Todo list

Electronics and Computer Science
Faculty of Physical and Applied Sciences
University of Southampton

Merlin Webster
April 18, 2015

Skydiving Formation Recognition

Project supervisor: Dr Jonathon S Hare
Second examiner: Prof Lie-Liang Yang

A project report submitted for the award of
MEng Electronic Engineering

Abstract

Abstract here...

Keywords — computer vision, formation skydiving, pose detection, template matching, active shape models

Contents

List of Figures

1 Background

1.1 Formation Skydiving

Skydiving is a competitive and technical sport, requiring careful control of body position in order to move around the sky in a controlled manner whilst in free-fall.
A popular discipline in the sport is formation skydiving (FS). This involves multiple skydivers forming set shapes with their bodies, in order to score as many points as possible.

1.1.1 Rules

A point is awarded for each successful formation in a sequence ***(see figure 3)***. A successful formation is defined only by the correct hands of each skydiver holding the correct grips on the other skydivers ***(see figure 1)***, and not by the orientation of the skydivers themselves.

Figure 1: Diagram showing all of the available grips on a formation skydiving suit.

In 4 person FS (4-way FS) each skydiver has a specific slot in the formation. This is the position that they will always take when making each formation. This is done in such a way that the amount of movement required for each skydiver when transitioning between formations is minimised. The four slots in 4-way FS are known as Point, Tail, Outside Center (OC) and Inside Center (IC).

1.1.2 Training Practices

In order for the team to create formations in the sky, it is important that they practice the skydive multiple times on the ground. This is known as "dirt diving" and is often done using partially triangular wheeled platforms that each skydiver lays on, known as "creepers" ***(see figures 2a and 2b)***.

(a) FS practice platforms "creepers"

(b) "Creepers" in use while "dirt diving" a "Donut" formation

Figure 2: Images reproduced from `www.skydivespaceland.com/blog/images/creepers2.jpg` and `www.skydiveaz.com/images/old-images/creepers.jpg`

Another way that FS teams train is to use a vertical (indoor skydiving) wind tunnel. A fan is mounted at the bottom or bottom of the tunnel creating a strong wind that the skydivers can float on; simulating the environment of a regular skydive, but in a much more controlled environment ***(see figure 4)***.

1.2 Project Brief

The original focus of the project was to analyse video of dirt diving and identify which formations have been performed. The focus has been changed slightly, and is now to analyse footage of formation skydiving in a vertical wind tunnel. This change is largely due to the fact that the body position of a skydiver free-fall is much more similar to their body position in a wind tunnel, than when dirt diving. Another benefit of analysing footage taken in a wind tunnel is that the camera is mounted above the formation, remaining static ***(see figure 4)***. This allows the skydivers to much more easily be separated from the background via background subtraction[8]. The set of formations that will be analysed are the BPA rookie class of formations known as 'randoms'. The software must be able to recognise all 16 of the randoms ***(see figure 3)***.

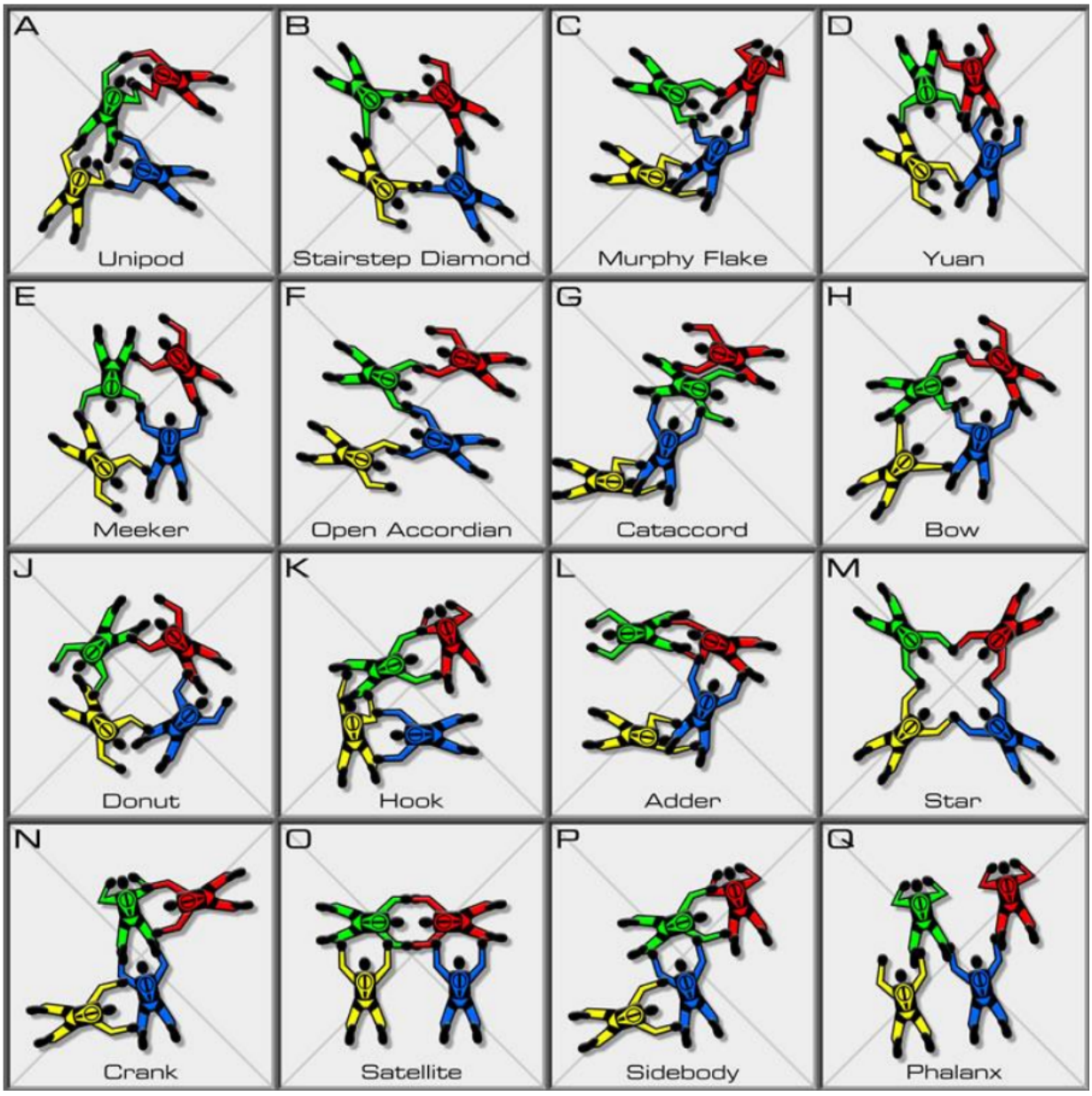


Figure 3: All 16 BPA rookie class FS 'randoms' formations, with their names. The colours for each slot are: Point red, OC green, IC blue, Tail yellow.
Reproduced from `http://www.teamsatori.co.uk/New%204W%20Random%20Dive%20pool.pdf`

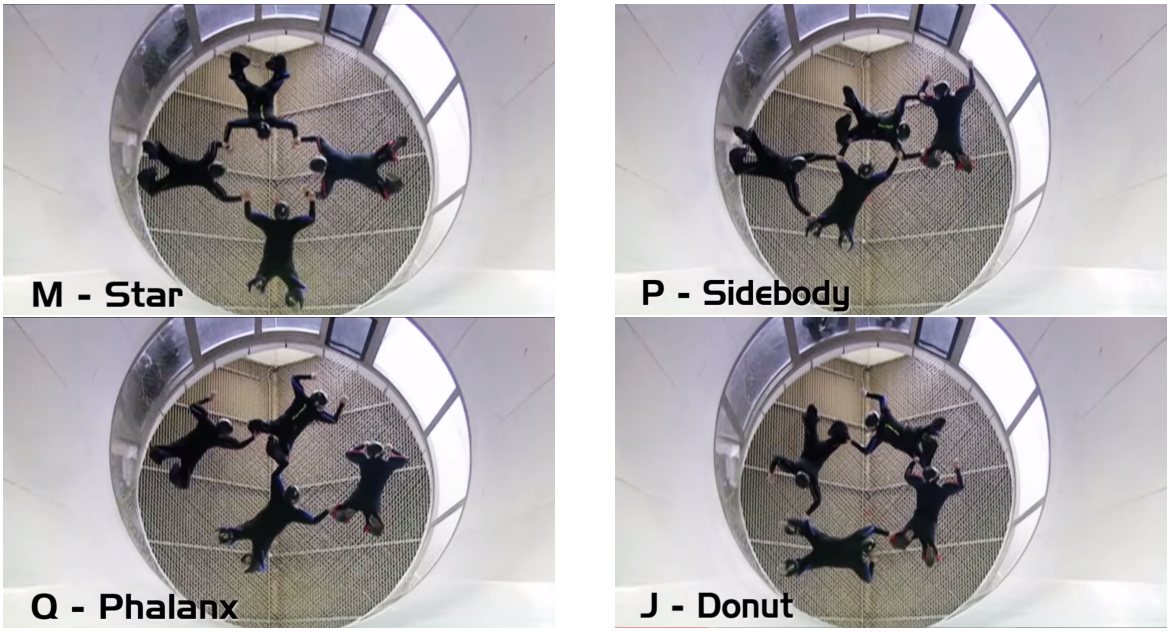


Figure 4: Sample FS formations performed in a wind tunnel, with their names.
Images reproduced from International Bodyflight Association, `www.youtube.com/watch?v=Y2B4S31Gf54&list=LLsEkKn0qzIHSGefmSGT_4og`.

An image of a skydiver in an FS body position can be simplified to a set of line segments between 11 key points; one for the waist, neck, head, left and right hands, elbows, knees and feet ***(see figure 5)***.

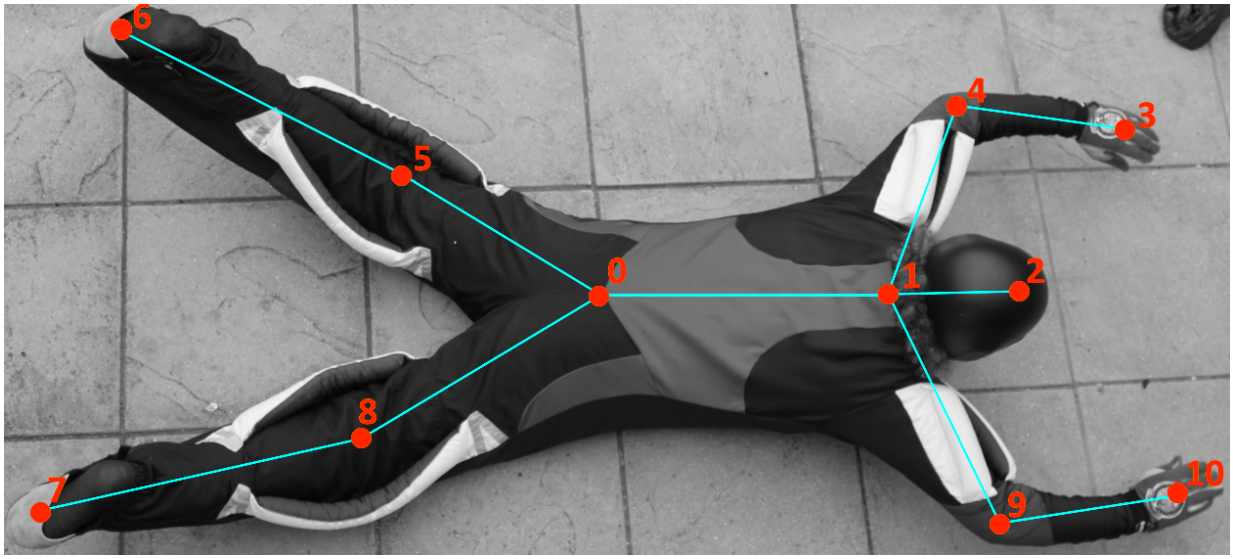


Figure 5: Diagram showing how an image of a skydiver can be simplified to a set of line segments between 11 points.

2 Literature Report

Due to the specialist nature of the project, relevant existing literature is somewhat limited. However, there does exist relevant literature on various methods for human posture recognition.

2.1 Pose Detection

Human pose detection can be a difficult problem to tackle; many factors contribute towards this, such as shadows, occlusions, complicated backgrounds and noisy source images.
There are many different proposed methods to tackle the task of human posture detection; over a wide variety of technologies. Some methods use depth cues extracted from 3d scanner information, such as [13] which uses a Microsoft Kinect to detect the human body. Other methods are designed to work with video[14], creating a 3d estimation of human pose in order to detect people. A common goal is to try to detect the pose of a person from a single static image. There are a variety of ways of doing this such as, using a model-based method, where an explicit pose model must be specified, often requiring complicated 3d modelling and rendering[15]. Another method used relies on texture cues in order to perform "Parts-based" object detection[12], this method neither uses an explicit model, or pre-labelled data.
However, the method that is deemed to most fit goal of recognising the pose of a skydiver is an adaptive method, such as Active Shape Models. Active Shape Models are more often used in facial feature detection[7][16]; given that it only relies on a set of simple points, labelled from a set of training images it should be ideal for the task at hand.

2.2 Active Shape Models

Active Shape Models are statistical shape models that are able iteratively deform in order to fit a given example image. They can also be used to characterise the ways in which a set of points in an image can move. Using principal component analysis[6], it is possible to find a minimal set of feature variables which control the overall shape of an object. Varying these enables every possible permutation of the set of image points to be found.
In order for this to be done, a training set of points must be collected by hand from images similar to those that will be analysed. This This training set of points forms the initial point matrix.

$$M = \begin{bmatrix} x & y & x & y & \dots & \dots & x & y \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ x & y & x & y & \dots & \dots & x & y \end{bmatrix}$$

Figure 6: Initial point matrix. Each row corresponds to a different image (blue). Pairs of columns contain coordinates of the same point across each of the images (red).

The iterative algorithm generalised procrustes analysis (GPA) is then used to optimally rotate, scale, and translate the points from each image. The goal of this is to have the points from each image match as closely as possible when superimposed on each other ***(see figure 7)***.

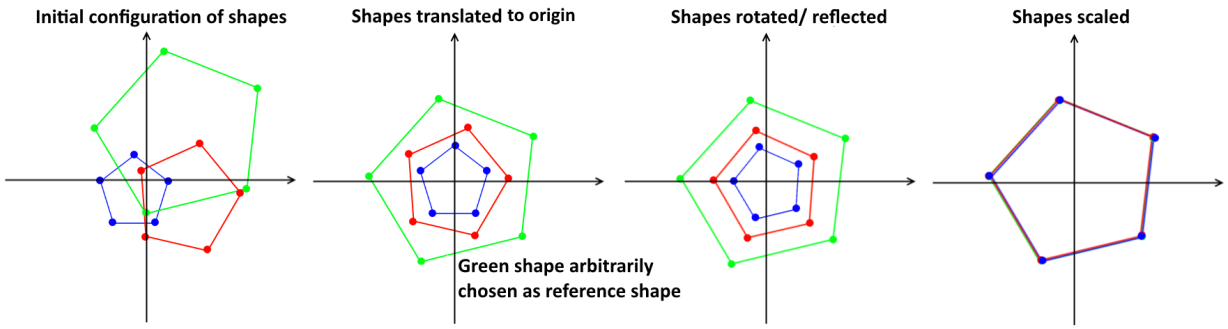


Figure 7: Performing generalised procrustes analysis on three shapes.

A shape matrix can be created using the same format as the initial point matrix, but containing the coordinates of the aligned image points. Finally, principal component analysis is used to reduce the dimensions of the shape matrix. This is done by finding the eigenvalues and eigenvectors of the covariance matrix. A more in depth explanation can be found at [6].

Using this method it is possible to find weather a skeleton of a skydiver is viable or not. This is done by creating the initial point matrix from a set of images of a skydiver in a variety of FS body positions. The resulting active shape model will contain information defining how a skydiver's body position can change. A set of test points that make up the skydiver's skeleton ***(see figure 5)*** can then be compared to this model to find if the skeleton is possible or not. This allows the skeleton to be recalculated, should it be found to be impossible.

2.3 Template Matching

3 Problem Analysis and Specification

Define terms to be used somewhere, e.g. landmark model, skydiver mask

4 Final System Design

In order to complete this project C++ was used, making use of the open source computer vision library OpenCV 3.0[] in order to reduce the time required to perform standard tasks, such as loading an image file, or receiving a video feed.

4.1 Computing Background Image

4.2 Foreground Extraction

4.2.1 Binary Median Filter

The classic way of doing a median filter is to apply a median go through each point in the image, and take the surrounding 8 pixels values, sort them by value, pick the median, and use that as the new value for the central pixel. However, for a binary image, this process can be optimised. Since we know each pixel will either be 0 or 1, it is unnecessary to sort the values of the surrounding pixels ,instead they are just added, and the total examined.

If the total is greater than 4, then the median is 1, and 0 otherwise.
This can be efficiently implemented by creating an n by n mask filled with ones, and convolving it with the binary image to create a new greyscale image with each pixel containing the sum of it's neighbours. The image is then thresholded at 4.5, causing any pixel which had more neighbours containing ones than zeroes to take the value of 1, and any with more 0 neighbours to take the value of 0 This gives the same result as applying the median filter in the classic way, but with much lower computation time, as only convolution and thresholding are required.

Add a figure for each step, with comparison images of classic way, and binary optimisation

Add a figure to show the ones mask used

4.3 Rotation Metric

4.3.1 Image Moments

Mention principle components, and how the formula conveniently resembles that of image moments, and give formulae for calculation image moments. Second image moment.

Another method that was tried was to find the minimum area rotated rectangle of the mask, and take it's angle to be the major axis of the skydiver. OpenCV has a function for finding the minimum area rectangle. Unfortunately, the angle it returns is the perpendicular angle to the rectangle's side closest to 0 degrees. This means that if this angle were used for the major axis, it may be 90, 180, or 270 degrees off. As a result, this method could not be used.

Add a figure showing the min area rotated rect around the skydiver mask, with the major axis drawn, and the the angle returned from the min area rotated rect drawn.

4.3.2 Skydiver Orientation

The major axis defines the angle of the line that runs along the skydivers spine. The line will either point towards the skydivers head, or crotch, but it is not possible to know which. This meant that the angle returned from the function to calculate the major axis was often off by 180 degrees.
In order to test if the angle calculated points towards the head or feet of the skydiver, the assumption is made that the distance from the skydivers centroid to their groin will be less than the distance from their centroid to their head. These distances are calculated by starting at the centroid, and then iteratively moving a test point further away in the direction of the major axis angle This is done until the test point is no longer in the skydiver mask. This gives the first distance, labelled the 'up' distance. The second distance, the 'down' distance, is found by repeating the process, but instead moving the test point in the direction 180 degrees from the major axis angle. The distances are then compared, and if the 'down' distance is greater than the 'up' distance the major axis angle is thought to be 180 degrees out. In this case 180 degrees is added or subtracted accordingly, so as to make sure the angle stays in the range 0 - 360 degrees.

Add figures showing the up and down distances labelled on the skydiver mask

4.4 Translation Metric

Another method that could be used is to find the minimum area rotated rectangle enclosing the skydiver mask or landmark model. The center point of the rotated rectangle would then be assumed to be the center point of the mask or landmark model. However, this would only work if the mask or landmark model was symmetric. If this was not the case, this method would give an incorrect centroid point.

include figure showing skydiver mask centroid using the two methods, demonstrating that the min area rect center does not work.)

4.5 Scale Metric

A scale metric was required in order to compare the sizes of different binary images, and contours.

Add an explanation as to why this was required at the top of the section.

Two separate methods were tried in order to find a reliable metric that could be used to compare the sizes of different binary images, and contours. The first of which was to use the centroid size, which is the square root of the summed squared distances from each point to the centroid.

Include centroid size equation and ref

While this could be used to compare the relative size of the landmark model between iterations, it was not useful when comparing the size of the binary mask to the landmark model.

The method that was finally employed was to find the minimum area rotated rectangle enclosing the landmark model, or binary mask, and use it's enclosed area as a scale metric. This proved to be more reliable when comparing the landmark model to a binary mask.

define how "it was not useful"

4.6 Template Matching

The search patch is a rotated square with side length $l_I = l_T + 2S$, where l_T and S are the side length of the template square, and the additional search area respectively. It is centred at the landmark point which is to be updated via template matching. The the search patch is extracted from the frame, and rotated, in the same way that the sub-images used to create the templates were. This means that the body part in the search patch has the best chance of lining up with the template, allowing for template matching. The template image (T) slides over the search patch (I) and it is compared to section of the search patch underneath it at every pixel. The result image (R) is formed from the template match amount at every pixel location. The method used to determine the template match at each point was OpenCV's method *CV_TM_CCORR_NORMED*.

$$R(x,y) = \frac{\sum_{x',y'}(T(x',y').I(x+x',y+y'))}{\sqrt{\sum_{x',y'} T(x',y')^2 . \sum_{x',y'} I(x+x',y+y')^2}}$$

The OpenCV function matchTemplate(...) was used for this.

Include figures showing the template, search area, and match amount, with max match point circled

ref opencv function website

R is then normalised, and the highest value pixel represents the location where the template best matched.

This point will be used for the updated landmark point, but first it must be transformed back to the correct coordinate space. This is done by reversing the effective translation, and rotation that was applied to the sample patch.

4.7 Model Plausibility Test

4.8 Model Initialisation

4.8.1 Finding Skydivers

Add a figure for each step

The purpose of this part of the system was to find a suitable frame to be used in order to create the initial contours, and binary masks of the skydivers. This frame should clearly show four separate skydivers, who are not touching. This was achieved in the following way:

1. Take a greyscale frame from the source video
2. Perform background subtraction on the frame in order to get an approximate binary mask of the skydivers
3. Apply the binary median filter to
4. Find the external contours of the binary mask
5. Calculate the standard deviation of the area of the list of contours
6. Keep only contours whose area is greater than one standard deviation of the mean area
7. Sort the remaining contours by area, in descending order
8. Take the four largest contours to represent the four skydivers. If less than four four contours remain after removing the smaller contours, then go back to step 1.

If four contours with an area greater than one standard deviation of the mean cannot be found, it is likely to be for one of the following reasons:

- The frame was not an image of the four skydivers as expected
- The frame was an image of the four skydivers, but two or more of the skydivers were touching. This would be the case if a formation was made. This failure would occur because the external contour of the overlapping skydivers would be seen as a single contour, and thus the number of contours of an appropriate size would be less than four.
- The calculated background image was not representative of the video background

Only finding the external contours caused any internal contour information to be lost. This meant that when the contours were drawn to create the binary masks, any gaps between the skydivers were filled unintentionally. This was fixed by creating a second mask image that retained the hole information, and taking the final skydiver blobs as the result of performing the logical AND operation on the two masks.

Add a figure to demonstrate this as well

4.9 Formation Detection

4.10 Landmark Collection Tool

In order to create the landmark model, a training set of landmarks must first be collected. These landmarks could then be used to perform principle component analysis

Ref

and collect the templates

Ref

. A separate program was written to do this whose function is as follows:

1. Load a training video of skydivers performing 4-way formation skydiving in the wind tunnel
2. Take a number of sample frames from this video
3. On each frame the user clicks on each landmark position in the order waist, neck, head, left hand, left elbow, left knee, left foot, right foot, right knee, right elbow, right hand, as shown in figure ***(see figure 5)***
4. The user then presses space in order to save the collected points
5. A stick man is then drawn over the points that were just labelled in order to verify that they were collected correctly
6. This is done four times, once for every skydiver in the frame, allowing 4 sets of landmarks to be collected for every frame
7. This process is repeated for each sample frame, and the data is saved to a file in the matrix format shown in ***figure 6***

Each frame used to collect landmarks is also saved as an image to be used by the template creation program.

4.11 Template Creation Tool

In order to perform template matching, a template image for each landmark must be made. A separate program was written to do this. As it's input it takes the landmark matrix, and all saved frames produced by the landmark collection program. It then takes rotation independent samples of each frame, centred at each of the landmarks, and creates a template image for each landmark point.

In order to create a template, all the sample images for that landmark point must be rotated in such a way that they are aligned.

include a figure showing aligned and unaligned images (different rotation)

In order the calculate the rotation (θ) for a landmark point (P_l), a reference point (P_r) is used (***see table 1***).

$$\theta = \arctan 2(\frac{P_r.y - P_l.y}{P_r.x - P_l.x})$$

Landmark Point	Waist	Neck	Head	Left Hand	Left Elbow	Left Knee	Left Foot	Right Foot	Right Knee	Right Elbow	Right Hand
Reference Point	Neck	Waist	Neck	Left Elbow	Neck	Waist	Left Knee	Left Knee	Waist	Neck	Right Elbow

Table 1: Table showing the reference point used for each landmark point

The reference points were chosen in order to make sure that body part in the sample is always pointing to the left, but it should be arbitrary which point is used as a reference, so long as the same point is used consistently.
The landmark point representing the left hand will be used to demonstrate the process:

1. Calculate find which reference point should be used from a look up table
2. Calculate the rotation angle θ
3. Rotate a copy of the frame by θ
4. Create a square sub-image from the sample, centred at P_l, with side length l
5. Save the sub-image, to be used later in creating the template for that landmark point

This process is repeated for every landmark vector in the landmark matrix 6, generating n sub-images for each landmark point, where n is the number of landmark vectors. The template image is then calculated as mean of all of the sub-images for that landmark point.

$$T = \hat{S}$$

Where T and S are the template image and the set of sub-images for the landmark point respectively.
Initially the nave approach used was to simply sum all the sub-images, and divide each pixel value in the resulting image by n to give the template image.

$$T = \frac{1}{n} \sum_{I \in S} I$$

However, this method was unsuccessful as the sub-images were 8-bit; summing them quickly caused the pixels in the sum image to reach the maximum value of 255. When they were then divided by n, it resulted in a very dark image, containing containing little to no information.
In order to avoid this issue, the cumulative moving average of the sub-images was used instead.

$$T_{i+1} = T_i + \frac{S_{i+1} - T_i}{i+1}, T_1 = S_1$$

The formula is iterated until $i = n - 1$ at which point $T_i = \hat{S}$ and the template image is complete. This prevented the templates from saturating, and produced usable template images (***see figure***

Ref

). Taking the median of the sample images was also considered as a possibility, but this method was rejected as all of the sample images had a very similar background, likely causing the median image to be almost entirely background, which is not useful for the templates.

4.12 Principle Component Analysis Tool

4.12.1 Generalised Procrustes Analysis

4.12.2 Principle Component Analysis

5 Additional Work

5.1 Principle Component Analysis Demonstration

5.2 Skeletonisation

Software has been written that creates a minimal, connected, 1 pixel thick skeleton from a binary mask. The function uses a medial axis transformation based method [1] to create the skeleton and the Ramer–Douglas–Peucker[4][5] (RDP) algorithm to minimise it.

5.2.1 Skeletonisation Algorithm

1. Find all boundary pixels (edge of the binary mask) using a binary laplacian mask[3]. For each boundary pixel do the following:

2. Test pixels neighbours

p8	p1	p2	y
p7	p0	p3	
p6	p5	p4	↓
x		→	

3. Count Np0, number of non-zero neighbours of p0.

4. Count Tp0, number of 0-1 transitions in sequence p1, p2, ..., p8.

5. Check the initial conditions; mark for deletion any point that satisfies them all. Initial conditions:

   - $cA : 2 <= Np0 <= 6$
   - $cB : Tp1 = 1$
   - $cC : p1.p3.p5 = 0$
   - $cD : p3.p5.p7 = 0$

6. Delete any points marked for deletion.

7. For each remaining boundary pixel, recalculate Np0 and Tp0, then check secondary conditions; marking for deletion any point that satisfies them all:
   Secondary conditions:

   - $cA : 2 <= Np0 <= 6$
   - $cB : Tp1 = 1$
   - $cC\_ : p1.p3.p7 = 0$
   - $cD\_ : p1.p5.p7 = 0$

8. Delete any points marked for deletion.

Repeat process until no points are deleted.
For more a more in depth explanation of the algorithm, see [1].

Initially a simpler algorithm[2] was used in order to create the skeleton using morphological operations to perform thinning. However, this algorithm did not ensure that the skeleton created was minimal or connected, and was therefore insufficient.

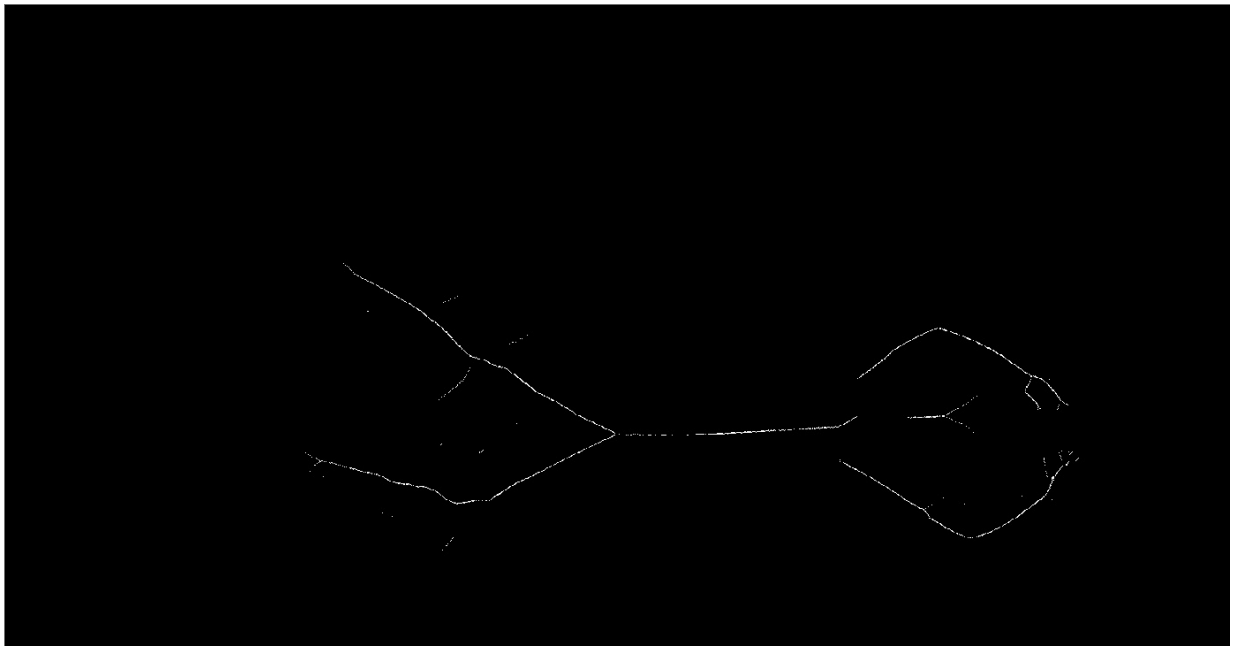


Figure 8: Result of using the morphological thinning algorithm on the binary image shown in ***figure 10b***.

5.2.2 Skeleton Simplification Algorithm

The RDP algorithm is designed to simplify a curve, while still maintaining its overall shape. A curve can be thought of as a collection of linked line segments, with each line segment stretching between two consecutive points on the curve. The basic principle of the algorithm is to reduce the number of line segments in a curve; creating a similar, but simplified curve. The algorithm is initially given an ordered array of all the points in the curve. The algorithm is also given the variable ϵ; this defines the degree to which the curve should be simplified and represents a minimum distance between line segments on the curve. It marks the start and end points of the curve as the first line segment (L), and then finds the point furthest from this initial line segment. If the distance from this point to the line segment (d) is less than ϵ then it is marked for removal. If the point is further than ϵ then it is kept and the algorithm recursively called with this point as the new end point. In this way the algorithm is able to remove every point that is closer than ϵ to the line segment that joins its neighbours.

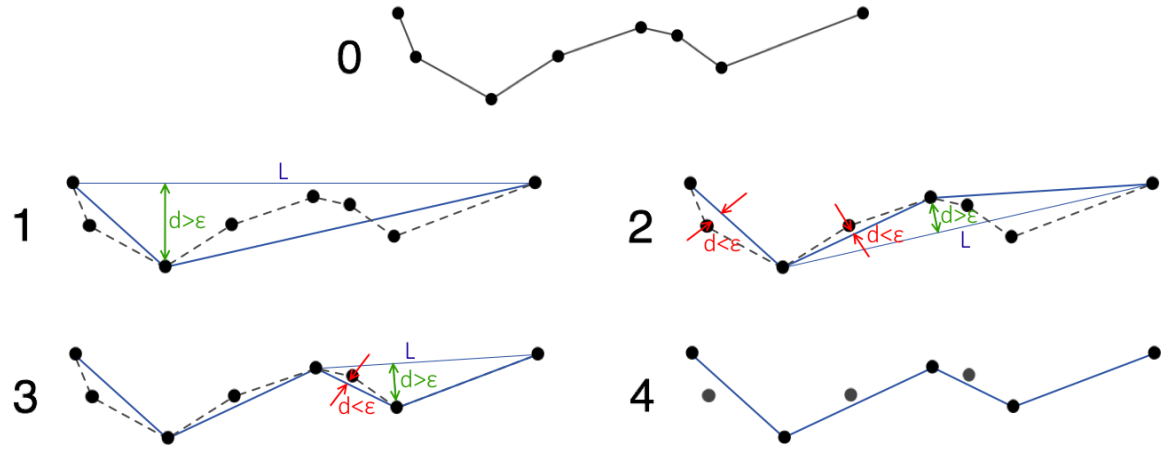


Figure 9: Diagram showing the RDP algorithm working on a curve. 0 being the initial curve to be simplified, and 4 being the fourth result the algorithm. Red arrows show points marked for deletion, green arrows show points to be kept
Modified from `http://upload.wikimedia.org/wikipedia/commons/9/91/Douglas_Peucker.png`.

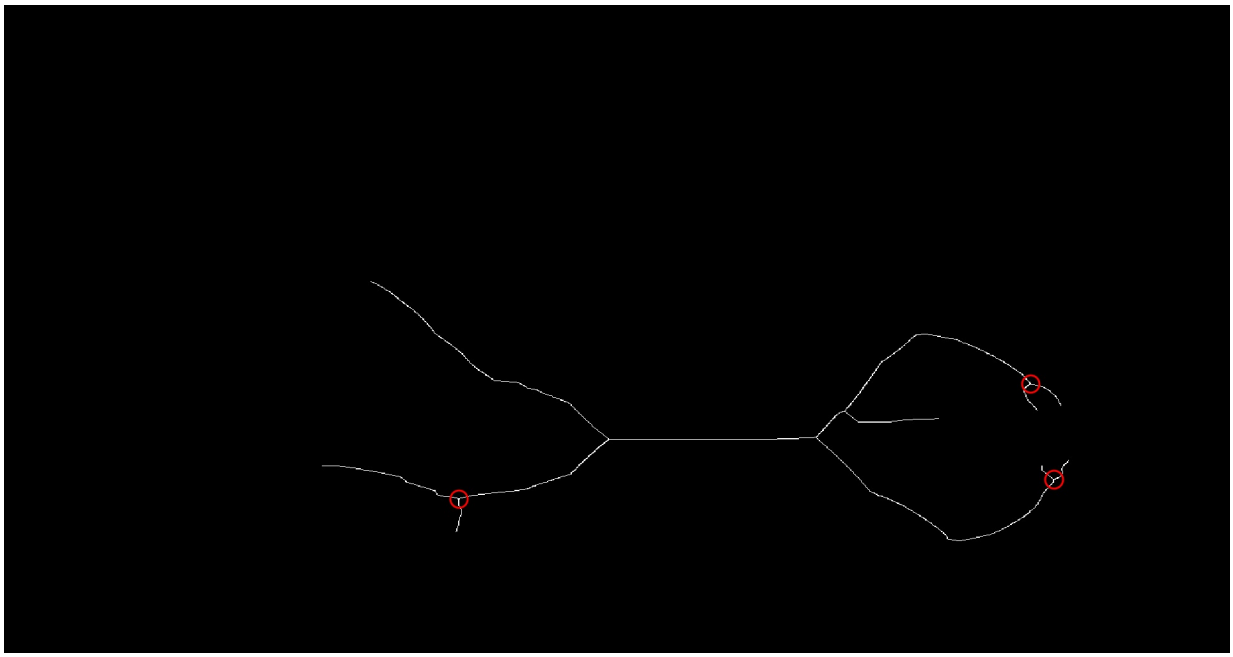
The RDP algorithm is used to reduce the number of points in the skeleton, simplifying it; however, it is only able to simplify curves with no bifurcations. This proves to be a problem when trying to simplify a skeleton produced by the skeltonisation algorithm, as it will always have bifurcations at the waist and neck. It may also also produce erroneous bifurcations at other points on the skeleton. These bifurcations can be seen in ***figure 10c***. The location of these bifurcations can be detected by applying a modified version of condition cB from the skeletonisation algorithm. Tp0 is calculated in the same way, by looking at the number of 0-1 transitions in the sequence p1, p2, ... , p8.

p8	p1	p2	y
p7	p0	p3	
p6	p5	p4	↓
x		→	

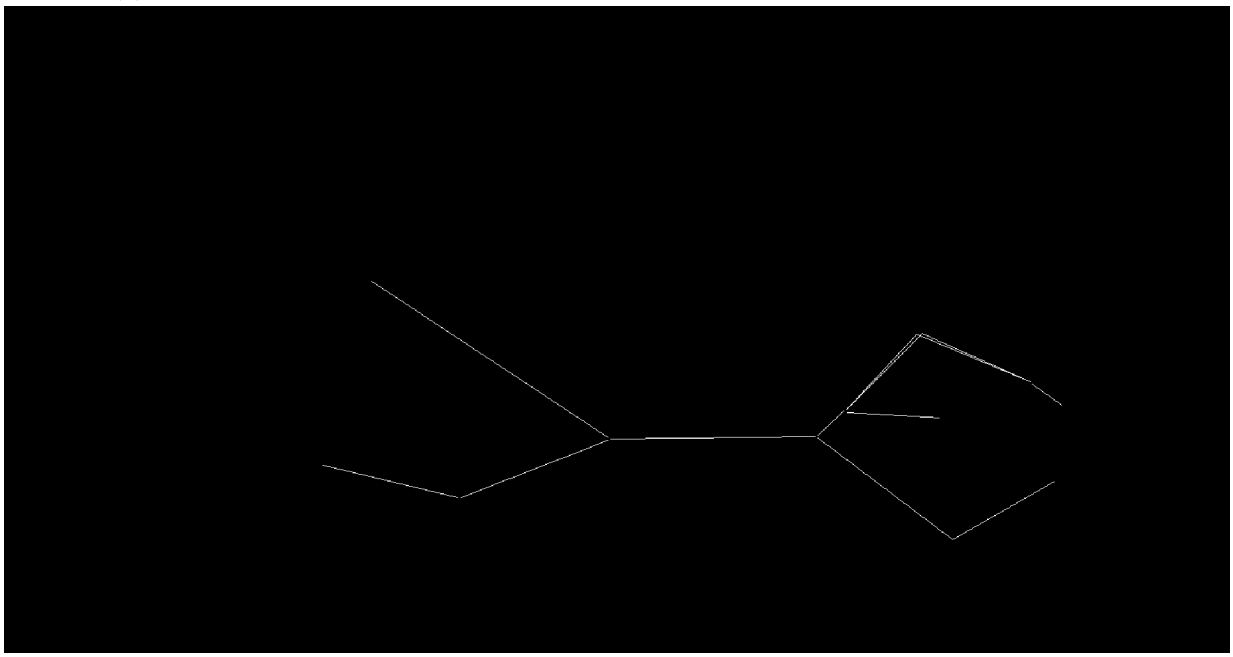
If $Tp0 > 2$ then a bifurcation is present at p0. This is always true, as the skeletonisation algorithm ensures that the skeleton is a single pixel thick 8-connected component.
Once the location of every bifurcation is found, the skeleton can be split into separate curves, by removing the bifurcation point. The RDP algorithm is then run on each of these curves independently. The simplified curves can then be recombined to form a simplified skeleton. At this point it is also possible the remove the false skeleton sections created by the skeletonisation algorithm forming erroneous bifurcations. The simplest way to do this is to delete any simplified skeleton section that is shorter than a given length. This works as the false skeleton sections tend to be shorter than the correct ones. A more reliable method for detecting false skeleton sections could be used in the final design.

(a) Skeletonisation source image.

(b) Binary mask of source image.



(c) Result of skeletonisation function. Erroneous bifurcations are circled in red.



(d) Result of skeleton simplification algorithm.

Figure 10: Results of skeletonisation and simplification algorithms.

6 Project Management

7 Conclusion

7.1 Critical Evaluation

7.2 Future Work

References

[1] Rafael C. Gonzalez & Richard E. Woods, "Skeletons", Digital Image Processing – Second Edition — International Edition (2001), pg650-653

[2] Rafael C. Gonzalez & Richard E. Woods, "Skeletons", Digital Image Processing – Second Edition — International Edition (2001), pg543-545

[3] Rafael C. Gonzalez & Richard E. Woods, "The Laplacian", Digital Image Processing – Second Edition — International Edition (2001), pg581-585

[4] Urs Ramer, "An iterative procedure for the polygonal approximation of plane curves", Computer Graphics and Image Processing (1972), pg 244–256

[5] David Douglas & Thomas Peucker, "Algorithms for the reduction of the number of points required to represent a digitized line or its caricature", The Canadian Cartographer (1973), pg112–122

[6] Imperial College London, "Lecture 15: Principal Component Analysis", `www.doc.ic.ac.uk/~dfg/ProbabilisticInference/IDAPILecture15.pdf`, Last access: 9 Dec 2014

[7] Tim Cootes, E. R. Baldock, & J. Graham, "An introduction to active shape models.", Image Processing and Analysis(2000), pg223—248

[8] Piccardi, M., "Background subtraction techniques: a review," Systems, Man and Cybernetics, 2004 IEEE International Conference, pg3099—3104 vol.4, (10-13 Oct 2004), doi: 10.1109/ICSMC.2004.1400815

[9] OpenCV, "The OpenCV Reference Manual Release 2.4.9.0"(April 21, 2014), `http://docs.opencv.org/opencv2refman.pdf`, Last access: 09 Dec 2014

[10] Qt Project, "Qt Reference Pages", QtDoc 5.3, `http://docs.opencv.org/opencv2refman.pdf`, Last access: 09 Dec 2014

[11] "Automatic Detection of 2D Human Postures Based on Single Images", QtDoc 5.3, `http://docs.opencv.org/opencv2refman.pdf`, Last access: 09 Dec 2014

[12] Wachs, Juan P., Deborah Goshorn, and Mathias Kolsch, "Recognizing human postures and poses in monocular still images."(2009),

[13] Lu Xia, Chia-Chih Chen, Aggarwal, J.K., "Human detection using depth information by Kinect," Computer Vision and Pattern Recognition Workshops (CVPRW), 2011 IEEE Computer Society Conference on , vol., no., pp.15,22, 20-25 June 2011 doi: 10.1109/CVPRW.2011.5981811

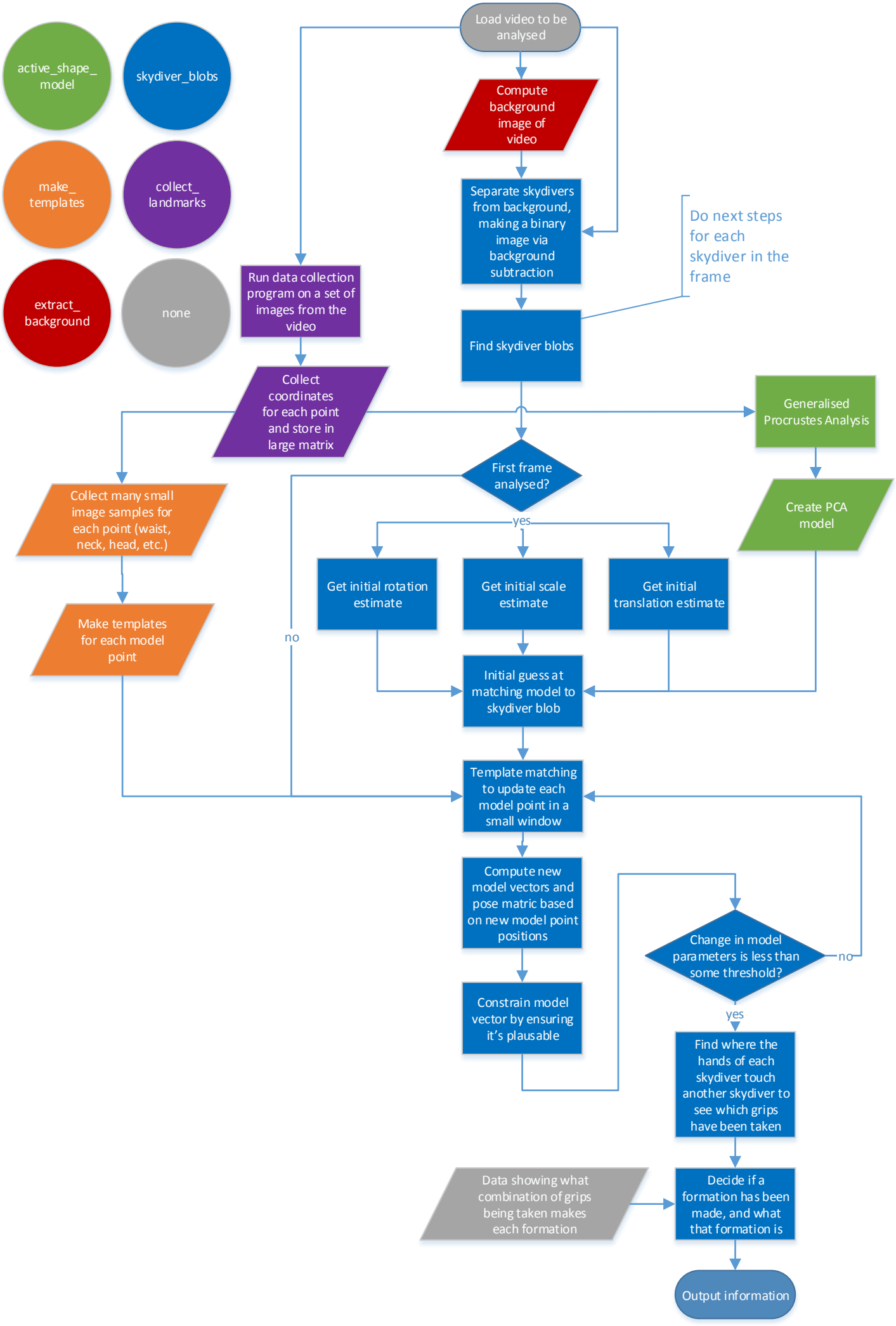
[14] Yu Tsz-Ho, Tae-Kyun Kim, and Roberto Cipolla, "Unconstrained monocular 3d human pose estimation by action detection and cross- modality regression forest.", In Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on, pp. 3642-3649. IEEE, 2013.

[15] Christophe Doignon (2007), An Introduction to Model-Based Pose Estimation and 3-D Tracking Techniques, Scene Reconstruction Pose Estimation and Tracking, Rustam Stolkin (Ed.), ISBN: 978-3-902613-06-6, InTech, pg360-376, Available from: http://www.intechopen.com/books/scene_reconstruction_pose_estimation_and_tracking/an_introduction_to_model-based_pose_estimation_and_3-d_tracking_techniques

[16] Prabhu, Utsav, and Keshav Seshadri, "Facial Recognition Using Active Shape Models, Local Patches and Support Vector Machines."

Appendices

A - Final System Flowchart


active_shape_model
skydiver_blobs
make_templates
collect_landmarks
extract_background
none
Load video to be analysed
Compute background image of video
Separate skydivers from background, making a binary image via background subtraction
Do next steps for each skydiver in the frame
Find skydiver blobs
Run data collection program on a set of images from the video
Collect coordinates for each point and store in large matrix
Generalised Procrustes Analysis
Create PCA model
First frame analysed?
yes
no
Collect many small image samples for each point (waist, neck, head, etc.)
Make templates for each model point
Get initial rotation estimate
Get initial scale estimate
Get initial translation estimate
Initial guess at matching model to skydiver blob
Template matching to update each model point in a small window
Compute new model vectors and pose matric based on new model point positions
Constrain model vector by ensuring it's plausable
Change in model parameters is less than some threshold?
no
yes
Find where the hands of each skydiver touch another skydiver to see which grips have been taken
Data showing what combination of grips being taken makes each formation
Decide if a formation has been made, and what that formation is
Output information

B - Initial Gantt Chart

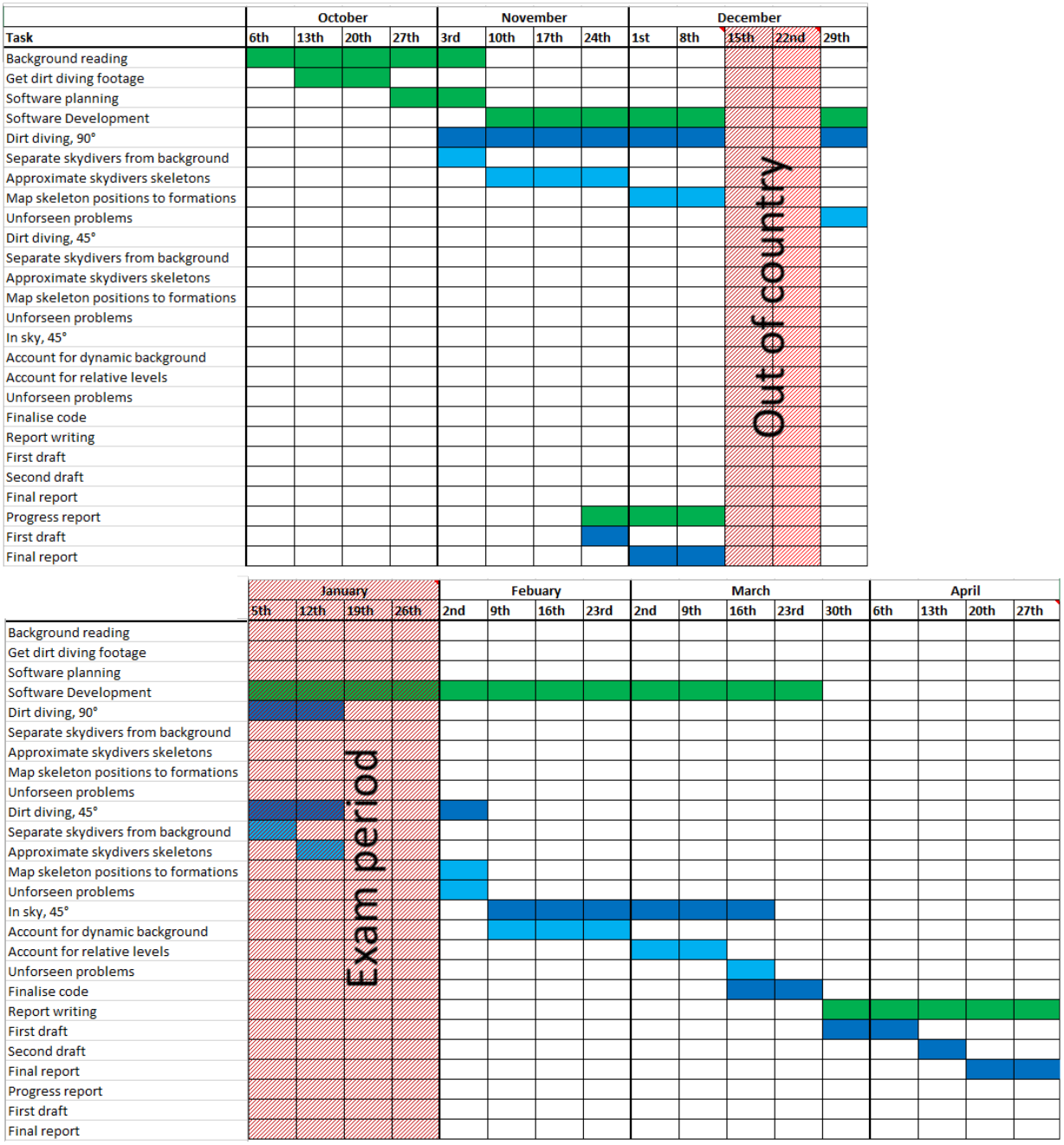


Figure 11: Initial Gantt Chart

C - Revised Gantt Chart

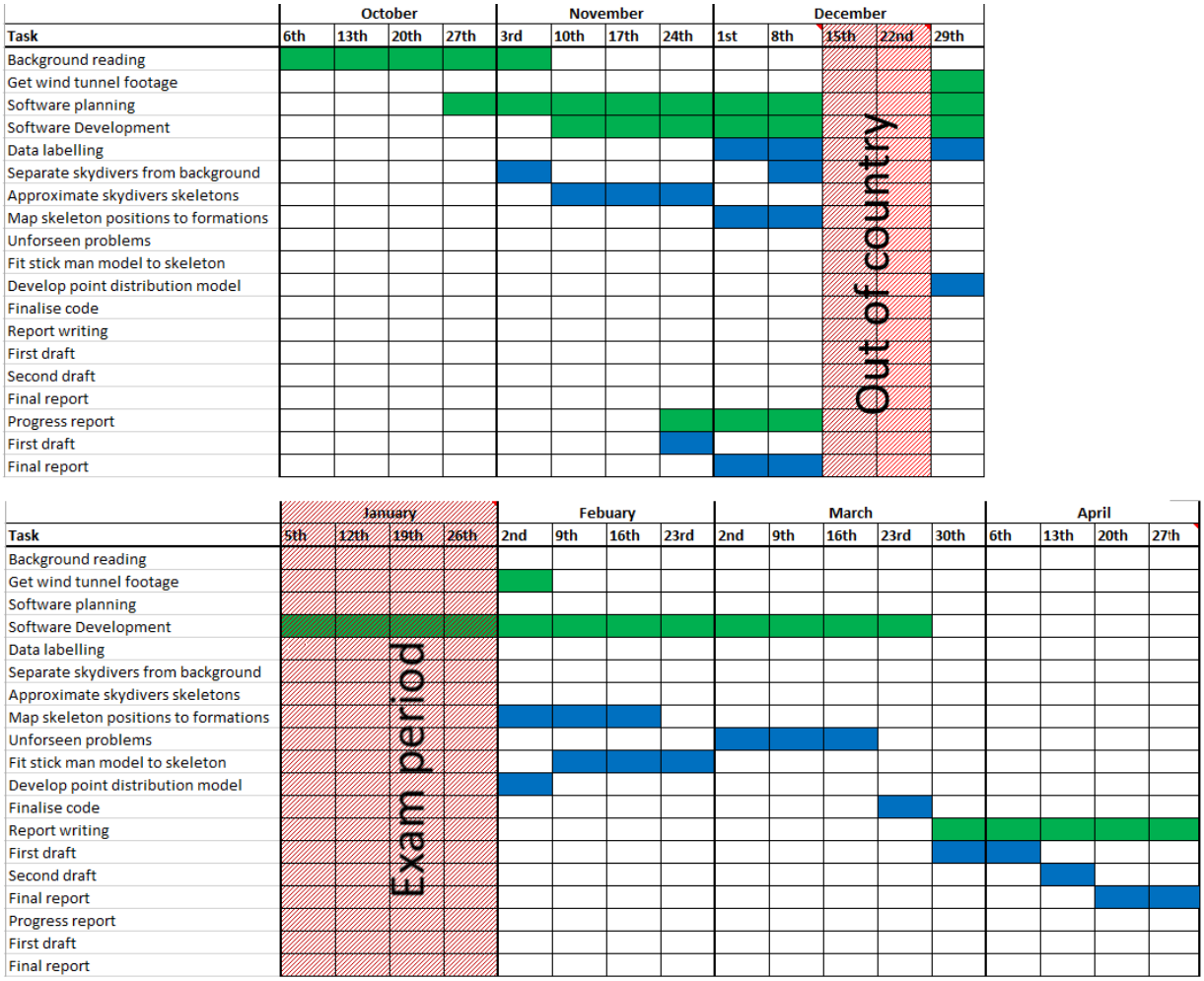


Figure 12: Revised Gantt Chart

D - Progress Report

Electronics and Computer Science
Faculty of Physical and Applied Sciences
University of Southampton

Merlin Webster
December 9, 2014

Skydiving Formation Recognition

Project supervisor: Dr Jonathon S Hare
Second examiner: Prof Lie-Liang Yang

A project progress report submitted for the award of
MEng Electronic Engineering

Abstract

Contrary to popular belief, skydiving is a competitive and technical sport, requiring careful control of body position in order to not only remain stable, but also to move around the sky in free-fall. A popular discipline in competitive skydiving is formation skydiving, where a group of skydivers form set shapes with their bodies whilst in free-fall. This Report proposes a software tool to be written; able to automatically judge formation skydiving footage. This will be done detecting the pose of each skydiver in the formation individually; this information will then be used to find the overall shape of the formation. This tool will combine and extend well established computer vision methods such as Active Shape Models, skeletonisation. The ultimate goal is to remove the need for manual competition judging, switching to a fully automated system.

Keywords — computer vision, video analysis, formation skydiving, posture detection

Contents

List of Figures

1 Background

1.1 Formation Skydiving

Skydiving is a competitive and technical sport, requiring careful control of body position in order to move around the sky in a controlled manner whilst in free-fall.
A popular discipline in the sport is formation skydiving (FS). This involves multiple skydivers forming set shapes with their bodies, in order to score as many points as possible.

1.1.1 Rules

A point is awarded for each successful formation in a sequence ***(see figure 3)***. A successful formation is defined only by the correct hands of each skydiver holding the correct grips on the other skydivers ***(see figure 1)***, and not by the orientation of the skydivers themselves.

Figure 1: Diagram showing all of the available grips on a formation skydiving suit.

In 4 person FS (4-way FS) each skydiver has a specific slot in the formation. This is the position that they will always take when making each formation. This is done in such a way that the amount of movement required for each skydiver when transitioning between formations is minimised. The four slots in 4-way FS are known as Point, Tail, Outside Center (OC) and Inside Center (IC).

1.1.2 Training Practices

In order for the team to create formations in the sky, it is important that they practice the skydive multiple times on the ground. This is known as "dirt diving" and is often done using partially triangular wheeled platforms that each skydiver lays on, known as "creepers" ***(see figures 2a and 2b)***.

(a) FS practice platforms "creepers"

(b) "Creepers" in use while "dirt diving" a "Donut" formation

Figure 2: Images reproduced from `www.skydivespaceland.com/blog/images/creepers2.jpg` and `www.skydiveaz.com/images/old-images/creepers.jpg`

Another way that FS teams train is to use a vertical (indoor skydiving) wind tunnel. A fan is mounted at the bottom of the tunnel creating a strong wind that the skydivers can float on; simulating the environment of a regular skydive, but in a much more controlled environment ***(see figure 4)***.

1.2 Project Brief

The original focus of the project was to analyse video of dirt diving and identify which formations have been performed. The focus has been changed slightly, and is now to analyse footage of formation skydiving in a vertical wind tunnel. This change is largely due to the fact that the body position of a skydiver free-fall is much more similar to their body position in a wind tunnel, than when dirt diving. Another benefit of analysing footage taken in a wind tunnel is that the camera is mounted above the formation, remaining static ***(see figure 4)***. This allows the skydivers to much more easily be separated from the background via background subtraction[8]. The set of formations that will be analysed are the BPA rookie class of formations known as 'randoms'. The software must be able to recognise all 16 of the randoms ***(see figure 3)***.

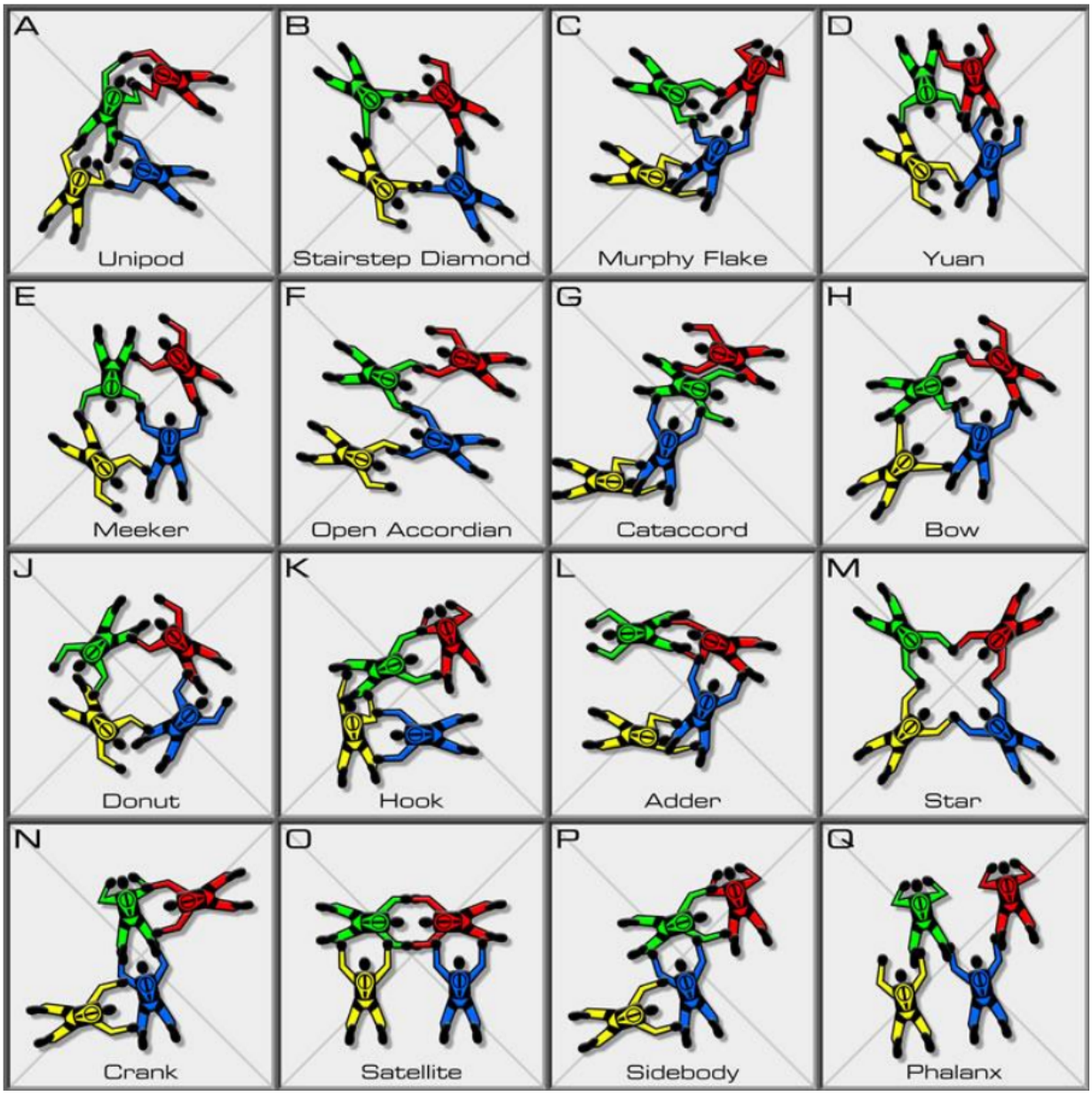


Figure 3: All 16 BPA rookie class FS 'randoms' formations, with their names. The colours for each slot are: Point red, OC green, IC blue, Tail yellow.
Reproduced from `http://www.teamsatori.co.uk/New%204W%20Random%20Dive%20pool.pdf`

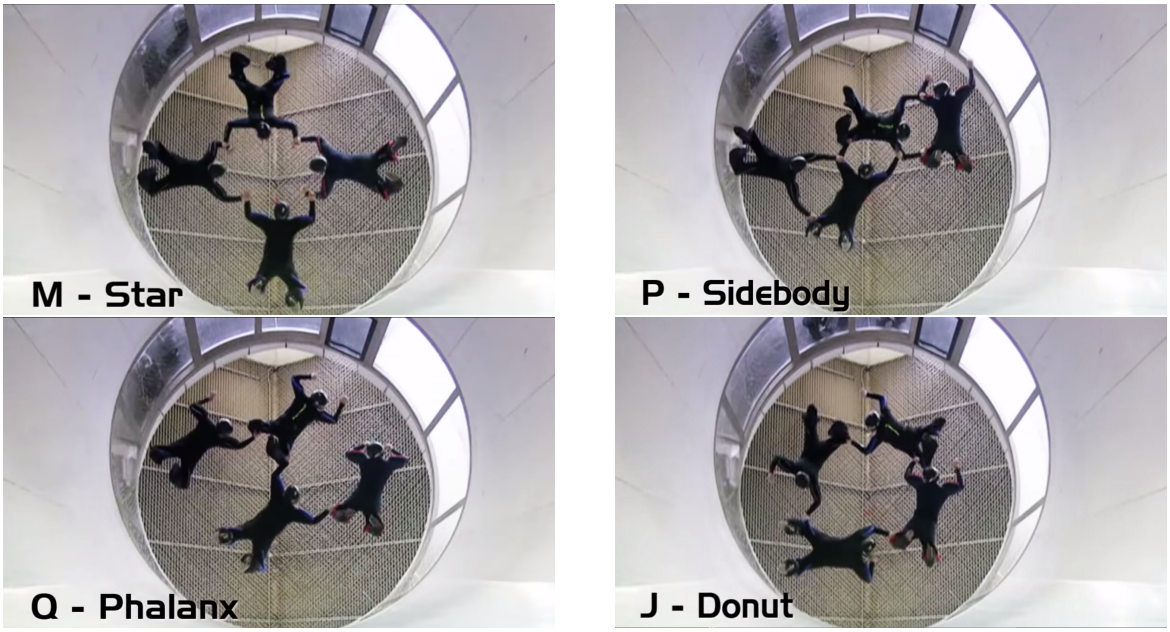


Figure 4: Sample FS formations performed in a wind tunnel, with their names.
Images reproduced from International Bodyflight Association, `www.youtube.com/watch?v=Y2B4S31Gf54&list=LLsEkKn0qzIHSGefmSGT_4og`.

An image of a skydiver in an FS body position can be simplified to a set of line segments between 11 key points; one for the waist, neck, head, left and right hands, elbows, knees and feet ***(see figure 5)***.

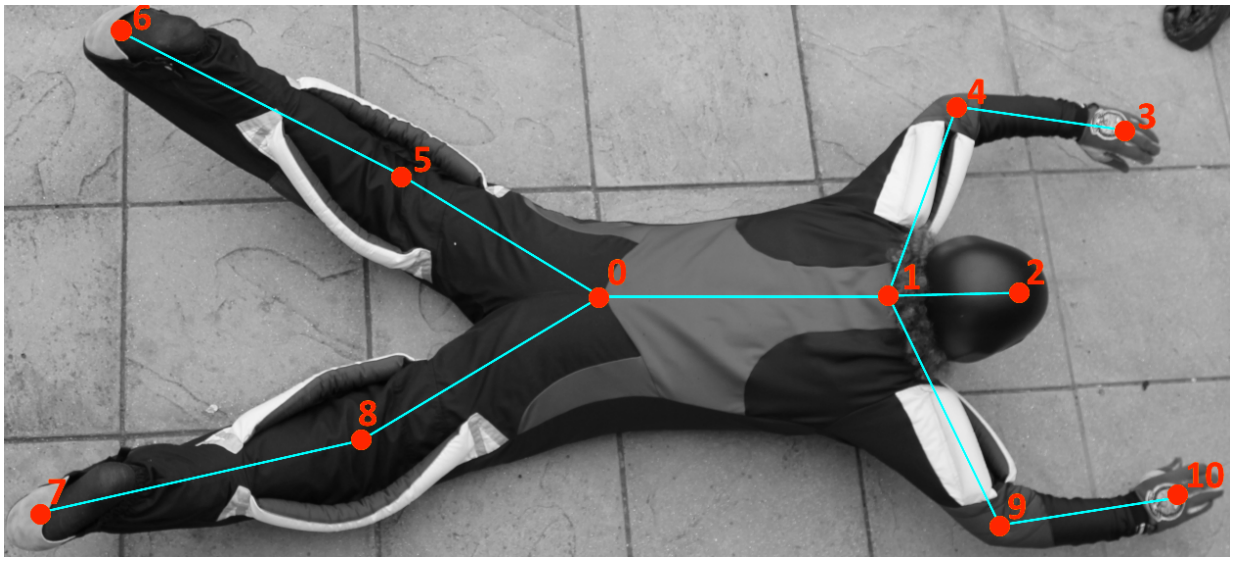


Figure 5: Diagram showing how an image of a skydiver can be simplified to a set of line segments between 11 points.

In order to complete this project C++ will be used, making use of the open source computer vision library OpenCV 3.0[] in order to reduce the time required to perform standard tasks, such as loading an image file, or receiving a video feed. If a graphical user interface (GUI) were to be implemented, the open source GUI library Qt 5.3[10] would be used. These libraries are platform independent; this should ensure that the program is not tied to any single operating system.

2 Literature Report

Due to the specialist nature of the project, relevant existing literature is somewhat limited. However, there does exist relevant literature on various methods for human posture recognition.

2.1 Posture Detection

Human pose detection can be a difficult problem to tackle; many factors contribute towards this, such as shadows, occlusions, complicated backgrounds and noisy source images.
There are many different proposed methods to tackle the task of human posture detection; over a wide variety of technologies. Some methods use depth cues extracted from 3d scanner information, such as [13] which uses a Microsoft Kinect to detect the human body. Other methods are designed to work with video[14], creating a 3d estimation of human pose in order to detect people. A common goal is to try to detect the pose of a person from a single static image. There are a variety of ways of doing this such as, using a model-based method, where an explicit pose model must be specified, often requiring complicated 3d modelling and rendering[15]. Another method used relies on texture cues in order to perform "Parts-based" object detection[12], this method neither uses an explicit model, or pre-labelled data.
However, the method that is deemed to most fit goal of recognising the pose of a skydiver is an adaptive method, such as Active Shape Models. Active Shape Models are more often used in facial feature detection[7][16]; given that it only relies on a set of simple points, labelled from a set of training images it should be ideal for the task at hand.

2.2 Active Shape Models

Active Shape Models are statistical shape models that are able iteratively deform in order to fit a given example image. They can also be used to characterise the ways in which a set of points in an image can move. Using principal component analysis[6], it is possible to find a minimal set of feature variables which control the overall shape of an object. Varying these enables every possible permutation of the set of image points to be found.
In order for this to be done, a training set of points must be collected by hand from images similar to those that will be analysed. This This training set of points forms the initial point matrix.

$$M = \begin{bmatrix} x & y & x & y & \dots & \dots & x & y \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ x & y & x & y & \dots & \dots & x & y \end{bmatrix}$$

Figure 6: Initial point matrix. Each row corresponds to a different image (blue). Pairs of columns contain coordinates of the same point across each of the images (red).

The iterative algorithm generalised procrustes analysis (GPA) is then used to optimally rotate, scale, and translate the points from each image. The goal of this is to have the

points from each image match as closely as possible when superimposed on each other ***(see figure 7)***.

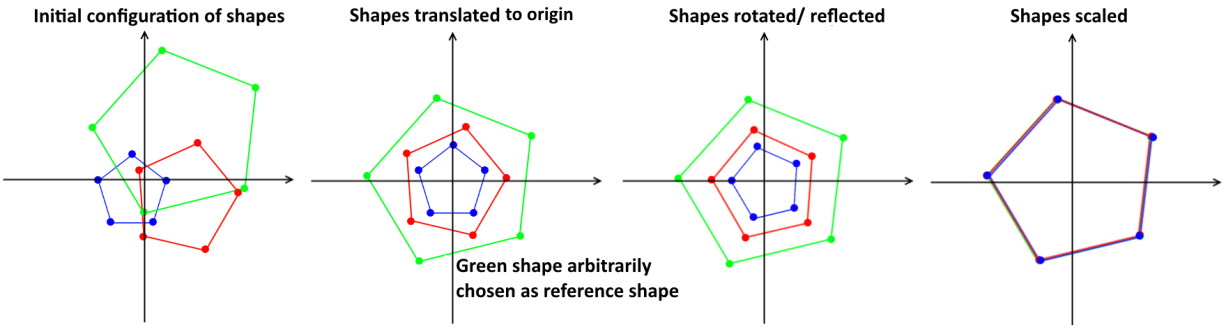


Figure 7: Performing generalised procrustes analysis on three shapes.

A shape matrix can be created using the same format as the initial point matrix, but containing the coordinates of the aligned image points. Finally, principal component analysis is used to reduce the dimensions of the shape matrix. This is done by finding the eigenvalues and eigenvectors of the covariance matrix. A more in depth explanation can be found at [6].

Using this method it is possible to find weather a skeleton of a skydiver is viable or not. This is done by creating the initial point matrix from a set of images of a skydiver in a variety of FS body positions. The resulting active shape model will contain information defining how a skydiver's body position can change. A set of test points that make up the skydiver's skeleton ***(see figure 5)*** can then be compared to this model to find if the skeleton is possible or not. This allows the skeleton to be recalculated, should it be found to be impossible.

3 Proposed Final Design

The proposed final design will take the form of a command line tool. It will take the path to a video of formation skydiving to be analysed as an argument. It's output will be a similar video file, but with an overlay showing which (if any) formations have been made at each stage in the video. It will also highlight the skeletons of the skydivers in different colours in order to show which slot in the formation they are flying. The skeletons will take the same colours as in ***figure 3***. Time permitting, a GUI may be implemented for the command line tool in order to make the software more user friendly.

Given that the project is entirely software based, additional support towards the project should not be necessary.

3.1 Flow Chart

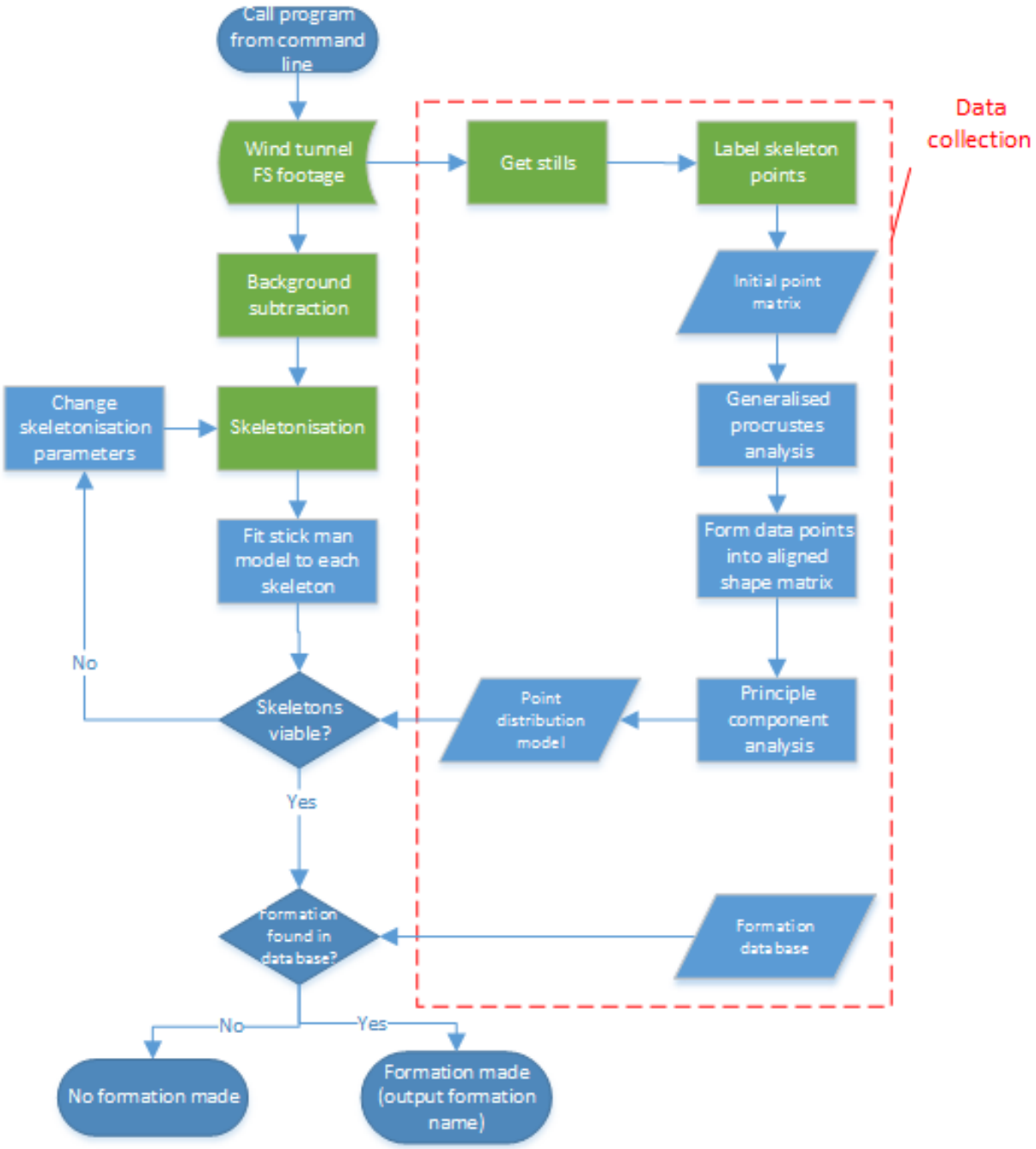


Figure 8: Flowchart showing overall functionality of proposed final design. Items shown in green have been completed. Items inside the red box are to be executed only once, and their results saved for later use.

4 Completed Work

4.1 Data Collection

Software has been written that handles collection of data for the initial point matrix to be used to create the point distribution model. In order to speed up data collection, the

program displays an image of a skydiver and asks the user to left click on different parts of the skydiver's body. The user can undo the last point by right clicking. Once all the points for the image are recorded, they are written to an output file and form the next row of the initial point matrix. Another image is then shown and the process repeats. This is a console based application and takes the file path of an image list as its only input ***(see figure 9)***. This image list is stored in an XML file, as OpenCV natively supports input and output to XML files. For each image, the program adds another row to the initial point matrix ***(see figure 6)***. Currently only 10 images have been used to create the initial point matrix, but this number will be increased before it is used to create the point distribution model.

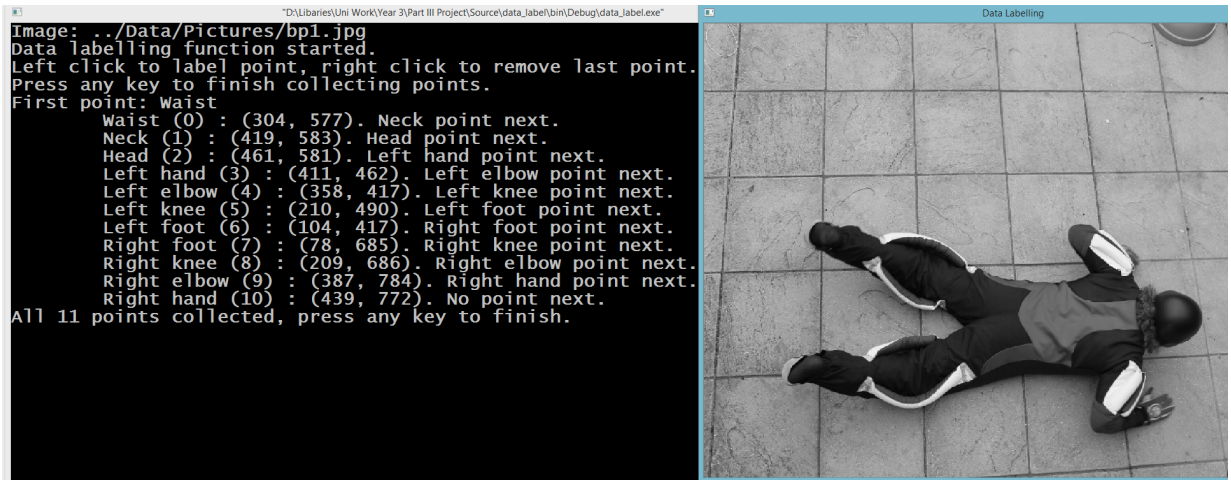


Figure 9: Data labelling program. Screenshot taken once all the required points of the image had been clicked on.

```
576,303,579,418,570,466,457,415,408,355,495,216,420,099,696,081,677,206,792,382,772,438
442,289,444,392,442,433,417,428,326,401,393,215,268,169,514,156,550,218,626,412,527,437
548,317,543,423,543,464,512,463,403,434,493,254,388,216,581,187,643,243,746,447,627,472
563,302,561,404,537,450,536,464,425,421,476,231,349,191,616,160,656,223,753,435,631,462
550,267,555,375,532,413,488,442,407,392,470,193,349,164,617,124,654,189,742,405,650,449
565,318,578,441,566,477,513,489,409,445,490,231,356,112,739,088,673,222,787,461,670,498
557,312,572,434,548,472,556,464,512,481,395,438,477,229,352,118,715,088,671,219,783,455
574,308,560,423,566,482,526,470,405,435,496,234,374,194,636,170,682,229,800,444,665,476
580,308,578,424,571,468,524,513,437,453,509,228,372,173,643,154,693,227,758,469,668,512
582,305,583,419,583,467,422,505,411,438,496,227,349,111,766,086,698,212,775,458,762,527
```

Figure 10: Matrix created when the program is supplied with a sample image list with 10 images in it.

4.2 Skeletonisation

Software has been written that creates a minimal, connected, 1 pixel thick skeleton from a binary mask. The function uses a medial axis transformation based method [1] to create the skeleton and the Ramer–Douglas–Peucker[4][5] (RDP) algorithm to minimise it.

4.2.1 Skeletonisation Algorithm

1. Find all boundary pixels (edge of the binary mask) using a binary laplacian mask[3]. For each boundary pixel do the following:

2. Test pixels neighbours

p8	p1	p2	y
p7	p0	p3	
p6	p5	p4	↓
x		→	

3. Count Np0, number of non-zero neighbours of p0.

4. Count Tp0, number of 0-1 transitions in sequence p1, p2, ..., p8.

5. Check the initial conditions; mark for deletion any point that satisfies them all.
   Initial conditions:

   - $cA : 2 <= Np0 <= 6$
   - $cB : Tp1 = 1$
   - $cC : p1.p3.p5 = 0$
   - $cD : p3.p5.p7 = 0$

6. Delete any points marked for deletion.

7. For each remaining boundary pixel, recalculate Np0 and Tp0, then check secondary conditions; marking for deletion any point that satisfies them all:
   Secondary conditions:

   - $cA : 2 <= Np0 <= 6$
   - $cB : Tp1 = 1$
   - $cC_{-} : p1.p3.p7 = 0$
   - $cD_{-} : p1.p5.p7 = 0$

8. Delete any points marked for deletion.

Repeat process until no points are deleted.
For more a more in depth explanation of the algorithm, see [1].

Initially a simpler algorithm[2] was used in order to create the skeleton using morphological operations to perform thinning. However, this algorithm did not ensure that the skeleton created was minimal or connected, and was therefore insufficient.

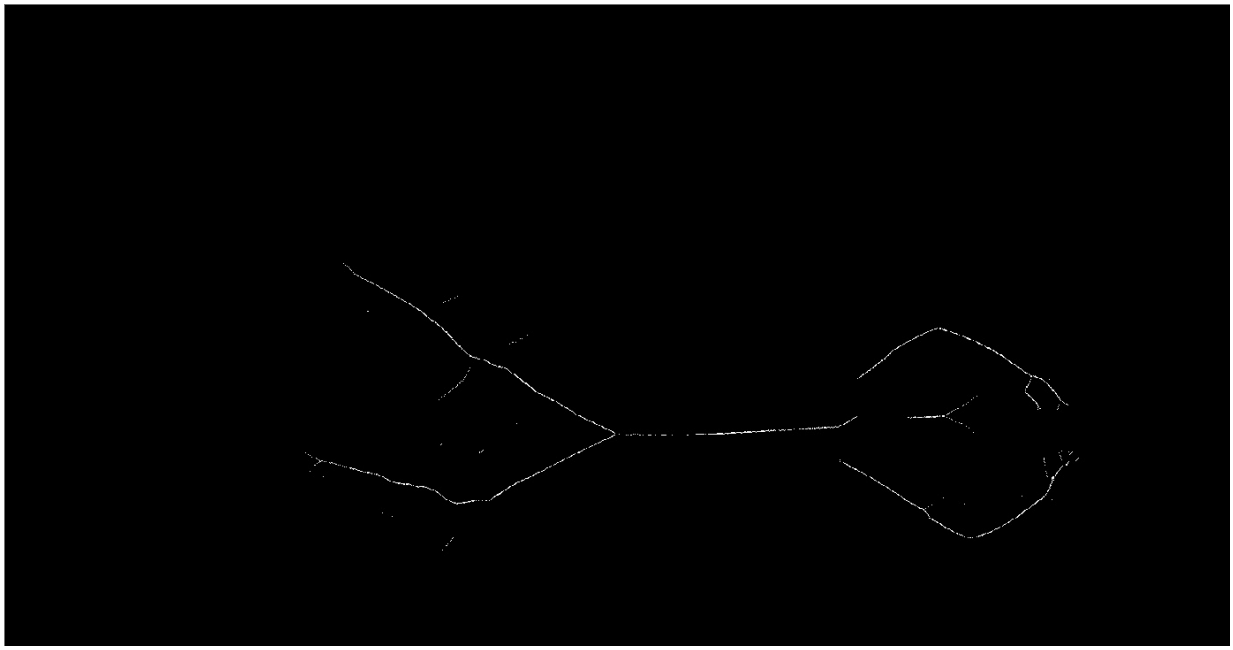


Figure 11: Result of using the morphological thinning algorithm on the binary image shown in ***figure 13b***.

4.2.2 Skeleton Simplification Algorithm

The RDP algorithm is designed to simplify a curve, while still maintaining its overall shape. A curve can be thought of as a collection of linked line segments, with each line segment stretching between two consecutive points on the curve. The basic principle of the algorithm is to reduce the number of line segments in a curve; creating a similar, but simplified curve. The algorithm is initially given an ordered array of all the points in the curve. The algorithm is also given the variable ϵ; this defines the degree to which the curve should be simplified and represents a minimum distance between line segments on the curve. It marks the start and end points of the curve as the first line segment (L), and then finds the point furthest from this initial line segment. If the distance from this point to the line segment (d) is less than ϵ then it is marked for removal. If the point is further than ϵ then it is kept and the algorithm recursively called with this point as the new end point. In this way the algorithm is able to remove every point that is closer than ϵ to the line segment that joins its neighbours.

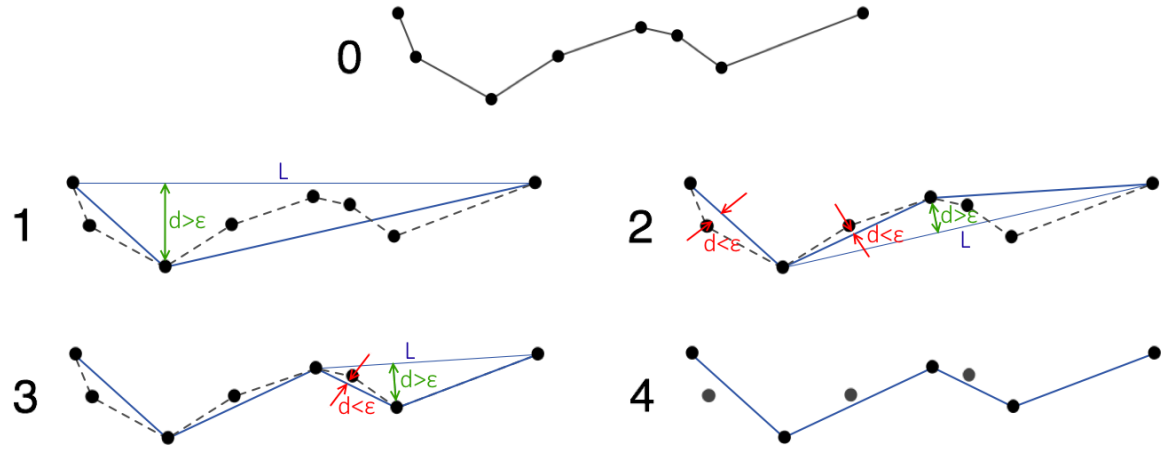


Figure 12: Diagram showing the RDP algorithm working on a curve. 0 being the initial curve to be simplified, and 4 being the fourth result the algorithm. Red arrows show points marked for deletion, green arrows show points to be kept
Modified from `http://upload.wikimedia.org/wikipedia/commons/9/91/Douglas_Peucker.png`.

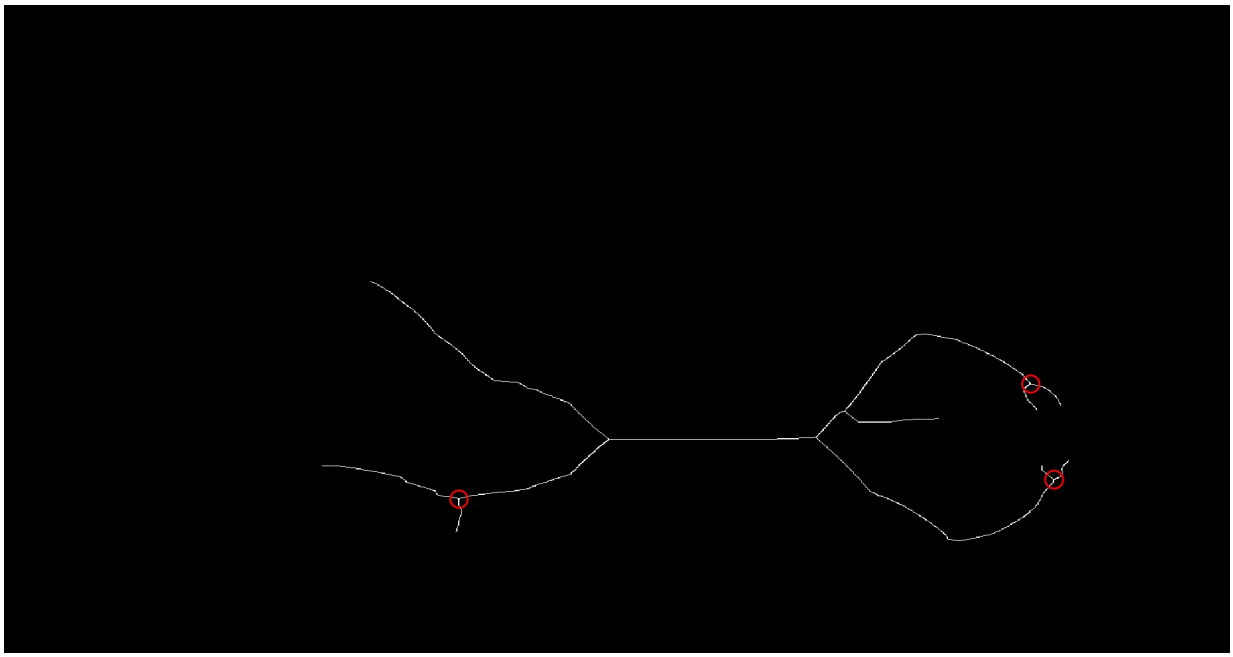
The RDP algorithm is used to reduce the number of points in the skeleton, simplifying it; however, it is only able to simplify curves with no bifurcations. This proves to be a problem when trying to simplify a skeleton produced by the skeltonisation algorithm, as it will always have bifurcations at the waist and neck. It may also also produce erroneous bifurcations at other points on the skeleton. These bifurcations can be seen in ***figure 13c***. The location of these bifurcations can be detected by applying a modified version of condition cB from the skeletonisation algorithm. Tp0 is calculated in the same way, by looking at the number of 0-1 transitions in the sequence p1, p2, ... , p8.

p8	p1	p2	y
p7	p0	p3	
p6	p5	p4	↓
x		→	

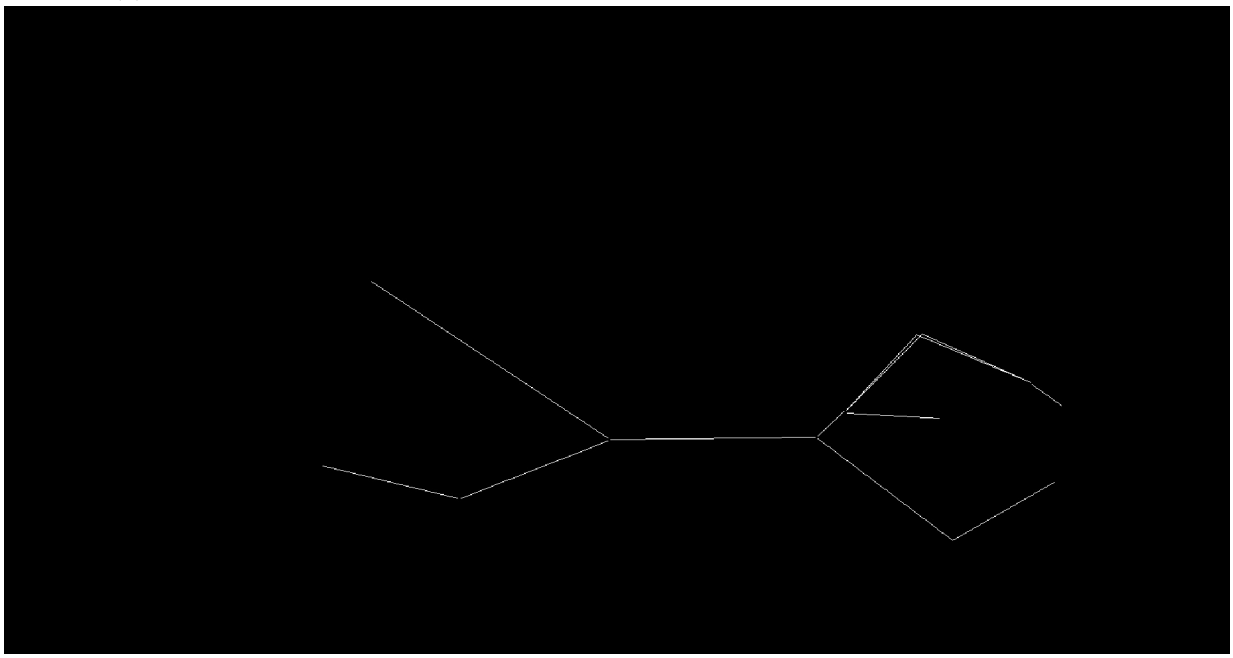
If $Tp0 > 2$ then a bifurcation is present at p0. This is always true, as the skeletonisation algorithm ensures that the skeleton is a single pixel thick 8-connected component.
Once the location of every bifurcation is found, the skeleton can be split into separate curves, by removing the bifurcation point. The RDP algorithm is then run on each of these curves independently. The simplified curves can then be recombined to form a simplified skeleton. At this point it is also possible the remove the false skeleton sections created by the skeletonisation algorithm forming erroneous bifurcations. The simplest way to do this is to delete any simplified skeleton section that is shorter than a given length. This works as the false skeleton sections tend to be shorter than the correct ones. A more reliable method for detecting false skeleton sections could be used in the final design.

(a) Skeletonisation source image.

(b) Binary mask of source image.



(c) Result of skeletonisation function. Erroneous bifurcations are circled in red.



(d) Result of skeleton simplification algorithm.

Figure 13: Results of skeletonisation and simplification algorithms.

4.2.3 Background Subtraction

A background subtraction algorithm has been implemented using the OpenCV libraries[9]. However, this cannot be properly tested, as the stock wind tunnel footage currently being used is too noisy.

5 Project Management

In order to properly manage the project, weekly meetings are conducted between the student and supervisor; progress and possible next steps are discussed.
An initial Gantt chart was created in order to aid in time management ***(see appendix A)***. However, as the project progressed this Gantt chart was seen to be irrelevant and A new chart was produced ***(see appendix B)***.

5.1 Risk Management

A risk assessment was carried out in order to assess potential problems, and how they could be prevented/ resolved.

Potential Problem	**Loss (1-5)**	**Probibility (1-5)**	**Risk**	**Plan**
Unable to get suitable stock FS footage	4	3	12	Contact FS coaches to ask for stock footage. If this is not possible, record a training session in the wind tunnel with my 4-way FS team.
Source code lost due to home computer failure	5	1	5	All source code and project documentation is backed up on a private github repository.
Severe Illness	4	1	4	Time is factored into the initial Gantt chart in order to account for unforeseen problems, such as illness.
Creating an active shape model is more difficult and time consuming than expected	4	3	12	More time can be contributed to the project, should this be the case.
Fitting a stick man model to a skeleton is more difficult and time consuming than expected	4	3	12	More time can be contributed to the project, should this be the case.
No project work is completed during the exam period, and the project falls behind schedule	3	4	12	After the exam period, extra time can be contributed to the project.

References

[1] Rafael C. Gonzalez & Richard E. Woods, "Skeletons", Digital Image Processing – Second Edition — International Edition (2001), pg650-653

[2] Rafael C. Gonzalez & Richard E. Woods, "Skeletons", Digital Image Processing – Second Edition — International Edition (2001), pg543-545

[3] Rafael C. Gonzalez & Richard E. Woods, "The Laplacian", Digital Image Processing – Second Edition — International Edition (2001), pg581-585

[4] Urs Ramer, "An iterative procedure for the polygonal approximation of plane curves", Computer Graphics and Image Processing (1972), pg 244–256

[5] David Douglas & Thomas Peucker, "Algorithms for the reduction of the number of points required to represent a digitized line or its caricature", The Canadian Cartographer (1973), pg112–122

[6] Imperial College London, "Lecture 15: Principal Component Analysis", `www.doc.ic.ac.uk/~dfg/ProbabilisticInference/IDAPILecture15.pdf`, Last access: 9 Dec 2014

[7] Tim Cootes, E. R. Baldock, & J. Graham, "An introduction to active shape models.", Image Processing and Analysis(2000), pg223—248

[8] Piccardi, M., "Background subtraction techniques: a review," Systems, Man and Cybernetics, 2004 IEEE International Conference, pg3099—3104 vol.4, (10-13 Oct 2004), doi: 10.1109/ICSMC.2004.1400815

[9] OpenCV, "The OpenCV Reference Manual Release 2.4.9.0"(April 21, 2014), `http://docs.opencv.org/opencv2refman.pdf`, Last access: 09 Dec 2014

[10] Qt Project, "Qt Reference Pages", QtDoc 5.3, `http://docs.opencv.org/opencv2refman.pdf`, Last access: 09 Dec 2014

[11] "Automatic Detection of 2D Human Postures Based on Single Images", QtDoc 5.3, `http://docs.opencv.org/opencv2refman.pdf`, Last access: 09 Dec 2014

[12] Wachs, Juan P., Deborah Goshorn, and Mathias Kolsch, "Recognizing human postures and poses in monocular still images."(2009),

[13] Lu Xia, Chia-Chih Chen, Aggarwal, J.K., "Human detection using depth information by Kinect," Computer Vision and Pattern Recognition Workshops (CVPRW), 2011 IEEE Computer Society Conference on , vol., no., pp.15,22, 20-25 June 2011 doi: 10.1109/CVPRW.2011.5981811

[14] Yu Tsz-Ho, Tae-Kyun Kim, and Roberto Cipolla, "Unconstrained monocular 3d human pose estimation by action detection and cross- modality regression forest.", In Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on, pp. 3642-3649. IEEE, 2013.

[15] Christophe Doignon (2007), An Introduction to Model-Based Pose Estimation and 3-D Tracking Techniques, Scene Reconstruction Pose Estimation and Tracking, Rustam Stolkin (Ed.), ISBN: 978-3-902613-06-6, InTech, pg360-376, Available from: http://www.intechopen.com/books/scene_reconstruction_pose_estimation_and_tracking/an_introduction_to_model-based_pose_estimation_and_3-d_tracking_techniques

[16] Prabhu, Utsav, and Keshav Seshadri, "Facial Recognition Using Active Shape Models, Local Patches and Support Vector Machines."

Appendices

A - Initial Gantt Chart

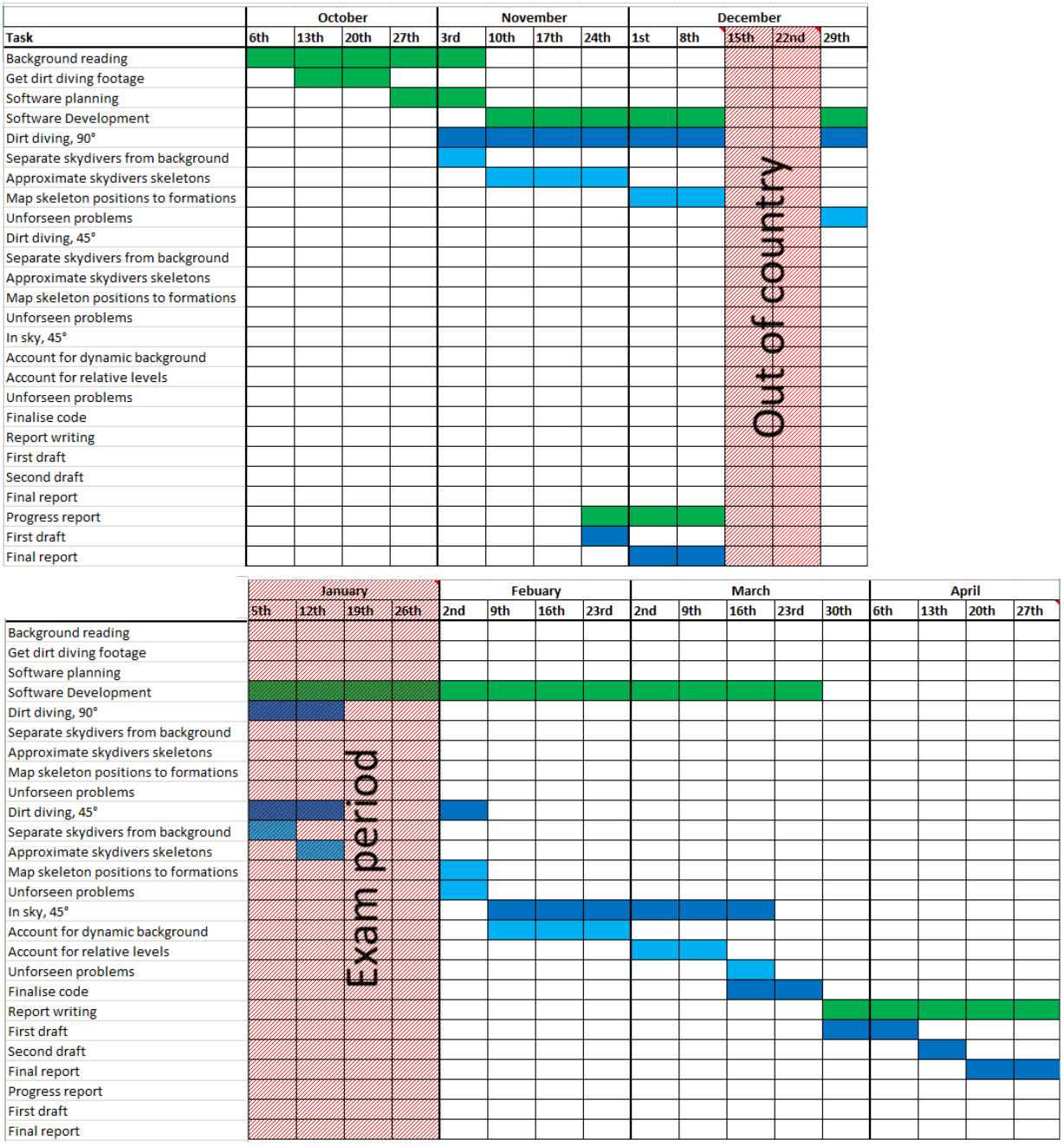


Figure 14: Initial Gantt Chart

B - Revised Gantt Chart

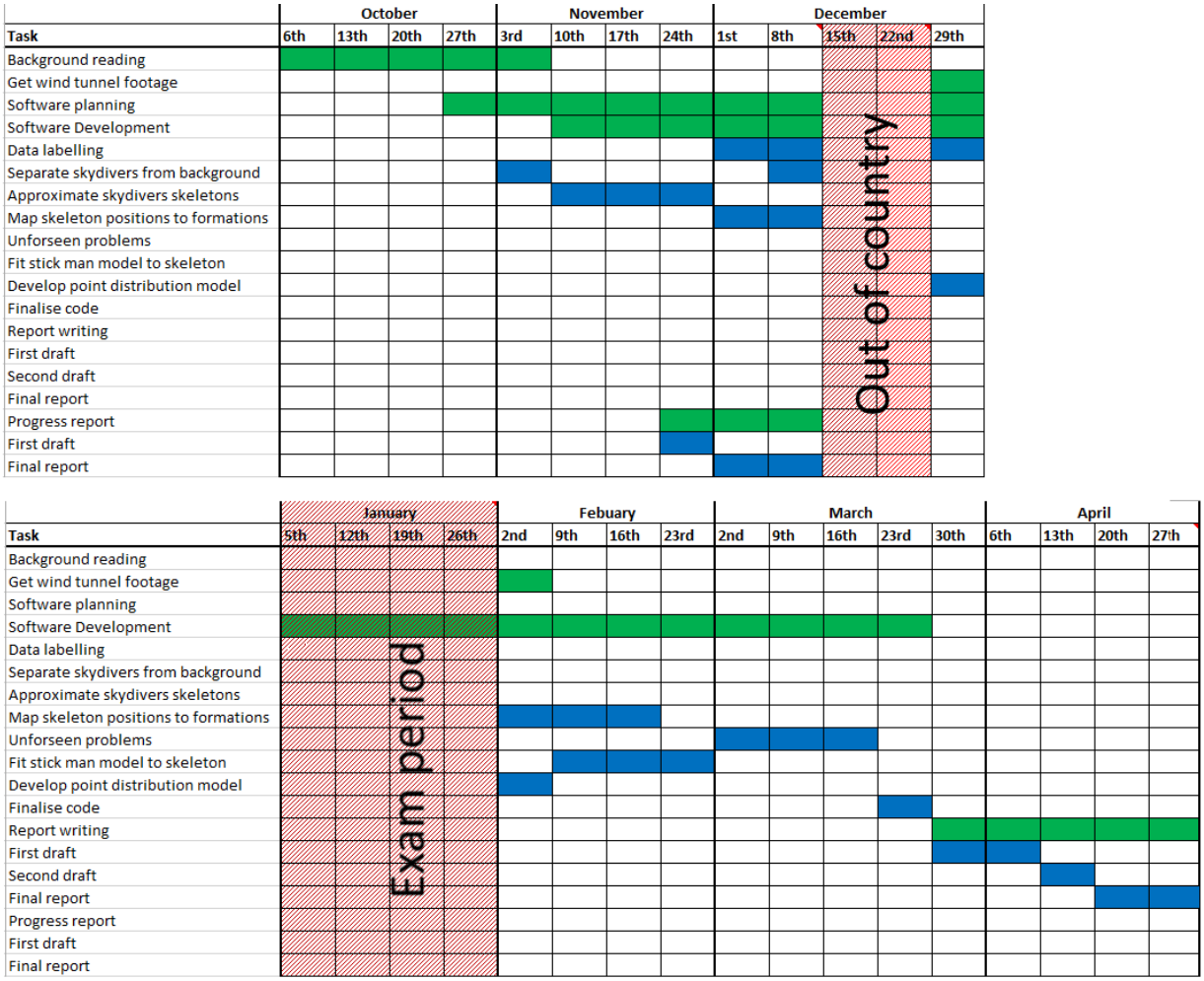


Figure 15: Revised Gantt Chart