Learning Hierarchical Shape Models from Examples

Alex Levinshtein[1], Cristian Sminchisescu[1,2], and Sven Dickinson[1]

[1] University of Toronto, Canada
{babalex, sven}@cs.toronto.edu
Phone number: (416) 978-3853
[2] TTI-C, Chicago, USA
crismin@cs.toronto.edu

Abstract. We present an algorithm for automatically constructing a decompositional shape model from examples. Unlike current approaches to structural model acquisition, in which one-to-one correspondences among appearance-based features are used to construct an exemplar-based model, we search for many-to-many correspondences among qualitative shape features (multi-scale ridges and blobs) to construct a generic shape model. Since such features are highly ambiguous, their structural context must be exploited in computing correspondences, which are often many-to-many. The result is a Marr-like abstraction hierarchy, in which a shape feature at a coarser scale can be decomposed into a collection of attached shape features at a finer scale. We systematically evaluate all components of our algorithm, and demonstrate it on the task of recovering a decompositional model of a human torso from example images containing different subjects with dissimilar local appearance.

1 Introduction

The early generic object models proposed by researchers such as Marr and Nishihara [11] and Brooks [10] not only decomposed a 3-D object into a set of volumetric parts and their attachments, but supported the representation of objects at multiple scales, using an abstraction hierarchy. Marr's classical example of a human consists of a single cylindrical part at the highest level, a torso, head, and arms appearing at the next level, an upper arm and lower arm appearing at the next level, etc. Modeling an object at different levels of abstraction is a powerful paradigm, offering a mechanism for coarse-to-fine object recognition. Unfortunately, such models were constructed manually, and the feature extraction and abstraction machinery required to effectively recover volumetric parts, much less their abstractions, was not available at the time.

The recognition community has recently returned to the problem of modeling objects as configurations of parts and relations, with the goal of automatically recovering (or learning) such descriptions from examples. For example, collections of interest points [1, 7] or affine-invariant image patches [2], forming a "constellation" of features, capture the "parts" and their geometric relations that define a view-based object category. Armed with powerful new machine learning techniques, complex configuration models can be automatically recovered from image collections or image sequences. For example, one can extract models based on motion and persistent appearance [4, 6, 12, 3, 5]. Global detectors that combine both motion and appearance have also been successfully applied to pedestrian tracking tasks [13].

As powerful as these part-based techniques are, they all rely on computing a one-to-one correspondence between low-level, appearance-based features. However, two exemplars belonging to the same class may not share a single appearance-based feature. Yet at some higher level of abstraction, the two exemplars may share the same coarse part structure. Local, appearance-based features simply do not lend themselves to the types of abstract object representations proposed by Marr and his peers – abstractions in which a single part may cover an entire subcollection of local, appearance-based features. One approach might be to try and group the appearance-based features into local collections each of which defines an abstract part. However, appearance-based features are texture encodings of neighborhoods centered at interest points, and do not reflect the underlying shape structure required for perceptual grouping. Granted, the analysis of moving interest point-based features can support their partitioning into groups. But again, this requires the tracking of an exemplar, for which one-to-one feature correspondence is assured. Moreover, it is not clear how to abstract a coarse part model from a sparse set of local features.

In this paper, we address the problem of recovering a Marr-like abstraction hierarchy from a set of examples. We begin by applying a multi-scale blob and ridge detector [18] to a set of images containing exemplars drawn from the same class. The extracted features become the nodes in a *blob graph* whose edges reflect nonaccidental proximity relations between pairs of features. Blobs and ridges capture the coarse part structure of an object, and represent low-order projections of restricted classes of volumetric part models, including generalized cylinders, superquadric ellipsoids, and geons. Unfortunately, as feature complexity increases, so does its reliability decrease, as seen in Figure 1, showing the extracted blob graphs from a set of images of different humans with varying appearance and arm articulations. Some parts are over-segmented, some are under-segmented, some are missing, and some are spurious (possibly representing background clutter). These segmentation errors all pose a significant challenge to a matching algorithm whose goal is to find common structure in a set of images. Whereas one-to-one matching of local appearance-based features can exploit the high dimensionality of the features to ensure robust matching, one-to-one matching of noisy blobs and ridges is ripe with ambiguity, and structural relations and context must be exploited for successful matching.

Still, there is an even more challenging problem to be solved here. In Figure 1, sometimes an arm may appear as a single, elongated ridge (when the arm is extended), while at other times, an arm is broken into two smaller ridges (due to articulation at the elbow). Any matching algorithm that assumes a one-to-one correspondence between features cannot match these two descriptions, therefore failing to capture the notion that a coarser feature can be decomposed (at a finer level of abstraction) into two smaller features. Detecting these decompositional or abstraction relations between features requires a matching strategy that can match features many-to-many. Only then can we recover the multi-scale abstraction models that support true generic object recognition or categorization.

In this paper, we propose a framework for learning a shape abstraction hierarchy from a set of examples with dissimilar local appearance. From a set of noisy, poorly-segmented blob graphs, capturing the articulated part structure of objects at different

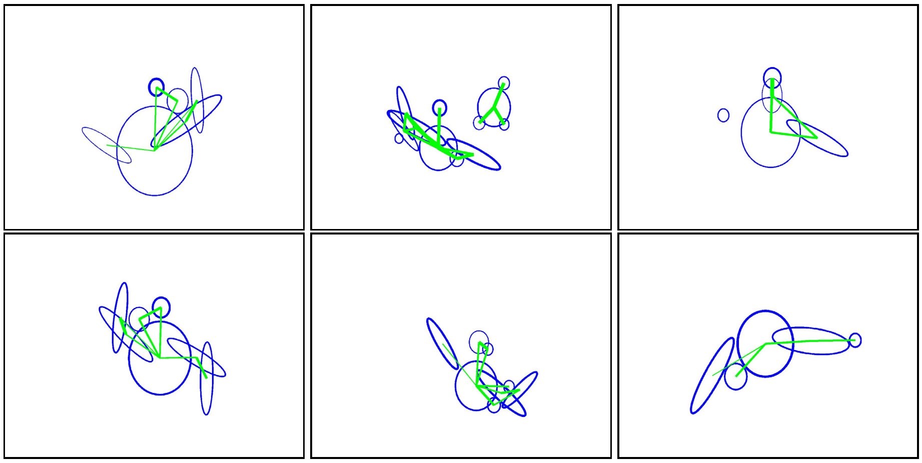

Fig. 1. Blob graphs extracted from a set of images, each containing the upper body of a different person (with different clothing). The high level of feature abstraction comes at the cost of increased segmentation errors in the form of under- and over-segmentation, missing features, and spurious features (including background clutter). Notice also that features may be extracted at different levels of abstraction, such as a straight arm (single ridge) or bent arm (two smaller ridges). Edges between blobs reflect a commitment to nonaccidental proximity-based grouping (see text) with edge width reflecting strength of grouping.

levels of abstraction, we construct an abstraction hierarchy, in the form of a graph, that contains both coarse-to-fine decompositional (abstraction) relations as well as attachment relations. Relaxing the one-to-one feature correspondence assumption common to most structure learning frameworks, we draw on recent results in many-to-many graph matching to match blob graphs many-to-many, allowing the matching of two exemplars whose parts may appear *at different levels of abstraction*. An analysis of the many-to-many matching results over all pairs of input exemplars ultimately yields the nodes and edges (both abstraction and attachment) in the final model.

We begin with a summary of related work (Section 2) and proceed to present our graph construction computed over a multi-scale blob and ridge decomposition (Section 3). We then describe our many-to-many graph matching technique (Section 4), our technique for identifying persistent parts (Section 5.1), and our technique for defining both attachment and abstraction relations (and their probabilities) (Section 5.2). We evaluate each stage of the pipeline using ground truth data, and explore the sensitivity of each step to changes in parameters (Section 6). Finally, we offer some conclusions (Section 8) as well as directions for future research.

2 Related Work

Many authors have attempted to learn categorical models from examples, and we highlight only a few due to space limitations. Constellation models have emerged as a popular representation for modeling general categories, such as motorbikes, faces, and cars

[1, 7]. Constellation models represent objects as configurations of local, appearance-based descriptors together with their spatial distributions. Learning these models can be accomplished robustly since appearance-based patches can be matched effectively and independently across training examples, therefore providing an efficient model bootstrapping step. The domain of human structural modeling from examples has also received considerable recent attention [8, 6, 5, 3]. Specific human structural tree models have been used in conjunction with efficient dynamic programming search methods by [8], while others [6, 4] assume an unknown (but tree-shaped) model structure and rely on ribbon detectors or temporal tracking [3] and clustering of body parts to recover a kinematic human representation using maximum weight spanning tree algorithms [9].

There are three critical differences between our approach and the above frameworks. The first is our use of generic shape features, as opposed to specific appearance-based features. Using appearance-based features not only constrains the training set to the same object exemplars, but yields a simple correspondence problem. Our generic features, in the form of ridges and blobs, are highly ambiguous, and cannot be tracked across training examples on the basis of their properties alone. This gives rise to the second major difference, whereby the context of a feature, i.e., the nature of its structural connections to nearby blobs, is critical to computing blob/ridge correspondence across training examples. The use of perceptual grouping to commit to this necessary structure *prior* to matching is in contrast to approaches in which the use of robust, local feature correspondences allows structural relations to be computed *following* matching. The final, and perhaps most critical difference, is our recovery of decompositional relations between features, allowing us to capture a coarse-to-fine representation of an object. Recovering such relations hinges on being able to match features many-to-many, as opposed to assuming a one-to-one feature correspondence. Without motion (of a single exemplar), appearance-based features cannot be matched many-to-many, due to their lack of generic structure.

3 Representing Qualitative Image Structure

We seek a decomposition of an image into a set of qualitative parts and attachment relations, and adopt the multi-scale blob and ridge decomposition proposed in [18]. Since blobs are generic features, they encode no appearance-specific information. Consequently, matching a blob in one image to a blob in another cannot be done on the basis of a blob's parameters, which include only a blob vs. ridge feature type, position (not translation or articulation invariant), orientation (not rotation invariant), ridge extent (not viewpoint invariant), and saliency. To overcome this tremendous ambiguity during matching, we need to draw on a blob's context, i.e., the structure of nearby blobs thought to be part of the same object. Specifically, we seek a set of edges that span features that are unlikely to be in close proximity by chance. Given our desire to describe objects at multiple levels of abstraction, spatial coherence and continuity dictate that, for example, when a coarse, elongated shape is decomposed into a set of smaller, elongated shapes, the latter will likely be attached end-to-end.

To set the edge weights, we must look ahead slightly to how they will be used at matching time. The many-to-many graph matching algorithm (to be described in more

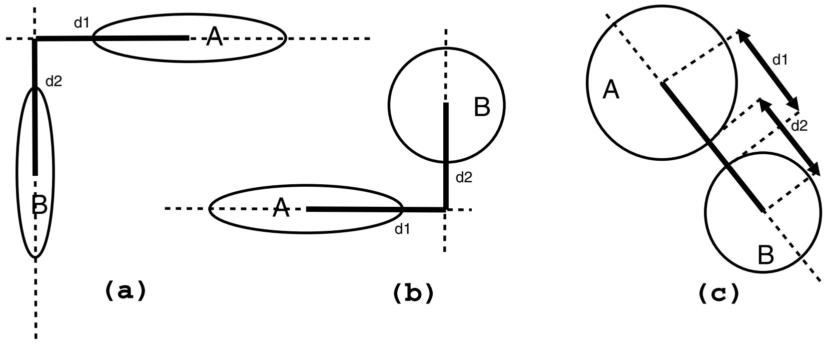

Fig. 2. Edge construction: (a) ridge-ridge; (b) ridge-blob; and (c) blob-blob. The total length of the bold lines represents the assigned edge weight between the two features in the graph.

detail later) first embeds the nodes of two graphs to be matched into two weighted point sets in Euclidean space. In this geometric space, a powerful many-to-many weighted point matching algorithm, the *Earth Mover's Distance (EMD)* [14], yields a solution which, in turn, specifies a many-to-many node correspondence between the original graphs. EMD will map (or "spread") a point from one graph to a collection of points from another graph if the members of the collection are in close geometric proximity. Therefore, if we want multiple parts at a finer scale in one graph to match a single part at a coarser scale in another graph, the edge weights (distances) linking the finer scale parts to be grouped must be relatively small.

A connectivity measure is computed for each pair of features, according to:

$$max\{d_1/major(A), d_2/major(B)\}, \tag{1}$$

where $major(X)$ is the length of the major axis of blob X. If this measure is greater than a threshold (whose sensitivity we evaluate in Section 6.1), the blobs are considered disconnected; if the measure is less than the threshold, an edge is inserted between the blobs whose weight is a function of d_1 and d_2, as shown in Figure 2. Due to scene clutter, the graph may have a number of connected components, representing multiple objects. We greedily choose the largest (in terms of number of nodes) connected component as a simple method for figure-ground separation, and discard the other components. Ultimately, a distance matrix over these remaining features is necessary to construct an embedding of the graph into a geometric space. To ensure that the distance matrix is invariant to part articulation, the distance between any two nodes is defined as the shortest path distance (along graph edges) between the nodes.

4 Computing Many-to-Many Blob Correspondences

Given an input training set of blob graphs, we compute a many-to-many matching between each pair of graphs. In the graph domain, this is an intractable problem that would require matching (perhaps connected) subsets of nodes in one graph to subsets of nodes in another. Our technique is based on a recent approach to this problem, proposed by Demirci et al. [16], which transforms the many-to-many graph matching problem to a many-to-many weighted point matching problem, for which an efficient algorithm exists. Given a shortest-path distance matrix encoding node-to-node distances, the algorithm employs a spherical coding technique to yield a low-distortion embedding of the nodes in a low-dimensional Euclidean space (we adopt a simpler, spectral embedding technique). The approach essentially throws out the original graph edges, and locates the points in space such that the Euclidean distances between points in the embedded space is close (with low distortion) to shortest path distances between nodes in the original graph.

The embedded points can now be matched many-to-many using the Earth Mover's Distance (EMD) under transformation [15]. If the points corresponding to one graph are viewed as piles of earth, while the points corresponding to the other graph are viewed as holes, the EMD algorithm computes the assignment of earth to holes that minimizes the amount of work required to move the earth to the holes. If we assume that mass is approximately conserved through levels of abstraction, then points should be assigned a weight that's proportional to the areas of their corresponding blobs. Returning to our "arm" example, the mass of the straight arm blob should roughly equal the sum of the masses of the broken arm blobs. The EMD under transformation is an iterative assignment/alignment process that quickly converges on a solution which can be mapped to a many-to-many node correspondence between the original graphs. In the following subsections, we provide the details on these steps.

4.1 Graph Embedding

A number of techniques are available for embedding the distance matrix into Euclidean space; examples include metric tree embedding [17], spherical codes [16], and ISOMAP [19]. We adopt a spectral embedding of a distance matrix computed in terms of shortest paths between nodes in a blob graph, similar to [19]. Each blob in the graph maps to a point which encodes the blob's embedded position and mass (blob area). The matching of two blob graphs can now be formulated as the matching of their embedded weighted point sets, in which a source point's mass can flow to multiple target points and a target point can receive flow from multiple source points. For our experiments, we embed the graph into a 2-D space. Though higher dimensional embeddings will result in a lower distortion and a more accurate many-to-many matching, the alignment computation may become underconstrained since more point correspondences will be needed. A 2-D space was chosen since it requires only 3 point correspondences for an affine alignment computation. Most of our exemplar images do share at least 3 feature correspondences. It is possible to adapt the dimensionality of the embedding space for each particular matching, the subject of future work.

4.2 Weighted Point Matching

The Earth Mover's Distance (EMD) algorithm under transformation [15] allows us to compute a many-to-many matching of the embedded points which, in turn, specifies

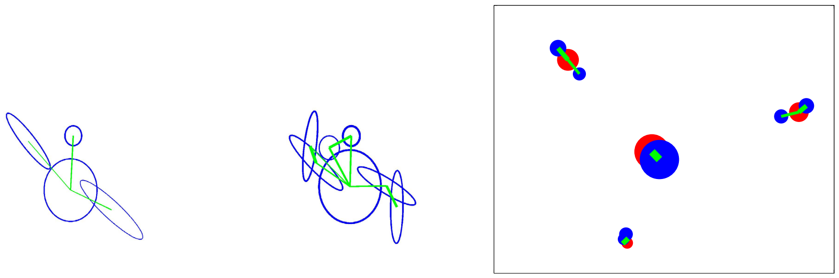

Fig. 3. Many-to-many matching of blob graphs using Earth Mover's Distance under transformation in embedded (Euclidean) space. Left two images show the detected blobs with green lines indicating blob connections and line width indicating edge strength (nonaccidental, proximity-based grouping strength). The right figure shows the embedded features (red for left image, blue for right image) after alignment, using the modified EMD under transformation. The flows are shown in green, with line width indicating amount of flow; note that since the blobs are well aligned, the flow distances are very small. The sizes of the circles correspond to the point masses (blob areas).

a many-to-many node correspondence between the nodes in the original graphs. The EMD is a global assignment problem, and assumes that the total masses of the two graphs are the same. However, with noise, occlusion, and clutter, this assumption is violated, and we must modify the algorithm to take a more local approach. Specifically, the mass of each feature in the first image is distributed among its nearby features in the second image in a greedy fashion, with both small flows and flows over large distances eliminated. If we compute the flows in the opposite direction, i.e., from the second image to the first image, the flows may be different, due to our greedy approximation. Augmenting the EMD cost function (the amount of work required to redistribute the mass) with terms that penalize for unmatched masses in the two images by adding the sum of untransferred masses in source nodes and the sum of unfilled masses in target nodes to the cost, we select the direction with minimum cost.

The flows associated with a given direction are used to compute an affine transformation between the corresponding point sets using a least-squares minimization of the sum of squared differences between the location of a point in the one set and a weighted (by the flows) average location of its matched points in the other set:

$$\sum_i \|(Pos(i) - T(\sum_j Flows(i,j) \times Pos(j)))\|^2, \tag{2}$$

where T is an $N-$dimensional affine transformation. In this approximation to the iterative **FT** (an optimal **F**low and an optimal **T**ransformation) algorithm [15], which alternates between computing the EMD flows and computing the affine transformation, the algorithm typically converges in 3-4 iterations. Figure 3 shows two blob graphs and the final matching, as computed by EMD under transformation. The final flow matrix computed by Algorithm 1 defines a direction of minimum cost. This matrix can be "inverted" to yield a consistent flow matrix for the opposite direction. These two matrices will play a key role in our procedure for extracting the parts in the final decompositional model.

Algorithm 1. EMD Under Transformation for Many-to-Many Matching of Two Weighted Point Sets

```
1: Compute the distance matrix $d(i,j) = \|P_{1_i} - P_{2_j}\|$.
2: Compute the $Flows$ matrix using the above distance matrix $d$.
3: repeat
4:    Compute the transformation $T$ that minimizes $\sum_i \|(P_{1_i} - T(\sum_j Flows(i,j) \times P_{2_j}))\|^2$.
5:    Transform each point $P_{2_j}$ from the second set with the computed transformation $T$.
6:    Compute a new distance matrix $d(i,j) = \|P_{1_i} - P_{2_j}\|$.
7:    Compute a new $Flows$ matrix using the new distance matrix $d$.
8: until the change in the $Flows$ matrix is small
9: Assign a cost to the computed $Flows$ matrix.
10: Return computed flow matrix $Flows$ and its cost.
```

5 Model Construction

Using the above feature matching framework, each pair of the P input exemplars is matched, resulting in $O(P^2)$ pairs of mass flow matrices (one per direction). Furthermore, each pair of flow matrices can be row normalized to 1, with each row entry indicating the fraction of mass flowing from the feature specified by the row to the feature specified by the column. These matrices are combined to form a single $N \times N$ matching matrix, M, where N is the total number of blobs in all of the exemplar images. M is a block matrix, where the (i,j)-th block stores the flows from features in image i to features in image j; diagonal blocks are identity matrices, reflecting the perfect one-to-one matching that would result from matching an image to itself.

The final decompositional model is derived from the matching matrix M and the original blob graphs. First, the one-to-one flows are analyzed to yield consistently appearing parts, i.e., parts that match one-to-one across many pairs of input images. Next, the many-to-many flows (M) between these extracted parts are analyzed to yield the decompositional relations among parts detected in the first step. Finally, the input blob graphs are analyzed to yield the attachment edges between the extracted parts. The extracted parts and their relations are used to construct the final decompositional model. The following subsections outline these steps in more detail.

5.1 Extracting Parts

Our goal in populating the final model is to select parts that occur frequently across many input exemplars, i.e., parts that match one-to-one. Recall that entry (p,q) in the matching matrix M contains the computed flow from blob p (in the image in which it was detected) to q (in the image it was detected) when the two images were matched; (q,p) contains the flow in the other direction. If both flows are close to 1.0, then the blobs are said to be in one-to-one correspondence. However, if part p or q is involved in a many-to-one decompositional relation, the flow in one direction will be less than 1.0.

By redefining both entries to be the minimum of the two flows, the entries representing one-to-one correspondences will retain their high values (close to 1.0) and the matrix becomes symmetric. Subtracting the entry from 1 turns the symmetric flow matrix into a symmetric distance matrix, setting up a clustering problem where clusters represent collections of nodes in one-to-one correspondences. Again, we draw on spectral techniques to embed the distance matrix in a low-dimensional space, and use the k-means[1] algorithm for clustering. The quality $[0,1]$ of the cluster is proportional to the "cliqueness" of the one-to-one matches among the members of the cluster. If a cluster is of sufficient size and quality, it becomes a node in the final decompositional model.

5.2 Extracting Relations

Two types of edges are used to link together the extracted parts (nodes). Decompositional edges are directed from one part to multiple parts, and capture the notion that a feature can appear alternatively as a set of component features, due to finer scale or articulation (or, in the reverse direction, a set of features can be abstracted to form a single feature). Attachment relations are the same nonaccidental proximity relations found in the blob graphs computed from the training images. An attachment edge is undirected, and implies that the blobs spanning the edge are connected. The many-to-many matching results (flows) between the extracted parts will be analyzed to extract the decompositional edges, while the attachment relations (in the original blob graphs) between the extracted parts will be analyzed to extract the attachment relations.

The K extracted parts represent clusters of matching blobs in the matrix M. For attachment relations, we compute the likelihood with which any two such parts not only co-appear in the images in which they were found, but are attached as well. If this likelihood of attachment exceeds a threshold, we define an attachment relation between the two extracted parts. The likelihood $[0,1]$ of attachment between parts i and j is defined by the $K \times K$ matrix PA (part attachment) as:

$$PA(i,j) = \frac{\sum_{p=1}^{P} \sum_{k=1}^{B(p)} \sum_{l=1}^{B(p)} [C_p(k) = i][C_p(l) = j] conn_p(k,l)}{\sum_{p=1}^{P} \sum_{k=1}^{B(p)} \sum_{l=1}^{B(p)} [C_p(k) = i][C_p(l) = j]} \tag{3}$$

where P is the number of training images, $B(p)$ is the number of blobs in training image p, $C_p(k)$ is the cluster that blob k in image p is assigned to, $[C_p(k) = i]$ is an indicator function whose value is 1 when $C_p(k) = i$ and 0 otherwise, and $conn_p(k,l)$ has value 1 if there is an attachment between blobs k and l in image p (and 0 otherwise). The expression captures the number of times blobs drawn from the two clusters were attached, normalized by the number of times blobs from the two clusters co-appeared in an image. Part attachment relations above a threshold T_{attach} are inserted into the final model. We found that $T_{attach} = 0.6$ worked well for our complete set of experiments, representing the condition that co-occurring blobs belonging to two different parts are connected in at least 60% of the input images.

[1] We first run k-means with a large value of k, resulting in over-segmented clusters. In a post-processing step, we reassign some blobs to more compatible clusters, remove noisy blobs from clusters, remove weak clusters, and merge similar clusters. The resulting procedure yields stable clusters that are less sensitive to the initial choice of k.

For decompositional relations, we restrict ourselves to one-to-many decompositional relations. A directed, one-to-many decompositional relation between one extracted part (parent) and a set of two or more extracted parts (children) must satisfy three conditions:

1. Most of the mass of the parent flows to the children.
2. In the reverse (many-to-one) direction, most of the mass of each child flows to the parent.
3. The children form a connected component, implying a spatial coherence constraint.

Testing the first two (flow) conditions requires a $K \times K$ part flow matrix, $PF(i,j)$, constructed by averaging the flows from all blobs in extracted part i's cluster to all blobs in extracted part j's cluster:

$$PF(i,j) = \frac{\sum_{k=1}^{N}\sum_{l=1}^{N}[C(k)=i][C(l)=j]M(k,l)}{(\sum_{k'=1}^{N}[C(k')=i]) \times (\sum_{l'=1}^{N}[C(l')=j])} \quad (4)$$

where N is the total number of blobs extracted from all images, $C(l)$ is the cluster that blob l is assigned to, and M is the $N \times N$ matching matrix. The expression represents the sum of all flows from blobs in cluster i to blobs in cluster j, normalized by the number of flows, yielding a mean flow. The entries in the matrix PF are in the range $[0, 1]$.

Given the part flow (PF) and part attachment (PA) matrices, Algorithm 2 extracts the part decomposition relations among the extracted parts in the final model. T_{child} (0.6) is determined empirically and reflects the degree to which a conservation of

Algorithm 2. Extracting Decompositional Relations

```
1: for $i = 1$ to $K$ do
2:     Find all parts $j \neq i$, s.t. $PF(j,i) \geq T_{child}$. Let $D$ be the set of all such parts, representing
       the potential children of $i$.
3:     for all subsets $D'$ of $D$ do
4:         Let $PA_{D'}$ be the upper triangular matrix of $PA(k,l)$, where $k, l \in D'$.
5:         The quality of the decomposition of part $i$ into the set $D'$ is $e^{-|1-\sum_{j \in D'} PF(i,j)|} \times$
           $min\{1, \frac{\sum_{k,l \in D'} PA_{D'}(k,l)}{|D'|-1}\}$ {The first term in the quality measure cost is high when
           most of the parent's mass flows to the children (and low otherwise). The second term
           encourages the children to form a connected component, where a connected component
           of $D'$ children implies at least $D' - 1$ attachment edges among them.}
6:     end for
7: end for
8: Choose decompositions whose quality exceeds $T_{decomp}$
```

mass constraint can be imposed between the children and their parent in a many-to-one mapping. A higher threshold, reflecting a stronger constraint, implies less blob over- or under-segmentation in the image domain in which the models are being learned. T_{decomp} is also set to 0.6, reflecting the fact that a parent distributes most of its mass to its children and that the children are attached (the product of the two terms needs to be larger than 0.6).

5.3 Assembling the Final Model Graph

The final model is a graph whose nodes represent the extracted parts and whose edges represent the extracted attachment and decompositional relations. Associated with each node is a saliency value, defined as the average of all the compatibility values of the blobs in a given cluster (defined in Section 5.1). The attachment relation between parts i and j has an associated likelihood, defined by $PA(i, j)$. The decompositional relation between a parent part and its constituent children has both an associated quality, defined by the algorithm above, and a probability reflecting how likely the decomposition is, i.e., the probability that the set of children will be observed in an image in lieu of the parent.

6 Experimental Results

We evaluate our model on a database of 86 torso images containing different individuals with different arm articulations; the blob graphs extracted from some of these images can be seen in Figure 1. Ground truth is provided for each input image in the form of a labeling of the extracted blobs in terms of the parts in an ideal torso decompositional model, shown in Figure 4; blobs that are not deemed (by a human observer) to correspond to a part on the ideal model are labelled as noise. This allows us to systematically evaluate each component of the system, including the detection of the blobs and attachment relations forming the input graphs, the many-to-many matching results, the detection of parts (clustering) that become the nodes in the final graph[2], and the attachment and decompositional relations that link the nodes together. Moreover, we can evaluate the sensitivity of each step as a function of any underlying parameters.

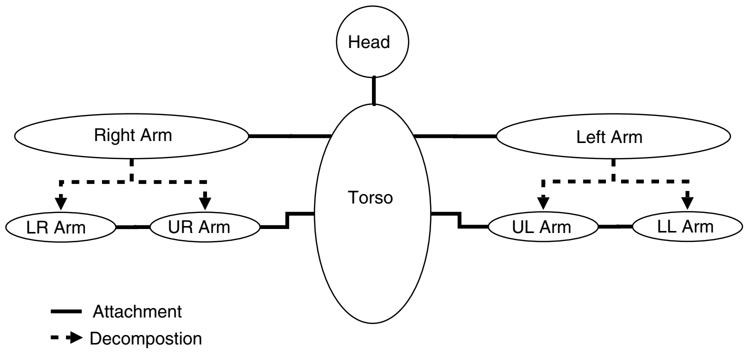

Fig. 4. The ideal torso decompositional model, representing ground truth for the experiments

[2] Since the clustering step is not deterministic (due to the random initialization of clusters), the clustering experiments, as well as all experiments that rely on the clustering results, were conducted 20 times for each value of the parameter being evaluated.

6.1 Evaluation of Input Blob Graphs

As mentioned in Section 1, the detection of blobs is a noisy process, resulting in over- and under-segmentation, spurious blobs, missing blobs, and poorly localized blobs. Given the ground truth labeling, we can evaluate the blob detection process. According to the part labels shown in Figure 4, the percentage of images in which the designated part was detected was: head (47%), torso (83%), left arm (50%), right arm (51%), left upper arm (37%), left lower arm (36%), right upper arm (40%), and right lower arm (37%). These relatively low percentages reflect the significant degree of noise in the detection of blobs (note that a straight arm and its two components cannot simultaneously appear). The attachment relations are governed by a single proximity threshold. Large threshold values cause all blobs to be attached and thus produce false positive attachment relations among parts, whereas small threshold values create sparse graphs with false negative attachment relations among parts. Figure 5(a) shows the error in individual attachment, representing the sum of the SSD error in the attachment matrices of all exemplar blob graphs.

6.2 Evaluation of Many-to-Many Matching

The error in the many-to-many matching component is computed by finding the sum of the SSD errors in the flow matrices of each pair of exemplars. Given the optimal proximity threshold, our matching algorithm yields a 9% error based on an element-by-element comparison of the computed matching matrix M to the ground truth data.

6.3 Evaluation of Part Extraction

The error in the clustering step comprising part extraction is a function of two parameters. The cluster error is computed by first finding the best cluster for every part in the ideal model. Given a labeling of each image in terms of the ideal model, we can then compute both precision and recall for each model part. The minimum (worst-case) of the precision and recall values is averaged across all clusters and then inverted to yield a final error measure. Figure 5(b) plots smoothed error as a function of embedding dimension. From the figure, we conclude that the choice of the embedding dimension is not critical. The second parameter is k, representing an upper bound on the true number of clusters. Figure 5(c) plots smoothed error as a function of k (maximum number of clusters). Since the minimum is rather shallow, our algorithm is not very sensitive to the choice of k.

6.4 Evaluation of Edge Extraction

Errors in the extraction of part attachment and part decomposition edges are computed by first finding the correspondence between the ideal model (ground truth) parts and the computed clusters, from which the SSD errors in the attachment and decomposition edges can be computed. Since the correspondence between ground truth and computed clusters is not necessarily one-to-one, and since a computed cluster does not necessarily correspond to any ground truth cluster, an additional error term is added to account for

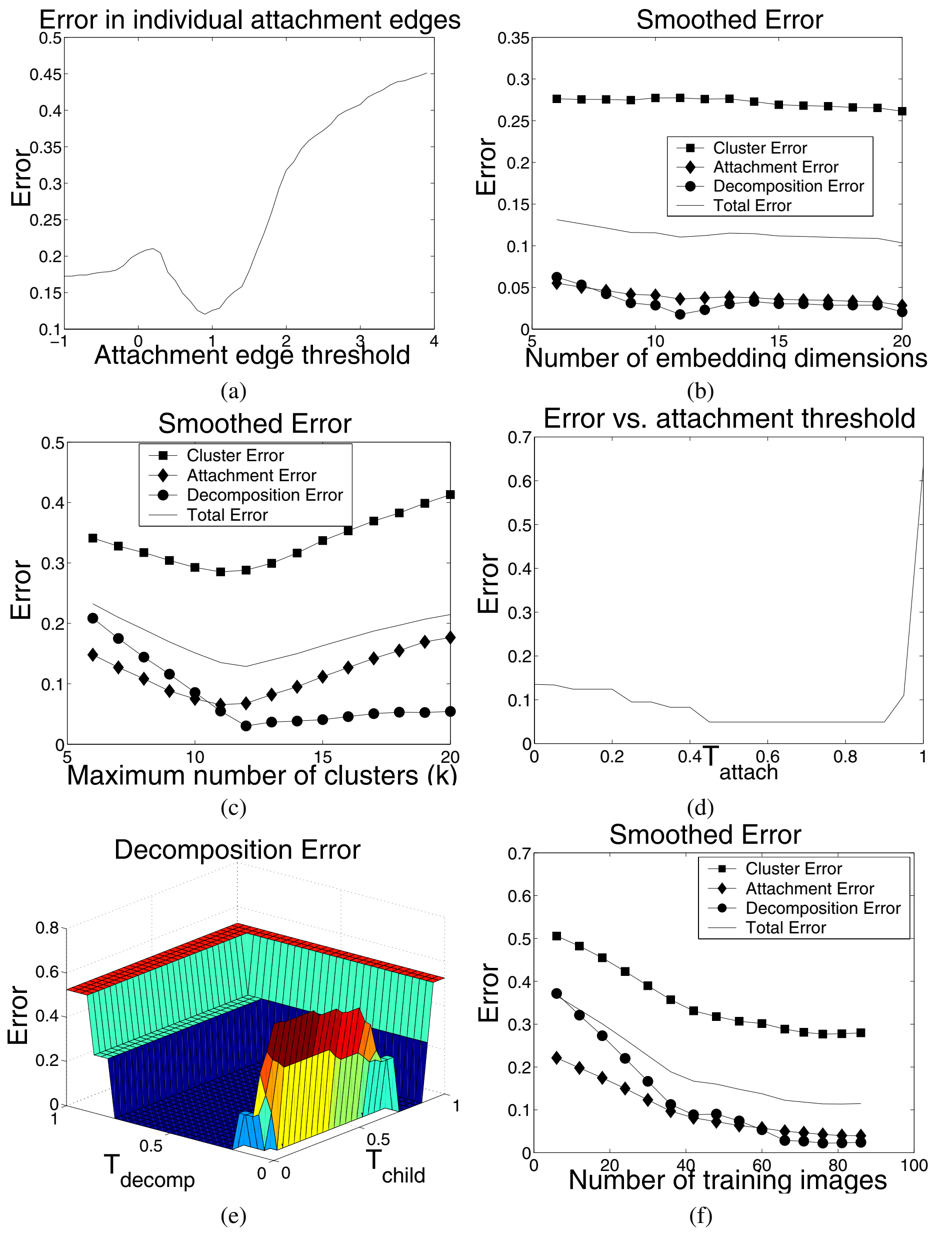

Fig. 5. Evaluating the Model: (a) Input attachment relation error as function of proximity-based grouping threshold; (b,c,f) The four curves represent clustering error, recovered attachment edge error, recovered decomposition edge error, and final decompositional model error as a function of dimensionality of embedding, the upper bound k on the number of putative clusters, and training set size, respectively; (d) Recovered attachment edge error as a function of T_{attach}; (e) Recovered decomposition edge error as a function of T_{child} and T_{decomp}

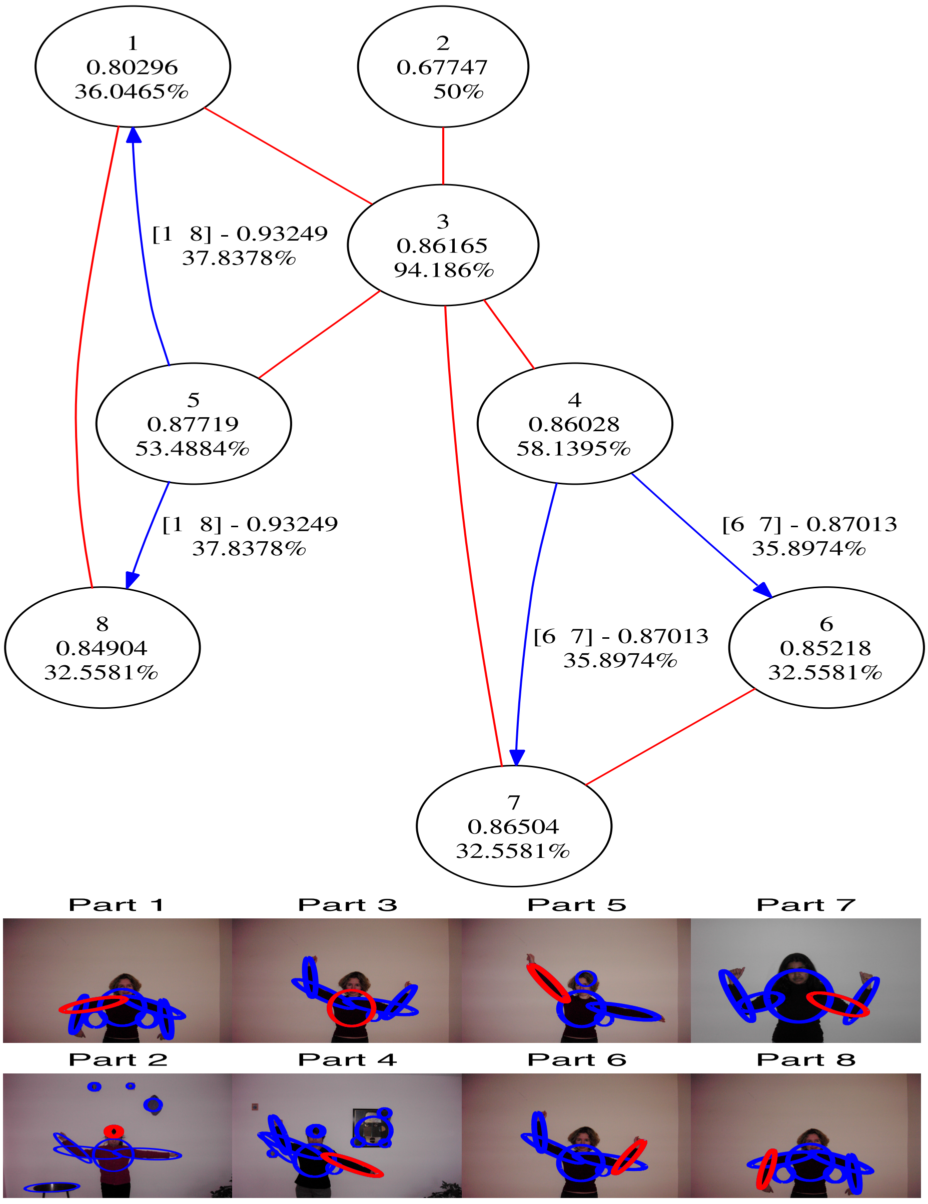

Fig. 6. Final decompositional model obtained by our system on 86 input images. Red edges indicate part attachment, while blue edges indicate part decomposition (or, inversely, abstraction). The values on the decomposition edges specify the children (square brackets), the quality of the decomposition, and the probability of the decomposition. The top number inside a node is its part number, the middle number is its cluster quality, and the bottom number is the probability of occurrence of the part. At the bottom is one example image for each part (shown in red), sampled from its cluster. The model not only captures the correct attachments between parts, but also captures the decompositional relations between each arm and its constituent subparts.

the dissimilarity in the number of edges between the ground truth model and the final recovered model. The error term is the difference in the number of edges relative to the maximum number of edges in the ground truth and retrieved models. Figures 5(d) and 5(e) show the error in attachment edges, as a function of the threshold T_{attach}, and decomposition edges, as a function of the thresholds, T_{child} and T_{decomp}. The same clustering results are used throughout these two experiments. As can be seen from the figures, there is a range of thresholds that results in good attachments and decomposition edges.

6.5 Evaluation of the Final Model

From the above experiments, we determined optimal values for the different parameters and manually entered them into the system. In our final experiment, we evaluate the error of the final decompositional model as a function of the size of the input training set. The error is defined by averaging the clustering error, the recovered part attachment error, and the recovered part decomposition error. Figure 5(f) shows the smoothed final model error and its three components as a function of training set size. It can be clearly seen that the errors decrease as the number of training images increases. The automatically generated model (for the full training set) is shown in Figure 6. The recovered model is isomorphic to the ideal model, reflecting our algoithm's ability to correctly recover a decompositional model from noisy examples.

7 Limitations

The many-to-many matching component of our framework can handle spurious noise in the form of missing features and small extraneous features. However, the presence of a large noise feature or large occluder may result in incorrect assignments during the EMD stage, for it may draw too much mass from correct features (if it's a "hole") or flood too many correct features (if it's a "pile"). The impact of such features can be minimized by assigning additional properties to the features, such as shape or appearance, and then using these properties as constraints in computing the mass flows during EMD. Another important limitation arises from the fact that positional information of the blobs is lost during the embedding. Although the use of distance between features yields articulation invariance, it also means that feature ordering may be lost, as exemplified by a head on one torso matching a similarly sized noise blob on the bottom or side of another torso. Again, the role of additional blob properties as constraints may help to alleviate this problem.

8 Conclusions and Future Work

We have presented an algorithm for automatically recovering a decompositional, generic shape model from examples; parts of the model can be represented at different levels of abstraction – an important representational goal originally proposed by Marr. Two important challenges face this task: 1) the inherent ambiguity in generic shape features, such as ridges and blobs; and 2) the need, due to articulation, scale, and segmentation error, to match such features many-to-many. By imposing a graph-based perceptual grouping on the parts, we provide the structural context necessary to match ambiguous parts many-to-many. Our algorithm requires a number of parameters, and we have established the relative insensitivity of the results to changes in the parameters. We have demonstrated the approach on recovering a decompositional torso model from example images of different subjects. Although our features and grouping rules are tuned for articulated objects, such as humans, the framework could be applied to other features and grouping rules that are more suitable for other categories. The correctness of the recovered model as a function of the size of the training set has been evaluated with respect to ground truth. Preliminary results are very encouraging, and current efforts are aimed at recovering more complex models having a higher incidence of decompositional structure. Moreover, we seek to augment the recovered model with both node and relation constraints, derived from the input data. Finally, we plan to apply machine learning techniques to recover optimal perceptual grouping parameters, and develop object recognition techniques adapted to part-based decompositional models.

Acknowledgements

The authors would like to thank Lars Bretzner for his blob extraction software. Fatih Demirci and Ali Shokoufandeh provided both feedback and support in the development of the many-to-many graph matching module, and also provided the EMD software. We are very grateful for their assistance. The authors would also like to thank Allan Jepson for his thoughtful feedback on the work. The support of NSERC, the NSF, CITO, and PREA is gratefully acknowledged.

References

1. Fei-Fei, L., Fergus, R., and Perona, P., A Bayesian Approach to Unsupervised One-Shot Learning of Object Categories, ICCV, Nice, 2003.
2. Lazebnik, S., Schmid, C., and Ponce, J., Semi-Local Affine Parts for Object Recognition, BMVC, 2004.
3. Song, Y., Goncalves, L., and Perona, P., Unsupervised Learning of Human Motion, *IEEE PAMI*, Vol. 25, No. 7, 2003.
4. Ramanan, D. and Forsyth, D.A., Using Temporal Coherence to Build Models of Animals, ICCV, 2003.
5. Ramanan, D. and Forsyth, D.A., Finding and Tracking People From the Bottom Up, IEEE CVPR, 2003.
6. Ioffe, S. and Forsyth, D.A., Human tracking with mixtures of trees, ICCV, 2001.
7. Fergus, R., Perona, P., and Zisserman, A., Object Class Recognition by Unsupervised Scale-Invariant Learning, IEEE CVPR, 2003.
8. P. Felzenszwalb and D. Huttenlocher, Efficient Matching of Pictorial Structures, IEEE CVPR, 2000.
9. Chow, C.K., Liu, C.N., Approximating discrete probability distributions with dependence trees, *IEEE Trans. Info. Theory*, IT-14, No.3, 1968, pp. 462-467.
10. Brooks, R.A., Model-Based Three Dimensional Interpretations of Two Dimensional Images, *IEEE PAMI*, March 1983.

11. Marr. D. and Nishihara, H.K., Representation and recognition of the spatial organization of three dimensional shapes, Proc. of Royal Soc. of London, 1978.
12. Jepson, A.D., Fleet, D.J., and Black, M.J., A layered motion representation with occlusion and compact spatial support, ECCV, 2002.
13. Viola, P., Jones, M.J., and Snow, D., Detecting Pedestrians Using Patterns of Motion and Appearance, ICCV, 2003.
14. Rubner, Y, Tomasi, C., and Guibas, L.J., A metric for distributions with applications to image databases, ICCV, 1998.
15. Cohen, S. and Guibas, L.J., The Earth Mover's Distance under Transformation Sets, ICCV, 1999.
16. Demirci, M.F., Shokoufandeh, A. Dickinson, S., Keselman, Y., Bretzner, L., Many-to-Many Feature Matching Using Spherical Coding of Directed Graphs, ECCV, 2004.
17. Matoušek, J., On Embedding Trees into Uniformly Convex Banach Spaces, *Israel J. of Mathematics*, Volume 237, 1999.
18. Lindeberg, T. and Bretzner, L., Real-time scale selection in hybrid multi-scale representations, Proc. Scale-Space'03, 2003.
19. Tenenbaum, J.B., de Silva, V., and Langford, J.C., A Global Geometric Framework for Nonlinear Dimensionality Reduction, *Science*, 290 (5500), 2000.