LEARNING DECOMPOSITIONAL SHAPE MODELS FROM EXAMPLES

by

Alex Levinshtein

A thesis submitted in conformity with the requirements
for the degree of Master of Science
Graduate Department of Computer Science
University of Toronto

Abstract

Learning Decompositional Shape Models from Examples

Alex Levinshtein

Master of Science

Graduate Department of Computer Science

University of Toronto

2005

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.

Acknowledgements

It is my pleasure to give thanks to the many people who made my thesis possible.

Foremost, I would like to thank my supervisor, Sven Dickinson, who gave me many hours of his time discussing my work. Our discussions not only gave rise to new ideas, some of which are implemented in this thesis, but also gave me a true appreciation for the difficult problem of generic recognition and the motivation to study it. Sven also provided invaluable help with writing this thesis and various other documents I had to write during my Masters career.

I am also very grateful to Cristian Sminchisescu, who co-supervised me with Sven and participated in our discussions. He greatly helped with the machine learning aspect of the project, introducing me to recent and relevant work in the field. I greatly appreciate the help of Ali Shokoufandeh and his student Fatih Demirci from Drexel University for their discussion and help with the code. My gratitude goes to Allan Jepson, my second reader, for providing excellent comments on the thesis, and to Avner Magen for the interesting discussion about some of the technical details. I also would like to thank my fellow graduate students in computer vision for our interesting discussions about our work.

Finally, I am grateful to Suzanne Stevenson for introducing me to Sven, and to all other professors who introduced me to the field of artifitial intelligence.

Contents

Chapter 1

Introduction

The early generic object models proposed by researchers such as Marr and Nishihara [12] and Brooks [13] 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 (Figure 1.1). 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, 9] 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 mo-

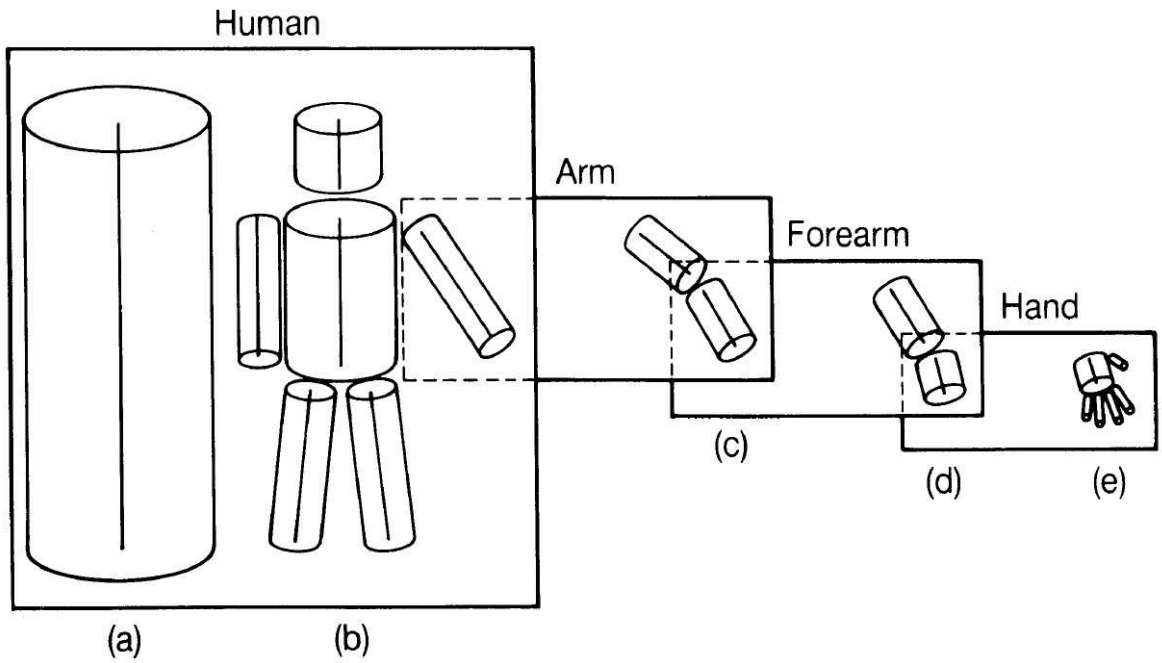


Figure 1.1: Hierarchical models proposed by Marr for the purpose of categorical modeling (taken from [12]).

tion and persistent appearance [5, 6, 15, 3, 4]. Global detectors that combine both motion and appearance have also been successfully applied to pedestrian tracking tasks [19].

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 neighbourhoods 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. Note that a hierarchy can arise for different reasons. In Marr's work, such a hierarchy arises from examining the same object at different levels of abstraction. In our experiments, a hierarchy arises due to object articulation. Though the causes are slightly different, the end result is the same – the same part of the object can be represented by different numbers of features.

We begin by applying a multi-scale blob and ridge detector [27] 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 1.2, 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 1.2, 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

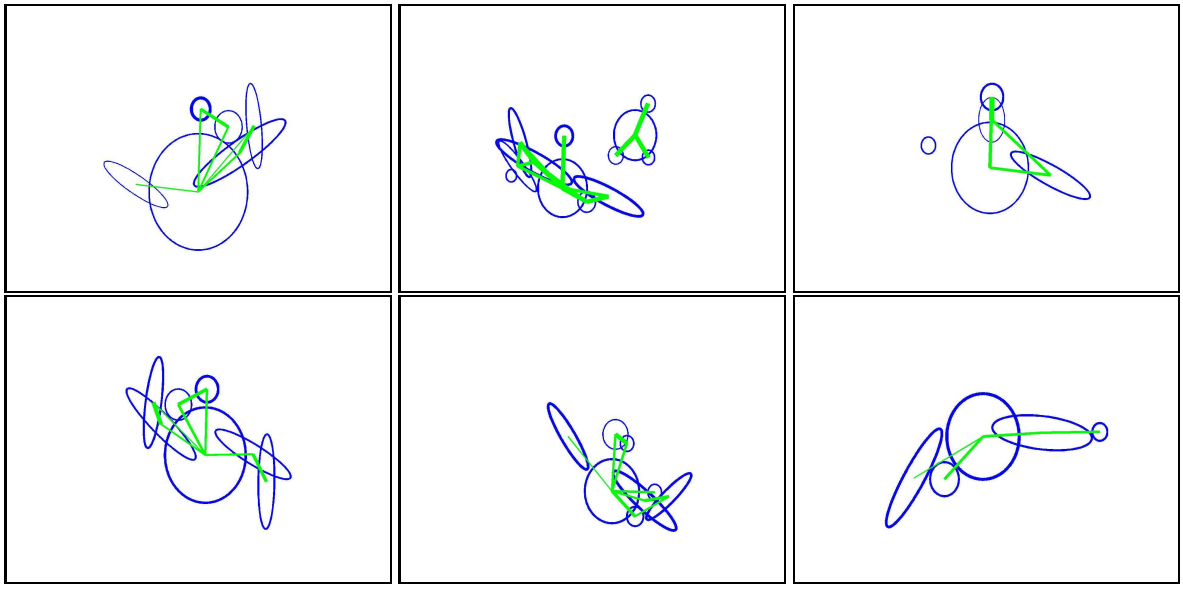


Figure 1.2: Blob graphs extracted from a set of images, each containing the upper body of a different person (with different clothing). The high level of feature abstraction comes at the cost of increased segmentation errors in the form of under- and over-segmentation, missing features, and spurious 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.

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 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 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 (Chapter 2) and an overview of our system (Chapter 3). We proceed to present our graph construction computed over a multi-scale blob and ridge decomposition (Chapter 4). We then describe our many-to-many graph matching technique (Chapter 5), our technique for identifying persistent parts (Chapter 6.1), and our technique for defining both attachment and abstraction relations (and their probabilities) (Chapter 6.2). We evaluate each stage of the pipeline using ground truth data, and explore the sensitivity of each step to changes in parameters (Chapter 7). Finally, we offer some limitations and conclusions as well as directions for future research (Chapters 8,9).

Chapter 2

Related Work

Object recognition contains several subproblems, including modelling, indexing and identification. Though some researchers address all these issues in a recognition framework, we will concentrate only on automatic model acquisition. Object recognition can be roughly divided into two tasks: identification (or low-level recognition) and categorization (or high-level recognition). The first involves recognizing specific objects. The second involves recognizing an object from a large category, such as people or animals. We are trying to learn an appropriate model for an object category. During categorical model acquisition, it is important to construct a sufficiently general model to accomodate all the objects in a given category, yet a sufficiently restrictive model to reject outliers from other categories. As mentioned in Chapter 1, Marr and his peers proposed generic, hierarchical models; however, the tools to build such models automatically did not exist then. Although today, with the help of new machine learning techniques, there is growing research in automatic model acquisition, most acquired models are appearance-based and are not geared toward categorical recognition. In this paper, our focus will be on recovering decompositional generic shape models.

Model acquisition is only part of the object recognition problem. As mentioned previously, some authors attempt to address all components in an integrated framework. Though our model will eventually be used for recognition in the real world, where indexing and matching are

important, we leave such concerns for future research. In terms of model construction, some integrated systems learn models dynamically (on-line). For example, given region segmented scenes as input, Xu et al. [38] group regions based on the quality of the template matching results, forming composite nodes. The model evolves by updating the composite nodes as new templates become available. Segen [44] provides a way of constructing graph-based models by representing feature relations (and not features) as nodes. The modeled relations can be n-ary, and are not restricted to be pairwise. An efficient algorithm is provided to dynamically learn such models as new input graphs become available. On the other hand, our goal during model construction is to obtain natural high-level parts and relations between them. Our model will be built off-line with pairwise decomposition and attachment relations. Thus, we will concentrate only on model recovery, with no concern for issues such as dynamic model acquisition or modeling more complex relations.

The main choices facing the model acquisition community are the type of the model and the features used in its construction. A model can be part- or non-part-based. Non-part-based, or global models, try to represent an object as a whole, either modelling its appearance or shape. Part-based, or local models, try to model the object as a collection of parts, which again can be appearance- or shape-based.

A lot of early work in modeling relates to global models. Such work includes PCA models, such as eigenfaces [17], which are based on the appearance of the object, and active shape models, originally proposed by Cootes et al. [18], which model the contour of the object. In PCA models, the objects are linearly projected to a lower dimensional space. The model consists of several basis objects that represent the largest eigenvectors of the space of all the training exemplars. In active shape models, the contour of the object is modeled with a snake (a sequence of points) in case of 2D modeling; or with a mesh in the 3D case. The model is obtained by registering the points from each exemplar together. A mean shape for the snake or the mesh is obtained together with information about potential variability in the position of each point.

Several extensions to the original active shape models were proposed. Duta et al. [39] focus on the robust registration of the exemplars, and obtain an average contour of an object. Maurel and Sapiro [37] extend active shape models to deal with video sequences as input. This adds an additional problem of temporal alignment. Ueda and Suzuki [42] provide a multiscale representation for contour description and match contours at different scales. If a portion of one contour contains more detailed information than another contour, the final model contour will contain the coarser representation. Though global models are a highly researched field, they do not handle occlusion or noise very well. Such models require that only the modeled object be present in the training images. Therefore, we will concentrate on part-based models in our work.

Part-based models have been shown to contain many advantages over global models. Part-based models store the object's description as a collection of several components, and recognition is possible even if some parts are occluded or missing. The parts themselves often contain statistical information about their parameters, allowing for robust recognition. The number of parts can range from hunderds to tens or less. Some part-based models also contain the relations between parts, such as distance or orientation. Moreover, the topic of finding the best relational structure for the model has gained considerable attention, as it restricts the recognition stage even further and provides a solution for dealing with the exponential number of matches of a new exemplar to the model. The work in part based-modeling can be divided into several areas of interest.

2.1 Type of input

A lot of part-based model acquisition work goes hand in hand with the tracking field. There is a dual advantage in using part-based models for tracking purposes. On the one hand, part-based models provide more robust tracking that can handle occlusion and missing parts. One the other hand, since tracking deals with video sequences of moving exemplars as its input, it provides a

simpler way of correctly matching the images and thus simplifying part correspondence. The latter consequence gives a significant advantage to the use of tracking for automatic model acquisition.

The area of human tracking has gained particular attention in the part-based modelling field [4, 5, 6, 15]. Ramanan and Forsyth [4, 5] provide a full framework for learning a part-based model together with building a robust tracker, and apply their method to tracking articulated objects such as humans and animals. They learn both the average appearance of each part, and relations between parts which can encode either distance or articulation. Their framework can be used both as a tracker without any prior knowledge of the object in question, and as an object recognition system that uses the model obtained through the tracking process.

Ioffe and Forsyth [6] emphasize the fact that due to occlusion, missing parts and other factors, the same object may consist of different parts in different images. If each such configuration is a tree, the final model is a mixture of trees. The authors propose an efficient way to store and search such a mixture. Jepson et al. [15] deal with the occlusion of parts by adding a notion of depth (a numbered layer based on the distance from the camera) to each part, which is not present in other methods where only the location is modeled in most cases. The number of parts is adjusted automatically based on visibility and motion data. Moving parts that are visible for sufficient time are included in the model. Though motion data is very useful, it generally consists of a specific moving object. Part-based shape models can be constructed given a motion sequence of a particular exemplar. However, we try to solve the harder problem of obtaining such a model from different exemplars from the same category whose appearance may be dissimilar.

Though part-based models have received considerable attention in the tracking community, their use is not restricted to tracking. Based on the work of Weber, Welling and Perona [7, 8], Fergus et al. [9] obtain a constellation model (part-based model with pairwise relations between parts) from several training images, where both appearance and shape (pairwise relations between parts) are being learned. Since the authors still use appearance-based features,

they can robustly match features from each exemplar without motion information. They use the framework to recognize objects such as cars and motorcycles. Fei-Fei et al. [1] extend the work to be able to learn a new category from a single or a few training images, relying on statistical information from the already learned categories. Felzenszwalb and Huttenlocher [10] show how to efficiently match such constellation structures to image data. Given that the model is a tree, the authors propose an efficient dynamic programming algorithm to match the model with the image. The field of constellation models seems closer to our goal. However, all of the above work relies on the ability to match image features independently. Though it is possible for appearance-based features, it is not possible for the types of shape features we are using, since blobs are much more ambiguous than appearance-based features.

2.2 Parts of the model

All part-based models try to cluster image features into coherent model parts. Features that are used for the model can range from generalized cylinders [12] proposed in the early days of computer vision (Figure 2.1) at one end, to appearance-based patches, such as the ones proposed by David Lowe [16], at the other end of the spectrum. Today, appearance-based features are a very popular choice since they are robust to many deformations and are relatively easy to match. Moreover, such features can be matched independently without the need for examining the context of each feature.

Some appearance-based modeling papers exploit this last fact and emphasize the feature extraction stage. Lazebnik et al. [2, 36] identify regions that are affinely invariant throughout the training set. Each such region becomes a part in the final model, consisting of several affine invariant appearance descriptors that are rigidly arranged. Dorko and Schmid [35] not only cluster features into parts but concentrate on finding suitable parts for best recognition results based on a measure of the classification rate for each part. Appearance-based features restrict the matching to objects with similar appearance. This restriction is non-problematic for mod-

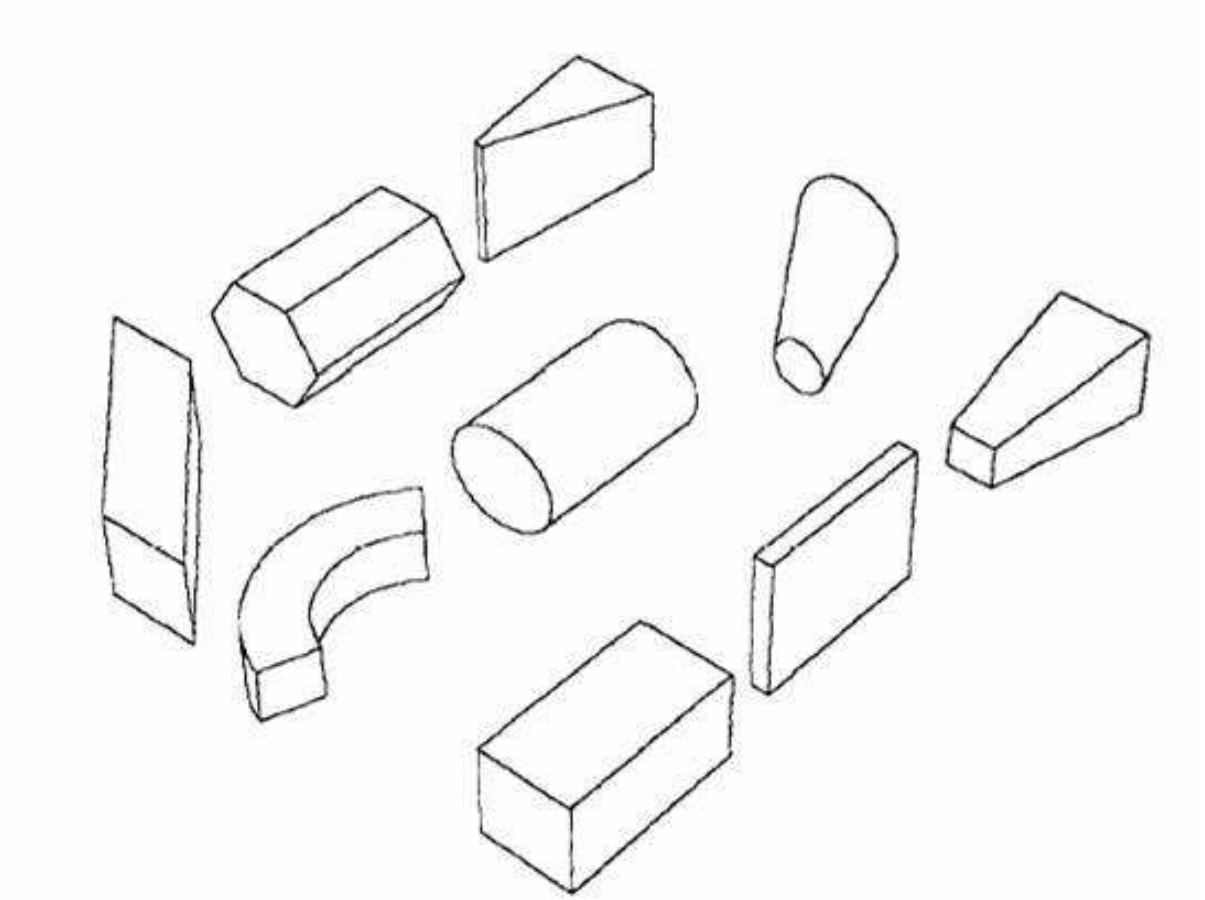


Figure 2.1: Geons. Generic features proposed in the early computer vision days [14].

eling of a single object or a category of objects with similar appearance, but many categories have no appearance similarity among their members. Whereas the automatic recovery of volumetric parts is still very ambitious, other descriptions for categorization are becoming popular. Such descriptions include segmented regions, an object's silhouette, and its decomposition into blobs (circles and ellipses encompasing different regions of the object). In our work, we choose to represent an object's parts by blobs and ridges ([27]). Though such features allow for a more generic description, the matching becomes far more ambiguous.

2.3 Model structure

Most part-based models contain spatial relations between parts. The final models usually have parts as nodes of a graph and relations as edges. Constellation models do not restrict the final graphs to be trees, but since many objects (such as humans, for example) exhibit independent behaviour among their parts, such independence is usually modeled. This modeling of independence greatly helps during recognition time, since it prunes out the exponential number of potential matchings.

In 1969, Chow and Liu [11] showed that a tree-based approximation to an arbitrary distribution represented as a graph could be obtained optimally using a Minimum Weight Spanning Tree algorithm. This is a remarkable fact due to the exponential number of possible trees. This method is very useful for recovering part dependencies during object modelling. For example, taking an articulated object such as a human, a given part is conditionally independent of other parts given the information about the parts that are connected to it by a joint. Taycher et al. [34] use this method to obtain a tree structure given feature correspondences from the different frames of a video sequence. Anguelov et al. [33] have several meshes with point-wise correspondences. As opposed to the previously mentioned work that deals with mutual information among parts, they recover the dependence structure in a different manner. First, they partition the points into rigid parts. Since all the points belonging to a given rigid part undergo the same rigid transformation, points that agree on their transformation get assigned to the same part. Secondly, they recover the skeleton without the tree-like structure assumption that was used before. Points that agree with several transformations are likely to be on the boundary between parts and are used to compute the skeleton.

In our work, feature matching is only possible through the initial commitment to a perceptual grouping stage. Unlike the previously mentioned methods that recover relations only after the parts are extracted, we impose a relational structure on the parts of individual exemplars to facilitate our exemplar matching stage. Since such a commitment has already been made, relations in the final model can be found by collecting relational statistics among the different

exemplars in the training set.

2.4 Hierarchical models

In most part-based models, the relations among parts are spatial, such as articulation of parts or distance between them. All the features composing the parts of the model have been observed in one image or another in such a case, and the parts are not ordered in any way. However, the original models proposed by Marr [12] contained hierarchical relations. Parts may be ordered in a hierarchy based on their scale or generality. A parent part may be an abstraction of its children. In the early days of computer vision, hierarchical models were constructed manually. Since then, however, some researchers have attempted to recover such models automatically.

Connell and Brady [43] proposed learning such models automatically using semantic nets. Nishida and Mori [45] build a structural model for recognizing hand-written digits. Strokes are grouped based on the quality of the matching, making the final digit model into a hierarchy, where the leaves correspond to indiviual strokes and the inner nodes correspond to stroke groupings. After recent advances in machine learning, many authors began approaching the problem from a Bayesian network perspective. Bouchard and Triggs [31] use a constellation model; however, they add a hidden layer of parts above the layer of parts extracted from the model. Such hidden variables correspond to meaningful collections of image parts and provide an abstraction layer over the parts that were actually observed in an image. These high-level parts were not observed in the image, yet they represent an important abstraction that is useful during learning and recognition. Utans [32] also tries to learn a hidden layer in his model. The low-level variables correspond to the observed features, whereas the higher levels of the Bayesian network represent conditionally independent groupings of the image parts. Utans applies his work to digit modeling.

Others approach the problem from a graph matching perspective. Keselman and Dickinson [25] compute the largest common abstraction of several exemplars represented as graphs

through a process of merging nodes. Each part (region) in the final model represents a subset of connected regions in each exemplar whose merging together yields a set of qualitatively similar shapes. Jiang et al. [26] introduce the notion of a median graph, which is graph whose sum of squared distances to a set of input graphs is minimal, where graph distance can be defined as graph-edit distance or any other graph distance measure. They propose an algorithm for efficiently computing median graphs based on graph-edit distance. In our work, we will also concentrate on modeling hierarchical relations. As mentioned previously, we commit to a relational structure for each individual exemplar. Therefore, since each of our input exemplars is a graph, we also approach the problem of hierarchical model recovery from a graph matching perspective.

2.5 Summary

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, as used by most part-based models these days. 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 (grouping features based on nonaccidental relations) 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.

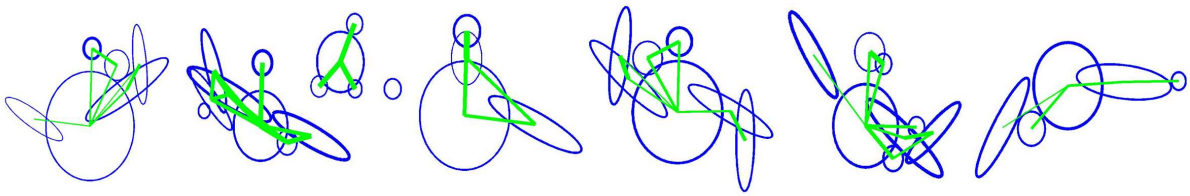
Chapter 3

Overview

The main objective of this work is to construct a decompositional shape model from exemplar objects of a given category. For example, consider the task of constructing a model of a human torso given exemplar images of torsos of different people, as shown in Figure 3.1(a). This section provides an overview of our approach to recovering such a model.

Most authors in the appearance-based modelling community describe the parts of their models as clusters of appearance based features. In previous chapters, we have argued that such an approach is unsuitable for generic modelling as appearance is not necessarily a persistent cue among different exemplars from the same category. Shape seems to be a more persistent cue in categorical model acquisition. We therefore choose features, specifically blobs, that encode only shape information. The final parts of our model will be clusters of such features from the different exemplars provided during the training stage. The final model will also contain relational links between the parts. As opposed to most previous work, our system supports two types of relations: attachment and decompositional relations. Attachment relations will indicate a spatial attachment between pairs of parts that were frequently observed to be attached. Decompositional links will indicate a relation where one part may be split into several others. Such a relation means that a given part of the object was observed at different levels of granularity among the different exemplars. An example of an ideal final decompositional shape

(a)



(b)

Figure 3.1: (a) Sample input images for constructing the torso model. (b) Extracted blobs from sample exemplar images. Green edges indicate attachment based on our perceptual grouping rules. The width of the edges indicates the strength of attachment.

model for the human torso images (from Figure 3.1(a)) is shown in Figure 3.2.

Our system is a pipeline of four primary steps, as shown in Figure 3.3. Initially, blobs are detected in every exemplar image. Since blobs cannot be matched in isolation, as was argued in previous chapters, we construct a graph for every input exemplar. The nodes in the graph correspond to the recovered blobs and the edges encode perceptual grouping relations, which capture non accidental attachment relations between blobs. Next, given an input blob graph for each exemplar, each pair of exemplars is matched many-to-many. These many-to-many matching results are used in the final two stages to construct a decompositional shape model.

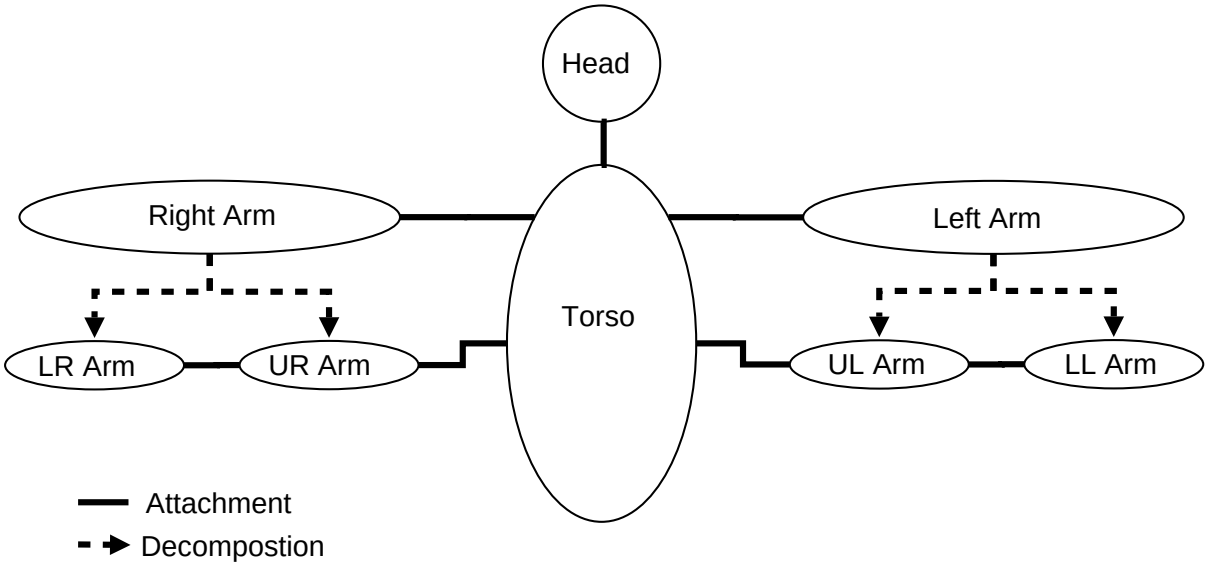


Figure 3.2: Ideal decompositional torso shape model.

During the part extraction stage, the matching results are used to group consistently matching features into clusters which will be the parts of the model. The final stage adds attachment and decomposition links to the model. Attachment links are added based on statistical information about blob attachment for each pair of parts. Decompositional links are added based on statistical information about many-to-many matching results between pairs of exemplars.

During the feature detection stage, we use the approach of Lindeberg and Bretzner [27] to efficiently recover blobs and ridges through filter responses at different scales. The results of the blob detector are post-processed, resulting in the elimination of fully included blobs and blobs with weak support. Extracted blobs from a few sample images of human torsos can be seen in Figure 3.1(b). Since many-to-many graph matching is intractable, we transform the problem into a weighted point matching problem (known as Earth Mover's Distance, or EMD), where the goal is to match two sets of weighted points lying in the same Euclidean space. The result of the EMD algorithm consists of mass flows from features of one exemplar to the features of another exemplar that indicate the many-to-many matching results. The flows store the portion of each feature in the first exemplar that matches a feature in the second exemplar. For the EMD algorithm, for every exemplar, each blob is assigned a mass and each

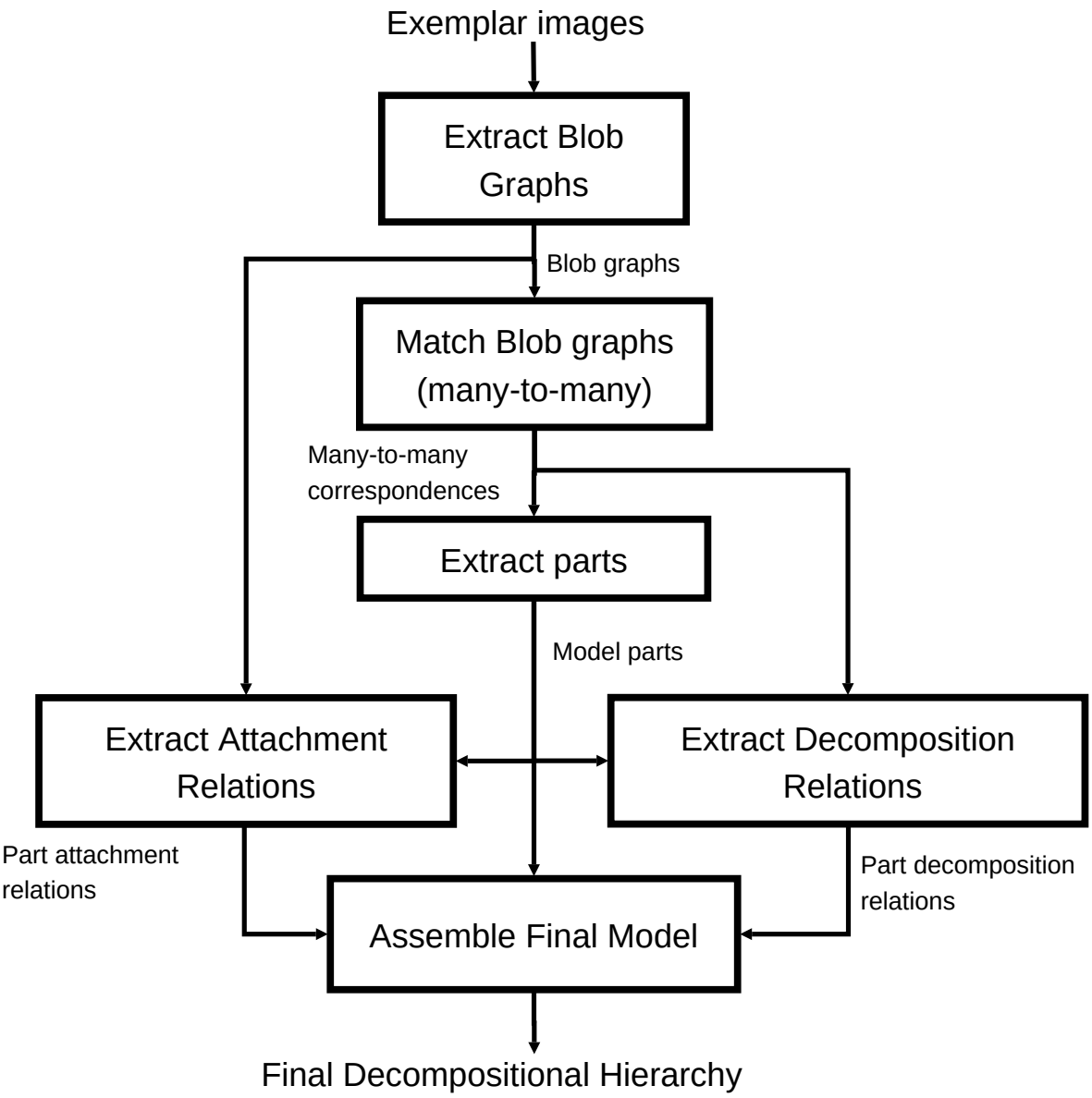


Figure 3.3: Different stages in our system.

pair of blobs is assigned an edge weight (the motivation behind this choice will be explained in Chapter 4). The mass assignment corresponds to the area of the blob, which intuitively describes the size of the corresponding part. The edges between blobs are formed based on joint connectivity, with large weights assigned to blobs that were deemed disconnected by the perceptual grouping stage. Sample results of this stage applied to human torso images can be seen in Figure 3.1(b).

The feature matching stage uses the masses and the edge weights to embed the features in a Euclidean space and match them many-to-many using an EMD under transformation algorithm [21]. Each pair of exemplars is matched, resulting in $\binom{N}{2}$ matching results, where N is the number of training exemplars (see Figure 3.4). Each matching result contains the flows from the first set of features to the second set, where the flows store a real valued assignment of each feature from the first set to each feature in the second set.

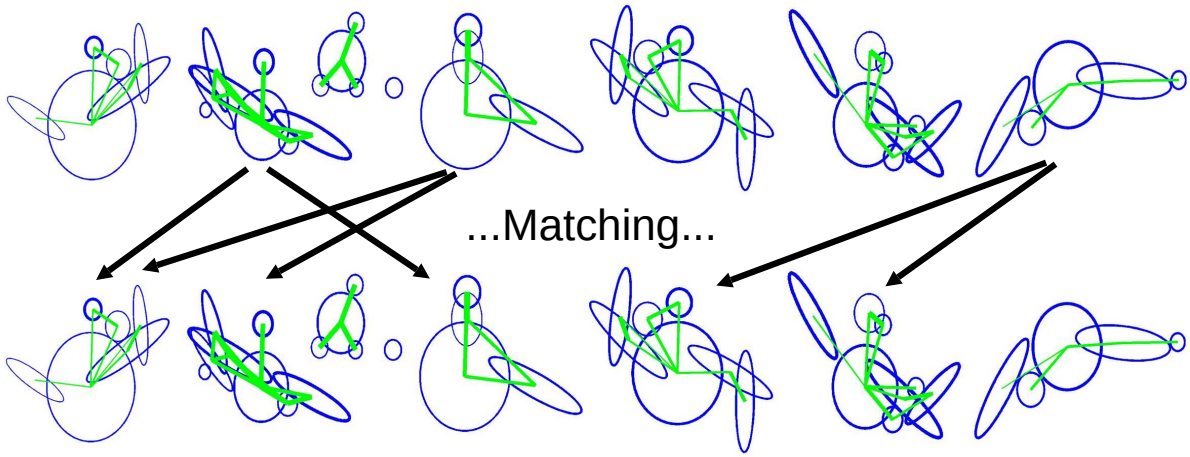


Figure 3.4: Matching each pair of training exemplars many-to-many.

The matching results are used in the part extraction stage to cluster the blobs into parts. A group of features that match strongly among themselves forms a part. Salient parts (or blob clusters) are supported by consistent one-to-one matching results (Figure 3.5 shows sample clusters for right, upper right, and lower right arm parts). The more features that match consistently, the more evidence there is for the existence of a part in the final model. Once the parts

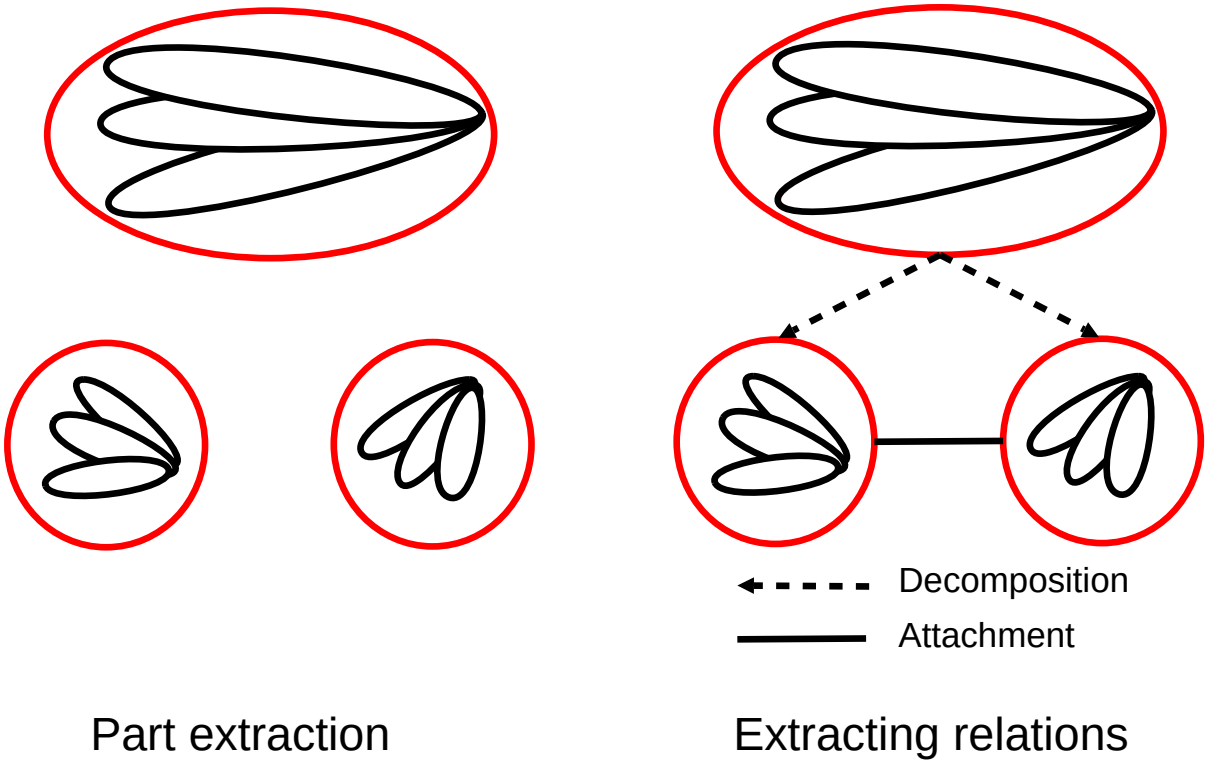


Figure 3.5: Model construction stages. On the left, three parts are formed by clustering blobs from individual exemplars. On the top, the cluster for the right arm is shown. On the bottom, the upper right and the lower right arm clusters are shown. The right portion of the figure shows the relations recovery stage. The right arm is related to the two half-arm parts through a decompositional relation. The two half-arms are connected by an attachment relation.

are formed, decomposition and attachment connections between the parts are recovered based on individual matching and attachment results from the features represented by the extracted model parts (Figure 3.5 shows the relations between the right, upper right, and lower right arm parts in the human torso model). The final result is a part-based decompositional model.

Chapter 4

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 [27]. Accordingly, the input signal f is convolved with Gaussian kernels $g(\cdot;\ t)$ of different variance t, giving $L(\cdot;\ t) = g(\cdot;\ t) * f(\cdot)$. To detect compact parts (blobs), we search for scale-space local maxima in the square of the normalized Laplacian operator,

$$\nabla^2_{norm} L = t\,(L_{xx} + L_{yy}). \tag{4.1}$$

Similarly, elongated parts (ridges) are located at scale-space local maxima in:

$$\begin{aligned} \mathcal{R}_{norm} L &= t^{3/2}\,|L_{pp} - L_{qq}|^2 \\ &= t^{3/2}\,((L_{xx} - L_{yy})^2 + 4L^2_{xy}) \end{aligned} \tag{4.2}$$

where the (p,q) space is obtained by aligning the space to the eigendirections of the Hessian matrix of the brightness function (see [27] for more details on the derivation).

To represent the spatial extent of a detected image structure, a windowed second moment matrix

$$\Sigma = \int_{\eta \in \mathbb{R}^2} \begin{pmatrix} L_x^2 & L_x L_y \\ L_x L_y & L_y^2 \end{pmatrix} g(\eta; t_{int})\, d\eta \tag{4.3}$$

is computed at the detected feature position and at an integration scale t_{int} proportional to the scale of the detected image feature. There are two parameters of the directional statistics that

we make use of here: the *orientation* and the *anisotropy*, given from the eigenvalues λ_1 and λ_2 ($\lambda_1 > \lambda_2$) and their corresponding eigenvectors $\vec{e}_{\lambda_1}$ and $\vec{e}_{\lambda_2}$ of Σ. The anisotropy is defined as $\tilde{Q} = \frac{1-\lambda_2/\lambda_1}{1+\lambda_2/\lambda_1}$, while the orientation is given by the direction of $\vec{e}_{\lambda_1}$. We found the results of the blob detector to be quite noisy even on relatively simple images. To reduce the effect of noise on further system components we "clean" the blob results. We remove the non-salient blobs (by thresholding the saliency parameter), large blobs (the size of the image or larger), and blobs that are mostly included in others.

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 solely 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 (strength of the blob's response, i.e., the size of the detected minimum of the Laplacian in a given scale). 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 detail later) first embeds the nodes of two graphs to be matched into two weighted point sets in Euclidean space. For a given graph, the Euclidean distance between two points is proportional to the weight of the shortest path between their corresponding nodes in the graph, with small distortion. In this geometric space, a powerful many-to-many weighted point matching algorithm, the *Earth Mover's Distance (EMD)* [20], 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 in the embedded space. 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 linking the finer scale parts to be grouped must be relatively small.

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

$$max\{d_1/major(A), d_2/major(B)\}, \tag{4.4}$$

where $major(X)$ is the length of the major axis of blob X, and d_1, d_2 are defined in Figure 4.1. If this measure is greater than a threshold (whose sensitivity we evaluate in Section 7.1), 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 4.1. The connectivity measure is not used for edge computation, since it results in a measure that is far from a metric and causes bad embedding results. Instead, the edge weights are computed as follows:

- **ridge-ridge**: Let p be the intersection point of the major axes of the ridges. The edge weight is the sum of the distances of the center of each ridge from p.
- **blob-ridge**: Let p be the closest point to the blob center that is on the major axis of the ridge. The edge weight is the sum of the distances of the center of the ridge and the center of the blob from p.
- **blob-blob**: The edge weight is the distance between blob centers.

Due to scene clutter, the graph may have a number of connected components, representing multiple objects. We greedily choose the largest connected component as a simple method for figure-ground separation, and discard the other components. Ultimately, a distance matrix over these chosen 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.

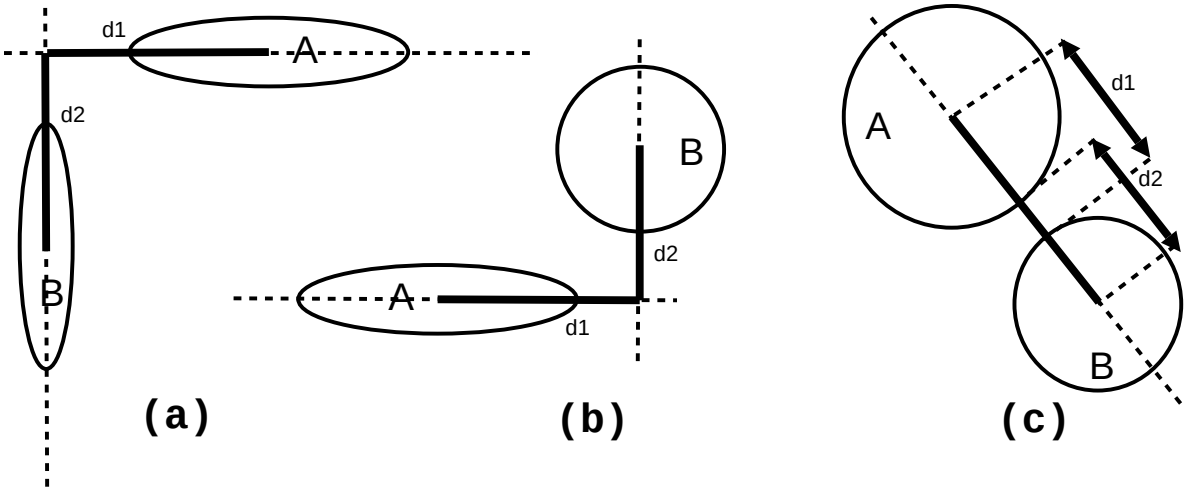
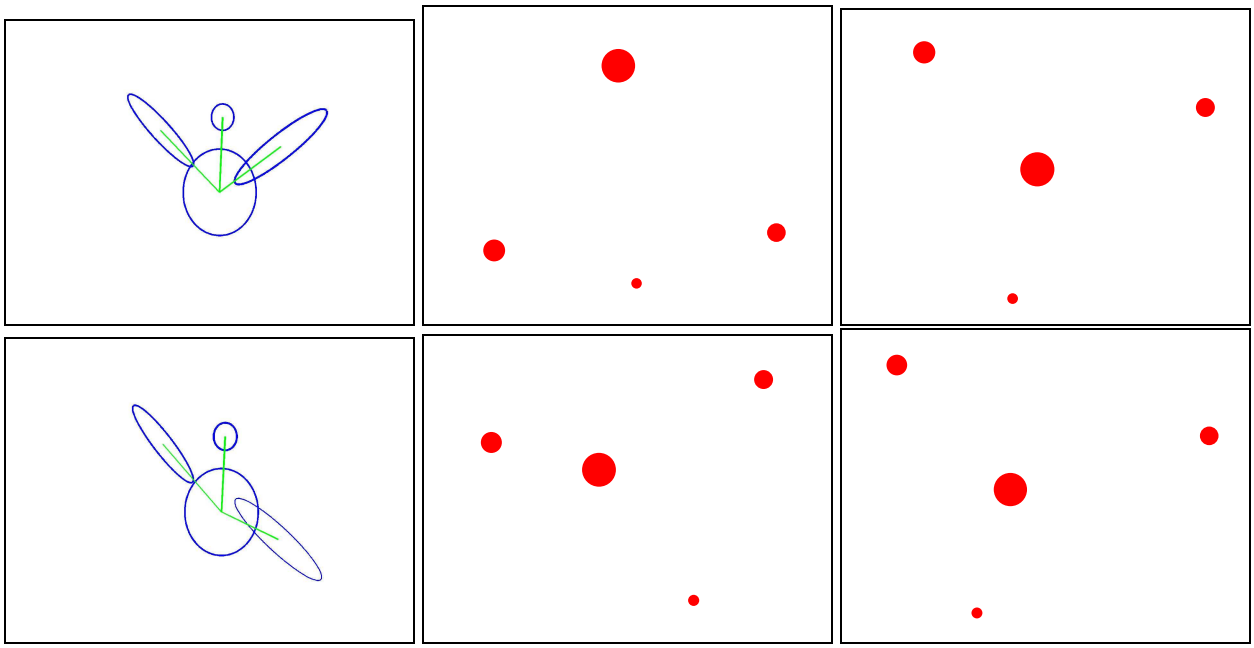


Figure 4.1: Edge construction: (a) ridge-ridge; (b) ridge-blob; and (c) blob-blob. The total length of the bold lines represents the assigned edge weight between the two features in the graph.

For any two previously disconnected nodes, the edge weight becomes the sum of the edges along the shortest path between the two nodes. Figure 4.2 shows the embedding resulting with our choice of distance measure, as opposed to using Euclidian distances between blob centers. The actual embedding procedure is explained in Chapter 5. Our embedding achieves articulation invariance, whereas using the Euclidean distance during embedding does not achieve articulation invariance.



Extracted blobs Euclidean embedding Our embedding

Figure 4.2: Embedding results. The upper row shows one of the exemplars with its embedding using Euclidean distance and our distance measure. The lower row shows similar information for an exemplar with similar parts under different articulation. Our distance measure achieves similar embeddings (similar relative Euclidean positions for corresponding points) for both exemplars under different articulations, whereas using Euclidean distance results in significantly different embeddings.

Chapter 5

Computing Many-to-Many Blob Correspondences

Many-to-many matching of blob graphs is a form of inexact graph matching. The topic can be approached from two directions. Either there is a cost for matching graphs in the original graph space, or the graphs are first embedded and the matching is done in the embedding space. Among the works that are in the first group, Pelillo et al. [41] transform the problem of finding isomorphic subgraphs into the problem of finding maximal cliques. They later extend their work to deal with many-to-many matching, although salient nodes are still assumed to be in a one-to-one correspondence. Sebastian et al. [48] approach graph matching by computing the shortest path in the transformation space of a graph, where graph transformation is defined over shock graphs. Transforming the graph results in the modification of the underlying skeleton that the graph represents. The shortest path corresponds to the smallest graph-edit distance that aligns the two graphs. They apply their approach to shock graphs. In the graph embedding community, Demirci et al. [22] embed the graphs using a novel spherical embedding technique and then match the embedded graphs using the Earth's Movers Distance algorithm. Macrini et al. [49] obtain topological signature vectors based on the sum of the eigenvalues of the graph's adjacency matrix. The resulting vectors represent an abstraction of graph structure, and can be

used to compare graphs (or subgraphs) of different size. Kosinov and Caelli [47] project the graphs onto the first few eigenvectors corresponding to the largest eigenvalues of the graph's adjacency matrix. The actual matching consists of clustering the nodes of the two graphs in the embedding space. One disadvantage in this method (along with some of the aforementioned methods) is that it does not account for edge weights in the input graphs. Also, the methods that match graphs in the graph domain (using graph-edit distance, for example) are computationally expensive, especially when many-to-many matching is needed.

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 Keselman et al. [24] and Demirci et al. [22], 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. Though Demirci et al. deal with affine transformations during matching, experimental results have shown that some graphs cannot be aligned with an affine transformation when embedding our shortest path graphs using spherical embedding. Therefore, we adopt a simpler, spectral embedding technique, similar to [28]. 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 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 [21]. 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 converges to a locally optimal 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.

5.1 Graph Embedding

A number of techniques are available for embedding a distance matrix (encoding, for example, shortest path distances between all pairs of nodes) into Euclidean space; examples include metric tree embedding [23], spherical codes [22], and ISOMAP [28]. In [24], Keselman et al. first convert each graph into a tree and then use Matoušek's embedding method [23] to embed the trees into a Euclidean space. However, the two embeddings do not necessarily have the same dimensionality, and an additional normalization step is needed. In [22], Demirci et al. extend that work by introducing a method that embeds two trees into the same space. However, the two point sets are not aligned and an alignment step is necessary in order to match them. Due to the nature of the embedding, however, it is not clear how to parameterize such an alignment. We adopt a simpler spectral embedding of a distance matrix computed in terms of shortest paths between nodes in a blob graph, similar to [28], which is summarized in Algorithm 1. Though such an embedding is global and not invariant to local mismatches in graph structure, our experiments have shown that the resulting points can usually be aligned using an affine transformation.

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,

Algorithm 1 Embedding of features for EMD

1: Let A be the distance matrix between all the blobs in a given exemplar image.

2: Find the K largest eigenvectors of A and stack them in columns, forming a $N \times K$ matrix X, where N is the number of blobs in a given exemplar. X is a matrix of coordinates for the embedded blobs in a given exemplar, where its i-th row gives the coordinates of the i-th embedded blob.

we embed the graph into a 2-D space, for reasons that will be discussed later.

5.2 Weighted Point Matching

The Earth Mover's Distance (EMD) algorithm under transformation [21] allows us to compute a many-to-many matching of the embedded points which, in turn, specifies a many-to-many node correspondence between the original graphs. Computing the EMD is based on a solution to the well-known *transportation problem*, whose optimal value determines the minimum amount of "work" required to transform one distribution into the other. More formally, let $P = \{(p_1, w_{p_1}), \ldots, (p_m, w_{p_m})\}$ be the first distribution with m points, and let $Q = \{(q_1, w_{q_1}), \ldots, (q_n, w_{q_n})\}$ be the second distribution with n points. Let $D = [d_{ij}]$ be the ground distance matrix, where d_{ij} is the ground distance between points p_i and q_j. Our objective is to find a flow matrix $F = [f_{ij}]$, with f_{ij} being the flow between points p_i and q_j, that minimizes the overall cost:

$$\text{Work}(P, Q, F) = \sum_{i=1}^{m} \sum_{j=1}^{n} f_{ij} d_{ij}$$

subject to the following list of constraints:

$$f_{ij} \geq 0,\ 1 \leq i \leq m,\ 1 \leq j \leq n$$
$$\sum_{j=1}^{n} f_{ij} \leq w_{p_i},\ 1 \leq i \leq m$$
$$\sum_{i=1}^{m} f_{ij} \leq w_{q_j},\ 1 \leq j \leq n$$
$$\sum_{i=1}^{m} \sum_{j=1}^{n} f_{ij} = \min\left(\sum_{i=1}^{m} w_{p_i}, \sum_{j=1}^{n} w_{q_j}\right)$$

The optimal value of the objective function Work(P, Q, F) defines the Earth Mover's Distance between the two distributions, and is recovered using a standard linear programming algorithm.

The original EMD formulation 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 EMD 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 (Algorithm 2). 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, we select the direction with minimum cost. Note that in our greedy version of EMD there is a parameter ϵ which controls the percentage of the remaining mass in source blobs that is distributed. If it is less than 1, then not all the remaining mass would flow to its closest neighbour (even if it could all fit). This gives each feature the opportunity to match other features instead of matching only the closest one, thus trying to counter the negative effects of the greedy solution. The lower this parameter is, the more many-to-many matches will result. During the actual runs of the EMD code, we sample ϵ from 0.5 to 1 and choose the solution with the best cost. A few iterations from the greedy EMD algorithm are shown in Figure 5.1.

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 locations of points in one set and weighted (by the flows) average

Algorithm 2 Greedy EMD

1: Let M_1, M_2 be the vectors containing the masses of the two exemplars, containing N_1, N_2 features respectively.

2: Let $M_R = M_1, M_U = M_2$ be the remaining masses in the first exemplar and the unfilled masses in the second exemplar.

3: Sort the distance matrix d in increasing order, obtaining the index vectors S_1, S_2 of the sorted results, such that $d(S_1(1), S_2(1)) = min_{i,j}\{d(i,j)\}$.

4: Let $Flows$ be the $N1 \times N2$ flow matrix that is intialized with zeros.

5: **for** $i = 1$ to $N1 \times N2$ **do**

6: Let $M_T = min\{\epsilon \cdot M_R(S_1(i)), M_U(S_2(i))\}$ be the transfered mass between features $S_1(i), S_2(i)$, thereby selecting the minimum between the mass remaining in feature $S_1(i)$ and the unfilled mass in feature $S_2(i)$.

7: $M_R(S_1(i)) = M_R(S_1(i)) - M_T$, removing the transferred mass from the first set.

8: $M_U(S_2(i)) = M_U(S_2(i)) - M_T$, adding the transferred mass to the second set (thus decreasing the remaining mass in the second set).

9: $Flows(S_1(i), S_2(i)) = M_T$, storing the amount of mass transferred between features $S_1(i), S_2(i)$.

10: **end for**

11: Flows below a certain threshold and flows over distances above a certain threshold are zeroed out.

12: Let $Cost = \sum_{i=1,j=1}^{N1,N2} (Flows(i,j) \times d(i,j)) + \sum_{i=1}^{N1} M_R(i) + \sum_{j=1}^{N2} M_U(j)$ be the cost of the computed flows, including the original EMD cost in the first term and additional penalty costs for remaining and unfilled masses in the other terms.

13: Normalize the flows by dividing each row in $Flows$ by the sum of that row.

14: Return $Flows$ and $Cost$.

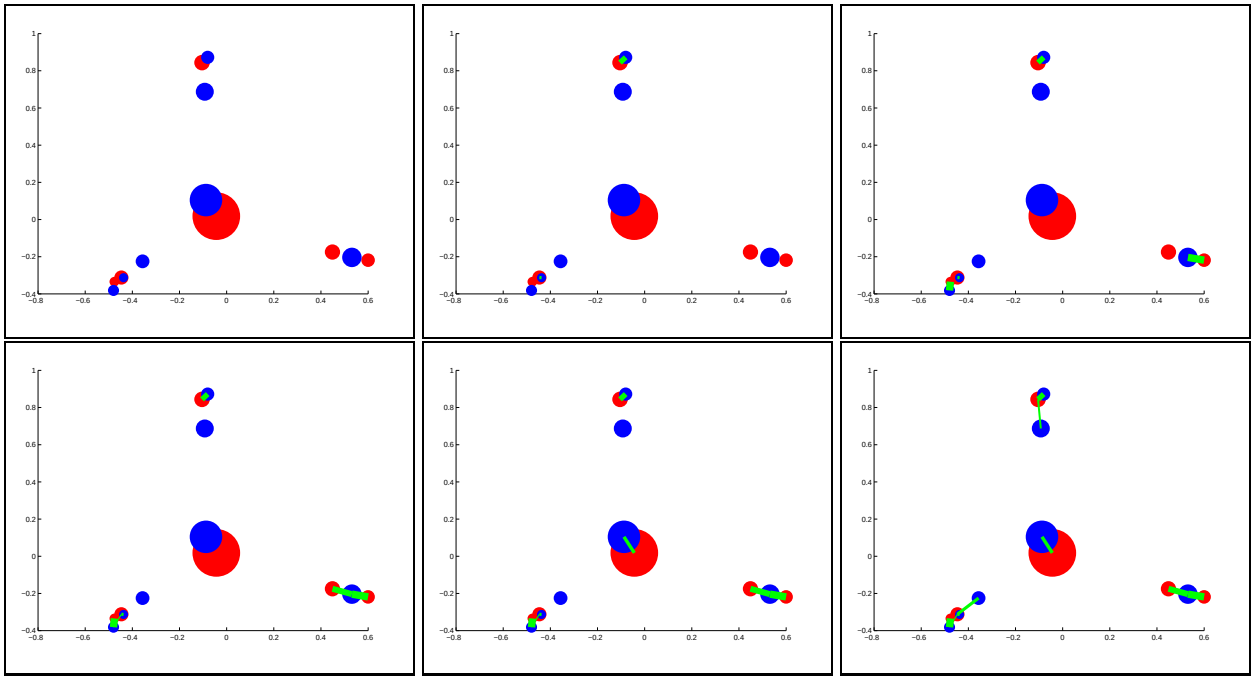


Figure 5.1: Greedy EMD computation between two exemplars given their best alignment. The aligned features, as well as the results after iterations 2,5,7,8,10 are displayed. Blue and red discs indicate features from the two point sets. The area of a disc corresponds to blob mass. Green lines indicate flows from the red to the blue points. Notice that the shortest flows are recovered first, followed by flows over longer distances until the mass from all the red features has been moved or until the blue features have no more capacity for incoming mass.

locations of matched points in the other set:

$$\sum_i \|(P_{1_i} - T(\frac{\sum_j Flows(i,j) \times P_{2_j}}{\sum_j Flows(i,j)}))\|^2, \tag{5.1}$$

where T is a $D-$dimensional affine transformation. D is a parameter of our algorithm. Higher values of D result in a lower distortion during embedding and result in a more accurate many-to-many matching result using our algorithm. However, for the alignment computation not to be underconstrained, there needs to be at least $D+1$ point correspondences between the two point sets each time the alignment is computed. Though some of our exemplar pairs satisfy this restriction for higher values of D, most only satisfy it for low values such as 2 or 3. We therefore choose a 2D embedding and affine transformation in our experiments. It is possible

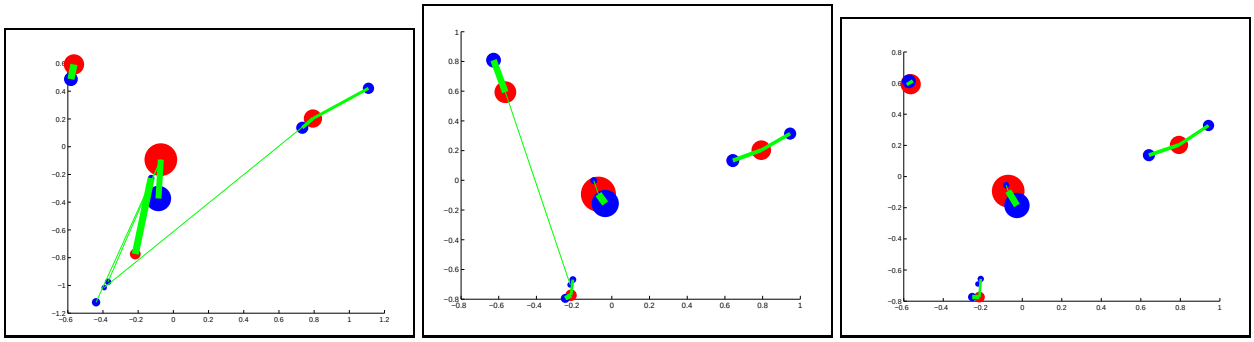


Figure 5.2: 1^{st},2^{nd} and 3^{rd} iterations from the EMD under Transformation algorithm. The corresponding features become aligned while weak correspondences get pruned.

to adapt the dimensionality of the embedding space for each particular matching, the subject of future work.

Algorithm 3 formalizes the alignment computation. We have tested our system on human torso figures. Human torso figures have an inherent right-left symmetry. Since our embedding disregards the actual Euclidean positions of blobs, our matching algorithm cannot distinguish between the right and the left sides of the human torso. Without a good initial alignment for the algorithm, the recovered local solution would be far from the optimal one. For this reason, we first roughly align the blobs from the two exemplars based on their horizontal positions to account for right-left ambiguity. The remaining alignment procedure is described by the alignment algorithm.

In this approximation to the iterative **FT** (an optimal **F**low and an optimal **T**ransformation) algorithm [21], which alternates between computing the EMD flows and computing the affine transformation, the algorithm typically converges in 3-4 iterations. The final flow matrix computed by the algorithm (Algorithm 4) defines a direction of minimum cost. This matrix can be "inverted" to yield a consistent flow matrix for the opposite direction. If $Flows_{12}$ was the normalized $N_1 \times N_2$ flow matrix from points P_1 to points P_2, then the "inverted" flow matrix $Flows_{21}$ is computed by following these steps:

1. Multiply the rows of $Flows_{12}$ by the corresponding entries of M_1 (the mass of features

Algorithm 3 Alignment of points for EMD

1: Let $Flows$ be the the $N_1 \times N_2$ normalized flow matrix. Let P_1, P_2 be $N_1 \times D, N_2 \times D$ matrices of points where N_1, N_2 indicate the number of points in the two point sets respectively and D is the dimensionality of the embedding space.

2: Let $P = \{i \in 1..N_1 | (\sum_{j=1}^{N_2} Flows(i,j)) > 0\}$, indicating a set of matched points from the first exemplar. Let $N = |P|$.

3: **for** $i = 1..N$ (for every matched point of the first set) **do**

4: $VP_i = \sum_j Flows(P(i), j) \times P_{2_j}$, computing the flow-weighted average of points from the second set.

5: **end for**

6: Let $A =$

$$\begin{pmatrix}
VP_{1,1}, VP_{1,2}, \cdots, VP_{1,D}, 1 & 0, \cdots, 0 & \cdots & 0, \cdots, 0 \\
0, \cdots, 0 & VP_{1,1}, VP_{1,2}, .., VP_{1,D}, 1 & \cdots & 0, \cdots, 0 \\
 & \ddots & & \\
0, \cdots, 0 & \cdots & 0, \cdots, 0 & VP_{1,1}, VP_{1,2}, .., VP_{1,D}, 1 \\
VP_{2,1}, VP_{2,2}, \cdots, VP_{2,D}, 1 & 0, \cdots, 0 & \cdots & 0, \cdots, 0 \\
0, \cdots, 0 & VP_{2,1}, VP_{2,2}, .., VP_{2,D}, 1 & \cdots & 0, \cdots, 0 \\
 & \ddots & & \\
0, \cdots, 0 & \cdots & 0, \cdots, 0 & VP_{2,1}, VP_{2,2}, .., VP_{2,D}, 1 \\
VP_{N,1}, VP_{N,2}, \cdots, VP_{N,D}, 1 & 0, \cdots, 0 & \cdots & 0, \cdots, 0 \\
0, \cdots, 0 & VP_{N,1}, VP_{N,2}, .., VP_{N,D}, 1 & \cdots & 0, \cdots, 0 \\
 & \ddots & & \\
0, \cdots, 0 & \cdots & 0, \cdots, 0 & VP_{N,1}, VP_{N,2}, .., VP_{N,D}, 1
\end{pmatrix},$$

be a $(N \cdot D) \times (D \cdot (D+1))$ matrix, x be the $(D \cdot (D+1)) \times 1$ vector representing the unknowns of the affine transfortmation T ($D \times (D+1)$ matrix) written in a row-wise order, and

$$b = \begin{pmatrix}
P1_{P(1),1} \\
P1_{P(1),2} \\
\vdots \\
P1_{P(1),D} \\
\vdots \\
\vdots \\
P1_{P(N),1} \\
P1_{P(N),2} \\
\vdots \\
P1_{P(N),D}
\end{pmatrix},$$

storing the coordinates of each point from the first set, such that $Ax = b$ becomes an overdetermined system of linear equations. Flow weighted points from set 2 that are stored in A undergo a linear transformation (the same for each point), stored in x, to become the points from set 1 that are stored in b.

7: $x = (A^T A)^{-1} A^T b$. Solve the overdetermined system using normal equations.

8: Let T be the resulting $D \times (D+1)$ affine transformation, such that row i of T is $x(((i-1)(D+1)+1) \ldots i(D+1))$ (storing the $D+1$ linear coefficients that produce the i-th coordinate in the transformed set of points).

from the first set), thus making the flows unnormalized again.

2. Transpose the resulting matrix.

3. Normalize the resulting matrix by dividing each element by the sum of its row entries.

These two matrices will play a key role in our procedure for extracting the parts in the final decompositional model. Figure 5.2 shows a few iterations from the EMD under transformation algorithm, and Figure 5.3 shows pairs of blob graphs and the final matching between them, as computed by the algorithm.

Due to the limitations in embedding and noise present in the input graphs in the locations and sizes of the input blobs (which affects our perceptual grouping and matching stage), our matching algorithm has some drawbacks. The first problem arises due to the fact that we discard the Euclidean positions of the blobs in the images and store only a shortest path distance between them as computed by our perceptual grouping stage. Two blobs with similar global relations to the rest of the blobs in the image get embedded similarly. For example, a head blob in one image can match a noisy blob in another image that is located on the side or underneath the body. Shortest path distance does not maintain node ordering, e.g., outgoing edges of a node. Therefore, since the distributions of shortest path distances of these two blobs relative to the other blobs in their exemplars are the same, they get embedded similarly.

The second problem arises due to noisy input. Since EMD relies on the mass conservation principle, our algorithm assumes that a given part on an object will have the same mass, whether it is detected as one or as several features. Though our algorithm is somewhat robust to mass mismatches, due to its greedy nature nearby features will get filled first. If, for example, a part that is detected as one feature in the first exemplar is much smaller in mass than the total mass of features that represent it in another exemplar, then only a portion of the features from the second exemplar will match the feature from the first exemplar. Figure 5.4 illustrates these two problems.

Algorithm 4 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.

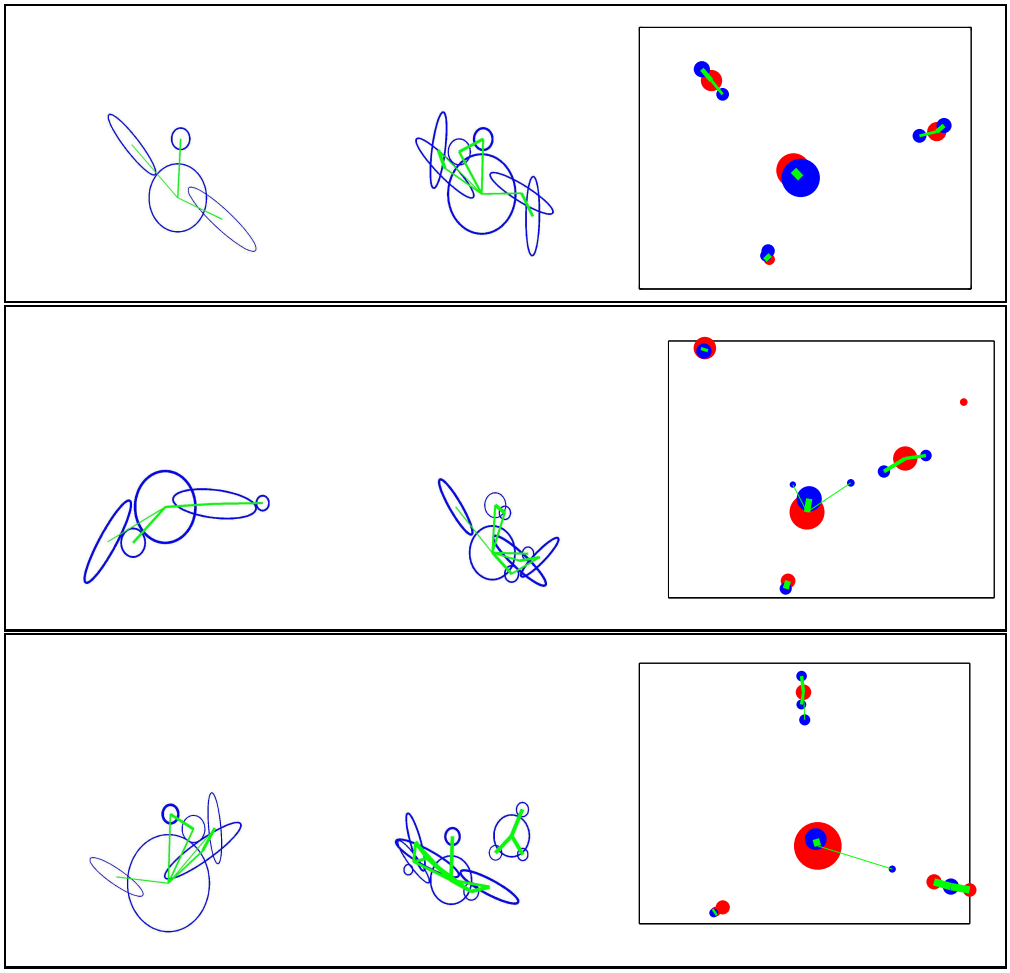


Figure 5.3: Many-to-many matching of blob graphs using Earth Mover's Distance under transformation in embedded (Euclidean) space. Left two images show the detected blobs with green lines indicating blob connections and line width indicating edge strength (nonaccidental, proximity-based grouping strength). The right figures show the embedded features (red for left images, blue for right images) 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). Note that some blobs are nearly unseen in the right column due to their small mass and proximity to other bigger blobs. Also, the 2^{nd} image in the final row has a second small connected componnent which was removed prior to the embedding.

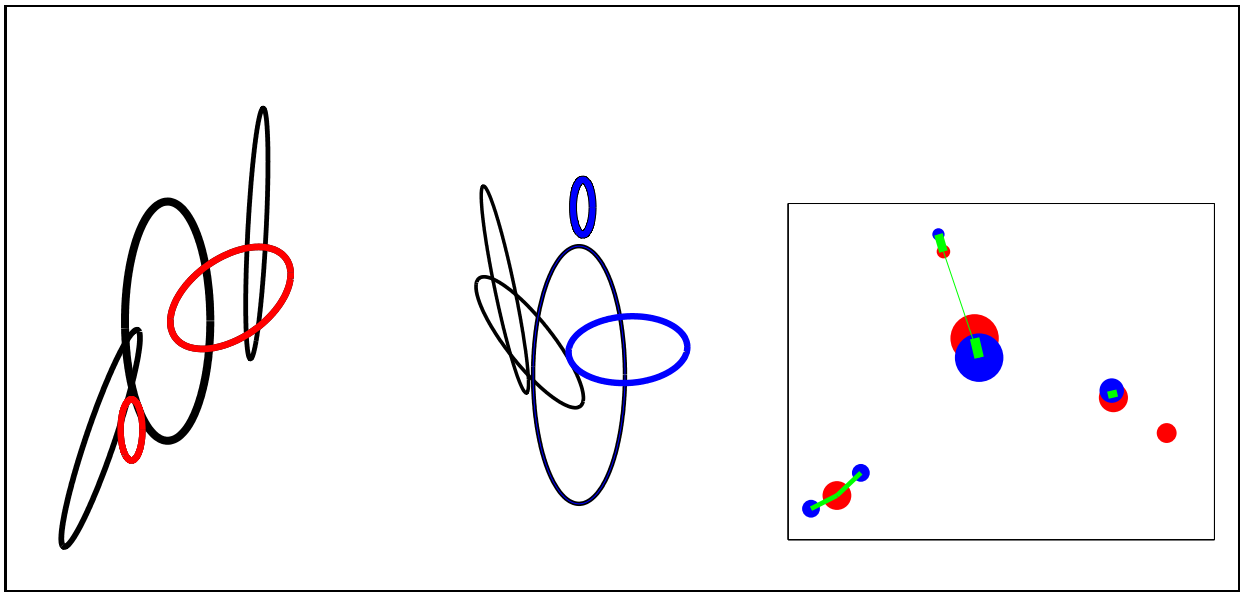


Figure 5.4: Failures in matching show where our algorithm fails. Note that the head in the center figure is embedded (top blue blob in the right figure) approximately the same way as the noisy blob in the left figure (on the left side of the torso - the red blob in the top of the right figure). Moreover, the left arm is detected as two features in the left image and as a single feature in the center image. The mass of the upper arm in the left figure (right red blob) is larger than the mass of the full arm in the center figure, preventing a 2-to-1 match (the forearm matches a full arm 1-to-1 as shown on the right side of the right figure). The red blobs in the left image match the blue blobs in the right image (they also correspond to two of the the red and blue points in the embedding figure).

Chapter 6

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 in M 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. The detected parts and their relations are used to construct the final decompositional model. The following subsections outline these steps in more detail.

6.1 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 blob 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 1.0 turns the symmetric flow matrix into a symmetric distance matrix, setting up a clustering problem where clusters represent collections of nodes, pairs of which are in one-to-one correspondence. Again, we draw on spectral techniques to embed the distance matrix into a low-dimensional space, and use the k-means[1] algorithm for clustering [29]. The quality of the cluster is in range $[0, 1]$ and it is proportional to the "cliqueness" of the one-to-one matches among the members of the cluster, where the quality is computed by averaging the pairwise one-to-one matching results for all pairs of blobs in a given cluster. If a cluster is of sufficient size and quality, it becomes a node in the final decompositional model. The algorithm is formalized below. Figure 6.1 shows four exemplar images and the distance between every pair of blobs. Black indicates a small distance (i.e., a good match).

[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.

Algorithm 5 Spectral Clustering Algorithm (taken from Ng, Jordan and Weiss [29])

1: Let A be the distance matrix between all the blobs in a given graph.
2: Let D be the diagonal matrix whose (i, i)-element is the sum of A's i-th row, and let $L = D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$.
3: Find the K largest eigenvectors of L and stack them in columns, forming a $N \times K$ matrix X, where N is the total number of blobs and K is the number of clusters.
4: Form the matrix Y by normalizing each row of X to have unit length.
5: Cluster the rows of Y into K clusters using the k-means clustering algorithm, treating each row as a point in R^K
6: Blob i is assigned to cluster k if and only if row i of Y is in cluster k.

6.2 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:

$$PA(i,j) = \frac{\sum_{p=1}^{P}\sum_{k=1}^{B(p)}\sum_{l=1}^{B(p)}[C_p(k)=i][C_p(l)=j]conn_p(k,l)}{\sum_{p=1}^{P}\sum_{k=1}^{B(p)}\sum_{l=1}^{B(p)}[C_p(k)=i][C_p(l)=j]} \quad (6.1)$$

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.

For decompositional relations, we restrict ourselves to one-to-many decompositional relations. This restriction, compared to having many-to-many relations, was imposed for two main reasons: (1) Most domains have a hierarchical structure in which one part may decompose into several, and there are few situations in which two subsets of more than one node match; and (2) Such a simplification makes the matching stage easier and more stable, allowing the use of a greedy approach instead of a global optimization. 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 normalized flows from all blobs in extracted part i's cluster to all blobs in extracted part j's cluster:

$$PF(i,j) = \frac{\sum_{k=1}^{N}\sum_{l=1}^{N}[C(k)=i][C(l)=j]M(k,l)}{(\sum_{k'=1}^{N}[C(k')=i]) \times (\sum_{l'=1}^{N}[C(l')=j])} \tag{6.2}$$

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 normalized 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 $[0, 1]$. Figure 6.2 shows a decomposition under idealized conditions given by the PA and the PF matrices.

Given the part flow (PF) and part attachment (PA) matrices, Algorithm 6 extracts the part decomposition relations among the extracted parts in the final model. T_{child} (0.6) is determined

Algorithm 6 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 $e^{-\left|1-\sum_{j\in D'} PF(i,j)\right|} \times min\{1, \frac{\sum_{k,l\in D'} PA_{D'}(k,l)}{|D'|-1}\}$ {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}

empirically and reflects the degree to which a conservation of 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).

6.3 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 quality value, defined as the average of all the pairwise one-to-one matching results of blobs in a given cluster (defined in Section 6.1). 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.

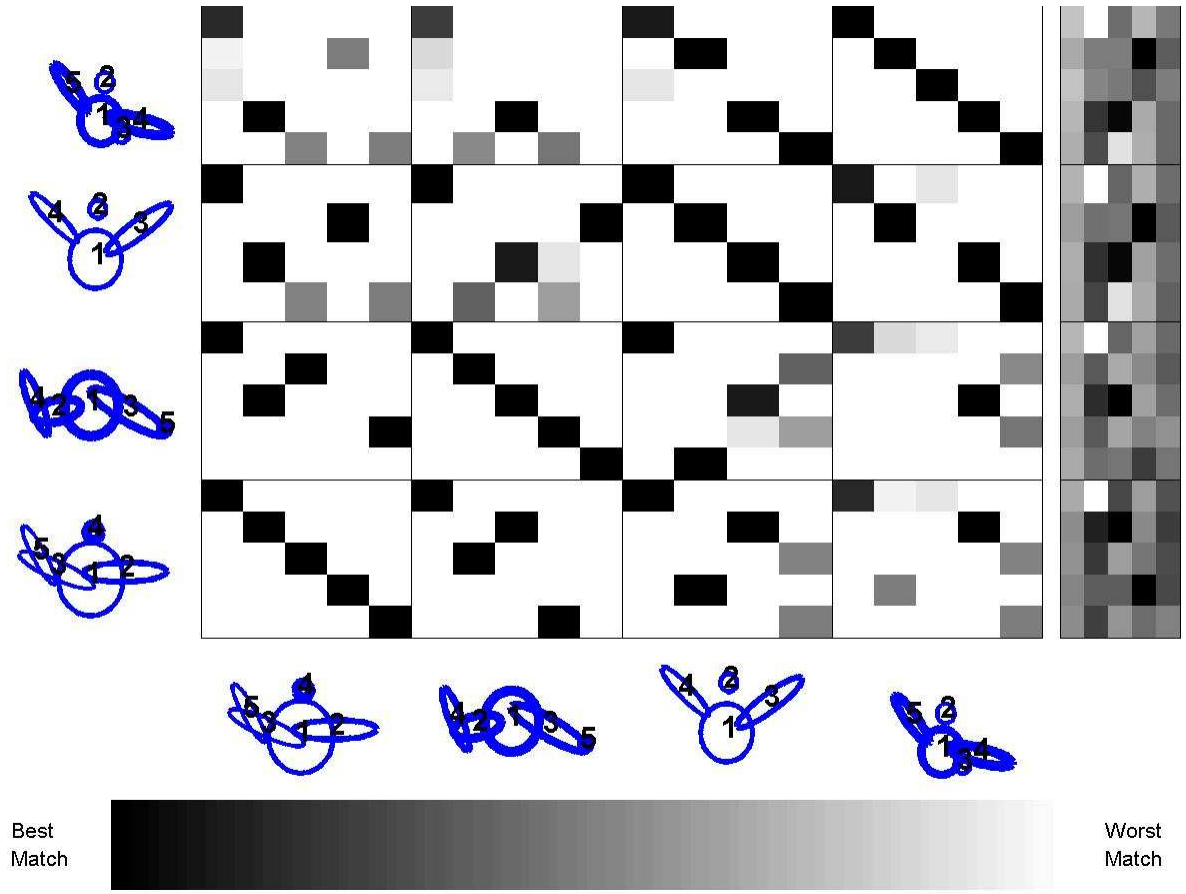


Figure 6.1: Distance matrices between blob graphs. A portion of the distance matrix A that is used for spectral clustering. Blobs that match well one-to-one are clustered together. For example, the left arms of the subject, corresponding to blobs 2,3,3,4 in images 1-4 respectively (where the origin is in the bottom left of the figure), all match well one-to-one. The last column shows the first 5 dimensions in the embedding space, where the range 0..1 is colour coded from black to white. Note that blobs corresponding to the same part have similar coordinates. For example, the torso blobs, coresponding to the first blob in each image, have similar coordinates.

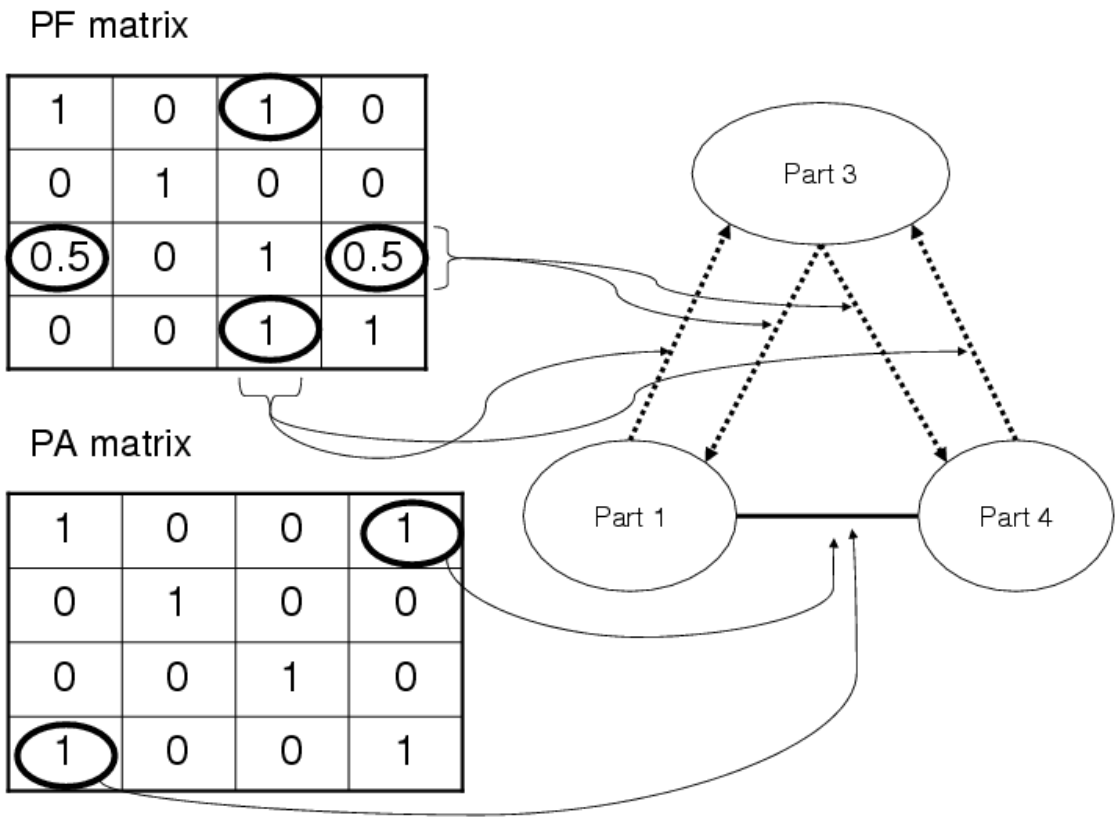


Figure 6.2: Ideal decomposition. In this example a total of 4 parts were recovered. Of the 4 recovered parts, the right part of the figure illustrates a portion of the resulting model where a decomposition of part 3 into parts 1 and 4 is shown. Notice the supporting evidence for such a decomposition given by the PA and PF matrices. These matrices satisfy the three conditions for a good decomposition mentioned above.

Chapter 7

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 1.2. Usually, models are evaluated through recognition performance. We have not yet developed a recognition component that would use our model. Therefore, ground truth is provided for each input image in the form of a manual labeling of the extracted blobs in terms of the parts in an ideal torso decompositional model, shown in Figure 3.2; blobs that are not deemed (by a human observer) to correspond to a part on the ideal model are labelled as noise. Given the ideal model and a user-defined labelling of blobs in an input image according to the model, the ground truth attachment edges and pairwise matching correctness can be induced automatically. 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[1], 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 (Table 7.1).

[1]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.

Parameter	Description
Perceptual grouping threshold	The threshold determining the pairwise connectivity between blobs in the exemplar images.
Embedding dimensionality during matching	The dimensionality of the embedding space in which every pair of exemplars is embedded and matched.
Embedding dimensionality during part extraction	The dimensionality of the embedding space in which all blobs from all exemplars are embedded and clustered into parts based on one-to-one matching.
k	The maximum number of parts in the model.
T_{attach}	The threshold for accepting attachment between parts in the final model, based on the entries of the PA matrix.
T_{child}	The threshold for considering a part to be a potential child of another part, based on the entries of the PF matrix.
T_{decomp}	The threshold for accepting a decomposition based on its quality measure.

Table 7.1: List of system parameters

7.1 Evaluation of Input Blob Graphs

As mentioned in Chapter 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 3.2, the percentage of images in which the designated part was present in the ground truth data 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 7.1(a) shows the error in individual attachment, representing the sum of the SSD error in the attachment matrices of all exemplar blob graphs. As can be seen from the figure, there is a clear minima in the error function for a proximity threshold of about 1.0. Our system is relatively sensitive to this parameter. Choosing a threshold that is far from the optimal will result in incorrect matching, and the error will propagate to the other stages of the model construction process.

7.2 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 matrix of each pair of exemplars. The matching component does have a number of parameters, such as the mass fraction ϵ that is distributed during the greedy EMD algorithm, and the number of dimensions used for graph embedding. However, in the case of ϵ, the parameter is automatically sampled and the best value is chosen for each individual matching. In the case of the number of dimensions, the low number of object features in the

training set that was used prevents us from using a larger number of dimensions (as explained in Chapter 5). 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.

7.3 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 7.1(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 7.1(c) plots smoothed error as a function of k (max number of clusters). Since the minimum is rather shallow, our algorithm is not very sensitive to the choice of k.

7.4 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 the dissimilarity in the number of edges between the ground truth model and the final recovered model. Figures 7.1(d) and 7.1(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.

7.5 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 7.1(f) shows the smoothed final model error and its three components as a function of set size. It can be clearly seen that the errors decrease as the number of training images increases. Figures 7.2–7.4 show correctly generated decompositional shape models given the full set of the torso images as input. Since the part extraction stage contains a clustering component that includes a random initialization of clusters, the model acquisition is non-deterministic. These three figures show the models recovered with different cluster initializations. The models are structurally the same with minor differences in the contents of the part clusters. Figures 7.8–7.15 show the different parts of the first of these recovered models (Figure 7.2). The recovered models are isomorphic to the ideal model, reflecting our algorithm's ability to correctly recover a decompositional model from examples.

Figures 7.5–7.7 show potential problems during model recovery, which include recovering extra parts and relations; however, these problems arise mostly due to bad choices of the different thresholds that were analyzed in the previous sections. For example, in Figure 7.5, an extra head was detected. Due to the nature of the exemplar images and our matching algorithm, the head blob often matches with other blobs connected to the body. This results in a less consistent matching for the head cluster; thus, depending on the initialization of clusters, the head

may be detected as one or as two clusters.

Figure 7.6 shows a model with an extra attachment between the upper left and upper right arms. Due to the nature of the detected parts and our perceptual grouping stage, blobs belonging to the upper left and upper right arms are sometimes considered attached. In this experiment, the value of T_{attach} was lowered from the optimal value of 0.6 to 0.4. This resulted in the detection of this extra attachment relation, which is still strongly supported by the recovered blob graphs.

Figure 7.7 shows several problems. In this experiment, the value of the initial number of clusters, k, was changed from its optimal value of 12 to 20 initial clusters. Although some of the 20 clusters were merged or removed, the resulting model still has 13 parts. Some of these clusters correspond to the same part; for example, clusters 1,2 and 4 all correspond to the torso. Moreover, if a part that is a parent in a decomposition is detected as two or more clusters (as happened with the left arm, detected as clusters 11 and 12), each of the correponding clusters will decompose into its constituent parts, resulting in extra decompositional relations.

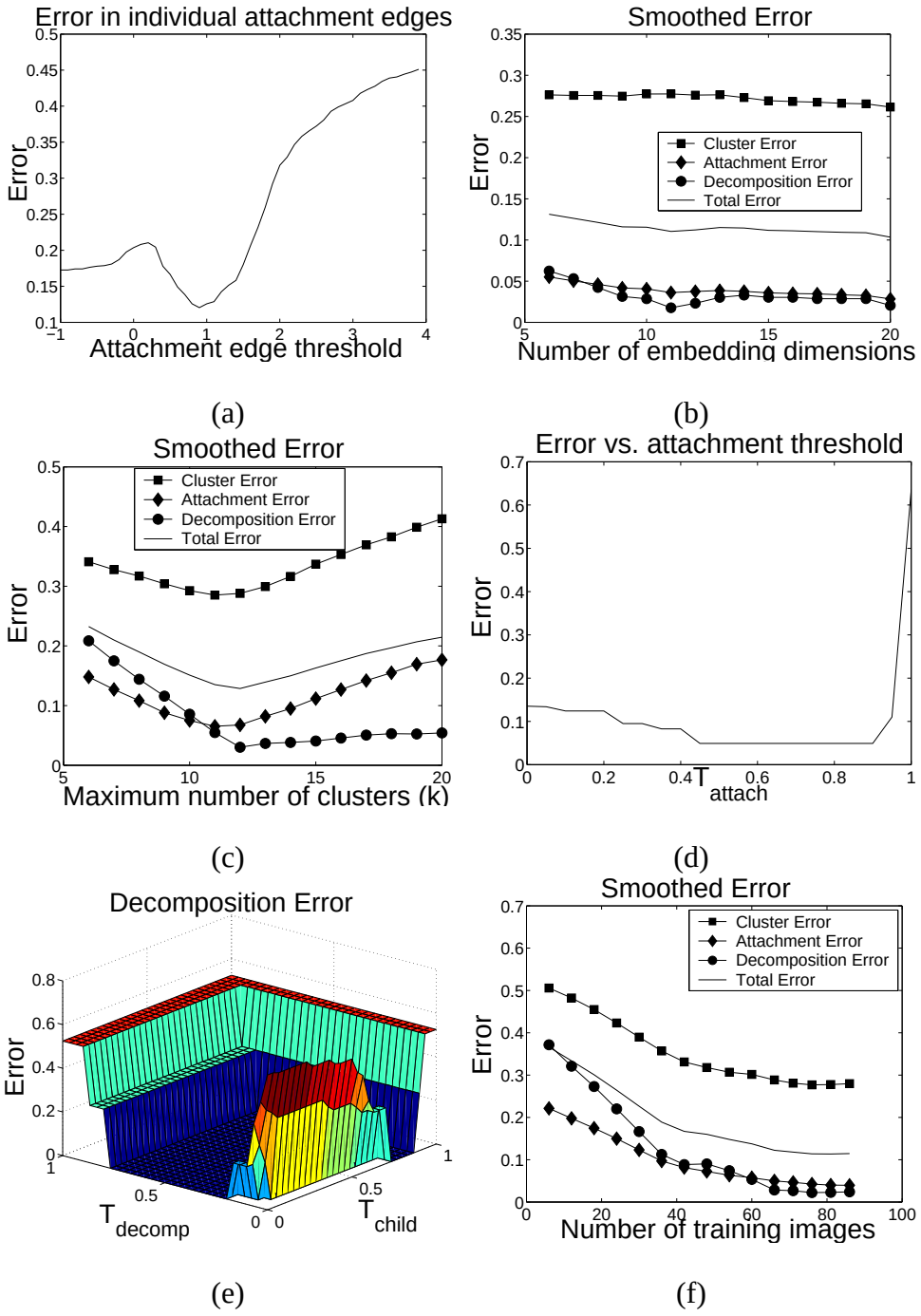


Figure 7.1: Evaluating the Model: (a) Input attachment relation error as function of proximity-based grouping threshold; (b,c,f) The four curves represent clustering error, recovered attachment edge error, recovered decomposition edge error, and final decompositional model error as a function of dimensionality 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}.

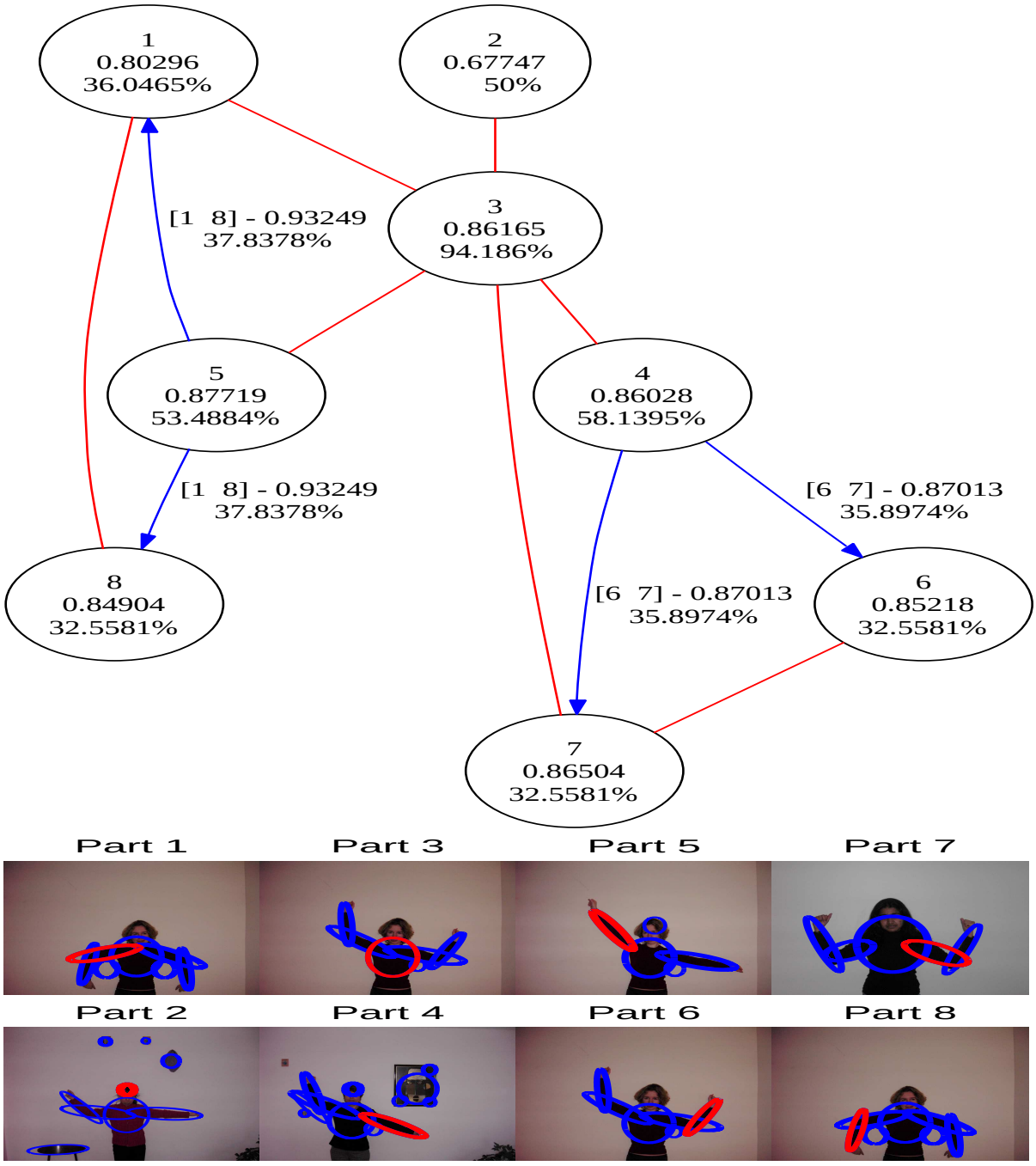


Figure 7.2: Correct final decompositional model obtained by our system on 86 input images. Red edges indicate part attachment, while blue edges indicate part decomposition (or, inversely, abstraction). The values on the decomposition edges specify the children (square brackets), the quality of the decomposition, 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.

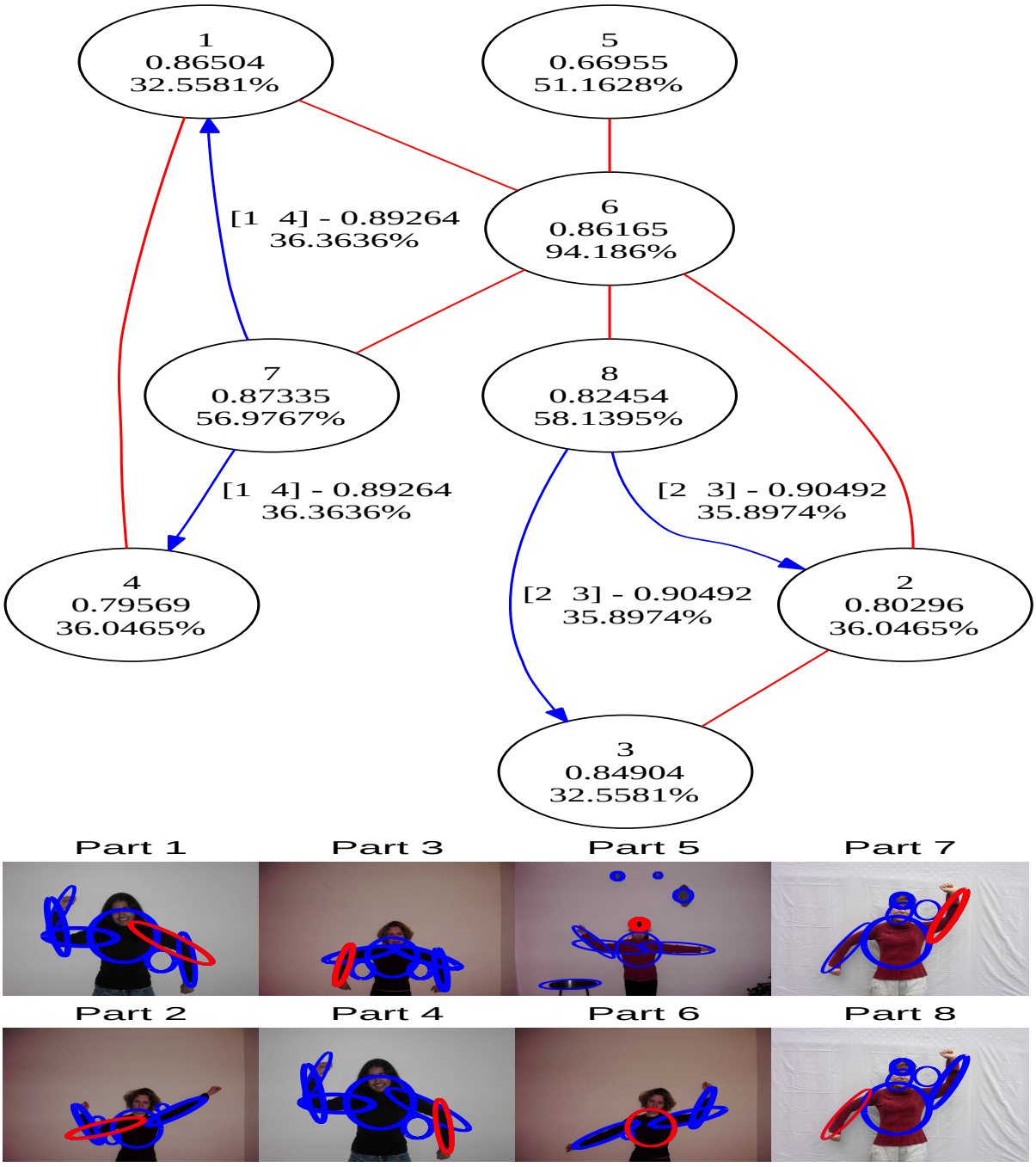


Figure 7.3: Correct final decompositional model obtained by our system on 86 input images. This model was obtained with a different part cluster initialization.

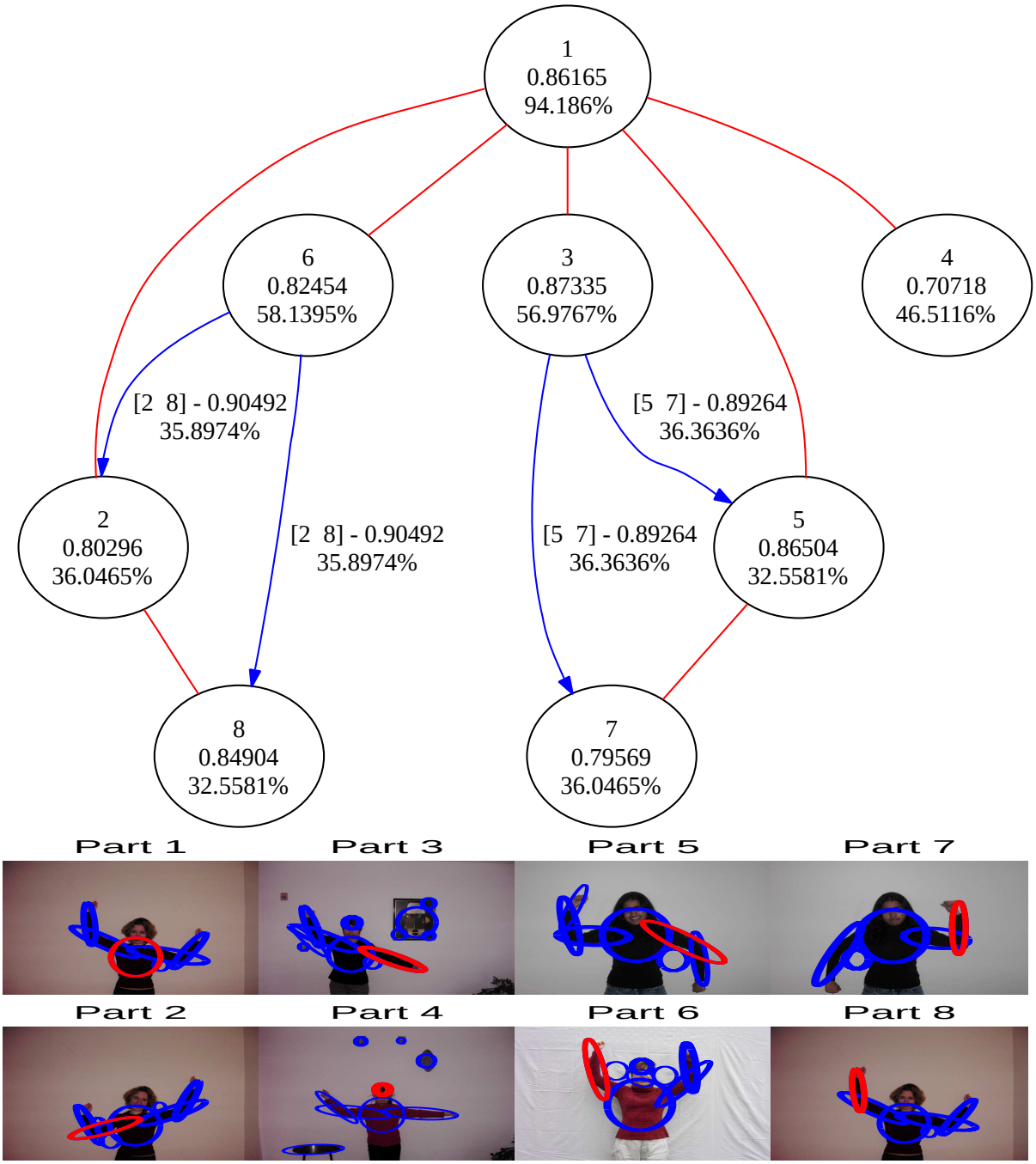


Figure 7.4: Correct final decompositional model obtained by our system on 86 input images. This model was obtained with a different part cluster initialization.

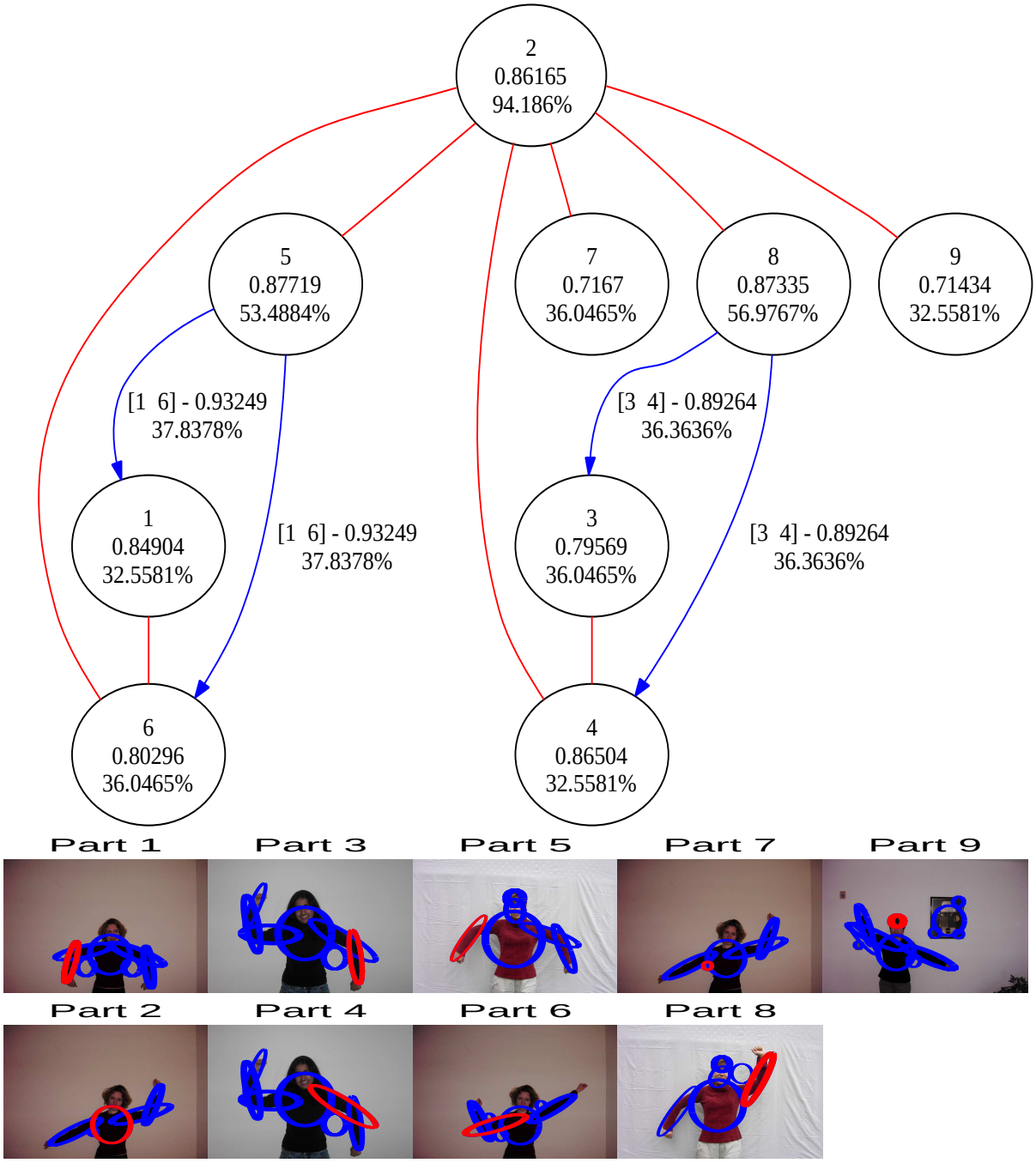


Figure 7.5: Incorrect final decompositional model obtained by our system on 86 input images. Here, two clusters for the head were found. Both clusters are strong enough to become parts, yet not strong enough to be detected as the same part.

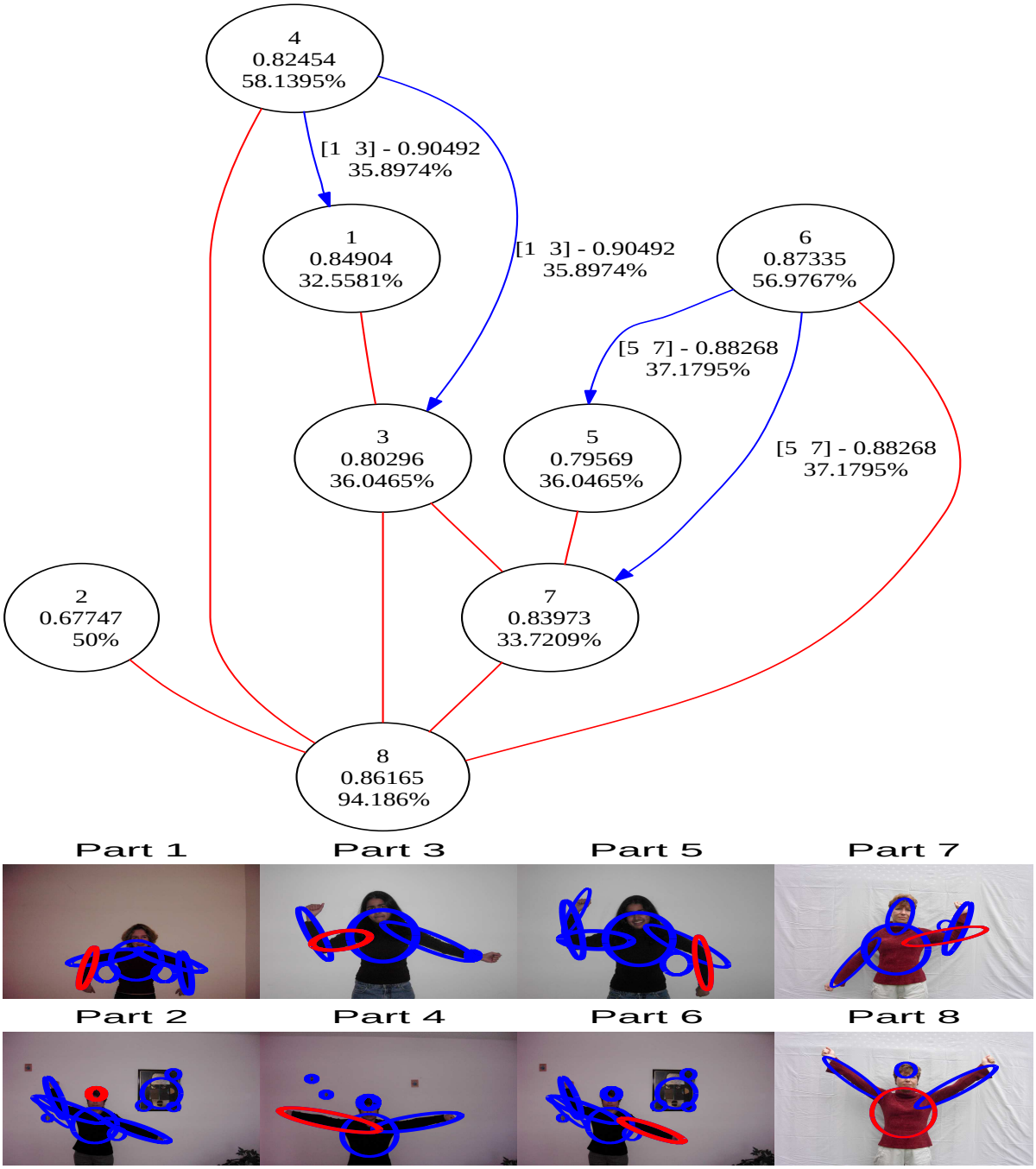


Figure 7.6: Incorrect final decompositional model obtained by our system on 86 input images. Here, the left and the right upper arms are attached. In many exemplar images, the corresponding blobs appear closely together and are attached. If enough such exemplars exist, the parts will become attached.

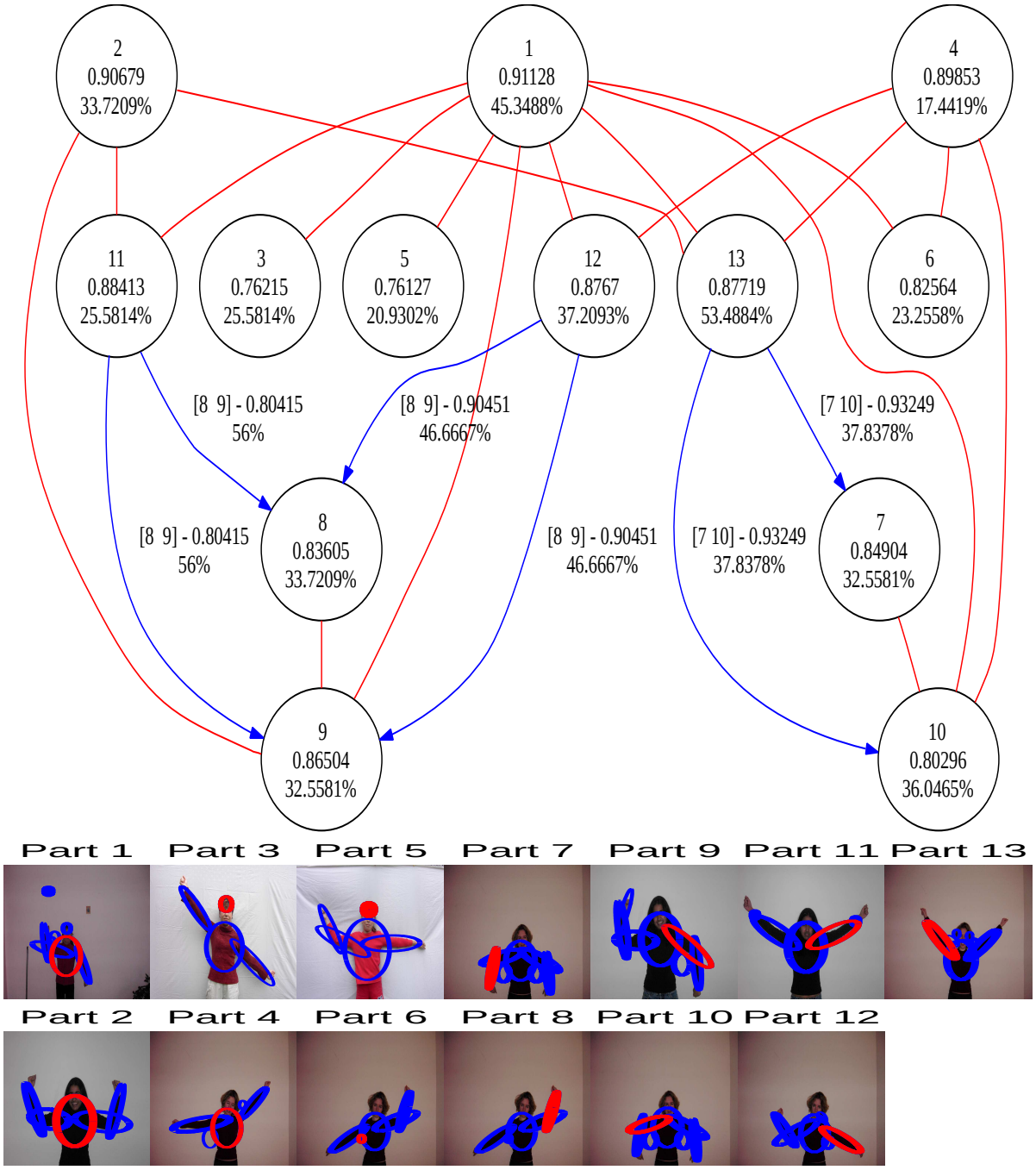


Figure 7.7: Incorrect final decompositional model obtained by our system on 86 input images. Due to the large initial number of clusters, k, many extra clusters are detected. Notice that not only are there different clusters representing the same part, such as the torso or the head, but that the left arm was detected as two clusters, both of which decompose into half-arm parts.

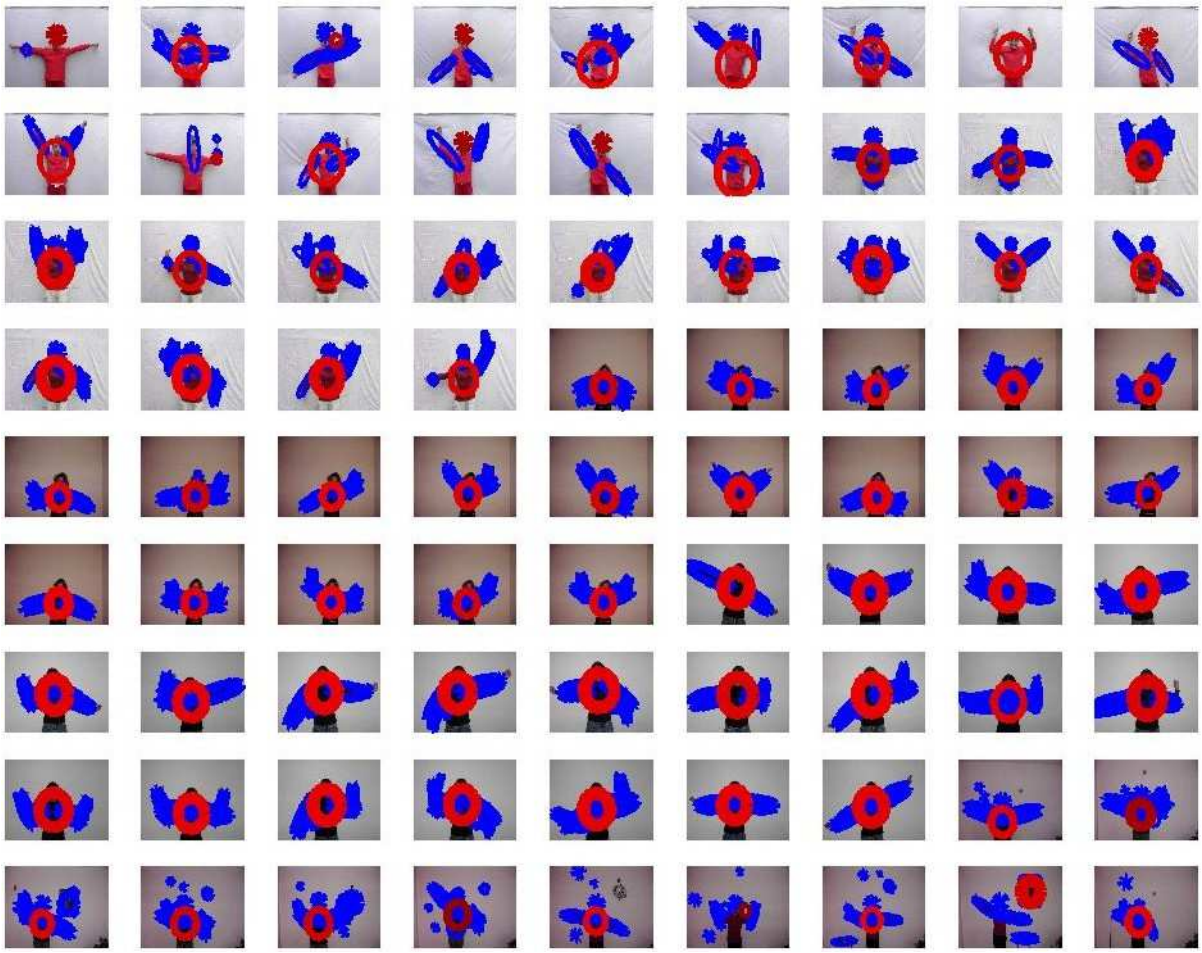


Figure 7.8: The torso part in the resulting model. Most blobs in the torso cluster, shown in red, do correspond to a torso. Some noisy blobs or other parts are included as well.

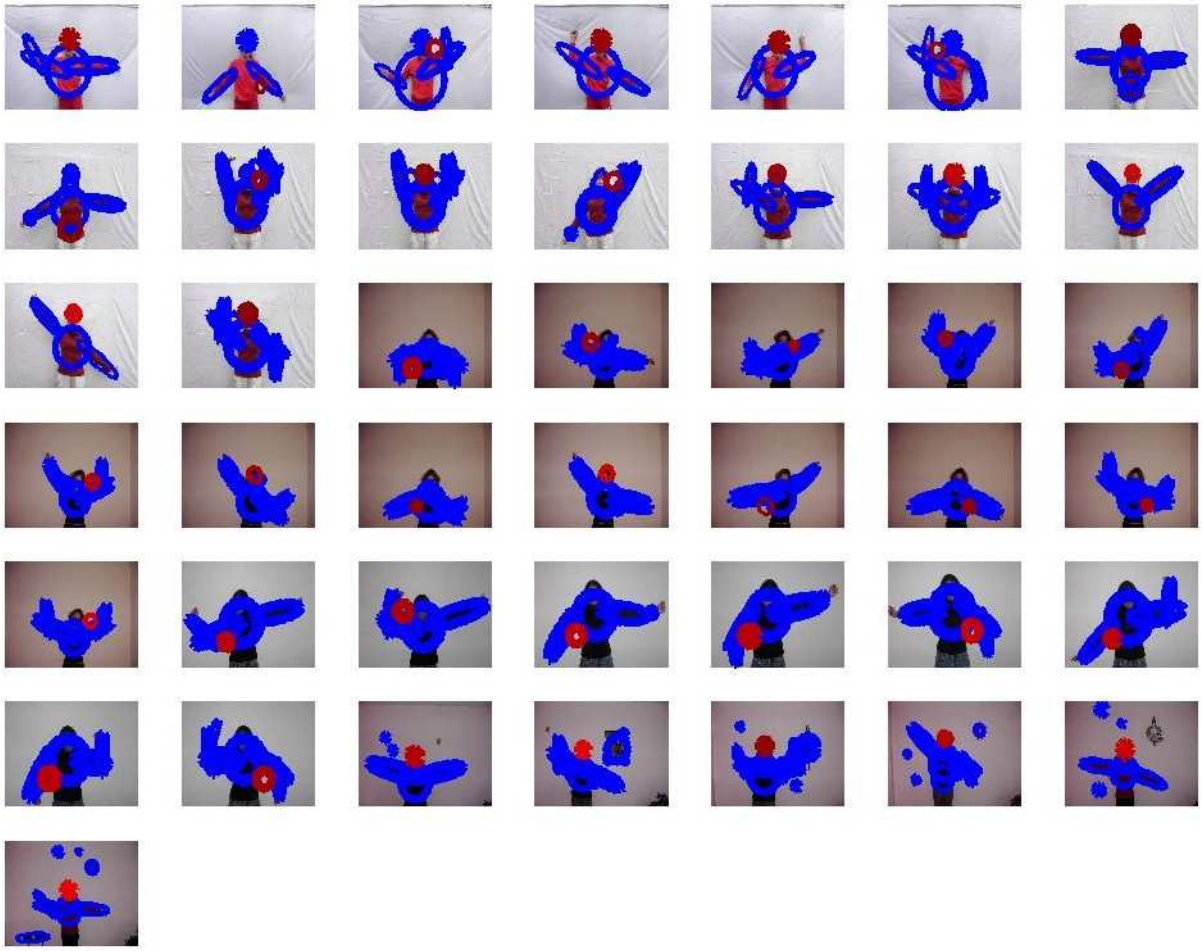


Figure 7.9: The head part in the resulting model, shown in red.

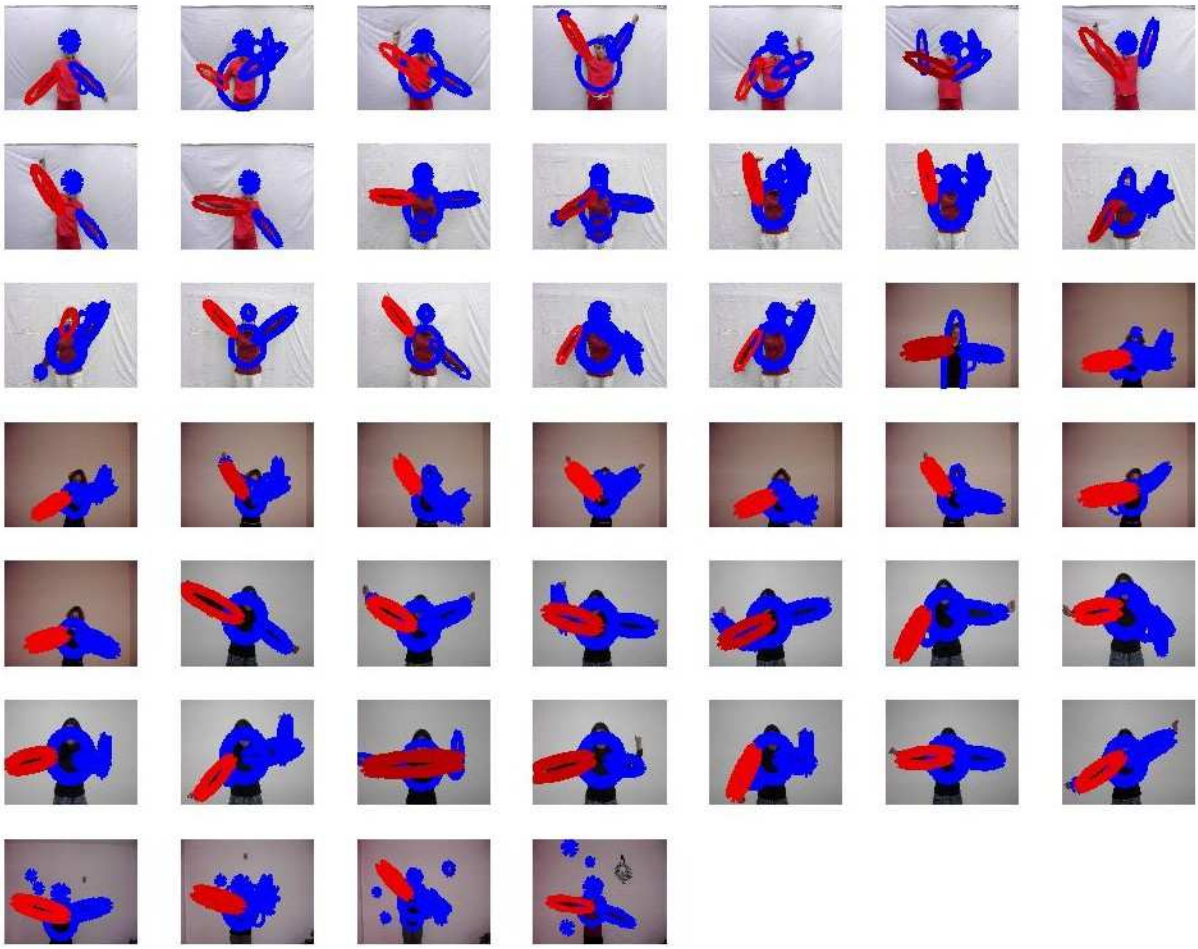


Figure 7.10: The right arm part in the resulting model, shown in red.

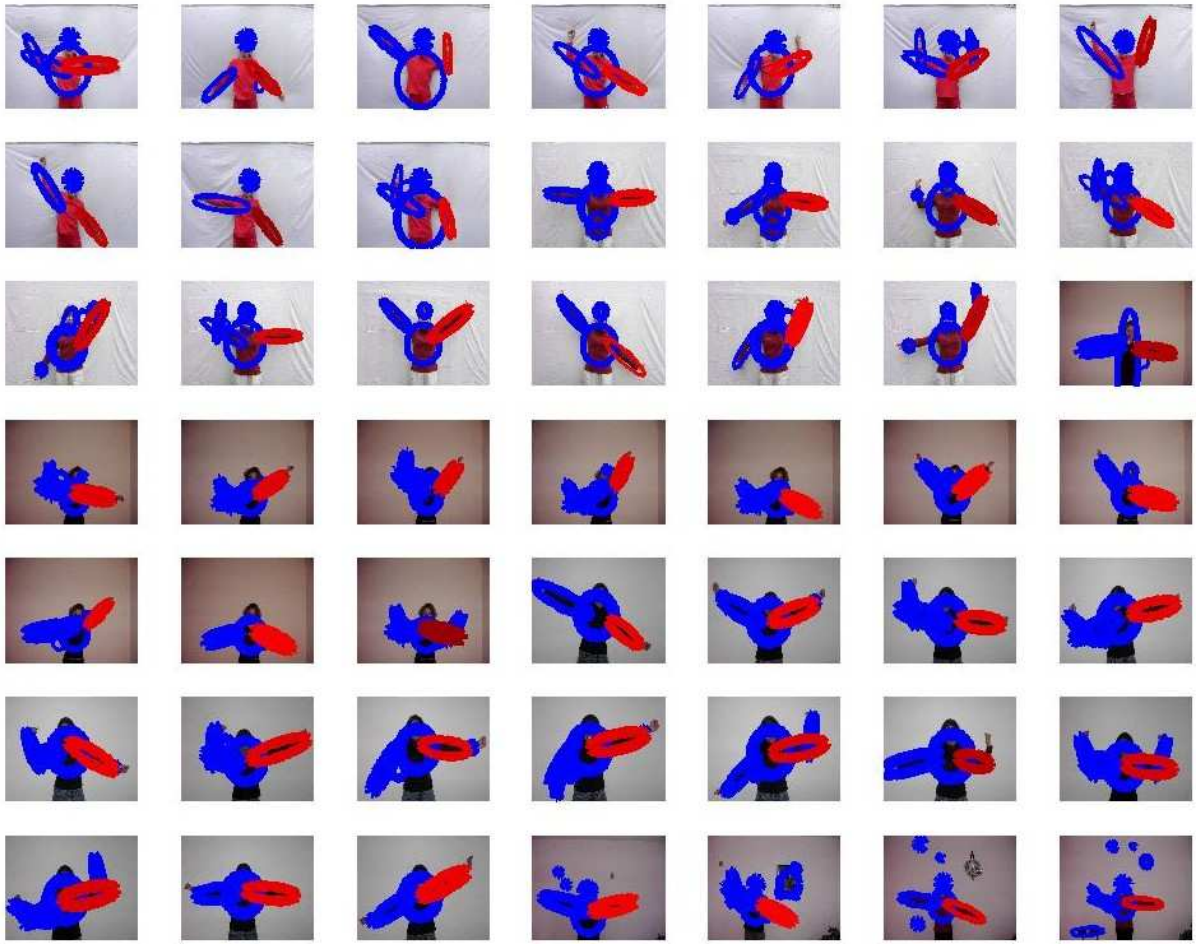


Figure 7.11: The left arm part in the resulting model, shown in red.

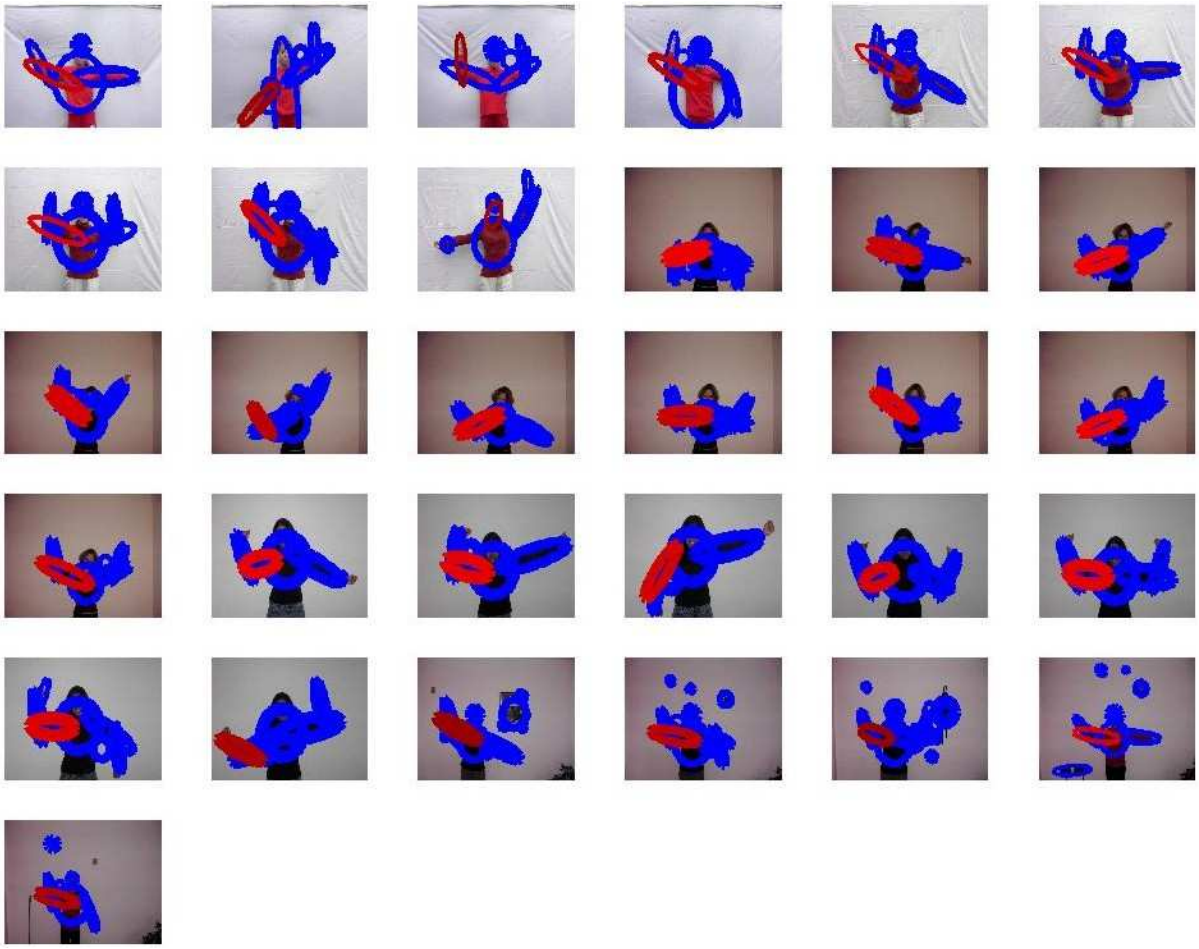


Figure 7.12: The upper right arm part in the resulting model, shown in red.

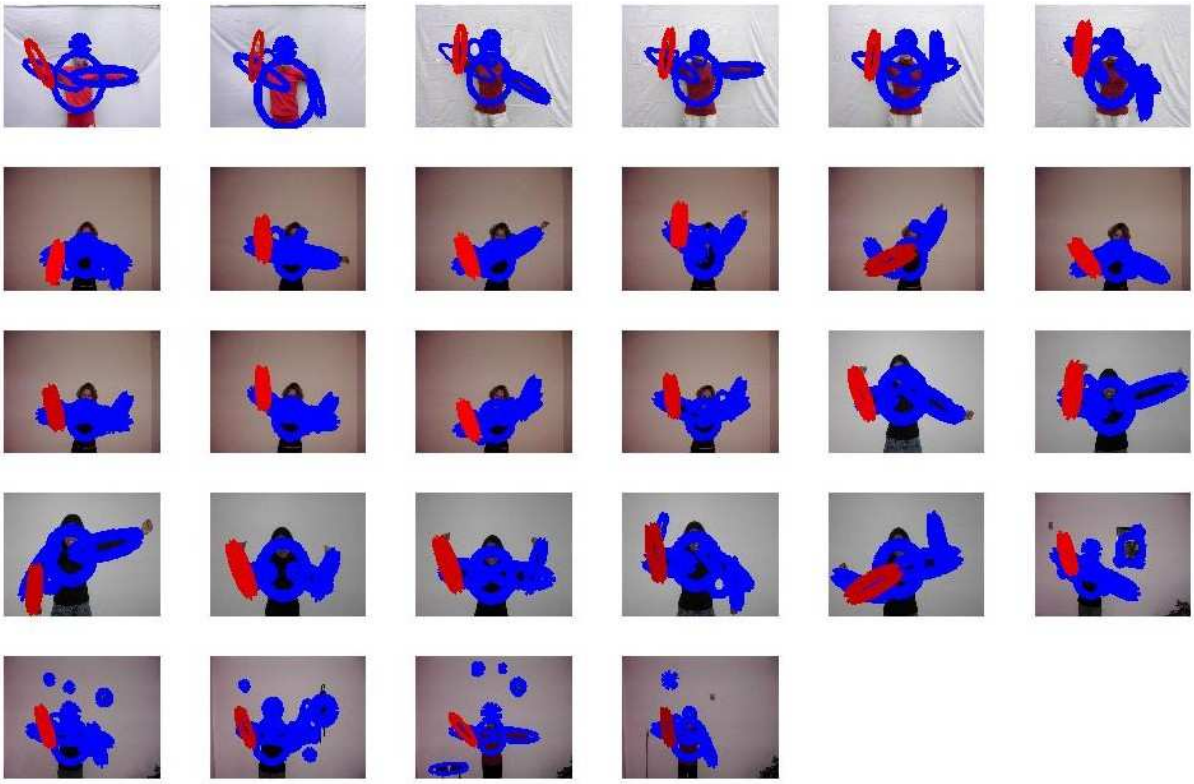


Figure 7.13: The lower right arm part in the resulting model, shown in red.

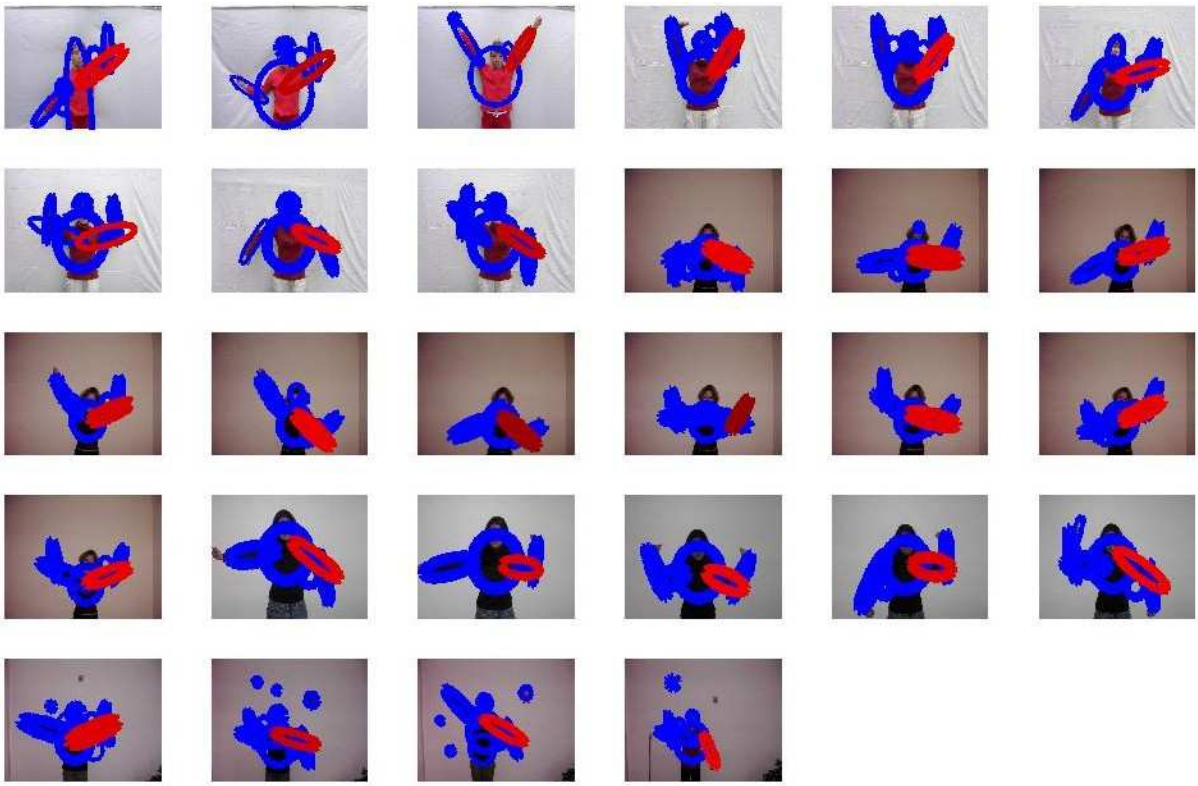


Figure 7.14: The upper left arm part in the resulting model, shown in red.

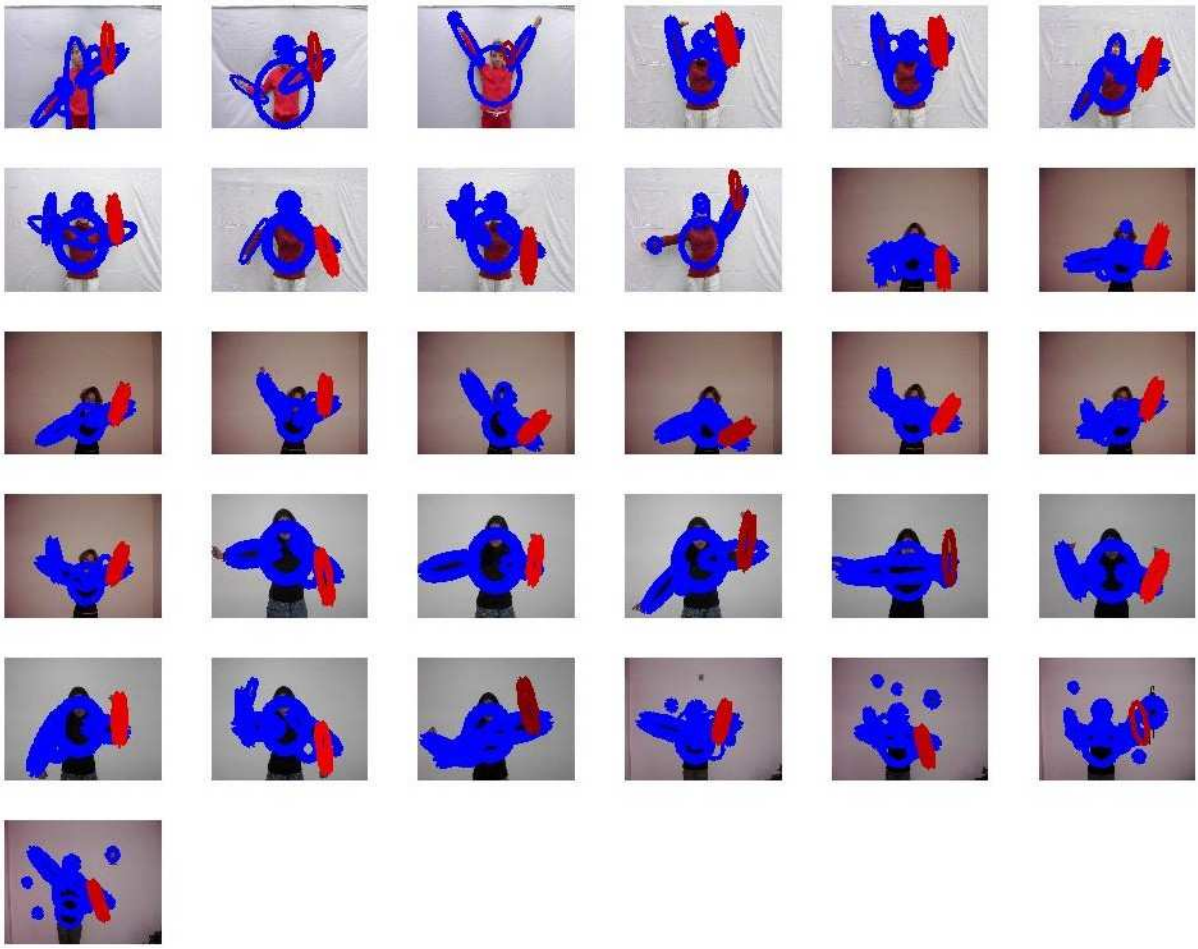


Figure 7.15: The lower left arm part in the resulting model, shown in red.

Chapter 8

Limitations

Our system consists of three main components: feature detection, matching, and model construction (includes part extraction and part relation recovery). The most important component is the matching component, since the quality of the extracted parts depends highly on the quality of the matching. However, matching two exemplars without relying on appearance is a difficult task. We have to deal with noise both in the form of spurious or missing parts, as well as incorrectly segmented parts. We aim to minimize the effects of the noise by removing some distracting features and choosing the largest connected component in the hope that it contains the object we are modeling. Though such measures do help, our system cannot correctly deal with large amounts of noise.

Most of the detected features need to correspond to object parts and need to be correctly segmented (located in appropriate positions for the correctness of the perceptual grouping stage and having appropriate masses for the EMD matching). Since we are working with a global connected component embedding and matching, all the selected features (the features in the largest connected component) are matched. If the majority of such features are noisy, the matching results will not be sufficiently consistent to recover the parts in the next stage. The matching algorithm is not scale invariant, since both the blob masses and the distances between them are functions of the absolute blob parameters. Moreover, since we choose to work with

a greedy EMD algorithm and not a global optimization EMD algorithm, correct flows are computed only given a sufficently close alignment of our input graphs in the first place. For our experiments, a sufficiently good alignment is achieved through the horizontal symmetry alignment where features from a given side of the people in each exemplar image are aligned together (i.e., features on the right side of the body in one exemplar are aligned with features on the right side of the body in another exemplar, and the same for the left side). This step solves both the initial alignment problem and the problem of ambiguity between the left and right sides in case of matching humans. However, it can be seen that our system often makes the mistake of matching a blob corresponding to the head with a blob located on the side or the bottom of the torso. This result is a direct consequence of the fact that we are not using any information about the actual Euclidean positions of the features in the images, but using distances from the perceptual grouping stage instead. Though such a choice provides our system with articulation invariance, it introduces additional ambiguity.

Chapter 9

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. 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.

For future work, we plan to apply machine learning techniques to recover optimal perceptual grouping parameters. As mentioned in Chapter 8, incorrect segmentation is a potential problem for our system since it can cause the mass conservation constraints to be violated. We plan to explore methods for better feature recovery, for example, using a blob we obtain

currently to initialize an active contour that better converges on the part's boundary. Since matching is the major component of our system, further research into many-to-many matching techniques is needed, allowing the system to deal with larger amounts of noise. Other techniques for part recovery based on the matching results will be examined, e.g., techniques that do not rely on the extra step of feature embedding, such as correlation clustering. In the future, we plan to apply our method to model recovery in other domains. Our ultimate goal is to study object recognition techniques adapted to the part-based decompositional models we have recovered in this work.

Bibliography

[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., Finding and Tracking People From the Bottom Up, *IEEE CVPR*, 2003.

[5] Ramanan, D. and Forsyth, D.A., Using Temporal Coherence to Build Models of Animals, *ICCV*, 2003.

[6] Ioffe, S. and Forsyth, D.A., Human tracking with mixtures of trees, *ICCV*, 2001.

[7] M. Weber, M. Welling, and P. Perona, Unsupervised Learning of Models for Recognition, *ECCV*, 2000.

[8] M. Weber, M. Welling, and P. Perona, Towards Automatic Discovery of Object Categories , *CVPR*, 2000.

[9] Fergus, R., Perona, P., and Zisserman, A., Object Class Recognition by Unsupervised Scale-Invariant Learning, *IEEE CVPR*, 2003.

[10] P. Felzenszwalb and D. Huttenlocher, Efficient Matching of Pictorial Structures, *IEEE CVPR*, 2000.

[11] 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.

[12] Marr. D. and Nishihara, H.K., Representation and recognition of the spatial organization of three dimensional shapes, *Proc. of Royal Soc. of London*, 1978.

[13] Brooks, R.A., Model-Based Three Dimensional Interpretations of Two Dimensional Images, *IEEE PAMI*, March 1983.

[14] I. Biederman, Recognition by components: A theory of human image understanding, *Psychological Review*, 94, pp. 115-147, 1987.

[15] Jepson, A.D., Fleet, D.J., and Black, M.J., A layered motion representation with occlusion and compact spatial support, *ECCV*, 2002.

[16] D. Lowe, Object recognition from local scale invariant features, *ICCV*, 1999.

[17] M. Turk and A. Pentland, Face recognition using eigenfaces, *CVPR*, 1991.

[18] T. F. Cootes, D. H. Cooper, C. J. Taylor, and J. Graham, A trainable method of parametric shape description, *BMVC*, 1991.

[19] Viola, P., Jones, M.J., and Snow, D., Detecting Pedestrians Using Patterns of Motion and Appearance, *ICCV*, 2003.

[20] Rubner, Y, Tomasi, C., and Guibas, L.J., A metric for distributions with applications to image databases, *ICCV*, 1998.

[21] Cohen, S. and Guibas, L.J., The Earth Mover's Distance under Transformation Sets, *ICCV*, 1999.

[22] Demirci, M.F., Shokoufandeh, A. Dickinson, S., Keselman, Y., Bretzner, L., Many-to-Many Feature Matching Using Spherical Coding of Directed Graphs, *ECCV*, 2004.

[23] Matoušek, J., On Embedding Trees into Uniformly Convex Banach Spaces, *Israel J. of Mathematics*, Volume 237, 1999.

[24] Y. Keselman, A. Shokoufandeh, M. Demirci, and S. Dickinson, Many-to-Many Graph Matching via Metric Embedding, *CVPR*, 2003.

[25] Y. Keselman and S. Dickinson, Generic Model Abstraction from Examples, *Proceedings, IEEE Conference on Computer Vision and Pattern Recognition*, Kauai, Hawaii, December, 2001.

[26] Xiaoyi Jiang, Andreas Müunger, and Horst Bunke, On Median Graphs: Properties, Algorithms, and Applications, *PAMI*, 2001.

[27] Lindeberg, T. and Bretzner, L., Real-time scale selection in hybrid multi-scale representations, *Proc. Scale-Space*, 2003.

[28] Tenenbaum, J.B., de Silva, V., and Langford, J.C., A Global Geometric Framework for Nonlinear Dimensionality Reduction, *Science*, 290 (5500), 2000.

[29] Ng, A.Y., Jordan, M.I., and Weiss, Y., On Spectral Clustering: Analysis and Algorithm, *NIPS*, 2001.

[30] Jianbo Shi and Jitendra Malik, Normalized Cuts and Image Segmentation, *PAMI*, 2000.

[31] Guillaume Bouchard and Bill Triggs, A Hierarchical Part-Based Model for Visual Object Categorization, *International Workshop on Object Recognition*, Taomina, Sicily, Italy, October 2004.

[32] Joachim Utans, Learning in Compositional Hierarchies: Inducing the Structure of Objects from Data, *Advances in Neural Information Processing Systems*, Vol. 6, 1994.

[33] D. Anguelov, D. Koller, H.-C. Pang, P. Srinivasan, S. Thrun, Recovering Articulated Object Models from 3D Range Data *UAI*, 2004.

[34] Taycher, L., Fisher, J. W., and Darrell, T., Recovering Articulated Model Topology from Observed Motion, *Proc. IEEE Workshop on Statistical Methods in Video Processing*, 2002.

[35] Gyorgy Dorko and Cordelia Schmid, Object Class Recognition Using Discriminative Local Features, *PAMI*, 2004.

[36] Svetlana Lazebnik,Cordelia Schmid, and Jean Ponce, Affine-Invariant local descriptors and neighborhood statistics for texture recognition, *ICCV*, 2003.

[37] Pierre Maurel and Guillermo Sapiro, Dynamic Shapes Average, *Tech Report 1924, Institute for Mathematics and its Applications, University of Minesota*, May 2003.

[38] Xu, Y.W., Saber E., Tekalp A.M., Dynamic learning from multiple examples for semantic object segmentation and search, *Computer Vision and Image Understanding*, 2004.

[39] Nicolae Duta, Anil K. Jain, Marie-Pierre Dubuisson-Jolly, Automatic construction of 2D shape models, *PAMI*, 2001.

[40] Alexandre R.J. François and Gérard G. Medioni, Generic Shape Learning and Recognition, *Proceedings of the International Workshop on Object Representations in Computer Vision*, 1996.

[41] Marcello Pelillo, Kaleem Siddiqi, Steven W. Zucker, Matching Hierarchical Structures Using Association Graphs, *PAMI*, 1999.

[42] Ueda N. and Suzuki S., Learning visual models from shape contours using multiscale convex/concave structure matching, *PAMI*, 1993.

[43] Jonathan H. Connell and J. Michael Brady, Generating and Generalizing Models of Visual Objects, *Artificial Intelligence*, 31(2), pp. 159-183, 1987.

[44] J. Segen, Learning graph models of shape, *ICML*, 1988.

[45] H. Nishida, S. Mori, An Algebraic Approach to Automatic Construction of Structural Models, *PAMI*, 1993.

[46] K. Cho, S.M. Dunn, Learning Shape Classes, *PAMI*, 1994.

[47] S. Kosinov and T. Caelli, Inexact multisubgraph matching using graph eigenspace and clustering models, *Proceedings of SSPR/SPR*, volume 2396, pages 133-142, Springer, 2002.

[48] Thomas B. Sebastian, Philip N. Klein, Benjamin B. Kimia, Recognition of Shapes by Editing Shock Graphs, *ICCV*, 2001.

[49] D. Macrini, A. Shokoufandeh, S. Dickinson, K. Siddiqi, and S. Zucker, Spectral Methods for View-Based 3-D Object Recognition using Silhouettes, *Proceedings, Joint IAPR International Workshop on Syntactical and Structural Pattern Recognition*, Windsor, ON, August, 2002.