Graph Cut with Ordering Constraints on Labels and its Applications

Xiaoqing Liu
xliu65@uwo.ca

Olga Veksler
olga@csd.uwo.ca

Jagath Samarabandu
jagath@uwo.ca

University of Western Ontario
London, Canada

Abstract

In the last decade, graph-cut optimization has been popular for a variety of pixel labeling problems. Typically graph-cut methods are used to incorporate a smoothness prior on a labeling. Recently several methods incorporated ordering constraints on labels for the application of object segmentation. An example of an ordering constraint is prohibiting a pixel with a "car wheel" label to be above a pixel with a "car roof" label. We observe that the commonly used graph-cut based α-expansion is more likely to get stuck in a local minimum when ordering constraints are used. For certain models with ordering constraints, we develop new graph-cut moves which we call order-preserving *moves. Order-preserving moves act on all labels, unlike α-expansion. Although the global minimum is still not guaranteed, optimization with order-preserving moves performs significantly better than α-expansion. We evaluate order-preserving moves for the geometric class scene labeling (introduced by Hoiem* et al.*) where the goal is to assign each pixel a label such as "sky", "ground", etc., so ordering constraints arise naturally. In addition, we use order-preserving moves for certain simple shape priors in graph-cut segmentation, which is a novel contribution in itself.*

1. Introduction

Pixel labeling problems involve assigning a label from a finite set of possibilities to each image pixel. Pixel labeling is often solved in a global optimization framework. An energy function on the labeling is formulated and minimized. An energy function often incorporates coherence assumption, that is most nearby pixels should have similar labels. A frequently used special case is Potts model [2], which corresponds to assuming that the majority of nearby pixels have exactly the same label. For Potts model, the graph-cut based α-expansion [2] performs best in terms of speed and accuracy [30] when compared to other popular minimization methods such as TRW [20] and BP [33]. In this paper, we restrict our attention to graph-cut optimization.

Recently several authors [32, 16] used a new interesting constraint in graph-cut object segmentation. The object is divided into several parts, for example, roof, wheels, etc. Each part corresponds to a label. In addition to smoothness, there are ordering constraints on labels. For example, the "car wheel" label cannot be above the "car roof" label, etc. Ordering constraints rule out improbable segmentations and therefore improve results. Optimization with ordering constraints, however, is harder, and the commonly used α-expansion is more likely to get stuck in a local minimum.

We propose new *order-preserving* moves for graph-cut optimization with certain ordering constraints. These moves are developed for a specific model suitable for our applications. However, the construction behind the order-preserving moves can be reused for other models, with different number of parts. The advantage of order-preserving moves over α-expansion is that they act on all labels simultaneously, giving each pixel a larger choice of labels.

We assume that an image is to be segmented into five parts, namely "center", "left", "right", "top", "bottom". The ordering constraints between the labels are easy to read from their names: a pixel labeled as "left" cannot be to the right of any pixel labeled as "center", etc. In addition, we can enforce a more stringent set of constraints: if a pixel p labeled as "center" has a neighbor q with a different label, then q must have label "left", "right", "top", "bottom" if it is to the left, right, above, or below p, respectively. These additional constraints imply that the region labeled as "center" is a rectangle, see Figs. 3(a-d).

We first evaluate order-preserving moves on the application of geometric class scene labeling, inspired by Hoiem *et al*. [13, 12]. In [12], a rough 3D reconstruction of a scene is constructed. They train a classifier, which assigns a scene region its most likely geometric label, such as "vertical", "sky", etc. A rough 3D scene description (unlike traditional 3D reconstruction [9]), is useful for scene visualization [12], object recognition [14], etc. Like [12], we train a classifier to find out individual label preferences for each pixel. Unlike [12], we formulate the problem in a global optimization framework, using our five part model. We demonstrate the usefulness of the extracted 3D structure for

978-1-4244-2243-2/08/$25.00 ©2008 IEEE

virtual scene walk-through. See also [26] for a related work on 3D reconstruction from a single image.

Our second application is incorporating certain simple shape priors in graph-cut segmentation of an object from its background. Shape priors for segmentation in general [22, 25, 4] and segmentation with a graph-cut [11, 21] is an area much interest recently. General segmentation with a shape prior is usually based on local optimization, and therefore the solution is prone to getting stuck in a local minimum. The graph-cut methods in [11, 21] have to register the shape model with the image during the segmentation process, which is a difficult task in itself.

Instead of shape priors specific to some object, like in [11, 21], we investigate simple generic shapes such as "rectangle", "trapezoid", etc. We observe that by splitting an image into parts with ordering constraints between them, we can enforce the "center" region to be of a certain shape, for example, a rectangle, as in Fig. 3. Usually the object/background segmentation is formulated as a binary labeling: the labels are the object and the background. We use more than two labels to incorporate a shape prior: the object corresponds to the "center" label and the other labels correspond to the background. This is a new approach for incorporating shape priors. It is the relative order of the parts that enforce a certain shape for the object. We use a rectangular and a trapezoidal shape, although other simple shapes can be implemented too. In [19], they use a similar idea but only for rectangular shapes.

The paper is organized as follows. Sec. 2 reviews graph cut optimization. Sec. 3 explains order-preserving moves. Sec. 4 and 5 present order-preserving moves for the geometric scene labeling and for simple shape priors.

2. Graph-Cut optimization

In this section we describe the graph-cut optimization framework of [2]. Suppose we have a pixel labeling problem where the task is to assign to each image pixel p some label from a finite label set $\mathcal{L}$. Let $\mathcal{P}$ be the set of all pixels in an image, and f_p be a label assigned to a pixel p (i.e. $p \in \mathcal{P}$, $f_p \in \mathcal{L}$). Let $f = \{f_p | p \in \mathcal{P}\}$ be the collection of all pixel/label assignments. The energy function is:

$$E(f) = \lambda \sum_{p \in P} D_p(f_p) + \sum_{(p,q) \in \mathcal{N}} V_{pq}(f_p, f_q) \qquad (1)$$

In Eq. (1), $D_p(f_p)$ and $V_{pq}(f_p, f_q)$ are called the data and the smoothness terms, respectively, and $\mathcal{N}$ is a neighborhood system on $\mathcal{P}$. Our $\mathcal{N}$ is the 4-connected grid, which consists of ordered pixel pairs (p, q) s.t. $p < q$. $D_p(f_p)$ is the penalty for p to have label f_p, and thus it encourages each pixel to be assigned the label of smallest penalty.

The smoothness term $V_{pq}(f_p, f_q)$ encourages spatial consistency by penalizing neighboring pixels p and q that are not assigned the same label. For example for Potts model, $V_{pq}(f_p, f_q) = 0$ if $f_p = f_q$ and $V_{pq}(f_p, f_q) = w_{pq}$ if $f_p \neq f_q$, where w_{pq} is a positive coefficient that can depend on the particular pixel pair (p, q). To encourage discontinuities to align with the image edges, typically w_{pq} is small if there is an intensity edge between pixels p and q.

f_p \ f_q	L	R	C	T	B
L	0	∞	w_{pq}	w_{pq}	w_{pq}
R	∞	0	∞	∞	∞
C	∞	w_{pq}	0	∞	∞
T	∞	w_{pq}	∞	0	∞
B	∞	w_{pq}	∞	∞	0

(a) p is the left neighbor of q

f_p \ f_q	L	R	C	T	B
L	0	∞	∞	∞	w_{pq}
R	∞	0	∞	∞	w_{pq}
C	∞	∞	0	∞	w_{pq}
T	w_{pq}	w_{pq}	w_{pq}	0	∞
B	∞	∞	∞	∞	0

(b) p is the top neighbor of q

Figure 1. Smoothness terms $V_{pq}(f_p, f_q)$

For Potts model, in case of two labels, the energy in Eq. (1) can be minimized exactly, and in the multi-label case a solution that is optimal within a factor of two can be found with the α-expansion [2]. The α-expansion finds a local minimum with respect to expansion moves. Given a labeling f and a label α, a move from f to f^α is called an α-expansion if $f_p \neq f_p^\alpha \Rightarrow f_p^\alpha = \alpha$, i.e the set of pixels labeled as α "expands" in f^α. The optimal α-expansion can be found efficiently using a min-cut/max-flow algorithm. The α-expansion algorithm iterates over all labels α, finding the best α-expansion, until convergence.

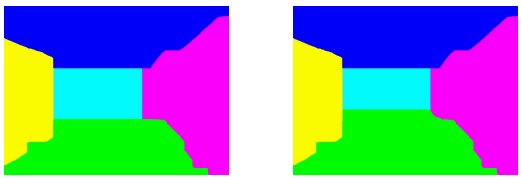
In addition to spatial consistency, V_{pq} can be used to incorporate ordering constraints on labels. For example, if p is immediately to the left of q, to prohibit $f(p) = "center"$ and $f(q) = "left"$, we set $V_{pq}("center","left") = \infty$. After adding ordering constraints to Potts model, the factor of 2 approximation no longer holds.

3. Order-Preserving Moves

In this section we explain order-preserving moves. For compactness, we abbreviate label names with their first letter, i.e. L, R, T, B, C, correspond, respectively, to "left", "right", "top","bottom","center". The smoothness terms V_{pq} for horizontal neighbors are in Fig. 1(a), and for vertical neighbors are in Fig. 1(b). Positive coefficient w_{pq} is chosen as discussed in Sec. 2. The model in Fig. 1 is Potts plus the ordering constraints. Under this model, a labeling has a finite energy only if the "center" part is a rectangle, and the "left", "right", "top", "bottom" parts are to the left, right, above, below the "center" part, respectively. For example, all labelings in Figs. 2 and 3 have finite energy.

To motivate order-preserving moves, we first illustrate that with ordering constraints, it is easier for α-expansion to get stuck in a local minimum. This problem is also reflected by the fact that the factor of 2 bound does not hold if ordering constraints are added to the Potts model. Authors in [32, 16] who used ordering constraints cannot achieve good results with α-expansion alone.

Consider Fig. 2, which shows the results of α-expansion



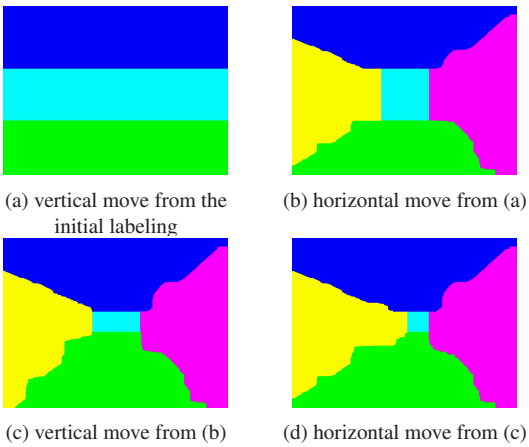
(a) after one iteration (b) after two iterations

Figure 2. Results with α-expansion. Initial labeling, not shown, is all pixels labeled as "center". Color scheme: green = "bottom", yellow = "left", cyan = "center", magenta = "right", blue = "top". This color scheme is consistent throughout the paper.

for an instance of a geometric class labeling problem. Fig. 2(a) shows the labeling after one iteration, where one iteration means one α-expansion for each label $\alpha \in \{L, R, T, B, C\}$. Fig. 2(b) shows the labeling after two iterations. Only the B region expands from (a) to (b), and the algorithm, in fact, converges after 2 iterations. However, the labeling in Fig. 2(b), which has energy of $1,590,159$, is far from the optimum. Fig. 3(d) shows the labeling (found by our algorithm) that has a much better energy of $1,443,150$. The problem with the local minimum in Fig. 2(b) is as follows. To get to a better labeling, a smaller C region is needed. Labels B, T, L, and R need to expand, each one separately, to obtain a smaller C region. However, each individual expansion on the B, T, L, R does not result in a lower energy, and so the expansion algorithm gets stuck in a local minimum. We also show experimentally in section 4.2 that the energies obtained by the order-preserving moves are significantly better than those of α-expansion.

In order to improve on α-expansion moves in presence of ordering constraints, we should allow a pixel to have a choice of labels to switch to as opposed to just a single label α. Let L_p be a subset of labels that pixel p is allowed to switch to in one move. Typically, graph-cut algorithms use the same rule for choosing L_p for every pixel. For α-expansion, L_p consists of α and the old label of pixel p. For α-β swap [2], $L_p = \{\alpha, \beta\}$. For global optimization methods in [18, 27], $L_p = \mathcal{L}$, but they can handle only a restricted type of energies, and ours is not of that type.

Our insight is that by using different rules when selecting L_p for different pixels, we can have a larger L_p for each pixel, as compared to α-expansion, that is there is more labels to choose from for each pixel in a single move. Notice that the choice of L_p precisely defines the allowed moves. That is a move from f to f' is allowed if $f'_p \in L_p$, $\forall p \in \mathcal{P}$. We must, therefore, select L_p's in such a way that the allowed move of minimum energy can be computed efficiently. In addition, L_p must have the old label of pixel p, so that the set of allowed moves contains the old labeling and therefore the best allowed move is not worse than the old labeling. We found two such moves, we call them *horizontal* order-preserving and *vertical* order-preserving.



(a) vertical move from the initial labeling (b) horizontal move from (a)

(c) vertical move from (b) (d) horizontal move from (c)

Figure 3. Results with order-preserving moves on the same problem as in Fig. 2. Initial labeling (not shown) was all "center".

Fig. 3 shows labelings for order-preserving moves on the same example of geometric labeling as in Fig. 2. A horizontal move (from Fig. 3 (a) to Fig. 3(b)) allows any change in labels except the region labeled as C cannot change its height. Either increase or decrease in width of the C region is allowed. The name "horizontal" reflects the fact that the C region can change in the horizontal, but not in the vertical direction. Similarly, a vertical move (from Fig. 3(b) to Fig. 3(c)) allows any change in labels except the region labeled as C cannot change its width.

Let f be a labeling, and x_p the horizontal coordinate of pixel p. Let $\underline{x}$ be the smallest x coordinate of any pixel that has label C in f, that is $\underline{x} = \min\{x_p | f_p = C\}$. Similarly, let $\overline{x}$ be the largest x coordinate of any pixel that has label C in f, that is $\overline{x} = \max\{x_p | f_p = C\}$. Recall that L_p is the set of allowed labels that p can switch to in a move. It is easy to see that for a vertical move, the following rules apply. If $p_x < \underline{x}$, then $L_p = \{T, L, B\}$. If $\underline{x} \leq p_x \leq \overline{x}$, then $L_p = \{T, C, B\}$. Finally, if $p_x > \overline{x}$, then $L_p = \{T, R, B\}$. In words, divide f into three rectangles with two vertical lines, one passing through the border of the L and C regions, and the other passing through the border of C and R regions. Then pixels in the left rectangle can switch their labels to T, or L, or B; pixels in the middle rectangle can switch their labels to T, or C, or B, and, finally pixels in the right rectangle can switch their labels to T, or R, or B.

To find an optimal vertical move, we use a very important result from Schlesinger et.al. [27]. In [27], they define a submodular energy in the case of multiple *ordered* labels, and give a graph construction that can be used to optimize a submodular energy globally with the minimum cut. An energy is submodular [27] if every V_{pq} term is submodular. In turn, V_{pq} is submodular, if for any $\alpha < \beta$, and $\alpha' < \beta'$, $V_{pq}(\alpha, \alpha') + V_{pq}(\beta, \beta') \leq V_{pq}(\alpha, \beta') + V_{pq}(\beta, \alpha')$. See also [6] for an equivalent result.

It is easy (but tedious) to check that the vertical move

energy with V_{pq}'s in Fig. 1 and label order $T < L < B$, $T < C < B$, and $T < R < B$ is submodular. Notice that we do not have to order labels L, C, R with respect to each other because a single pixel under vertical move never has to choose between L, C, and R. There is no way to order all labels L, C, R, B, T so that our energy is submodular. Thus the main idea behind our moves is choosing L_p's for each p in such a way that the energy function restricted to the corresponding move is submodular.

Due to symmetry, horizontal moves are handled similarly to vertical. In practice, we compute the optimal horizontal move by transposing the image, swapping labels L and T, R and F, and computing the optimal vertical move.

Thus an order-preserving move gives every pixel a choice of 3 labels to switch to, while α-expansion gives a choice of only 2 labels. In addition, α-expansion effectively acts on only one label, since only α label is allowed to increase its territory during the move. Our moves act on all labels simultaneously, since any label has a chance to increase (as well as shrink) its territory during a single move.

We compute a local minimum with respect to the order-preserving moves. We start with an initial labeling and alternate between the best vertical and horizontal order-preserving moves until no move that would result in a lower energy can be found. The initial labeling has to be order-preserving. In practice, we start with all pixels labeled as C. Fig. 3 shows the sequence of labelings that we obtain under the order-preserving moves from the first one in (a) to the one at the convergence after 4 steps in (d). For the first move (in (a)), no L and R labels are allowed, since the initial labeling has all pixels labeled as C. That is why the labeling in (a) looks odd.

4. Geometric Class Scene Labeling

In this section, we apply the order-preserving moves to the geometric class scene labeling, inspired by Hoiem *et al*. [12]. In [12], the goal is to automatically extract a coarse 3D scene structure from a single 2D image by assigning each image pixel its rough geometric label, such as "sky", "ground", etc. Unlike traditional 3D reconstruction, [12] extracts only an approximate 3D structure. Traditional 3D reconstruction [9] requires special equipment, such as multiple cameras, or range scanners, etc. Furthermore, the 3D reconstruction methods that are based on pixel correspondences between several images are often unreliable, especially for indoor scenes which tend to be low-textured. Even though 3D description from geometric scene labeling is coarse, it is useful for many applications. We use it for virtual scene walk-through.

In addition to [13, 12], there are other single view approximate reconstruction methods. Most require user interaction, see [17, 7, 28, 5]. Some are automatic [8, 3], but make relatively restrictive assumptions about the scene.

Unlike [12], we formulate the problem in a global optimization framework, using the five part model discussed in Sec. 3, and optimizing the energy in Eq. (1) with order-preserving moves. Our five-part model is a less general model, compared to [12]. Nevertheless, it is still appropriate for many indoor/outdoor environments.

Our label set is $\mathcal{L} = \{bottom, left, center, right, top\}$. The V_{pq} terms in Eq. (1) are as in Sec. 3. The set of ordering constraints from Sec. 3 ensures that the boundaries between the parts agree with the directions caused by the perspective effects under the standard camera orientation, that is the boundary between the "left" and "bottom" parts is a diagonal slanted down and to the left, etc. For the data terms in Eq. (1), we train a classifier in a manner similar to [12], the details are in Sec. 4.1.

In a later version, Hoiem *et al*. [15] tried global optimization for geometric labeling, without improvement. Our improvement is probably due to the following factors. In [15], optimization is performed on superpixel level, not on pixel level as we do. Therefore in case when a superpixel contains pixels with different true labels, [15] cannot assign the true labels to all the pixels in that superpixel. By optimizing on pixel level we are able to break apart any superpixel, as needed. In particular, we are able to better align the boundaries between the geometric labels with the intensity edges in the image, which also helps. In addition, our stringent set of ordering constraints and better optimization with order-preserving moves contributes to the improvement in results.

4.1. Data term

Ideally, we want to model the data term as $D_p(f_p) = -\log Pr(f_p|F_p)$, where F_p is an observed feature vector at pixel p. However, for the geometric labels image data at a single pixel is not enough for a useful likelihood model.

We follow Hoiem *et al*. [12] who observe that an image region frequently does contain enough data to reliably classify it with a geometric label. We first partition images into "superpixels"[1] using the algorithm by Felzenszwalb *et al*. [23]. Then for each superpixel we compute a large set of features which are the statistics on location, color, geometry, texture and edges, similar to [12]. Finally, we train the SVM classifier [31] on the extracted feature vectors. The output of SVM is an uncalibrated value, not a probability distribution. We use the method in [24] to convert to the distribution $Pr(S = l|F_S)$ where l is a label, F_S is a feature vector computed on superpixel S, and $S = l$ is the event that all pixels inside S have the same label l.

We apply the distributions learned on superpixels to the pixel based data term. That is $D_p(f_p) = -\log Pr(S^p = f_p|F_{S^p})$, where S^p is the superpixel that contains p. This makes sense since the energy in Eq. (1) does not require the true log probabilities. It is sufficient to use a reasonable penalty scheme for the $D_p(f_p)$ term, namely a scheme that

[1] A superpixel is a region returned by a segmentation algorithm.

imposes higher penalties for the less likely labels. It is reasonable to assume that if $Pr(S = l_1) < Pr(S = l_2)$, then for most pixels $p \in S$, $Pr(p = l_1) < Pr(p = l_2)$.

4.2. Results

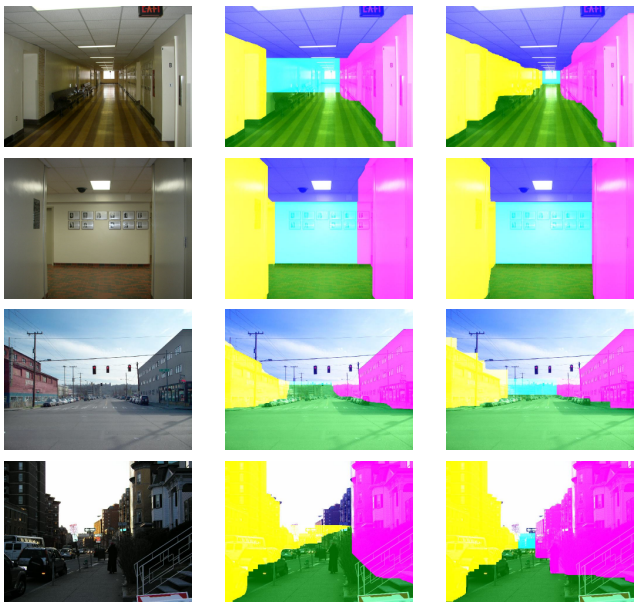
We have collected 600 images from different indoor environments, and downloaded 84 outdoor street images from the Web and PASCAL database. All images were manually labeled. We used half of the images for training and half for testing (separately for indoor/outdoor).

Fig. 4 shows some results of SVM classification in (b), α-expansion without ordering constraints in (c), and the order-preserving moves in (d). SVM labelings are not nearly as spatially consistent as those obtained with graph-cut optimization. In the bottom row of Fig. 4 (b) SVM fails to label most of the floor correctly. The spatial smoothness constraints help to correct this, see Fig. 4 (c,d). Comparing graph-cut without and with ordering constraints, in columns (c) and (d), respectively, implausible regions are frequent in Fig. 4 (c): center patches appear in the middle of left patches, etc. In the bottom row of Fig. 4(c), the center region is significantly distorted, compared to (d). Ordering constraints clearly help to rule out implausible solutions.

(a) (b) (c) (d)

Figure 4. Results: (a) original images (b) SVM labeling, (c) α-expansion, no ordering constraints (d) order-preserving moves.

In Fig. 5 we show some results of α-expansion (in (b)) and order-preserving moves (in (c)) when ordering constraints are used in the energy function. As expected, α-expansion gets stuck in a local minimum easier. We also directly compare the energy values produced by the two algorithms. The order-preserving moves always give a smaller energy compared to α-expansion. On the 300 indoor images, on average, the energy is 27.3% smaller (σ = 9.8%). On the 42 outdoor images, on average, the energy is 29.2 % smaller (σ = 18.5%).

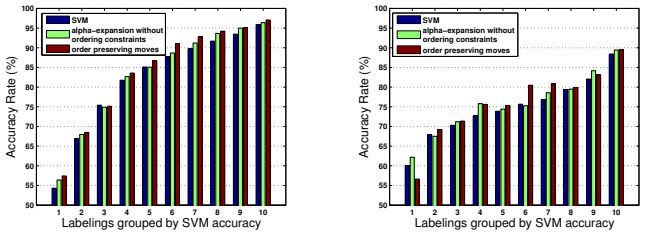


(a) (b) (c)

Figure 5. Results: (a) original images, (b) α-expansion with ordering constraints, (c) order-preserving moves.

When SVM gives reasonable label probabilities, our algorithm can significantly improve SVM results. When SVM results are far from reasonable, order-preserving moves can worsen SVM results, trying to satisfy the ordering constraints that cannot be reasonably satisfied, see Fig. 8. Therefore, the overall accuracy improvement over SVM computed for all the images is not large. However, when SVM results are not reasonable, they are hardly useful for applications anyway, see section 4.3.

We put SVM results in 10 equal bins, ordered from least accurate to most accurate. The higher the bin number, the more accurate are the SVM labelings in that bin. Fig. 6



(a) indoor images (b) outdoor images

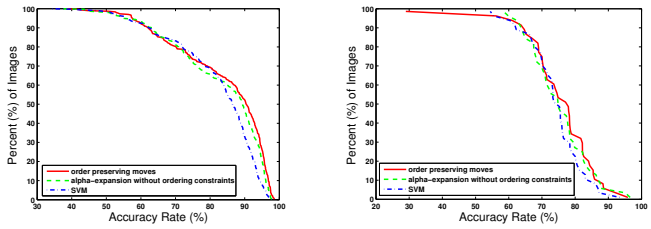
Figure 6. SVM labelings, in bins by quality vs. accuracy rate

Table 1. Performance Summary

	Percentage of Successful Labelings (%)			
	SVM	α-exp. no OC	α-exp. with OC	order-pres.
Indoor	29.3	61.0	72.3	74.3
Outdoor	16.7	40.5	38.1	61.9
	Overall Accuracy Rate (%)			
	SVM	α-exp. no OC	α-exp. with OC	order-pres.
Indoor	83.0	84.1	84.7	85.0
Outdoor	74.0	75.2	71.0	75.3

shows, for each group, the accuracy of the algorithms. For the worst bin (unreliable SVM results), order-preserving moves actually decrease SVM accuracy for outdoor images. For the best bins (very accurate SVM results), order-preserving moves do not improve SVM results significantly, because there is not much improvement do be done anyway. However, in the middle range, from about 4th bin to the 8th bin, there is significant improvement over SVM and α-expansion, especially for the outdoor images. For example, in the 6th bin, order-preserving moves have about 80% accuracy, followed by approximately 75% accuracy for α-expansion and SVM.

Fig. 7 shows the percentage of labelings that have the at least the accuracy rate specified on the horizontal axis. For example, for indoor images, 52% of order-preserving labelings have the accuracy rate of at least 90%, whereas only 33% and 46% of SVM and α-expansion labelings, respectively, have this accuracy rate. Order-preserving moves always have a higher percentage of images at any given accuracy rate in the range between 75% and 100%.

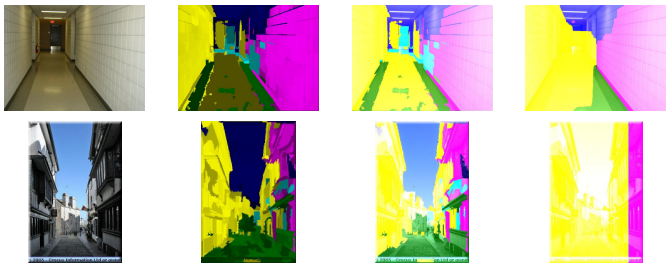


(a) indoor images (b) outdoor images

Figure 7. Accuracy rate vs. % of images

We also compute percentage of "successful" labelings, where a labeling is successful if it has at least 90% overall accuracy or at least 80% overall accuracy and the "center" region is at least 60% accurate. We found experimentally that such labelings can be used successfully for virtual scene walk-through. Table 1 summarizes, in this order, the performance of SVM, α-expansion without and with ordering constraints, and the order-preserving moves. Order-preserving algorithm is a clear winner when it comes to the percentage of successful labelings, and also shows a modest improvement for the overall accuracy rate.

Some failures are in Fig. 8. The ordering constraints are not violated in Fig. 8(d), but the "center" region between the "left" and the "right" regions it is too thin to see at this resolution. Most failures occur when the "center" data terms are far from reasonable, as in Fig. 8(b).



(a) (b) (c) (d)

Figure 8. Failure cases (a) original images (b) SVM generated label probabilities (c) SVM labeling, (d) Order-preserving moves.

Figs. 9, 10 show more results, illustrating the accuracy the proposed method can achieve with no user interaction.

The average processing time for the order-preserving moves is 62.3 (σ = 25.6) seconds, calculated on a personal computer with 2.4GHz CPU and 2048MB memory. This time includes segmentation, feature extraction, data terms calculation, and energy minimization. We use the efficient max-flow algorithm of [1] for min-cut computation.

4.3. Application: Virtual Scene Walk-Through

We now illustrate the use of the obtained 3D structure for automatic virtual scene walk-through. We use the spidery mesh [17] to fit perspective projection and mimic 3D camera transformations to navigate through the scene. Spidery mesh is composed of four parts (vanishing point, radial lines, inner and outer rectangles), which partition the 2D image into five regions (left wall, right wall, rear wall, floor, and ceiling). Since we have already labeled the indoor image into exactly these five regions, generating the spidery mesh is trivial. We fit the radial lines with the RANSAC [10] based on the boundary between differently labeled regions. Vanishing point is calculated as the weighted average of the intersection of the radial lines, the inner rectangle is the "center" region, and the rest are outer rectangles. Parts of the virtual scene walk-through are in Fig. 11 and the video results are in the supplemental material. Fig. 12 shows that using SVM results directly fails to produce satisfactory results. The room appears to have crooked walls and floor. We applied the same algorithm as above to a reasonable (93.4 % accuracy) SVM labeling.

5. Shape Prior for Segmentation

We now explain how to incorporate simple geometric shape priors in graph-cut segmentation of an object from its background. A recent related work is [29], who segment rectangles using generalized eigenvectors.

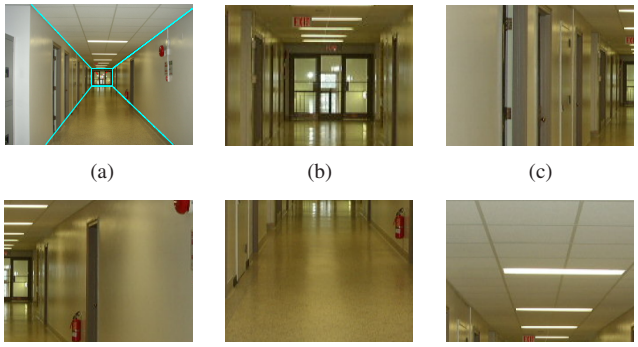
By splitting an image into several parts with ordering constraints between them, we can enforce the "center" region to be of a certain shape, for example, a rectangle, as

(a) (b) (a) (b) (a) (b)

Figure 9. Some results on indoor images (a) original images, (b) order-preserving moves.

(a) (b) (a) (b) (a) (b)

Figure 10. Some results on outdoor images (a) original images, (b) order-preserving moves.



(a) (b) (c) (d) (e) (f)

Figure 11. Virtual scene walk-through using order-preserving labeling (a) spidery mesh overlayed on image (b) walk forward (c) look left (d) look right (e) look down (f) look up

(a) (b) (c)

Figure 12. Virtual scene walk-through using SVM labeling (a) spidery mesh overlayed on image (b) walk forward (c) look left

in Fig. 3. This is a new approach for shape priors in segmentation. It is the relative order of the parts that enforces a certain shape for the object. Instead of being a binary (object/background) labeling, we have a multi-label problem now. The "center" region is the object, and the rest are the background. We evaluate a rectangular and trapezoid shape, although other simple shapes can be implemented too.

For a rectangle, we use the same V_{pq} as in Fig. 1, except now any V_{pq} not involving label C is set to 0, since a discontinuity between, say L and B labels *does not* correspond to the border between the object and the background.

We consider a trapezoid with parallel sides in horizontal orientation, and the shorter side on top (for other trapezoids, an image just needs to be rotated). To get a trapezoid, we relax the following constraints in Fig. 1: for vertical neighbors, we set $V_{pq}(L, C) = V_{pq}(R, C) = w_{pq}$, instead of ∞. This change allows the borders between the L and C regions and C and R regions to be diagonals, slanted to the left and to the right, respectively. This shape prior is not, strictly speaking, a true trapezoid, since we cannot enforce the borders between the L and C regions and C and R regions to be straight lines. We still use "trapezoid" for the lack of a better name. We have to slightly change the horizontal order-preserving move, the details are straightforward, we omit them for the lack of space.

We can use object-specific data terms based on brightness, user interaction, etc. However here, to study the effect of the shape prior in isolation from regional influences, we opted to find regions with strong intensity edges on the boundary and agreeing with the shape prior. An object-

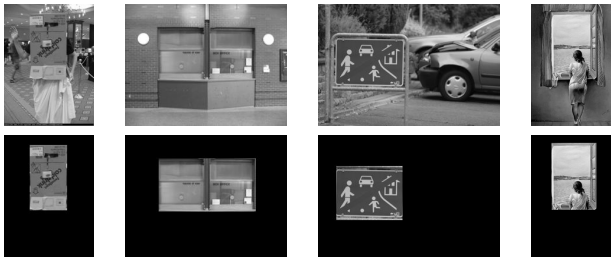


Figure 13. Rectangle shape prior.

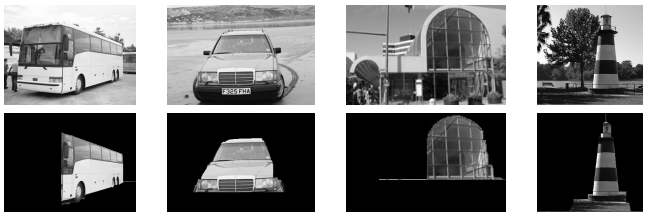


Figure 14. Trapezoid shape prior.

specific D_p can always be added. We do have to set D_p for any p on the image border. We set each border p to strongly prefer its own border, i.e. for p on the left border, $D_p(L) = 0$ and $D_p(C) = D_p(R) = D_p(T) = D_p(B) = \infty$, etc.

Thus our cost function (ignoring the border data terms, which are constant for finite energy labelings) is the sum of the w_{pq} on the boundary between the object and the other regions. To avoid a trivial solution (the object of size 1 pixel), we make w_{pq}'s negative whenever there is a strong intensity edge between pixels p and q, biasing towards a larger boundary coinciding with intensity edges. In general, making $w_{pq} < 0$ is not always possible, but it is possible for our vertical/horizontal moves, we omit the details due to the lack of space. Figs. 13 and 14 show the results with rectangular and trapezoid prior, illustrating the ability to pick out interesting regions obeying the corresponding shape priors without any knowledge of the object/background regional properties. In both figures, the original images are in the top row, and the results are in the second row. All results were obtained with the same parameter settings.

References

[1] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE TPAMI*, September 2004.

[2] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE TPAMI*, 23(11):1222–1239, November 2001.

[3] J. Coughlan and A. Yuille. Manhattan world: Compass direction from a single image by bayes. inf. In *ICCV*, 1999.

[4] D. Cremers, S. Osher, and S. Soatto. Kernel density estimation and intrinsic alignment for shape priors in level set segmentation. *IJCV*, 69(3):335–351, September 2006.

[5] A. Criminisi, I. D. Reid, and A. Zisserman. Single view metrology. *IJCV*, 40(2):123–148, 2000.

[6] J. Darbon. Global optimization for first order markov random fields with submodular priors. In *IWCIA*, 2008.

[7] P. E. Debevec, C. J. Taylor, and J. Malik. Modeling and rendering architecture from photographs: a hybrid geometry- and image-based approach. In *SIGGRAPH 96*, pages 11–20, New York, NY, USA, 1996. ACM Press.

[8] E. Delage, H. Lee, and A. Y. Ng. A dynamic bayesian network model for autonomous 3d reconstruction from a single indoor image. In *CVPR*, pages II:2418 – 2428, 2006.

[9] O. Faugeras. *Three-Dimensional Computer Vision: A Geometric Viewpoint*. MIT Press, 1993.

[10] M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Comm. of the ACM*, 24:381–395, 1981.

[11] D. Freedman and T. Zhang. Interactive graph cut based segmentation with shape priors. In *CVPR*, 2005.

[12] D. Hoiem, A. Efros, and M. Hebert. Automatic photo pop-up. In *ACM SIGGRAPH*, pages –, 2005.

[13] D. Hoiem, A. Efros, and M. Hebert. Geometric context from a single image. In *ICCV*, pages 654 – 661, 2005.

[14] D. Hoiem, A. Efros, and M. Hebert. Putting objects in perspective. In *CVPR*, pages II:2137 – 2144, June 2006.

[15] D. Hoiem, A. Efros, and M. Hebert. Recovering surface layout from an image. *IJCV*, accepted.

[16] D. Hoiem, C. Rother, and J. Winn. 3d layout crf for multi-view object class recognition and segmentation. In *CVPR*, pages 1–8, 2007.

[17] Y. Horry, K. Anjyo, and K. Arai. Tour into the picture: using a spidery mesh interface to make animation from a single image. In *SIGGRAPH'97*, pages III:225–232, 1997.

[18] H. Ishikawa. Exact optimization for markov random fields with convex priors. *IEEE TPAMI*, 25:1333–1336, 2003.

[19] J. Keuchel. Multiclass image labeling with semidefinite programming. In *ECCV*, pages II: 454–467, 2006.

[20] V. Kolmogorov. Convergent tree-reweighted message passing for energy minimization. *IEEE TPAMI*, 28(10):1568–1583, 2006.

[21] M. Kumar, P. Torr, and A. Zisserman. Obj cut. In *CVPR*, pages I: 18–25, 2005.

[22] M. Leventon, W. Grimson, and O. Faugeras. Stat. shape influence in geodesic active contours. In *CVPR*, pages 316–323, 2000.

[23] P.Felzenszwalb and D.Huttenlocher. Efficient graph-based image segmentation. *IJCV*, 59(2):–, 2004.

[24] J. C. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. 1999.

[25] M. Rousson and N. Paragios. Shape priors for level set representations. In *ECCV*, page II: 78 ff., 2002.

[26] A. Saxena, S. Chung, and A. Ng. 3-d depth reconstruction from a single still image. *IJCV*, 76(1):53–69, January 2008.

[27] D. Schlesinger and B. Flach. Transforming an arbitrary minsum problem into a binary one. Technical Report TUD-FI06-01, Dresden University of Technology, 2006.

[28] H. Shum, M. Han, and R. Szeliski. Interactive construction of 3d models from panoramic mosaics. In *CVPR*, pages 427–433, 1998.

[29] A. K. Sinop and L. Grady. Uninitialized, globally optimal, graph-based rectilinear shape segementation. In *ICCV*, 2007.

[30] R. Szeliski, R. Zabih, D. Scharstein, O. Veksler, V. Kolmogorov, A. Agarwala, M. Tappen, and C. Rother. A comparative study of energy minimization methods for markov random fields. In *ECCV*, pages II: 16–29, 2006.

[31] V. Vapnik. *The Nature of Statistical Learning Theory*. Springer Verlag, New York, 1995.

[32] J. Winn and J. Shotton. The layout consistent random field for recognizing and segmenting partially occluded objects. In *CVPR*, pages I: 37–44, 2006.

[33] J. Yedidia, W. Freeman, and Y. Weiss. Bethe free energy, kikuchi approximations, and belief propagation. pages xx–yy, 2001.