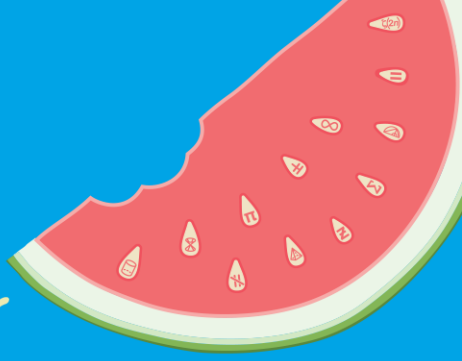
AMSI VACATIONRESEARCH SCHOLARSHIPS 2021–22

Get a taste for Research this Summer



Using Dense Correspondence Between 3D Morphable Faces to Determine Expression

Paimoe Tapsell

Supervised by Syed Zulqarnain Gilani, PhD

Edith Cowan University

AMSI AUSTRALIAN MATHEMATICAL SCIENCES INSTITUTE

Contents

1 Abstract **2**

2 Introduction **2**

3 Statement of Authorship **3**

4 Dense Correspondence **4**

4.1 Goals . . . 4

4.2 Current Work . . . 4

5 Algorithm **5**

5.1 Cost Function . . . 5

5.1.1 Distance Term . . . 5

5.1.2 Stiffness Term . . . 6

5.1.3 Landmark Term . . . 6

5.2 Quadratic Form . . . 7

5.3 Expression Extraction . . . 7

5.3.1 Example: Front View . . . 8

5.3.2 Example: Side View . . . 9

6 Conclusion **10**

7 References **11**

1 Abstract

Dense correspondence is a fundamental problem in computer science. It is a pre-requisite to many applications such as computer vision, facial recognition, and computer graphics. A correspondence between two surfaces is a mapping from one to the other, and has two distinct types. Sparse correspondence maps only selected features of each surface, while dense correspondence aims to match the entire surface. Common deformation mappings for morphable models include rigid, non-rigid, affine and diffeomorphic, which can complicate the process of finding a correspondence. To our knowledge, dense correspondence has not been applied to expression analysis. We show that dense correspondence can be used to extract an expression from two facial surfaces, one neutral and one expressive, using the BU3DFE dataset [6]. We also gained an understanding of the NICP algorithm [1], its applications, and possibilities for further research.

2 Introduction

For two 2D images, a correspondence gives a mapping between similar features in the first image, to the same features in the second. For example, consider the case of matching two images of a house in Figure 1. We know that point A is the same roof, point B the same door, and point C the same path. However, while they look different due to the time of year the picture is taken, the occlusion of the trees, and light levels, a correct correspondence can determine that they are still the same features of the house, and therefore the images represent the same house. The question becomes how are we able to determine this mapping; that is, how can we automatically map similar features from one image to the other, and conclude that it is the same house?

Figure 1: Example of the same house, in different seasons. While it is the same house, in the second image it is less visible due to tree coverage. A correspondence algorithm needs to be robust against this. (Steve Dunwell/Getty Images).

There are two types of correspondence: sparse and dense. The example above describes sparse correspondence. This is a mapping of only several feature points in the image. In an example of facial recognition, those

key features would include the nose, lips and eyes.

Conversely, dense correspondence finds a mapping between all the points in one image onto another. A correct correspondence should be a one-to-one mapping for noiseless and complete surfaces. Certain instances may require cropping, transformation or regularisation. This can occur, for example, in situations where the two images have a different shape.

This correspondence problem also extends to 3D surfaces. In three dimensions, the correspondence is a mapping of key points that exist on both surfaces. Correspondence is only considered for images and surfaces sharing similar key features, otherwise the algorithms are ineffective. For example, two faces or two cars would be appropriate, but not a face and a car.

In this project, the 3D surfaces used were facial scans, with one a neutral pose and the other an expressive pose. The overall goal was to determine if an expression could be extracted after performing dense correspondence between the two non-rigid surfaces. An initial rigid correspondence was made. Then, the Non-rigid Iterative Closest Point algorithm [1] performs dense correspondence between the two surfaces. The difference between the transformed neutral face, and the original neutral face, formed the expression.

3 Statement of Authorship

With the guidance of my supervisor, I ran the scripts provided by the NICP paper using MATLAB and then applied the algorithm to the BU3DFE [6] dataset to extract the expression. The MATLAB code was written by Charlie Nash. This report was written by myself with guidance from Syed Zulqarnain Gilani and Erchuan Zhang.

4 Dense Correspondence

4.1 Goals

The goals of this project were to gain an understanding of the current frameworks that establish dense correspondence between surfaces, specifically the NICP algorithm [1]. We also aimed to understand how establishing a correspondence between two similar faces, one neutral and one expressive, represents an expression. Data cleaning of the BU3DFE dataset and application of the chosen algorithm to this dataset was also pursued. AMSI funded this research.

4.2 Current Work

Correspondence presents a fundamental problem in computer science, particularly in the subfield of computer vision. Determining a correct mapping between two related surfaces, such as two faces, is not a straightforward procedure. For example, scans of faces may have different orientations, can have underlying differences due to either gender or ethnicity [2], or the source faces can have undergone cosmetic change [3]. Computational issues can also arise when the surfaces have different numbers of vertices, making matching difficult, as well, the surfaces can suffer from occlusion, contain artefacts or have missing data.

This project deals with dense correspondence. Sparse correspondence is used to initialise the algorithm, determining a mapping between a subset of vertices known as the landmark points. Then, the algorithm attempts to match the remaining points on our template surface to the target in an optimal and robust way.

The optimal dense correspondence result is a bijection between the two surfaces; a one-to-one mapping. In reality, given the difficulties outlined above, this is often not possible; in those cases, template vertices that do not have any corresponding vertex in the target are ignored. This correspondence will return a matrix $\mathbf{X}$ that will transform our template surface into the target surface. Since we are dealing with human faces for this dataset, and by definition an expression is a transformation from a neutral face to an expressive one, the resulting mapping constitutes the expression itself. That is, a mapping $f : X \to Y$ can represent X as a neutral face, Y as an expressive face, and the mapping f as the expression itself.

The algorithm explored in this project is titled Nonrigid Iterative Closest Point, given in the paper Optimal Step Nonrigid ICP Algorithms for Surface Registration, by Amberg, Romdhani and Vetter (2007) [1]. It incrementally moves the template surface towards the target surface using an optimal affine transformation for each vertex. It also uses a series of stiffness levels to regularise the transformation, which ensures that vertices undergo movement of a similar amount to their neighbours.

It is worth noting that we are treating these surfaces as non-rigid, and performing non-rigid deformation. Rigid deformation is defined as a transformation where the distance between vertices remains the same. This would usually restrict the operations to translation, rotation and reflection. By definition, non-rigid transformation does allow a change in the distance between the points, and allows the establishment of a correspondence between two similar surfaces that have minor changes between them. In our example, this would be a neutral

face and an expressive face.

5 Algorithm

Let $S(\nu_n, \epsilon)$, $T(\mu_m, \delta)$ be the template and target surfaces, respectively, where ν_n, μ_m denote vertices and ϵ, δ denote edges. In what follows, we may use S and T without specifying vertices and edges for simplicity. S_T will denote the transformed template surface.

The algorithm determines locally optimal affine transformations that deform the surface S towards the surface T. Our surfaces are structured as triangulated meshes.

For each vertex v_n in the template, a k-nearest neighbours search determines the closest point u_n in the target template T. After, given that each v_n has a destination, the optimal deformation matrix $\mathbf{X}$ is determined, taking into account the current stiffness level. Initially, the stiffness is high, meaning a less flexible template, and less of a deformation. The optimal deformation is then applied to the vertices, resulting in $S_T = \nu(\mathbf{X})$. The stiffness is then lowered, the k-nearest neighbours search repeats, until the algorithm terminates when the given cost function has been minimised enough (in our case, 10^{-4}).

5.1 Cost Function

The cost function to be minimised contains three parts: the distance between S and T, E_d; the difference between transformations of neighbouring vertices E_s; and the distance between the initial landmark terms E_l; giving

$$E(\mathbf{X}) = E_d(\mathbf{X}) + \alpha E_s(\mathbf{X}) + \beta E_l(\mathbf{X}) \tag{1}$$

where α and β give the weights for the stiffness and landmark terms, respectively. The weight α influences the flexibility, while β is used to deprioritise landmark terms towards the end of the algorithm. Expanding on this function and rewriting into a canonical form allows for differentiation, and therefore minimisation.

5.1.1 Distance Term

The distance term E_d is defined as

$$E_d(\mathbf{X}) = \sum_{v_i \in S} w_i \|X_i v_i - u_i\|^2 \tag{2}$$

where w are the associated weights and X the unknowns. The distance term assumes fixed correspondences, and that both surfaces are in the same coordinate system $v_i = [x, y, z, 1]^T$. The target surface vertices are given by u_i, and the transformed template surface vertices $X_i v_i$. We sum the weighted distance between the paired vertices in the template and target. This can be rewritten as

$$E_d(\mathbf{X}) = \|(W \otimes I_3)(\mathbf{X}_d \mathbf{v}^T - \mathbf{u}^T)\|^2 \tag{3}$$

where $W = \text{diag}(w_i, ..., w_n)$, I_3 is the 3x3 identity matrix, $\otimes$ denotes the Kronecker product, and with $\mathbf{X}_d$ denoting a diagonal matrix of unknowns. The weights vector w is set to 1 when a corresponding vertex is found, and 0 otherwise. These weights can also be manually adjusted.

This form is not easy to differentiate. Therefore, Amberg et. al. [1] use a sparse matrix $\mathbf{D} = \text{diag}(v_1^T, v_2^T, ..., v_n^T)$, and arrange the target points such that $U = [u_1, ..., u_n]^T$, giving the canonical form

$$E_d(\mathbf{X}) = \|\mathbf{W}(\mathbf{DX} - \mathbf{U})\|_F^2 \tag{4}$$

where $\|\cdot\|_F$ represents the Frobenius norm.

5.1.2 Stiffness Term

$$E_s(\mathbf{X}) = \sum_{i,j \in \epsilon} \|(\mathbf{X}_i - \mathbf{X}_j G)\|_F^2 \tag{5}$$

A series of descending stiffness values α are used, which penalise differences of transformations between neighbouring vertices. That is, two adjacent vertices should not move much relative to one another, keeping the local deformations small while retaining the global, larger deformations. As the stiffness level is lowered, local deformations are increased. In our project, the stiffness values were $[50, 20, 5, 2, 0.8, 0.5, 0.35, 0.2]$.

The stiffness term can be represented as a node-arc incidence matrix [4]. Let M be a matrix with rows for each edge ϵ and columns for each vertex ν. If an edge connects the vertices (i, j), then the nonzero entries in the row for that edge are $M_{\epsilon i} = -1$ and $M_{\epsilon j} = 1$. In Equation (5), $G = \text{diag}(1, 1, 1, \gamma)$, though in our case $\gamma = 1$ regardless. Using this, we can rewrite (5) into a canonical form as

$$E_s(\mathbf{X}) = \|(M \otimes G)\mathbf{X}\|_F^2 \tag{6}$$

where $\otimes$ again denotes the Kronecker product and $\|\cdot\|_F$ represents the Frobenius norm.

5.1.3 Landmark Term

$$E_l = \sum_{(v_i, l) \in L} \|\mathbf{X}_i v_i - \mathbf{l}\|^2 \tag{7}$$

The landmark term uses the initial landmark points, and finds the distance between their transformed forms, and the landmark vertices in the target T. This component of the cost function mirrors the distance function, only restricted to the given landmark terms L.

We can use Equation (4) to bring this into a canonical form as well. Let $\mathbf{D}_L$ denote only the rows containing the landmark vertices from $\mathbf{D}$, and $\mathbf{U}_L$ the same for the target vertices. Then,

$$E_l(\mathbf{X}) = \|\mathbf{D}_L\mathbf{X} - \mathbf{U}_L\|_F^2 \tag{8}$$

5.2 Quadratic Form

Combining Equations (4), (6) and (8) gives the quadratic function

$$\begin{aligned} E(\mathbf{X}) &= \left\| \begin{bmatrix} \alpha\mathbf{M} \otimes \mathbf{G} \\ \mathbf{WD} \\ \beta\mathbf{D}_L \end{bmatrix} \mathbf{X} - \begin{bmatrix} \mathbf{0} \\ \mathbf{WU} \\ \mathbf{U}_L \end{bmatrix} \right\|_F^2 \\ &=: \|\mathbf{AX} - \mathbf{B}\|_F^2, \end{aligned} \tag{9}$$

where $\mathbf{A} = \begin{bmatrix} \alpha\mathbf{M} \otimes \mathbf{G} \\ \mathbf{WD} \\ \beta\mathbf{D}_L \end{bmatrix}$, $\mathbf{B} = \begin{bmatrix} \mathbf{0} \\ \mathbf{WU} \\ \mathbf{U}_L \end{bmatrix}$.

This can be minimised exactly by setting its derivative to zero. After solving the system of equations, the minimum is given by

$$\mathbf{X} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{B} \tag{10}$$

This gives the optimal deformation which minimises the cost function, for our current level of stiffness. After applying this transformation to our template surface with

$$S_T = S\mathbf{X} \tag{11}$$

the stiffness is lowered, and we repeat the algorithm, starting with acquiring a new set of correspondences using k-nearest neighbour. If

$$\|S_T - S\| < 10^{-4} \tag{12}$$

the algorithm terminates, with S_T representing the transformed template vertices that approximates T.

5.3 Expression Extraction

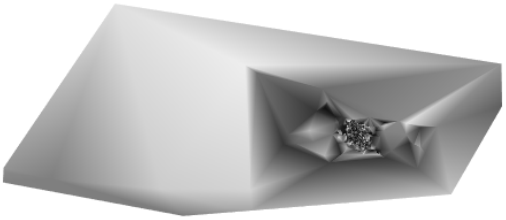
Since $S_T \approx T$, we can simply take the difference $S_T - S$ and get the expression itself. In this context the expression is the operations to apply to our template surface, in order for it to closely approximate the target surface.

5.3.1 Example: Front View

Template Surface S

Target Surface T

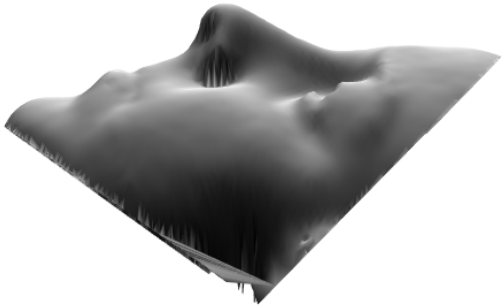
Transformed Template S_T



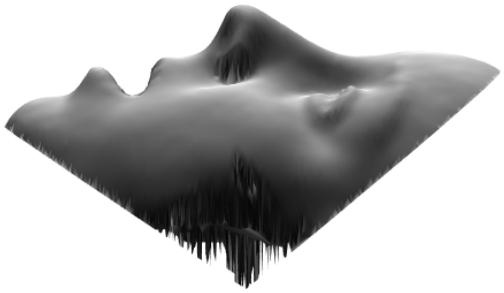
Expression (visual, rotated)

Our template surface S and target surface T are shown above. After the dense correspondence algorithm finishes, the result is the transformed template S_T. This closely approximates (visually) the target surface, which would be a perfect correspondence. The extracted expression visually is not useful, but presented as an example.

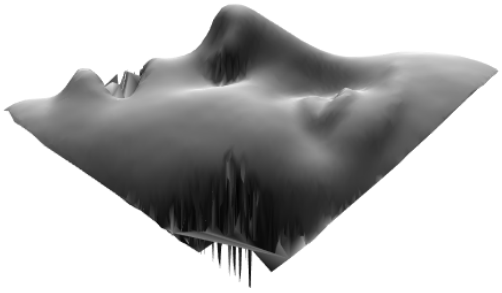
5.3.2 Example: Side View



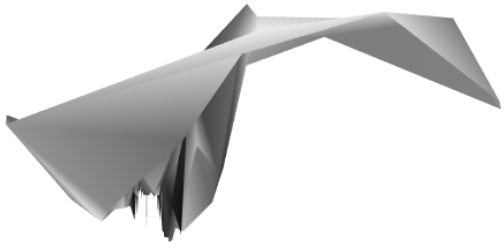
Template Surface S



Target Surface T



Transformed Template S_T



Expression (visual)

6 Conclusion

The NICP algorithm provided an ability to build a dense correspondence between two non-rigid surfaces. When these surfaces are human faces, the correspondence becomes the expression.

Further research into the NICP algorithm would aim to find a correspondence between multiple faces and attempt to extract the expression from each of them. This would give the ability to add and remove expressions on any face in the corresponded set.

An issue that appeared in this research is artefacts in the surfaces, which complicate the correspondence process. A more robust approach that handled noisy data well would help the algorithm handle real-world data. As well, if the neutral template surface still contained some expression (for example, a small smile or expressive eyes) then the correspondence process can suffer from incorrect registration. Further work should attempt to handle these cases.

7 References

1. B. Amberg, S. Romdhani and T. Vetter, 2007, 'Optimal Step Nonrigid ICP Algorithms for Surface Registration', Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on, pp 1-8.

2. J. Booth, A. Roussos, S. Zafeiriou, A. Ponniah and D. Dunaway, 2016, 'A 3D Morphable Model learnt from 10,000 faces', IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

3. A. Chanchua and N. Chentanez, 2021, 'DeltaFace: Fully Automatic 3D Facial Cosmetic Surgery Simulation', 25th International Computer Science and Engineering Conference (ICSEC), pp. 246-251, doi: 10.1109/ICSEC53205.2021.9684623.

4. M. Dekker, 1986, 'Mathematical Programming', CRC.

5. S. Z. Gilani, A. Mian, F. Shafait and I. Reid, 2018, 'Dense 3D Face Correspondence', IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol 40, No. 7.

6. L. Yin, X. Wei, Y. Sun, J. Wang and M. H. Rosato, 2006, 'A 3D facial expression database for facial behavior research', 7th International Conference on Automatic Face and Gesture Recognition, 10-12 April 2006, pp 211-216.