Order Preserving Moves for Graph Cut based Optimization

Xiaoqing Liu, *Member, IEEE,* Olga Veksler, *Member, IEEE,* and Jagath Samarabandu, *Member, IEEE*

***Abstract*— In the last decade, graph-cut optimization has been popular for a variety of labeling problems. Typically graph-cut methods are used to incorporate smoothness constraints on a labeling, encouraging most nearby pixels to have equal or similar labels. In addition to smoothness, ordering constraints on labels are also useful. For example, in object segmentation, a pixel with a "car wheel" label may be prohibited above a pixel with a "car roof" label. We observe that the commonly used graph-cut α-expansion move algorithm is more likely to get stuck in a local minimum when ordering constraints are used. For a certain model with ordering constraints, we develop new graph-cut moves which we call *order-preserving*. The advantage of order-preserving moves is that they act on all labels simultaneously, unlike α-expansion. More importantly, for most labels α, the set of α-expansion moves is strictly smaller than the set of order-preserving moves. This helps to explain why in practice optimization with order-preserving moves performs significantly better than α-expansion in presence of ordering constraints. 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.**

***Index Terms*— Energy minimization, graph cuts, max-flow, SVM, geometric class labeling, shape prior.**

I. Introduction

PIXEL labeling problems involve assigning a label from a finite set of possibilities to each image pixel. Many problems in computer vision can be formulated as pixel labeling problems. Some examples are image restoration, stereo correspondence, background subtraction, interactive segmentation, video editing, etc. [1]. While pixel labeling problems can be solved with local methods, global optimization framework gives better results [1]. In global optimization framework, the constraints on the solution that come from prior knowledge and data can be explicitly incorporated into an energy function, which is then optimized, either exactly or approximately.

A frequent constraint in an energy function is the smoothness of the labeling, that is most nearby pixels are expected to have similar labels. A useful 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 [1] when compared to other popular minimization methods such as TRW [3] and BP [4]. For this reason, we restrict our attention to graph-cut optimization.

X. Liu is with UtopiaCompression Corp. Los Angeles. This work was done while X. Liu was a student at the University of Western Ontario. E-mail: xliu65@alumni.uwo.ca

O. Veksler is with the University of Western Ontario. E-mail: olga@csd.uwo.ca.

J. Samarabandu is with the University of Western Ontario. E-mail: jagath@uwo.ca.

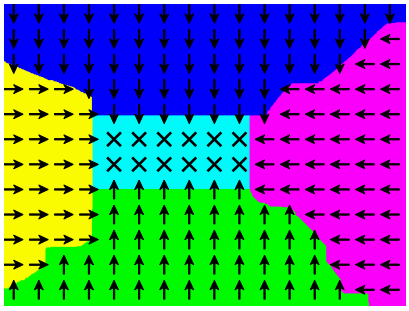


Fig. 1. An illustration of the five-part model. Color scheme for labels: "bottom" is green, "left" is yellow, "center" is cyan, "right" is magenta, and "top" is blue. This color scheme is consistent throughout the paper.

In addition to coherence constraints, ordering constraints are also useful in practice. For example, in [5] ordering constraints handle occlusions in stereo reconstruction. In [6], [7], ordering constraints are used for object segmentation. The object is divided into several parts: roof, wheels, etc. Each part corresponds to a label. Ordering constraints prohibit the "car wheel" to be above the "car roof" label, etc. This rules out improbable segmentations and therefore improves results. However α-expansion, the commonly used algorithm for optimization, is more likely to get stuck in a local minimum when ordering constraints are used.

We propose new *order-preserving* moves for graph cut optimization. These moves are developed for a specific model suitable for our applications. We assume that an image is to be segmented into five parts, namely "center", "left", "right", "top", and "bottom", see Fig. 1. The ordering constraints can be read from the names: a pixel labeled as "left" cannot be to the right of a pixel labeled as "center", a pixel labeled as "top" cannot be below a pixel labeled as "center", etc. In addition, we can enforce more stringent 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 "center" region is rectangular, see Fig. 1. Not all parts have to be present.

Order-preserving moves are strictly larger than expansion moves for all labels except the "center". However "center" expansions are hardly useful because their number is severely limited by ordering constraints. Another advantage is that unlike the expansion, order-preserving moves act on all labels simultaneously, giving each pixel a larger choice of labels.

First we evaluate order-preserving moves on the application of geometric class scene labeling, inspired by Hoiem *et al.* [8], [9]. The goal in [8] is a rough 3D reconstruction of a scene. Our five part model is applicable to a variety of (mostly indoor) scenes. Our only essential difference from [8] is a global optimization

framework with the five part model.

Our second application is incorporating certain simple shape priors, like a "rectangle" or a "trapezoid" in a segmentation of an object from its background. When 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. 1. Usually the object/background segmentation is formulated as a binary labeling. 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 to shape priors. It is the relative order of the parts that enforces a certain shape for the object. In [10], they use a similar idea but only for rectangular shapes.

Even though the order-preserving moves developed in this paper are for a specific five-part model, the construction and ideas behind them can be extended to other models that have a partial label ordering. Our principal idea is to find a large subset of labels such that exact optimization can be performed when the energy is restricted to this subset. This idea is transferable.

A preliminary version of this paper appeared in [11]. In the current version, we have expanded the properties and derivation of the order-preserving moves. In particular, we include detailed graph constructions of order-preserving moves and describe their theoretical properties, including a comparison to expansion moves. We add more experimental results and comparisons with the previous work [9]. We also add more details on shape priors.

The paper is organized as follows. Sec. II reviews graph cut optimization. Sec. III defines and explains properties of order-preserving moves. Sec. IV gives detailed construction. Sec. V and VI apply order-preserving moves for the geometric scene labeling and shape priors.

II. Optimization with Graph Cuts

This section reviews the graph-cut optimization framework of [2]. Let $G =< V, E >$ be a graph consisting of a set of vertices V and a set of edges E connecting the vertices. Each edge $(u, v) \in E$ in G is assigned a non-negative cost $w(u, v)$. There are two special vertices called *terminals* identified as the *source*, s and the *sink*, t. A cut $\mathcal{C}$ is a partition of V into two disjoint sets S and $T = V - S$ such that $s \in S$ and $t \in T$. The cost of the cut $\mathcal{C}$ is defined as: $|\mathcal{C}| = \sum_{u \in S, v \in T} w(u, v)$. The minimum cut is the cut with the smallest cost, and can be computed with a max-flow/mincut algorithm [12]. We use the algorithm of [13], which was designed specifically for computer vision applications and has the best performance in practice.

In a pixel labeling problem the task is to assign to each image pixel p a label from a finite set $\mathcal{L}$. Let $\mathcal{P}$ be the set of all pixels in an image, f_p be the label assigned to p (i.e. $p \in \mathcal{P}$, $f_p \in \mathcal{L}$), and let f denote 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 $\mathcal{P}$. We use the standard 4-connected $\mathcal{N}$, which consists of ordered pixel pairs (p, q) s.t. $p < q$. Pixels are indexed row-wise with consecutive integers. Therefore, if p and q are neighbors and $p < q$ then either p is to the left of q, or p is above q.

The data term $D_p(f_p)$ specifies the penalty for pixel p to have label f_p, encouraging each pixel to have a label of small penalty. The smoothness term $V_{pq}(f_p, f_q)$ encourages spatial consistency by penalizing neighbors 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 a 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 p and q.

For Potts model, in case of two labels, the energy in Eq. (1) can be minimized exactly, by finding a minimum cut on a certain graph [2]. In [14] they show which two-label energies can be optimized exactly with a graph cut. In the multi-label case, exact optimization is NP-hard, but a solution 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 with a minimum cut [2]. The expansion algorithm iterates over all labels α, finding the best α-expansion, until convergence.

In addition to spatial consistency, V_{pq} can model label ordering constraints. For example, if p is a neighbor 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 does not hold, even though the optimization problem may be easier, see Sec. III-D.

III. Order-Preserving Moves

In this section we define and explain the properties of order-preserving moves for our five part model. The following abbreviation is used: L, R, T, B, C, correspond to "left", "right", "top","bottom", and "center", respectively. Table I gives V_{pq} terms. For example, if $p < q$ are horizontal neighbors, then $V_{pq}(L, R) = \infty$ and $V_{pq}(L, B) = w_{pq}$, where $w_{pq} > 0$. Table I is the Potts model plus ordering constraints. Under this model, a labeling has a finite energy only if the "center" part is a rectangle, and the L, R, T, and B parts are to the left, right, above, below the C part, respectively. For example, all labelings in Fig. 4 have a finite energy.

A. Motivation

With ordering constraints, α-expansion gets stuck in a local minimum easier. In fact, the factor of two bound does not hold if ordering constraints are added to the Potts model. Authors in [6], [7], who use ordering constraints, cannot achieve good results with α-expansion alone.

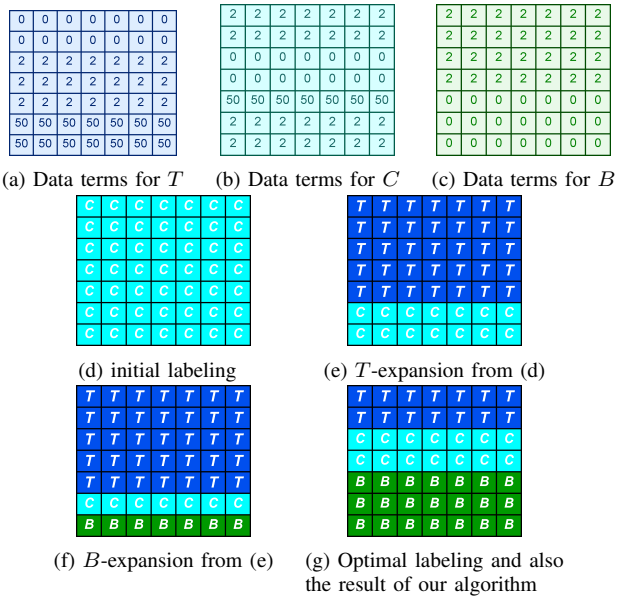
Fig. 2 is a simple example which illustrates the problem. Only the labels T, C and B are possible. The data terms are in Figs. 2(a-c), and the discontinuity cost $w_{pq} = 1$. All pixels are initialized with C, Fig. 2(d). After a T-expansion the result is as in Fig. 2(e). After a B-expansion the result is as in (f). This is, in fact, the final labeling since due to the ordering constraints, no other expansion move gives a lower energy. The energy of this local minimum is 70. The global minimum, with the energy of 14, is in Fig. 2(g), and it is very far from the local minimum. Our algorithm finds the global optimum in Fig. 2(g), initialized with Fig. 2(d).

Note that with the ordering constraints in Table I, optimization of the energy in Eq. (1) is no longer NP-hard, see Section III-D. Therefore it is even less acceptable that the expansion algorithm is more prone to getting stuck in a local minimum.

TABLE I
SMOOTHNESS TERMS V_{pq}

Horizontal Neighbors $p=(x,y), q=(x+1,y)$					
$f_p \backslash f_q$	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

Vertical Neighbors $p=(x,y), q=(x,y+1)$					
$f_p \backslash f_q$	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

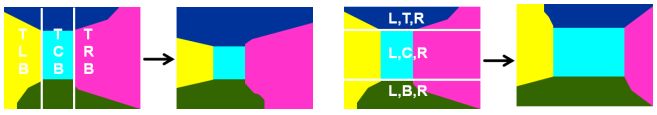


(a) Data terms for T (b) Data terms for C (c) Data terms for B

(d) initial labeling (e) T-expansion from (d)

(f) B-expansion from (e) (g) Optimal labeling and also the result of our algorithm

Fig. 2. Illustration of local minimum problems with α-expansion. The numbers in (a-c) specify data costs for labels T, C, and B. Discontinuity cost is 1. Initial labeling, which is all pixels labeled C, is in (d). After T-expansion and B-expansion, shown in (e) and (f), respectively, the expansion algorithm gets stuck in a local minimum, shown in (f). The optimum labeling is in (g), and our algorithm would give the same result, starting from the same initial labeling as α-expansion, in (d).



(a) Vertical move $f \rightarrow f^v$ (b) Horizontal move $f \rightarrow f^h$

Fig. 3. Illustration of order-preserving moves.

B. Definition of Order Preserving Moves

Our intuition is that to improve on α-expansion moves in presence of ordering constraints, we have to act on more than one label at the same time. 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 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 p. For α-β swap [2], $L_p = \{\alpha, \beta\}$. For global optimization methods in [15], [16], $L_p = \mathcal{L}$, but they can handle only a restricted type of energies, and ours is not of that type.

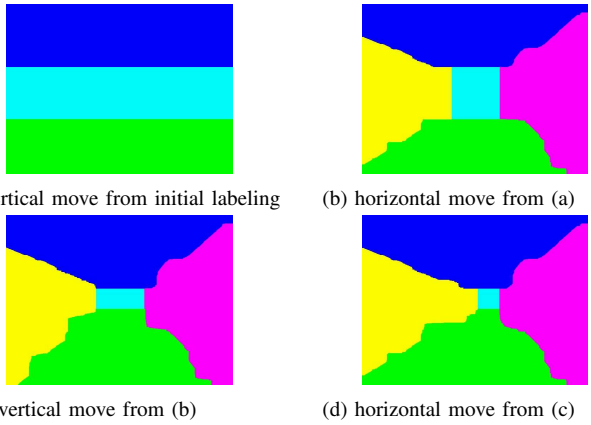
Our insight is that by using different rules when selecting L_p for different pixels, we can have a larger L_p for each pixel, as compared to α-expansion, that is there are more labels to choose from for each pixel in a single move. Notice that the choice of L_p precisely defines the allowed moves. That is a move from f to f' is allowed if $f'_p \in L_p$. We must, therefore, select L_p's in such a way that the allowed move of smallest 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. This ensures that the best allowed move is not worse than the old labeling. We found two such moves and call them *horizontal* order-preserving and *vertical* order-preserving.

First we give an informal illustration of a vertical move from f to f^v. The first requirement is that both f and f^v have a finite energy, i.e. they obey the ordering constraints. Consider Fig. 3(a). Divide f into three rectangles with two vertical lines, one passing through the border of the L and C regions (yellow and cyan) and the other passing through the border of C and R (cyan and magenta) regions. In f^v, pixels in the left rectangle can have labels to T, L or B, pixels in the middle rectangle can have labels to T, C or B, and, finally, pixels in the right rectangle can have labels T, R or B. Notice that the C region can disappear after a vertical move, for example, a labeling can consist of only T's. However if the C region remains, its width is not changed, whereas its height can change arbitrarily. The name "vertical" reflects the fact that C region, if present after the move, can change in the vertical, but not in the horizontal direction.

A horizontal order preserving move from f to f^h is illustrated in Fig. 3(b). Labeling f is divided into three rectangles with two horizontal lines, one passing through the border of the T and C regions (blue and cyan) and the other passing through the border of C and B (cyan and green) regions. In a horizontal order-preserving move, pixels in the top rectangle can switch their labels to L, T or R. Pixels in the middle rectangle can switch their labels to L, C or R, and finally pixels in the bottom rectangle can switch their labels to L, B or R. The name "horizontal" reflects that the C region, if present after the move, can change in the horizontal, but not in the vertical direction.

We now give a formal definition of a vertical order-preserving move $f \rightarrow f^v$. First requirement is that f, f^v have finite energy. Let x_p and y_p be the coordinates 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 $\overline{x} = \max\{x_p | f_p = C\}$. Let L_p^v is the set of allowed labels that p can switch to in a single vertical move, defined as follows. If $x_p < \underline{x}$, then $L_p^v = \{T, L, B\}$. If $\underline{x} \leq x_p \leq \overline{x}$, then $L_p^v = \{T, C, B\}$. Finally, if $x_p > \overline{x}$, then $L_p^v = \{T, R, B\}$. Note that a vertical move finds the global minimum for the example in Fig. 2.

We now give a formal definition of a horizontal order-



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

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

Fig. 4. A sequence of order-preserving moves on a real example. Initial labeling (not shown) was all "center".

preserving move $f \rightarrow f^h$. First requirement is that f, f^h have finite energy. For a horizontal move, L_p^h is defined as follows. Let $\underline{y}$ be the smallest and $\overline{y}$ the largest y coordinate of any pixel that has label C in f. If $y_p < \underline{y}$, then $L_p^h = \{L, T, R\}$. If $\underline{y} \leq y_p \leq \overline{y}$, then $L_p^h = \{L, C, R\}$. Finally, if $y_p > \overline{y}$, then $L_p^h = \{L, B, R\}$.

Notice that the definition of order preserving moves relies on the fact that the C region is present. If this is not the case, then a simplified variant of an order preserving move can be applied, where each pixel has a choice of the same 3 labels. That is either the left or the right rectangle of the illustration in Fig. 3(a) is used for a vertical move, and either the top or the bottom rectangle in Fig. 3(b) is used for a horizontal move. However, in practice, we initialize with a labeling which has every pixel assigned label C, and the C region does not disappear.

C. Optimizing with Order Preserving Moves

Algorithm 1: Optimization with order-preserving moves

foreach $p \in \mathcal{P}$ **do** $f_p^c = C$
$f^v \leftarrow$ the optimal vertical order-preserving move from f^c
$f^h \leftarrow$ optimal horizontal order-preserving move from f^c
if $E(f^v) < E(f^h)$ **then** $f^c = f^v$ **else** $f^c = f^h$
repeat
  $f^v \leftarrow$ optimal vertical order-preserving move from f^c
  $f^c \leftarrow$ optimal horizontal order-preserving move from f^v
until $E(f^c)$ *stops decreasing*

We now explain our optimization procedure, summarized in Algorithm 1. We maintain a current labeling f^c, initialized to all pixels labeled as C. We compute an optimal vertical move f^h and an optimal horizontal move f^v from f^c. Labeling f^c gets updated to whichever of f^h and f^v has a smaller energy. Then we iteratively apply horizontal and vertical moves to f^c until convergence. Fig. 4 illustrates a sequence of labelings obtained for a real example, run to convergence. For the first move (Fig. 4(a)), only T, C, and B are allowed, since the initial labeling is all C's. That is why this move gives horizontal bands of T, C, and B.

D. Properties of Order Preserving Moves

In this section we discuss the properties of the order preserving moves. Here is a summary of the main results. Expansions on labels L, T, B, and R are less general moves than either a horizontal or a vertical order preserving move, and expansion on the C label is not a "large" (i.e. useful) move. The optimization problem is not NP-hard and can be solved exactly (although expensively) with either a horizontal or a vertical move.

We start with an intuitive argument about the advantage of our new moves over expansion moves. An order-preserving move gives every pixel a choice of 3 labels, while α-expansion gives a choice of only 2 labels. In addition, α-expansion effectively acts on only one label, since only the label α is allowed to increase its territory during the move. Our order preserving 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 now show that T and B expansions are strictly contained in the set of vertical order preserving moves, and L and R-expansion moves are strictly contained in the set of all horizontal order preserving moves. We provide the proof only for a T-expansion, the proofs for other case are identical due to symmetry.

Proposition 1: f^T be an T-expansion from a labeling f. Then f^T is also a vertical order preserving move from f.

Proof: When f has no pixels with label C the proof is trivial. In this case, due to ordering constraints, both f and f^T contain at most three labels, and a simplification of an order preserving move described at the end of Sec. III-B would cover expansion on any individual label. Therefore for the remainder of the proof, we assume that f has pixels with label C.

As before, let $\underline{x}$ be the smallest and $\overline{x}$ the largest x coordinate of any pixel that has label C in f. As in Fig. 3(a), divide all pixels vertically into three rectangles, P^l, P^m and P^r, as follows. $P^l = \{p | x_p < \underline{x}\}$, $P^m = \{p | \underline{x} \leq x_p \leq \overline{x}\}$, and $P^r = \{p | x_p > \overline{x}\}$. Due to ordering constraints, if $p \in P^l$ then $f_p \in \{T, L, B\}$, if $p \in P^m$ then $f_p \in \{T, C, B\}$, and, if $p \in P^r$ then $f_p \in \{T, R, B\}$.

Case1: Labeling f^T contains pixels with label C. In this case, due to the ordering constraints, $\{x_p | f_p = C\} = \{x_p | f_p^T = C\}$. In particular, the smallest and largest x coordinate of pixel having label C are the same in f and f^T. Therefore, due to ordering constraints, if $p \in P^l$ then $f_p^T \in \{T, L, B\}$, if $p \in P^m$ then $f_p^T \in \{T, C, B\}$, and, finally, if $p \in P^r$ then $f_p^T \in \{T, R, B\}$. This implies that $f \rightarrow f^T$ is a vertical order preserving move.

Case2: Labeling f^T does not have label C. Then any pixel with label C in f must have label T in f^T. Therefore if $p \in P^m$ then f_p^T must have label T[1]. As for the other two sets, if $p \in P^l$ then $f_p^T \in \{T, L\}$, and if $p \in P^r$ then $f_p^T \in \{T, R\}$, which implies that $f \rightarrow f^T$ is a vertical order preserving move. ■

We already saw that the converse of Proposition 1 is not true. For the example in Fig. 2, the move from (d) to (g) is a valid vertical move, but not a valid expansion on any label.

Only C-expansion is not contained in the set of either horizontal or vertical moves. However, C-expansion is a very weak move when ordering constraints are present. The number of valid C-expansions is less than the number of pixels in the image. Consider Fig. 1(a). After a C-expansion, the top left corner of the new C region must lie along the old boundary between regions L and T, and the bottom right corner must lie along the old boundary between regions R and B. Assuming an image is an $k \times k$ square, there are approximately k^2 expansions. While we can still use C-expansion, we did not find it useful in practice.

Now it is clear that optimizing the energy in Eq. (1) with V_{pq} terms in Table I is not NP-hard. Adding our ordering constraints

[1] If $p \in P^m$ then $f_p^T \neq B$ because ordering constraints would be violated due to the absence of label C in *Case 2*.

to the Potts model makes optimization easier. That is we have a subclass of an NP-hard problem which is not NP-hard itself. Let n be the height of the image. Let f^* be the global optimum of Eq. (1), and let $Y^* = \{y_p | f_p^* = C\}$. Y^* consists of k consecutive y coordinates, $1 \leq k \leq n$. There are exactly $\frac{(n+1)n}{2}$ possible candidates for the set Y^*. We can iteratively find the optimal horizontal order preserving move for each Y^* candidate, and the one with the smallest energy is the global minimum. Since the cost of computing the optimal horizontal move is linear in practice, the total cost is quadratic, which is too expensive. Dynamic approach of [17] could improve the speed.

In addition to the arguments above, we also show experimentally in Sec. V-B that the energies obtained with order-preserving moves are significantly better than those of α-expansion.

IV. Computing the Optimal Order Preserving Move

In this section we give detailed implementation of order-preserving moves for our five part model. We only explain how to find the optimal vertical move. In practice, due to symmetry, we compute the optimal horizontal move by transposing the image, swapping labels L and T, R and B, computing the optimal vertical move, and finally transposing the image back.

A. Computing the Optimal Vertical Order Preserving Move

Given a current labeling f, we need to find the vertical move giving the largest decrease of the energy. As previously, $\underline{x}$ is the smallest and $\overline{x}$ the largest x coordinate of any pixel that has label C in f. Recall that L_p^v is the set of labels that p can switch to in a vertical move. If $x_p < \underline{x}$, then $L_p^v = \{T, L, B\}$, if $\underline{x} \leq x_p \leq \overline{x}$, then $L_p^v = \{T, C, B\}$, and if $x_p > \overline{x}$, then $L_p^v = \{T, R, B\}$.

To find an optimal move, one can use results from [16], [18]. They define a submodular energy in case of multiple *ordered* labels and give a graph construction for global optimization with a minimum cut. A V_{pq} is submodular, if for any $\alpha \leq \beta$, and $\alpha' \leq \beta'$, we have $V_{pq}(\alpha, \alpha') + V_{pq}(\beta, \beta') \leq V_{pq}(\alpha, \beta') + V_{pq}(\beta, \alpha')$. An energy is submodular if every V_{pq} is submodular [16].

It is easy but tedious to check that the vertical move energy with V_{pq}'s in Table I 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^v's for each p in such a way that the energy function restricted to the corresponding move is submodular.

B. Graph Construction for Vertical Order Preserving Move

We now give a graph whose minimum cut corresponds to the optimal vertical order preserving move. Our construction is simpler than in [16]. The number of nodes is only twice larger than for the expansion, so the efficiency is not compromised.

Figs. 5 and 6 illustrate the construction. We create two terminal nodes s and t. Then for each image pixel p, we create two nodes, p_1 and p_2. Let p_0, p_3 be other names for terminals s and t, respectively. Each node p_i is connected to p_{i+1} by an edge e_i^p, for $i = 0, 1, 2$. For any p, if edge e_0^p is severed by the cut[2], then p is assigned label T; if e_2^p is severed by the cut, then p is assigned label B. The meaning of edge e_1^p depends on the value of x_p.

[2]That is the endpoints of edge e_0^p are in two disjoint sets S and T.

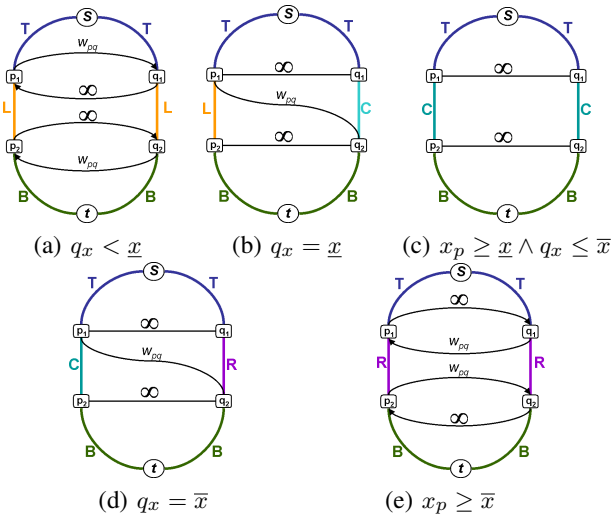


Fig. 5. Part of graph construction for horizontal neighbors $p < q$. All five possible cases are illustrated. Edges without arrows have the same cost in both directions, for example, between p_1 and q_1 in (b). Edges with arrows have different costs in both directions, for example, between p_1 and q_1 in (a).

Let f be the current labeling. Suppose edge e_1^p is severed by the cut. If $x_p < \underline{x}$, then p is assigned label L, if $x_p > \overline{x}$, then p is assigned label R, and, finally, if $\underline{x} \leq x_p \leq \overline{x}$, then p gets label C. In Figs. 5 and 6, each e_i^p is color-coded, depending on the label p gets assigned if e_i^p is severed. Blue, yellow, green, magenta, cyan correspond to labels T, L, B, R, C respectively.

If cutting e_i^p corresponds to assigning pixel p label $l \in \{L, R, T, B, C\}$, then the weight of e_i^p is $D_p(l)$ plus an additional factor, to be discussed below. To prevent cutting more than one e_i^p, we connect p_i to p_{i-1} with an infinite cost edge for $i = 1, 2, 3$.

If pixels p and q are neighbors, we put edges between p_1, p_2 and q_1, q_2. The construction depends on whether p and q are horizontal or vertical neighbors. First we discuss the case of horizontal neighbors. Assume $p < q$, that is p is to the left of q. Since the allowed labels of pixels p and q in a vertical move depend on x coordinates, there are five different cases, illustrated in Fig. 5(a-e). The graph construction is simple to understand. For example, consider Fig. 5(a). The assignment $V_{pq}(T, L) = \infty$ is implemented by the infinite weight edge from q_1 to p_1. Another example is $V_{pq}(L, B) = w_{pq}$, which is implemented through the edge with capacity w_{pq} from q_2 to p_2.

Let $p < q$ be vertical neighbors now, i.e. p is above q. There are three distinct cases, depending on the x coordinates of p and q. All the three cases are handled with exactly the same construction. In Fig. 6, we only illustrate the case $x_p = q_x < \underline{x}$.

Let us identify labels T, L, and B with integers 0, 1 and 2, respectively. Let $l, l' \in \{L, R, T, B, C\}$. According to Table I, we need to implement the following:

$$V_{pq}(l, l') = \begin{cases} 0 & \text{if } l = l' \\ w_{pq} & \text{if } l + 1 = l' \\ \infty & \text{otherwise} \end{cases} . \quad (2)$$

Let c be the initial labeling energy. The infinite weight can be replaced by c. Let r be an even integer s.t. $2^r > c$. Instead of $V_{pq}(l, l')$ in Eq.(2), we can use the equivalent $V_{pq}(l, l') = V_{pq}^1(l, l') + V_{pq}^2(l, l')$, where $V_{pq}^2(l, l') = w_{pq} \cdot (l - l')^r$, and

$$V_{pq}^1(l, l') = \begin{cases} c & \text{if } l > l' \\ 0 & \text{otherwise} \end{cases} .$$

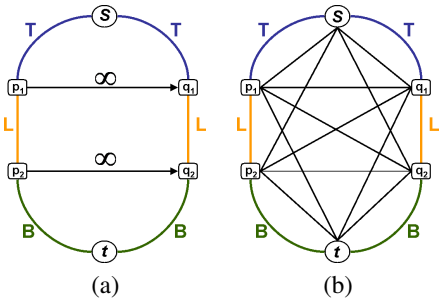


Fig. 6. Part of graph construction for vertical neighbors $p < q$. Out of three cases, only the case $x_p = q_x < \underline{x}$ is shown. For clarity the graph is broken into two graphs in (a) and (b). The actual construction merges (a) and (b).

The construction Fig. 6(a) implements the $V^1_{pq}(l, l')$, with infinite weight replaced by c.

The penalty $V^2_{pq}(l, l')$ is convex and therefore can be implemented by the construction in [15], illustrated in Fig. 6(b). For clarity, the edge weights are omitted. For $i = 0, ..., 3$, and $j = 0, ..., 3$, each node p_i is connected to q_j, except the source ($s = p_0 = q_0$) is not connected to the sink ($t = p_3 = q_3$). The weight of the edge between p_i and q_j is $\frac{w_{pq}}{2}\left(|i-j-1|^k - 2|i-j|^k + |i-j+1|^k\right)$.

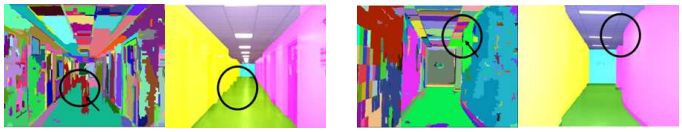
Finally for $i = 0, 1, 2$, the weight of e^p_i is defined as $w(e^p_i) = D_p(i) + \sum_{q \in \mathcal{N}_p} w_{pq} \cdot h(i)$, where $h(i) = \frac{1}{2}[(3-i)^k + (i+1)^k]$, and $\mathcal{N}_p$ is the set of neighbors of p. Integers $0, 1, 2$ are identified with labels T, L, and B. The explanation of why this construction works for $V^2_{pq}(l, l')$ is in [19].

V. Geometric Class Scene Labeling

In this section, we apply the order-preserving moves to the geometric class scene labeling, inspired by Hoiem *et al.* [8]. In [8], 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, [8] extracts only an approximate 3D structure. Traditional 3D reconstruction [20] requires special equipment, such as multiple cameras, or range scanners, etc. Furthermore, the 3D reconstruction methods that are based on pixel correspondences between 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. In addition to [8], [9], there are other single view approximate reconstruction methods. Most require user interaction [21], [22], [23], [24]. Some are automatic [25], [26], but make relatively restrictive assumptions about the scene. See [27] for an extension of [8] to sloped surfaces. Another related work is [28], who show that higher order interactions are useful for the geometric class labeling problem.

Unlike [8], we address the problem in a global optimization framework, using the five part model from Sec. III, and optimizing the energy in Eq. (1) with order-preserving moves. Our model is less general than [8]. Nevertheless, it is still appropriate for many indoor and some outdoor environments. We assume that scene is approximately a "box" and we are looking inside it. We cannot handle "convex" scenes, i.e. looking at a corner of a building.

In a later version, Hoiem *et al.* [9] did try global optimization framework, without a noticeable improvement. Our improvement is probably due to the following factors. In [9], optimization is performed on superpixel level, not on pixel level as we do.



(a) superpixels and final labeling (b) superpixels and final labeling

Fig. 7. Splitting of superpixels, illustrated for two scenes. The areas outlined by a circle contain a large superpixel, brown in (a) and green in (b), which is correctly broken in different regions in the final segmentation. The brown superpixel is broken into the floor and the left wall, the green superpixel is broken into the ceiling and the right wall.

Therefore [9] fails for a superpixel that contains pixels with different true labels. 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, which helps, see Fig. 7. In addition, the stringent set of ordering constraints and better optimization with order-preserving moves contributes to the improvement.

Notice that the ordering constraints ensure 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.

A. Data term

Ideally, we would like to model $D_p(f_p)$ in Eq. (1) as:

$$D_p(f_p) = -\log Pr(f_p|F_p), \tag{3}$$

where F_p is some observed feature vector at pixel p and $Pr(f_p|F_p)$ is the conditional probability of pixel p given feature F_p. However, for geometric labels, image data at a single pixel does not contain enough information to construct a useful likelihood model in Eq. (3).

We take an approach of Hoiem *et al.* [8], 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"[3] using the algorithm of [29]. Fig. 7 shows some superpixels. Then for each superpixel we compute a large set of features similar to those in [8]. The features are the statistics on location, color, geometry, texture and edges of the superpixels. These statistics include the location of the centroid, 10^{th} and 90^{th} percentile of the superpixel position, orientation, ratio of MajorAxis/MinorAxis, shape, eccentricity, mean RGB and HSV values, mean response of the Compass Filters and mean absolute response of DOOG filters. Finally, we use the superpixel feature vectors as training data for the SVM classifier [30].

The output of SVM is an uncalibrated value and not a probability distribution. We use the method proposed by Wu *et al.* [31], which is based on Platt [32] to convert the output of SVM into the distribution $Pr(S = l|F_S)$, where l is a label, F_S is a feature vector computed on superpixel S, and $S = l$ stands for the event that all pixels inside superpixel S have the same label l. Thus, for a given l, we learn the probability that all pixels inside a superpixel S have label l (under the assumption that all pixels within a superpixel should have the same label).

Ideally, we would like to learn the probability that a single pixel p has label l. As we already mentioned, we cannot learn these probabilities directly since there is not enough image information

[3] A superpixel is an image region returned by a segmentation algorithm.

Fig. 8. Sample indoor images.

Fig. 9. Sample outdoor images.

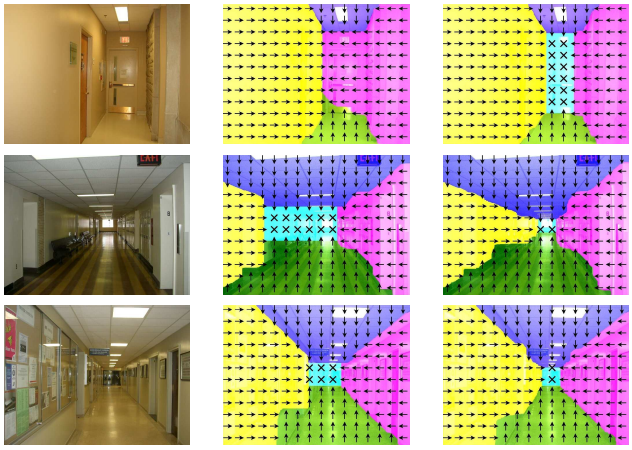
at an individual pixel p. Our solution is to simply to apply the distributions learned on the superpixels to the pixel based data term $D_p(f_p)$. That is $D_p(f_p) = -\log Pr(S^p = f_p|F_{S^p})$, where S^p is the superpixel that contains p and $\log Pr(f_p|F_{S^p})$ is the learned log probability of label f_p given the superpixel feature vector F_S s.t. $p \in S$. This approach makes sense since the energy in Eq. (1) does not require the true pixel based negative log probabilities. It is sufficient to come up with a reasonable penalty scheme for the $D_p(f_p)$ term, that is the scheme that for a given pixel p imposes higher penalties for the less likely labels. It is a reasonable assumption that if $Pr(S = l_1) < Pr(S = l_2)$, then for most pixels $p \in S$, $Pr(p = l_1) < Pr(p = l_2)$.

B. Results

We have collected 600 images from different indoor environments, and downloaded 84 outdoor street images from the Web and PASCAL database. Figs. 8 and 9 show some samples. All images were manually labeled. We used half of the images for training and half for testing, separately for indoor/outdoor collections.

In Figs. 10 and 11, for indoor and outdoor images, we compare the results of: (b) SVM classification, (c) α-expansion without ordering constraints, and (d) our order-preserving moves. First, let us compare the labelings produced by SVM (Fig. 10 (b) and Fig. 11 (b)) to the labelings produced with graph cut optimization (Fig. 10 (c,d) and Fig. 11 (c,d)). As expected, the SVM labelings are not nearly as spatially consistent as those obtained with graph cuts. Furthermore, incorporating spatial smoothness corrects not only small spurious regions, but also sometimes large erroneously labeled regions. For example, in the 4^{th} row in Fig. 10 (b), SVM fails to label most of the floor correctly. The spatial smoothness constraints help to label most of the floor correctly for the same image in Figs. 10(c,d). Also in the 1^{st} row in Fig. 11 (b), SVM fails to label large areas of ground correctly. The spatial smoothness constraints helps label most of the ground correctly for the same images in Figs. 11(c,d).

We now compare graph cuts with and without ordering constraints. Figs. 10 and 11 have the results of graph cuts without ordering constraints computed with α-expansion in columns (c), and the results with ordering constraints computed with the order preserving moves in columns (d). Implausible regions are frequent in columns (c) both in Fig. 10 and Fig. 11. For example, the back wall patch appears in the middle of the left wall in row 3; the left wall patches appear in the middle of the back wall in row 1 and floor in row 3. In almost all images shown in Fig. 10 (c), the back wall is present with a significantly distorted shape compared to Fig. 10 (d). In Fig. 11(c), in rows $2-4$, the left-side



(a) (b) (c)

Fig. 12. Indoor result comparison: (a) original image, (b) α-expansion with ordering constraints, (c) order-preserving moves.

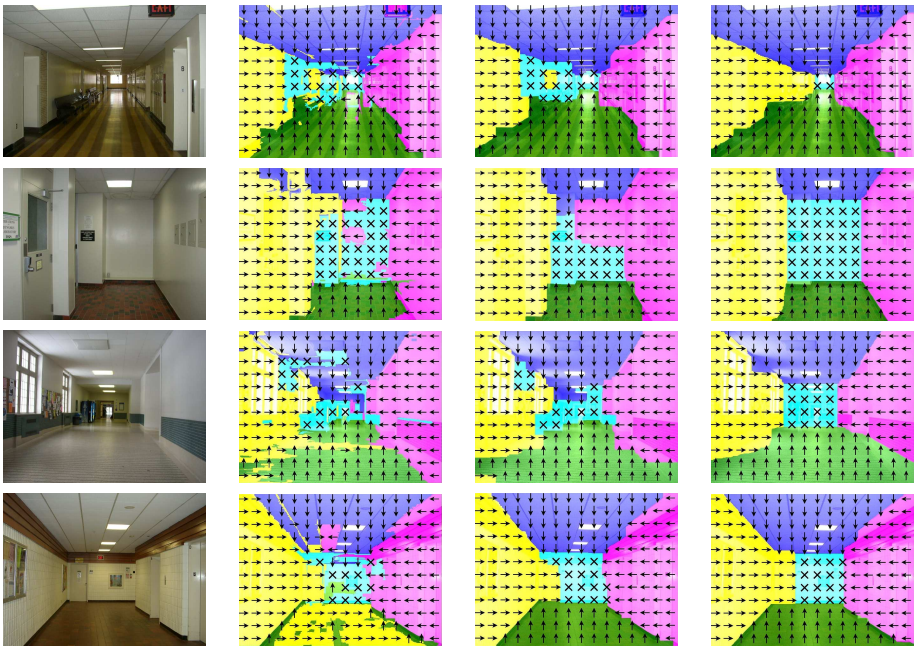
patches and the right-side patches are connected directly without the back-side patches in between. Clearly, ordering constraints tend to guide optimization away from implausible solutions.

We now compare the order-preserving moves and α-expansion on the same energy, i.e. the energy with ordering constraints. The order-preserving moves always give a smaller energy compared to α-expansion. On average, the energy is 27.3% smaller (σ = 9.8%). Some results are in Figs. 12 and 13. As expected, α-expansion gets stuck in a local minimum easier. For many outdoor scenes almost all the back-side parts are tiny, only a few pixels in size.

Figs. 14 and 15 show more results, illustrating the accuracy we can achieve without user interaction. Not all geometric parts have to be present, for example, the last image in the 1^{st} row in Fig. 14 does not have a right wall, but we produce correct results.

Since the data term in our energy function is based on the probabilities generated by SVM, when SVM gives reasonable probabilities, our algorithm can significantly improve SVM results. However, when SVM results are far from reasonable, the order-preserving moves can worsen them even further, trying to satisfy the ordering constraints that can not be reasonably satisfied (see Fig. V-B for some of the failure cases). Therefore, the overall accuracy improvement (see Table II) over SVM computed for all the images is not that large. However, when SVM results are not reasonable, they are hardly useful for applications anyway.

To illustrate how our accuracy improvement depends on the accuracy of SVM results, 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.



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

Fig. 10. Indoor result comparison: (a) original images (b) SVM labeling, (c) α-expansion without ordering constraints (d) order-preserving moves.

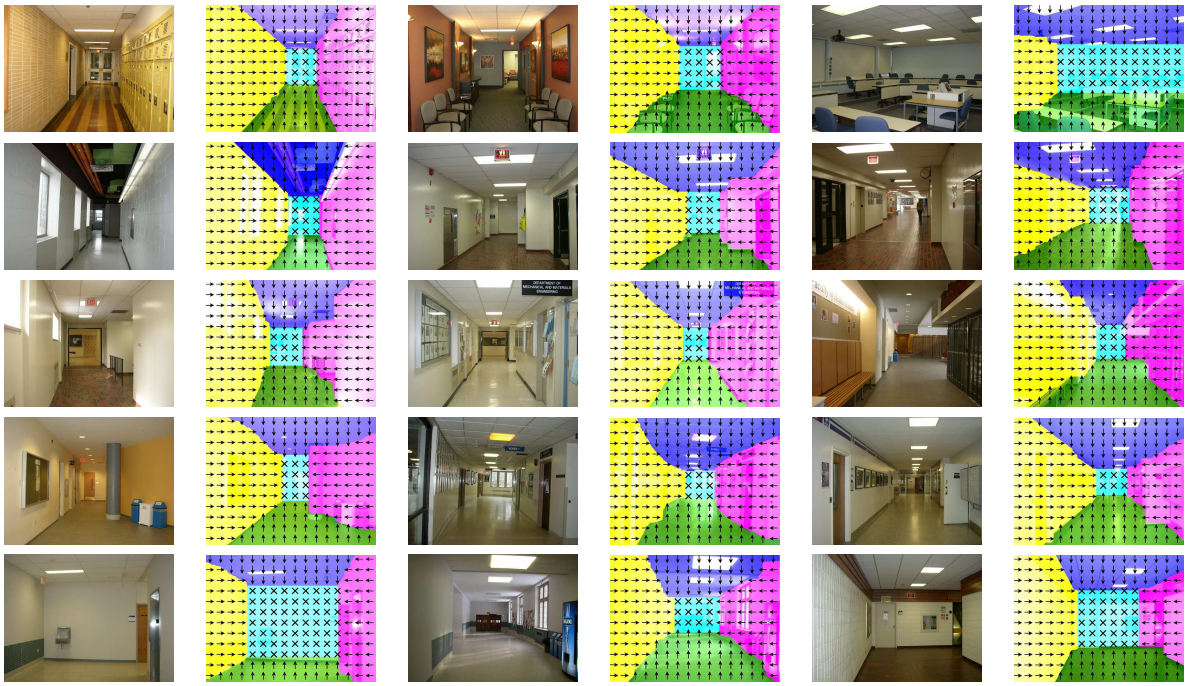
(a) (b) (c) (d)

Fig. 11. Outdoor result comparison: (a) original images (b) SVM labeling, (c) α-expansion without ordering constraints (d) order-preserving moves.

Fig. 16 shows the accuracy of the algorithms for each bin. For the worst bin (unreliable SVM results), accuracy of order-preserving moves is worse than that of SVM labeling for outdoor images. For the best bins (very accurate SVM results), order-preserving moves do not improve SVM results significantly since there is not much room for improvement. However, in the middle range (from about 4th bin to the 8th bin), there is a noticeable improvement over SVM and α-expansion, especially for the outdoor images. For example, in the 6th bin for the outdoor images, order-preserving moves have about 80% accuracy, followed by approximately 75% accuracy for α-expansion and SVM.

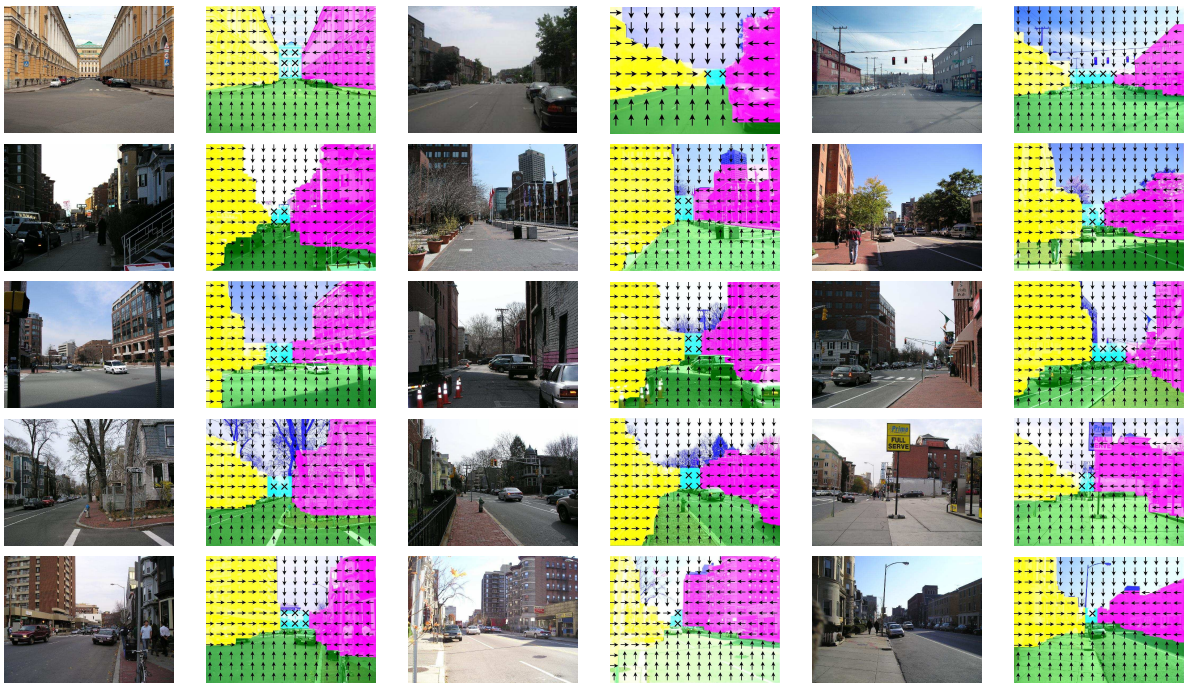
Fig. 17 shows the percentage of labelings that have the at least the accuracy rate specified on the horizontal axis. For example, for indoor images, 52% of order-preserving labelings have the accuracy rate of at least 90%, whereas only 33% and 46% of SVM and α-expansion labelings, respectively, have this rate. Order-preserving moves always have a higher percentage of images at any given accuracy rate in the range between 75% and 100%.

Some failures are in Fig. V-B. Most failures occur when the "center" data terms are far from reasonable, as in (b). The ordering



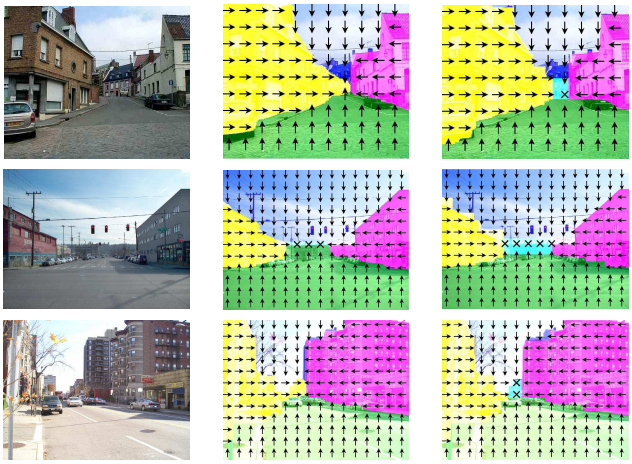
(a) (b) (a) (b) (a) (b)

Fig. 14. Some results on indoor images: (a) original images, (b) order-preserving moves.



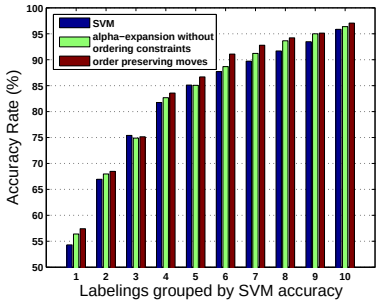
(a) (b) (a) (b) (a) (b)

Fig. 15. Some results on outdoor images: (a) original image, (b) order-preserving moves.

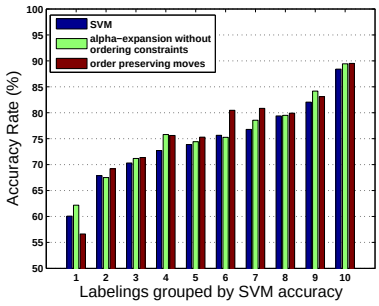


(a) (b) (c)

Fig. 13. Outdoor result comparison: (a) original images, (b) α-expansion with ordering constraints, (c) order-preserving moves.

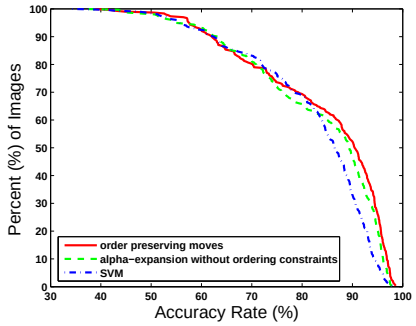


(a) indoor images

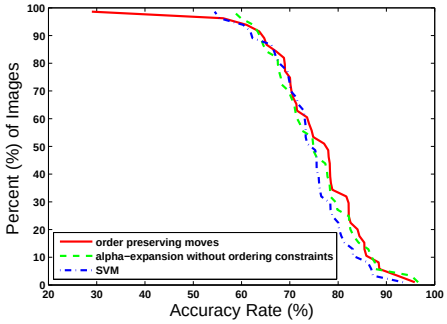


(b) outdoor images

Fig. 16. Accuracy comparison: bins sorted by quality vs. accuracy rate. SVM results are grouped in 10 equal bins, ordered from least accurate to most accurate.

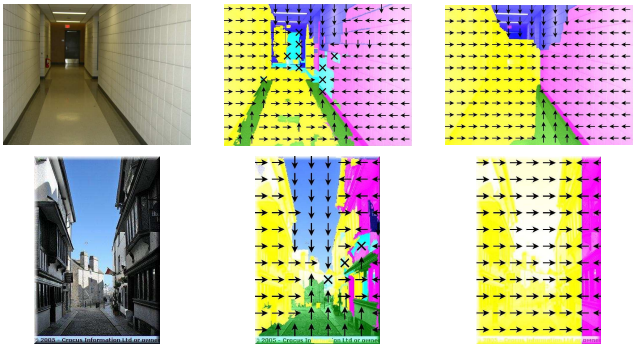


(a) indoor images



(b) outdoor images

Fig. 17. Accuracy rate vs. % of images.



(a) (b) (c)

Failure cases: (a) original images, (b) SVM labeling, (c) Order-preserving moves.

constraints are not violated in (c), but the "center" region is too thin to see at this resolution.

Even a labeling with a relatively good overall accuracy may be unsuitable for applications. For a virtual-walk through, it is the quality of the spidery mesh (Sec. V-C) that is important. We computed the percentage of "successful" labelings based on the spidery mesh. A labeling is successful if a spidery mesh generated from it satisfies the following. At least 7 out of 8 radial lines have less than 10 degree slope difference and 5 pixel intercept difference from the ground truth spidery mesh (the ground truth spidery mesh is generated from the ground truth labeling).

Table II summarizes the performance of SVM, α-expansion without and with ordering constraints, and the order-preserving moves (in that order). Order-preserving moves 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. Table II also shows that order-preserving move method is computationally more efficient than α-expansion. The average processing time in Table II was calculated on a PC with 2.4GHz CPU and 2048MB memory. The time includes superpixel segmentation, feature extraction, data terms calculation, and the corresponding energy minimization. We use the efficient max-flow algorithm of [13] for min-cut computation.

We also compare our results to Hoiem [9], using their code [4]. Given an input image, they assign each pixel its geometric class label, and also output confidence values for each geometric class. Since we have different training data and do not have access to their code to train the classifier, we do the following to make

[4]http://www.cs.uiuc.edu/homes/dhoiem/software/

TABLE II
PERFORMANCE SUMMARY

	Percentage of Successful Labelings (%)			
	SVM	α-exp. no OC	α-exp. with OC	OP moves
Indoor	59.7	73.3	77.1	78.1
Outdoor	41.4	55.1	51.7	65.6
	Overall Accuracy Rate (%)			
	SVM	α-exp. no OC	α-exp. with OC	OP moves
Indoor	83.0	84.1	84.7	85.0
Outdoor	74.0	75.2	71.0	75.3
	Processing Time (seconds)			
	SVM	α-exp. no OC	α-exp. with OC	OP moves
Indoor	34.5	45.5	85.5	62.3
Outdoor	29.3	43.8	286.4	56.5

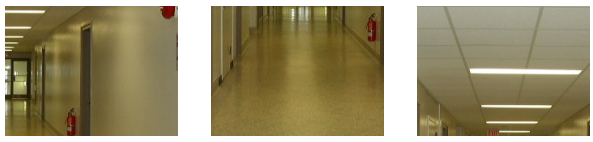
TABLE III
CONFUSION MATRIX FOR THREE CLASS PROBLEM
(ORDER-PRESERVING MOVES/HOIEM *et al.* [9])

Indoor			
	B (%)	V (%)	T (%)
B	86.5/79.3	13.4/20.7	0.11/0.0
V	2.1/2.6	95.2/96.1	2.7/1.3
T	0.0/0.0	10.5/24.5	89.5/75.5
Overall accuracy: 90.5/83.7 (%)			
Outdoor			
	B (%)	V (%)	T (%)
B	85.3/83.9	14.7/16.1	0.0/0.0
V	4.8/2.5	91.1/95.3	4.1/2.2
T	0.0/0.0	12.5/33.2	87.5/66.8
Overall accuracy: 87.9/82.0 (%)			

TABLE IV
CONFUSION MATRIX FOR FIVE CLASS PROBLEM
(ORDER-PRESERVING MOVES/HOIEM *et al.* [9])

Indoor images					
	B (%)	L (%)	C (%)	R (%)	T (%)
B	88.8/87.6	4.5/5.2	1.3/1.9	5.4/5.2	0.1/0.0
L	7.6/9.1	86.3/82.5	3.4/5.2	0.8/1.4	1.9/1.8
C	5.1/5.6	15.0/13.9	58.9/62.7	18.9/15.9	2.1/2.0
R	2.6/3.2	1.4/1.9	4.1/5.2	88.7/86.4	3.3/3.3
T	0.0/0.0	3.9/3.8	3.1/5.2	5.3/5.6	87.6/85.4
Overall accuracy: 84.1/82.9 (%)					
Outdoor images					
	B (%)	L (%)	C (%)	R (%)	T (%)
B	96.8/94.5	1.1/1.4	1.5/2.7	0.6/1.4	0.0/0.0
L	7.3/7.8	72.9/62.6	15.9/25.1	0.0/1.5	3.8/3.1
C	7.0/9.1	16.1/17.6	64.4/54.4	6.1/9.4	6.4/9.5
R	10.5/10.6	0.0/2.5	20.3/31.4	61.0/52.4	8.2/3.0
T	0.2/0.2	4.3/6.4	14.6/10.3	2.0/3.0	78.9/80.2
Overall accuracy: 74.8/68.8 (%)					

(a) (b) (c)



(d) (e) (f)

Fig. 19. Virtual scene walk-through from our labeling: (a) a spidery mesh, (b) walk forward, (c) look left, (d) look right, (e) look down, (f) look up.

the comparisons fair. Instead of using the data terms produced by our SVM classifier, we use the confidence values produced for each geometric class as the data term in our optimization framework. Therefore what we are measuring is how much spatial smoothness, ordering constraints, and optimization with order-preserving moves can improve on the classification approach of [9]. Another issue we have to address before comparison is the number of classes. In [8], they have three main classes "sky", "ground", and "vertical". The "vertical" class is further subdivided into "facing left", "facing right", "facing camera", "porous", and "solid non-planar". The classes "sky", "ground", "facing left", "facing right", "facing camera" are equivalent to our classes "top", "bottom", "left", "right", "center". However, classes "porous" and "solid non-planar" are not equivalent to any of our classes. To address this issue, we perform two different experiments.

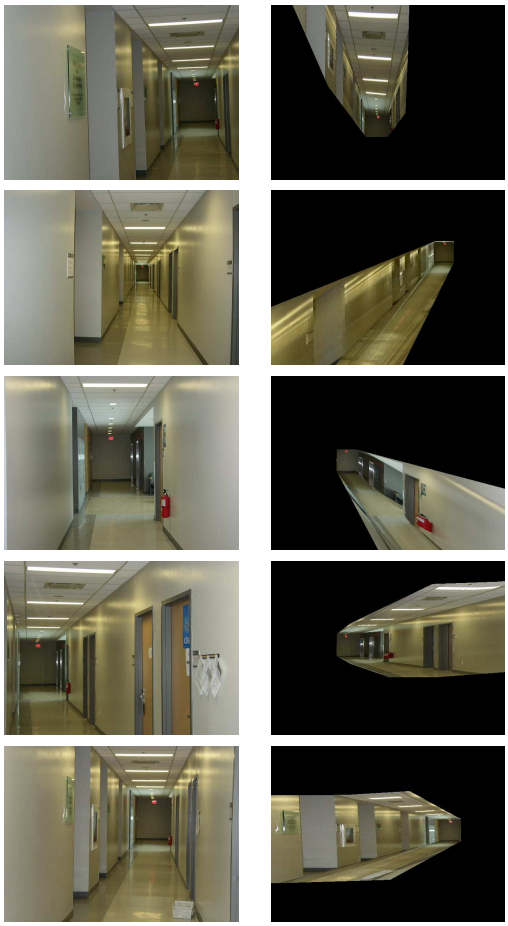
First, we reformulate the problem as 3 class classification, with classes "top", "vertical", "bottom". The vertical class combines our "left", "right", and "center" classes, and for [8] it combines all their classes except the "sky" and "ground", since all the other classes correspond to the "vertical" structures. When combining subclasses, their confidence levels are added up to form the confidence level for the larger class. The results are in Table III, where for each table entry the format is (order-preserving moves accuracy)/(accuracy in [9]). For example, for the "bottom" class, our accuracy is 86.5%, and the accuracy in [9] is 79.3%.

(a) (b) (c)

Fig. 18. (a) original image, (b) labeled image, (c) generated spidery mesh.

We also evaluated the five class problem by simply ignoring the confidence maps for the "porous" and "solid non-planar" classes, and renormalizing the confidence values for each pixel to sum up to 1. This makes sense because we are excluding irrelevant classes from consideration. The results are in Table IV. In both cases (Table III and Table IV), our algorithm significantly improves on the performance of the classifier in [9]. It is interesting to note that in the three class case, our improvement over [9] is larger. This may be because in the three class case we do, in fact, find the global optimum of the energy function, whereas in the five class case we are not guaranteed the global optimum.

We also evaluated our algorithm on the original data from [9]. This database contains outside images that are mostly inappropriate for our five-part model. Our overall accuracy rate for the 5 main classes was 57.6%, compared to 64.3% of the approach in [9]. This is as expected, since the majority of scenes significantly deviate from our model. We also hand-selected 44



(a) (b)

Fig. 20. 3D reconstruction results: (a) original images, (b) novel views.

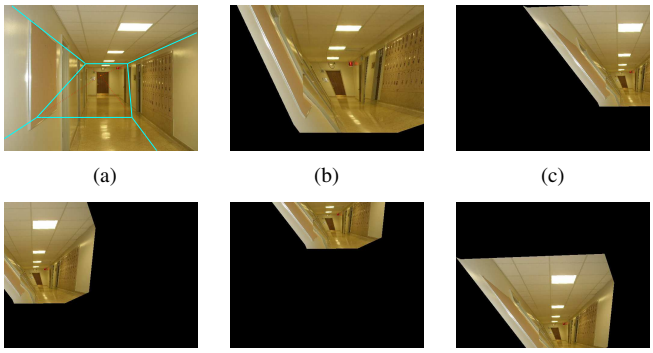
images from the datable in [9] that are appropriate for our model. On this smaller subset, our accuracy is 74.7%, which compares favorably to the 72.6% accuracy of [9] on the same subset.

C. Applications

We now illustrate the use of the obtained scene structure for automatic 3D reconstruction from a single view and virtual scene walk-through. We use a spidery mesh to fit perspective projection and mimic 3D camera transformations to navigate through the scene [21]. Spidery mesh is composed of four parts (vanishing point, radial lines, inner and outer rectangles), which partitions the 2D image into five regions (left, right, rear, 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 [33] 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. Fig.18 shows an example of spidery mesh generation.

Parts of the virtual scene walk-through for an indoor image are in Fig. 19. Some of the novel view 3D reconstruction results for the indoor images are shown in Fig. 20.

Fig. 21 shows that even a relatively good SVM labeling (more than 90% accuracy, in this case) may fail to produce satisfactory results. The room appears to have crooked walls and floor.



(a) (b) (c)

(d) (e) (f)

Fig. 21. Scene walk-through using an SVM labeling: (a) spidery mesh, (b) walk forward, (c) look left, (d) look right, (e) look down, (f) look up.

VI. Shape Prior for Segmentation

Shape priors for segmentation in general [34], [35], [36] and segmentation with a graph-cut [37], [38] is an area of 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 [37], [38] have to register the shape model with the image during the segmentation process, which is a difficult task in itself.

Instead of a shape prior specific to some object, like in [37], [38], we implement simple generic shapes such as "rectangle", "trapezoid", etc. 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, see Fig. 4. 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 are the background. This is a new approach to shape priors. It is the relative order of the parts that enforce a certain object shape. In [10], they use a similar idea but only for rectangles.

We now explain how to incorporate simple geometric shape priors in graph-cut segmentation of an object from its background. For a rectangle, we use the same V_{pq} as in Table I, 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 can be rotated). To get a trapezoid, we relax the following constraints in Table I: 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 is not, strictly speaking, a true trapezoid, since we cannot enforce the borders between the L and C regions and C and R to be straight lines.

For a parallelogram shape prior, we need to set $V_{pq}(L, C) = V_{pq}(C, R) = w_{pq}$ instead of ∞ in Table I(b). Essentially, the shape prior implemented with our five-part model is limited to rectangles, trapezoids, and parallelograms. For other shapes, we would need models with more than five parts and we would need to generalize order-preserving moves to those models.

For a rectangle prior, the order preserving moves stay exactly

the same. For the trapezoid and parallelogram, some hard constraints have been relaxed, so in the graph, the corresponding infinite weights are replaced by finite weights in Figs. 5 and 6.

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-specific D_p can always be added, of course. We do have to set D_p for any p on the image border. We set each border pixel 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.

Our cost function is the sum of w_{pq}'s on the object boundary. To avoid a trivial one-pixel solution, we make w_{pq}'s negative whenever there is a stronger than average intensity edge between p and q, biasing segmentation towards a longer boundary coinciding with intensity edges. Specifically, we set $w_{pq} = \sigma - |I_p - I_q|$, where I_p is the image intensity of pixel p, and σ is the average absolute intensity difference between neighboring pixels.

In general, making $w_{pq} < 0$ is not always possible in graph-cut optimization framework, but it is possible for our vertical/horizontal moves. Let us consider the vertical order preserving move, since the horizontal move is handled identically. First recall that for neighboring pixels p and q, $V_{pq}(l, l')$ may be negative only if exactly one of l or l' is label C. For horizontal neighbors p and q this corresponds to the construction in Figs. 5 (b) or (d). In this case, the edge with w_{pq} can be simply removed from the graph and the negative w_{pq} can be added to the yellow edge marked with C, i.e. the edge connecting pixel q_1 to q_2 in (b) and p_1 to p_2 in (d). The reason why a negative cost can be always added to these yellow edges is that the reverse edges (i.e. the edge connecting pixel q_2 to q_1 in (b) and p_2 to p_1 in (d)) have infinite weight.

In a vertical move, the number of vertical neighbors p and q such that V_{pq} is non-zero is a constant equal to the width of the central region, which stays fixed during the vertical move. Therefore, if pixels p and q are vertical neighbors we simply add a large enough positive constant (specifically we add $|w_{pq}|$ such that w_{pq} is the smallest in the graph) to all vertical neighbors to remove all the negative costs from the graph.

Figs. 22 and 23 show the results with a rectangular and a trapezoid prior, illustrating the ability to pick out interesting regions obeying the corresponding shape priors without any knowledge of the object/background regional properties. All results were obtained with the same parameter settings.

VII. Conclusions

We show the importance of choosing a right optimization method for a given optimization problem. We explain why with the ordering constraints, the popular α-expansion does not work as well. For a five part model with ordering constraints, we develop new graph-cut moves that are more general than α-expansion in theory and work significantly better in practice. The limitation of current approach is that it is model-specific, and while it is possible to extend it to more general models, significant time overhead is required in designing order-preserving moves for a new model. In the future, we plan to explore, given a model with ordering constraints, how to infer the set of all possible order-preserving moves automatically.

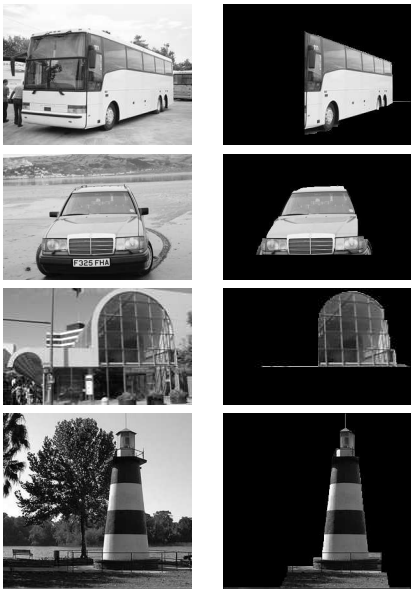


Fig. 23. Trapezoid shape prior. Left: originals, right: segmentations.

References

[1] 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 *European Conference on Computer Vision*, 2006, pp. II: 16–29.

[2] Y. Boykov, O. Veksler, and R. Zabih, "Fast approximate energy minimization via graph cuts," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 11, pp. 1222–1239, 2001.

[3] V. Kolmogorov, "Convergent tree-reweighted message passing for energy minimization," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 10, pp. 1568–1583, 2006.

[4] J. Yedidia, W. Freeman, and Y. Weiss, "Bethe free energy, kikuchi approximations, and belief propagation," in *Workshop on Statistical and Computational Theories of Vision*, 2001.

[5] V. Kolmogorov and R. Zabih, "Computing visual correspondence with occlusions via graph cuts," in *International Conference on Computer Vision*, 2001, pp. II: 508–515.

[6] J. Winn and J. Shotton, "The layout consistent random field for recognizing and segmenting partially occluded objects," in *Conference on Computer Vision and Pattern Recognition*, 2006, pp. I: 37–44.

[7] D. Hoiem, C. Rother, and J. Winn, "3d layout crf for multi-view object class recognition and segmentation," in *Conference on Computer Vision and Pattern Recognition*, 2007, pp. 1–8.

[8] D. Hoiem, A. Efros, and M. Hebert, "Geometric context from a single image," in *International Conference on Computer Vision*, 2005, pp. 654 – 661.

[9] ——, "Recovering surface layout from an image," *International Journal of Computer Vision*, vol. 75, no. 1, pp. 151–172, October 2007.

[10] J. Keuchel., "Multiclass image labeling with semidefinite programming," in *European Conference on Computer Vision*, vol. II, 2006, pp. 454–467.

[11] X. Liu, O. Veksler, and J. Samarabandu, "Graph cut with ordering constraints on labels and its applications," in *Conference on Computer Vision and Pattern Recognition*, 2008.

[12] L. Ford and D. Fulkerson, *Flows in Networks*. Princeton University Press, 1962.

[13] Y. Boykov and V. Kolmogorov, "An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 9, pp. 1124–1137, 2004.

[14] V. Kolmogorov and R. Zabih, "What energy function can be minimized via graph cuts?" *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 2, pp. 147–159, February 2004.

[15] H. Ishikawa, "Exact optimization for markov random fields with convex priors," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, pp. 1333–1336, 2003.

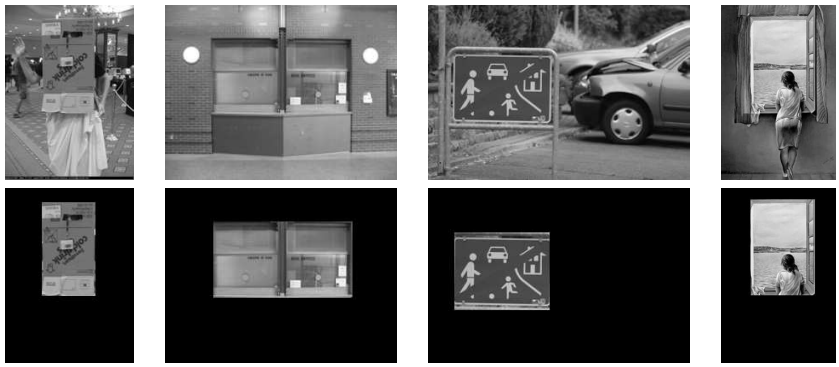


Fig. 22. Rectangle shape prior: originals on the top, segmentations on the bottom.

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

[17] P. Kohli and P. Torr, "Dynamic graph cuts for efficient inference in markov random fields," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 12, pp. 2079–2088, December 2007.

[18] J. Darbon, "Global optimization for first order markov random fields with submodular priors," in *12th International Workshop on Combinatorial Image Analysis*, 2008.

[19] O. Veksler, "Graph cut based optimization for mrfs with truncated convex priors," in *Conference on Computer Vision and Pattern Recognition*, 2007, pp. 1–8.

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

[21] Y. Horry, K. Anjyo, and K. Arai, "Tour into the picture: using a spidery mesh interface to make animation from a single image," in *ACM SIGGRAPH*, 1997, pp. III:225–232.

[22] P. E. Debevec, C. J. Taylor, and J. Malik, "Modeling and rendering architecture from photographs: a hybrid geometry- and image-based approach," in *ACM SIGGRAPH*, 1996, pp. 11–20.

[23] H. Shum, M. Han, and R. Szeliski, "Interactive construction of 3d models from panoramic mosaics," in *Conference on Computer Vision and Pattern Recognition*, 1998, pp. 427–433.

[24] A. Criminisi, I. D. Reid, and A. Zisserman, "Single view metrology," *International Journal of Computer Vision*, no. 2, pp. 123–148, 2000.

[25] E. Delage, H. Lee, and A. Y. Ng, "A dynamic bayesian network model for autonomous 3d reconstruction from a single indoor image," in *Conference on Computer Vision and Pattern Recognition*, 2006, pp. II:2418 – 2428.

[26] J. Coughlan and A. Yuille., "Manhattan world: Compass direction from a single image by bayes. inf." in *International Conference on Computer Vision*, 1999.

[27] A. Saxena, S. Chung, and A. Ng, "3-d depth reconstruction from a single still image," *International Journal of Computer Vision*, vol. 76, no. 1, pp. 53–69, January 2008.

[28] S. Ramalingam, P. Kohli, K. Alahari, and P. Torr, "Exact inference in multi-label crfs with higher order cliques," in *Conference on Computer Vision and Pattern Recognition*, June 2008.

[29] P. Felzenszwalb and D. Huttenlocher, "Efficient graph-based image segmentation," *International Journal of Computer Vision*, vol. 59, no. 2, pp. 167 – 181, 2004.

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

[31] T. F. Wu, C. J. Lin, and R. Weng, "Probability estimates for multi-class classification by pairwise coupling," *Journal of Machine Learning Research*, vol. 5, pp. 975–1005, 2004.

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

[33] 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*, vol. 24, pp. 381–395, 1981.

[34] M. Leventon, W. Grimson, and O. Faugeras, "Statistical shape influence in geodesic active contours," in *Conference on Computer Vision and Pattern Recognition*, 2000, pp. 316–323.

[35] M. Rousson and N. Paragios, "Shape priors for level set representations," in *European Conference on Computer Vision*, 2002, pp. 416–418.

[36] D. Cremers, S. Osher, and S. Soatto, "Kernel density estimation and intrinsic alignment for shape priors in level set segmentation," *International Journal of Computer Vision*, vol. 69, pp. 335–351, 2006.

[37] D. Freedman and T. Zhang, "Interactive graph cut based segmentation with shape priors," in *Conference on Computer Vision and Pattern Recognition*, 2005, pp. 755–762.

[38] M. Kumar, P. Torr, and A. Zisserman, "Obj cut," in *Conference on Computer Vision and Pattern Recognition*, 2005, pp. I: 18–25.

Xiaoqing Liu received the B.Eng degree in Electrical Engineering from Xi'an Jiaotong University, China in 1998. She received M.E.Sc and Ph.D degrees in Electrical and Computer Engineering from the University of Western Ontario in 2005 and 2008. She is currently a research scientist in the UtopiaCompression Corp. Los Angeles, USA. Her research interests include Computer Vision and Pattern Recognition, Image Analysis and Understanding, Image Processing and Visualization, and Computer Graphics. She is a member of the IEEE.

Olga Veksler received BS degree in mathematics and computer science from New York University in 1995 and a PhD degree from Cornell University in 1999. She was a postdoctoral research associate at NEC Research Institute. She is currently an assistant professor with Computer Science Department University of Western Ontario. Her research interests are energy minimization methods, graph algorithms, stereo correspondence, motion, and segmentation. She is a member of the IEEE.

Jagath Samarabandu received B. Sc (Eng) in Electronics and Tele-communication with first class honours from the University of Moratuwa, Sri Lanka in 1982. He was awarded the Fulbirght scholarship in 1987. He received M.S and Ph.D. degrees in Electrical Engineering from State University of New York at Buffalo in 1990 and 1994. He has been with the Department of Electrical and Computer Engineering at the University of Western Ontario since 2000. His research interests include image analysis, pattern recognition and intelligent systems. He is a member of the IEEE.