b5))
TLA=math.deg(T3LA)
print("3TLA angle is ",TLA)
outputs.set("TLA", TLA)

--[[ Calculation of STLA angle]]--
a6=((T7x-L1x)^2+(T7y-L1y)^2)^0.5
b6=((L1x-L4x)^2+(L1y-L4y)^2)^0.5
c6=((T7x-L4x)^2+(T7y-L4y)^2)^0.5
STLA=math.acos((c6^2-a6^2-b6^2)/(-2*a6*b6))
STLA=math.deg(STLA)
print("STLA angle is ",STLA)
outputs.set("STLA", STLA)

end
--[[888888888888888888888888888888888888888888888888888888888
   888888888888888888888888888888888888888888888888888888888]]--

end
end
--end of calculations
```

LUMBAR MODULE



```
-- Initilisation of variables
ini = ini or 0
allinputs = allinputs or {}
nrInputs = 270
markercoord=markercoord or {}
pmx=pmx or {}
pmy=pmy or {}
pmz=pmz or {}


-- Initialisation code
if ini == 0 then
   for i = 1, 270 do
       allinputs[i] = "input"..i

   end
   inputs.setchannels(unpack(allinputs))

outputs.setchannels("marx", "mary", "marz","rotx", "roty", "rotz", "marsx", "marsy", "marsz","rotsx", "rotsy",
"rotsz","Headx","Heady","Headz","C1x","C1y","C1z","C2x","C2y","C2z","C3x","C3y","C3z","C4x","C4y","C
4z","C5x","C5y","C5z","C6x","C6y","C6z","C7x","C7y","C7z"
,"T1x","T1y","T1z","T2x","T2y","T2z","T3x","T3y","T3z","T4x","T4y","T4z","T5x","T5y","T5z","T6x","T6y
","T6z","T7x","T7y","T7z","T8x","T8y","T8z","T9x","T9y","T9z","T10x","T10y","T10z","T11x","T11y","T11
z","T12x","T12y","T12z",
"L1x","L1y","L1z","rotL1x","rotL1y","rotL1z","L2x","L2y","L2z","rotL2x","rotL2y","rotL2z","L3x","L3y","L
3z","rotL3x","rotL3y","rotL3z","L4x","L4y","L4z","rotL4x","rotL4y","rotL4z","L5x","L5y","L5z","rotL5x","r
otL5y","rotL5z","objscalex","objscaley","objscalez","S1x","S1y","S1z","S2x","S2y","S2z"
,"ULA","SULA","CULA","LMA","SLMA","LL","LLA","CLLA","SLLA")


--[[Objects creation]]--


ini = 1
end

for j=1,270 do
pmx[j]=999
pmy[j]=999
pmz[j]=999
```

```
end

-- script update
i = 1
j = 1
markers = {}
for i =1,270 do -- this creates an array of marker data, j = the marker number
   markers[i] = {}
   markers[i] = inputs.get("input"..i)
pmx=pmx or {}
pmy=pmy or {}
pmz=pmz or {}
pmx[1]=markers[1]
pmy[1]=markers[2]
pmz[1]=markers[3]
pmx[2]=markers[4]
pmy[2]=markers[5]
pmz[2]=markers[6]



m=2

--[[Identifying channels]]--
Headx=markers[1]
Heady=markers[2]
Headz=markers[3]
C1x=markers[4]
C1y=markers[5]
C1z=markers[6]
C2x=markers[7]
C2y=markers[8]
C2z=markers[9]
C3x=markers[10]
C3y=markers[11]
C3z=markers[12]
C4x=markers[13]
C4y=markers[14]
C4z=markers[15]
C5x=markers[16]
C5y=markers[17]
C5z=markers[18]
C6x=markers[19]
C6y=markers[20]
C6z=markers[21]
C7x=markers[22]
C7y=markers[23]
C7z=markers[24]
T1x=markers[25]
T1y=markers[26]
T1z=markers[27]
T2x=markers[28]
T2y=markers[29]
T2z=markers[30]
T3x=markers[31]
T3y=markers[32]
T3z=markers[33]
T4x=markers[34]
T4y=markers[35]
T4z=markers[36]
```

```
T5x=markers[37]
T5y=markers[38]
T5z=markers[39]
T6x=markers[40]
T6y=markers[41]
T6z=markers[42]
T7x=markers[43]
T7y=markers[44]
T7z=markers[45]
T8x=markers[46]
T8y=markers[47]
T8z=markers[48]
T9x=markers[49]
T9y=markers[50]
T9z=markers[51]
T10x=markers[52]
T10y=markers[53]
T10z=markers[54]
T11x=markers[55]
T11y=markers[56]
T11z=markers[57]
T12x=markers[58]
T12y=markers[59]
T12z=markers[60]
L1x=markers[61]
L1y=markers[62]
L1z=markers[63]
L2x=markers[64]
L2y=markers[65]
L2z=markers[66]
L3x=markers[67]
L3y=markers[68]
L3z=markers[69]
L4x=markers[70]
L4y=markers[71]
L4z=markers[72]
L5x=markers[73]
L5y=markers[74]
L5z=markers[75]
S1x=markers[76]
S1y=markers[77]
S1z=markers[78]
S2x=markers[79]
S2y=markers[80]
S2z=markers[81]
outputs.set("objscalex", 0.44)
outputs.set("objscaley", 0.44)
outputs.set("objscalez", 0.44)
i = i+1
end
--setting up outputs
if L1x== 0 and L1y==0 and L1z==0 then

outputs.set("L1x", 999)
outputs.set("L1y",999)
outputs.set("L1z",999)
outputs.set("rotL1x", 45)
outputs.set("rotL1y", 0)
outputs.set("rotL1z", 0)
end
```

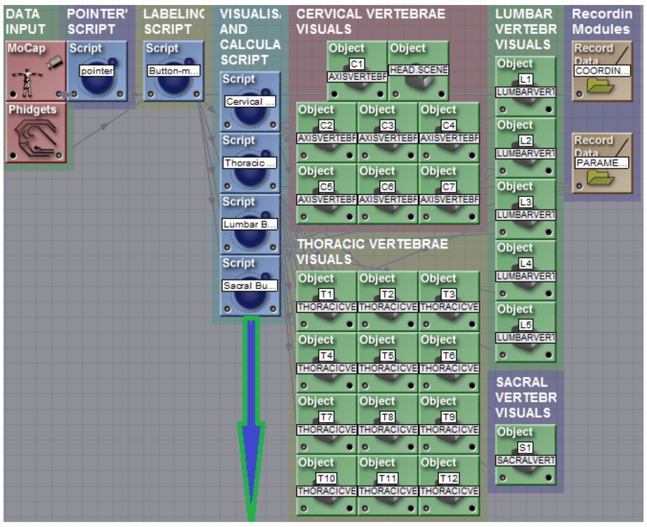
```
if
L1x ~= 0 and L1y ~=0 and L1z ~=0 then
outputs.set("L1x", L1x)
outputs.set("L1y", L1y)
outputs.set("L1z", L1z)
outputs.set("rotL1x", 0)
outputs.set("rotL1y", -90)
outputs.set("rotL1z", 0)
end
if L2x== 0 and L2y==0 and L2z==0 then
outputs.set("L2x", 999)
outputs.set("L2y",999)
outputs.set("L2z",999)
outputs.set("rotL2x", 45)
outputs.set("rotL2y", 0)
outputs.set("rotL2z", 0)
end
if
L2x~= 0 and L2y~=0 and L2z~=0 then
outputs.set("L2x", L2x)
outputs.set("L2y", L2y)
outputs.set("L2z", L2z)
thetax= (180*math.atan((Heady-C1y)/(Headz-C1z+0.00001)))/3.14
thetay= (180*math.atan((Headz-C1z)/(Headx-C1x+0.00001)))/3.14
thetaz= (180*math.atan((Heady-C1y)/(Headx-C1x+0.00001)))/3.14
print(thetax,thetay,thetaz)
outputs.set("rotL2x", 0)
outputs.set("rotL2y", -90)
outputs.set("rotL2z",0)
end
if L3x== 0 and L3y==0 and L3z==0 then
outputs.set("L3x", 999)
outputs.set("L3y",999)
outputs.set("L3z",999)
outputs.set("rotL3x", 45)
outputs.set("rotL3y", 0)
outputs.set("rotL3z", 0)
end

if L3x~= 0 and L3y~=0 and L3z~=0 then
outputs.set("L3x", L3x)
outputs.set("L3y", L3y)
outputs.set("L3z", L3z)
theta2x= (180*math.atan((C1y-C2y)/(C1z-C2z+0.00001)))/3.14
theta2y= (180*math.atan((C1z-C2z)/(C1x-C2x+0.00001)))/3.14
theta2z= (180*math.atan((C1y-C2y)/(C1x-C2x+0.00001)))/3.14

outputs.set("rotL3x", 0)
outputs.set("rotL3y",-90)
outputs.set("rotL3z",0)
end
if L4x== 0 and L4y==0 and L4z==0 then
outputs.set("L4x", 999)
outputs.set("L4y",999)
outputs.set("L4z",999)
outputs.set("rotL4x", 45)
outputs.set("rotL4y", 0)
outputs.set("rotL4z", 0)
end
```

```
if L4x~= 0 and L4y~=0 and L4z~=0 then
outputs.set("L4x", L4x)
outputs.set("L4y", L4y)
outputs.set("L4z", L4z)
theta3x= (180*math.atan((C1y-C2y)/(C1z-C2z+0.00001)))/3.14
theta3y= (180*math.atan((C1z-C2z)/(C1x-C2x+0.00001)))/3.14
theta3z= (180*math.atan((C1y-C2y)/(C1x-C2x+0.00001)))/3.14
outputs.set("rotL4x", 0)
outputs.set("rotL4y",-90)
outputs.set("rotL4z",0)

end
if L5x== 0 and L5y==0 and L5z==0 then
outputs.set("L5x", 999)
outputs.set("L5y",999)
outputs.set("L5z",999)
outputs.set("rotL5x", 45)
outputs.set("rotL5y", 0)
outputs.set("rotL5z", 0)
end

if L5x~= 0 and L5y~=0 and L5z~=0 then
outputs.set("L5x", L5x)
outputs.set("L5y", L5y)
outputs.set("L5z", L5z)
theta3x= (180*math.atan((C1y-C2y)/(C1z-C2z+0.00001)))/3.14
theta3y= (180*math.atan((C1z-C2z)/(C1x-C2x+0.00001)))/3.14
theta3z= (180*math.atan((C1y-C2y)/(C1x-C2x+0.00001)))/3.14
outputs.set("rotL5x", 0)
outputs.set("rotL5y",-90)
outputs.set("rotL5z",0)

end



--Calculations

if T12x==0 and T12y==0 and T12z==0 and L1x ==0 and L1y ==0 and L1z==0 and L3x==0 and L3y==0
and L3z ==0 and L4x==0 and L4y==0 and L4z==0  and L5x==0 and L5y==0 and L5z==0 then
outputs.set("ULA", 0)
outputs.set("CULA",0)
outputs.set("SULA",0)
outputs.set("LMA", 0)
outputs.set("SLMA",0)
outputs.set("LL",0)
outputs.set("LLA",0)
outputs.set("CLLA", 0)
outputs.set("SLLA",0)



else if T12x~=0 and T12y~=0 and T12z~=0 and L1x ~=0 and L1y ~=0 and L1z~=0 and L3x~=0 and L3y~=0
and L3z ~=0 and L4x~=0 and L4y~=0 and L4z~=0  and L5x~=0 and L5y~=0 and L5z~=0 then

--[[UPPER LUMBAR ANGLE1 (ULA) ]]--
a19=((T12x-L3x)^2+(T12y-L3y)^2+(T12z-L3z)^2)^0.5
b19=((T12x-L3x)^2+(T12z-L3z)^2)^0.5
U3LA=math.acos((a19/b19))
ULA=math.deg(U3LA)
```

```
if T12x > L3x and T12z > L3z then print("AR -quardrant 1", U3LA) outputs.set("ULA", ULA) end
if T12x > L3x and T12z < L3z then print("AL -quardrant 2",U3LA) outputs.set("ULA", 180-ULA) end
if T12x < L3x and T12z < L3z then print("PL -quardrant 3",U3LA) outputs.set("ULA", 180+ULA) end
if T12x < L3x and T12z > L3z then print("AL -quardrant 4", U3LA) outputs.set("ULA", 360-ULA) end

--[[CORONAL UPPER LUMBAR ANGLE1 (CULA) ]]--
a20=((T12z-L3z)^2)^0.5
b20=((T12y-L3y)^2+(T12z-L3z)^2)^0.5
CULA=math.acos((a20/b20))
CULA=math.deg(CULA)
if T12z < L3z then print("CULA angle is ", 180-CULA) outputs.set("CULA", 180-CULA)else print("CULA
angle is ", CULA) outputs.set("CULA", CULA) end

--[[SAGITTAL UPPER LUMBAR ANGLE1 (SULA) ]]--
a21=((T12x-L3x)^2)^0.5
b21=((T12x-L3x)^2+(T12y-L3y)^2)^0.5
SULA=math.acos((a21/b21))
SULA=math.deg(SULA)
if T12x < L3x then print("SULA angle is ", 180-SULA) outputs.set("SULA", 180-SULA)else print("SULA
angle is ", SULA) outputs.set("SULA", SULA) end

--[[Lumbar (lordisis) Angle (LMA) ]]--
if L1x~=nil and L1y~=nil and L1z~=nil and L3x ~=nil and L3y~=nil and L3z ~=nil and L5x~=nil and
L5y~=nil and L5z~=nil then
a6=((L1x-L3x)^2+(L1y-L3y)^2+(L1z-L3z)^2)^0.5
b6=((L3x-L5x)^2+(L3y-L5y)^2+(L3z-L5z)^2)^0.5
c6=((L1x-L5x)^2+(L1y-L5y)^2+(L1z-L5z)^2)^0.5
LMA=math.acos((c6^2-a6^2-b6^2)/(-2*a6*b6))
LMA=math.deg(LMA)
print("LMA angle is ",LMA)
outputs.set("LMA", LMA)

-- [[Sagittal lumbar lordosis angle]]--
a7=((L1x-L3x)^2+(L1y-L3y)^2)^0.5
b7=((L3x-L5x)^2+(L3y-L5y)^2)^0.5
c7=((L1x-L5x)^2+(L1y-L5y)^2)^0.5
SLMA=math.acos((c7^2-a7^2-b7^2)/(-2*a7*b7))
SLMA=math.deg(SLMA)
print("SLMA angle is ",SLMA)
outputs.set("SLMA", SLMA)

LL =((360-(2*SLMA))/2)
outputs.set("LL", LL)

--[[LOWER LUMBAR ANGLE1 (LLA) ]]--
a22=((L3x-L5x)^2+(L3y-L5y)^2+(L3z-L5z)^2)^0.5
b22=((L3y-L5y)^2+(L3z-L5z)^2)^0.5
LLA=math.acos((a22/b22))
LLA=math.deg(LLA)
if L3x > L5x and L3z > L5z then print("AR -quardrant 1", LLA)outputs.set("LLA", LLA) end
if L3x > L5x and L3z < L5z then print("AL -quardrant 2",LLA) outputs.set("LLA", 180-LLA) end
if L3x < L5x and L3z < L5z then print("PL -quardrant 3",LLA) outputs.set("LLA", 180+LLA) end
if L3x < L5x and L3z > L5z then print("AL -quardrant 4", LLA) outputs.set("LLA", 360-LLA) end

--[[CORONAL LOWER LUMBAR ANGLE1 (CLLA) ]]--
a23=((L3z-L5z)^2)^0.5
b23=((L3y-L5y)^2+(L3z-L5z)^2)^0.5
CLLA=math.acos((a23/b23))
CLLA=math.deg(CLLA)
```

```
if L3z < L5z then print("CLLA angle is ", 180-CLLA) outputs.set("CLLA", CLLA) else print("CLLA angle is
", CLLA) outputs.set("CLLA", CLLA) end

--[[SAGITTAL LOWER LUMBAR ANGLE1 (SLLA) ]]--
a24=((L3x-L5x)^2)^0.5
b24=((L3x-L5x)^2+(L3y-L5y)^2)^0.5
SLLA=math.acos((a24/b24))
SLLA=math.deg(SLLA)
if L3x < L5x then print("SLLA angle is ", 180-SLLA) outputs.set("SLLA", 180-SLLA) else print("SLLA
angle is ", SLLA) outputs.set("SLLA", SLLA) end
end

--end of calculations

end

end
```

SERVICAL MODULE



```
-- Initilisation of variables
ini = ini or 0
allinputs = allinputs or {}
nrInputs = 270
markercoord=markercoord or {}
pmx=pmx or {}
pmy=pmy or {}
pmz=pmz or {}


-- Initialisation code
if ini == 0 then
   for i = 1, 270 do
      allinputs[i] = "input"..i

   end
   inputs.setchannels(unpack(allinputs))

outputs.setchannels("marx", "mary", "marz","rotx", "roty", "rotz", "marsx", "marsy", "marsz","rotsx", "rotsy",
"rotsz","Headx","Heady","Headz","C1x","C1y","C1z","C2x","C2y","C2z","C3x","C3y","C3z","C4x","C4y","C
4z","C5x","C5y","C5z","C6x","C6y","C6z","C7x","C7y","C7z"
,"T1x","T1y","T1z","T2x","T2y","T2z","T3x","T3y","T3z","T4x","T4y","T4z","T5x","T5y","T5z","T6x","T6y
","T6z","T7x","T7y","T7z","T8x","T8y","T8z","T9x","T9y","T9z","T10x","T10y","T10z","T11x","T11y","T11
z","T12x","T12y","T12z",
"L1x","L1y","L1z","rotL1x","rotL1y","rotL1z","L2x","L2y","L2z","rotL2x","rotL2y","rotL2z","L3x","L3y","L
3z","rotL3x","rotL3y","rotL3z","L4x","L4y","L4z","rotL4x","rotL4y","rotL4z","L5x","L5y","L5z","rotL5x","r
otL5y","rotL5z","objscalex","objscaley","objscalez","S1x","S1y","S1z","rotS1x","rotS1y","rotS1z","S2x","S2y
","S2z","rotS2x","rotS2y","rotS2z"
,"LSA","SLSA","SCA","CSCA","SSCA","SL1","SL2","VHS1C1")


--[[Objects creation]]--


ini = 1
end

for j=1,270 do
pmx[j]=999
pmy[j]=999
pmz[j]=999

end

-- script update
i = 1
j = 1
markers = {}
for i =1,270 do -- this creates an array of marker data, j = the marker number
   markers[i] = {}
   markers[i] = inputs.get("input"..i)
pmx=pmx or {}
pmy=pmy or {}
pmz=pmz or {}
pmx[1]=markers[1]
pmy[1]=markers[2]
pmz[1]=markers[3]
pmx[2]=markers[4]
pmy[2]=markers[5]
pmz[2]=markers[6]



m=2

--[[Identifying channels]]--
Headx=markers[1]
Heady=markers[2]
Headz=markers[3]
C1x=markers[4]
C1y=markers[5]
C1z=markers[6]
C2x=markers[7]
C2y=markers[8]
C2z=markers[9]
C3x=markers[10]
C3y=markers[11]
C3z=markers[12]
C4x=markers[13]
C4y=markers[14]
C4z=markers[15]
```

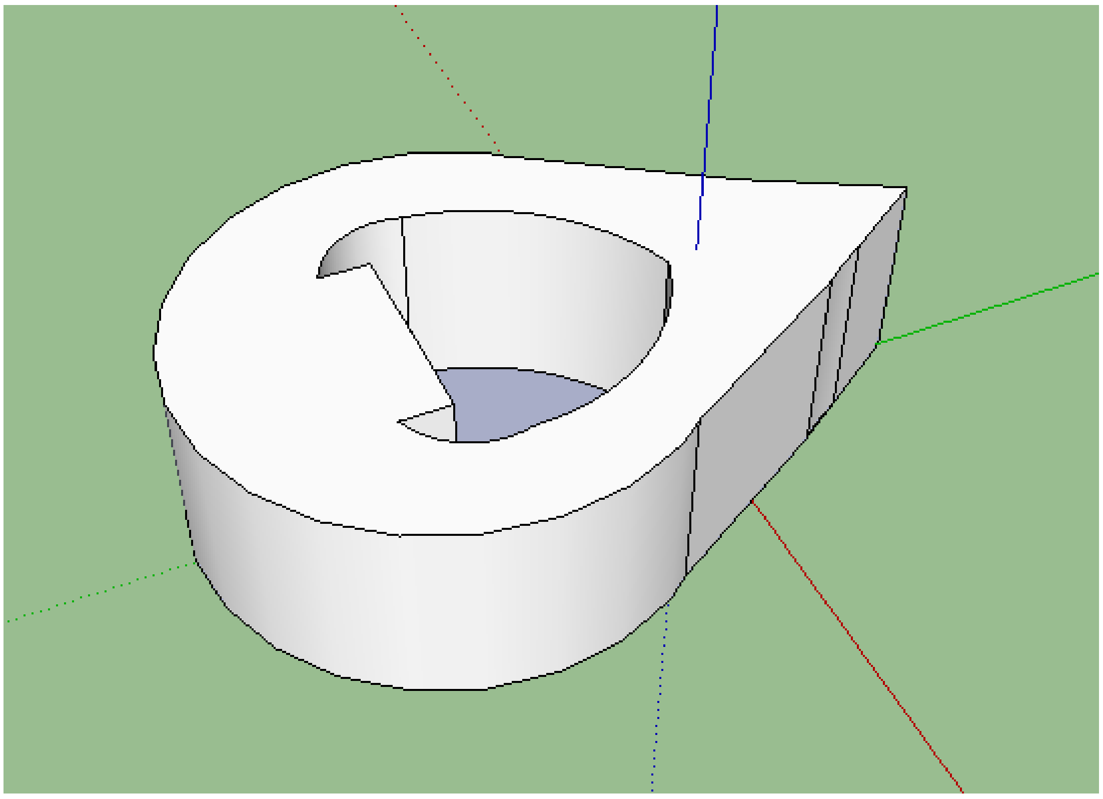
```
C5x=markers[16]
C5y=markers[17]
C5z=markers[18]
C6x=markers[19]
C6y=markers[20]
C6z=markers[21]
C7x=markers[22]
C7y=markers[23]
C7z=markers[24]
T1x=markers[25]
T1y=markers[26]
T1z=markers[27]
T2x=markers[28]
T2y=markers[29]
T2z=markers[30]
T3x=markers[31]
T3y=markers[32]
T3z=markers[33]
T4x=markers[34]
T4y=markers[35]
T4z=markers[36]
T5x=markers[37]
T5y=markers[38]
T5z=markers[39]
T6x=markers[40]
T6y=markers[41]
T6z=markers[42]
T7x=markers[43]
T7y=markers[44]
T7z=markers[45]
T8x=markers[46]
T8y=markers[47]
T8z=markers[48]
T9x=markers[49]
T9y=markers[50]
T9z=markers[51]
T10x=markers[52]
T10y=markers[53]
T10z=markers[54]
T11x=markers[55]
T11y=markers[56]
T11z=markers[57]
T12x=markers[58]
T12y=markers[59]
T12z=markers[60]
L1x=markers[61]
L1y=markers[62]
L1z=markers[63]
L2x=markers[64]
L2y=markers[65]
L2z=markers[66]
L3x=markers[67]
L3y=markers[68]
L3z=markers[69]
L4x=markers[70]
L4y=markers[71]
L4z=markers[72]
L5x=markers[73]
L5y=markers[74]
L5z=markers[75]
```

```
S1x=markers[76]
S1y=markers[77]
S1z=markers[78]
S2x=markers[79]
S2y=markers[80]
S2z=markers[81]
outputs.set("objscalex", 0.44)
outputs.set("objscaley", 0.44)
outputs.set("objscalez", 0.44)
i = i+1
end
--setting up outputs
if S1x== 0 and S1y==0 and S1z==0 then

outputs.set("S1x", 999)
outputs.set("S1y",999)
outputs.set("S1z",999)
outputs.set("rotS1x", 45)
outputs.set("rotS1y", 0)
outputs.set("rotS1z", 0)
end
if
S1x ~= 0 and S1y ~=0 and S1z ~=0 then
outputs.set("S1x", S1x)
outputs.set("S1y", S1y)
outputs.set("S1z", S1z)
outputs.set("rotS1x", 0)
outputs.set("rotS1y", 90)
outputs.set("rotS1z", 0)
end
if S2x== 0 and S2y==0 and S2z==0 then
outputs.set("S2x", 999)
outputs.set("S2y",999)
outputs.set("S2z",999)
outputs.set("rotS2x", 45)
outputs.set("rotS2y", 0)
outputs.set("rotS2z", 0)
end
if
S2x~= 0 and S2y~=0 and S2z~=0 then
outputs.set("S2x", S2x)
outputs.set("S2y", S2y)
outputs.set("S2z", S2z)
thetax= (180*math.atan((Heady-C1y)/(Headz-C1z+0.00001)))/3.14
thetay= (180*math.atan((Headz-C1z)/(Headx-C1x+0.00001)))/3.14
thetaz= (180*math.atan((Heady-C1y)/(Headx-C1x+0.00001)))/3.14
print(thetax,thetay,thetaz)
outputs.set("rotS2x", 0)
outputs.set("rotS2y", 0)
outputs.set("rotS2z",0)
end

--Calculations
if L5x==0 and L5y==0 and L5z==0 and S1x==0 and S1y==0 and S1z ==0 and S2x==0 and S2y==0 and
S2z==0 then
outputs.set("LSA", 0)
outputs.set("SLSA",0)
outputs.set("SCA", 0)
outputs.set("CSCA",0)
outputs.set("SSCA",0)
outputs.set("SL1",0)
outputs.set("SL2",0)
outputs.set("VHS1C1", 0)
else if L5x~=0 and L5y~=0 and L5z~=0 and S1x~=0 and S1y~=0 and S1z ~=0 and S2x~=0 and S2y~=0 and
S2z~=0 then
--[[LUMBO-SACRAL ANGLE (LSA) ]]--
a8=((L5x-S1x)^2+(L5y-S1y)^2+(L5z-S1z)^2)^0.5
b8=((S1x-S2x)^2+(S1y-S2y)^2+(S1z-S2z)^2)^0.5
c8=((L5x-S2x)^2+(L5y-S2y)^2+(L5z-S2z)^2)^0.5
LSA=math.acos((c8^2-a8^2-b8^2)/(-2*a8*b8))
LSA=math.deg(LSA)
print("LSA angle is ",LSA)
outputs.set("LSA", LSA)

a9=((L5x-S1x)^2+(L5y-S1y)^2)^0.5
b9=((S1x-S2x)^2+(S1y-S2y)^2)^0.5
c9=((L5x-S2x)^2+(L5y-S2y)^2)^0.5
SLSA=math.acos((c9^2-a9^2-b9^2)/(-2*a9*b9))
SLSA=math.deg(SLSA)
print("SLSA angle is ",SLSA)
outputs.set("SLSA", SLSA)

--[[SACRAL ANGLE (SCA) ]]--
a25=((L5x-S2x)^2+(L5y-S2y)^2+(L5z-S2z)^2)^0.5
b25=((L5x-S2x)^2+(L5z-S2z)^2)^0.5
SCA=math.acos((a25/b25))
SCA=math.deg(SCA)
if L5x > S2x and L5z > S2z then print("AR -quardrant 1", SCA) outputs.set("SCA", SCA) end
if L5x > S2x and L5z < S2z then print("AL -quardrant 2",SCA) outputs.set("SCA", 180-SCA)  end
if L5x < S2x and L5z < S2z then print("PL -quardrant 3",SCA) outputs.set("SCA", 180+SCA) end
if L5x < S2x and L5z > S2z then print("AL -quardrant 4", SCA) outputs.set("SCA", 360-SCA) end
--[[CORONAL SACRAL ANGLE (SSCA) ]]--
a261=((L5z-S2z)^2)^0.5
b261=((L5z-S2z)^2+(L5y-S2y)^2)^0.5
CSCA=math.acos((a261/b261))
CSCA=math.deg(CSCA)
if L5z < S2z then print("CSCA angle is ", 180-CSCA)  outputs.set("CSCA", 180-CSCA) else print("CSCA
angle is ", CSCA) outputs.set("CSCA", CSCA)  end

--[[SAGITTAL SACRAL ANGLE (SSCA) ]]--
a26=((L5x-S2x)^2)^0.5
b26=((L5x-S2x)^2+(L5y-S2y)^2)^0.5
SSCA=math.acos((a26/b26))
SSCA=math.deg(SSCA)
if L5x < S2x then print("SSCA angle is ", 180-SSCA) outputs.set("SSCA", 180-SCA) else print("SSCA angle
is ", SSCA) outputs.set("SSCA", SSCA)  end

--[[TOTAL SPINE LENGTH AND HEIGHT]]--
SL1 =((C1x-C2x)^2+(C1y-C2y)^2+(C1z-C2z)^2)^0.5 +((C2x-C3x)^2+(C2y-C3y)^2+(C2z-
C3z)^2)^0.5+((C3x-C4x)^2+(C3y-C4y)^2+(C3z-C4z)^2)^0.5
+((C4x-C5x)^2+(C4y-C5y)^2+(C4z-C5z)^2)^0.5 +((C5x-C6x)^2+(C5y-C6y)^2+(C5z-C6z)^2)^0.5+((C6x-
C7x)^2+(C6y-C7y)^2+(C6z-C7z)^2)^0.5
+((C7x-T1x)^2+(C7y-T1y)^2+(C7z-T1z)^2)^0.5 +((T1x-T2x)^2+(T1y-T2y)^2+(T1z-T2z)^2)^0.5 +((T2x-
T3x)^2+(T2y-T3y)^2+(T2z-T3z)^2)^0.5
+((T3x-T4x)^2+(T3y-T4y)^2+(T3z-T4z)^2)^0.5 +((T4x-T5x)^2+(T4y-T5y)^2+(T4z-T5z)^2)^0.5 +((T5x-
T6x)^2+(T5y-T6y)^2+(T5z-T6z)^2)^0.5
+ ((T6x-T7x)^2+(T6y-T7y)^2+(T6z-T7z)^2)^0.5 + ((T7x-T8x)^2+(T7y-T8y)^2+(T7z-T8z)^2)^0.5+ ((T8x-
T9x)^2+(T8y-T9y)^2+(T8z-T9z)^2)^0.5
+ ((T9x-T10x)^2+(T9y-T10y)^2+(T9z-T10z)^2)^0.5+((T10x-T11x)^2+(T10y-T11y)^2+(T10z-T11z)^2)^0.5
+((T11x-T12x)^2+(T11y-T12y)^2+(T11z-T12z)^2)^0.5
```

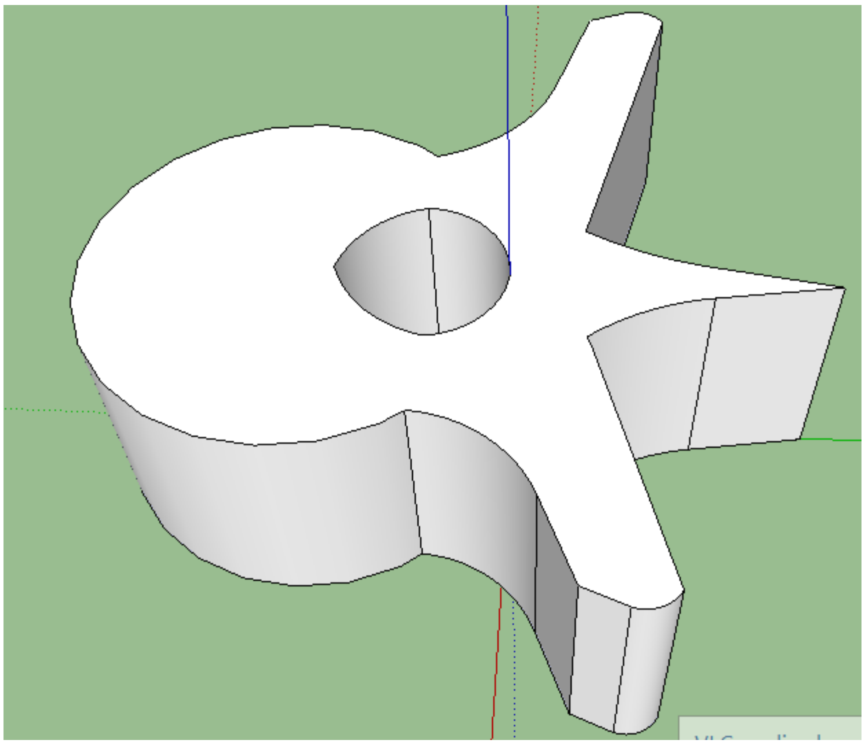
```
 + ((T12x-L1x)^2+(T12y-L1y)^2+(T12z-L1z)^2)^0.5 + ((L1x-L2x)^2+(L1y-L2y)^2+(L1z-L2z)^2)^0.5 +
((L2x-L3x)^2+(L2y-L3y)^2+(L2z-L3z)^2)^0.5
+ ((L3x-L4x)^2+(L3y-L4y)^2+(L3z-L4z)^2)^0.5 + ((L4x-L5x)^2+(L4y-L5y)^2+(L4z-L5z)^2)^0.5 + ((L5x-
S1x)^2+(L5y-S1y)^2+(L5z-S1z)^2)^0.5
+ ((S1x-S2x)^2+(S1y-S2y)^2+(S1z-S2z)^2)^0.5
print("Approximate Spine length", SL1)
outputs.set("SL1", SL1)

--[[SPINE LENGTH FROM C1 TO S1]]
SL2 =((C1x-C2x)^2+(C1y-C2y)^2+(C1z-C2z)^2)^0.5 +((C2x-C3x)^2+(C2y-C3y)^2+(C2z-
C3z)^2)^0.5+((C3x-C4x)^2+(C3y-C4y)^2+(C3z-C4z)^2)^0.5
+((C4x-C5x)^2+(C4y-C5y)^2+(C4z-C5z)^2)^0.5 +((C5x-C6x)^2+(C5y-C6y)^2+(C5z-C6z)^2)^0.5+((C6x-
C7x)^2+(C6y-C7y)^2+(C6z-C7z)^2)^0.5
+((C7x-T1x)^2+(C7y-T1y)^2+(C7z-T1z)^2)^0.5 +((T1x-T2x)^2+(T1y-T2y)^2+(T1z-T2z)^2)^0.5 +((T2x-
T3x)^2+(T2y-T3y)^2+(T2z-T3z)^2)^0.5
+((T3x-T4x)^2+(T3y-T4y)^2+(T3z-T4z)^2)^0.5 +((T4x-T5x)^2+(T4y-T5y)^2+(T4z-T5z)^2)^0.5 +((T5x-
T6x)^2+(T5y-T6y)^2+(T5z-T6z)^2)^0.5
+ ((T6x-T7x)^2+(T6y-T7y)^2+(T6z-T7z)^2)^0.5 + ((T7x-T8x)^2+(T7y-T8y)^2+(T7z-T8z)^2)^0.5+ ((T8x-
T9x)^2+(T8y-T9y)^2+(T8z-T9z)^2)^0.5
+ ((T9x-T10x)^2+(T9y-T10y)^2+(T9z-T10z)^2)^0.5+((T10x-T11x)^2+(T10y-T11y)^2+(T10z-T11z)^2)^0.5
+((T11x-T12x)^2+(T11y-T12y)^2+(T11z-T12z)^2)^0.5
 + ((T12x-L1x)^2+(T12y-L1y)^2+(T12z-L1z)^2)^0.5 + ((L1x-L2x)^2+(L1y-L2y)^2+(L1z-L2z)^2)^0.5 +
((L2x-L3x)^2+(L2y-L3y)^2+(L2z-L3z)^2)^0.5
+ ((L3x-L4x)^2+(L3y-L4y)^2+(L3z-L4z)^2)^0.5 + ((L4x-L5x)^2+(L4y-L5y)^2+(L4z-L5z)^2)^0.5 + ((L5x-
S1x)^2+(L5y-S1y)^2+(L5z-S1z)^2)^0.5
print("Approximate Spine length", SL2)
outputs.set("SL2", SL2)
--VERTICAL HEIGHT FROM S1 TO C1
VHS1C1 = C1y-S1y
print ("Verical height of the spine from C1 to S1 is ", VHS1C1)
outputs.set("VHS1C1", VHS1C1)
end
end
--end of calculations
```

THREE DIMENSIONAL VERTEBRA VISUALS

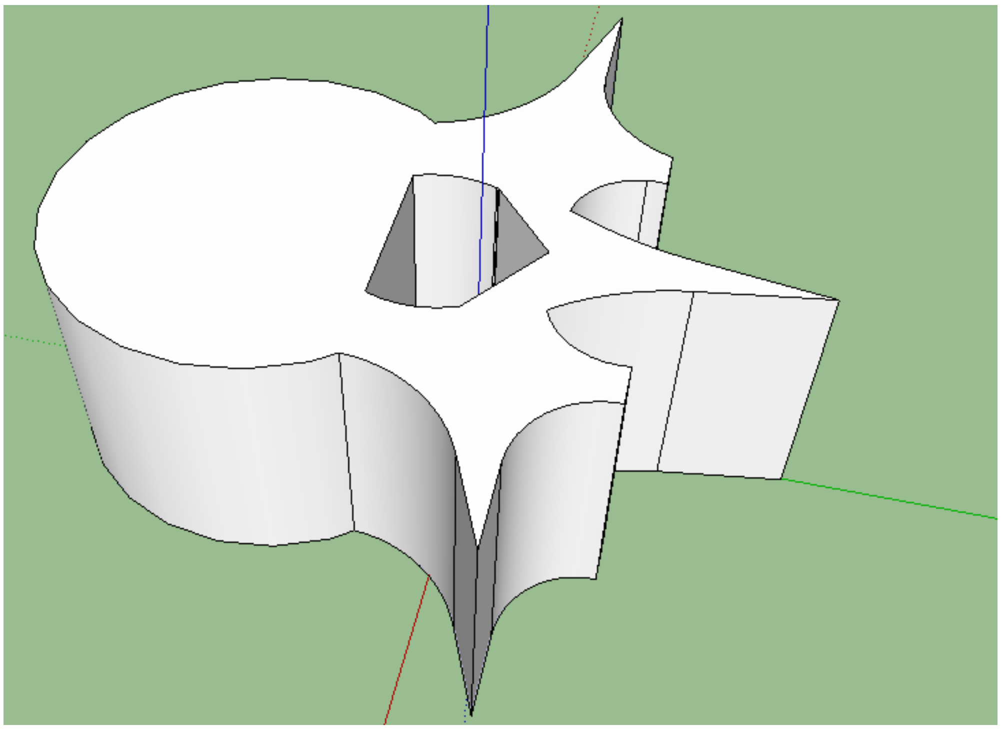
CERVICAL VERTEBRA



THORASIC VERTEBRA



LUMBAR VERTEBRA



SACRAL AND COCCYX VERTEBRAE