Joint Shape Segmentation with Linear Programming

Qixing Huang Vladlen Koltun Leonidas Guibas

Stanford University

Abstract

We present an approach to segmenting shapes in a heterogenous shape database. Our approach segments the shapes jointly, utilizing features from multiple shapes to improve the segmentation of each. The approach is entirely unsupervised and is based on an integer quadratic programming formulation of the joint segmentation problem. The program optimizes over possible segmentations of individual shapes as well as over possible correspondences between segments from multiple shapes. The integer quadratic program is solved via a linear programming relaxation, using a block coordinate descent procedure that makes the optimization feasible for large databases. We evaluate the presented approach on the Princeton segmentation benchmark and show that joint shape segmentation significantly outperforms single-shape segmentation techniques.

CR Categories: I.3.5 [Computing Methodologies]: Computer Graphics—Computational Geometry and Object Modeling;

Keywords: shape segmentation, shape correspondence, linear programming

Links: DL PDF

1 Introduction

Shape segmentation is a fundamental problem in shape analysis. Many shape processing and 3D modeling applications benefit from automatic segmentation of shapes into components that appear natural [Shamir 2008; Chen et al. 2009; Chaudhuri et al. 2011]. Classical shape segmentation techniques analyze the geometric structure of individual shapes in order to detect parts or part boundaries. A variety of geometric features have been investigated, but no single feature or collection of features is known to produce high-quality results for all classes of shapes [Chen et al. 2009]. The underlying difficulty is that a perceptually natural segmentation of a shape is often the result of prior familiarity with other similar shapes and their function. The surface geometry of an individual shape may lack sufficient cues to identify all parts that would be perceived as meaningful to a human observer. Alternatively, a shape may contain strong geometric features within a single perceived part. This can mislead algorithms that consider individual shapes in isolation [Golovinskiy and Funkhouser 2009; Kalogerakis et al. 2010].

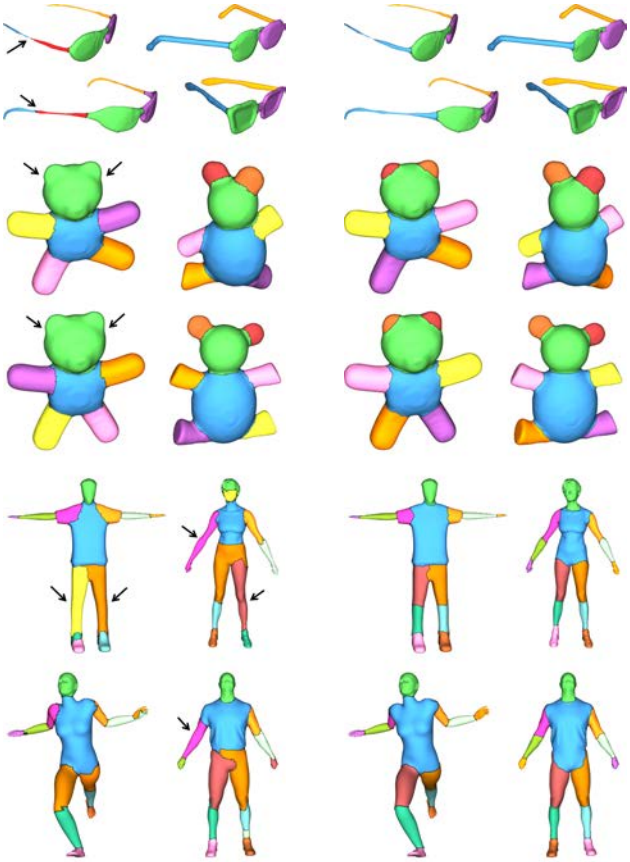


Figure 1: *Comparison of single-shape segmentation (left) and joint shape segmentation (right) on models from the Princeton segmentation benchmark [Chen et al. 2009]. Each segmentation on the left was produced by the top-performing algorithm in the benchmark for that shape. The segmentations on the right were produced by our approach, which jointly optimized segmentations and correspondences across the entire benchmark dataset. The new approach was able to identify meaningful parts despite extraneous geometric cues (top) and low saliency (middle, bottom).*

In this paper, we present an approach to shape segmentation that jointly analyzes a database of shapes. The approach optimizes segmentations on all shapes together with segment-level correspondences between similar shapes. By considering multiple shapes in concert, our approach is able to identify meaningful parts despite the lack of strong geometric cues on a particular shape. Likewise, the approach is able to identify coherent single parts even when the geometry of the individual shape suggests the presence of multiple segments. This is illustrated in Figure 1. Some eyeglass models (top, left) contain extraneous geometric cues that lead to over-segmentation by approaches that consider single shapes in isolation; our approach correctly detects the presence of a single logical part (the temple) due to counterpart segments in similar shapes in the database. In Figure 1 (middle), the ears of some of the teddy

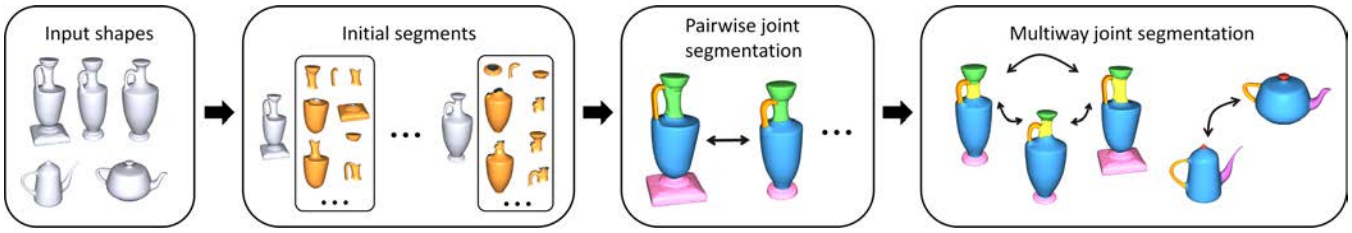


Figure 2: *Overview of our approach. In the first stage, we produce a set of initial segments for each shape. In the second stage, each pair of shapes is jointly segmented in order to identify similar shapes. In the third stage, a global optimization is performed over segmentations of all shapes together with correspondences between similar shapes.*

models have low saliency and are missed by single-shape segmentation; our approach (middle, right) identifies these parts due to corresponding salient parts on other shapes. In Figure 1 (bottom), human shapes with straight limbs have insufficient geometric cues to identify the upper and lower limbs; our approach (bottom, right) is able to recognize these parts due their prominence in other poses of the same or related shapes.

The key idea of our approach is to jointly optimize over segmentations of individual shapes and correspondences between segments on different shapes. This is difficult due to the exponential number of possible segmentations and correspondences. In order to make the joint segmentation problem computationally feasible, we first formulate it as an integer quadratic program. We then linearize the objective so as to obtain a linear programming relaxation. The resulting linear program is still computationally impractical for large databases. We thus present a block coordinate descent procedure that decomposes the program into a large number of small linear programs defined over individual shapes and pairs of shapes.

We evaluate the presented approach on the complete Princeton shape benchmark [Chen et al. 2009] and compare it to leading single-shape segmentation algorithms, as well as to the supervised approach of Kalogerakis et al. [2010]. Our results demonstrate that joint analysis produces better segmentations than the single-shape algorithms and is able to achieve comparable performance to the supervised approach, without requiring manually segmented training data.

1.1 Background

A survey of shape segmentation techniques is provided by Shamir [2008]. A large number of approaches have been developed for decomposing a single shape into parts [Shapira et al. 2008; Golovinskiy and Funkhouser 2008; Simari et al. 2009]. Such approaches suffer when the individual shape does not provide sufficient geometric cues to distinguish its meaningful parts, or when strong extraneous geometric features are present. In a recent evaluation, no segmentation algorithm performed well across all tested datasets [Chen et al. 2009].

To overcome the limitations inherent in single-shape analysis, researchers have turned to data-driven techniques that utilize information from multiple shapes in order to segment a given shape. Data-driven techniques are well-established in image processing: top-performing image segmentation algorithms all utilize manually segmented training sets [Shotton et al. 2009]. This *supervised* approach to segmentation has recently been extended to 3D shapes by Kalogerakis et al. [2010], who demonstrated significant improvement over single-shape segmentation algorithms. However, supervised approaches require a substantial number of manually segmented training shapes, in order to observe multiple examples for many kinds of recognizable shapes. Our approach does not require such training data.

A related line of work in shape segmentation aims to produce *consistent* segmentations of multiple shapes, for applications such as part-based 3D modeling [Kraevoy et al. 2007; Golovinskiy and Funkhouser 2009; Shapira et al. 2010; Xu et al. 2010]. Existing approaches to consistent segmentation either make restrictive assumptions on the set of shapes or ensure consistency by means of shape alignment. While Golovinskiy and Funkhouser [2009] observe that consistent segmentation can produce better individual segmentations, existing algorithms are limited both computationally and in their ability to treat heterogenous shape libraries. Our approach is designed to treat large shape libraries without prior correspondences or alignments.

In an independent work published in this issue, Sidi et al. [2011] describe an unsupervised co-segmentation algorithm that segments a set of shapes from a given shape class, obtaining a consistent segmentation throughout. However, this technique does not deal with heterogenous shape libraries.

2 Overview

Our joint shape segmentation pipeline proceeds in three stages, illustrated in Figure 2. In the first stage, we compute a set of initial segments for each input shape. Subsets of these initial segments form possible segmentations of each shape. In the second stage, we perform joint segmentation of each pair of input shapes in order to identify similar shapes. The third stage performs multiway joint segmentation by optimizing segmentations of all shapes and mappings between segmentations of similar shapes.

Initial segments. To generate initial segments, we first decompose each shape into patches. This decomposition is inspired by the computation of superpixels for image segmentation [Ren and Malik 2003]. Initial segments are generated by collecting distinct segments formed by randomized clustering of patches [Golovinskiy and Funkhouser 2008]. For efficiency in later steps, we prune redundant initial segments based on a popularity measure over the dataset.

Pairwise joint segmentation. In the second stage, we perform pairwise joint segmentation of each pair of shapes in the database. The main goal of this step is to identify pairs of similar shapes, so as to alleviate the computational burden on the third stage of the approach by identifying shapes in the database that should exchange information. To derive an algorithm for pairwise joint segmentation, we define an objective that optimizes segmentation quality on each shape alongside correspondence strength between the shapes. We then formulate an integer quadratic program that optimizes this objective over the initial segments for each shape and over possible correspondences between pairs of initial segments. This program is then relaxed to a linear program by relaxing the domains of the variables and linearizing the objective. The resulting linear program is sparse and can be solved with standard optimization packages.

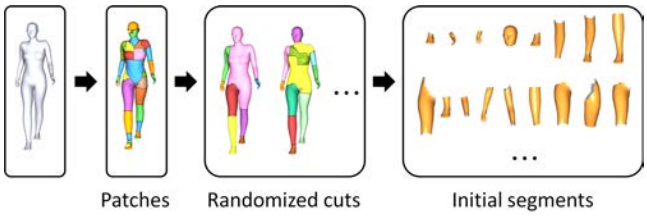


Figure 3: *Initial segment generation. First, each shape is partitioned into patches. Next, a large number of potential segments are generated by performing randomized cuts. The most promising of these are collected to form the set of initial segments.*

Multiway joint segmentation. In the third stage, we perform a global optimization over the entire shape dataset. The objective at this stage integrates individual segmentation quality with consistency of segmentations across pairs of similar shapes. We again formulate the optimization as an integer program that is solved via a linear programming relaxation. The resulting linear program is infeasible for large datasets due to its size. We thus present a block coordinate descent procedure that decomposes the program into a large number of small linear programs defined over individual shapes and individual pairs of similar shapes.

In summary, we show how the context provided by a library of unsegmented shapes allows improved segmentations of individual shapes in the library through a novel optimization that jointly extracts consistent part structure across the collection.

3 Segment Generation

Our joint segmentation pipeline begins by generating a set of initial segments for each shape in the database. The goal of this stage is to produce an overcomplete set of initial segments. The final segmentation of each shape will be given by a subset of these segments, identified by the optimization procedure described in Section 5.

Initial segments are generated in three steps, illustrated in Figure 3. First, we partition each shape into patches. These patches, which serve as building blocks for initial segments and for the final segmentation, are used to balance segmentation quality and computational cost. Second, we run randomized cuts [Golovinskiy and Funkhouser 2008] on these patches to form a large number of potential initial segments, which are connected subsets of patches. Third, for efficiency in later steps, we weigh each potential segment and prune segments with low weight. The remainder of this section describes each of these three steps in detail.

Patches. Each database shape W_i is independently partitioned into a set of patches $\mathcal{P}_i$ by performing normalized cuts [Shi and Malik 2000; Golovinskiy and Funkhouser 2008]. To further align patch boundaries with concave shape features, we use fuzzy cuts [Katz and Tal 2003]. We generate a set of 50 patches per shape, which we have found to be sufficient for the datasets used in our experiments.

Initial segments. Given a set of patches, we generate a large superset of potential initial segments. This is done by generating a large number of randomized segmentations on the patches. We let the target number of segments k vary from 2 to 10. For each k, we perform 100 randomized segmentations using randomized cuts over patches [Golovinskiy and Funkhouser 2008]. Consider the set $\bar{\mathcal{I}}_i$ of distinct segments generated by this process for the shape W_i.

The optimization problems derived in Sections 4 and 5 will be defined over a subset of these potential initial segments, $\mathcal{I}_i \subset \bar{\mathcal{I}}_i$. Due to the computational demands of the optimization procedure, we are interested in minimizing the size of $\mathcal{I}_i$ by identifying the most "promising" potential initial segments to be passed on to the optimization procedure for consideration. This is done by ranking all segments in $\bar{\mathcal{I}}_i$ and retaining only the most highly ranked.

For each segment s, let $r(s)$ denote the number of times this segment has been generated, across all randomized segmentations for the shape W_i. A natural approach would be to rank the segments by their repetition count r. However, this measure can overlook useful segments that are not salient on W_i and thus have low repetition count. Instead, we define a more global measure w_s of each segment's potential usefulness. The measure w_s takes into account not only the repetition count of s over the segmentations of W_i, but also the repetition counts of similar segments on other database shapes.

In order to define the measure w_s, we consider a shape distance measure $d(\cdot,\cdot)$ that evaluates the geometric similarity of two segments. The distance d factors out anisotropic scale variations and is described in detail in Appendix A. Using this distance, we define w_s by considering the repetition counts of the most similar segments to s on all database shapes $\{W_1,\ldots,W_n\}$:

$$w_s = \sum_{j=1}^{n} w_{(s,s_j^*)} r(s_j^*), \tag{1}$$

where $s_j^* = \arg\min_{s' \in \bar{\mathcal{I}}_j} d(s,s')$ is the most similar segment to s in the set $\bar{\mathcal{I}}_j$ and

$$w_{(s,s')} = \exp\left(-\frac{d^2(s,s')}{2\sigma^2}\right), \tag{2}$$

where σ is chosen as the median of distances between all pairs of most similar segments from all pairs of input shapes.

We retain the most highly weighted segments from $\bar{\mathcal{I}}_i$ as the set $\mathcal{I}_i$. In our implementation, the top 200 most highly weighted segments are retained. In addition, to make sure that $\mathcal{I}_i$ contains at least one complete segmentation of W_i, we include all segments from the random segmentation with the highest minimum segment weight. The resulting set $\mathcal{I}_i$ provides the set of initial segments for each shape W_i. The final segmentation of W_i will consist of a subset of segments from $\mathcal{I}_i$, identified by the optimization procedure described in Section 5.

4 Pairwise Joint Segmentation

The core of our joint segmentation pipeline is a new approach to optimizing a joint segmentation of two shapes. This approach will be extended to joint segmentation of sets of shapes in Section 5.

The optimization approach is described in three parts. In Section 4.1, we formally introduce pairwise joint segmentation as an optimization problem. The objective function is defined abstractly over all possible segmentations of the two shapes and over all possible mappings between pairs of segmentations, and is thus not immediately amenable to efficient computational solutions. In Section 4.2, we present an integer programming formulation that expresses the objective function in terms of indicator variables for initial segments and pairs of segments across shapes. This formulation has a tractable number of variables, but is still not practically feasible, due to the computational complexity of integer programming. In Section 4.3, we provide a linear programming relaxation of the integer program. This is done by linearizing the objective function, relaxing the domains of the variables, and specifying a simple

rounding procedure. The resulting linear program is sparse and can be efficiently solved with standard optimization packages.

4.1 Objective

Let W_1 and W_2 be two shapes and let $\mathcal{I}_1$ and $\mathcal{I}_2$ be the sets of initial segments generated for the two shapes as described in Section 3. Our approach optimizes over subsets of initial segments on each shape and finds segmentations $\mathcal{S}_1 \subset \mathcal{I}_1$ and $\mathcal{S}_2 \subset \mathcal{I}_2$, as well as mappings between them, that maximize both the quality of the individual segmentations and the consistency of the two segmentations under these mappings. Specifically, the objective for pairwise segmentation is defined as

$$\max_{\mathcal{S}_1 \subset \mathcal{I}_1,\ \mathcal{S}_2 \subset \mathcal{I}_2} \ \text{seg}(\mathcal{S}_1) + \text{seg}(\mathcal{S}_2) + \text{consistency}(\mathcal{S}_1, \mathcal{S}_2), \quad (3)$$

where $\text{seg}(\mathcal{S}_1)$ and $\text{seg}(\mathcal{S}_2)$ denote the segmentation score of $\mathcal{S}_1$ and $\mathcal{S}_2$, respectively, and $\text{consistency}(\mathcal{S}_1, \mathcal{S}_2)$ denotes their consistency score. Our approach aims to maximize the objective given in (3) over possible segmentations $\mathcal{S}_1$ and $\mathcal{S}_2$. We will now define the segmentation score and the consistency score.

Segmentation score. The segmentation score $\text{seg}(\mathcal{S}_i)$ is defined as the sum of normalized weights of all segments $s \in \mathcal{S}_i$:

$$\text{seg}(\mathcal{S}_i) = \sum_{s \in \mathcal{S}_i} \overline{w}_s = \sum_{s \in \mathcal{S}_i} \overline{\text{area}}(s) w_s, \quad (4)$$

where $\overline{\text{area}}(s) = \text{area}(s)/\text{area}(W_i)$ is the normalized area of segment s, and the unnormalized weight w_s is the measure of segment quality defined in Section 3. The weights are normalized by the relative area of the segment since otherwise the objective might be maximized by a decomposition into a large number of small segments.

Consistency score. The score $\text{consistency}(\mathcal{S}_1, \mathcal{S}_2)$ is defined in terms of directed mappings between segmentations:

$$\text{consistency}(\mathcal{S}_1, \mathcal{S}_2) = \sum_{ij \in \{12,21\}} \max_{\mathcal{M}_{ij}} \text{consistency}(\mathcal{M}_{ij}), \quad (5)$$

where $\mathcal{M}_{ij} \subset \mathcal{S}_i \times \mathcal{S}_j$ is a directed mapping from $\mathcal{S}_i$ to $\mathcal{S}_j$. We allow many-to-one mappings since corresponding segments can appear a different number of times on W_1 and W_2, as illustrated on the right. Since corresponding segments can be more numerous on either of the two shapes, we consider both mappings from $\mathcal{S}_1$ to $\mathcal{S}_2$ and mappings from $\mathcal{S}_2$ to $\mathcal{S}_1$.

The score $\text{consistency}(\mathcal{S}_1, \mathcal{S}_2)$ thus maximizes over all possible mappings from $\mathcal{S}_1$ to $\mathcal{S}_2$ and, independently, over inverse mappings from $\mathcal{S}_2$ to $\mathcal{S}_1$. The definition of $\text{consistency}(\mathcal{M}_{ij})$ is motivated by previous work on shape registration [Anguelov et al. 2005; Chang and Zwicker 2008] and aggregates correspondence terms for individual segments and for pairs of adjacent segments:

$$\text{consistency}(\mathcal{M}_{ij}) = \lambda \text{sim}(\mathcal{M}_{ij}) + \mu \text{adj}(\mathcal{M}_{ij}), \quad (6)$$

where λ and μ control the relative importance of the two terms. In all our experiments, we used $\lambda = 1$ and $\mu = 2$.

The term $\text{sim}(\mathcal{M}_{ij})$ evaluates the geometric similarity of individual corresponding segments in $\mathcal{M}_{ij}$:

$$\text{sim}(\mathcal{M}_{ij}) = \sum_{c \in \mathcal{M}_{ij}} \overline{\text{area}}(c) w_c = \sum_{c \in \mathcal{M}_{ij}} \overline{w}_c. \quad (7)$$

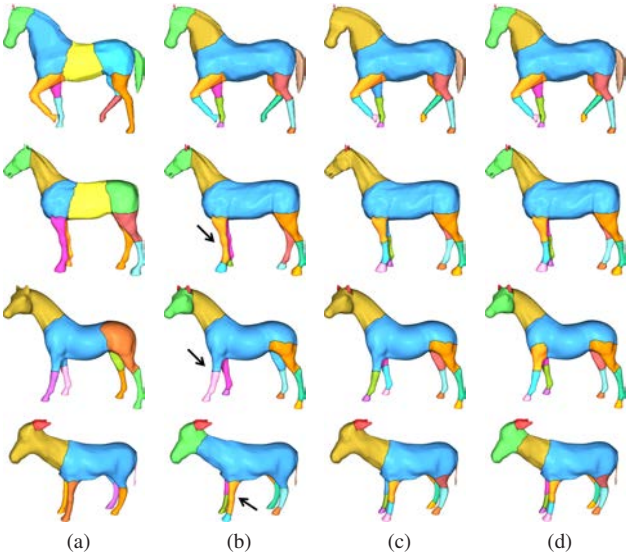


Figure 4: *The effect of different objective function terms, illustrated on a joint segmentation of four models from the FourLeg category. In (a), the consistency score was omitted and only the segmentation score was optimized. In (b), the segmentation score was omitted and only the consistency score was optimized. In (c), the adjacency term was omitted from the consistency score. In (d), the complete objective was optimized.*

The normalized area of a pair $c = (s, s')$ is defined as $\overline{\text{area}}(c) = \overline{\text{area}}(s)$ and the weight w_c is defined as in (2).

The term $\text{adj}(\mathcal{M}_{ij})$ prioritizes mappings of pairs of adjacent segments to pairs of adjacent segments:

$$\begin{aligned} \text{adj}(\mathcal{M}_{ij}) &= \sum_{(c,c') \in \mathcal{A}_{ij}} \big(\overline{\text{area}}(c) + \overline{\text{area}}(c')\big) w_{(c,c')} \\ &= \sum_{(c,c') \in \mathcal{A}_{ij}} \overline{w}_{(c,c')}, \end{aligned} \quad (8)$$

where $\mathcal{A}_{ij}$ is the set of *adjacent pairs* in $\mathcal{M}_{ij} \times \mathcal{M}_{ij}$. A pair $\big((s_i, s_j), (s_i', s_j')\big)$ is said to be adjacent if the segments s_i and s_i' are adjacent on W_i and the segments s_j and s_j' are adjacent on W_j.

The weight $w_{(c,c')}$ is defined as

$$w_{(c,c')} = w_c w_{c'} w_{\text{pose}}(c, c'),$$

where $w_{\text{pose}}(c, c')$ is a pose term that evaluates the quality of the cut between c and c'. This term is defined in Appendix B.

Aggregating equations (3), (4), (6), and (8) leads to the full definition of the pairwise segmentation objective:

$$\max \sum_{s \in \mathcal{S}_1 \cup \mathcal{S}_2} \overline{w}_s + \sum_{ij \in \{12,21\}} \left(\lambda \sum_{c \in \mathcal{M}_{ij}} \overline{w}_c + \mu \sum_{(c,c') \in \mathcal{A}_{ij}} \overline{w}_{(c,c')} \right), \quad (9)$$

which is maximized over segmentations $\mathcal{S}_1 \in \mathcal{I}_1$ and $\mathcal{S}_2 \in \mathcal{I}_2$ and mappings $\mathcal{M}_{12}$ and $\mathcal{M}_{21}$ in both directions. The effect of different terms in the objective function is illustrated in Figure 4. In the next section we show how to express this optimization problem as an integer quadratic program.

4.2 Integer Programming Formulation

In order to formulate the optimization problem defined in Section 4.1 as an integer program, we introduce indicator variables for segments in $\mathcal{I}_1$ and $\mathcal{I}_2$. These variables are used to formulate integer segmentation constraints that restrict the chosen subsets $\mathcal{S}_1 \subset \mathcal{I}_1$ and $\mathcal{S}_2 \subset \mathcal{I}_2$ to be valid segmentations. Likewise, we introduce indicator variables for segment correspondences in $\mathcal{I}_1 \times \mathcal{I}_2$ and $\mathcal{I}_2 \times \mathcal{I}_1$, which are used to define integer mapping constraints that restrict the subsets $\mathcal{M}_{12} \subset \mathcal{I}_1 \times \mathcal{I}_2$ and $\mathcal{M}_{21} \subset \mathcal{I}_2 \times \mathcal{I}_1$ to be valid mappings between the segmentations.

Segmentation constraints. A subset $\mathcal{S}_i$ of segments from $\mathcal{I}_i$ is a valid segmentation if the segments in $\mathcal{S}_i$ cover every patch in $\mathcal{P}_i$ exactly once. In order to formulate this constraint, we introduce a binary indicator x_s for every segment $s \in \mathcal{I}_i$. Given a candidate segmentation $\mathcal{S}_i$, a segment indicator x_s for $s \in \mathcal{I}_i$ is defined to be $x_s = 1$ if $s \in \mathcal{S}_i$ and $x_s = 0$ otherwise. Using this definition, we can formulate the segmentation constraints for $\mathcal{S}_i$ as

$$\sum_{s \in \text{cover}(p)} x_s = 1 \qquad \forall p \in \mathcal{P}_i,$$

where $\text{cover}(p) \subset \mathcal{I}_i$ is the set of all segments that cover patch p. The segmentation constraints can be expressed in matrix notation as

$$A_i \mathbf{x}_i = 1, \tag{10}$$

where the vector $\mathbf{x}_i$ contains all segment indicators for segments in $\mathcal{I}_i$. The segmentation score $\text{seg}(\mathcal{S}_i)$ can also be reformulated in terms of $\mathbf{x}_i$ as

$$\text{seg}(\mathcal{S}_i) = \sum_{s \in \mathcal{I}_i} x_s \overline{w}_s = \mathbf{x}_i^{\mathrm{T}} \mathbf{w}_i^{\text{seg}}, \tag{11}$$

where the vector $\mathbf{w}_i^{\text{seg}}$ contains all normalized segment weights for segments in $\mathcal{I}_i$.

Mapping constraints. A subset $\mathcal{M}_{ij}$ of pairs from $\mathcal{C}_{ij} = \mathcal{I}_i \times \mathcal{I}_j$ is a valid mapping from $\mathcal{S}_i$ to $\mathcal{S}_j$ if the pairs in $\mathcal{M}_{ij}$ contain segments that are included in $\mathcal{S}_i$ and $\mathcal{S}_j$ and if each segment from $\mathcal{I}_i$ is included in at most one pair in $\mathcal{M}_{ij}$. In order to formulate these constraints, we introduce a binary indicator y_c for every pair $c \in \mathcal{C}_{ij}$. Given a candidate mapping $\mathcal{M}_{ij}$, a pair indicator y_c for $c \in \mathcal{C}_{ij}$ is defined to be $y_c = 1$ if $c \in \mathcal{M}_{ij}$ and $y_c = 0$ otherwise. The above mapping constraints for $\mathcal{M}_{ij}$ can be expressed in terms of these variables via the following inequalities

$$\begin{aligned} \sum_{s' \in \mathcal{I}_j} y_{(s,s')} &\leq x_s \qquad \forall s \in \mathcal{I}_i \\ y_{(s,s')} &\leq x_{s'} \qquad \forall (s, s') \in \mathcal{C}_{ij} \end{aligned}$$

The mapping constraints can also be expressed in matrix notation as

$$\begin{aligned} B_{ij} \mathbf{y}_{ij} &\leq D_{ij} \mathbf{x}_i \\ B'_{ij} \mathbf{y}_{ij} &\leq D'_{ij} \mathbf{x}_j \end{aligned} \tag{12}$$

where the vector $\mathbf{y}_{ij}$ contains all pair indicators for pairs in $\mathcal{C}_{ij}$. The consistency score $\text{consistency}(\mathcal{M}_{ij})$ can now be reformulated in terms of $\mathbf{y}_{ij}$ as

$$\text{consistency}(\mathcal{M}_{ij}) = \lambda \mathbf{y}_{ij}^{\mathrm{T}} \mathbf{w}_{ij}^{\text{cor}} + \mu \sum_{(c,c') \in \mathcal{A}_{ij}} y_c y_{c'} \overline{w}_{(c.c')}, \tag{13}$$

where the vector $\mathbf{w}_{ij}^{\text{cor}}$ contains all normalized pair weights for segment pairs in $\mathcal{C}_{ij}$, and $\mathcal{A}_{ij}$ is the set of adjacent pairs in $\mathcal{C}_{ij} \times \mathcal{C}_{ij}$.

Integer quadratic program. Aggregating equations (10), (11), (12), and (13) leads to the formulation of the pairwise joint segmentation problem as an integer quadratic program:

$$\max \sum_{i \in \{1,2\}} \mathbf{x}_i^{\mathrm{T}} \mathbf{w}_i^{\text{seg}} + \sum_{ij \in \{12,21\}} \left(\lambda \mathbf{y}_{ij}^{\mathrm{T}} \mathbf{w}_{ij}^{\text{cor}} + \mu \sum_{(c,c') \in \mathcal{A}_{ij}} y_c y_{c'} \overline{w}_{(c,c')} \right)$$

$$\begin{aligned} \text{s.t.} \quad & A_1 \mathbf{x}_1 = 1 && A_2 \mathbf{x}_2 = 1 \\ & B_{12} \mathbf{y}_{12} \leq D_{12} \mathbf{x}_1 && B_{21} \mathbf{y}_{21} \leq D_{21} \mathbf{x}_2 \\ & B'_{12} \mathbf{y}_{12} \leq D'_{12} \mathbf{x}_2 && B'_{21} \mathbf{y}_{21} \leq D'_{21} \mathbf{x}_1 \\ \text{and} \quad & x \in \{0, 1\} && \forall x \in \mathbf{x}_1, \mathbf{x}_2, \mathbf{y}_{12}, \mathbf{y}_{21} \end{aligned} \tag{14}$$

4.3 Linear Programming Relaxation

In order to relax the integer quadratic program (14) to a linear program, we need to linearize the objective function and to make the domains of the variables continuous. Our relaxation technique is motivated by convex relaxation algorithms for MAP estimation in Markov random fields [Kumar et al. 2009], which formulate a similar objective to (14).

To linearize the objective, we associate each adjacent pair (c, c') with a latent indicator $z_{(c,c')} = y_c y_{c'}$. Let $\mathbf{z}_{ij}$ be a vector of these indicators for all pairs in $\mathcal{A}_{ij}$, and let $\mathbf{w}_{ij}^{\text{adj}}$ be the vector of corresponding normalized weights. The objective in (14) can be rewritten as

$$\max \sum_{i \in \{1,2\}} \mathbf{x}_i^{\mathrm{T}} \mathbf{w}_i^{\text{seg}} + \sum_{ij \in \{12,21\}} \left(\lambda \mathbf{y}_{ij}^{\mathrm{T}} \mathbf{w}_{ij}^{\text{cor}} + \mu \mathbf{z}_{ij}^{\mathrm{T}} \mathbf{w}_{ij}^{\text{adj}} \right).$$

The constraints $z_{(c,c')} = y_c y_{c'}$ can also be converted into linear constraints using a variant of the marginalization strategy described by Wainwright et al. [2005]:

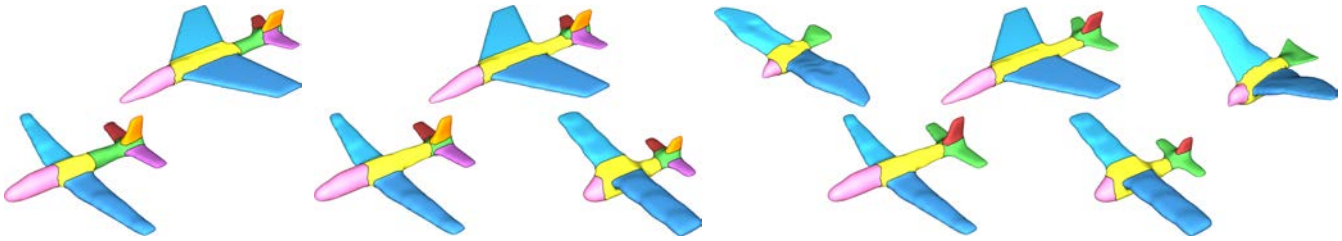
$$\sum_{s'_i \in \text{cover}(p_i)} \sum_{s'_j} z_{(c,(s'_i,s'_j))} \leq y_c \qquad \forall c \in \mathcal{C}_{ij}, p_i \in \mathcal{P}_i. \tag{15}$$

The derivation of these constraints is provided in Appendix C. They can be expressed in matrix notation as $E_{ij} \mathbf{z}_{ij} \leq F_{ij} \mathbf{y}_{ij}$. We thus obtain the following linear programming formulation for pairwise joint segmentation:

$$\max \sum_{i \in \{1,2\}} \mathbf{x}_i^{\mathrm{T}} \mathbf{w}_i^{\text{seg}} + \sum_{ij \in \{12,21\}} \left(\lambda \mathbf{y}_{ij}^{\mathrm{T}} \mathbf{w}_{ij}^{\text{cor}} + \mu \mathbf{z}_{ij}^{\mathrm{T}} \mathbf{w}_{ij}^{\text{adj}} \right)$$

$$\begin{aligned} \text{s.t.} \quad & A_1 \mathbf{x}_1 = \mathbf{1} && A_2 \mathbf{x}_2 = \mathbf{1} \\ & B_{12} \mathbf{y}_{12} \leq D_{12} \mathbf{x}_1 && B_{21} \mathbf{y}_{21} \leq D_{21} \mathbf{x}_2 \\ & B'_{12} \mathbf{y}_{12} \leq D'_{12} \mathbf{x}_2 && B'_{21} \mathbf{y}_{21} \leq D'_{21} \mathbf{x}_1 \\ & E_{12} \mathbf{z}_{12} \leq F_{12} \mathbf{y}_{12} && E_{21} \mathbf{z}_{21} \leq F_{21} \mathbf{y}_{21} \\ \text{and} \quad & 0 \leq x \leq 1 && \forall x \in \mathbf{x}_1, \mathbf{x}_2, \mathbf{y}_{12}, \mathbf{y}_{21}, \mathbf{z}_{12}, \mathbf{z}_{21} \end{aligned} \tag{16}$$

This linear program is sparse and can be efficiently solved using interior point methods [Boyd and Vandenberghe 2004]. After solving the LP, the values of the variables must be rounded to the integer values $\{0, 1\}$. We use a simple greedy rounding procedure that first rounds the segment indicators $\mathbf{x}_1, \mathbf{x}_2$ and then the remaining variables. Segment indicators are rounded iteratively. At each step of the iteration, we pick the highest unrounded segment indicator and snap its value to 1; the values of all overlapping segments are set to 0. This procedure is repeated until all segment indicators are



(a) (b) (c)

Figure 5: *Joint segmentation of (a) two, (b) three, and (c) five shapes from the Airplane and Bird categories. Incorporation of similar shapes improves segmentation performance.*

rounded. We then fix the values of the segment indicators and solve (16) again for updated values of the remaining variables. To round the values of pair indicators $\mathbf{y}_{12}, \mathbf{y}_{21}$, we consider each segment s such that $x_s = 1$. We then consider all pair indicators $y_{(s,s')}$ for pairs originating at s and snap the one with the highest value to 1 and the rest to 0. For segments s such that $x_s = 0$, all pair indicators $y_{(s,s')}$ for pairs originating at s are set to 0. The values of all latent indicators $\mathbf{z}_{12}, \mathbf{z}_{21}$ are determined by the rounded indicators $\mathbf{y}_{12}, \mathbf{y}_{21}$. This completes the rounding procedure.

5 Multiway Joint Segmentation

Joint segmentation of a set of shapes is performed by optimizing a generalization of program (16). The objective of the program is aggregated over pairs of similar shapes and the constraints are applied to each pair. Since the resulting optimization problem is impractical for the sizes of shape databases used in our experiments, we describe a block coordinate descent procedure that makes the optimization computationally feasible. In the remainder of this section, we formulate the generalization of (16) to multiple shapes, describe how pairs of similar shapes are identified, and derive the block coordinate descent procedure.

Linear program over shape sets. The generalization of program (16) to a set $\mathcal{W} = \{W_1, \ldots, W_n\}$ of shapes is formulated as follows:

$$\max \sum_{i=1}^{n} \mathbf{x}_i^{\mathrm{T}} \mathbf{w}_i^{\text{seg}} + \frac{n}{|\mathcal{E}|} \sum_{(i,j)\in\mathcal{E}} \left(\lambda \mathbf{y}_{ij}^{\mathrm{T}} \mathbf{w}_{ij}^{\text{cor}} + \mu \mathbf{z}_{ij}^{\mathrm{T}} \mathbf{w}_{ij}^{\text{adj}} \right)$$

$$\begin{aligned} \text{s.t.} \quad & A_i \mathbf{x}_i = \mathbf{1},\ \mathbf{0} \le \mathbf{x}_i \le \mathbf{1} \ \text{ for all } 1 \le i \le n \\ \text{and} \quad & B_{ij}\mathbf{y}_{ij} \le D_{ij}\mathbf{x}_i,\ B'_{ij}\mathbf{y}_{ij} \le D'_{ij}\mathbf{x}_j,\ E_{ij}\mathbf{z}_{ij} \le F_{ij}\mathbf{y}_{ij}, \\ & \mathbf{0} \le \mathbf{y}_{ij} \le \mathbf{1},\ \mathbf{0} \le \mathbf{z}_{ij} \le \mathbf{1} \ \text{ for all } (i,j) \in \mathcal{E} \end{aligned} \tag{17}$$

where $\mathcal{E}$ is the set of pairs (i, j) such that the shapes W_i and W_j are similar.

Similar shapes. Similar shapes are identified by performing pairwise joint segmentation between all pairs of shapes. To evaluate the similarity of two shapes W_i and W_j, we use the similarity score $\mathbf{y}_{ij}^{\mathrm{T}} \mathbf{w}_{ij}^{\text{cor}} + \mathbf{y}_{ji}^{\mathrm{T}} \mathbf{w}_{ji}^{\text{cor}}$, where $\mathbf{y}_{ij}$ and $\mathbf{y}_{ji}$ are the rounded optimized mapping indicators obtained by the program described in Section 4.3. This similarity score aggregates the weights of pairs of corresponding segments in $\mathcal{S}_i$ and $\mathcal{S}_j$, normalized by area. Due to the normalization, the similarity score ranges from 0 to 2. We define two shapes to be similar if their similarity score is greater than a similarity threshold ρ, which was set to 1 in our experiments. The effect of parameter ρ is examined in Section 6.

Block coordinate descent. Program (17) quickly becomes impractical as the number of shapes grows, due to its extensive memory requirements and the superlinear complexity of linear programming in practice. In order to optimize program (17) efficiently, we introduce a block coordinate descent procedure that solves the global optimization problem by a sequence of small constrained optimizations whose size is independent of the number of shapes in the database. Each step in the procedure optimizes a small subset of the variables while the remaining ones are fixed [Sontag and Jaakkola 2009]. In our formulation, each individual step optimizes over a segmentation of an individual shape or over a mapping between a pair of similar shapes. Since the objective function is convex, the procedure converges to the global optimum.

More formally, we wish to decompose the set of variables into segment indicators $\mathbf{x}_i$ for each individual shape W_i and pair indicators $\{\mathbf{y}_{ij}, \mathbf{z}_{ij}\}$ for each pair of similar shapes $(i, j) \in \mathcal{E}$. Such a decomposition is precluded by the constraints $B_{ij}\mathbf{y}_{ij} \le D_{ij}\mathbf{x}_i$ and $B'_{ij}\mathbf{y}_{ij} \le D'_{ij}\mathbf{x}_j$, which couple segment indicators and pair indicators. To decouple these variables in the constraint set, we replace the segment indicators $\mathbf{x}_i, \mathbf{x}_j$ with auxiliary pair indicators $\mathbf{x}_{ij}, \mathbf{x}_{ji}$ in the above constraints, and add terms to the objective that minimize the deviation of the auxiliary indicators from the corresponding segment indicators. Specifically, we reformulate program (17) as follows:

$$\max \sum_{i=1}^{n} \mathbf{x}_i^{\mathrm{T}} \mathbf{w}_i^{\text{seg}} + \frac{n}{|\mathcal{E}|} \sum_{(i,j)\in\mathcal{E}} (\lambda \mathbf{y}_{ij}^{\mathrm{T}} \mathbf{w}_{ij}^{\text{cor}} + \mu \mathbf{z}_{ij}^{\mathrm{T}} \mathbf{w}_{ij}^{\text{adj}} - \gamma \|\mathbf{x}_i - \mathbf{x}_{ij}\|^2)$$

$$\begin{aligned} \text{s.t.} \quad & A_i \mathbf{x}_i = 1,\ \mathbf{0} \le \mathbf{x}_i \le \mathbf{1} \ \text{ for all } 1 \le i \le n \\ \text{and} \quad & B_{ij}\mathbf{y}_{ij} \le D_{ij}\mathbf{x}_{ij},\ B'_{ij}\mathbf{y}_{ij} \le D'_{ij}\mathbf{x}_{ji},\ E_{ij}\mathbf{z}_{ij} \le F_{ij}\mathbf{y}_{ij}, \\ & A_i \mathbf{x}_{ij} = 1,\ \mathbf{0} \le \mathbf{x}_{ij} \le \mathbf{1},\ \mathbf{0} \le \mathbf{y}_{ij} \le \mathbf{1},\ \mathbf{0} \le \mathbf{z}_{ij} \le \mathbf{1} \\ & \text{for all } (i,j) \in \mathcal{E} \end{aligned} \tag{18}$$

Program (18) decouples the set of segment indicator variables $\mathbf{x}_i$ for each shape W_i from the set of pair indicator variables $\{\mathbf{x}_{ij}, \mathbf{y}_{ij}, \mathbf{z}_{ij}\}$ for each pair of similar shapes $(i, j) \in \mathcal{E}$ in the constraint set. Since programs (17) and (18) produce the same result when $\gamma \to \infty$, we increase γ from 10^{-3} to 10^5 in the course of the optimization. The individual steps of the optimization solve small linear quadratic programs (quadratic objective, linear constraints) whose size is independent of the number of shapes in the database. The steps are of two types: single shape segmentation optimization and pairwise mapping optimization.

A single shape segmentation optimization step fixes the variables $\{\mathbf{x}_{ij}, \mathbf{y}_{ij}, \mathbf{z}_{ij} | (i, j) \in \mathcal{E}\}$ and optimizes the variables $\mathbf{x}_i$ for a single shape W_i. This yields the following quadratic program:

$$\begin{aligned} \max_{\mathbf{x}_i} \quad & \mathbf{x}_i^{\mathrm{T}} \mathbf{w}_i^{\text{seg}} - \frac{n}{|\mathcal{E}|} \sum_{\{j | (i,j)\in\mathcal{E}\}} \gamma \|\mathbf{x}_i - \mathbf{x}_{ij}\|^2 \\ \text{s.t.} \quad & A_i \mathbf{x}_i = 1,\ \mathbf{0} \le \mathbf{x}_i \le \mathbf{1} \end{aligned}$$

A pairwise mapping optimization step fixes the variables $\{\mathbf{x}_i | 1 \leq i \leq n\}$ and optimizes the variables $\mathbf{x}_{ij}, \mathbf{y}_{ij}, \mathbf{z}_{ij}$ for a single pair of shapes $(i, j) \in \mathcal{E}$. The corresponding quadratic program is

$$\max_{\mathbf{x}_{ij}, \mathbf{y}_{ij}, \mathbf{z}_{ij}} \quad \lambda \mathbf{y}_{ij}^{\mathrm{T}} \mathbf{w}_{ij}^{\mathrm{cor}} + \mu \mathbf{z}_{ij}^{\mathrm{T}} \mathbf{w}_{ij}^{\mathrm{adj}} - \gamma \|\mathbf{x}_i - \mathbf{x}_{ij}\|^2$$

$$\text{s.t.} \quad B_{ij}\mathbf{y}_{ij} \leq D_{ij}\mathbf{x}_{ij},\ B'_{ij}\mathbf{y}_{ij} \leq D'_{ij}\mathbf{x}_{ji},\ E_{ij}\mathbf{z}_{ij} \leq F_{ij}\mathbf{y}_{ij},$$
$$A_i\mathbf{x}_{ij} = 1,\ \mathbf{0} \leq \mathbf{x}_{ij} \leq \mathbf{1},\ \mathbf{0} \leq \mathbf{y}_{ij} \leq \mathbf{1},\ \mathbf{0} \leq \mathbf{z}_{ij} \leq \mathbf{1}$$

The block coordinate descent procedure consists of alternating single shape optimization and pairwise optimization stages. In a single shape optimization stage the procedure optimizes the segmentations of all database shapes, for each shape in turn. In a pairwise optimization stage the procedure optimizes the mappings between all pairs of similar shapes, for each pair in turn. In practice, we find that the procedure converges in less than 10 iterations over all shapes and pairs of similar shapes. Since the number of variables in each optimization step is independent of the number of shapes, the block coordinate descent procedure is both fast and memory efficient.

6 Results

6.1 Experimental Setup

We have evaluated the presented approach on the Princeton segmentation benchmark [Chen et al. 2009]. The benchmark provides 19 categories of shapes, with 20 shapes in each category. For each shape, a set of human-generated segmentations are provided, which serve as a plausible ground truth for the dataset. On average, 11 human-generated segmentations are provided for each shape. The benchmark also provides segmentations produced by seven popular segmentation algorithms, all of which analyze each individual shape in isolation.

The Princeton segmentation benchmark evaluates segmentation algorithms against human-generated segmentations. To compare segmentations, we mainly use the Rand index measure [Rand 1971], which was employed in the benchmark and in subsequent work on supervised data-driven segmentation [Kalogerakis et al. 2010]. We have also tested other measures described by Chen et al. [2009] and found that the results are consistent with the Rand index measure, as shown in Figure 7.

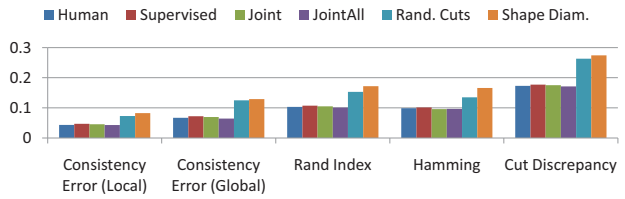


Figure 7: *Performance of different segmentation algorithms with respect to the evaluation criteria described by Chen et al. [2009].*

Given two segmentations of the same shape, the Rand index r measures the degree of agreement between them, in terms of the proportion of pairs of faces for which the two segmentations agree on whether to assign the faces to different segments or to the same segment. Following Chen et al. [2009], we report $1 - r$ as the segmentation score, thus lower values are better. The Princeton segmentation benchmark evaluates the output of each segmentation algorithm against the set of human-generated segmentations. The Rand index of a particular segmentation of a given shape is defined as the average Rand index of this segmentation with respect to all human-generated ground truth segmentations of the shape. The Rand index of the human-generated ground truth set for a given shape is the average Rand index of all human-generated segmentations of the shape (with respect to all other such segmentations). This implies that a segmentation generated by a given algorithm can have a lower Rand index than the human-generated ground truth set.

All tests were performed on a 2.4 GHz CPU with 12GB of RAM. Our implementation used the CVX package [Grant and Boyd 2011] to solve the linear programs. We have evaluated the presented joint segmentation approach in two conditions. In the first condition (Joint), joint segmentation was performed within each category separately. Thus, for each category, 20 shapes were optimized jointly. In the second condition (JointAll), the complete benchmark dataset of 380 shapes was optimized jointly. Figure 6 shows representative segmentations produced in the JointAll condition. The supplementary material includes the complete segmentation results on the benchmark.

In the Joint condition, the average processing time of the approach was 23 minutes per category, with highest processing time of 126 minutes (for the Human category) and lowest processing time of 10 minutes (for the Cup category). In the JointAll condition, the processing time for the complete database was 31 hours and 48 minutes. The total time spent in each of the three stages of the approach in the JointAll condition was as follows. In the first stage, 146 minutes were spent on computing patches and performing randomized cuts, and 253 minutes were spent on computing the weights w_s and pruning the sets of potential segments. In the second stage, the total time to perform pairwise joint segmentation for each pair of shapes was 636 minutes. The number of pairs of similar shapes identified in the second stage was 7130. In the third stage, multiway joint segmentation over the complete database took 876 minutes. The average running time for a single shape-wise optimization step in the block coordinate descent procedure was 1 millisecond and the average time for a single pairwise optimization step in the procedure was 0.71 seconds. In total, 10 iterations over all shapes and pairs of similar shapes were performed.

6.2 Parameters

The key parameters of our algorithm are the number of patches into which each shape is partitioned (set to 50 in our implementation), the number of initial segments constructed for each shape (set to 200), the similarity threshold ρ (set to 1), and the objective function weights λ and μ (set to 1 and 2, respectively). We have found that the performance of the algorithm is quite stable to changes in these parameters and that setting them was quite easy, as summarized below.

The number of patches generated for each shape balances segmentation quality and computational efficiency. We found that setting the number of patches to 2-3 times the estimated maximum number of segments per shape is sufficient. The precise setting of this parameter is not critical. In the Joint setting, using 25 patches instead of 50 increases the Rand index by only 0.5 percent while using 100 patches improves the Rand index by 0.3 percent, albeit at the cost of significantly greater computational burden on subsequent stages of the algorithm. Likewise, the performance of the algorithm is quite insensitive to the precise setting of the number of initial segments per shape. Setting this number to 100 instead of 200 increases the Rand index by 0.5 percent while using 400 initial segments improves the Rand index by only 0.4 percent.

There exist many possibilities for defining segment weights and correspondence weights. For example, we have tested the weight-

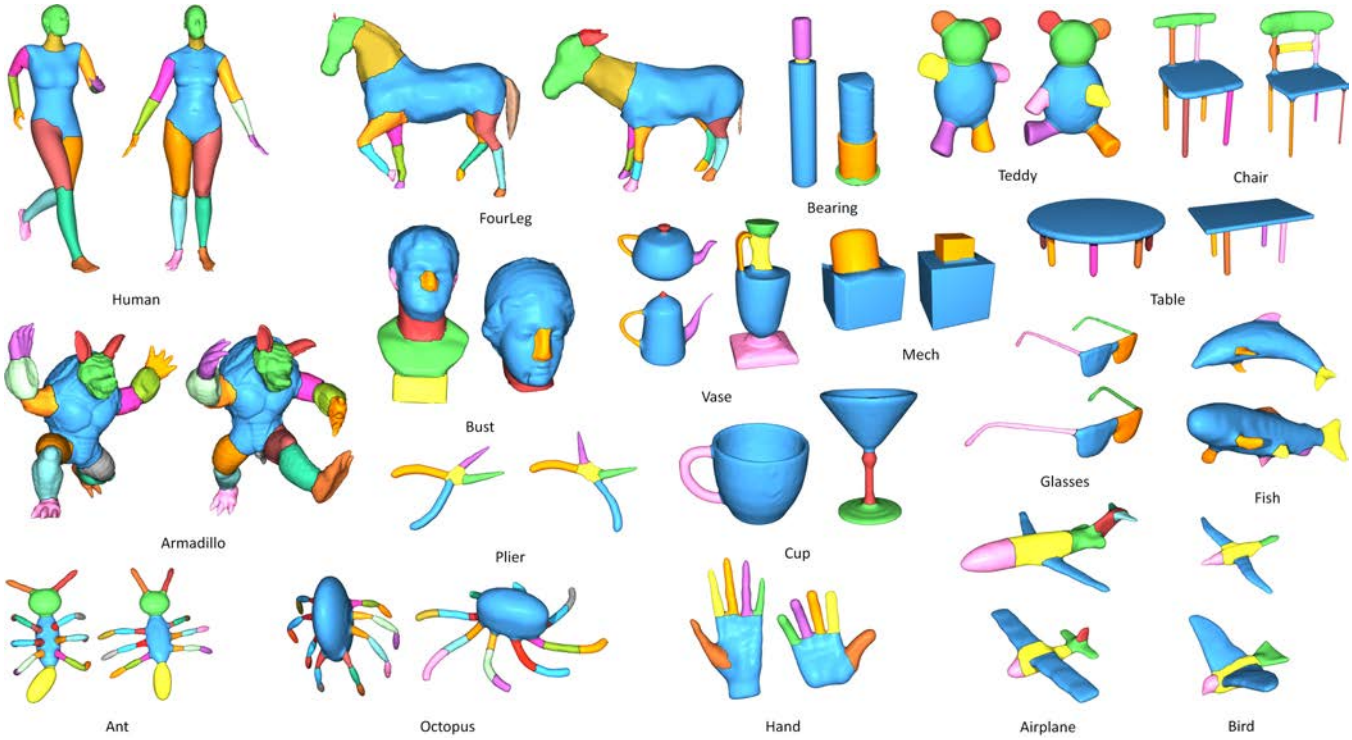


Figure 6: *Representative segmentations produced by our approach on the Princeton segmentation benchmark.*

ing schemes described by Kraevoy et al. [2007] and found that they can improve the Rand index by 0.1 and 0.3 percent in the Joint and JointAll conditions, respectively. However, these weighting schemes rely on a number of additional parameters. We thus retained the simple weighting schemes used in Section 4.

The objective function weights λ and μ were set by hand to rather natural values (1 and 2, respectively). We examined the performance of the method in extreme cases where some terms in the objective function are removed completely (see Figure 4). When the consistency score is omitted and only the segmentation score is used (Figure 4(a)), our approach reduces to single-shape segmentation; the average Rand index obtained on the complete benchmark is 16.3%. When the segmentation score is omitted and only the consistency score is used (Figure 4(b)), the approach produces segmentations with good correspondences but does not take into account the geometric complexity of individual segments; the corresponding Rand index is 17.4%. When the adjacency term is omitted from the consistency score (Figure 4(c)), the produced segmentations do not consider the articulation of corresponding segments on other shapes; the corresponding Rand index is 12.3%. When the complete objective is optimized (Figure 4(d)), the approach achieves an average Rand index of 10.1%. Finally, we analyzed the effect of the similarity threshold ρ, used to identify pairs of similar shapes that

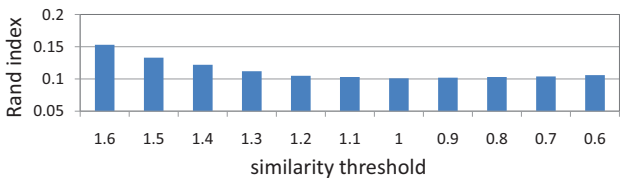


Figure 8: *The effect of the similarity threshold ρ.*

should exchange information (Section 5). The effect of the threshold is visualized in Figure 8, which plots the average Rand index in the JointAll condition, as a function of ρ. When the threshold is high ($\rho = 1.6$), only very similar shapes are selected for joint analysis and the joint segmentation reduces, in the limit, to single-shape segmentation. When the threshold is low ($\rho = 0.6$), the computational requirements of the procedure grow considerably and occasional spurious correspondences begin to degrade overall segmentation quality. The value $\rho = 1$ was used in all our experiments.

6.3 Analysis of Segmentations

Table 1 provides the main results of the evaluation on the Princeton segmentation benchmark. All results were obtained with the same set of parameters, listed in the first paragraph of Section 6.2. The joint segmentation approach produces significantly better results than single-shape segmentation and is competitive with supervised data-driven segmentation [Kalogerakis et al. 2010]. Among the seven single-shape segmentation approaches included in the benchmark, Table 1 only lists the performance of the leading two: shape diameter (SD, [Shapira et al. 2008]) and randomized cuts (RC, [Golovinskiy and Funkhouser 2008]); statistics for the other algorithms are provided in the original benchmark [Chen et al. 2009].

The segmentations produced by our approach in the JointAll condition were generally better or on par with those produced in the Joint condition. Segmentation performance was more similar in the two conditions on shape categories that already exhibit significant variability within the category. One example is the Armadillo category, which contains articulated armadillos in a broad range of poses. Joint segmentation of this category by itself (in the Joint condition) is sufficient to produce very accurate segmentations. Categories that benefitted most from the JointAll condition were Human and FourLeg, and Bird and Airplane. In each case, meaningful correspondences were established across categories, providing

	SD	RC	Supervised	Joint	JointAll	Human
Human	17.1	12.9	12.9	12.3	11.3	13.5
FourLeg	16.6	13.6	13.9	12.9	11.2	14.9
Armadillo	8.9	9.2	8.4	7.4	7.4	8.3
Teddy	5.9	4.6	3.2	3.3	3.3	4.9
Ant	2.0	2.6	2.2	2.4	2.4	3.0
Plier	37.1	11.2	9.0	7.5	7.5	7.1
Glasses	19.9	10.0	14.1	9.9	9.9	10.1
Bird	11.7	11.1	14.8	10.4	7.6	6.2
Fish	25.6	29.2	13.2	13.2	13.1	15.5
Mech	23.4	23.6	11.8	13.3	13.3	13.1
Bearing	12.1	12.7	17.6	11.3	11.3	10.4
Bust	28.8	23.6	22.2	19.5	19.8	22.0
Cup	35.1	21.2	9.9	11.2	11.2	13.6
Vase	23.6	12.7	17.1	13.5	13.2	14.4
Airplane	9.3	13.4	8.2	12.9	10.2	9.2
Chair	11.1	17.8	5.6	9.6	9.6	8.9
Table	17.8	38.1	6.6	6.6	6.6	9.3
Hand	19.2	9.0	11.2	13.2	13.1	9.1
Octopus	4.8	6.4	1.8	6.7	7.2	2.4
Average	**17.2**	**15.3**	**10.7**	**10.5**	**10.1**	**10.3**

Table 1: *Rand index scores for leading single-shape segmentation algorithms [Shapira et al. 2008; Golovinskiy and Funkhouser 2008], supervised segmentation [Kalogerakis et al. 2010], our approach, and human-generated segmentations. Lower values indicate closer similarity to human-generated ground truth.*

additional information that was exploited by the technique. For example, necks in the human models are not clearly identifiable as a distinct part by the geometry of human models themselves; by segmenting them jointly with quadruped shapes that have more salient necks, such as dogs and horses, the technique correctly identified necks as distinct segments.

The only category on which performance was significantly worse in the JointAll condition was Octopus. This is because some octopus models in the database have jointed tentacles that are similar to insect legs in the Ant category (Figure 6, lower left). Thus strong correspondences were established between these models, resulting in over-segmentation of octopus tentacles. This shows that the variability exploited by joint segmentation can also lead to confusion. Another evidence for this is the gradual decline in performance that can be observed when the similarity threshold is significantly lowered, as seen in Figure 8, although joint segmentation continues to produce superior results to single-shape segmentation even when the similarity threshold is disabled completely, indicating that the benefits of joint analysis outweigh the negative impact of spurious correspondences.

Comparison with supervised segmentation. We have also compared the presented technique to the supervised data-driven approach recently introduced by Kalogerakis et al. [2010]. Table 1 lists the quantitative performance of the supervised technique on the benchmark and Figure 9 provides a qualitative comparison. For the supervised approach, 12 out of the 20 models in each category were segmented manually. These 12 models were used as training data, with performance evaluated on the remaining 8. (This is the "SB12" condition described by Kalogerakis et al. [2010].) Our technique used no manually segmented training data.

The overall performance of the two approaches is similar, although their behavior on individual categories is often quite different. Supervised segmentation works particularly well when the variation in a given category is small and the features learned from the training segmentations map well to other models in the category. In contrast,

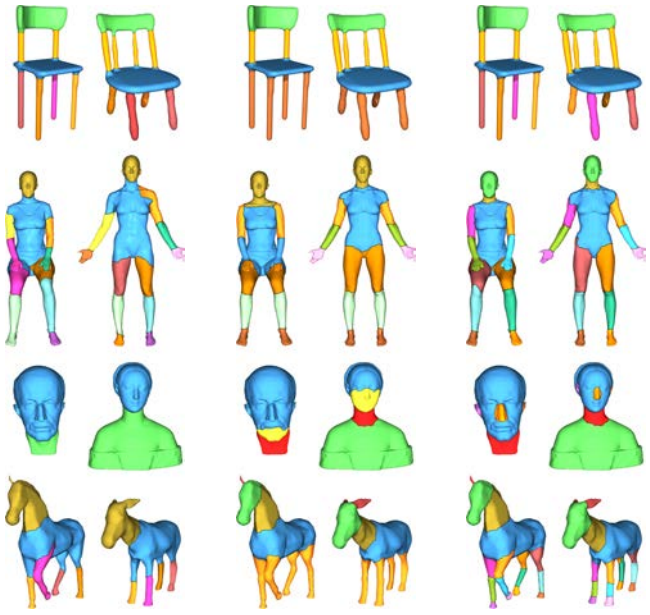


Figure 9: *Comparison with prior data-driven segmentation techniques. From left to right: consistent segmentation [Golovinskiy and Funkhouser 2009], supervised data-driven segmentation [Kalogerakis et al. 2010], our approach.*

our technique favors large variation in the input shapes, exploiting salient cuts on some models in order to identify less salient cuts on other models. This difference is illustrated by the Airplane category, which contains many highly similar shapes. The supervised approach outperforms our technique in this case, indicating that the training data provides more information than the variability within the dataset itself. On the other hand, our technique outperforms the supervised approach on categories that feature many geometrically dissimilar shapes, especially when the shapes are also articulated, as in the Bird and Glasses categories. Likewise, our technique outperforms the supervised approach on the Bust and Vase categories, which feature significant variation between the shapes.

Comparison with consistent segmentation. We have also compared the presented technique to the consistent segmentation approaches of Golovinskiy and Funkhouser [2009] and Xu et al. [2010]. Representative results are shown in Figures 9 and 10. The approach of Golovinskiy and Funkhouser relies on rigid alignment in order to establish correspondences. Thus the performance of the approach suffers when rigid alignment does not yield good results. The comparison with the approach of Xu et al. was performed on the Humanoids dataset provided by the authors, which contains 15 roughly humanoid shapes. Both approaches were initialized with the same set of patches. Figure 10 shows the segmentations produced for three of the models; the rest are provided in supplementary material. The approach of Xu et al. relies on the existence of strong cuts on all shapes in order to identify consistent segments. While this strategy is appropriate for the application of swapping segments for shape synthesis, it can easily overlook meaningful segments that are not demarcated by strong cuts in a given shape. In contrast, our technique finds segments based on average cut strength across the shapes and is able to more robustly identify natural parts.

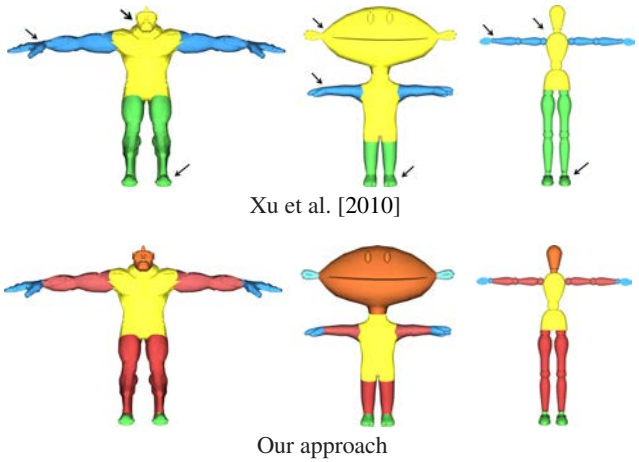


Figure 10: *Comparison with the approach of Xu et al. [2010] on the Humanoids dataset provided by the authors. Our technique successfully identifies the head, hands, and feet, which are missed by the prior approach.*

7 Conclusion

In this paper, we have demonstrated that performing joint segmentation on a set of shapes can significantly improve the segmentation quality of each individual shape. We have developed a joint segmentation technique that simultaneously optimizes segmentations of shapes and correspondences between similar shapes at the segment level. We have implemented the technique using efficient optimization procedures and have demonstrated its practicality on a standard benchmark.

A major limitation of the presented approach is that the segments in the final segmentation of each shape are generated from the initially computed patches. In other words, if an underlying segment is not captured well by the patches, then it is not present in the final segmentation. Furthermore, our method exploits variability in the database. If there is no variability, our approach reduces to single-shape segmentation. When there is variation, similar shapes exchange information, leading to better segmentations. However, as discussed in Section 6, variation in the input can also lead to confusion.

There are many avenues for future work. We would like to investigate alternative approaches to determining which pairs of shapes should directly communicate during the optimization. Likewise, we believe that both the computational efficiency and the segmentation performance of the approach can be further improved, perhaps with alternative optimization formulations and objective functions. More broadly, we believe that a variety of problems in geometry processing can benefit from joint shape analysis.

Acknowledgements

We are grateful to Mirela Ben-Chen, Siddhartha Chaudhuri, and Evangelos Kalogerakis for their comments on this paper. This work was supported in part by NSF grants 0808515 and 1011228, a Stanford-KAUST AEA grant, and a Stanford Graduate Fellowship.

References

ANGUELOV, D., SRINIVASAN, P., PANG, H., KOLLER, D., THRUN, S., AND DAVIS, J. 2005. The correlated correspondence algorithm for unsupervised registration of nonrigid surfaces. In *Proc. Neural Information Processing Systems (NIPS)*, The MIT Press.

BOYD, S., AND VANDENBERGHE, L. 2004. *Convex Optimization*. Cambridge University Press.

CHANG, W., AND ZWICKER, M. 2008. Automatic registration for articulated shapes. In *Proc. Symposium on Geometry Processing*, Eurographics Association.

CHAUDHURI, S., KALOGERAKIS, E., GUIBAS, L., AND KOLTUN, V. 2011. Probabilistic reasoning for assembly-based 3d modeling. In *Proc. SIGGRAPH*, ACM.

CHEN, X., GOLOVINSKIY, A., AND FUNKHOUSER, T. 2009. A benchmark for 3d mesh segmentation. In *Proc. SIGGRAPH*, ACM.

GOLOVINSKIY, A., AND FUNKHOUSER, T. 2008. Randomized cuts for 3d mesh analysis. In *Proc. SIGGRAPH Asia*, ACM.

GOLOVINSKIY, A., AND FUNKHOUSER, T. 2009. Consistent segmentation of 3d models. *Computers & Graphics 33*, 3, 262–269.

GRANT, M., AND BOYD, S. 2011. *CVX: Matlab Software for Disciplined Convex Programming*. `http://www.stanford.edu/~boyd/cvx/`.

KALOGERAKIS, E., HERTZMANN, A., AND SINGH, K. 2010. Learning 3d mesh segmentation and labeling. In *Proc. SIGGRAPH*, ACM.

KATZ, S., AND TAL, A. 2003. Hierarchical mesh decomposition using fuzzy clustering and cuts. In *Proc. SIGGRAPH*, ACM.

KAZHDAN, M., FUNKHOUSER, T., AND RUSINKIEWICZ, S. 2004. Shape matching and anisotropy. In *Proc. SIGGRAPH*, ACM.

KRAEVOY, V., JULIUS, D., AND SHEFFER, A. 2007. Model composition from interchangeable components. In *Proc. Pacific Graphics*, IEEE Computer Society, 129–138.

KUMAR, M. P., KOLMOGOROV, V., AND TORR, P. H. S. 2009. An analysis of convex relaxations for MAP estimation of discrete MRFs. *Journal of Machine Learning Research 10*, 71–106.

OSADA, R., FUNKHOUSER, T., CHAZELLE, B., AND DOBKIN, D. 2002. Shape distributions. *ACM Transactions on Graphics 21*, 4, 807–832.

RAND, W. M. 1971. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association 66*, 846–850.

REN, X., AND MALIK, J. 2003. Learning a classification model for segmentation. In *Proc. IEEE International Conference on Computer Vision*.

SHAMIR, A. 2008. A survey on mesh segmentation techniques. *Computer Graphics Forum 27*, 1539–1556.

SHAPIRA, L., SHAMIR, A., AND COHEN-OR, D. 2008. Consistent mesh partitioning and skeletonisation using the shape diameter function. *The Visual Computer 24*, 4, 249–259.

SHAPIRA, L., SHALOM, S., SHAMIR, A., COHEN-OR, D., AND ZHANG, H. 2010. Contextual part analogies in 3d objects. *International Journal of Computer Vision 89*, 309–326.

SHI, J., AND MALIK, J. 2000. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence 22*, 8, 888–905.

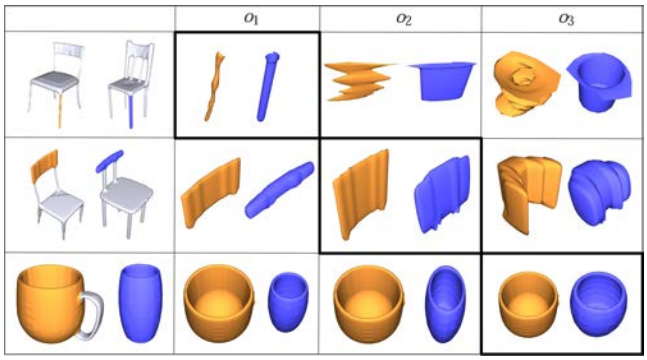


Figure 11: *Normalization operators for comparing segment shapes. The compared segments are shown in the leftmost column in the context of the source database shapes. The effects of the three normalization operators are illustrated in the three columns to the right. The most effective operator for each pair of segments is indicated by a bold outline.*

SHOTTON, J., WINN, J., ROTHER, C., AND CRIMINISI, A. 2009. Textonboost for image understanding: multi-class object recognition and segmentation by jointly modeling texture, layout, and context. *International Journal of Computer Vision 81*, 2–23.

SIDI, O., VAN KAICK, O., KLEIMAN, Y., ZHANG, H., AND COHEN-OR, D. 2011. Unsupervised co-segmentation of a set of shapes via descriptor-space spectral clustering. *ACM Trans. Graph. 30* (December), 126:1–126:9.

SIMARI, P., NOWROUZEZAHRAI, D., KALOGERAKIS, E., AND SINGH, K. 2009. Multi-objective shape segmentation and labeling. *Computer Graphics Forum 28*, 5, 1415–1425.

SONTAG, D., AND JAAKKOLA, T. 2009. Tree block coordinate descent for MAP in graphical models. *Journal of Machine Learning Research - Proceedings Track 5*, 544–551.

WAINWRIGHT, M. J., JAAKKOLA, T., AND WILLSKY, A. S. 2005. MAP estimation via agreement on trees: message-passing and linear programming. *IEEE Transactions on Information Theory 51*, 11, 3697–3717.

XU, K., LI, H., ZHANG, H., COHEN-OR, D., XIONG, Y., AND CHENG, Z.-Q. 2010. Style-content separation by anisotropic part scales. *ACM Trans. Graph. 29* (December), 184:1–184:10.

A Shape Distance Measure

In this appendix we define the shape distance measure $d(\cdot,\cdot)$ that is used to evaluate the geometric similarity of individual segments. We follow the common idea of computing descriptors for each segment and comparing segments through their geometric descriptors. As we have to compute descriptors for segments from all randomized segmentations and the geometric details could change drastically across shapes, we use the D2 descriptor [Osada et al. 2002] due to its simplicity and robustness. It is possible that a more sophisticated similarity metric could further improve the results.

The main challenge we have encountered for estimating the geometric similarity of individual segments is anisotropic scale variation, which is common in large shape databases. To factor out shape anisotropy, we build on the approach of Kazhdan et al. [2004]. Specifically, we define three normalization operators that factor out scale variation along one, two, and three principal directions respectively. These normalization operators are illustrated in Figure 11. The first operator, $o_1(s)$, scales the shape s isotropically such that the first principal eigenvalue equals 1: this operator is particularly suitable for comparing highly elongated segments. The second operator, $o_2(s)$, scales the shape anisotropically such that the first two principal eigenvalues equal 1: this operator is effective for comparing largely flat segments. Finally, the third operator, $o_3(s)$, scales the shape such that all three principal eigenvalues equal to 1: this is the operator used by Kazhdan et al. [2004].

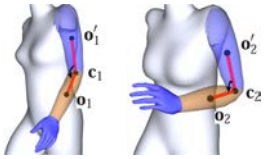
As pointed out by Kazhdan et al., o_3 is less effective when the underlying shape is highly anisotropic, as in the case of very elongated or flat segments. For this reason, we also employ operators o_1 and o_2, and define the distance function d in terms of the operator that minimizes the difference between the descriptors of normalized segments. The distance function is defined as follows:

$$d(s,s') = \left(\min_{1\leq i\leq 3} \|\mathbf{d}_i(s) - \mathbf{d}_i(s')\|^2 + \alpha \|\Lambda(s) - \Lambda(s')\|^2 \right)^{\frac{1}{2}},$$

where $\mathbf{d}_i(s)$ denotes the D2 descriptor of $o_i(s)$ [Osada et al. 2002], $\Lambda(s)$ denotes the vector of principal eigenvalues of segment s, and α controls the importance of anisotropic segment scales. In this paper, we set $\alpha = 0.1$.

B Pose Term

In this appendix we define the pose term $w_{\text{pose}}\big((s_i,s_i'),(s_j,s_j')\big)$ that partly evaluates the quality of the corresponding cuts between adjacent pairs of segments s_i and s_i' on W_i and s_j and s_j' on W_j. The pose term is defined to be higher for cuts that separate corresponding segments whose relative pose is different in the two shapes. Let $\mathbf{o}_i$ and $\mathbf{o}_i'$ denote the barycenters of s_i and s_i', respectively. Let $\mathbf{c}_i$ denote the barycenter of the cut between s_i and s_i'. Let $\alpha_i = \angle \mathbf{o}_i\mathbf{c}_i\mathbf{o}_i'$ denote the angle between vectors $\mathbf{o}_i\mathbf{c}_i$ and $\mathbf{c}_i\mathbf{o}_i'$. The pose term is defined as follows:



$$w_{\text{pose}}\big((s_i,s_i'),(s_j,s_j')\big) = 1 + 2|\alpha_i - \alpha_j|.$$

C Variable Linearization

In this appendix we prove that the linear constraints defined in (15) are equivalent to the quadratic constraints $z_{(c,c')} = y_c y_{c'}$ over all adjacent pairs (c,c'). First we show that the linear constraints are implied by the quadratic constraint:

$$\sum_{s_i'\in \text{cover}(p_i)} \sum_{s_j'} z_{(c,(s_i',s_j'))} = \sum_{s_i'\in \text{cover}(p_i)} \sum_{(c,(s_i',s_j'))\in\mathcal{A}_{ij}} y_c y_{(s_i',s_j')}$$
$$\leq y_c \sum_{s_i'\in \text{cover}(p_i)} \sum_{s_j'} y_{(s_i',s_j')} \leq y_c \sum_{s_i'\in \text{cover}(p_i)} x_{s_i'} = y_c.$$

Next we prove that the linear constraints imply the quadratic constraint. If either $y_c = 0$ or $y_{c'} = 0$ then (15) directly implies $z_{c,c'} = 0$. If $y_c = y_{c'} = 1$ then $y_{c''} = 0$ for all $c'' = (s_i'', s_j'') \neq c'$, $s_i'' \in \text{cover}(p_i)$. Replacing c in (15) by each c'' in turn, we see that adjacent pair indicators on the left side of (15) are zero except $z_{(c,c')}$, which is either 0 or 1. Similarly, $z_{(c,c')}$ on the left side of (15) can be either 0 or 1 when c is swapped with c'. Since $z_{(c,c')}$ only appears on the left side of (15) twice, it follows that $z_{(c,c')} = 1$ because the objective function is maximized and the coefficient of $z_{(c,c')}$ is positive.