Graphical Templates for Model Registration

Yali Amit and Augustine Kong

Abstract—A new method of model registration is proposed using graphical templates. A graph of landmarks is chosen in the template image. All possible candidates for these landmarks are found in the data image using local operators. A dynamic programming algorithm on decomposable subgraphs of the template graph finds the optimal match to a subset of the candidate points in *polynomial time*. This combination of local operators to describe points of interest/landmarks and a graph to describe their geometric orientation in the plane, yields fast and precise matches of the model to the data, with no initialization required.

Index Terms—Graphical templates, decomposable graphs, model registration, dynamic programming, image matching.

1 Introduction

In recent years there has been a growing interest in deformable models for the analysis of images of families of objects which manifest a continuous type of variability, in particular in the context of medical and biological imaging. The ultimate goal is to describe each image in the family in terms of the *deformation* of a template or model which yields the best match to the image. This description can then be used in a variety of ways. First it provides an automatic means of identifying the various components of the object in the image, and in some cases of segmenting the image. Secondly it provides a means of studying the variability in the family, classifying subgroups and identifying abnormalities. Thirdly it can provide a means for coding or data compression. The most extensively studied models come under the general title of elastic matching.

1.1 Elastic Matching Versus Landmark Matching

Grenander [13] introduced the idea of elastic deformations of one and two dimensional templates. These ideas were implemented in a Bayesian framework in [2] for hand x-rays and in [21] for MRI images of the brain. Similar ideas were proposed in [3] for MRI brain images, using techniques developed in the optical flow and image sequence analysis literature, [17], [18], [22], [27]. A comparison of these methods both from the point of view of the optimization problem being posed and the numerical techniques can be found in [1].

There are several limitations to these elastic deformations. First the template is simply one image from the family, no modeling is involved at this level. Second the matching of the deformed template to the data image is pixel intensity driven. In other words the matching criterion is minimizing the mean square over all pixels of the difference between the intensity of the deformed template and that of the data image. This does not ensure that specific points of interest or landmarks be matched with great precision. Third the deformations being used are generic or non-parametric in nature and do not depend on the specific family of objects. Fourth, because of the inherent non-linearity of the problem, and the fact that the deformations are high dimensional, the computational tools for calculating the match must use relaxation techniques which run the risk of converging to a local minimum which corresponds to a poor match.

These issues point to three categories which must be addressed *simultaneously* and in an integrated manner in any approach to the template matching problem. The data term which drives the matching. The model and its variability. The computational tools and their limitations.

The main goal of this paper is to present a new *computationally efficient* model registration method consisting of a graphical template of landmark points which describes their planar arrangement, together with robust local operators which identify candidates for the various landmarks in the data image. This is essentially a new approach to the general program outlined in [15]. Namely *image features* are defined and then correspondence is obtained through a *consistency model*, which is optimized through an efficient algorithm consisting of dynamic programming on decomposable graphs.

Thus more emphasis is put on the precise matching of landmarks, see [9], local features, or points of interest [15]. These landmarks are both important for understanding and analyzing the image and can be identified using various local operators which employ more information than individual pixel intensity. Moreover modeling the variability in terms of the relative locations of the landmarks yields a more specific and lower dimensional description of the variability within a certain family of objects. The same point is made in [10] with respect to the 1D elastic contour models.

These ideas can also be found in the 3D object recognition, stereo and robot vision literature, in the context of the well

• *The authors are with the Department of Statistics, University of Chicago, Chicago, IL 60637. E-mail: amit@galton.uchicago.edu.*

Manuscript received Aug. 25, 1994; revised Nov. 6, 1995. Recommended for acceptance by L. Shapiro.
For information on obtaining reprints of this article, please send e-mail to: transactions@computer.org, and reference IEEECS Log Number P95171.

known correspondence problem. Another related problem is stellar constelation matching where the graphs provided by the Delaunay triangulation are used to impose hard constraints on the arrangement of the landmarks, [16]. In these applications the variations in the locations of the landmarks are subject to very stringent constraints due to the inherent rigidity of the objects. Here we will be dealing with non-rigid objects which exhibit a larger variability.

1.2 Local Operators and Graph Matching

The local operators used here involve only rank relations between intensities in the neighborhood of a pixel. These turn out to be very robust, and can be used both to describe local boundary shape and local topographies such as maxima, minima, saddle points etc. The algorithm however can be implemented with any choice of local operators. Usually the same local operator will be used for several landmarks, and many more candidates will be found in the data image, than landmarks associated with that operator in the model, see Fig. 7. To find the correct match of the landmarks in the model to the candidate pixels in the image it is necessary to introduce constraints on the relative locations of the landmarks in the plane. Note that the variability of these relative locations *can not* be expressed as a uniform affine transformation on all points.

The constraints are described through a collection of triangles between triples of landmarks. A cost function is associated with each such triangle which penalizes its shape deviation from some mean triangle. It is important to note that these constraints need not be *local*, indeed they could involve far apart landmarks. This is in contrast to the implicit constraints used in the elasticity models which essentially penalize large changes in the local lattice elements.

The total cost function is the sum of the cost functions over all the triangles in the model. The collection of triangles can also be expressed as a coloured graph, where the nodes are the landmarks, the colour or type of a node is given by the type of local operator used to find its candidates. The edges in the graph exist between each two landmarks which belong to a triangle. This graph is called the template graph. Note that in this context the deformable template is not an image but a graph model.

Finding the optimal match then reduces to an *inexact consistent-labeling problem* [15]. The inexact consistent labeling problem is generically exponential in complexity, however if the template graph is chosen to be decomposable, it is possible to find the optimal match in *polynomial time* using dynamic programming on the graph, see, for example, [26]. Decomposability in the present context means that there exists an order in which the triangles of the graph can be successively eliminated, such that each triangle in its turn has a free vertex contained in no other triangle. When the free vertex and the two edges emanating from it are removed, one of the vertices of the next triangle in the order is freed and so on (see Fig. 1). Dynamic programming has been used in image analysis in a variety of contexts such as road tracking [4], stereo [23], and artery tracking [24]. All these settings are one dimensional in nature, and the constraints enforced by the underlying graph are all local. To our knowledge this is the first attempt to incorporate the efficient computational tool of dynamic programming in an inherently two dimensional and non-local imaging problem.

Decomposability imposes certain limitations on the graph, and may limit the geometric and topological information it contains. This has been addressed by iterating between two graphs as described in Section 2. Another limitation of this method is that the graph is constructed by hand as opposed to the automatic generation of a Delaunay graph for example as in [16].

1.3 Interpolation and Initialization of Elastic Deformation Algorithms

Once the matching of the template graph to a subset of the landmark candidates in the data image is completed, we are in the same situation described in Bookstein's work on image warping. In the latter case both the landmarks in the template are chosen manually, as well as their matches in the data. The following step is calculating a planar mapping which interpolates between the existing point matches. We refer the reader to [20] and [8] for further details.

The planar mapping can now serve as an initial point to an elastic deformation algorithm for image matching. We employ a spectral method described in [1] where the parameters of the deformation are the coefficients of its expansion in some orthonormal basis (Fourier or wavelet). The cost function in that approach is non-linear and the solution corresponds to a local minimum obtained through gradient descent where the initial point was always taken to be the identity map (no deformation). This method worked well in several cases and was able to identify large warpings. However since the deformations are not dedicated to the specific image families and in view of the non-linearity of the functional, mistakes are bound to occur as illustrated in Fig. 9. This image represents the final deformation of the template, obtained from the elastic deformation algorithm, subtracted from the target image. The fingers got confused.

Now that an interpolated match of the landmarks is available, it is taken as the initial point for the gradient descent. The large scale global matching has already been obtained through the matching of the landmarks. The image warping should now serve for small scale corrections and adjustments, see Figs. 10 and 11.

The strength of this algorithm is its computational efficiency and the absence of any need to initialize the matching algorithm or the optimization procedure. This is one of the drawbacks of many *relaxation* techniques. If the template or model is not initially set nearby the correct location the relaxation algorithm may end up in completely erroneous solutions. Indeed the match provided by this method can serve to initialize deformable template methods using relaxation techniques, such as [28] and [10].

To demonstrate the power of combining decomposable graphs of triangles with additive shape cost functions, together with dynamic programming, to match a model to the data, we start out in the extreme case where only *one* local operator is used, namely a local maximum operator. This is the hardest case because no assistance is provided by having a variety of colours in the graph. On the other hand since both the cost function and the local operator are

scale, rotation and translation invariant, the entire algorithm will be invariant to such transformations. The data we use are 128 by 128 low resolution digitizations of x-ray images of hands.

The components of the implementation together with the details of the graph matching using dynamic programming on appropriate subgraphs of the template graph are described in Section 2. In Section 3, we describe the experiments and show an example where this algorithm was implemented on high resolution x-ray images using a variety of local operators, thus obtaining very precise matchings of various anatomical components. We also present an example of the use of the graph match to initialize the image warping algorithm.

2 Local Operators, Decomposable Graphs, and Dynamic Programming

2.1 The Local Operators

The local operators were designed to be very robust to changes in pixel intensities and noise. A certain size neighborhood is chosen, say $m \times n$. Two $m \times n$ arrays L_1 of 1-s and 0-s, and L_{-1} of -1-s and 0-s are chosen, which characterize the local operator. The supports (sets of pixels for which the value is non-zero) of the two arrays should be disjoint but their union need not be the entire $m \times n$ neighborhood. Let n_1 and n_{-1} be the number of non-zero elements in the two arrays, respectively.

The *sign* of the difference between the intensity at a given pixel i, j, and the intensity at each of the pixels in its $m \times n$ neighborhood is calculated, to yield an array $A^{(i,j)}$ of 1-s and -1-s, at pixel i, j. The next step is calculating

$$s_1 = \frac{L_1 \cdot A^{(i,j)}}{n_1}, \text{and } s_{-1} = \frac{L_{-1} \cdot A^{(i,j)}}{n_{-1}},$$

where '$\cdot$' indicates inner products of the two arrays. If s_1 and s_{-1} are larger than some tolerances, a candidate for operator (L_1, L_{-1}) has been found at pixel (i, j).

The simplest example is L_1 all 1-s, and L_{-1} all 0-s. This corresponds to a local maximum operator. The reverse corresponds to a local minimum operator. Other configurations yield crude descriptors of the topography of the pixel intensity in the neighborhood of a landmark. The pixels corresponding to a certain local operator typically occur in clusters. These were identified and the average location calculated. In Section 5 a description of the four local operators used for the high resolution x-rays is given, together with the tolerances.

For simplicity the graph matching procedure is described assuming only one local operator.

2.2 The Graph

To create the template graph together with the "mean" for every triangle in the graph, the local maxima of an arbitrary image, henceforth called the template image, were displayed and those which appeared the most descriptive of the anatomy of the image were chosen to be the landmarks, namely the vertices of the template graph. These points were either at the joints of the fingers or at certain locations in the palm where there is a high bone concentration. The edges of the graph were then introduced manually by connecting vertices which seem to have an important geometric relation with each other.

The problem now is to determine which of the many local maxima found in the *data* image are the true landmarks and how to correctly match them to the corresponding landmarks of the template.

The graph which we call the *template graph*, is constructed so that all its cliques are triangles. Cliques being maximal subsets of the vertices, all pairs of which are neighbours in the graph, see Fig. 1. These triangles are assigned a certain cost function. A match of the template graph to some subset of the local maxima found in the data image, will have a cost associated with it. This cost penalizes deviations from the *shape* of the original triangles of the graph of landmarks in the template image. Here shape refers to the angles of the triangles, not to scale or rotation. Hopefully the choice of triangles and cost functions are such that the optimal match to the data will yield a match of the anatomical locations which the landmarks represent.

Finding the optimal match of the template graph to a subset of the local maxima extracted from the data image is a very difficult problem. It is clearly impossible to evaluate all possible combinations, this will quickly lead to a computational explosion, even for relatively small collections of data points and small graphs.

The graph is chosen so that it is *decomposable*. In the framework described here, where all cliques contain exactly three elements, this property can be described as follows. There exists an order of successive elimination of the cliques, such that in each clique, in its turn, there exists a free vertex, contained in no other clique, which can be eliminated, so that the remaining subgraph is again a collection of cliques of three. When a free vertex is eliminated the next clique in the ordering will again have a free vertex to eliminate, and so on until the last clique. Figuratively speaking the graph can be *folded* away triangle by triangle, see Fig. 1. These concepts will be defined formally in the next section. For decomposable graphs the optimization can be translated into a dynamic programming algorithm and therefore done in *polynomial time*.

This choice of graph, which is motivated solely by computational considerations, necessarily leads to the loss of a significant amount of topological information. Many closed loops, which severely constrain the possible configurations of maxima are eliminated. This loss in constraints is offset by iterating between two or more graphs, each of which is decomposable and which have at least two vertices in common, see Figs. 2c and 2d. A subset of common vertices of the two graphs is chosen. An optimal match is found using the first graph. When the optimization procedure begins for the second subgraph the match for this subset of data points is fixed as the match provided by the first subgraph. Now the optimal match is found for the remaining vertices of the second subgraph. The match found through this iteration procedure is not guaranteed to be the global optimum, however it appears that a good choice of subgraphs yields very good matches which should be very close to the optimum. In the following section we describe

the technical details of the matching algorithm for *decomposable graphs* through dynamic programming.

2.3 Decomposable Graphs

Let $x_1, \ldots, x_n$ be n landmarks identified on the template. Let $V = \{v_1, \ldots, v_m\}$ be the set of candidate landmarks in the data image with $m \geq n$. Any sequence $(i_1, \ldots, i_n)$ satisfying $i_k \in \{1, \ldots, m\}$ for all k denotes a match of candidate landmarks $v_{i_1}, \ldots, v_{i_n}$ to the template landmarks $x_1, \ldots, x_n$. Let S be the index set $\{1, \ldots, n\}$ and let ε be a set of pairs of elements in S. The pair (S, ε) is a graph; S is the set of *vertices* and ε is the set of *edges*. See, for example, Fig. 1. Two vertices are called neighbors if they correspond to an edge. For any graph, a *clique* $C \subset S$ is a maximal subset of S whose elements are all neighbors to each other. Let $\mathcal{C}$ be the set of cliques of a graph (S, ε). (Note that ε and $\mathcal{C}$ can be determined from each other.) In the present context each clique consists of exactly three vertices. A cost function $\psi_C(i_k; k \in C)$ is assigned to each clique C, which depends only on the candidate landmarks v_{i_k} for $k \in C$. Of interest is the candidate match that minimizes

$$\Psi(i_1, \ldots, i_n) = \sum_{C \in \mathcal{C}} \psi_C(i_k; k \in C).$$

The minimizer is denoted by $(i_1^*, \ldots, i_n^*)$ and will be referred to as the *best match*. The graph and the functions $\psi_C(\cdot)$ are chosen to ensure that the geometric configuration of the points in the best match is close to that of the template landmarks with respect to some geometric criteria which fall under the general title of *shape*. In the experiments this is formulated in terms of relative distances between landmarks and the functions $\psi_C(\cdot)$ are constructed so that a candidate match will be penalized if the relative distances between v_{i_k}, $k \in C$ do not match closely with those between x_k, $k \in C$ for each clique C. Specific choices of $\psi_C(\cdot)$ are discussed below. We describe here the computations required to find the best match.

Because there are m possibilities for each of the n landmarks, there are a total of m^n candidate matches. Evaluating Ψ for every candidate match is feasible only for very small n. However, by applying dynamic programming, see [6], and putting certain restrictions on the graph, it is demonstrated that the amount of computations necessary to find the best match is proportional to nm^3. For computational and modeling purposes, we focus on a class of graphs having the following properties.

PROPERTY I. The vertices of the graph can be ordered in such a way that all the neighbors of vertex 1 are neighbors of each other, so that together with 1 they form a clique, denoted C_1; If the vertices are eliminated one at a time from the graph in increasing order, after vertex t (for $t = 1, \ldots, n-3$) is eliminated, all the neighbors of vertex $t+1$ in the new subgraph are neighbors of each other, namely together with $t+1$ they form a clique, denoted C_{t+1}. Henceforth this will be the ordering of the vertices of the graph.

PROPERTY II. Following the order of elimination in the description of Property I, any vertex t, for $t = 1, \ldots, n-2$, has exactly two neighbors a_t, b_t when it is being eliminated, namely it is not in any other clique of the current subgraph.

Property I is one of many possible characterizations of a well studied class of graphs called *decomposable graphs* [5], [26].

It can be shown that properties I and II imply that for $t = 1, \ldots, n-3$, the vertices a_t, b_t of property II belong to at least one other clique C_{ut} such that $u_t > t$. This is called the *running intersection property* and is discussed in more detail below. These concepts are illustrated in Fig. 1.

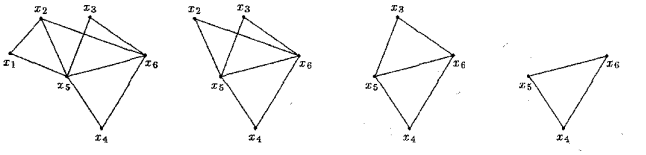
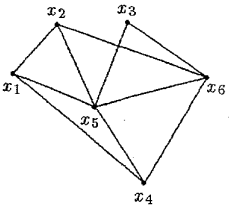


Fig. 1a. The steps in the successive elimination of the vertices and the corresponding cliques. The cliques of the graph are: $C_1 = \{x_1, x_2, x_5\}, C_2 = \{x_2, x_5, x_6\}, C_3 = \{x_3, x_5, x_6\}, C_4 = \{x_4, x_5, x_6\}$. The indices u_t defined in the running intersection property are: $u_1 = 2$, $u_2 = 3$, $u_3 = 4$. Also $(a_1, b_1) = (2, 5)$ and $(a_2, b_2) = a_3, b_3) = (5, 6)$.



With an edge between x_1 and x_2 the graph is not decomposable

Fig. 1b. Dynamic programming for minimizing the function $\psi(x_1, \ldots, x_6) = \sum_{t=1}^{4} \psi_C(x_k, k \in C)$ amounts to carrying out the following iterative minimization from the interior parentheses and out. Observe that each minimization only involves calculations on triples of variables:

$$\min_{x_4, x_5, x_6} \left[\psi_{C_4}(x_4, x_5, x_6) + \min_{x_3} \left(\psi_{C_3}(x_3, x_5, x_6) + \min_{x_2} \left(\psi_{C_2}(x_2, x_5, x_6) + \min_{x_1} \psi_{C_1}(x_1, x_2, x_5) \right) \right) \right].$$

2.4 Dynamic Programming

The idea behind dynamic programming can be illustrated in the following simple example. Let $f(x_1, x_2, x_3, x_4) = g(x_1, x_2, x_3) + h(x_2, x_3, x_4)$, where x_i are variables in a finite set V. For each $x_2, x_3 \in V$ let $g^*(x_2, x_3) = \min_{x_1} g(x_1, x_2, x_3)$, where $x_1^*(x_2, x_3)$ denotes the value of x_1 at which the minimum is achieved. Then

$$\min_{x1,x2,x3,x4} f(x_1, x_2, x_3, x_4) = \min_{x2,x3,x4} g^*(x_2, x_3) + h(x_2, x_3, x_4) = \min_{x2,x3,x4} \tilde{f}(x_2, x_3, x_4).$$

The minimum on the right is achieved at (x_2^*, x_3^*, x_4^*) if and only if the minimum on the left is achieved at $(x_1^*(x_2^*, x_3), x_2^*, x_3^*, x_4^*)$. The number of combinations needed to obtain g^* and x_1^* for each $x_2, x_3 \in V$ is $|V|^3$ where $|V|$ is the cardinality of V. The number of combinations needed to find the minimum of $\tilde{f}$ is the same. Doing the minimization by brute force would require trying $|V|^4$ combinations.

More formally, since 1 is the vertex in C_1 which is not in any other clique, we can write

$$\Psi(i_1, \ldots, i_n) = \sum_{t=1}^{T} \psi_{C_t}(i_k; k \in C_t) = \psi_{C_1}(i_k; k \in C_1) + \sum_{t=2}^{T} \psi_{C_t}(i_k; k \in C_t)$$
$$\stackrel{\text{def}}{=} \psi_{C_1}(i_k; k \in C_1) + \Lambda(i_2, \ldots, i_n).$$

Assume $C_1 = \{1, a, b\}$. For any fixed pair (i_a, i_b) of candidate points, let $i_1^*[i_a, i_b]$ be the choice of i_1 which minimizes $\psi_{C_1}(i_1, i_a, i_b)$. Since $\Lambda(\cdot)$ does not depend on i_1, it is easy to see that in the optimal match $i_1^*, \ldots, i_n^*$ satisfies $i_1^* = i_1^*[i_a^*, i_b^*]$. Note that finding $i_1^*[i_a, i_b]$ for all possible pairs $i_a, i_b = 1, \ldots, m$ requires evaluating $\psi_{C_1}(i_1, i_a, i_b)$ for all possible combinations of i_1, i_a and i_b. Hence, the amount of computations is proportional to $|V|^3 = m^3$. For each pair (i_a, i_b) the index $i_1^*[i_a, i_b]$ is stored as well as $\psi_{C_1}(i_1^*[i_a, i_b], i_a, i_b)$, so that the amount of storage required is proportional to m^2.

By the running intersection property there exists a clique C_{u_1} which contains vertices a and b. Let $C_{u_1} = \{a, b, c\}$. Define

$$\phi_{C_{u_1}}(i_k; k \in C_{u_1}) = \phi_{C_{u_1}}(i_a, i_b, i_c) = \psi_{C_{u_1}}(i_a, i_b, i_c) + \psi_{C_1}(i_1^*[i_a, i_b], i_a, i_b),$$
$$\phi_{C_t}(i_k; k \in C_t) = \psi_{C_t}(i_k; k \in C_t) \text{ for } t = 2, \ldots T,\ t \neq u_1,$$

and

$$\Phi(i_2, \ldots, i_n) = \sum_{t=2} \phi_{C_t}(i_k; k \in C_t).$$

Note that $\psi_{C_1}(i_1^*[i_a, i_b], i_a, i_b)$ is a function of i_a and i_b only, and hence $\phi_{C_{u_1}}(\cdot)$ is a function of $(i_k, k \in C_{u_1})$. The amount of computations required to evaluate the function $\phi_{C_{u_1}}(\cdot)$ for all triples of candidate points is again proportional to m^3. It is easily seen that $\Phi(i_2^*, \ldots, i_n^*) = \Psi(i_1^*, \ldots, i_n^*)$ and $(i_2^*, \ldots, i_n^*)$ is the minimizer of $\Phi(\cdot)$. The original problem is thus reduced to a similar one with one vertex *eliminated*. By *eliminating* the vertices one at a time in increasing order of the labels, the problem can be solved when only the last three vertices $n-2$, $n-1$, n, are left (Fig. 1). Note that when $i_{n-2}^*, i_{n-1}^*, i_n^*$ are determined, we can go backwards and determine $i_{n-3}^*, \ldots, i_1^*$ sequentially. Because the amount of computations needed to eliminate one vertex is proportional to m^3, the total amount of computations is proportional to nm^3. From the discussion in the next section it follows that this procedure necessarily covers all cliques in the graph and thus obtains the minimum of the function Ψ.

In light of the discussion above it is now possible to appreciate the importance of Properties I and II. Computationally, requiring that the graph is decomposable and that $|C| = 3$ for all $C \in \mathcal{C}$, ensures that each step of dynamic programming involves computations of order m^3 and hence is manageable in many circumstances. Moreover the homogeneity in the local structure of the graph greatly simplifies the implementation of the dynamic programming algorithm in terms of book keeping requirements, as can be seen in the implementation section below.

2.5 Some More on Decomposable Graphs

To further understand the implications of Properties I and II, it is helpful to re-express them in the following way.

PROPERTY I*. *The running intersection property*: There exists an ordering of the cliques, $C_T, C_{T-1}, \ldots, C_1$, such that, for $T > t \geq 1$, there exists $u_t > t$ such that

$$C_t \cap \left(\bigcup_{s=t+1}^{T} C_s \right) = C_t \cap C_{u_t}.$$

PROPERTY II*. The cliques of the graph are all triples and the number of cliques is equal to the number of vertices minus 2, i.e, $|C| = 3$ for $C \in \mathcal{C}$ and $|\mathcal{C}| = T = n - 2$.

Results established by [14] and [12] demonstrate that Properties I and I* are equivalent. Given Property I, Properties II and II* are equivalent as can be seen from the following discussion.

Suppose the cliques are labeled so that the running intersection property is satisfied. Because C_t is a clique, $C_t \neq C_t \cap C_{u_t}$, which implies that C_t, $t = T-1, \ldots, 1$, must contain at least one vertex which is not in the cliques with higher labels. Hence, $n \geq |C_T| + T - 1$. From Property II*, $|C_T| = 3$, so $n \geq T + 2$. However, Property II* also specifies that n is exactly $T + 2$, which implies that C_t, $t = T-1, \ldots, 1$, has exactly one *new* vertex which is not in the cliques with higher labels. This means $|C_t \cap C_{u_t}| = 2$. for $t = T-1, \ldots, 1$. We now relabel the vertices in the following fashion. Let the three vertices in C_T be arbitrarily labeled as n, $n-1$, $n-2$. For $t = T-1, \ldots, 1$, let the new vertex in C_t be labeled as t. From here on, the vertices are assumed to be labeled this way. This labeling obviously satisfies Properties I and II. Indeed, the two neighbors of vertex i, $i = 1, \ldots, n-2$, at the time it is eliminated are the two other vertices in C_i. From the above it also follows that if Properties I and II are satisfied then,

PROPERTY III. For $t < T$, the subgraph defined with vertex set $S_{t+} = \{t, t+1, \ldots, n\}$ and clique set $\mathcal{C}_{t+} = \{C_t, C_{t+1}, \ldots, C_T\}$ satisfies properties I, II as well.

Property II allows us to introduce genuinely rotation and scale invariant functionals. Consider the cliques and vertices in reverse order as is done in the statement of Property I*. With Property II*, every time a new vertex t, $t = n-3, n-2, \ldots, 1$, is added, a new clique C_t is in-

troduced. The associated function $\psi_{C_t}(i_k;\ k \in C_i)$ allows us to model the location of the new (tth) landmark relative to *two of the existing* landmarks. Namely in terms of relative lengths and angles. The fact that it is two of the existing landmarks instead of one is very important. To understand this, suppose Property II* is violated so that for some t, $C_t \cap C_{u_t}$ is a singleton set $\{\tau\}$. The graph can then be partitioned into two pieces which are only connected through the vertex τ. If the functions $\psi_{C_t}(\cdot)$ are restricted to those which are scale and rotation invariant, the angle between the two pieces cannot be modeled.

Remark: Suppose a graph satisfies Property I but does not necessarily satisfy Property II. Then, by applying dynamic programming in a similar fashion, it can be shown that the amount of computations required to find the best match is proportional to

$$\sum_{t=1}^{T} m^{|C_t|}.$$

In general, if $G = (S, \varepsilon)$ does not satisfy Property I, i.e., it is not decomposable, then the computations for finding the best match will be given by the above formula, however summing over the cliques of some decomposable graph $G^{\Delta} = (S, E^{\Delta})$ of which G is a subgraph. The graph G^{Δ} depends on the order of elimination when dynamic programming is applied. Algorithms related to finding the optimal elimination order can be found in [6]. For example in order to apply dynamic programming to the original graph in Fig. 2a, it would be necessary to embed it in some G^{Δ} which would have some cliques of more than three elements.

2.6 Implementation

The dynamic programming procedure described above is implemented as follows. Two arrays *book_keepf* and *book_keepi* are created with rows corresponding to all possible pairs of landmark candidates, and columns corresponding to each

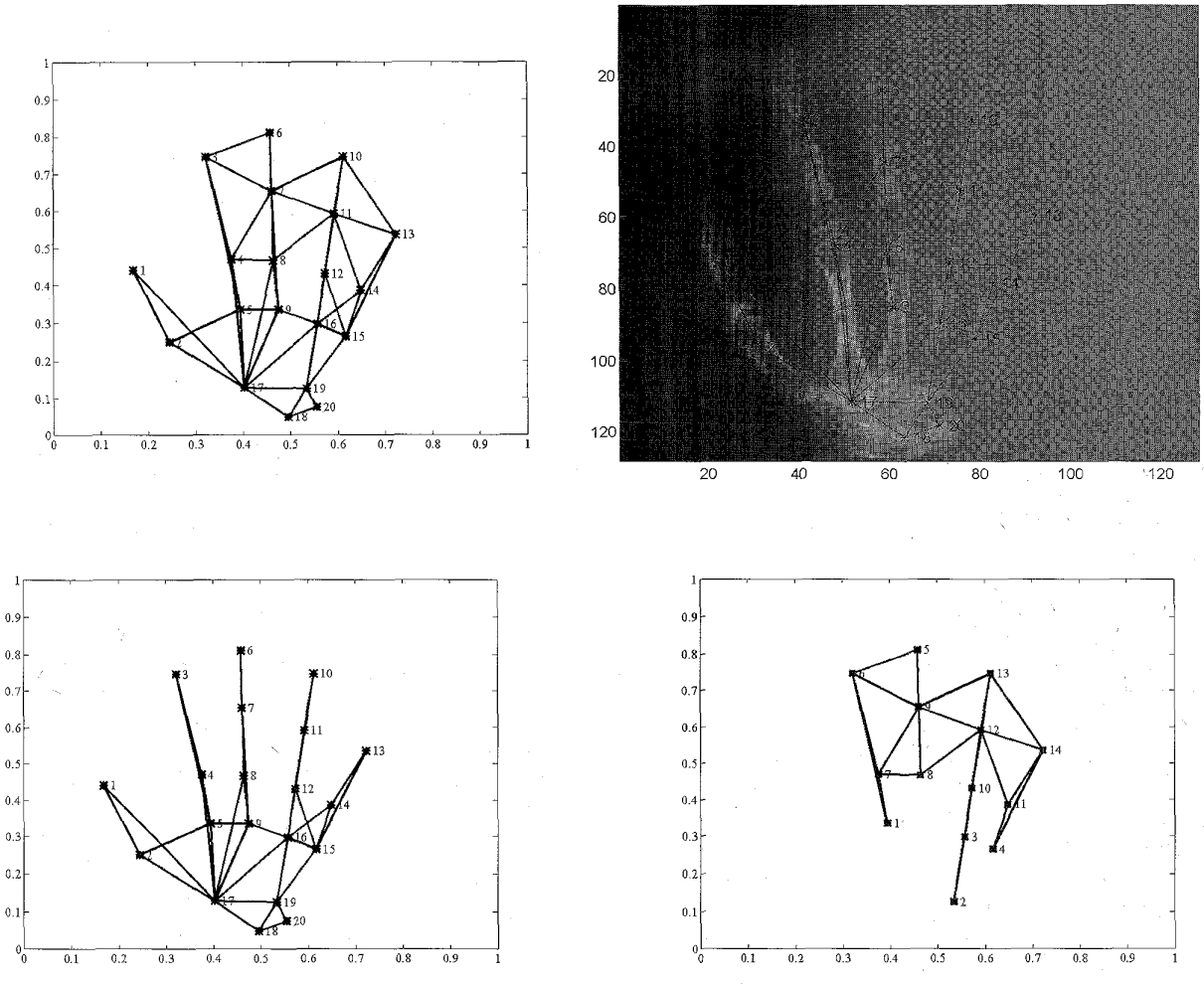


Fig. 2. Top left: The full graph: The triples **(3, 4, 5), (6, 7, 8), (7, 8, 9), (10, 11, 12), (11, 12, 16)** are all cliques for which the three corresponding landmarks are colinear. Top right: The graph overlayed on the template image. Bottom left: First sub-graph. Bottom right: Second sub-graph. The vertices in all graphs are numbered according to the order of elimination.

clique in the decomposable subgraph. Thus both arrays are of dimension $m^2 \times T$. In the first step for each possible assignment i_a, i_b of a pair of candidates to vertices a, b in the first clique of the graph, the value of $\psi_{C_1}(i_1, i_a, i_b)$ is calculated for each possible assignment i_1 of a candidate to vertex 1. As above let $i_1^*[i_a, i_b]$ denote the index of the data point at which the minimum is achieved. The value of $\psi_{C_1}(i_1^*[i_a, i_b], i_a, i_b)$ is stored in *book_keepf*[(i_a, i_b), 1]. The index $i_1^*[i_a, i_b]$ is stored in *book_keepi*[(i_a, i_b), 1]. Looping over all pairs, the amount of computation involved in this step is proportional to m^3.

At the tth clique with graph vertices (α, β, γ), for each assignment i_β, i_γ, the function $f(i_\alpha) = \psi_{C_t}(i_\alpha, i_\beta, i_\gamma)$ is calculated. If (a, b) is contained in any previously eliminated clique, namely in any clique C_τ with $\tau < t$, the one with largest index τ^* is found, i.e. the last such clique. The value of *book_keepf*[(i_α, i_β), τ^*] is added to $f(i_\alpha)$. Similarly, if (a, c) is contained in any eliminated clique $\tau < t$, the value of *book_keepf*[(i_α, i_γ), τ^*] is added to $f(i_\alpha)$, where τ^* is the last such clique. Similarily for (b, c). For given (i_β, i_γ) this is done for every $i_\alpha = 1, \ldots, m$. The index $i_a^*[i_\beta, i_\gamma]$ which minimizes the updated $f(i_a)$ for the pair (i_β, i_γ) is stored in *book_keepi*[(i_β, i_γ), t], the value of $f(i_a^*[i_\beta, i_\gamma])$ is stored in *book_keepf*[(i_β, i_γ), t].

After looping through all cliques find the row in *book_keepf* which contains the minimal value in the *last* column T. Rename the corresponding pair as (i_{n-1}^*, i_n^*). The corresponding data points match the graph vertices $n-1$, n, respectively. The index $i_{n-2}^* = book\_keepi[(i_{n-1}^*, i_{n-2}^*), T]$ will match the graph vertex $n-2$. Now go to vertex $n-3$ which must lie in clique $T-1$ and which must have a pair in common, say ($n-2$, n), with clique T. Then $i_{n-3}^* = book\_keepi[(i_{n-2}^*, i_n^*), T-1]$. Continue backwards in the same way. Vertex t in the graph is in clique C_t which must have an edge in common with a subsequent clique $\tau > t$. Since all the vertices in C_τ have already been matched to data points, the same lookup in *book_keepi* will yield the data point matching the tth vertex. At the end of the reverse loop on the cliques we obtain the optimal match of data points to the graph vertices, with respect to the functional Ψ.

2.7 The Clique Functions

The clique functions on the triangles of the graph were set as follows. Given a triangle a_1, a_2, a_3 of landmarks corresponding to a clique C in the graph (S, ε), labeled according to the order in which they are eliminated, and with angles α_1, α_2, α_3 respectively, let l_i denote the length of the edge opposite the point a_i for $i = 1, 2, 3$. The cost function is taken to be the sum of squares of the differences of the log ratios of the lengths in the data and the log ratios of the lengths in the templates. More precisely let b_1, b_2, b_3 be three data points with indices i_1, i_2, i_3. Let r_1, r_2, r_3 be the respective lengths of the edges in the triangle, and let β_1, β_2, β_3 the respective angles, the following quantity is calculated

$$\psi_C(i_1, i_2, i_3) = J(r_1, r_2, r_3) = \left(\log(r_1 / r_2) - \log(l_1 / l_2)\right)^2 + \left(\log(r_2 / r_3) - \log(l_2 / l_3)\right)^2 + \left(\log(r_3 / r_1) - \log(l_3 / l_1)\right)^2.$$

Although triangles with similar log-ratios have similar shapes they may have different orientations. Keeping the triangles oriented in the same way as in the template graph is necessary for preserving the planar configuration of the vertices. On the other hand certain triangles with very small angle α_1 for example, should be allowed to change orientation around the vertex α_1, namely they are allowed to flip around the edges a_1a_2 or a_1a_3.

These considerations were implemented as follows. A certain tolerance ε is set. If $|\sin(\alpha_1)| < \varepsilon$ the sign of β_1 is ignored. However the triangle b_1, b_2, b_3 is ruled out if $|\sin(\beta_1)| > \varepsilon$. If on the other hand $|\sin(\alpha_1)| > \varepsilon$ then the triangle b_1, b_2, b_3 is ruled out if $sign(\beta_1) \neq sign(\alpha_1)$. The same procedure is applied to α_2, β_2. In the experiments $\varepsilon = .25$.

The J functional is very convenient because it involves only pairs of points. The log-distances between every two points in the data are calculated and stored ahead of time. This does not require too much memory for data sets of 50-100 points. Calculating J thus reduces to three multiplications. Most of the calculation is concentrated in checking the angle constraints. The angles involve triples of points and precalculating them requires much more memory. In our implementation we have calculated them online.

There are many other functionals which are translation and rotation invariant and describe the shape of the triangle in some sense. For example one could work directly with the three angles and penalize deviations from the sines of the angles. Other possibilities can be found in the literature on statistical models of shape see for example [7] and the discussion thereof. In general the relationships between these different types of cost functions are highly non-linear and do not lend themselves to any simple form of analysis.

It should be emphasized that the clique function ψ_C could include terms of the form $\sum_{k \in C} \sigma(v_{i_k}, x_k)$ where σ quantifies how wellthe candidate v_{i_k} matched the local operator corresponding to the kth vertex in the graph. The presence of such a term does not affect the possibility of minimizing through dynamic programming. In principle such a term could replace the "hard" threshold used for the local operators, however this would imply that every pixel in the image is a candidate for each vertex thus

causing an explosion in the size of V, and subsequently in the computation time and memory requirements. Thus the threshold serves as a pruning mechanism to limit the size of the state space.

3 Description of Experiments

3.1 Low Resolution X-Rays

The experiments were carried out on a collection of x-ray images of hands. One of the images was chosen arbitrarily, as the template and is referred to as x-ray no. 1. The local maxima of the image were highlighted. Of these local maxima 20 were chosen which seemed to represent important landmarks in the hand. The next step was to create a graph with these 20 vertices. This again was done manually. The vertices together with the edges of the graph are shown in Fig. 2a. Fig. 2b shows the graph overlaid on the original x-ray no. 1. The two subgraphs used iteratively are shown in Figs. 2c and 2d. Note that both subgraphs are decomposable even though the full graph is not. First the optimization was carried out on graph 2c. Then vertices 5, 15, 16, and, 19 were held fixed and the optimization was carried out on graph 2d.

At the outset of the project only graph 2c was used. Attempts were made to prevent erroneous matchings such as in Fig. 3b, by creating complicated cost functions and introducing a host of hard constraints, to prevent extreme distortions of some triangles while allowing more flexibility for others. This soon led to the introduction of an intractable number of parameters and is clearly conceptually the wrong way to do things. Subsequently graph 2d was introduced to complement graph 2c, the cost functions were simplified as well as the hard constraints. One iteration on each subgraph produced matches with the proper planar configuration. It should be noted that even if the matching of the first subgraph produced non-overlapping matches for the fingers, their length may turn out disproportional, see for example Fig. 4b. The additional triangles between the fingers in the second subgraph help in adjusting these proportions.

The data points were extracted by scanning over all pixels and choosing those at which the intensity was greater than the intensity at a fraction ρ of the neighbors in a $L \times L$ neighborhood. We used $\rho = .9$ and $L = 9$. This procedure creates clusters which are thinned out by averaging a certain minimum distance between any two data points. The algorithm is robust to variations in the parameters related to the extraction of the data points, as long as correct points are found corresponding to each point in the model in addition to the erroneous ones. Thus the algorithm will perform fine with more data points, for example by decreasing ρ or L. The only problem is that an increase in data points severely slows down the dynamic programming procedure. Thus with 40-50 data points the minimization would take less than a minute on a SPARC 10. With 70 data points it would take 3-4 minutes. In the first two experiments (Figs. 3 and 4) we eliminated local maxima with a pixel intensity lower than some fraction of the maximal pixel intensity. This took care of many spurious local maxima in the background. The third experiment shows the result without

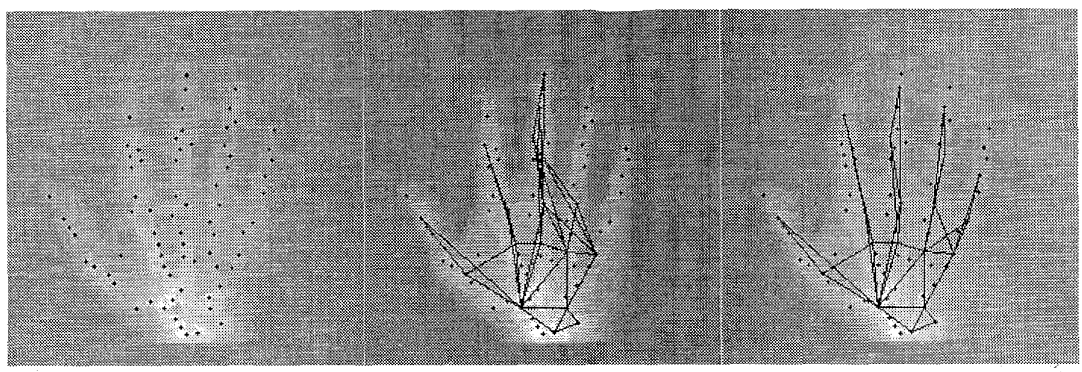


Fig. 3. Left: Extracted local minima. Middle: First match. Right: Second match.

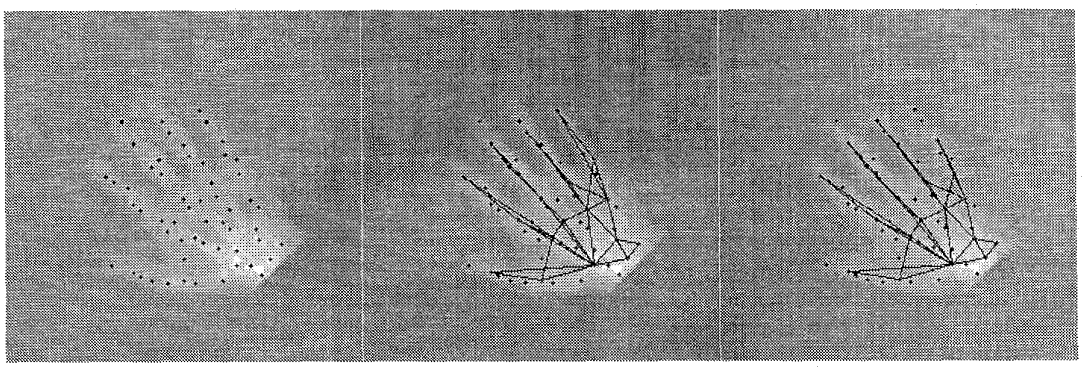


Fig. 4. Left: Extracted local minima. Middle: First match. Right: Second match.

eliminating the background points. The algorithm still performs fine however slows down considerably due to the presence of more data points.

Fig. 3a shows the image of x-ray no. 2 with the points identified as local maxima. Fig. 3b shows the initial optimal match of the template graph to these points using the first subgraph. Fig. 3c shows the final match after using the second subgraph.

Figs. 4a, 4b, and 4c shows the same sequence for x-ray no. 3 which has also been rotated and scaled. Since the local operator and the cost functions are inherently scale and rotation invariant, the algorithm was able to find the correct match of the template, and the interpolation automatically incorporated an appropriate scale and rotation.

Finally Figs. 5a, 5b, and 5c shows a match where the background local maxima are not eliminated.

3.2 High Resolution X-Rays

We now describe an experiment with high resolution x-rays, and four types of local operators. The aim here was to obtain precise locations of specific anatomies. The local maximum operator was again used to identify the finger joints. The second operator identified the points of contact of the finger bones, the third which is the second flipped along the horizontal axis identified a point of bifurcation in the bone thickness which is present in all fingers, and the fourth operator identified the point of contact between the thumb and the palm. The matrices L_1, L_{-1} overlayed for second and fourth operator are given below.

$$\begin{pmatrix} -1 & -1 & -1 & -1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & 0 & 0 & 0 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & 0 & 0 & 0 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 \\ -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \end{pmatrix}$$

The four thresholds were 0.7, 0.85, 0.8, 0.7 To eliminate multiple points in the background another threshold was introduced for each local operator so that candidate pixels with intensity lower than the threshold were ruled out. These operators are definitely not rotation invariant, although they are robust to small rotations. Again two sub-

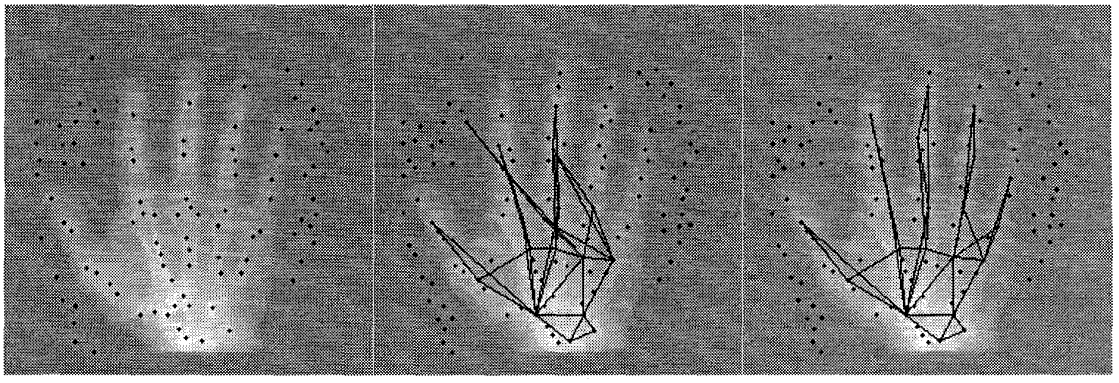


Fig. 5. Left: Extracted local minima. Middle: First match. Right: Second match.

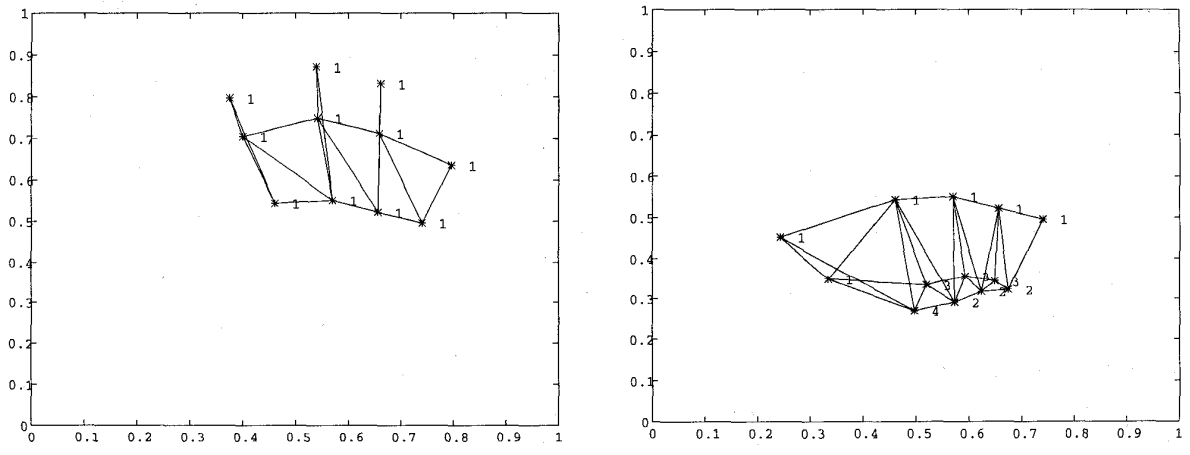


Fig. 6. The two decomposable subgraphs with color number at each vertex.

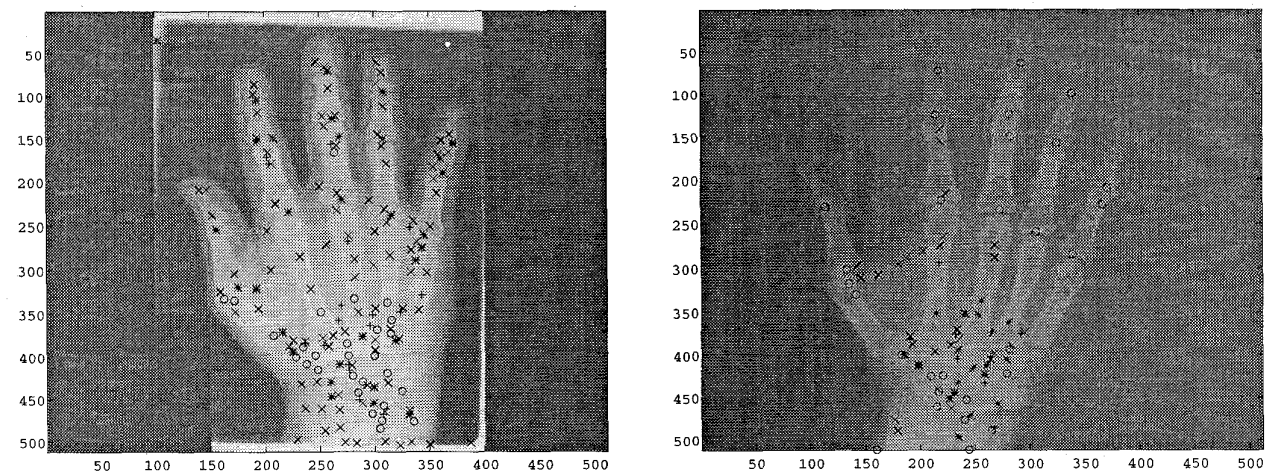


Fig. 7. The candidates for the four colors found in the data. $* = 1$, $+ = 2$, $o = 3$, $x = 4$.

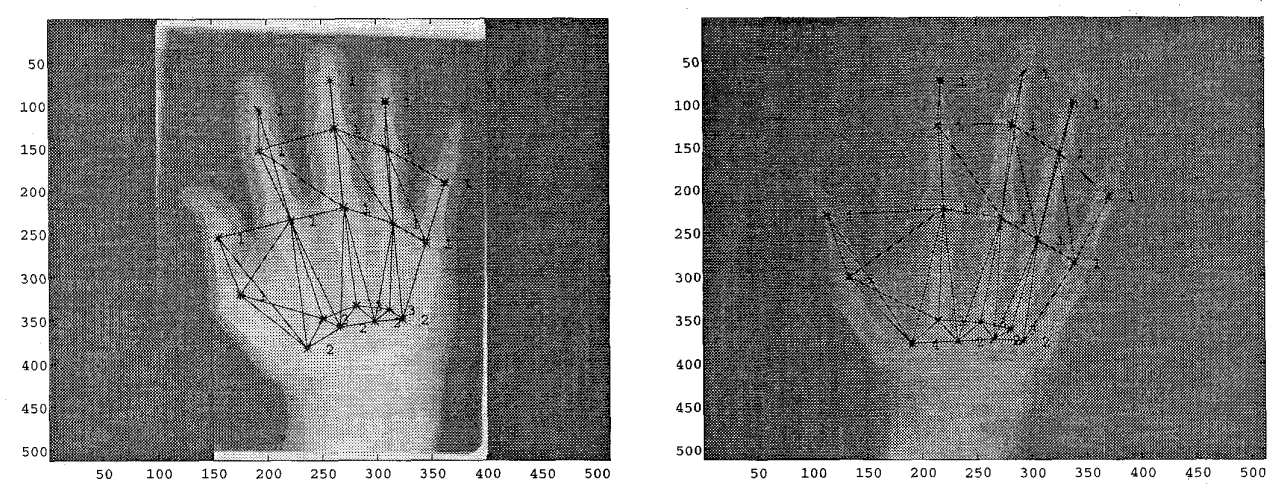


Fig. 8. The match of the the two graphs overlayed.

graphs were used, one for the area of the fingers and one for the upper palm. Fig. 6 shows the two components of the the template graph with the operator or color number.

The left hand graph was matched first. The four points at the bottom were held fixed and then the right hand graph was matched. Observe that the right hand graph is decomposable only because the top four points are given. The reason is that once the location of these points is determined, the cost on triangles they belong to is simply added to the cost of any other triangle with a common edge. The dynamic programming is done on the decomposable graph obtained by cutting out these four points and the edges emanating from them, with updated cost functions.

Fig. 7 shows the candidate points in two data images, along with their color number. Fig. 8 shows the match of the two subgraphs to the data images.

This procedure has been successful on a collection of ten high resolution hand x-rays of highly varying sizes, shapes and in particular angles between the fingers.

Fig. 9 shows the difference image of a failed match using only the elastic deformation algorithm between two hands.

For the same two hands Fig. 10 shows one as the template image and the interpolated warp of the template image obtained from the graph match. Fig. 11 shows the improvement obtained by running the elastic deformation algorithm initialized with the warped template (10b), and the target image.

4 Conclusion

We have suggested the matching of landmark graphs to collections of extracted data points in the data image. The cliques of the landmark graphs are all triangles and the cost of a match is expressed in terms of the deviations of the shapes of the triangles. The actual implementation of the match is carried out through dynamic programming on *decomposable* subgraphs of the original template graph. In the experiments described above we iterated between two such subgraphs keeping certain points of the first match fixed. The matched points can then interpolated to create a

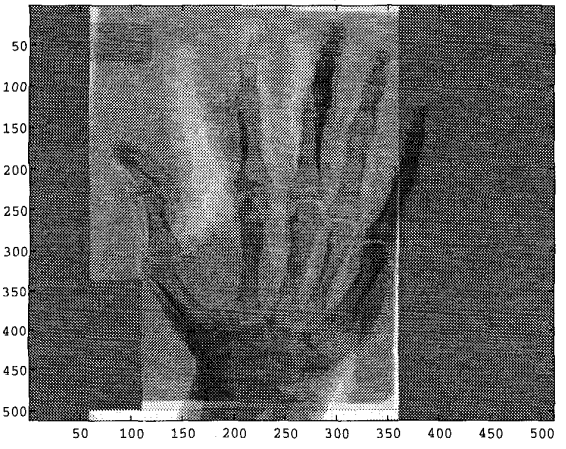


Fig. 9. Failure of elastic deformation algorithm: difference between warped template and target image.

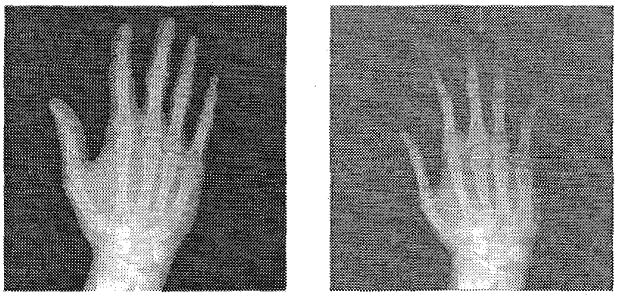


Fig. 10. Left: Template image. Right: Warped template after graph match.

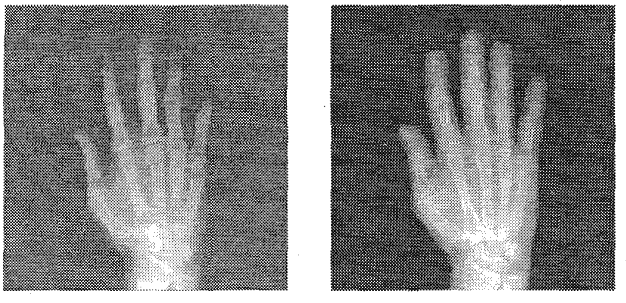


Fig. 11. Left: Elastic deformation algorithm initialized with 10(upper right). Right: The target image.

deformation which provides an input to an elastic deformation algorithm.

The power of this method is that robust local operators can be used to identify candidate landmark points, together with a computationally feasible optimization method which does not require initialization, to extract the optimal match. At this point the method fails entirely if one of the true locations is not picked up by the local operators. This could occur due to occlusion, but also in the presence of abnormalities. Another problem is the lack of clear guidelines as to how to construct the graph. Here we were essentially guided by our intuition. It is easy however to automatically generate decomposable graphs from collections of vertices, it would therefore be interesting to study how sensitive the method is to different graphs.

Given a large enough data set from a family of images it would be possible to actually estimate the appropriate cost functions or certain relevant parameters. This could either be done manually by pointing out the correct matches for the images in the data set, or using the cost functions suggested above to calculate these matches and then estimate the parameters, which are subsequently used to update the cost functions. Once the parameters are estimated it may be possible to deal with occlusion and missing components, to characterize subfamilies and identify abnormalities.

ACKNOWLEDGMENTS

We would like to express our thanks to Stu Geman for some very fruitful conversations. Special thanks are due to Dr. Roderick Birnie and Dr. Maryellen Giger of the University of Chicago Hospital for providing us with the high resolution digitized x-ray images.

This research has been supported in part by the University of Chicago Block Fund, ARO DAAL03-92-G-0322, and National Institutes of Health grant no. R01-GM46800.

REFERENCES

[1] Y. Amit, "A non-linear variational problem for image matching," *SIAM J. of Sci. Comput.*, vol. 15, no. 1, pp. 207-224, 1994.

[2] Y. Amit, U. Grenander, and M. Piccioni, "Structural image restoration through deformable templates," *J. American Statistical Assoc.*, vol. 86, no. 414, pp. 376-387, 1991.

[3] R. Bajcsy and S. Kovacic, "Multiresolution Elastic Matching," *Computer Vision, Graphics, and Image Processing*, vol. 46, pp. 1-21, 1989..

[4] M. Barzohar and D.B. Cooper, "Automatic finding of main roads in aerial images by using geometric—Stochastic models and estimation," *Proc. ARPA IU Workshop Washington, D. C.*, 1993.

[5] C. Berge, *Graphs and Hypergraphs*, North Holland, Amsterdam, 1973.

[6] U. Bertele and F. Brioschi, "Nonserial dynamic programming," Mathematics in Science and Engineering series, vol. 91, Academic Press, 1972.

[7] L.F. Bookstein, "Size and shape spaces for landmark data in two dimensions," *Statistical Science*, vol. 1, no. 2, pp. 181-242, 1986.

[8] L.F. Bookstein, "Toward a notion of feature extraction for plane mappings," *Proc. Tenth Int'l Conf. Information Processing in Medical Imaging*, 1987.

[9] L.F. Bookstein, *Morphometric Tools for Landmark Data: Geometry and Biology*. Cambridge, Mass. Cambridge University Press, 1991.

[10] T.F. Cootes, C.J. Taylor, D.H. Cooper, and J. Graham, "Training models of shape from sets of examples," *Proc. BMVC*, pp. 9-19, 1992.

[11] T.F. Cootes, and C.J. Taylor, "Active shape models—Smart snakes," *Proc. BMVC*, pp. 267-275, 1992.

[12] J.N. Darroch, S.L. Lauritzen, and T.P. Speed, "Markov fields and log-linear interaction models for contingency tables," *Ann. of Stat.*, vol. 8, pp. 522-539, 1980.

[13] U. Grenander, "A unified approach to pattern analysis," *Advances in Computers*, vol. 10, 1970.

[14] S.J. Haberman, *The Analysis of Frequency Data*, IMS monographs, Chicago: Univ. Chicago Press, 1974.

[15] R.M. Haralick and G.L. Shapiro, *Computer and Robot Vision*, vols. 1 - 2, Reading Mass: Addison Wesley, 1992.

[16] H. Ogawa, "Labeled point pattern matching by Delaunay triangulation and maximal cliques," *IEEE Pattern Analysis and Machine Intelligence*, vol. 19, no. 1, pp. 35-40, 1986.

[17] B.K.P. Horn and B.G. Schunck, "Determining optical flow," *Artificial Intelligence*, vol. 17, pp. 185-203, 1981.
[18] T.S. Huang and R.Y. Tsai, "Image sequence analysis: Motion estimation," *Image Sequence Analysis*, T.S. Huang, ed., New York: Springer-Verlag, 1981.
[19] Z. Jin and P. Mowforth, "A discrete approach to signal mapping," Technical Report TIRM-88-036, the Turing Institute, Glasgow, Scotland, 1988.
[20] J. Meinguet, "Multivariate interpolation at arbitrary points made simple," *Zeit. für Angewandte Mathematik und Physik (ZAMP)*, vol. 30, pp. 292-304, 1979.
[21] M. Miller, G. Christensen, Y. Amit, and U. Grenander, "A mathematical textbook of deformable neuro-anatomies," *Proc. National Academy of Science*, vol. 90, pp. 11,944-11,948, 1993,.
[22] H.H. Nagel, "Displacement vectors derived from second-order intensity variations in image sequences," *Computer Vision, Graphics and Image Processing*, vol. 21, pp. 85-117, 1983.
[23] Y. Ohta and T. Kanade, "Stereo by intra and inter scanline search using dynamic programming," *IEEE Pattern Analysis and Machine Intelligence*, vol. 7, no. 2, pp. 139-154, 1985.
[24] R.R. Petrocelli, J.L. Elion, and M.M. Manbeck, "A new method for structure recognition in unsubtracted digital angiograms," *Proc. Computers in Cardiology*, IEEE Computer Soc., pp. 207-210, 1992.
[25] D.B. Phillips and A.F.M. Smith, "Bayesian Faces," Technical Report TR-93-02, Dept. of Mathematics, Imperial College 1993.
[26] D.J. Rose, R.E. Tarjan, and G.S. Leuker, "Algorithmic aspects of vertex elimination on graphs," Siam J. Comput., vol. 5, pp. 266-283, 1976.
[27] D. Terzopoulos, "Image analysis using multigrid relaxation methods," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 8, no. 2, 1986.
[28] A.L. Yuille, D.S. Cohen, P. Hallinan, "Feature extraction from faces using deformable templates," *Proc. CVPR*, pp 104-109, 1989.

Yali Amit received his BS degree in mathematics in 1983 and his MS degree in mathematics in 1984, both from the Hebrew University, and his PhD in mathematics in 1988 from the Weitzman Institute of Science. In 1988, he received the Weitzman fellowship for post-doctoral research and spent the period between 1988 and 1991 as a visiting assistant professor in the Division of Applied Mathematics at Brown University. Since 1991, Dr. Amit has been an assistant professor in the Department of Statistics at the University of Chicago. His research interests include image analysis, deformable template models, medical imaging, Monte Carlo simulation, and information theory.

Augustine Kong received his BS degree in mathematics in 1980 from the California Institute of Technology, and his PhD in statistics in 1986 from Harvard University, where he spent one year as a post-doctoral research fellow. In 1987, Dr. Kong moved to the Department of Statistics at the University of Chicago, where he is now an associate professor. His research interests include genetic mapping, graphical models, Monte Carlo simulation, missing data problems, image analysis, and expert systems.