PCA Based Body Model Parametrisation

Acquisition and Processing of 3D Geometry: Final Project

Team Members: Fen Jin and Elliott Thompson

Report written by Elliott Thompson

Paper Summary

Papers Aim

The paper that we chose to work on is titled "The space of human body shapes: reconstruction and parameterization from range scans" [1]. We decided to base our project on this paper as we were both interested in the topic of PCA and this paper makes use of it in the parameterization of full body range scans. We hoped to gain an understanding of its advantages and get some experience implementing this particular algorithm.

3D range scans are one method that is used to create 3D body models, however this paper highlights a few of the hurdles that must to be overcome if the models are to be valuable for modelling and animation. It mentions how range scans are generally noisy and contain many gaps in them in surface areas, where scanning equipment was occluded or at a grazing angle. It also mentions how 3D scans do not give information about the space that bodies occupy. This makes it difficult for 3D scan on their own to be manipulated to generate new and different body models. The aim of this paper then, was to take a set of 250 full body range scans which each includes a set of sparse 3D markers and create a parameterization which tackles these issues.

Papers Method

This parameterization was achieved through the use of a default high resolution 3D template mesh, which was fitted to each of the body range scans using an optimization technique. This led to each of the 250 fitted template meshes having the same structure and point to point correspondence as each other but maintaining the shape of the individual scan they were fitted from. The aforementioned optimization technique was suggested as being an objective function that minimises the error of three different terms. They are as follows:

1. Data error:
   The first error attempts to minimise the sum of squared differences between each of the template surface points and the point of minimal distance on the current 3D scan surface. The

notion being that minimising this error should produce a template surface that closely matches the shape of the 3D scan.

2. Smoothness error:
   They suggest that the data error alone will not produce an attractive mesh as currently neighbouring points on the template mesh may move to completely separate parts of the 3D scan. The smoothness error abates this by constraining the objective function further. This error looks to minimise the differences between the transformations that are applies to neighbouring points on the template surface. This should ensure that the final template mesh will be formed in a smooth manner.
3. Marker error:
   The third and final error suggested by this paper is a marker error which avoids the situation of the function getting stuck in a local minimum caused by bad initial alignment of the template mesh and 3D scan. It is mentioned that this can occur when the right arm of one initially matches with the left arm of the other which brings the function into a local minimum that is cannot get out of. By minimising the sum of the squared differences between the distance of a small set of known marker positions on both models, it ensures that the function will not get stuck in these local minimums.

These three error terms are combined into a single objective function, each with their own weight as follows:

$$E = \alpha E_d + \beta E_s + \gamma E_m$$

The paper then says that the alpha, beta and gamma weights can then be tweaked as necessary to rebalance the three error terms, depending on the particular data set given.

Papers Applications

The paper describes various applications that this parameterisation can be useful in. The first such application is the transfer of textures and morphing between different individual models.

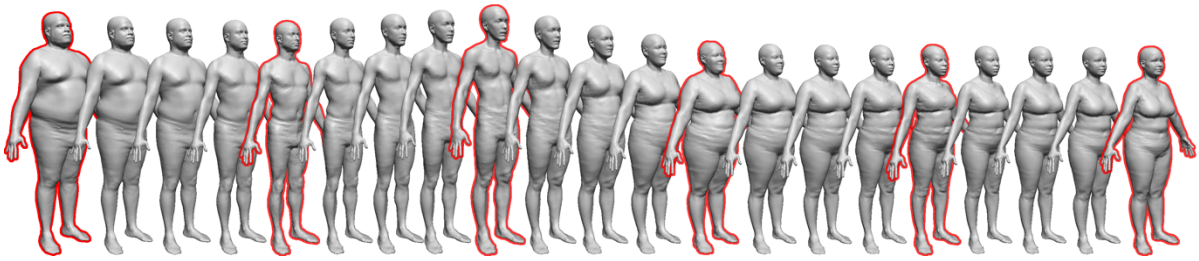


Figure 1 A set of full body models shown in red have been interpolated between. Taken from [1].

In figure 1, from the paper, it can be seen that the models in the red highlight can be morphed between, producing models which are a combination of two adjacent red models. As the parameterisation that was suggested offers index correspondence between each vertex in every template model, this type of morphing can be achieved by simply interpolating the position of each given vertex between the two models. Likewise, the transfer of textures can be achieved by applying the textures of every mesh face of one model to the mesh face of the other.

The paper also discusses how instrumentation transfer can be used to animate a set of many models, given the skeleton and skinning information of a single model. A set of markers are selected based on a single model, around a skeleton. These markers are then transformed to the positions they occupy on

other models' skeletons, given their relative position to joints on each skeleton. This process is made possible as the parameterisation between model skeletons can be found. Inverse kinematics is then used to determine the best position of each marker on the models based on their relative positions to the skeleton joints and their global position in the template mesh.

Principle Component Analysis

This is the part of the paper that Fen and I focused our attention on as it gives a practical example of using PCA to parameterise a given model. The preceding parts of the paper also rely on having access to a large amount of full body scans and accompanying feature markers. As we did not have access to these models, we decided to focus on generating new and unique body models from a set of 3D full body template meshes, of which we had access to from [2]. Therefore, our project would focus more on the generation of new models based on a given parameterisation, rather than fitting 3d scans to a template.

It also mentions that one of the benefits of using such a technique is the data compression it provides. This compression comes from the low variance vectors that PCA produces which can be discarded while still retaining an accurate representation of the original model, which was another benefit of choosing PCA.

Using this method, they propose taking the vertices of k models and stacking them into a set of k column vectors each of length $3n$ (with n being the number of vertices). The column wise mean is taken of this matrix and this mean is then subtracted from each column. PCA analysis is then conducted on this mean normalised matrix to produce a set of principle vectors and variances. It then mentions how new random models can be randomly generated by sampling from the gaussian distributions that the principle variances are from. The paper falls short of describing explicitly how to conduct the principle component analysis on the set of models and is therefore left open to interpretation. It would have been beneficial if it had described in more detail exactly how the principle vectors were developed and subsequently used. However, it was from this point that we began to test the ideas laid out in this paper and develop our own algorithm in code.

Report

We decided to develop our project in the C++ language as we had both gained valuable experience handling meshes in C++ from the previous coursework's in this module. We therefore had knowledge on how to use the Eigen, ANN and igl libraries that would be necessary for a project like this. As mentioned previously, our focus would be on the development of a system to generate new and unique models that would be derived from a parameterisation of a set of base models.

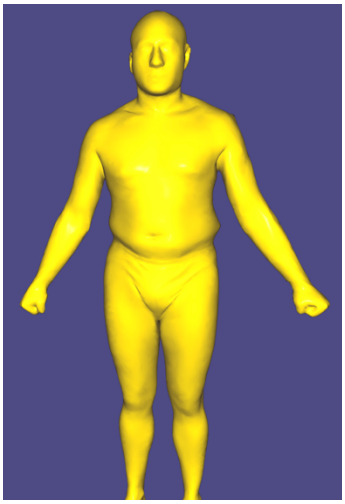


Figure 2 The mean model taken as the average of 25 male models.

To begin, we followed the principle component analysis section of the paper and tried to develop a system to compute this PCA given a set of k models. We loaded in k models using the igl *readOBJ* function and rearranged each n vertex model into a single column of size $3n$. These columns were then stacked into a single matrix called $\boldsymbol{P}$ and the column mean of this matrix was taken using the appropriate Eigen function,

which we called $\bar{\boldsymbol{p}}$. The result of the $\bar{\boldsymbol{p}}$ mesh can be seen in figure 2, where we resized this column vector into an n x 3 matrix. We then negated $\bar{\boldsymbol{p}}$ from every column vector in $\boldsymbol{P}$ to produce a new matrix of mean centred columns called $\tilde{\boldsymbol{P}}$.

Having produced our mean centred matrix of k models, we could begin principle component analysis on them. After discussing different approaches, we decided to use singular value decomposition as we had learned in another module that SVD can be used to decompose a matrix that is rectangular and derive the two sets of orthonormal eigenvectors which could be used in our parameterisation. Our mean centred $\tilde{\boldsymbol{P}}$ matrix is of size $3n$ x k and therefore rectangular in nature. The Eigen library also provides an efficient SVD solver called '*BDCSVD*' which lead us to use the SVD method of PCA rather than building the covariance matrix and running eigen analysis on that. The covariance of the mean centred matrix $\tilde{\boldsymbol{P}}$ can be described using SVD using the following reasoning:

$$\tilde{\boldsymbol{P}}\tilde{\boldsymbol{P}}^{\mathrm{T}} = (\mathrm{U}\Sigma V^T)(V\Sigma U^T)$$

$$\tilde{\boldsymbol{P}}\tilde{\boldsymbol{P}}^{\mathrm{T}} = \mathrm{U}\Sigma(V^T V)\Sigma U^T$$

$$\tilde{\boldsymbol{P}}\tilde{\boldsymbol{P}}^{\mathrm{T}} = \mathrm{U}\Sigma\Sigma U^T$$

$$\tilde{\boldsymbol{P}}\tilde{\boldsymbol{P}}^{\mathrm{T}} = \mathrm{U}\Sigma^2 U^T$$

Therefore, we could take the left singular vector matrix U which will be of size $3n$ x $3n$ as the principle vectors as described in the paper. To achieve this, we took the $\tilde{\boldsymbol{P}}$ matrix and passed it into the 'Eigen::*BDCSVD*' function, which returned the left singular vector U.

Next, we looked at how we could define the mean model $\bar{\boldsymbol{p}}$ in terms of this set of principle orthonormal vectors. To achieve this, we orthogonally projected $\bar{\boldsymbol{p}}$ onto each of the principle column vectors u_i of U and took the linear combination of these projections as follows:

$$\mathrm{U}_{proj}(\bar{\boldsymbol{p}}) = \sum_{i=1}^{3n} \langle \bar{\boldsymbol{p}}, u_i \rangle \, u_i$$

We now had our mean model $\bar{\boldsymbol{p}}$ described as a linear projection of the principle vectors that we developed through PCA. We also knew that the principle vectors were organised by their corresponding principle variance magnitude. Therefore, the principle vector that contributed most to the variance in the shape of our models appears in the first column of our U matrix. Therefore, by scaling the first few principle vectors we could modify the overall shape of the mean mesh to generate new models. We modified our expression to include a set of modifiable scalars for the first 6 principle vectors present in U:

$$\mathrm{U}_{scaledProj}(\bar{\boldsymbol{p}}) = \sum_{i=1}^{6} \alpha_i \langle \bar{\boldsymbol{p}}, u_i \rangle \, u_i + \sum_{i=7}^{3n} \langle \bar{\boldsymbol{p}}, u_i \rangle \, u_i$$

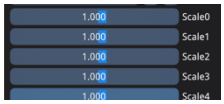


Figure 3 Principle vector Scaling sliders in the GUI

Fen then created a gui menu (figure 3) that would allow us to manipulate the 6 α values. This allowed us to generate new meshes, by modifying the dominant principle vectors. In figure 4 it can be seen how the second principle vector accounts for a lot of the k models variance in weight as scaling this projection adjusts the projected mean model weight.

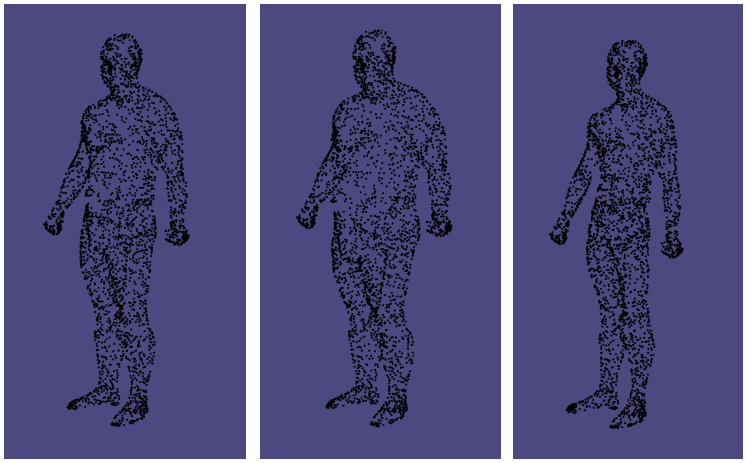


Figure 4 A set of generated point clouds using the PCA projection algorithm and scaling the second principle vector.

It can also be seen in figure 4 that the models at this stage do not have face information and are therefore displayed as point clouds. This was because in order to run PCA in near real time so that we could test different scalar values, we had to sample our mesh vertices. This was because running SVD on a *9375* x *25* matrix was not efficient in memory or computational power, taking upwards of 10 minutes and *10gb* of ram to compute. Therefore, we sampled our mesh so that our U matrix had *937* x *25* values.

EXTENSION

As previously discussed, the computation necessary to run PCA on 25 full resolution models was too great. Therefore, we decided to focus on optimization for the extension part of our project. We had begun this work already as we had already down sampled each full resolution model in order to get the results visible in figure 4. However, this produced other issues with our system. Firstly, the mesh faces that had been supplied with each model no longer matched the down sampled vertices. This meant that it could no longer be used in the form it was given to us. Secondly, the down sampled point clouds that we produced were of a much lower resolution when compared to the models that we received. The goal of our extension then, was to find a balance between decreasing the 3n value via down sampling which would increase performance and retaining the complexity and resolution of the models that were provided to us in the data set.

We decided that we would attempt to retain the full 3n points of the original meshes, so that we could keep the same connectivity as before and reuse the faces that were provided. We would need to apply the results of the PCA alpha scaling to all points, despite only running PCA on a subsample of points. We came up with the idea of defining our mean model points $\bar{\boldsymbol{p}}$ as barycentric coordinates of our sampled mean model $s\bar{\boldsymbol{p}}$. Therefore after we transformed our sampled points using $U_{scaledProj}(s\bar{\boldsymbol{p}})$ we could redefined all the rest of our points as the barycentre's of the new positions**.** The goal then was to move all points in $\bar{\boldsymbol{p}}$ that were not sampled in $s\bar{\boldsymbol{p}}$ to a position that closely matches where they would reside had they been transformed directly with the function $U_{scaledProj}(\bar{\boldsymbol{p}})$.

To achieve this, we iterated over all points in $\bar{\boldsymbol{p}}$ that were not present in $s\bar{\boldsymbol{p}}$, each of which we can call point $\boldsymbol{q}$. We then found their four nearest neighbours within $s\bar{\boldsymbol{p}}$ using the ANN libraries function '*ANNkd_tree*'. We then described each $\boldsymbol{q}$ as a linear combination of the barycentric coordinates of these 4 neighbouring sampled points in $s\bar{\boldsymbol{p}}$. Using these barycentric coordinates we would be able to find these same four neighbouring points within $U_{scaledProj}(s\bar{\boldsymbol{p}})$ and position $\boldsymbol{q}$ according to these new

positions. We found the barycentric coordinates (a-d) of each $\boldsymbol{q}$ by solving the following system of linear equations:

$$\begin{bmatrix} N(1)_x & N(2)_x & N(3)_x & N(4)_x \\ N(1)_y & N(2)_y & N(3)_y & N(4)_z \\ N(1)_z & N(2)_z & N(3)_z & N(4)_z \\ 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} q_x \\ q_y \\ q_z \\ 1 \end{bmatrix}$$

Where each N(i) is one of the four nearest neighbours of $\boldsymbol{q}$ in the sampled mean points $s\bar{\boldsymbol{p}}$. We then repositioned each $\boldsymbol{q}$ by using a linear combination of these barycentric coordinates as follows:

$$\boldsymbol{q}_{updated} = a * uN(1) + b * uN(2) + c * uN(3) + d * uN(4)$$

Where each uN(i) is one of the four nearest neighbours of $\boldsymbol{q}$ but at their new position within $U_{scaledProj}(s\bar{\boldsymbol{p}})$. We then combined all points $\boldsymbol{q}_{updated}$ with all the points in $U_{scaledProj}(s\bar{\boldsymbol{p}})$, in the correct order to create the final full resolution output model. As we now had all points in their correct order within the output model, we could use the original face data to display a full resolution mesh of our generated PCA scaled models, seen in figure 5.

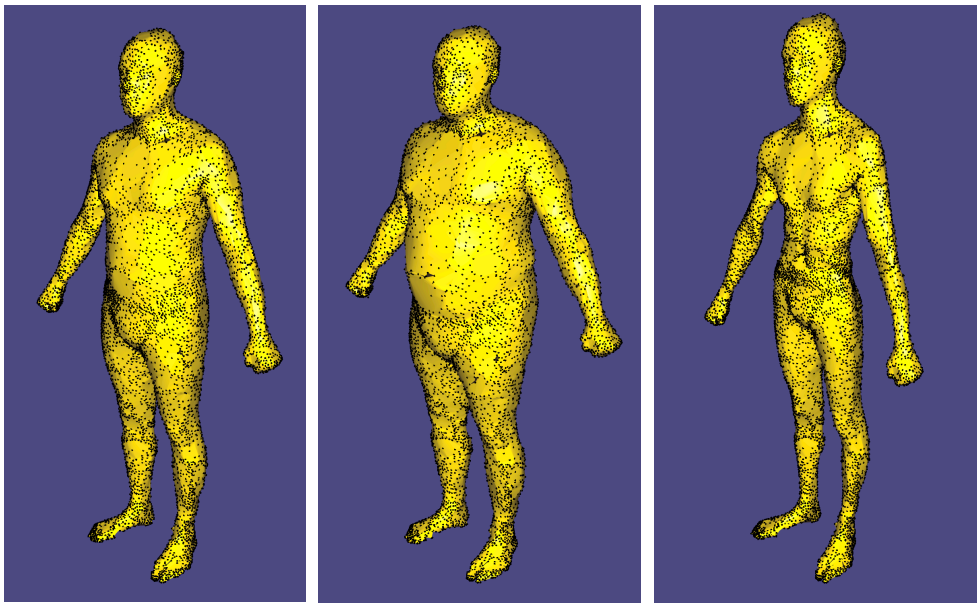
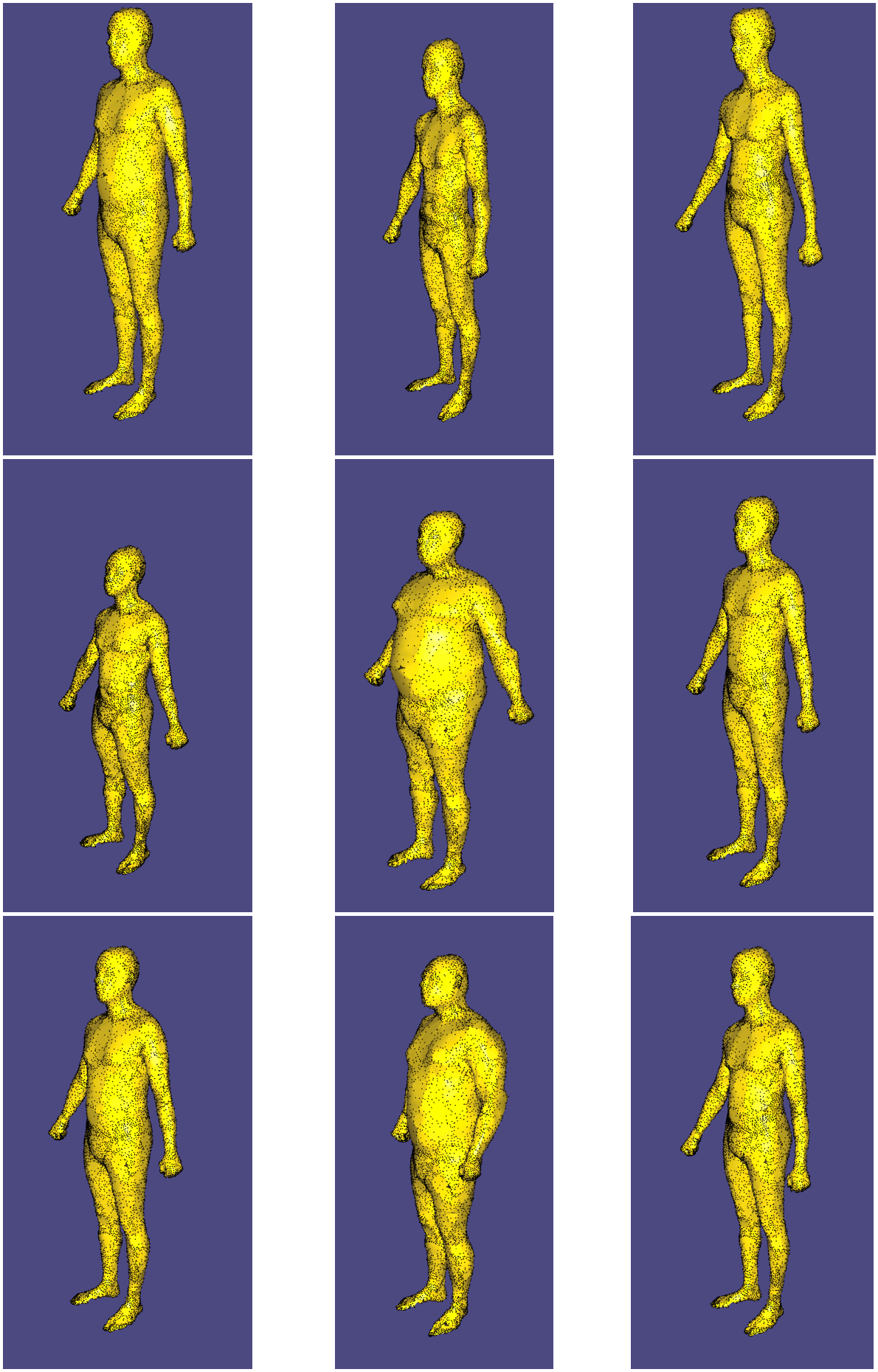


Figure 5 The set of upscaled and transformed PCA point clouds and their accompanying faces

EVALUATION

RESULTS

Having concluded the project, I will now evaluate the progress we made during it. This project was based solely on the work described within the paper "The space of human body shapes: reconstruction and parameterization from range scans" [1]. However, as mentioned previously we only had access to full body models rather than 3d range scans and marker data. This limited how much of the paper we could focus on implementing. Given the limitation of data, I believe that we provided a good implementation of the paper as we managed to create a PCA based parameterisation of a set of 3D models and went on to create new models based on these. An array of different full resolution body meshes that we managed to generate can be seen below:



I also believe that we gained a full understanding of the said paper, even if we were not able to directly implement the 3D scan parameterisation. This understanding informed all of the discussions that we had during this project. This afforded us the opportunity to extend the algorithms that were described throughout the paper and eventually come up with our own technique for optimization.

It also meant that we could discuss how this project could be further developed in the future. One of the issues that arose with our optimization method was that points that had nearest neighbours that were all close to each other, but far away from the point itself produced inaccurate barycentric coordinates. This produced points in the output mesh that were distanced greatly from the average mesh surface. We curtailed this issue by thresholding each points distance to the neighbouring surface and snapping them onto the surface if their distance was too great. However as can be seen in figure 5 these results were not perfect, with some points producing faces that overlap. This was an area of the project that I would like to look into for further study, using concepts such as remeshing and mesh smoothing.

We also discussed how we could handle the implementation of interpolation between models, using the PCA technique. As we had managed to produce a projection of a given model onto the principle vectors from PCA, we could potentially interpolate between two different point clouds ($\boldsymbol{p}_1$ and $\boldsymbol{p}_2$) by projecting them both onto the principle vectors U and interpolating between the resulting coefficients as follows:

$$\mathrm{U}_{projInterp}(\boldsymbol{p_1}, \boldsymbol{p_2}, \alpha) = \sum_{i=1}^{3n} (\alpha \langle \boldsymbol{p_1}, u_i \rangle + (1-\alpha)\langle \boldsymbol{p_2}, u_i \rangle) u_i$$

This should be able to produce results much like those shown in figure 1.

TEAM EVALUATION

I think that Fen and I worked well together on this project. We had daily discussions and coding sessions during the development process which provided a constant forum to explore and test ideas. We also began development early enough to produce early results and give ourselves time to get through any roadblocks that presented themselves. One such roadblock was the interpretation of the PCA algorithm that was described in the paper. We were initially unsure about the intricate details of this algorithm, as our implementation produced severely deformed models. However, given the time we afforded we were able to move past these issues.

In terms of coding, we both worked on the same codebase through a shared repository and developed most of the algorithms together. Fen produced most of the UI for the application using the igl library while I produced the code for the extension part of the project. The GUI for our application can be seen in figure 6. It allows a user to select how many *k* models to load, how many points of the model to sample during the reduction step and allows the user to generate new models using the 6 most important principle vector scalar sliders.

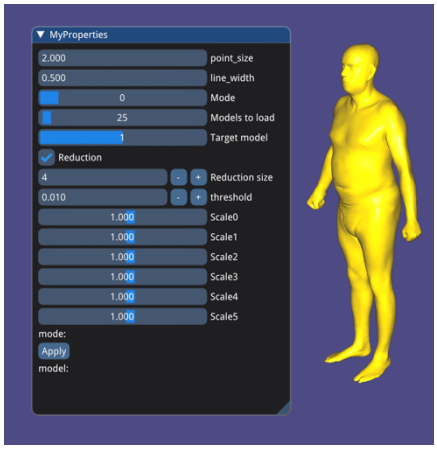


Figure 6 The complete GUI for this project's application

CONCLUSION

This project has resulted in an implementation of PCA based parameterization in C++, which has led to a system that can generate many different new and unique models (figure 5), given a set of initial models. It has given me a good understanding of the capabilities of PCA techniques and how they can be harnessed for use in 3D modelling. It also gave me practical experience in implementing a PCA based algorithm which will be highly valuable in the future. Overall, I believe that this project has been successful in its goal and opens the way for more study in this area.

REFERENCES

[1] B. Allen, B. Curless and Z. Popović, "The space of human body shapes: reconstruction and parameterization from range scans," *ACM Transactions on Graphics (TOG),* vol. 22, no. 3, pp. 587-594 , 2003.

[2] Y. Yang, Y. Yu, Y. Zhou, S. Du, J. Davis and R. Yang, "Semantic Parametric Reshaping of Human Body Models," [Online]. Available: https://graphics.soe.ucsc.edu/data/BodyModels/index.html.