October 30, 2012

Surface to Surface Rigid Alignment

Uri Merhav [1], Shay Dekel, Guy Lavi[2]

Abstract

We explain the fundamentals of ICP algorithm, a method for aligning pairs of 3D geometrical surface models, and discuss numerical feasibility and acceleration techniques. Successful alignment of two real-world 3D surfaces is shown.

1 Introduction

The problem of matching pairs of 3D geometrical surface models and measuring their resemblance is common both in academic literature and commercial applications [4]. One application of rigid shape alignment would be 3D classification of a human face as outlined in [1]. Another example, implemented in 2D, is the widely successful biometric verification of a person's identity based on retinal shape [5]. This article discusses the theory and implementation of a dominant technique in rigid surface alignment, called the Iterative Closest Point (ICP). This approach was implemented to meet a customer's stringent requirements in the course of field testing a novel medical device.

2 Goal

During the conduct of field testing, our customer needed a robust, efficient solution for comparing various surface reconstructions of a body organ. Given two surfaces in 3D, that are defined by a set of vertices and faces, our goal is to register the two items with respect to one another, i.e. to align their surfaces such that they overlap as much as possible.

3 Method

3.1 Representation of a Surface Mesh

The surface of a body $\mathbf{X}$ in 3D space may be defined as a set of point triplets, each triplet defining a single face of the object. Formally, we may define an object $\mathbf{X}$ using the following

[1] uri@volumeelements.com
[2] guy@volumeelements.com

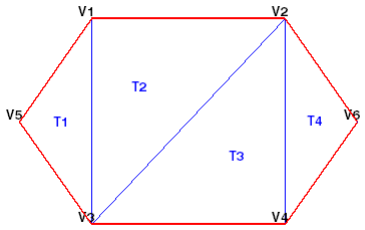
array of vertices:

$$\mathbf{X} = \begin{pmatrix} v_{11} & v_{12} & v_{13} \\ v_{21} & v_{22} & v_{23} \\ . & . & . \\ . & . & . \\ v_{N1} & v_{N2} & v_{N3} \end{pmatrix} \tag{1}$$

An example for this definition is shown in fig. 1.

Figure 1: Example of a surface defined by a sequence of vertex triplets, the surface in this example may be defined by :

$$\mathbf{X} = \begin{pmatrix} v_1 & v_3 & v_5 \\ v_1 & v_2 & v_3 \\ v_2 & v_4 & v_3 \\ v_2 & v_6 & v_4 \end{pmatrix} \tag{2}$$



3.2 Surface Alignment

3.2.1 Algorithm Outline

Goal: given two shapes $\mathbf{X}$ and $\mathbf{Y}$, rotate and translate $\mathbf{Y}$ using rotation matrix $\mathcal{R}$ and translation matrix $\mathcal{T}$ such that $\mathbf{Y}' = \mathcal{R}\mathbf{Y} + \mathcal{T}$. The goal is to find *optimal* $\mathcal{R}$ and $\mathcal{T}$ such that

$$d_{optimal} = \min_{\mathcal{R},\mathcal{T}} d(\mathbf{X}, \mathbf{Y}'). \tag{3}$$

The algorthimic scheme is as follows:

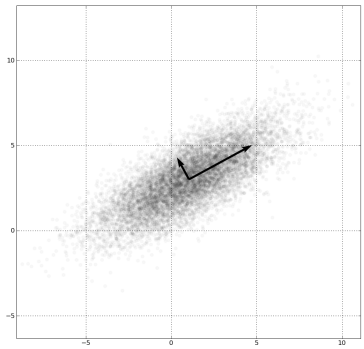
1. As a pre-processing step, align the surfaces $\mathbf{X}$ and $\mathbf{Y}$ such that their geometric moments are the same, i.e.: choose $\mathcal{T}$ and $\mathcal{R}$ such that the two surfaces have, as an initial guess, the same *canonical configuration*.

2. Calculate d for the current $\mathcal{R}$ and $\mathcal{T}$.

3. Iterate small changes in $\mathcal{R}, \mathcal{T}$ until convergence to a minimum in d.

For the optimization algorithm to be well defined, we need to define the *canonical basis* and define a *distance function*. As for the minimization process itself, only key features will be mentioned. A numerical recipe for ICP implementation will be mentioned without proof.

Figure 2: A 2D example of principal component analysis. Note that the longer arrow is the first principal component. The smaller arrow is the second principal component. It is perpendicular to the first component, as is always the case with any pair of principal components.



3.2.2 Canonical Basis

As in many other numerical methods, a decent first guess is key for correct convergence of ICP. A poor initial alignment of the two shapes $\mathbf{X}$ and $\mathbf{Y}$ may lead to slow convergence or local minimum. In order to improve the prospects of convergence, the pre-processing is comprised of two steps:

1. Approximate the center of mass of both surface by the vertices $\mathbf{x_i}$, $\mathbf{y_i}$ that define them

$$\mathbf{M_X} = \sum_i \mathbf{x_i}, \tag{4}$$

$$\mathbf{M_Y} = \sum_i \mathbf{y_i}. \tag{5}$$

   Define $\mathcal{T} = \mathbf{M_X} - \mathbf{M_Y}$. This eliminates any major translation errors between the two surfaces.

2. In order to eliminate rotational ambiguity, align the two shapes such that their *principal components* point in the same directions. A formal definition of principal components would be: the eigenvectors of the covariance matrix of vertices that comprise the surface, ranked by their corresponding eigenvalues from largest to smallest. Intuitively, if we were to project all the vertices onto a single vector (a single spatial direction), the first principal component is the direction along which the variance of the projection would be maximized. The second principle component maximizes the variance in the residual dimensions, and so on until the last dimension, which in the 3D case would be the third dimension. A thorough discussion of PCA may be found in the literature [6].

WEBSITE: www.volumeelements.com E-MAIL:info@volumeelements.com TEL:+972-9-7964646

3.2.3 Shape to Shape Distance Function

A variety of distance functions that measure the incongruity between two surfaces may be defined. For our application we chose a one way distance function that is relatively robust to outliers, that is the squared distance norm. We treat object $\mathbf{X}$ vertices as a cloud of points $\mathbf{x_i}$, and define

$$d(Y', X) = \sum_i d(\mathbf{x_i}, \mathbf{Y})^2, \tag{6}$$

whereas d is an approximation of the distance between $\mathbf{x_i}$ and the shape $\mathbf{Y}$. A simple definition of the distance function would be to treat $\mathbf{Y}$ as a cloud of points as well and measure the minimal point to point distance. That turned out to be sufficient in our application, but we caution against using such a metric without bounding the vertex point to point distance. If the distance between vertices is unbounded or large, point to point metrics will impair performance, as locations on the surface defined by a triangle may be far away from its vertices . In the first order of correction, we measure the point to plane distance between each *point* x_i and the nearest approximate *plane* defined in $\mathbf{Y}$, namely

$$d^2 = \left(\mathbf{N}(\mathbf{Y}) \cdot \mathbf{x_i}\right)^2, \tag{7}$$

with $\mathbf{N}(\mathbf{Y})$ being a vector pointing in the normal to the plane in $\mathbf{Y}$ that is nearest to the vertex $\mathbf{x_i}$, which is also a vector. It's worth noting that $d(\mathbf{X}, \mathbf{Y}) \neq d(\mathbf{Y}, \mathbf{X})$, which becomes evident if one considers one shape that is a subset of the other, larger shape. Distance functions that are symmetric (i.e. $d(\mathbf{X}, \mathbf{Y}) = d(\mathbf{Y}, \mathbf{X})$) perform poorly for partial matches, as was required for our application. Hence a one sided square norm distance function was chosen in our implementation.

3.2.4 Optimization steps

In our implementation of ICP, we treat the vertices of both shapes as two clouds of points. The ICP procedure is as follows:

1. For each point in $\mathbf{x_i}$, find the closest point in $\mathbf{Y}$. Note: it is preferable that $\mathbf{Y}$ be the larger surface of the two in case of partial matches. Also note that the resultant data set $\mathbf{y_i}$ is guarnteed to include the same number of points as $\mathbf{x_i}$, and may include repeated values of the same $\mathbf{y_i}$.

2. Given $\mathbf{x_i}$ and the corresponding $\mathbf{y_i}$ found in the last step, the goal is now to find $\mathcal{R}$ and $\mathcal{T}$ that minimize

$$E(\mathcal{R}, \mathcal{T}) = \sum_{i=1}^{N} \|\mathbf{y_i} - (\mathcal{R}\mathbf{x_i} + \mathcal{T})\|^2. \tag{8}$$

   Solving the above optimization problem using SVD decomposition is rather straightforward [2]. For completeness, the method is henceforth described in brevity.

   (a) define:

$$\tilde{\mathbf{x}}_\mathbf{i} = \mathbf{x_i} - \sum_i \mathbf{x_i}/N \tag{9}$$

Correspondingly for the subset of matching $\mathbf{y_i}$

$$\tilde{\mathbf{y}}_\mathbf{i} = \mathbf{y_i} - \sum_i \mathbf{y_i}/N \tag{10}$$

(b) Calculate

$$\mathbf{H} = \tilde{\mathbf{x}}_\mathbf{i}^T \cdot \tilde{\mathbf{y}}_\mathbf{i}. \tag{11}$$

Note that this is the *Cross Correlation* matrix between the two datasets.

(c) Perform SVD decomposition

$$\mathbf{H} = \mathbf{U}\mathbf{\Lambda}\mathbf{V} \tag{12}$$

(d) (8) is minimized by:

$$\mathcal{R} = \mathbf{V} \cdot \mathbf{U}^T \tag{13}$$

$$\mathcal{T} = \sum_i \mathbf{y_i}/N - \mathcal{R} \sum_i \mathbf{x_i}/N \tag{14}$$

3. repeat steps (1-2) until E stops changing below a predefined threshold and the algorithm converges.

3.2.5 K-d Trees

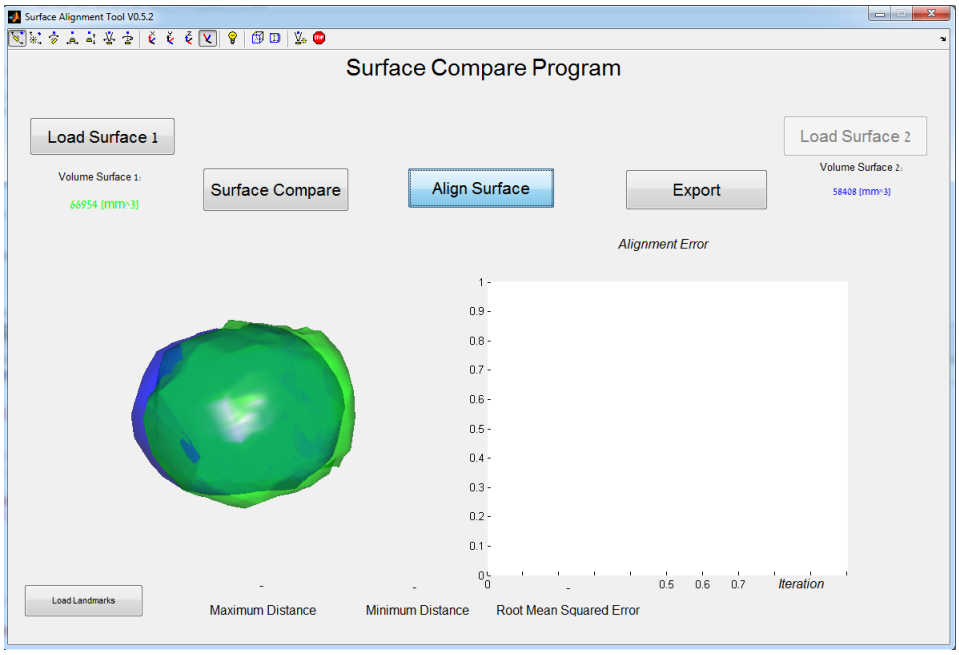
Our discussion so far was done free of considerations of processing time for numerical convergence. In this article we will only touch upon one key aspect of the ICP computations: finding the nearest neighbour to a given point $\mathbf{x_i}$ from the cloud $\mathbf{y_i}$.

Assuming N points define $\mathbf{Y}$ and M points define $\mathbf{X}$, solving the problem for a single $\mathbf{x_i}$ is $O(M)$, and so $O(NM)$ computations are required for each pass in the ICP algorithm. Runtime may be improved significantly if one uses a k-d tree to partition the volume enclosed by the vertices.

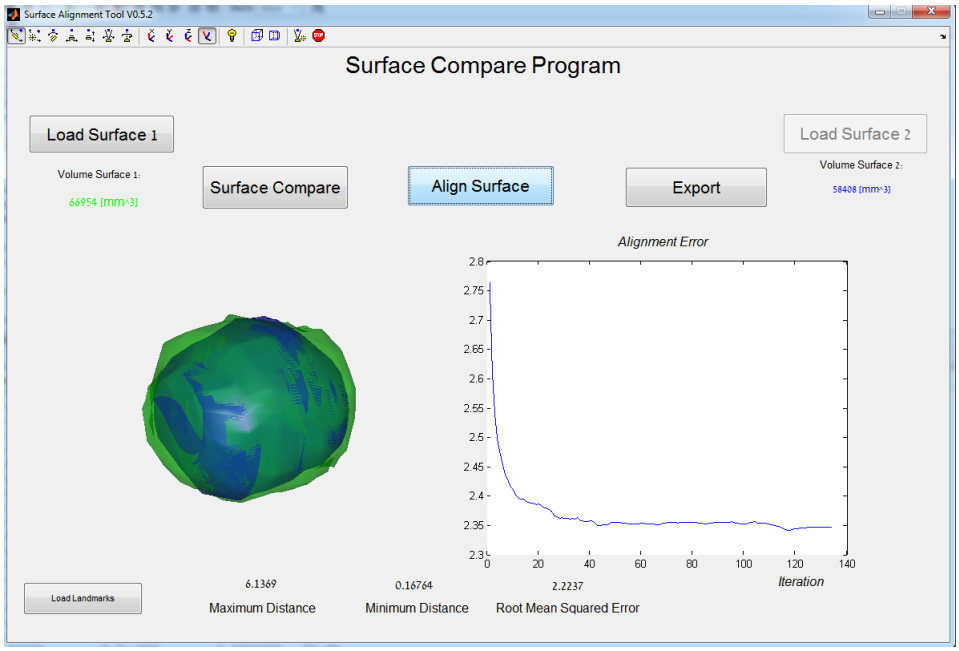
The k-d tree is just a binary search tree in which each node represents the partition of the k-dimensional space, and each leaf of the tree represent a small rectangular box in space. Correct implementation of a k-d tree allows us to find the closest points in $\mathbf{y_i}$ to any given point in the volume enclosed by $\mathbf{Y}$ in $O\left(log(M)\right)$, Thus each ICP iteration of shape to shape distance only costs $Nlog(M)$ calculations. Computational costs may be further reduced if one relaxes the requirement of Nearest Neighbour to those of Approximate Nearest Neighbour - that is, finding a neighbour that is ϵ close to the actual nearest neigbhour, with ϵ a predetermined bound [3].

4 Results

A full implementation of ICP was developed in MATLAB environment and deployed with compiled MEX files for improved runtime. Surfaces comprised of $O(10^4)$ triangles complete entire ICP registrations in order of 1 second. The code is incorporated in standard tests, delivering a solution that is both computationally feasible, and mathematically optimal in some sense (see fig 3).



(a) Initial iteration. Two measurements of the same objects imaged in 3D using different physical methods. Note that the two objects are both translated and rotated in relation to another.



(b) Final iteration of ICP. On the right one can also note in this figure that the error metric is non-montonically declining, which is a known drawback of ICP implementations.

Figure 3: ICP implementation developed by *Volume Elements*

WEBSITE: www.volumeelements.com E-MAIL:info@volumeelements.com TEL:+972-9-7964646

References

[1] J. Cook, V. Chandran, S. Sridharan, and C. Fookes. Face recognition from 3d data using iterative closest point algorithm and gaussian mixture models. In *3D Data Processing, Visualization and Transmission, 2004. 3DPVT 2004. Proceedings. 2nd International Symposium on*, pages 502 – 509, sept. 2004.

[2] Shaoyi Du, Nanning Zheng, Shihui Ying, and Jishang Wei. Icp with bounded scale for registration of m-d point sets. In *Multimedia and Expo, 2007 IEEE International Conference on*, pages 1291 –1294, july 2007.

[3] M. Greenspan and M. Yurick. Approximate k-d tree search for efficient icp. In *3-D Digital Imaging and Modeling, 2003. 3DIM 2003. Proceedings. Fourth International Conference on*, pages 442 – 448, oct. 2003.

[4] S. Rusinkiewicz and M. Levoy. Efficient variants of the icp algorithm. In *3-D Digital Imaging and Modeling, 2001. Proceedings. Third International Conference on*, pages 145 –152, 2001.

[5] C.V. Stewart, Chia-Ling Tsai, and B. Roysam. The dual-bootstrap iterative closest point algorithm with application to retinal image registration. *Medical Imaging, IEEE Transactions on*, 22(11):1379 –1394, nov. 2003.

[6] Svante Wold, Kim Esbensen, and Paul Geladi. Principal component analysis. *Chemometrics and Intelligent Laboratory Systems*, 2(1–3):37 – 52, 1987. Proceedings of the Multivariate Statistical Workshop for Geologists and Geochemists.

WEBSITE: www.volumeelements.com E-MAIL:info@volumeelements.com TEL:+972-9-7964646