3D Correspondence by Breadth-First Search Frontiers

Yusuf Sahillioğlu
Computer Science Dept., Koç University, Istanbul, 34450, Turkey

Abstract— *This paper presents a novel, robust, and fast 3D shape correspondence algorithm applicable to the two snapshots of the same object in arbitrary deformation. Given two such frames as triangle meshes with fixed connectivity, our algorithm first classifies vertices into Breadth-First Search (BFS) frontiers according to their unweighted shortest path distance from a source vertex. This is followed by the rigid or non-rigid alignment of the corresponding frontiers of two meshes as the second and final step.*

This algorithm is flexible; high-resolution meshes are welcome. It is robust; results approved by human intuition as well as our own numerical correspondence error metric. It is fast; sequential running time turns out to be quadratic in number of vertices, whereas this upper bound can be pulled down as low as subquadratic $O(V^{1.5})$ once the second step is naturally and easily parallelized. Due to consistent frontier selection, second step does the optimal work of $O(V^3)$ Hungarian assignment in less than a quadratic time.

Keywords: 3D shape correspondence, BFS

1. Introduction

The shape correspondence problem takes two snapshots of the same shape, or two different shapes, and tries to determine where a vertex in the first shape is transformed in the second one. This task has received great attention in the computer graphics and computer vision disciplines mainly because it stands as the initial problem to be solved for generating many useful applications, such as shape morphing/deformation, alignment, and animation. Lee [1] have provided a pioneering morphing application when the correspondence between input shape and target shape is known. Accuracy of the well-known Iterative Closest Point (ICP) alignment algorithm by Besl and Mckay [2] can be improved further if in each iteration ICP is forced to associate vertices that are known to be corresponding pairs. A fundamental animation technique known as keyframe animation uses the correspondence information between two keyframes to compute those in between. Correspondence can also be of help to simplify the shape matching problem where one tries to return the most similar item(s) to a given query. Another, yet not last, application area of this problem is in texture mapping as the work of Kraevoy [10] forces feature correspondence for planar parameterization of meshes.

Jain and Zhang [3] have provided a satisfactory shape correspondence method whose efficiency and accuracy decrease as the resolution of shapes, i.e., number of vertices, increases. Mateus et al. [4], on the other hand, have employed the Graph Laplacian operator to address this problem. In addition to being inefficient for large inputs, the adaptation of this voxel-based method to mesh-based representation tries to pair vertices only with the same connectivity information. However, this is not always the case. This method is only valid when one shape is exactly the transformed version of the other, i.e., the vertices are displaced but the connectivity is fixed. We currently have this restriction too, but it will be easier to drop it in our case. It should also be noted that there are nevertheless applications that use fixed connectivity input; e.g., for the transmission of a large animation sequence, one can transmit only few keyframes over the poor network and interpolate the in-betweens in the host with a fast correspondence algorithm like ours.

Umeyama [8] has managed to produce binary correspondence matrix between two graphs with same number of vertices using the eigendecomposition of their adjacency matrices. He implied the fixed connectivity requirement too since he establishes the correspondence that respects vertex degrees. Scott and Higgins [9] have extended solutions to weighted graph matching problem by introducing Gaussian proximity matrix.

From an asymptotic complexity view, our spatial-domain approach outperforms existing methods that work in spectral domain [11], [12], [8] as well as in spatial domain [13], [7] while producing competitive results. On the other hand, most of these aforementioned methods do not require fixed connectivity and same vertex count requirement that we currently do.

The rest of the paper is organized as follows. In Section 2, we state the problem rigorously and define the objects used for the correspondence algorithm we describe at Section 3. Section 4 provides experimental results, followed by the the future work and conclusion on Section 5 and 6, respectively.

2. Problem Statement

Given two separate frames obtained from the motion of an object, we want to track any given vertex, hence the correspondence problem. To accomplish this, we make use of several components defined on both input meshes.

2.1 Frontiers

Frontiers are subset of vertices that have the same unweigthed shortest path distance from a given source vertex

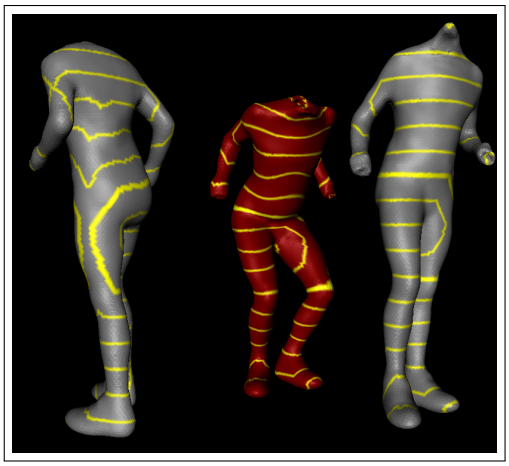

Fig. 1: Several frontiers of distance $10m$, for several m, $from the source at right foot toe$.

(Fig. 1). Vertices that go into a given frontier are invariant to rigid body transformations, scaling, and bending; making frontiers valuable components for shape correspondence.

One application of modified BFS computes this subset for two meshes. Number of frontiers, f, is relatively small, e.g., 130 for a $16K$ mesh, yet sufficiently large to guarantee $f = \Omega(\sqrt{V})$. This lower bound is necessary to achieve the claimed overall complexity and simplifies the asymptotic run-time analysis. (Another simplification is $E \leq 3V - 6 = O(V)$ and omitting E for sparse 2-manifold meshes (= simple planar graphs). Although $f = \Omega(\sqrt{V})$ is achieved naturally for our test cases, it can be forced via edge splits that introduce additional layers between existing layers.

2.2 Base vertices

Base vertices are subset of vertices that are uniformly distributed over the whole mesh (Fig. 2). Modified Dijkstra algorithm by Hilaga [5] computes this subset, which will be useful for source vertex selection in frontier generation as well as for our fast correspondence error metric.

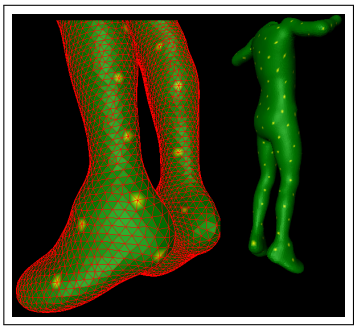

Fig. 2: 180 base vertices (yellow) sampled on our high-resolution input $16K$ mesh (green).

3. Correspondence Algorithm

This algorithm first establishes a corresponding source vertex for both meshes from which BFS will be launched to end up with the desired frontiers. The k^{th} frontier is the set of vertices that are of distance k from the source. Therefore, true correspondences of the vertices in k^{th} frontier of the first mesh must lie in exactly the k^{th} frontier of the second mesh, which triggers us to align these corresponding frontiers.

3.1 Source selection

In order to obtain frontiers consistently, one needs to start the frontier computation from the same/corresponding source vertex in both meshes. Although this selection could easily be done manually, for the sake of keeping system fully automated, we suggest a method that uses critical points that can be computed by Katz [6]. Once the critical points are achieved, we select the most critical one for each mesh as their source.

3.2 Frontier computation

We run BFS from the source vertex and add the vertices of the same distance to the same row of our *frontiers* matrix. Under the fixed connectivity assumption, it is certain that corresponding frontiers will involve the true vertex correspondences (possibly displaced though); however, once the connectivity changes, true correspondences may fall into different frontiers that are not paired up (Fig. 3). This limitation can be dropped by using shortcut edges of [5] to make the directions of edges isotropic.

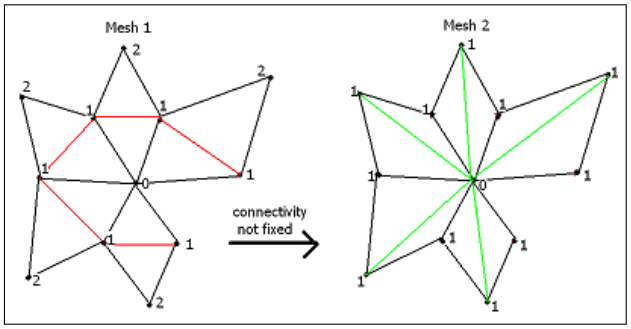

Fig. 3: Under arbitrary connectivity, 6 vertices of left mesh in *frontiers[1]* are paired up with 12 in *frontiers[1]* of right mesh. Note that, shortcut edges make directions isotropic, i.e., union of red and green edges, and heal the problem..

3.3 Correspondences by frontiers

Having obtained consistent frontiers from two meshes, final step is to pair them up. An example of two corresponding frontiers is given in Fig. 4.

Some frontiers correspond to the merely-translated portion of the object for which a center of mass alignment is sufficient. For the ones that exhibit significant amount of

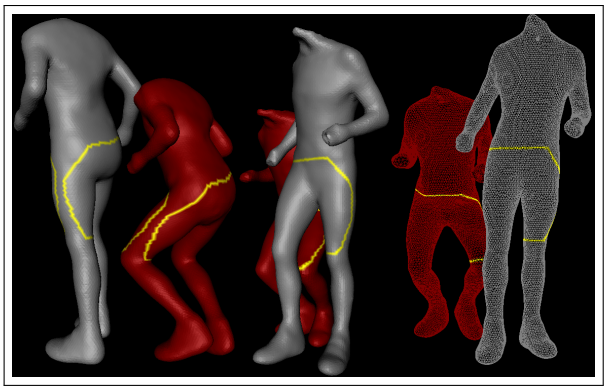

Fig. 4: Two corresponding (k^{th}) frontiers to be paired up. Note that true correspondences of vertices in k^{th} frontier of gray mesh must lie on k^{th} frontier of the red one.

change in geometry, e.g., rotated or bent portions, we use non-rigid iterative alignment algorithm (TPS-RPM) of [7]. To make the difference, we sum the squared L2 distances of closest points of the two center of mass aligned frontiers. If this sum S is sufficiently small, then we are satisfied with the alignment already. Otherwise, we employ TPS-RPM, with an initial temperature of $10S$, to align the frontiers under bending or rotation as depicted in Fig. 5. We finally pair the aligned frontiers up with the closest L2 distance metric.

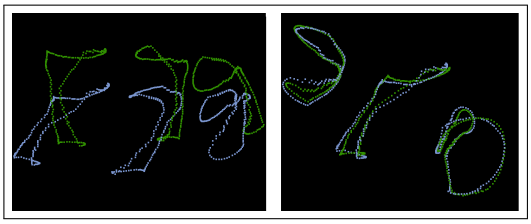

Fig. 5: Two frontiers of Fig. 4 need non-rigid alignment whose result is at right.

Under the observation that corresponding frontiers involve corresponding vertices, going through these two frontiers only once finds the optimal correspondence which is no further required to be updated. So, many updates of Hungarian algorithm [15] in V^3 iterations to get to the optimal is achieved with one update and in much less iterations, which can further be parallelized as described next.

3.4 Run-time analysis

It is $O(VlgV)$ to accomplish the base vertices with modified Dijkstra for source vertex decision. We employ our modified BFS once to obtain frontiers in $O(V)$. Cost of TPS-RPM alignment is $\sum(F_i^3)$ for $i = 1, .., f$, where F_i is the number of vertices in i^{th} frontier. This sum is essentially $O(f \times (\frac{V}{f})^3 = \frac{V^3}{f^2} = \frac{V^3}{V} = V^2)$ by assuming that each frontier is uniform and has same number of vertices of $\frac{V}{f}$. Similarly, pairing up *aligned* frontiers takes $O(f \times (\frac{V}{f})^2 = \frac{V^2}{\sqrt{V}} = V^{1.5})$. Sum of these three terms, BFS, TPS-RPM, and final-pairing, is dominated by $O(V^2)$ TPS-RPM for the sequential case which is far better than a $O(V^3)$ bipartite perfect matching algorithm like Hungarian's.

Despite this impressive asymptotic complexity, there is still room for improvement after observing that correspondence of i^{th} frontiers are independent of others and hence can be parallelized easily. With the availability of f processors or threads, we only pay one TPS-RPM and final-pairing cost instead of f, leading to the the drop of f factor and hence the $O((\frac{V}{f})^3 = \frac{V^3}{V^{1.5}} = V^{1.5})$ subquadratic cost.

4. Results

In addition to visual results that draw subset of corresponding vertices in Fig. 7 through 10, we also provide quantitative results computed by our correspondence error metric introduced next.

4.1 Correspondence error metric

A correspondence error metric is defined as $C = \sum g(i, corresp(i))^2$ for $0 \leq i \leq V$, with g being the geodesic distance between two points, and $corresp$ being the correspondence of i^{th} vertex computed by our algorithm. This metric works because our ground truth values are known as $corresp(i) = i$. g is computed efficiently by running Dijkstra's shortest paths from each base vertex. This fills g for base-to-base, base-to-nonbase, and nonbase-to-base entries. For the remaining nonbase-to-nonbase entries, we simply use the known distances between representative bases of two patches that involve the nonbases. This fast yet accurate heuristic of ours computes geodesic distances between any two points of V vertices in $O(k \times VlgV)$ time with $k \ll V$ being the number of base vertices, instead of the perfectly accurate but $O(V^3)$ time Floyd-Warshall's all-pairs-shortest path algorithm [14]. For large number of vertices, e.g., $V = 16K$, computation is done in a minute instead of a couple of hours with sufficiently accurate results (Fig. 6). Note that, what we quickly create here is the geodesic proximity matrix which can also be of use for other shape correspondence algorithms, such as [3].

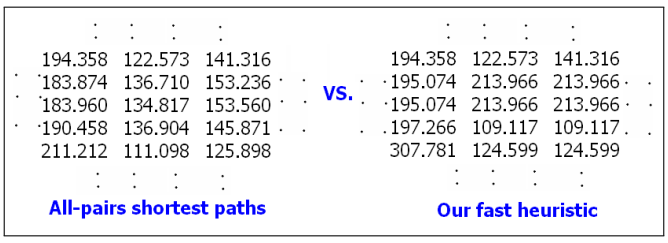

Fig. 6: Same portion of the geodesic distance matrix computed by all-pairs algorithm (left) and our heuristic (right).

4.2 Visual and scalar results

High-quality correspondences between two consecutive frames achieved, an example of which provided in Fig. 7. For the sake of comparison, we implemented naive algorithm that just pairs up closest pairs after center of masses of two meshes made coincide. This naive quadratic matching causes larger $C = 1.7$ in 124 seconds, whereas a better result ($C = 0.09$) was achieved by our algorithm in just 5 seconds. Moreover, the naive quadratic scheme does not change the correspondence of a vertex once it is set. The cure, e.g., a cubic Hungarian scheme, would need a couple of hours to achieve our C for these high-resolution ($16K$) meshes.

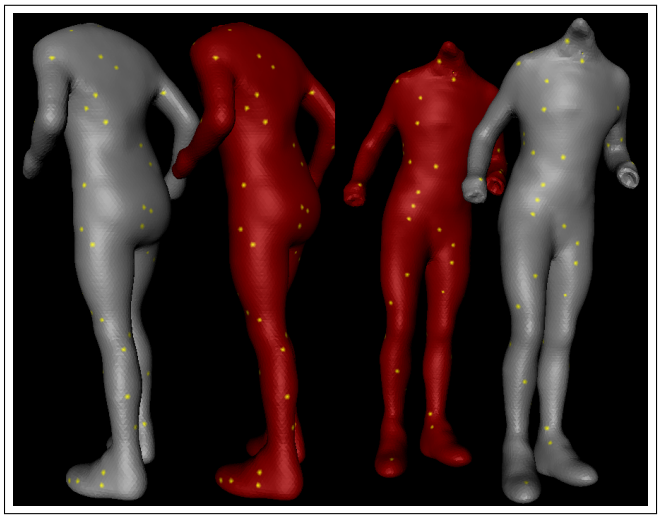

Fig. 7: Correspondence between two consecutive frames with $C = 0.09$.

In order to see how the correspondence is established between apart frames, we run the algorithm with frames that exhibit a significant change in pose, and obtain the result in Fig. 8 with $C = 5.9$. It took 20 minutes in a sequential C++ code due to the frequent TPS-RPM calls that were almost never needed for the 5-second case above. (Note that, MatLab implementation of TPS-RPM is known to be much faster due to frequent matrix operations.) A parallel system which would collapse all TPS-RPM times into one would decrease our running time significantly. Also, naive algorithm finished this correspondence with $C = 105$.

Another test reveals another satisfactory result (shown in Fig. 9) with $C = 7.7$ in 10 minutes on our $2GHz$ dual processor machine while the error of the naive algorithm goes as high as $C = 91$. In Fig. 10, we provide the correspondence result for two high-resolution, e.g., $V = 20K$, input meshes of a different animation sequence. This is obtained after 40 minutes with $C = 180$.

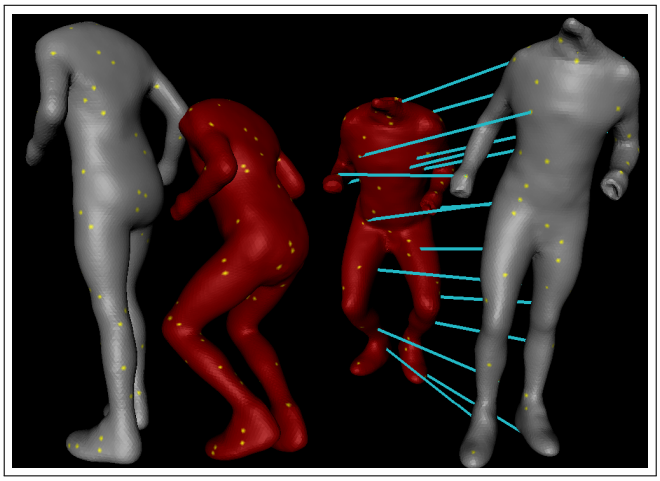

Fig. 8: Correspondence under significant deformation ends with $C = 5.9$.

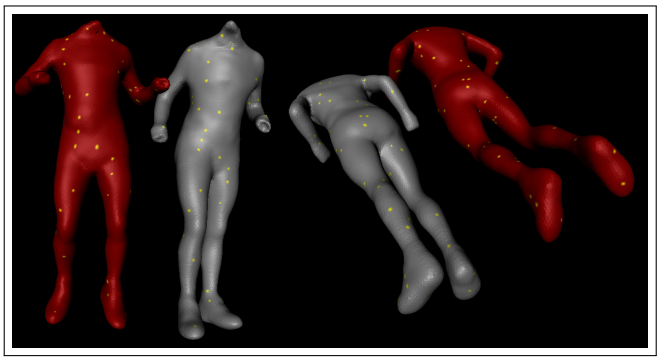

Fig. 9: Randomly picked 100 yellow vertices from first mesh (gray) are displayed with their correspondences in second mesh (red).

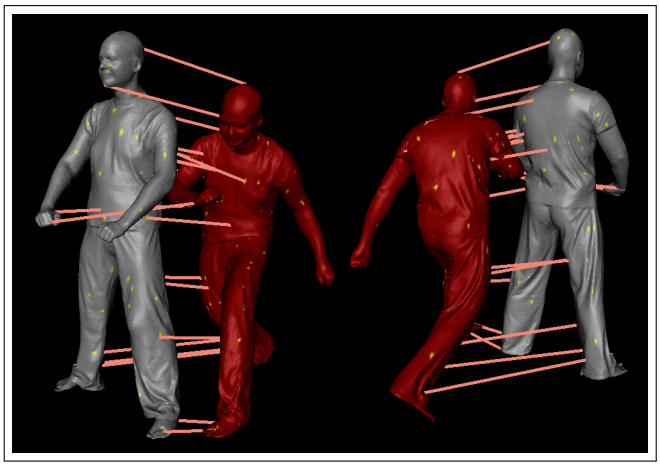

Fig. 10: Correspondence between two apart frames of high-resolution meshes with $20K$ vertices.

5. Comments and Future Work

Using this generic framework, new correspondence metrics other than or in addition to L2 metric can be used during the stage where correspondence between the aligned frontiers are sought. Again in this stage, one can run a cubic bipartite perfect matching algorithm for the most accurate result although this is not so crucial given that our current aligner (TPS-RPM or simple center of mass alignment according to situation) leaves us with almost coinciding frontiers, and therefore the first closest correspondences in the subquadratic final-pairing process will generally be the optimal ones anyway.

Essential future work would be the relaxation of the fixed connectivity constraint with the shortcut edges scheme mentioned in Section 3.2 or with another method. Also, another graph search algorithm instead of the connectivity-based BFS can be investigated to produce the frontiers consistently in the absence of fixed connectivity.

6. Conclusions

A novel, robust, and efficient algorithm that addresses to the problem of 3D shape correspondence between arbitrary poses of the same object is provided.

Starting from the same source vertex in both meshes, which is suggested to be the most critical point, we establish consistent BFS frontiers, and then pair up BFS frontiers of same distance from the source. In order to render frontiers consistent, we rely on fixed connectivity although an arbitrary connectivity may still be successful after some preprocessing that are partly mentioned and mostly left as future work. Besides, we mentioned an application of fixed connectivity input which can be efficiently handled by our current algorithm.

Algorithm is naturally adaptable to other dimensions and works fine with high-resolution meshes. Although the sequential asymptotic complexity is more than acceptable and surpasses related works, it can further be improved once the algorithm is easily parallelized.

References

[1] A. Lee, D. Dobkin, W. Sweldens and P. Schroder, "Multiresolution Mesh Morphing," *ACM SIGGRAPH*, pp. 343–350, 1999.

[2] P. J. Besl and N. D. Mckay, "A method for registration of 3-D Shapes," *PAMI*, vol. 14, pp. 239–256, 1992.

[3] V. Jain and H. Zhang, "Robust 3D Shape Correspondence in the Spectral Domain," *Shape Modeling and Applications*, pp. 118–129, 2006

[4] D. Mateus, R. Horaud, D. Knossow, F. Cuzzolin and E. Boyer, "Articulated Shape Matching Using Laplacian Eigenfunctions and Unsupervised Point Registration," *Computer Vision and Pattern Recognition*, pp. 1–8, 2008.

[5] M. Hilaga, Y. Shinagawa, T. Kohmura and T. Kunii, "Topology Matching for Fully Automatic Similarity Estimation of 3D Shapes," *ACM SIGGRAPH*, pp. 203–212, 2001.

[6] S. Katz, G. Leifman and A. Tal, "Mesh Segmentation Using Feature Point and Core Extraction," *The Visual Computer*, pp. 649–658, 2005.

[7] H. Chui and A. Rangarajan, "A New Point Matching Algorithm for Non-rigid Registration," *Computer Vision and Image Understanding*, vol. 89, pp. 114–141, 2003.

[8] S. Umeyama, "An Eigendecomposition Approach to Weighted Graph Matching Problems," *PAMI*, vol. 10, pp. 695–703, 1988.

[9] G. Scott and L. Higgins, "An Algorithm for Associating the Features of Two Images," *Biological Sciences*, vol. 244, pp. 21–26, 1991.

[10] V. Kraevoy, A. Sheffer and C. Gotsman, "Matchmaker: Constructing Constrained Texture Maps," *ACM SIGGRAPH*, pp. 326–333, 2003.

[11] M. Carcassoni and E. Hancock, "Spectral Correspondence for Point Pattern Matching," *Pattern Recognition*, vol. 36, pp. 193–204, 2003.

[12] L. Shapiro and J. Brady, "Feature Based Correspondence: An Eigenvector Approach," *Image and Vision Computing*, vol. 10, pp. 283–288, 1992.

[13] S. Belongie, J. Malik and J. Puzicha, "Shape Matching and Object Recognition Using Shape Contexts," *PAMI*, vol. 24, pp. 509–523, 2002.

[14] T. H. Cormen, C. E. Leiserson, R. L. Rivest and C. Stein, "Introduction to Algorithms (2nd ed.)," *MIT Press and McGraw-Hill*, 2001.

[15] C. H. Papadimitriou, and K. Steiglitz, "Combinatorial Optimization: Algorithms and Complexity," *Prentice-Hall*, 1982.

[16] M. Alexa, "Recent Advances in Mesh Morphing," *Computer Graphics Forum*, vol. 21, pp. 173–196, 2002.

[17] N. Gelfand, N. Mitra, L. Guibas and H. Pottmann, "Robust Global Registration," *Symposium on Geometry Processing*, 2005.

[18] C. Gotsman, X. Gu and A. Sheffer, "Fundamentals of Spherical Parameterization for 3D Meshes," *ACM SIGGRAPH*, vol. 22, pp. 358–363, 2003.

[19] T. Zinber, J. Schmidt and H. Niemann, "A Refined ICP Algorithm for Robust 3D Correspondence Estimation," *Image Processing*, 2003.

[20] D. Vlasic, I. Baran, W. Matusik and J. Popovic, "Articulated Mesh Animation from Multi-view Silhouettes," *ACM SIGGRAPH*, vol. 27, pp. 97:1–97:9, 2008.

[21] G. Dewaele, F. Devernay, R. Horaud and F. Ferbes, "The Alignment between 3-D Data and Articulated Shapes with Bending Surfaces," *ECCV*, 2006.

[22] D. Anguelov, P. Srinivasan, H. Pang, D. Koller, S. Thrun and J. Davis, "The Correlated Correspondence Algorithm for Unsupervised Registration of Nonrigid Surfaces," *NIPS*, 2004.