Project Report - A Survey of Recent Advances in Shape Matching

Nihshanka Debroy

April 30, 2007

1 Introduction

Shape matching has been an active area of research for several years now, and it has applications to several disciplines (such as computer vision and molecular biology). In shape matching, the aim is to transform a shape and measure its resemblance to another shape, based on some measure of similarity. The properties of this measure depend on the specific matching problem at hand.

1.1 Types of Problems

In [1], an overview of techniques and measures in shape matching is provided. Given two patterns and a measure of similarity, the kinds of problems addressed by shape matching (as described in [1]) are as follows:

1. Computation problems: compute the similarity between two patterns.
2. Decision problems: given a threshold, decide if the similarity/dissimilarity is larger/smaller than the threshold.
3. Decision problem: given a threshold, decide if there is a transformation after which the dissimilarity between the transformed shape and the other shape is less than the threshold.
4. Optimization problem: find the transformation that minimizes the dissimilarity between the transformed shape and the other shape.
5. Approximate optimization problem: Often, the complexities of solving the above problems are extremely high. For such a case, an approximation algorithm finds a transformation that permits a dissimilarity between the two shapes that is within a constant factor from the minimum dissimilarity.

1.2 Applications

Solutions to the above problems have diverse applications, broadly categorized as follows in [1]:

- Shape retrieval: search for shapes in a large database that are similar to a query shape. Often, it is impractical to compute the similarity between the query shape and every shape in the database. In these cases, indexing structures are often used to exclude large sections of the database from comparison.
- Shape recognition and classification: determine if a given shape is sufficiently similar to another shape, or find the most similar shape from a set of shapes.
- Shape alignment and registration: transform one shape to find the best matching to a second shape.
- Shape approximation and simplification: create a shape that is less complex (with fewer vertices, triangles, etc.), but still similar to the original shape.

1.3 Similarity Measures

Similarity measures usually need to have certain specific properties for them to be useful in shape matching, although the properties vary with the application. Assume we have three shapes, X, Y and Z, and the distance between X and Y is represented as $d(X, Y)$. The following are common properties of similarity measures, as categorized in [1]:

1. Metric properties:
   - Non-negativity, which means $d(X, Y) \geq 0$.
   - Identity, $\Rightarrow d(X, X) = 0$.
   - Uniqueness, $d(X, Y) = 0, \Rightarrow X = Y$.
   - Strong Triangle Inequality, $d(X, Y) + d(X, Z) \geq d(Y, Z)$.
   - Relaxed Triangle Inequality, $c(d(X, Y) + d(Y, Z)) \geq d(X, Z)$, for $c \geq 1$.
   - Symmetry, $d(X, Y) = d(Y, X)$

   If a distance function obeys the identity, uniqueness and strong triangle inequality properties, it is referred to as a metric [1]. If the distance function only satisfies identity and the strong triangle inequality, it is called a semimetric.

2. Continuity properties: With respect to a similarity function, robustness is considered a form of continuity [1]. In the paper, continuity properties that allow the distance function to be robust "against the effects of discretization" are described (For example, robustness against minute affine distortions/perturbations is a desirable property.).

3. Invariance properties: A distance function d is considered invariant under a group of transformations, G, if $\forall g \in G$, $d(g(X), g(Y)) = d(X, Y)$.

Paper [1] then goes on to describe common measures of similarity, some of which are as follows:

- Discrete Metric: $d(X, Y) = 0$, if X equals Y. Otherwise, $d(X, Y) = 1$. A disadvantage of this metric is that if shape X is even minutely distorted, to form shape X', the discrete distance $d(X, X')$ will be 1. If one uses the discrete metric, computing the smallest $d(X, Y)$ over all transformations in a set G is equivalent to looking for a transformation $g \in G$ such that $g(X) = Y$. This is called the "exact congruence matching."

- Minkowski Distance (L_p Distance): For two points x, y in $\mathbb{R}^k$, the L_p distance is defined as: $L_p(x, y) = (\Sigma_{i=0}^{k} |x_i - y_i|^p)^{\frac{1}{p}}$.

- Bottleneck Distance: Assume A and B are two point-sets of size n. Let $d(a, b)$ be the distance between two points a and b. The bottleneck distance, $F(A, B)$, is the "minimum over all 1-1 correspondences f between A and B of the maximum distance $d(a, f(a))$."

- Hausdorff Distance: Let P and Q be two sets of points in $\mathbb{R}^d$. The directed Hausdorff distance from P to Q, denoted by $h(P, Q)$, is:

$$\max_{p \in P} \min_{q \in Q} ||p - q|| \tag{1}$$

  The Hausdorff distance between P and Q, denoted by $H(P, Q)$, is:

$$\max\{h(P, Q), h(Q, P)\}. \tag{2}$$

  Intuitively, the function $h(P, Q)$ finds the point $p \in P$ that is farthest from any point in Q and measures the distance from p to its nearest neighbor in Q. However, as is apparent from its definition, the Hausdorff distance is extremely sensitive to noise (outliers). A measure that seems to be less sensitive to noise is the partial Hausdorff distance, defined as:

$$H_k(P, Q) = \max\{h_k(P, Q), h_k(Q, P)\} \tag{3}$$

  where $h_k(P, Q)$ is the "k-th value in increasing order of the distance" from a point in P to Q. Thus, $h_k(P, Q) = k^{th}_{p \in P} \min_{q \in Q} d(p, q)$. However, the

partial Hausdorff distance is not a metric as it does not satisfy the triangle inequality property mentioned earlier [1].

- Fréchet distance: This is typically used to measure the similarity between curves. The Fréchet distance between two curves is defined as follows:

$$Fr(P,Q) = \inf_{\alpha,\beta} \max_{t\in[0,1]} ||P(\alpha(t)) - Q(\beta(t))|| \tag{4}$$

where $P, Q : [0,1] \rightarrow \mathbb{R}^2$ are parametrizations of the two curves and $\alpha, \beta : [0,1] \rightarrow [0,1]$ range over all continuous and monotone increasing functions. A couple of variations of the Fréchet distance are mentioned in [1]. Firstly, the monotonicity condition of the parametrization could be dropped. Partial matching could also be considered, where the goal is to find the section of one curve to which the other has the smallest Fréchet distance.

In general, transformations can be classified as rigid of affine. In a rigid transformation, distances and orientations are preserved. There are two types of rigid transformations in $\mathbb{R}^3$ - translations, which preserve "difference vectors" and rotations, that preserve the origin. Examples of affine transformations are scaling and shear.

1.4 Algorithms

Paper [1] also describes a few general algorithms for shape matching, such as geometric hashing and generalized Hough transforms. In geometric hashing, the goal is to decide if there is a "transformed subset of the query point set that matches a subset of a target point set" [1,2]. The transformation space is represented as a six-dimensional table, and the best transformation is determined by means of a voting scheme. Given two point-sets A and B, for each triplet of points in A and in B, the transformation between the triplets is calculated and a vote recorded in the corresponding cell of the table. The entry with maximum votes defines the best transformation. To circumvent the problem of searching through all possible transformations, geometric hashing saves off-line all "candidate transformations" in a hash table [1,2].

Similarly, the generalized Hough transform is also a voting scheme [3]. In this case, affine transformations are represented by six coefficients, and the transformation space is represented as a table of six dimensions. For each triplet of points in one set, the transformation to each triplet in the other set is computed, and a vote is counted in the respective entry of the table. The entry receiving the most votes is the optimal transformation.

2 Distance Functions

In the paper on registration using distance functions, the main contribution is the use of a robust shape representation (using distance functions) for global-to-local alignment [4]. In other words, their algorithm allows for global registration (for the entire shape) as well as local deformations (in small neighborhoods). The shapes are represented using "signed Euclidean distance transforms" as the feature space. The sum of squared differences between the source shape and the target shape is considered the optimization criterion.

The following is a description of the shape representation used in [4]. Let $\phi : \Omega \rightarrow R^+$ be a Lipschitz function, referring to a distance transform representation for a given shape S. Let R_S denote the convex hull of the shape S. They consider the following shape representation:

$$\phi_S(x,y) = \begin{cases} 0 & , (x,y) \in S \\ +ED((x,y),S) > 0 & , (x,y) \in R_S \\ -ED((x,y),S) < 0 & , (x,y) \in [\Omega - R_S] \end{cases}$$

where $ED((x,y),S)$ is the minimum Euclidean distance between point (x,y) and the shape S. In the paper, a proof was given that this representation is invariant to translation and rotation.

For global registration between two 2-D shapes S and D (where D is the shape that undergoes transformation), involving a rotation angle θ, a translation vector $T = (T_x, T_y)$ and a scale factor s, the basic form of the objective function that is used to recover the optimal registration parameters is:

$$E(s,\theta,T) = \int\int_{\Omega} (s\Phi_D(x,y) - \Phi_S(A(x,y)))^2 d\Omega \tag{5}$$

where

$$A(x,y) = s \begin{pmatrix} cos\theta & sin\theta \\ -sin\theta & cos\theta \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} T_x \\ T_y \end{pmatrix}$$

A gradient descent approach was used to recover the optimal registration parameters. However, it is also desirable to allow for local deformations. Hence, the objective function was modified to also include local deformations. Based on a parameter, the contributions of the two aspects (global and local deformations) are balanced out. However, local deformations appear to significantly increase the complexity of the algorithm. Hence, the search space in case of local deformations is constrained to the vicinity of the source shape. However, this is still not a complete model, as real objects undergoing transformations have certain constraints. For example, a real object is composed of "connected elements," and so, local deformations must show similar transformations - this is referred to as the "smoothness" condition. The objective function is, therefore, further modified to include for smoothness. Using gradient descent, partial differential equations are solved for the estimation of the optimal registration paramters.

The paper also discussed possible approaches to the registration problem when the target is not a shape, but a shape model with "local degrees of variability" [4]. The authors found encouraging results when their algorithm was applied to several empirical tests (that included cases of occlusion, local deformations and random global transformations applied to the source shape). They found that when strong local deformations were not present, the algorithm converged to the same global minimum each time. They noted that since their proposed objective function was smooth and continuous, the gradient descent method could converge to the global minimum. However, their approach does not handle symmetric shapes too well. This is because symmetries create "similar distance transform representations," resulting in the convergence to local minima. The computational cost of their approach depends on the initial conditions, the sizes of the source and target shapes and the extent of local deformations. They note that the use of classical numerical methods hampers the complexity, as very small time-steps are required for stability.

3 Protein Docking

3.1 Shape Complementarity

Understanding how proteins interact with each other is essential to understanding cellular processes. The increase in protein structural information now enables the study of protein-protein interactions and molecular docking. It is fairly agreed upon today that a key requirement for two proteins to form a complex is good surface complementarity. Paper [5] discusses the first algorithm that successfully predicts the re-docking of known protein-protein complexes with no false positives. The algorithm is based only on shape complemetarity, implying that shape matching could be sufficient for docking prediction. In contrast, most previous approaches reduced the complexity of the protein surface and also supplemented algorithms that used geometry with information on properties such as electrostatics, hydrogen bonding and hydrophobicity [5].

In order to reduce the complexity to six degrees of freedom (three for rotation and three for translation), the authors treated the proteins as rigid bodies. Their approach involves searching for "potential binding patches or pronounced geometric shapes" and using a scoring function that evaluates complementarity in geometry with/without calculating any free energies of binding [5]. The scoring function counts the number of non-overlapping pairs of spheres on each protein that are at most a distance of 1.5 Å from each other. They also allow a maximum of 5 collisions/bumps between spheres. Also, apart from using $RMSD$ (root mean-squared distance) as a criterion for evaluating docking solutions, they also used something called $RMSD^*$, which reflects the deviation of atoms in the vicinity of the binding pocket. This makes sense because otherwise, correct docking solutions that only have a slightly incorrect angle could lead to high, but misleading, $RMSD$ values due to the effect of residues far way from the docking site [5].

In order to express rotations, quaternions were used in [5]. Specifically, a point $x = (x_1, x_2, x_3)$ was represented as a 4-vector $\mathbf{x} = (0, x_1, x_2, x_3)$. A rotation was performed by multiplying $\mathbf{x}$ with a unit quaternion $\mathbf{q} = (q_0, q_1, q_2, q_3)$ from the left and the conjugate $\mathbf{q}^* = (q_0, -q_1, -q_2, -q_3)$ from the right, giving a 4-vector $\mathbf{y} = (y_0, y_1, y_2, y_3)$ corresponding to a point $y = (y_1, y_2, y_3)$ in $\mathbb{R}^3$. The geometric interpretation of the map that moves x to y is that it is the rotation by the angle $\varphi = 2\ arccos\ q_0$ about the line spanned by the vector (q_1, q_2, q_3) in $\mathbb{R}^3$ [5].

Initially, the two proteins to be docked were translated so that both the centroids lay on the origin of $\mathbb{R}^3$. Two cubes were then defined, each one containing one of the proteins, such that any translation that led to non-overlapping cubes resulted in "protein placement with zero score" and, so, did not have to be considered further in the algorithm [5]. Rotations were treated in an analagous manner.

In re-docking, given two proteins as a complex, the problem is to form the complex from the the individual proteins. To avoid using knowledge of the given position, a random rigid transformation was applied to one of the proteins, and that conformation was used as input to the docking algorithm. Given a rotation ∂ and a translation τ, the number of collisions/bumps is the number of pairs of van der Waals spheres (one sphere from each protein) that intersect. The score is the number of non-intersecting pairs of spheres at distance at most some threshold 2λ from each other. Mathematically,

$$bump(\partial, \tau) = card\{(i, j) \mid ||a_i - b_j|| < \alpha_i + \beta_j\} \tag{6}$$

$$score(\partial, \tau) = card\{(i, j) \mid \alpha_i + \beta_j \leq ||a_i - b_j|| \leq \alpha_i + \beta_j + 2\lambda\} \tag{7}$$

where α_i and β_j are the radii of the van der Waals spheres with centers a_i and b_j respectively. For each rigid motion, the 'bump' and 'score' can be computed in time $O(mn)$, where m and n are the numbers of atoms in the two proteins. The process was sped up by computing the bump and score values for all translations corresponding to a single rotation at once.

They were able to successfully re-dock all the complexes in their data set and did not find any high scoring solution that gave an incorrectly docked conformation. The running time of the algorithm is dominated by the number of rigid motions that are explored. The three-dimensional space of translations is covered with a grid of step-size 2∂, and the three-dimensional space of rotations is covered with a grid of step-size 2ϵ. Hence, the total number of rigid motions generated is approximately $\frac{1}{\partial^3 \epsilon^3}$.

There have been prior approaches to predicting protein docking based on shape complementarity. For example, the Connolly function is a way of ascribing to each point on a protein's surface a number that represents the extent of convexity or concavity near that point [6]. When this function was first introduced, the number was calculated by constructing a sphere centered at the point and determining how much of the sphere was in the interior of the protein's surface (measured as

a solid angle). Subsequent improvements proposed the use of additional metrics such as local volume to determine shape, instead of just the solid angle. Several of these approaches were used to identify knobs and depressions on protein surfaces, with the goal of predicting complexes between proteins with complementary structural features [6]. However, the complexity of the molecular docking involved and the lack of satisfactory results led most researchers to conclude that the Connolly function is not the best approach to the molecular docking problem.

3.2 Surface Matching and Supervised Machine Learning

In a very recent paper ([7]) a docking procedure has been described, that achieves efficient sampling of conformations by matching surface normal vectors, "fast filtering for shape complementarity by $RMSD$ clustering" and the scoring of docked conformations using supervised learning. A classifier called the Random Forest is used, with features such as contacting residue pair frequencies, evolutionary conservation and shape complementarity [7].

Initially, a geometrical surface matching approach was used to create protein docking conformations with good shape complementarity. The conformations were then filtered to select the near-native structures. Each protein's surface was defined using surface points, spaced about 1.5 Å apart, and a set of normal vectors. The initial docking was carried out by aligning each point on one protein surface with each point on the other, anti-aligning the surface normal vectors on each protein's surface and considering rotations about the axis common to both proteins. A major advantage of this method, compared to previous methods, is that it uses a larger set of evenly spaced points on the protein surface, thereby providing a thorough sampling of possible conformations. Having generated the initial docking configurations, some of them that possess large "steric clashes" or insufficient shape complementarity are removed carefully, with the help of a filtering function. The algorithm also takes advantage of any symmetry in the protein structures to reduce the space of possible conformations and improve prediction accuracy [7].

It is expected that several similar conformations will be generated by the docking procedure, especially in cases of significant shape complementarity. All docking configurations are then clustered by $RMSD$ (that is, the $RMSD$ to the C_α), retaining only one conformation per cluster to remove redundancy. Most hierarchical clustering algorithms take time $O(n^2 \log n)$, where n is the number of points (here conformations) to be clustered. The algorithm implemented in [7] takes time $O(n \log n)$ - it goes through the list of conformations, outputting a new conformation if the $RMSD$ between it and any previously accepted conformation is greater than a threshold (say, this is $RMSD_{max}$) [7]. The docking conformations are determined by six degrees of freedom (three for translation and three for rotation). The paper suggests that the clustering could be sped up first performing an orthogonal range search (using a k-d tree), with a box of 6 dimensions that is large enough to retrieve conformations that had been output previously and had $RMSD$ less than $RMSD_{max}$, and then calculating the $RMSD$ only for the conformations

that were retrieved [7]. They derived the smallest dimensions of the 6-dimensional bounding box for a given value of $RMSD_{max}$.

Further filtering out of the docking configurations is carried out by casting into a supervised machine learning, binary-classification (near-native or non-native) problem [7]. Near-native conformations are said to have $> 50\%$ native residue contacts. The Random Forest (RF) classifier was the method chosen as it can be trained on data that is not homogeneous [7], while maintaining high prediction accuracy. Moreover, the RF classifier appears to have a fast learning time, few model parameters, resistance to overfitting and no need for the normalization of data [7]. The RF classifier also permits the easy estimation of the importance of each variable to the prediction. It works by creating a set of several independent classification trees, or a forest, from random samples of the training data. The prediction is made depending on the fraction of trees predicting each class [7]. If each class is almost equally likely to occur, the predicted class is the one assigned by most of the decision trees in the forest. For cases where one class is a lot less likely to occur (like the near-native state), heuristic ideas were used to balance out the error rates of the two classes. One example is "stratified sampling," where an equal number of near-native and non-native conformations were chosen in the data set [7].

The input data to the RF classifier consisted of features such as shape complementarity, contacting residue types at the interface, solvent accessible surface areas and evolutionary conservation, for each docking conformation. An interesting note was that immune system related complexes (such as antibodies and MHC complexes) were left out of the data sets, as they tend to have a lot of mutations, resulting in very low evolutionary conservation values [7]. In my research, I have been studying MHC-peptide binding surfaces. Therefore, if I were to apply this algorithm for docking predictions, I would probably discard the evolutionary conservation feature altogether. Previous studies have suggested that near-native conformations tend to lie in the largest clusters. Hence, the researchers also included the cluster size as a feature. However, this did not lead to any signficant improvement in prediction accuracy. The RF classifier was able to distinguish the correct docking conformations from the incorrect ones and to successfully generate near-native conformations among the highest-ranked docking predictions. Moreover, it also assigns a score that could be used to rank conformations, based on the probability of their being correct [7].

In order to compute the importance of each feature, the average decrease in the prediction accuracy was calculated after randomizing the values of the feature in question. It was found that the most significant features are evolutionary conservation, shape complementarity score, under-represented and over-represented residue types in interface regions, and contacting residue pairs. A suggestion made at the end of the paper was to make the shape complementarity function cutoff less strict and include the protein size as a feature in the training data. This would allow the RF classifier to select conformations with favorable shape complementarity, instead of a hard cutoff earlier in the algorithm.

4 Partial Shape Matching

Apart from matching shapes globally, it is also useful to be able to match approximately similar surface regions. This is referred to as partial shape matching. The parts that are matched do not have to be predefined. They could also be in any orientation or scale. Partial matching is harder than global matching, as the subparts need to be defined before measuring similarities. A recent paper describes the concept of local shape descriptors to represent the geometry of local regions on a triangulated shape/surface [8]. These descriptors are independent of the triangulation, but allow the matching of surfaces with different triangulations. The paper also defines what are called salient geometric features, as high-level features of non-trivial local shapes [8]. From their analysis, it appears that relatively few salient geometric features are sufficient to characterize the surface. The salient geometric features are stored in a geometric hash table and can be queried quickly for partial matching purposes.

The local surface descriptors (LSDs) represent local regions on the surface and provide a way to to measure the similarity between regions on triangulated surfaces. An LSD is defined as a point p on a surface and its corresponding "quadric patch" that approximate the surface in the vicinity of p. Smooth regions tend to get represented by few descriptors, as each quadric patch is able to locally approximate a large region. As the number of local surface descriptors is usually much smaller than the number of vertices, the complexity of the surface representation is reduced [8]. The number of descriptors is often still too large for efficient matching. Hence, salient geometric features (SGFs), defined as clusters of descriptors that describe local regions on the surface, are also used as a basis for measuring similarity that is not global [8]. The determination of what parts classify as SGFs is based on the size of the part relative to the whole object and the part's surface curvature. The SGFs enable complicated tasks such as identifying similar parts in a single surface (Self-similarity) or across several surfaces. This partial shape matching method supports rigid transformations and uniform scaling (called similarity transformations). Unlike in previous work, the surface descriptors described in [8] are independent of scale and efficiently represent mesh regions arbitrarily large. However, the effectiveness of the LSDs and the SGFs depends heavily on the quality of the mesh and the curvature analyis [8].

For an implicit surface $F(x, y, z) = 0$, a quadric with nine coefficients is used. The fitting of the surface is defined as a least squares minimization problem, enabling the method to work on any point cloud in $\mathbb{R}^3$, irrespective of orientation. For each vertex v of the mesh, an implicit surface is fitted. Assume the projection of v onto the implicit surface is v'. The curvature tensor and curvature derivatives of v are then estimated by those calculated for v'. For increased robustness, a number of neighborhood sizes were tried around each vertex, and the quadric with median curvature was selected [8]. To determine the set of local surface descriptors, the mesh vertices were sorted based on their Gaussian curvatures. For each vertex, the largest possible "quadric path" that approximated its neighborhood was grown. All

vertices included in the local path were removed from the sorted list, and the iterative procedure then continued on to define the next patch [8]. This approach was greedy since at each step, the largest possible region around the selected vertex that met an error threshold was defined. The quality of the quadric fitting was measured as the sum of squared errors of distances between the fitted points and the fitting surface. After each patch was defined, a point at the center of mass was selected and associated with the highest curvature in the patch. The authors noted that the curvature maps and the distribution of local surface descriptors were similar over different triangulations.

The SGFs were computed by clustering sets of descriptors with high curvatures (relative to surroundings) and high variance of curvature values. Each SGF could be defined in terms of the radius of the sphere bounding it. The saliency grade S of a cluster F, consisting of $d \in F$ descriptors, is defined as follows:

$$S = \sum_{d \in F} W_1 Area(d) Curv(d)^3 + W_2 N(F) Var(F) \tag{8}$$

where $Area(d)$ is the area of the patch corresponding to d relative to the sphere size, $Curv(d)$ is the curvature associated with d, $N(F)$ is the number of local minimum(s)/maximum(s) curvatures in the cluster and $Var(F)$ is the curvature variance in the cluster. The weights W_1 and W_2 were set to 0.5. The term $Area(d)Curv(d)^3$ represents the importance of the region, as it includes size and curvature information. For each descriptor, a cluster was grown by adding descriptors that maximized the saliency grades at that point [8]. The clusters with the top saliency grades (like, the top 10%) define the salient geometric features.

The chosen SGFs were extracted offline and stored in a "rotation-and-scale-invariant database.' Each SGF was associated with a vector index/signature (based on the same factors that determine the saliency grade) and inserted into a geometric hash table [8]. Geometric hashing allows the speedy retrieval of partial matches from a database and the determination of the explicit transformation that matches the shapes [2]. The implementation in [8] used SGFs as the object and the local surface descriptors as points, to define the triplets and votes needed for the hash table. Typically, each SGF was found to contain about 20-30 LSDs. The combination of the vector indexing and geometric hashing (which is scale-independent) helped identify partial matches between the queries and salient features stored in the hash table.

The following three applications of partial matching were described:

1. Self-similarity: The goal here is to identify salient features that occur several times on the same surface, although their sizes could vary. Copies of all the salient features, with rotations and uniform scale transformations applied to them, are searched for. Initially, the salient features of the surface are sorted based on saliency grades, and for the top k ranked salient features, a partial matching search is carried out [8].

2. Shape Alignment: The partial matching of salient geometric features makes it possible to align, where they match, two models with different global shapes. An intuitive heuristic for alignment, used in [8], is to find the transformation that satisfies most of the salient features.

3. Partial Shape Retrieval: This enables the retrieval of subparts of a larger shape that are similar to a query, even if the larger shape is different from the query.

The LSDs are a set of disconnected points. Hence, as a topic for future work, the authors suggest the extension of partial matching to non-manifold objects and point-sets as well.

5 Other Approaches

A couple of other papers that I surveyed talked about 'Skeleton Based Shape Matching' and protein structure similarity [9,10]. In skeleton based matching, the geometric and topological properties of the shape (such as local shape descriptors) are saved in the nodes/joints of a skeletal graph [9]. The skeletal graph is a $1-D$ approximation of what is called the "Medial Surface" [9]. The information saved could include the mean, radius, the degrees of freedom about the joint or the importance of specific joint/node. Along with the graph, a "topological signature vector" of the graph is stored in an indexed database [9]. This vector is an index of low dimensions that captures the global and local properties of the shape. The database can be queried to find the best match for a given shape. A subsequent graph matching step on the top responses from the query on the indexed database allows the comparison of multiple skeletal graphs. An advantage of this approach is its ability to do partial matching.

In another paper, the motivation was to design an algorithm to compare the similarity of protein structures using dynamic programming [10]. A complication when working with structures instead of strings is "orientation." Hence, in addition to finding similar subsequences, the optimal orientations for the two proteins also need to be computed. The design of an algorithm to compare structures involves discrete optimization (finding corresponding subsequences) as well as continuous optimization (finding the optimal orientation) [10]. Given the solution to one optimization problem, the other is easy to solve. So the challenge is to carry out the optimizations simultaneously. The approach described in [10] produces a globally optimal solution. It builds upon the dynamic programming local-similarity algorithm for strings. However, instead of comparing sequences of characters, sequences of vectors (vectors of the protein backbone) are compared [10]. The algorithm described in [10] utilizes "extreme points," which are the vertices of a convex hull, in 10-dimensional space to solve the protein structure matching problem in $\mathbb{R}^3$ [10]. The running time of the algorithm is $O(mnp^2)$, where m and n

are the lengths of the two sequences of vectors and p is the maximum number of points held in each dynamic programming node (details available in [10]).

References

[1] R. C. Veltkamp, *International Conference on Shape Modeling and Applications*, pp. 188-197, 2001.
[2] Y. Lamdan, H.J. Wolfson, *2nd International Conference on Computer Vision*, pp. 238-249, 1988.
[3] D. H. Ballard, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 111-122, 1981.
[4] N. Paragios, M. Rousson, V. Ramesh, *Computer Vision and Image Understanding*, pp. 142-165, 2003.
[5] S. Bespamyatnikh, V. Choi, H. Edelsbrunner, J. Rudolph, Accurate protein docking by shape complementarity alone, Manuscript, Duke Univ., Durham, NC, 2004.
[6] M. L. Connolly, *Biopolymers,*, vol. 25, pp. 1229-1247, 1986.
[7] A. J. Bordner, A. A. Gorin, *Proteins:Structure, Function, and Bioinformatics*, 2007.
[8] R. Gal, D. Cohen-Or, *ACM Transactions on Graphics*, pp. 130-150, 2006.
[9] H. Sundar, D. Silver, N. Gagvani, S. Dickinson, *Proceedings of the Shape Modeling International*, 2003.
[10] L. Paul Chew, *Symposium of Computational Geometry*, 2006.