University of Auckland

Department of Engineering Science

The Face is the Soul of the Body

Author:
Liam Morris

Supervisors:
Professor Poul M.F. Nielsen
Professor Martyn Nash
Dr Amir HajiRassouliha

October 2019

Abstract

Facial expressions are a fundamental part of human communication used to convey non-verbal cues during interaction with other people. Humans are adept at reading these cues, although they are difficult to capture with traditional camera-based methods. Analysis of these expressions is an increasingly important field with applications in psychology, animation, medicine, and security.

Recent advancements in the field of image registration have opened up opportunities for improvement in the modelling of facial expressions. A two-camera stereoscope was designed and constructed. The stereoscope is capable of capturing time-synchronous photographs of a subject transitioning from a neutral face to an expression. Stereo reconstruction was then used to estimate the 3D geometry of the neutral face, and individual pixels were tracked throughout the expression to produce point clouds with point-to-point correspondence.

This process was able to be applied to a simplified case of tracking rigid body motion on a highly textured rock, however, difficulties arose when applying tracking to facial expressions. It is suspected this is due to a lack of texture captured with the current setup. A principle component analysis pipeline was also developed which has the potential to extract dominant modes of deformations from expressions once accurate models are produced with point-to-point correspondence.

Acknowledgements

I would like to express my gratitude towards my supervisors, Professor Poul Nielsen, Professor Martyn Nash, and Dr Amir HajiRassouliha. I would like to thank Poul and Martyn for providing their expertise in the field and keeping me on the right track. I would like to thank Amir for his hands-on approach to supervising and explanations of complex ideas in layman's terms.

Many thanks also to my project partner, Gary Qian, for being a source of ideas and discussion throughout the project.

Finally, I would like to extend my appreciation to Mr. Sam Richardson, for providing his expertise with LabVIEW, and his patience when answering our many questions.

Declaration of Contribution

This report is my own work, and was not written in collaboration with any other person.

The stereoscope shown in Figure 4.2 (a) was designed with my project partner. The framework for fitting B-spline surfaces to point clouds discussed in Section 4.5 was developed by my project partner, and the extraction of principal components in Section 5.3 was done together. All other work discussed in the report is my own unless otherwise stated.

Contents

List of Figures

List of Tables

1 Introduction

Humans are capable of reading peoples' emotions by analysing facial expressions. However, some of these expressions are difficult to capture with traditional camera-based systems. The Auckland Bioengineering Institute (ABI) are world leaders in the fields of image registration and tracking and have developed a new method of image registration with subpixel accuracy [1]. Consequently, there is opportunity for the methods behind expression representation to be improved by using the latest techniques and technology.

This project aimed to develop a framework for reconstructing faces using a two-camera stereoscope, and evaluate the feasibility of applying the recently developed image registration algorithms for tracking skin during expressions. The project consisted of three stages.

The first stage was done in conjunction with my project partner, Gary Qian, and required designing and constructing a stereoscope. The stereoscope was then used to capture time-synchronous images of an individual forming expressions.

The second stage of the project was done independently, and involved reconstructing a 3D point cloud of the individual's neutral expression, and tracking skin during the expression to create subsequent point clouds with point-to-point correspondence. An overview diagram for this stage of the project is shown in Figure 1.1, which shows the stereo images through time, with tracking performed on sequential frames to maintain point to point correspondence of the point clouds reproduced.

Finally, combining the methods developed in our independent projects, we used principal component analysis (PCA) to demonstrate its potential to determine the dominant modes of deformation during expressions, and to represent facial expressions efficiently.

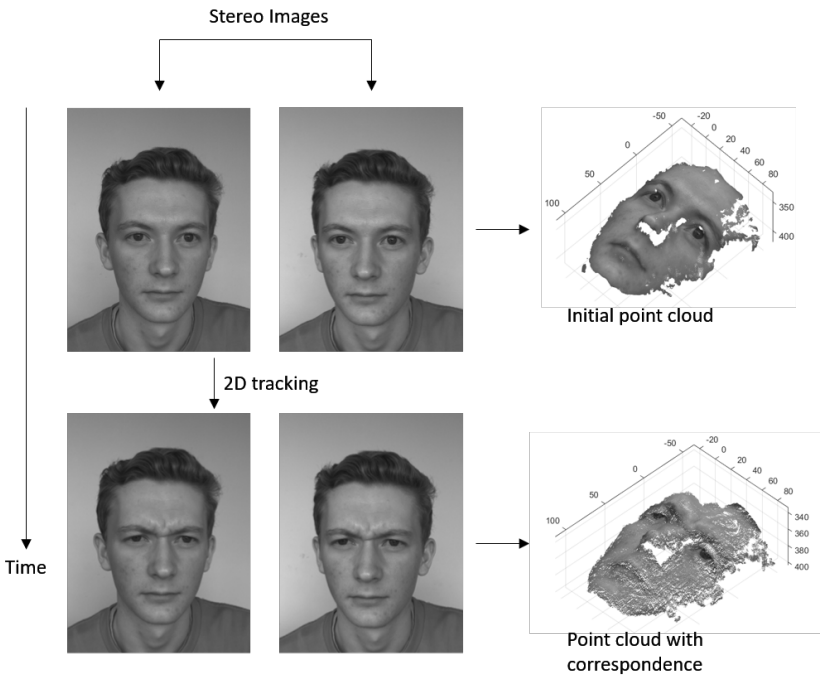


Figure 1.1: Project overview diagram

2 Literature Survey

2.1 Introduction

Facial expression analysis is a growing field of research, with many applications in psychology, animation, medicine, and security. It is therefore a heavily researched area, and the methods used for analysis are frequently improving [2].

This survey will summarise the methods used in literature for creating a 3D point clouds with point-to-point correspondence from video of a subject, the methods of constructing smooth surfaces from point clouds, and finally, techniques for reducing the parameters needed for representing 3D models.

2.2 3D Model Estimation

2.2.1 Structured Light

One of the common methods used in literature are structured light techniques [3]. By projecting an encoded light pattern onto the face of a subject (example shown in Figure 2.1) and measuring the shape of the pattern, a geometry can be estimated.

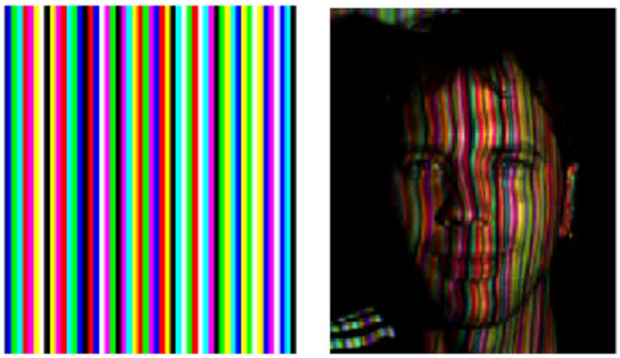


Figure 2.1: Illustration of structured light. (a) The light pattern to be projected. (b) The projection onto a subjects face. Images from [4]

These methods have been adapted to flash between patterned and white light to allow for colour film to be recorded at the same time, or to project infrared light outside the visible spectrum to avoid distracting the subject.

The disadvantages of this method include:

- Rendering of light patterns can slow the capture rate—though some high-speed systems have been proposed which address this [5].
- The subject is limited to movements within the light pattern.
- Parts of the face not covered by structured light lead to holes in the model.

2.2.2 Photometric

Photometric methods were first proposed in 1980 [6], but are still commonly discussed in literature. The premise of these methods is to illuminate the subject from different angles and capture images. These images can then be used to estimate the surface normals of the subject's face by inspecting the observed shadows. Integration must be performed to obtain a surface mesh from the normals, which results in extra computational time and potential errors [7]. Another clear disadvantage of this method is that the subject must remain still while the lighting conditions are adjusted. This has been addressed in [8] by illuminating the subject with different colours from different angles. However, this method requires laborious calibration, and the coloured lights can distract the subject.

2.2.3 Stereo-Cameras

Stereo-camera systems can synchronously capture a scene from multiple angles. Points of correspondence in the scene can then be found in features of the images and used to reconstruct a point cloud [9]. This method allows the subject to behave more naturally, and the images can be applied to the 3D model as texture maps. Although, this requires the use of image registration algorithms (see Section 2.2.3) [1, 10, 11], as well as calibration of intrinsic (those relating an object to its image in the camera image plane) camera parameters and extrinsic (those relating the cameras in a global coordinate system) camera parameters. The calibration must also be done offline as this process is computationally expensive, with current algorithms taking over twenty minutes [10], although this typically needs to be done only once. A recent method by HajiRassouliha et al. [12] has been proposed for achieving accurate calibration of multi-camera stereo systems.

Image Registration

Image registration (also called image alignment) is a process that transforms sets of images to be in terms of a common coordinate system. Image registration has applications in medical imaging, video stabilisation, military target recognition, etc. A. Ardeshir Goshtasby details multiple methods which either rely on evaluating an objective function based on patterns of colour concentration or by identifying features of correspondence which may include points or contours [10].

For applications in facial modelling, a simple method of creating points of correspondence is marking spots on the subjects face. However, the recently developed method at the Auckland Bioengineering Institute has been used to track jugular venous pulse [1, 11]. The algorithm is able to identify features on the skin to use as points of correspondence.

Disparity maps

Disparity maps are built from two images of a scene, and represent horizontal shift of corresponding pixels between the two images. They are a common method of creating 3D point clouds [9]. Algorithms for computing disparity maps rely on the image pair coming from horizontally aligned, parallel cameras [13]. However, image rectification can be used to apply disparity methods to generalised multi-camera stereoscopes [14]. Rectification projects images onto a common plane, such that points of correspondence are aligned in the horizontal plane (have the same vertical coordinate). This makes the images appear as though they have been taken by horizontally aligned, parallel cameras.

Through camera calibration and disparity maps, pixels can be projected into a 3D point cloud representing the scene captured in the two images [13]. Methods for creating disparity maps are comprised of two elements: the first is the algorithm used to search for the matching pixels between images; the second is the image registration algorithm used to measure similarity of pixels (or pixel groups) between the images, and hence calculate a matching cost [13].

Searching Algorithms

Searching algorithms are further divided into local and global methods [15]. Local methods only consider the intensity values of a local window of pixels during the search, providing low computational complexity and short run times. The disadvantage of these methods is that the winner take all optimisation can lead to noisy disparity maps, especially in scenes with lots of uniform texture. Another shortcoming of local algorithms is that they commonly assume pixels within the window of support have similar disparity values. This is often not true, especially for faces which have sudden and large changes in depth around the nose and eyes.

Global methods address this by assigning disparity values by minimising an energy function designed to smooth the disparity map. Such energy functions typically include one term for matching costs similar to local methods, and one term to apply penalties for changes in disparity between neighbouring pixels. In contrast, these methods are computationally expensive and are not practical for real-time systems.

Semi-global methods [16] have also been proposed to reduce the computational effort required by minimising over a subset of the image, e.g., each horizontal line of the disparity map. These methods produce smoother maps than those from local matching with minimal addition to the computational effort required. Implementations of these algorithms typically use dynamic programming or belief propagation to enforce smoothness of disparity maps [13].

Matching Costs

The criterion for measuring match quality during the search is crucial to the overall performance of the algorithm. The criterion must be efficient as it is called many times during the search. It must also be consistent across intensity values so that matching costs are not dependent on how bright the images are [13]. Typical methods used are the sum of absolute differences (SAD) and sum of squared differences (SSD). Although, other metrics such as normalised cross correlation (NCC), census transforms (CT), and structural similarity index (SSIM) are also discussed in the literature. These methods are computationally expensive, but are more reliable for images taken with poor lighting conditions [13].

2.2.4 Time-of-Flight Cameras

Time-of-flight cameras are another frequently discussed method for depth estimation in the literature. They estimate depth by projecting structured light and measuring the time taken to travel from the camera, to the object, and back to a sensor [17]. They are therefore subject to noise from background light, which can interfere with reconstruction accuracy. Another drawback is that emitted light may be reflected from multiple surfaces, causing inaccuracies in depth readings.

2.3 Surface Reconstruction

The goal of surface reconstruction is to approximate an unknown surface from a set of points in $\mathbb{R}^3$. These points may not lie exactly on the true surface. Therefore, surface models that use a 'best fit' approach are common in the literature. This is referred to as an implicit surface,

as opposed to an explicit surface which joins all points [18]. Implicit methods are typically parametric models fitted to the raw point cloud (B-spline, NURBS, etc.). Whereas triangular meshes are a typical example of explicit methods.

Triangular methods are the core part of reconstruction algorithms [19], and the most commonly used for representing faces [4, 10]. These methods connect neighbouring points to form a surface mesh. There are various triangulation methods discussed in [20, 19, 21] such as Delaunay, Voronoi, and Crust. The literature also discuss the various methods of tackling the problems that may be encountered when attempting to fit a surface to a point cloud. These problems are shown in Figure 2.2.

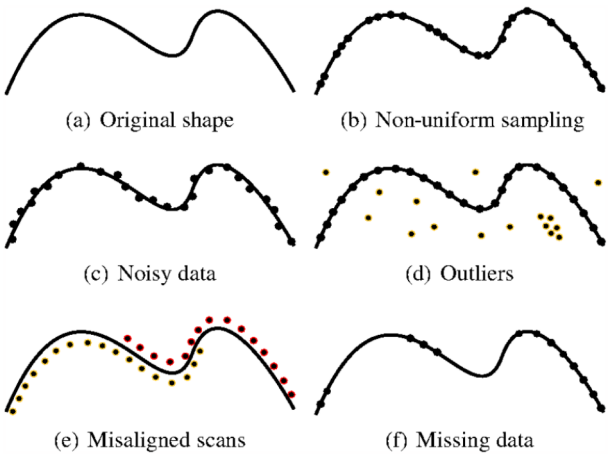


Figure 2.2: Different artefacts shown on a 2D curve. Images taken from [21]

2.4 Feature Reduction

Feature reduction methods are used to create lower-dimensional representations of data for efficient processing and analysis. This can be done by removing irrelevant or redundant features, or by amalgamating features into more significant predictors. Principal component analysis (PCA) is widely used in literature for dimensionality reduction and is often employed in 3D facial recognition methods. PCA converts data to be in terms of linearly uncorrelated variables (principal components), by projecting the data into a lower dimensional space. One drawback of PCA is that it assumes variables are related to each other linearly [22]. Eigenmaps and ISOMAPs are also discussed in literature, which are non-linear methods for feature reduction [23], although these require optimisation, and results may be less interpretable.

3 Statement of Research Intent

This project aims to develop a framework for reconstructing faces using a two-camera stereoscope. Dense point clouds will be reconstructed using stereo reconstruction algorithms. The image registration algorithms developed in [1] will then be used to attempt to track the skin during filmed expressions. This pipeline will later be used for determining the number of parameters required to efficiently represent facial expressions, and investigate trade-offs between model accuracy and the number of parameters used.

3.1 Resources

Two FLIR Flea 3 monochrome cameras will be used for recording our expressions. These cameras can record 1280×1024 video at up to 150 fps. The cameras will be equipped with a 6 mm DF6HA-1B Fujifilm lens. A stereoscope will be constructed using the two cameras, mounted onto a tripod with an acrylic base.

LabVIEW [24] will be used to record video with the cameras and vary the image capture settings. Therefore, a licence will be required for this.

A MATLAB licence with the Computer Vision Toolbox [25] will be required to use the ABI's newly developed image registration algorithms.

Regular meetings with supervisors Professor Poul Nielsen, Professor Martyn Nash, and Dr Amir HajiRassouliha will be crucial for staying on track with the project, and their expertise in the field will be vital to the success of this project.

3.2 Methodology

Image Acquisition

Using the Flea 3 cameras, a two-camera stereoscope will be designed and constructed. LabVIEW will then be used to capture expression videos with the stereoscope.

Geometry Estimate

Stereo reconstruction algorithms will be used to construct disparity maps and point clouds of the initial geometry.

Deforming the Geometry to Mimic Expressions

The frames obtained in the image acquisition step will then be fed through the ABI's image registration algorithms to determine the 2D vectors of deformation. Once these are obtained, the same deformations can be applied to the 3D model to mimic the facial expressions in the recordings.

3.3 Validation and Verification

The success of our project can be measured in the first instance by inspecting the quality of the 3D models that are produced. This could be done visually, but also by taking measurements of the models produced and comparing them to the real object.

3.4 Limitations of Scope

Some potential areas to be explored in subsequent projects could be:

Fitting parametric surfaces to point clouds

Fitting a parameterised surface model to point clouds reducee the number of features required to represent the facial expressions. It will also provide smoother models, which will serve to remove noise from point clouds.

Determining a set of Basis Features

PCA could be used to extract the underlying components that describe facial expressions. This would allow the modelling of expressions as a linear combination of basis features.

3.5 Significance of Research

Development of a pipeline that can take a video recording of a subject making various expressions, and create 3D models of the expressions would be a useful tool for future research at the Auckland Bioengineering Institute. It will also begin the first phase in examining methods for efficient expression representation.

4 Methodology

The methodologies for the project are divided into four sections; image acquisition, 3D reconstruction, tracking, and analysis. They are explained in more detail below.

4.1 Image Acquisition

4.1.1 Intel RealSense

The Intel RealSense D435i is an off-the-shelf stereo camera system which includes two monochrome cameras, an infrared projector, and an RGB camera. This camera was purchased prior to the commencement of our project and was the intended medium for capturing our stereo images.

Figure 4.1: RealSense Camera. Image from [26]

In order to interact with the cameras, a Python interface was designed. This interface allowed the user to control image capture settings such as the camera's frames per second (fps), resolution, and turning the infrared projector on/off.

Using our Python interface, we successfully captured images for initial testing of the image registration algorithms described in Section 4.4. However, after extensive testing with lighting conditions, recording at various distances, and varying parameters in the tracking algorithm, it was found that the results of applying the algorithms to these images were unsatisfactory for our applications. The fixed focal length of the Intel cameras would not allow us to be close enough to the cameras to capture the required texture detail while still being in focus.

As the Intel cameras were found to be unsuitable, it was decided that the project progress using FLIR Flea 3 cameras, as these can be equipped with different lenses to adjust focal length, while still meeting the expected fps and resolution requirements.

4.1.2 Flea 3 Stereoscope

Design and Construction

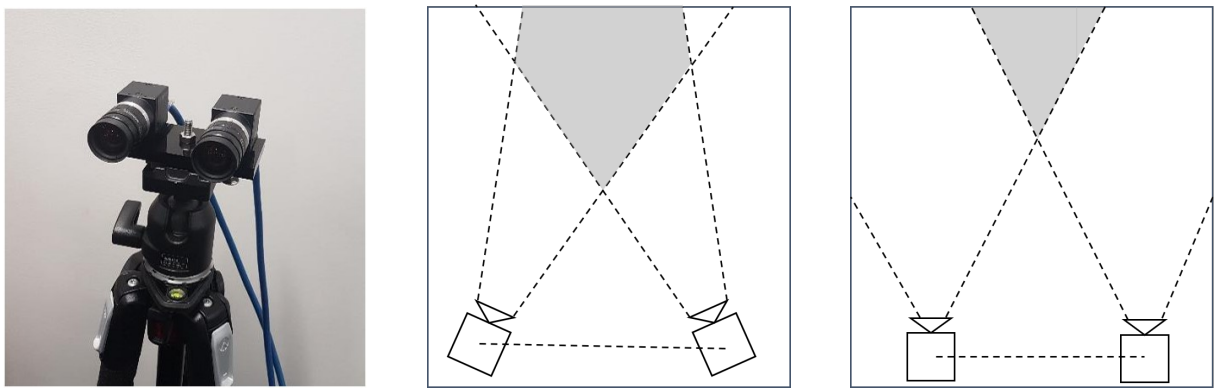
Using two FLIR Flea 3 cameras, we designed and built our own stereoscope, shown in Figure 4.2 (a). Preliminary tests were performed by equipping the cameras with different lenses (with focal lengths of 6 mm and 12.5 mm) and assessing the texture detail in photos at different depths. From this, it was determined that there needed to be less than twenty centimetres between the cameras and the subject for sufficient detail to be captured. In general, a shorter focal length increases the field of view (FOV), but causes more lens distortion. We found that the 12.5 mm focal length did not provide an adequate FOV to capture the entire face at close distances, and

therefore the 6 mm lens was used. The increased distortion caused by the shorter lens were addressed using the methods described in sections 4.2 and 4.3.1.

Choosing the physical placement of the cameras involved determining a suitable baseline (distance between the cameras) and camera angle. Increasing the angle of the cameras provides more overlap of the camera FOVs, and allows more areas of the face to be captured. However, if the angles are too steep, slanted regions (areas of the face where the angle between cameras and surface normals are large) of the face may not appear in both photos, and consequently will not be able to have their depth measured. This trade-off is visualised in Figure 4.2 (b) and (c).

10 mm acrylic sheets were used for the base of our stereoscope because it is readily available, easy to cut, and sufficiently rigid for our purposes as the cameras will not be moving during image capture.

Three different base plates were constructed, with the cameras 50 mm, 75 mm, and 150 mm apart. Together, with some variation in distance, this allowed for experimentation with a range of angles. When creating the point clouds discussed in section 4.3, it was found that the angles required on the 150 mm base plate were often too steep, resulting in loss of depth resolution around the nose and eyes. Using the smaller base plates captured these areas better, but resulted in a loss of information near the sides of the face. These areas of the face are expected to be less critical for representing expressions, so the smaller base plates were preferable.



(a) Our stereoscope, fitted with the 75 mm base plate

(b) Diagram showing view overlap with angled cameras

(c) Diagram showing view overlap with parallel cameras

Figure 4.2: Stereoscope design

Image Capture

Flea 3 cameras are routinely used by researchers in the ABI, and the preferred software for interacting with the cameras is LabVIEW. Existing code for interacting with an older version of the FLIR cameras was provided obtained from a previous project [11]. This code was adapted and updated to work with our stereoscope, and an interface was designed to allow the user to vary the camera settings; fps, gain, exposure time, and the directory for saving images and video.

When our interface was used to capture images, it was observed that there were uniformly distributed lines on the images which had a higher intensity than the rest of the image. These are shown in Figure 4.3. The lines caused the image registration algorithms discussed in sections 4.3.2 and 4.4 to fail and halted our progress on the project. It was suspected that the lines were due to the indoor artificial lights in flickering at high frequencies, but testing the cameras in sunlight proved this was not the case as the lines still appeared. Consulting with our supervisors, it was suggested that radio frequency interference had caused similar problems in the past, and could be causing the lines, However, changing our environment to test this theory also did not remove the lines.

It was eventually determined that the source of the lines was an internal sensor artefact that was imposing fixed pattern noise on the images. The newer cameras have a correction for this hardcoded into the firmware. The codes we inherited had this correction disabled as it had previously caused issues on the older firmware versions. Re-enabling this correction removed the lines from images and we were able to progress with the project.

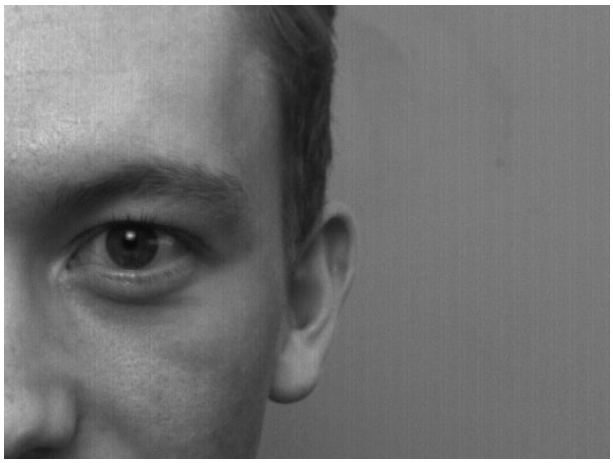


Figure 4.3: Vertical lines caused by fixed pattern noise

4.2 Camera Calibration

Camera calibration is the process of estimating the parameters of a stereo camera system. These are the intrinsic (focal length, optical centre, and lens distortion), and the extrinsic (camera location and orientation) parameters.

A checkerboard pattern was printed onto a sheet of acrylic for estimating the parameters of our stereo system. Approximately one hundred photos were taken of the checkerboard at various angles, positions, and depths with our stereo cameras.

MATLAB [25] and OpenCV [27] (an open-source computer vision library) were used to estimate the parameters of our stereoscope. OpenCV detects the corners of the checkerboards in each photo. Because the dimensions of the squares are known, the camera parameters can be estimated. A method for optimising camera parameters has been developed at the ABI [12], and was used to achieve accurate calibration. The calibration procedure consists of taking initial estimates, and then measuring reprojection error with three metrics:

- Traditional back-projection errors—projecting of the corners into 3D space, then back onto the images, and observing the error induced.
- Absolute 3D length error—projecting corners into 3D space, and observing the error in average square size.
- 3D shape error—projecting the corners into 3D space, and observing the error between the points and the plane that best fits the points.

A trust-region method [12] is then used to optimise these three error functions. This process takes approximately two to three hours to complete, and was required every time the parameters of the stereoscope changed. This can occur if the cameras are moved, or if the focus is changed.

4.3 Reconstruction

Using two photographs of a scene, and the parameters of the stereo camera system that captured them, a 3D representation of the scene in the photographs can be reconstructed.

This is achieved by locating matching features from the scene in both images and measuring its shift (in pixels) between the two images. If the location of the stereo cameras in space, their orientation, and other similar parameters are known, the depth of the feature can be calculated using projective geometry.

The original plan for this project was to use the image registration and 3D reconstruction algorithms developed at the ABI for our reconstructions. However, the algorithm assumed smooth changes in depth and slope for its reconstruction process, and was therefore unable to capture areas with sudden changes like the nose and eyebrows. Attempts were made to resolve this by reconstructing the face piece by piece. However, this led to inaccuracies at joins, and still failed to capture all regions of interest. Alternative methods for 3D reconstruction using disparity maps were therefore explored, with a focus on keeping the methods generalised so that more cameras may be added in the future.

4.3.1 Image Preparation

Some pre-processing steps are used to prepare the images and simplify the reconstruction process. These steps are the removal of lens distortion, and the rectification of images.

Undistortion

Lens distortion is an intrinsic parameter that was estimated for each camera during the calibration. OpenCV [27] was used to remove lens distortion from the images using the estimated lens distortion parameters.

Rectification

Stereoscopic cameras generally view a scene from different angles and locations. Because of this, the scene will appear different in the two images. For example, the left cheek will be closer to the left camera, and will consequently appear larger than it does in the right camera.

Image rectification can be used to correct this discrepancy. Rectification projects images onto a common plane, such that points of correspondence are aligned in the horizontal plane (have the same vertical coordinate). This makes the images appear as though they have been taken by horizontally aligned, parallel cameras. The simple case of horizontally aligned, angled images can be visualised, as in Figure 4.4. The red and green planes show how the observed scene (blue plane) is seen from the perspective of the two cameras. The boxes in the back show how rectification brings the images onto a common plane.

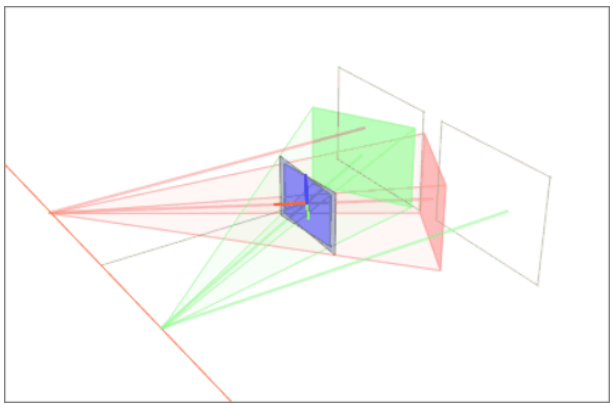


Figure 4.4: Rectification for horizontally aligned cameras. Image from [28]

Rectification was performed using MATLAB's `rectifyStereoImages()` function [25], which is passed the two images and the camera parameters as inputs. The function returns projective transforms to rectify the images, as well as the reprojection matrix Q, which is used in section 4.3.3 to project image points into 3D coordinates in the rectified system. A projective transform, H, can be applied to an image with Equation 4.3.1 [14]:

$$H = \begin{bmatrix} R_1 & R_2 & T_x \\ R_3 & R_4 & T_y \\ P_1 & P_2 & 1 \end{bmatrix}, \qquad \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = H \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}, \tag{4.3.1}$$

where:

- R_i form a rotation matrix
- P_i form a projection vector
- T_i form a translation vector
- x, y represent pixel coordinates in the original image
- x', y' represent corresponding coordinates in the rectified image

The reprojection matrix is of the form

$$Q = \begin{bmatrix} 1 & 0 & 0 & -c_x \\ 0 & 1 & 0 & -c_y \\ 0 & 0 & 0 & f \\ 0 & 0 & \alpha & \beta \end{bmatrix},$$

where c_x and c_y are the coordinates of the focal centre, α and β are constants from the rectification process, and f is the focal length.

For cameras in arbitrary positions, rectification generalises to an optimisation problem, where a projective transform must be identified from matching points in photos from each camera. This can also be done with checkerboards. This method is therefore generalisable to stereoscopes with more than two cameras that are not horizontally aligned.

Figure 4.5 shows a pair of rectified images. Each pair of coloured lines intercept the same features in both images, as rectification has aligned the vertical axis of the images. This simplifies the correspondence problem as it reduces the search space from 2D to 1D, since all features will have the same vertical coordinate.

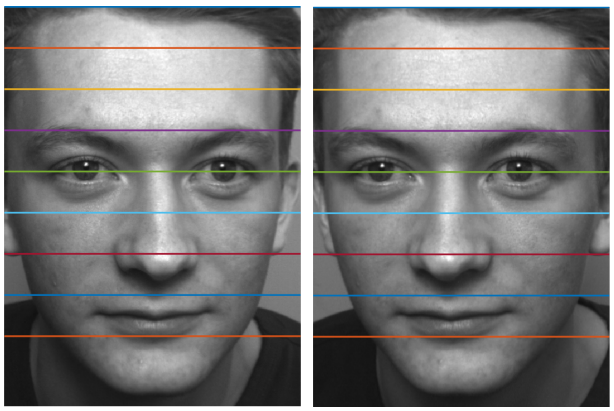


Figure 4.5: Horizontal feature alignment from rectification

Cropping

After rectification, the images were cropped down by manually specifying the coordinates of the bounding box that encloses the region of interest. This reduced the search space and controlled disparity ranges for algorithms described in 4.3.2.

4.3.2 Disparity Maps

The disparity map of a scene is the same size as the stereo images, but each pixel value represents the difference in horizontal coordinates between the images. Pixels with high disparity are closer to the cameras, and pixels with lower disparity are farther away [13]. Disparity maps can be used to reconstruct a 3D scene from rectified images.

Algorithms for computing disparity maps have two parts: the first part determines how the algorithm searches the images for matching pixels; and the second is the metric used for measuring the similarity of subimage pairs. These metrics are the simpler part of the process, and commonly use one of the following methods [13]:

For two subimages S_1 and S_2, of size $(2W+1) \times (2W+1)$,

Sum of absolute differences (SAD) is calculated as:

$$SAD = \sum_{-W \le i \le W} \sum_{-W \le j \le W} |S_1[x+i, y+j] - S_2[x+i, y+j]| \,. \tag{4.3.2}$$

Sum of squared differences (SSD) is calculated as:

$$SSD = \sum_{-W \le i \le W} \sum_{-W \le j \le W} (S_1[x+i, y+j] - S_2[x+i, y+j])^2 \,. \tag{4.3.3}$$

Alternatively, a structural similarity index measure (SSIM) [29] could be used. The aim for this section of the project was to find the best method for consistently finding good disparity maps. Various versions of disparity algorithms were implemented and tested to do this, including commercial and open-source algorithms from MATLAB [25] and OpenCV [27]. Some code was also adapted from [30]. However, many of these methods were found to perform poorly and are omitted from this report. The final algorithms used are described below.

Block Matching

Block matching [14] is the simplest approach for computing a disparity map. It works by taking a $w \times w$ window (usually square, but this is not a requirement) in one image, and searching for the best matching window in the other image. This gives a disparity value for the pixel at the centre of the matched windows. Completing this search for every possible window in one image then gives a complete disparity map.

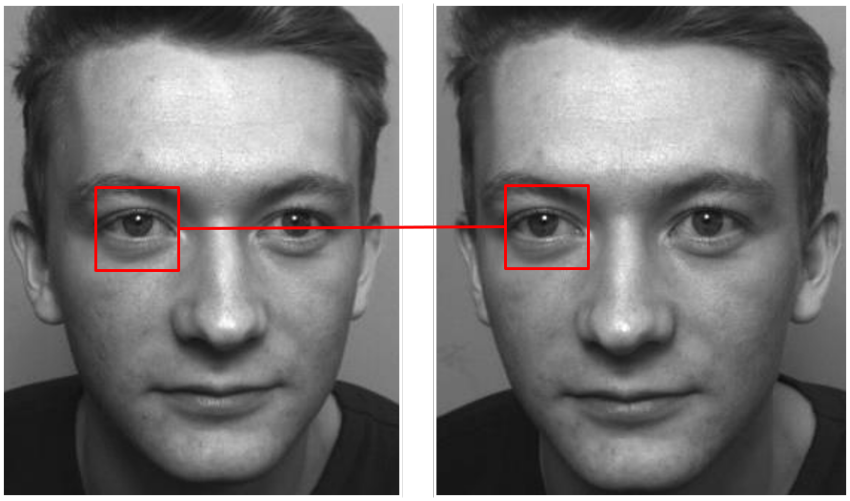


Figure 4.6: Example of potential matched block from block matching

For a given image similarity measure, S (SAD, SSD or similar), block matching can be thought of choosing $D(x, y)$ such that Equation 4.3.4 is minimised

$$\sum_y \sum_x S\left(I_1[x, y, w], I_2[x + D(x, y), y, w]\right), \tag{4.3.4}$$

where:

- I_1 and I_2 are the two rectified images.
- $I_1[x, y, w]$ represents the $w \times w$ subimage of I_1, centered at x, y.

Restrictions can be placed on D to reduce the search space and improve efficiency. For example, if we constrain $D(x, y) \in [0, 64] \quad \forall x, y$, then each window in I_1 is compared with 65 windows in I_2. This was useful for our images as they are typically 600 pixels wide, even after cropping down to the region of interest. A confidence threshold can also be included by flagging pixels which did not have an acceptable match in the second image. These pixels can then be excluded from the disparity map to have higher accuracy at the cost of depth resolution.

Block matching can also be modified to find subpixel disparities. This is done by finding the minimum matching cost, and the matching costs of the two neighbouring pixels. Fitting a parabola to these three costs allows the turning point to be used as a (usually) fractional disparity value.

Block Matching With Dynamic Programming

Block matching can be improved by including smoothness penalties with dynamic programming [15]. During the search for matching features performed in regular block matching, the similarity

of windows is stored in a table to be used as a 'cost' of matching each window. When this table is acquired for each scanline (row), the matching costs can be weighted with a smoothness cost (based on change in disparity between neighbouring pixels) and each scanline of the disparity map is then optimised through dynamic programming.

Mathematically:

$$D(1,1) = \underset{d_L \leq d \leq d_U}{\operatorname{argmin}} \{C(x+d, y)\}$$
$$D(x,y) = \underset{d_L \leq d \leq d_U}{\operatorname{argmin}} \{C(x+d, y) + P\left[D(x+d+1, y) + D(x+d-1, y) - 2D(x+d, y)\right]\}$$

where P is the penalty per change in disparity between neighbouring pixels, and $C(x-d, y)$ is the image matching cost, calculated with SAD, SSD, or similar.

For the example matching costs in Table 4.1, regular block matching will choose the path outlined by the red dots (disparity $= 1, 1, 0, 1, 1$) as these are columns with the minimum matching costs. Dynamic programming with $P > 0.5$ will instead choose the path outlined by the blue dots (disparity $= 1, 1, 1, 1, 1$). For this case, dynamic programming has smoothed the disparity values by removing the outlier. Note that the ∞ in column one is there to enforce feasibility as a negative disparity would leave the bounds of the image.

For example:

Table 4.1: Matching costs for disparity range $-1 \leq d \leq 1$ for each column

Example Cost Table			
	Disparity		
Column	-1	0	1
1	∞	4	3 ● ●
2	4	4	3 ● ●
3	4	3 ●	4 ●
4	4	4	3 ● ●
5	4	4	3 ● ●

subpixel disparities disparites can be found by applying the same principles as in regular block matching, as described in Section 4.3.2

Semi-Global Matching

OpenCV has an implementation of block matching with dynamic programming that optimises over eight directions instead of two (up, down, left, right, and diagonals) as an approximation of global optimisation.

This version differs slightly to the implementation above as it applies fixed costs for changes in disparity, rather than a multiplicative cost. The algorithm applies a cost P_1 for smaller disparity changes, and a larger penalty P_2 for large changes in disparity. The algorithm is also built in C, which means it runs much faster, and therefore makes it easier to tune parameters. The full algorithm is described in detail in [27].

4.3.3 Projection

Once a satisfactory disparity map is found, the reprojection matrix (found via image rectification in Section 4.4), Q, is used to reconstruct the point cloud by using Equation 4.3.5.

$$w \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = Q \begin{bmatrix} x \\ y \\ d \\ 1 \end{bmatrix} \tag{4.3.5}$$

where x, y, d are the pixel coordinates and associated disparity, X, Y, Z are the corresponding world coordinates, and w is a normalisation parameter

Note that point clouds produced using this method are consequently in the rectified coordinate system. This is corrected during the back-projection steps discussed in Section 4.4.1.

4.3.4 Post-Processing

Moving Average for Smoothing

In order to improve the smoothness of the reconstructed point clouds, a weighted moving average smoother was developed. The smoothing algorithm works as follows:

1. Choose an $n \times n$ template matrix, $\mathbf{x}$ (n must be odd and ≥ 3). This will be rotated in later steps, so asymmetric matrices were typically used. For example:
$$\mathbf{x} = \begin{bmatrix} 1 & 2 & 1 \\ 1 & 3 & 1 \\ 1 & 2 & 1 \end{bmatrix}.$$
2. Extract an $n \times n$ window from the disparity map.
3. Flag and ignore disparity values very different to the centre value (typically a threshold of 2-3 was used). This means that areas with high gradient are not flattened by the smoothing.
4. From the valid disparity values, calculate the average second derivative (using finite differences) in the horizontal and vertical direction, δ_H and δ_V respectively.
5. If $|\delta_H| > |\delta_V|$, assign higher weights to the disparities in the vertical direction, or vice versa. This is done by rotating the template matrix $\mathbf{x}$ so that weights are greater in the desired direction. For example:
$$\mathbf{w} = \begin{cases} \mathbf{x} & \text{if } |\delta_H| > |\delta_v| \\ \mathbf{x}^\top & \text{otherwise} \end{cases}$$
This is useful for areas have more noise in one direction than in the other.
6. Replace the centre value of the current window with the weighted average from valid disparity values, and rotated weights matrix $\mathbf{w}$.
7. Repeat for every $n \times n$ window in the disparity map, as many times as necessary.

This method was found to be very successful in smoothing the point cloud with minimal loss to the more delicate features such as the lips.

Denoising

Some areas of the photos are consistently prone to errors. These are typically near the edge of the face where camera views are significantly different, or due to specular reflection. This can lead to groups of pixels that are not in line with the rest of the model. These typically slip through the dynamic programming smoothing as they are larger groups of pixels, so the image matching costs outweigh the smoothness costs during the optimisation.

These patches can be removed by using MATLAB's denoising function, `pcdenoise()` [25]. This algorithm works by iterating through the point cloud, calculating the mean and standard deviation in depth of the points n nearest neighbours (in the x, y plane). If the point has a depth that is more than a threshold number of standard deviations away from the mean, it is removed.

4.4 Expression Tracking

HajiRassouliha et al. [1] developed a tracking algorithm, which traces features through a series of undistorted frames. This algorithm can be applied to our photos to track features during an expression. For this tracking to be useful, point-to-point correspondence must be maintained throughout the expression. This is achieved by tracking all the points from the point cloud of the neutral face.

The raw matched coordinates from the disparity matching could be used here, however, this would render the post-processing performed on the point cloud null. A better method is to back-project the processed point cloud onto the images and track those points during the expression. This means the point cloud must be back-projected into points on the rectified images, and then mapped back to the undistorted images for the tracking algorithm to work with.

4.4.1 Back-Projection

Point Cloud to Rectified Images

Rearranging Equation 4.3.5, world points can be translated into image coordinates (on the reference image) with matrix multiplication via

$$\frac{1}{w}\begin{bmatrix} x \\ y \\ d \\ 1 \end{bmatrix} = Q^{-1}\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Adding the disparity to the x coordinate gives the corresponding coordinate in the second image (as the images are rectified).

Rectified to Undistorted Images

Next, the points on the rectified images must be mapped back to the undistorted, non-rectified images. As discussed in section 4.3.1, image rectification creates projective image transforms to rectify the images. The inverse of these transform matrices is used to undo the rectification transform similar to the previous step. This maps the matched points onto the undistorted images to be ready for tracking.

Projection

As the points are now back in the original coordinate system, the matched points are no longer guaranteed to be horizontally aligned. This means that the projection methods discussed for rectified images in section 4.3.3 are no longer appropriate. Instead, code from [31] has been adapted, which works by computing the intercept of two lines projected from the camera origins through the features image coordinate. This means that all point clouds are now in the coordinate system of the reference camera.

4.4.2 2D Tracking

Now that all the points from the point cloud have been mapped to the undistorted images, the ABI's tracking algorithm can be applied [1].

The tracking algorithm estimates pixel displacements between two time-sequential images. If the shift between images is too great, then the tracking may fail. On the other hand, if points are tracked across too many frames, then the accumulation of error can become significant and affect results. Therefore, time steps must be chosen carefully. Our recordings were typically done at 90 fps, which provided flexibility in choosing which frames to use for tracking. At slower stages in the expression, up to four or five frames may be skipped, but during faster parts, each successive frame would be used.

This process is repeated until the expression is completed. This means that point-to-point correspondence is held throughout the expression, and each stage of the expression can be projected into its own 3D point cloud.

4.5 Principal Component Analysis

PCA was used to determine the dominant modes of depth change during expressions. Firstly, point clouds were reconstructed for several expressions, and brought into a common coordinate system using MATLABs `pcregister()` function [25]. Basis-spline (B-spline) [32] functions were then fitted to each point cloud using the framework developed in [33]. B-splines are a parametric surface represented by a grid of control points, and are discussed in depth in [33]. Once the B-spline surfaces were fitted to the different expressions, the parameters of each model are arranged in a matrix like so:

$$Z = \begin{bmatrix} \phi_{11} & \phi_{12} & \dots & \phi_{1n} \\ \phi_{21} & \phi_{22} & \dots & \phi_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \phi_{p1} & \phi_{p2} & \dots & \phi_{pn} \end{bmatrix},$$

where ϕ_{ij} represents the j'th model parameter in the i'th B-spline surface, n is the number of parameters per model, and p is the number of surfaces. The principal components were then extracted by computing the eigenvalues and eigenvectors of the covariance matrix, $Z^\top Z$, with singular value decomposition [34]. The first principal component is the eigenvector with the largest eigenvalue, the second component has the next largest eigenvalue, and so on. As the model parameters are not tied to material points on the face, the modes (components) correspond to the areas which experience the largest changes in depth during expressions, rather than generalised deformations.

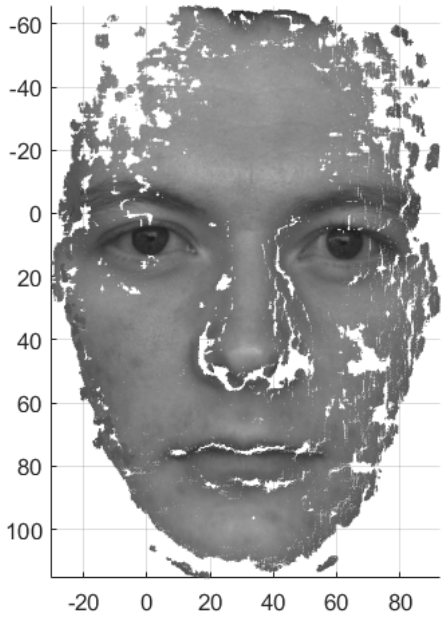
5 Results and Discussion

3D point clouds have been reconstructed from stereo images using the methods described in section 4.3. Tracking was able to be applied to a simpler case of rigid body motion, but accuracy was lost when applied to expressions and results produced were invalid. This is likely caused by a lack of texture present in the captured images.

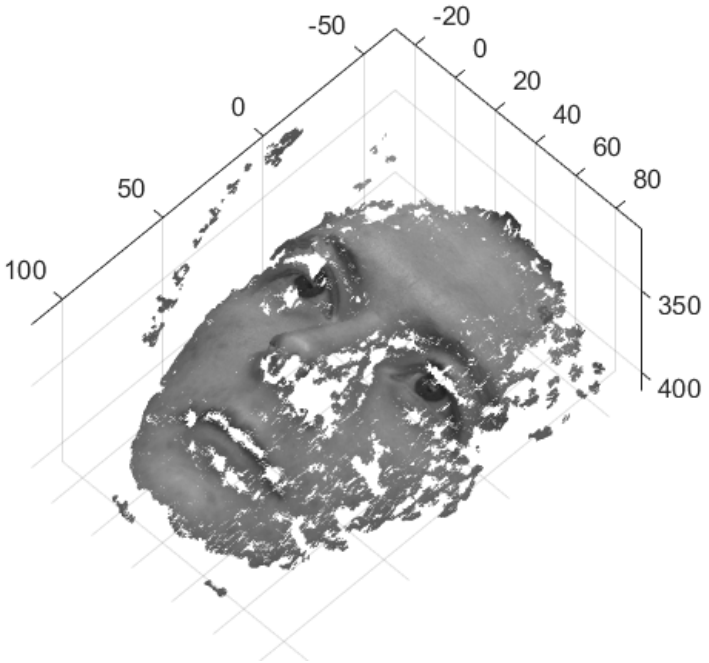
5.1 Reconstruction

5.1.1 Block Matching

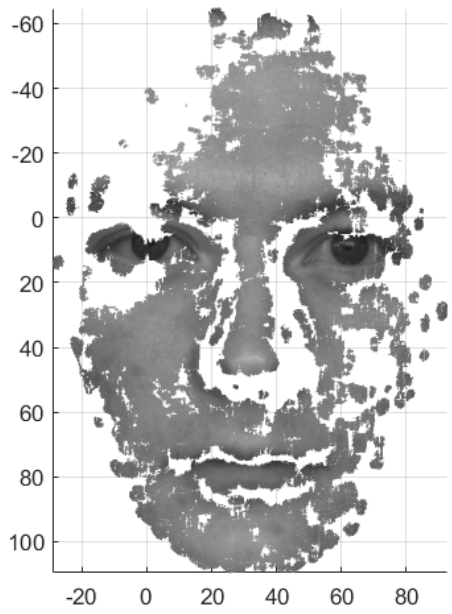
As block matching does not consider the disparity of neighbouring pixels during its search, it can produce noisy point clouds with many speckles due to false matches. These can be seen in Figure 5.1 (b). Figure 5.1 (c) and (d) show that increasing the minimum acceptable confidence of accepted matches removed these outliers, but many of the correct matches were also lost, particularly in areas of the face with sudden depth changes as these appear the most different across images. The missing areas could be useful for representing expressions, therefore, it was concluded that regular block matching is not appropriate for our applications.



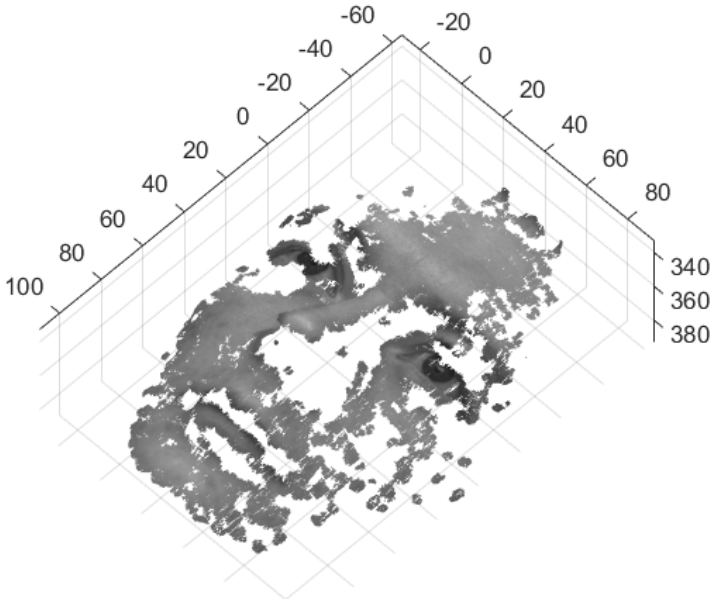
(a) Low confidence front on



(b) Low confidence angle shot



(c) High confidence front on

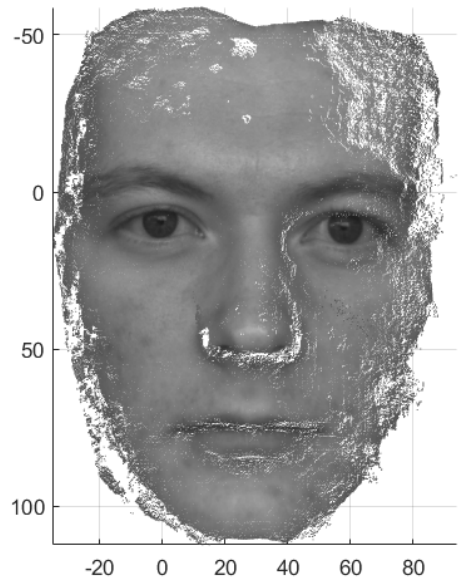


(d) High confidence angle shot

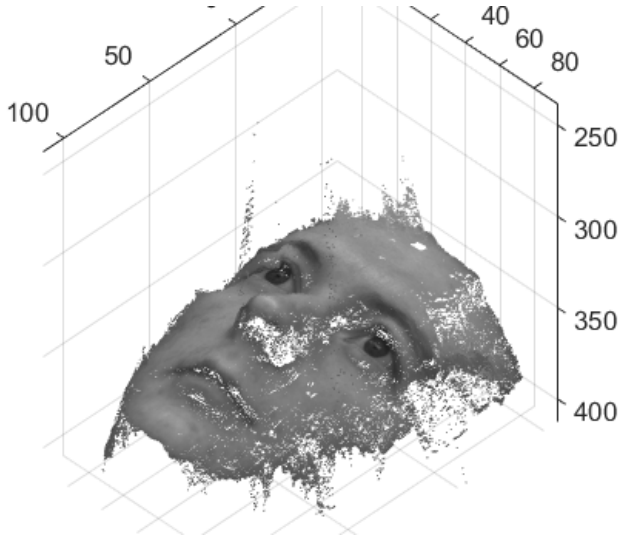
Figure 5.1: Point clouds produced with block matching

5.1.2 Semi-Global Matching

Point clouds produced with the semi-global block matching algorithm described in 4.3.2 are significantly better than those from pure block matching. More areas of the face have been captured, and large outliers have been prevented. There is still some noise, as shown in Figure 5.2 (b), but this is less significant than in block matching and can be corrected with some post-processing.



(a) Semi-global matching — front on



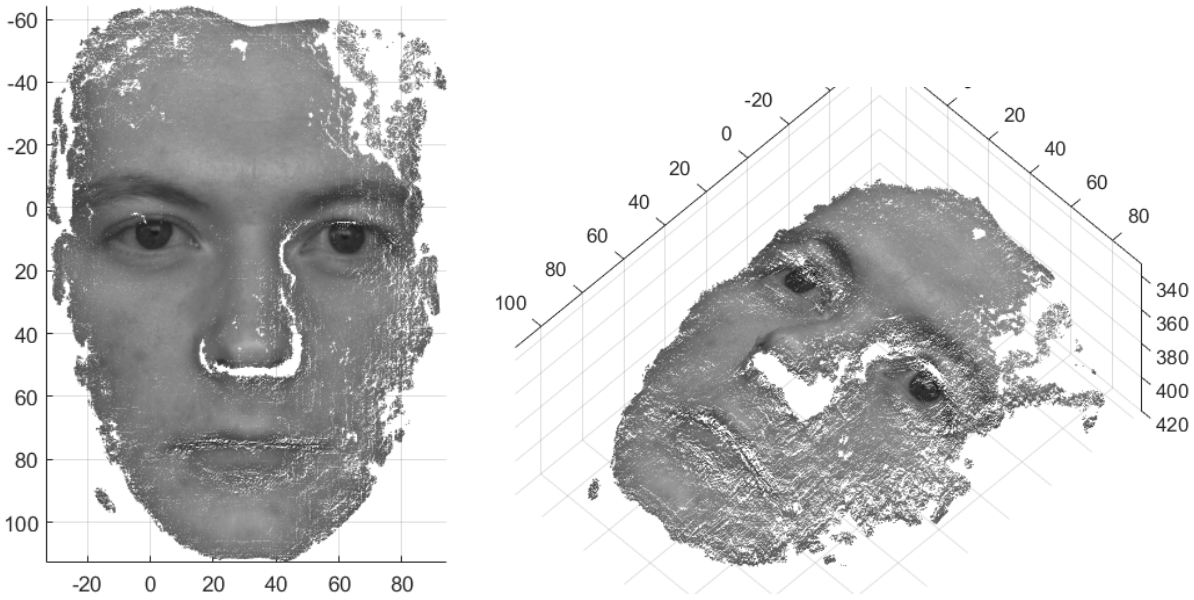
(b) Semi-global matching — angle shot

Figure 5.2: Point cloud produced with semi-global matching

5.1.3 Post-Processing

Applying the post-processing methods from section 4.3.4 significantly improved the quality of the reconstructions. The denoising successfully removed the outliers that remained after semi-global matching, this is shown by comparing Figure 5.3 to Figure 5.2. The weighted average smoother also corrected the roughness of the raw point clouds, and successfully maintained the shape of features on the face, as shown in Figure 5.4.

Denoising

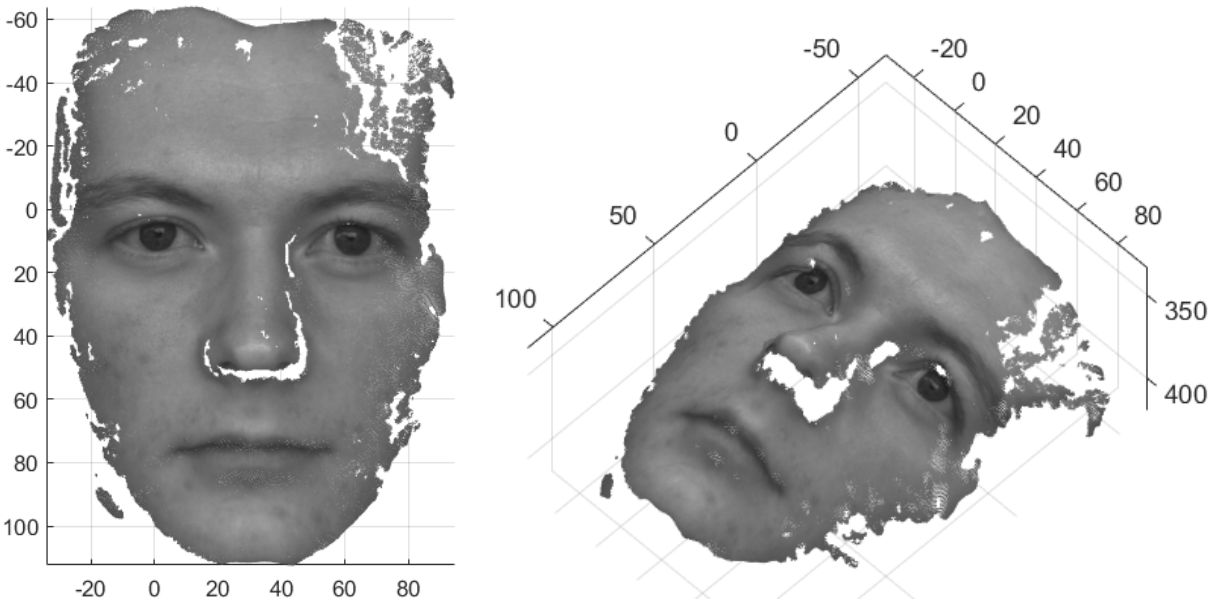


(a) Denoised cloud — front on

(b) Denoised cloud — angle shot

Figure 5.3: Point cloud after denoising

Smoothing



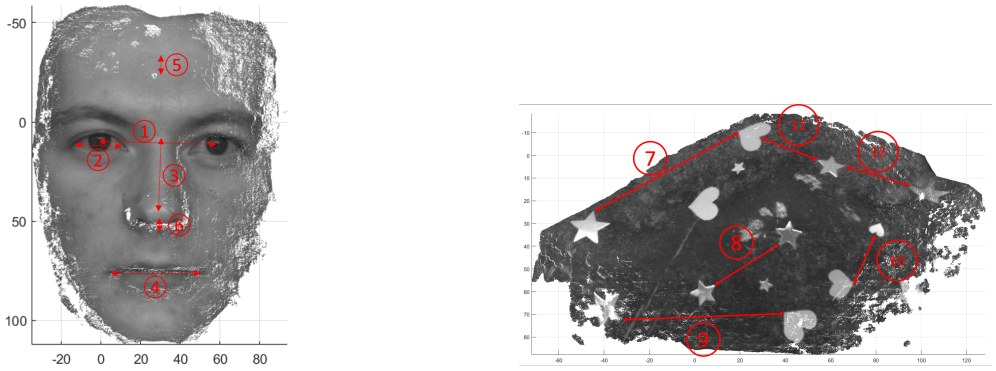
(a) Smoothed cloud — front on

(b) Smoothed cloud — angle shot

Figure 5.4: Point cloud after denoising and smoothing

5.1.4 Reconstruction Accuracy

While the shape of the cloud looks good by eye, it was necessary to assess the accuracy by taking measurements of the reconstruction and comparing them with the true model. This process was completed both for my reconstructed face, and also a rock, as its dimensions are known more precisely (note that there are some stickers on the rock from a previous project to assist image registration algorithms in matching). The measurements taken are shown in Figure 5.5, and results are summarised in tables 5.1 and 5.2. Overall the dimensions matched very well. There were some slight differences in the measurements of the face, but this was expected as dimensions vary depending on facial expressions.



(a) Measurements taken on face

(b) Measurements taken on rock

Figure 5.5: Measurements taken to assess accuracy

Table 5.1: Face measurements summary

Label	Description	Actual (mm)	Model (mm)
1	Pupillary distance	61	62
2	Eye width	24	24
3	Nose length	50	48
4	Mouth width	41	42
5	Distance between forehead wrinkles	12	12
6	Nose depth	13	14

Table 5.2: Rock measurements summary

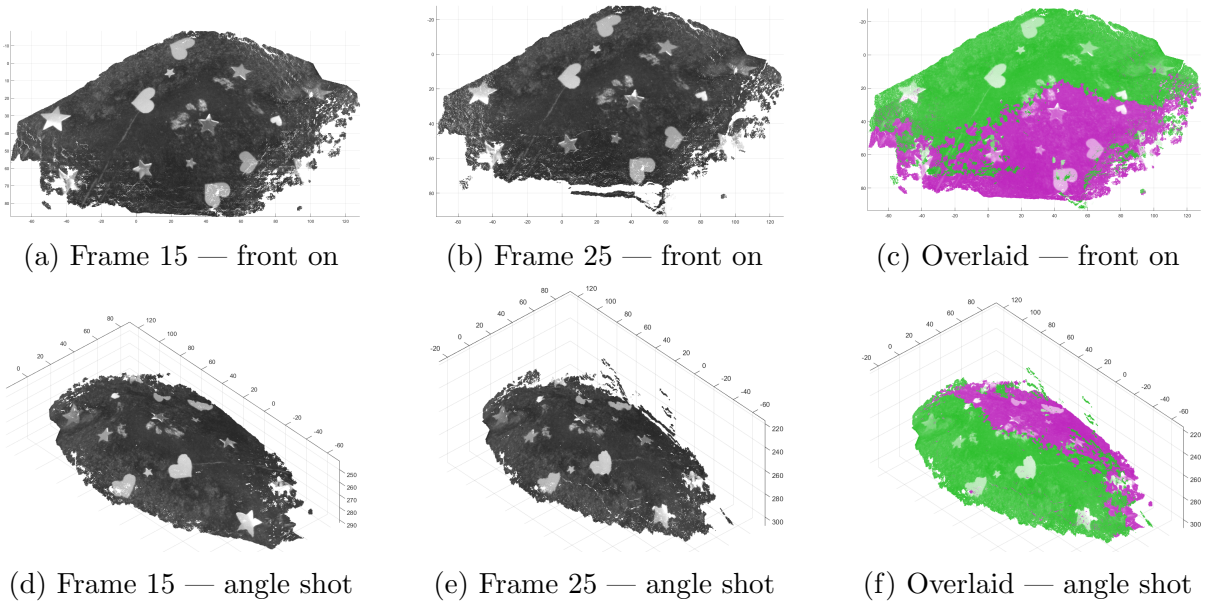
Label	Actual (mm)	Model (mm)
7	76	76
8	35	35
9	75	75
10	37	37
11	34	34
12	33	33

5.2 Tracking

5.2.1 Translation of a Rigid Object

As a proof of concept for the project, the reconstruction and tracking were tested on the simpler case of rigid body movement with a rock. A sequence of photos capturing an upwards translation of the rock were taken. A 3D point cloud of rock was reconstructed from frame 15 using the methods in Section 4.3, and then its movement was tracked from frame 15 to frame 25. This is an arbitrary length of tracking that was chosen as the displacement of the rock across these frames is expected to be similar in magnitude to the expressions I planned on tracking.

Figure 5.6 shows reconstructions of the rock before and after tracking. (a) and (d) show the original reconstruction from frame 15, and (b) and (e) show reconstruction after tracking was performed up to frame 25. Green represents the rock in frame 25 after tracking, and purple represents the original reconstruction from frame 15. These two clouds are overlayed in (c) and (f) to show the movement. The reconstruction has maintained the correct shape of the rock, and the rigid upwards shift has been captured. However, there are some outliers that have appeared from tracking, mostly along the bottom edge. These can be seen by comparing Figure 5.6 (d) and (e), and also appear in (c) and (f). These could be dealt with by flagging points which have extreme changes in depth when compared to neighbouring values.



(a) Frame 15 — front on (b) Frame 25 — front on (c) Overlaid — front on

(d) Frame 15 — angle shot (e) Frame 25 — angle shot (f) Overlaid — angle shot

Figure 5.6: Comparison of point clouds before and after tracking

Figure 5.7 shows a random sample of points before and after tracking. This confirms that the points have been tracked accurately as they are tied to the same material points on the rock in both images.

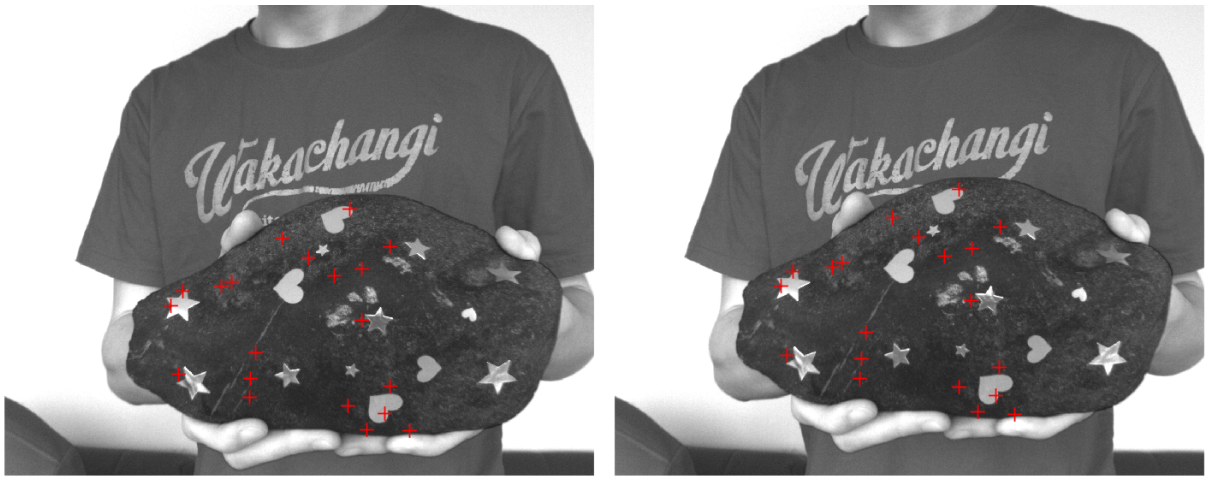
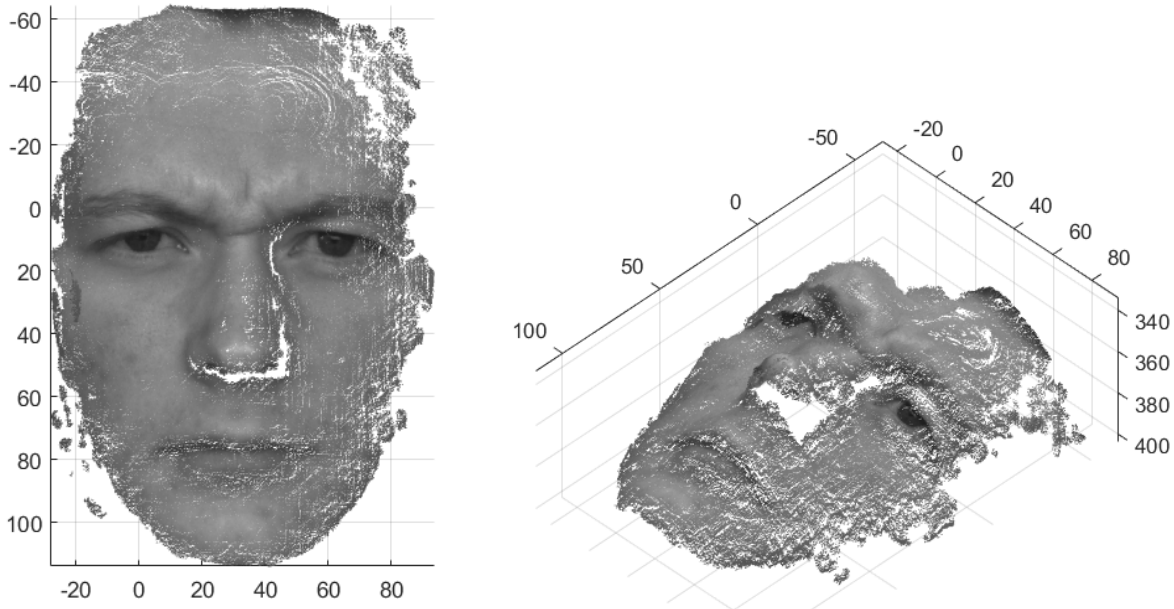


Figure 5.7: Sample points tracked from frame 15 (left) to frame 25 (right)

5.2.2 Frown

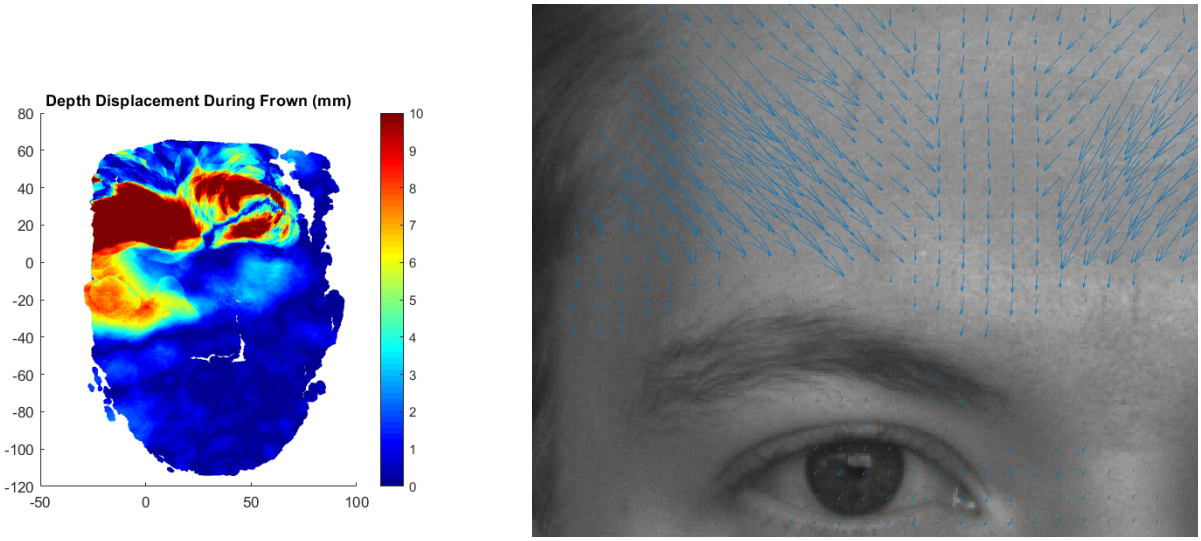
Tracking a frown was not as successful as the rock example. Most of the expected areas have moved in the right direction, however, the magnitudes of displacement do not appear to be correct, and the eyebrows in particular have not moved. When inspecting the photos, the eyebrows clearly move the most during the frown, so the tracking has not been successful in this instance.



(a) Frown — front on

(b) Frown — angle shot

Figure 5.8: Tracking of features during a frown



(a) Depth displacement

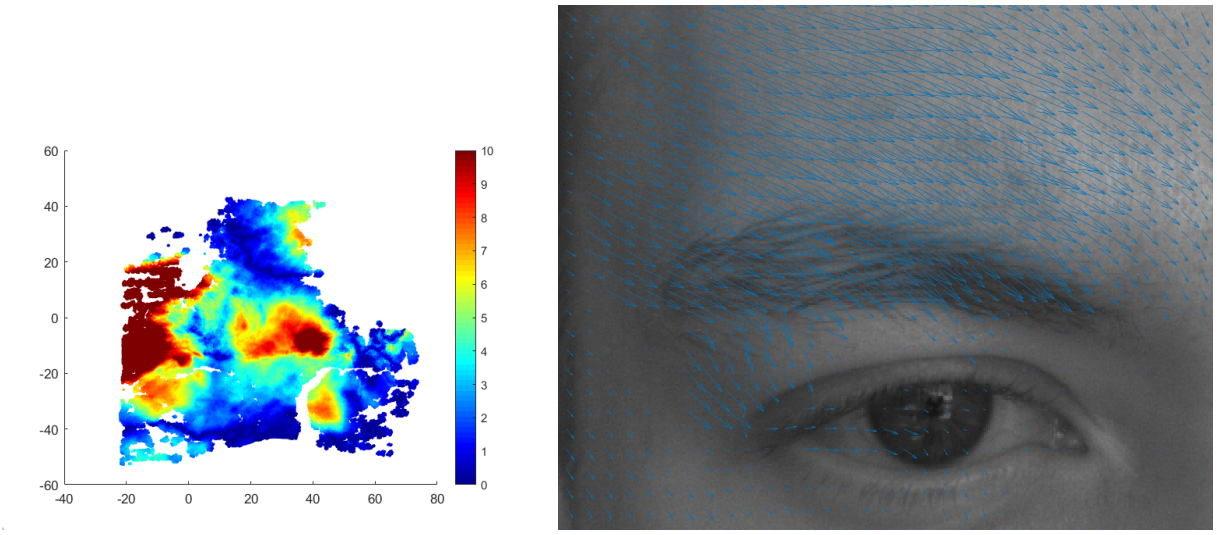
(b) Tracking during frown

Figure 5.9: Tracking on my eyebrow

The depth displacement confirms that the magnitudes and positions of displacement are not quite correct. Depth displacements are reaching 20 mm in some parts of the brow. The true maximum depth displacement is expected to be closer to 5 mm. Figure 5.9 shows that there is a discontinuity in the tracking near the eyebrows. Movement has been identified in the forehead above the eyebrows, but the eyebrows themselves are almost stationary. This shows the eyebrows have not been tracked successfully. This is likely due to the fuzziness of the eyebrows which could be interfering with the image registration during tracking.

To improve the results of tracking, a new set of photos were taken with the face 150 mm from the cameras. Being this close results in more texture being captured, but the whole face is no longer seen in the field of view, so the results are restricted to the forehead. This showed promising tracking results over my eyebrows as shown in Figure 5.10 (b). However, the depth displacement

map shown in (a) still looked concerning, with displacements reaching up to 20 mm once again. It was found that even with the improved texture, the tracked points were not tied to the same material points throughout the expression. This is shown in Figure 5.11, and is most obvious looking at the two points on the eyebrow, but also note that the points around the forehead creases have had their movement exaggerated. Given more time, it would have been good to continue experimenting with distances to determine how close the cameras must be for accurate tracking. Alternatively, higher resolution cameras would also be useful in improving the accuracy of the tracking.



(a) Depth displacement for close frown

(b) Tracking during close frown

Figure 5.10: Tracking with close up photos

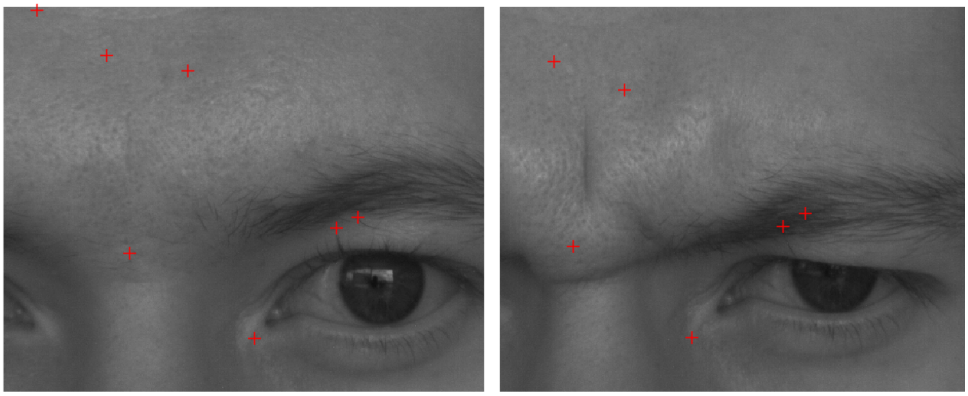
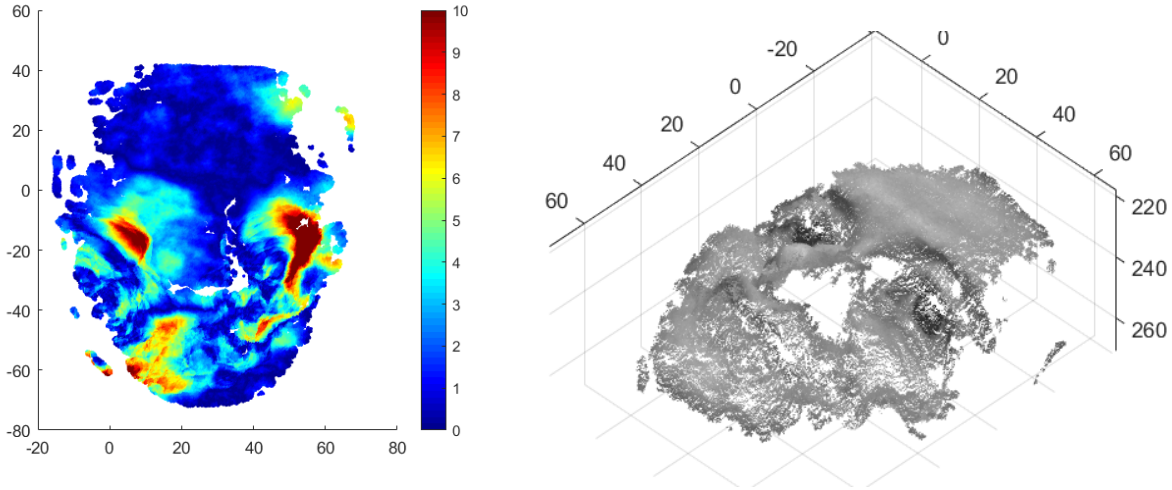


Figure 5.11: Sample points tracked from frame 26 (left) to frame 58 (right) during frown

5.2.3 Smile

Tracking features during a smile yielded similar results to the frown. Displacements are inaccurate, and sometimes in incorrect areas. The depth displacement map shown in Figure 5.12 (a) is patchy and not continuous as was expected. The 3D reconstruction in (b) also shows this as the cheeks and mouth have become irregular and non-smooth.



(a) Depth displacement for smile

(b) Tracking during close frown

Figure 5.12: Tracking features during a smile

While we were unable to complete testing for this due to time constraints, the tracking has been shown to work with elastic deformations in [1]. Therefore, the likely source of error in the tracking is a lack of texture in the source images.

5.3 Principal Component Analysis

Point clouds were reconstructed using the methods described in Section 4.3 for five expressions, shown in Figure 5.13.

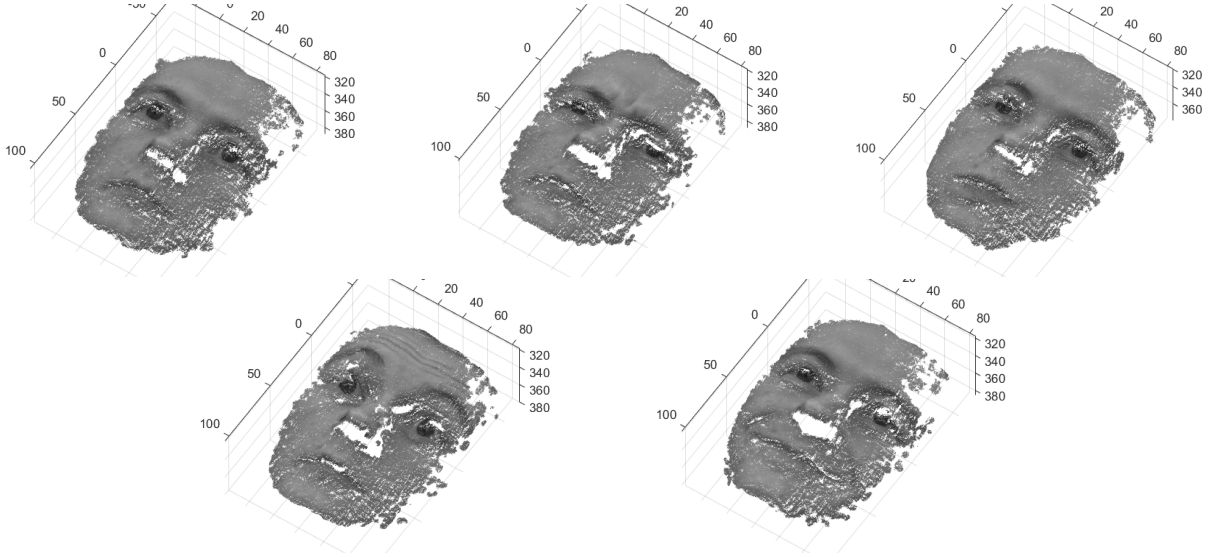
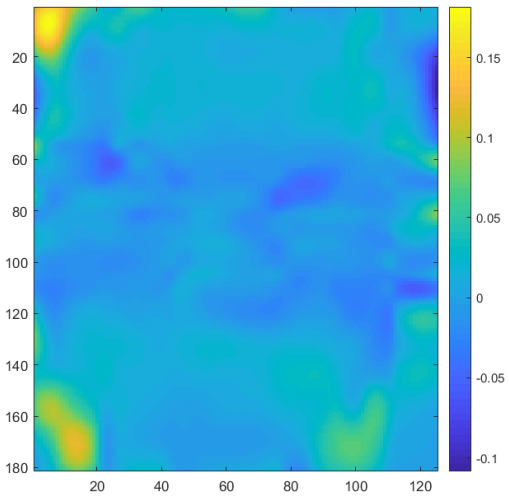


Figure 5.13: Point clouds produced from various expressions

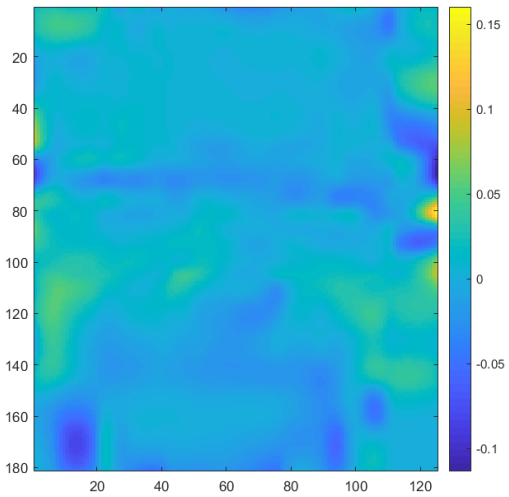
B-spline surfaces were then fitted to the point clouds using the process outlined in [33]. This created a lower-dimensional representation of the depth for each expression. Applying PCA to the model parameters in the various expressions as discussed in Section 4.5, we were able to obtain the principal components. These are vectors corresponding to the areas of the face with

the most depth change across expressions, and are generated with respect to the mean face, i.e., the averaged depth of each surface.

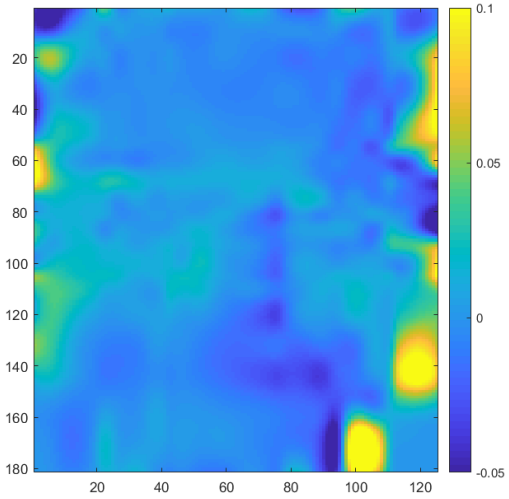
The surfaces models were then evaluated using the extracted components as model parameters to create a depth representation of the modes. These are shown in Figure 5.14, and resemble different areas of the face as expected.



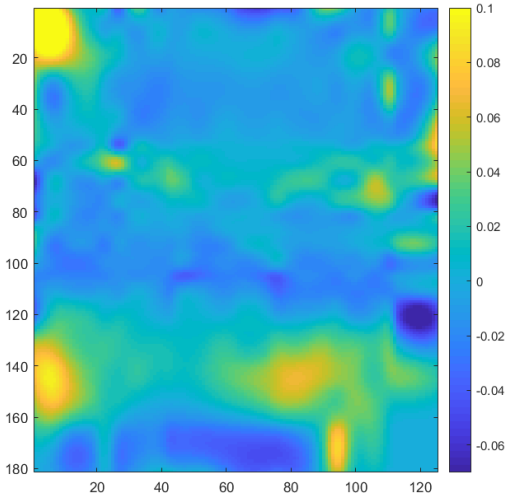
(a) First principal component



(b) Second principal component



(c) Third principal component



(d) Fourth principal component

Figure 5.14: Extracted principal components

6 Future Work and Recommendations

6.1 Possible Improvements

This section details how improvements could be made in completed areas of the project.

6.1.1 More Cameras

Using more cameras will allow cameras to be closer to the subject and therefore improve tracking results. The whole face will then be reconstructed from each pair of cameras. This is possible as all reconstructions will be in the same coordinate system due to the multi-camera calibration procedure used. Alternatively, using cameras with higher resolution would also improve the tracking results.

6.1.2 Hardware Triggering

Currently, we use LabVIEW to trigger both cameras and capture time-synchronous images. This software triggering approach is expected to have more significant delays than physically connecting the cameras and triggering image capture via hardware. By recording a stopwatch, it is estimated that the delay in capture between the two cameras is typically $< 1\,\text{ms}$, which is sufficient for this project. However, hardware triggering may be more reliable for keeping the time delays low, and could be required for very swift movements or if more than two cameras are being used, as this could slow down the software triggering.

6.1.3 Other Calibration Targets

It has been shown that checkerboards are not the best calibration target. Circles are better targets as they contain more information per shot which can result in extremely accurate camera calibration [35]. This is also not essential for our project purposes as the checkerboards with optimisation are still able to calibrate the cameras to a high degree of accuracy.

6.1.4 Correspondence Between Expressions

Currently, expressions are filmed separately, and compared to the neutral face at the beginning of the recording. As a result, there is no point-to-point correspondence between expressions and PCA can not be used in this format. This could be addressed in three ways:

- Overlaying point clouds to find the best correspondence match between the sets of neutral faces.
- Sequentially filming a wide range of expressions, and tracking features throughout the entire video.
- Fitting parametric models to the clouds. This creates a lower-dimensional model described by fewer parameters. These parameters would serve as correspondence points and reduce the computation required for PCA.

6.1.5 Combining Disparity Maps

It was found that the window size used in the semi-global matching algorithm had a substantial influence on the disparity map produced. If the window was too large, then the reconstructed face would lose some of the finer details such as the shape of the lips and eyes. However, having a smaller window produced a less smooth point cloud with much more gaps.

It is possible to run the algorithm with a variety of window sizes and compare the disparity maps produced. Outliers for each pixel can be removed, and the remaining data can be combined with an averaging process. This produced significantly better point clouds with excellent resolution and almost no noise. However, the averaging process resulted in a non-smooth surface. In future, better methods than averaging could be explored for combining disparity maps to maximise accuracy and maintain smoothness.

6.2 Next Steps

There are several options for improving tracking accuracy in order to produce point clouds with point-to-point correspondence. They include; using more cameras placed closer to the face, using higher resolution cameras, or using colour cameras instead of monochrome.

Once accurate point clouds are achieved with the tracking algorithm, the PCA pipeline developed in this project can be applied to extract a set of basis faces, in order to represent expressions efficiently and accurately.

7 Conclusions

This project developed a framework for modelling a face in 3D, and assessed the feasibility of tracking skin during expressions to deform the 3D models.

The pipeline for image acquisition, camera calibration, and 3D reconstruction has been successfully developed and tested.

The tracking process was demonstrated to work for the simplified case of tracking rigid body motion on a highly textured rock, however, difficulties arose when applying tracking to facial expressions. It is suspected this is due to a lack of texture captured with the current setup.

As accurate models with point-to-point correspondence were not able to be produced, parametric surfaces were fitted to 3D point clouds of several expressions. PCA was then applied to the parameters of the surface models to determine a set of basis vectors.

In future works, point clouds or models with point-to-point correspondence may be achieved. This PCA process can then be applied to these models to produce a set of basis faces, such that a linear combination of the set can represent any expression.

In order to achieve point clouds with point-to-point correspondence, further testing should be done in the following areas:

- Testing close up ($< 100\,\text{mm}$) tracking with a single camera to verify the possibility of accurate tracking using Flea 3 cameras. If this is unsuccessful, alternative cameras may be required.
- Performing another simplified case of non-rigid motion, perhaps on a highly textured balloon or similarly deformable object. This will serve to rule out the possibility that the non-rigid deformations are causing issues with tracking.

Bibliography

[1] A. HajiRassouliha, A. J. Taberner, M. P. Nash, and P. M. Nielsen, "Subpixel phase-based image registration using savitzky-golay differentiators in gradient-correlation," *Computer Vision and Image Understanding*, vol. 170, pp. 28–39, 2018.

[2] M. Pantic, A. Nijholt, A. Pentland, and T. S. Huanag, "Human-centred intelligent human computer interaction (hci): how far are we from attaining it?," *International Journal of Autonomous and Adaptive Communications Systems*, vol. 1, no. 2, p. 168, 2008.

[3] M. Vieira, L. Velho, A. Sa, and P. Carvalho, "A camera-projector system for real-time 3d video," *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR05) - Workshops*.

[4] F. Tsalakanidou, F. Forster, S. Malassiotis, and M. G. Strintzis, "Real-time acquisition of depth and color images using structured light and its application to 3d face recognition," *Real-Time Imaging*, vol. 11, no. 5-6, p. 358369, 2005.

[5] P. S. Huang, "High-speed 3-d shape measurement based on digital fringe projection," *Optical Engineering*, vol. 42, p. 163, Jan 2003.

[6] R. J. Woodham, "Photometric method for determining surface orientation from multiple images," *Optical Engineering*, vol. 19, Jan 1980.

[7] A. Agrawal, R. Chellappa, and R. Raskar, "An algebraic approach to surface reconstruction from gradient fields," *Tenth IEEE International Conference on Computer Vision (ICCV05) Volume 1*, 2005.

[8] G. J. Brostow, C. Hernandez, G. Vogiatzis, B. Stenger, and R. Cipolla, "Video normals from colored lights," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 10, p. 21042114, 2011.

[9] B. H. Bodkin, *Real-Time Mobile Stereo Vision*. PhD thesis, 2012.

[10] "2-d and 3-d image registration for medical,remote sensing, and industrial applications," *Journal of Electronic Imaging*, vol. 14, p. 039901, Jan 2005.

[11] E. J. L. P. Tang, A. HajiRassouliha, M. P. Nash, P. M. F. Nielsen, A. J. Taberner, and Y. O. Cakmak, "Non-contact quantification of jugular venous pulse waveforms from skin displacements," *Scientific Reports*, vol. 8, no. 1, 2018.

[12] A. HajiRassouliha, A. Taberner, J. Nash, and P. Nielsen, "Robust and accurate multiple camera stereographic calibration [under review],"

[13] R. A. Hamzah and H. Ibrahim, "Literature survey on stereo vision disparity map algorithms," *Journal of Sensors*, vol. 2016, p. 123, 2016.

[14] A. Kaehler and G. R. Bradski, *Learning OpenCV 3: Computer vision in C with the OpenCV library*. OReilly, 2017.

[15] H. Kim and K. Sohn, "3d reconstruction from stereo images for interactions between real and virtual objects," *Signal Processing: Image Communication*, vol. 20, no. 1, p. 6175, 2005.

[16] C. Banz, H. Blume, and P. Pirsch, "Real-time semi-global matching disparity estimation on the gpu," pp. 514–521, 11 2011.

[17] M. Zollhfer, P. Stotko, A. Grlitz, C. Theobalt, M. Niener, R. Klein, and A. Kolb, "State of the art on 3d reconstruction with rgb-d cameras," *Computer Graphics Forum*, vol. 37, no. 2, p. 625652, 2018.

[18] J. Digne, D. Cohen-Steiner, P. Alliez, F. D. Goes, and M. Desbrun, "Feature-preserving surface reconstruction and simplification from defect-laden point sets," *Journal of Mathematical Imaging and Vision*, vol. 48, no. 2, p. 369382, 2013.

[19] F. Remondino, "From point cloud to surface: The modeling and visualization problem," *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 2004.

[20] Khatamian and Arabnia, "Survey on 3d surface reconstruction," *Journal of Information Processing Systems*, 2016.

[21] M. Berger, A. Tagliasacchi, L. Seversky, P. Alliez, G. Guennebaud, J. Levine, A. Sharf, C. T.Silva, "A survey of surface reconstruction from point clouds," *COMPUTER GRAPHICS-forum*, 2017.

[22] P. Jindal and D. Kumar, "A review on dimensionality reduction techniques," *International Journal of Computer Applications*, vol. 173, no. 2, p. 4246, 2017.

[23] G. Sandbach, S. Zafeiriou, M. Pantic, and L. Yin, "Static and dynamic 3d facial expression recognition: A comprehensive survey," *Image and Vision Computing*, vol. 30, no. 10, p. 683697, 2012.

[24] National Instruments, *LabVIEW 2019*, 2019.

[25] The Mathworks, Inc., Natick, Massachusetts, *MATLAB version 9.6.0.1135713 (R2019a) Update 3*, 2019.

[26] "Depth camera d435." https://www.intelrealsense.com/depth-camera-d435.

[27] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2019.

[28] "Wikimedia," *Wikimedia*. https://commons.wikimedia.org/w/index.php?curid=54749634.

[29] A. Brooks, X. Zhao, and T. Pappas, "Structural similarity quality metrics in a coding context: Exploring the space of realistic distortions," *IEEE Transactions on Image Processing*, vol. 17, no. 8, pp. 1261–1273, 2008.

[30] J. Čech and R. Šára, "Efficient sampling of disparity space for fast and accurate matching," in *BenCOS 2007: CVPR Workshop Towards Benchmarking Automated Calibration, Orientation and Surface Reconstruction from Images*, IEEE, 2007. Software GCS 2.0.

[31] J. Xiao, "Princeton vision and robotics."

[32] K. J. Liew, A. Ramli, and A. A. Majid, "B-spline surface fitting on scattered points," *Applied Mathematics and Information Sciences*, vol. 10, p. 273281, Jan 2016.

[33] G. Qian, "The face is the soul of the body," Oct 2019.

[34] "Singular value decomposition," *Wiley StatsRef: Statistics Reference Online*, 2014.

[35] A. Hajirassouliha, A. J. Taberner, M. P. Nash, and P. M. Nielsen, "A model-based technique for calibration of a mutli-camera stereoscopic system [under review],"