Fast Dynamic Programming for Labeling Problems with Ordering Constraints

Junjie Bai* Qi Song* Olga Veksler† Xiaodong Wu*

*Department of Electrical and Computer Engineering
The University of Iowa
junjie-bai, qi-song, xiaodong-wu@uiowa.edu

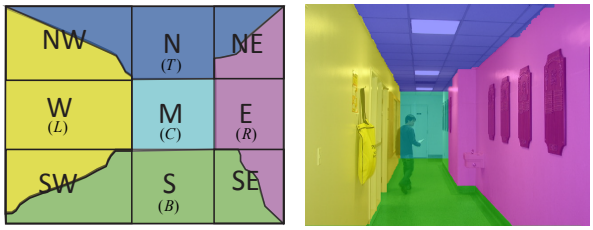
†Computer Science Department
University of Western Ontario
olga@csd.uwo.ca

Abstract

Many computer vision applications can be formulated as labeling problems. However, multilabeling problems are usually very challenging to solve, especially when some ordering constraints are enforced. We solve in this paper a five-parts labeling problem proposed in [6, 7]. In this model, one wants to find an optimal labeling for an image with five possible parts: "left", "right", "top", "bottom" and "center". The geometric ordering constraints can be read naturally from the names. No previous method can solve the problem with globally optimal solutions in a linear space complexity. We propose an efficient dynamic programming based algorithm which guarantees the global optimal labeling for the five-parts model. The time complexity is $O(N^{1.5})$ and the space complexity is $O(N)$, with N being the number of pixels in the image. In practice, it runs faster than previous methods. Moreover, it works for both 4-neighborhood and 8-neighborhood settings, and can be easily parallelized for GPU.

1. Introduction

Given an image, the multilabeling problem seeks to assign a label to each pixel from a set of fixed labels, which, in general, is NP-hard [1]. Multilabel oprimization is a very active area of research in computer vision since a wide variety of vision problems can be formulated as multilabeling problems. Ishikawa developed an exact optimization method for Markov Random Fields with convex priors [5], which was among the first computational frameworks for efficiently solving the multilabeling problems. Wu and Chen's algorithm for convex multilabeling works in a more general setting, restricting the label transition between two neighboring pixels in a certain range of the linearly ordered labels [10]. For more general cost functions, Boykov*et al.*'s α-expansion based graph cut approach [1] is widely used in the vision community due to its accuracy and efficiency. However, this method provides no optimality guarantees in some cases, including the cases studied in this paper.



(a) (b)

Figure 1. The five-parts labeling model. (a) The partition of the image into nine regions. The letter in the parenthesis indicates the label of each region. (b) An example labeling. Although the person in the hall way occludes significant area of the image, our method can correctly label the end of the hallway.

Recently, the introduction of label ordering constraints into the multilabel optimization, which allows substantially generalized cost functions, attracts noticeable attention [3, 6, 7, 8]. In [6], Liu *et al*. proposed a five-parts labeling model with the ordering constraints, in which the image is to be labeled into five parts, namely, "left", "right", "top", "bottom" and "center", as shown in Fig. 1. The geometric ordering constraints can be read from the names: (1) a pixel labeled as "left" cannot be to the *right* of a pixel with any other label; (2) a pixel labeled as "right" cannot be the *left* of a pixel with any other label; (3) a pixel labeled as "top" cannot be *below* a pixel with any other label; (4) a pixel labeled as "bottom" cannot be *above* a pixel with any other label; and (5) if a pixel p labeled as "center" has a neighbor with a different label, then the neighbor pixel has to be labeled as "left", "right", "top", or "bottom" if it is to the left, right, above, or below p, respectively. The last constraint indicates that the "center " region is a rectangle. Note that not all parts have to be present.

To solve this five-parts labeling problem, Liu *et al*. [6] proposed the *ordering preserving moves* for the graph cut optimization, which was demonstrated more effective than the α-expansion method [1]. However, their method does not guarantee to find the globally optimal solution.

Our main technical contribution is a dynamic programming algorithm for computing a globally optimal five-parts labeling of an $N = n \times n$ image in $O(N^{1.5})$ time. This algorithm runs quite fast in practice, taking just seconds to compute an optimal labeling for a rather large images. In addition, our algorithm takes only a linear $O(N)$ memory space.

The key idea for solving the five-parts labeling problem in our algorithm is to guess the possible "center" rectangles, and then for each rectangle, compute an optimal two-labeling for each of those four conner regions incident to the rectangle (Fig. 1(a)) by finding a shortest path. Note that there are $O(N^2)$ possible rectangles, and thus a straightforward algorithm, with an efficient $O(N)$ shortest path algorithm called for each possible rectangle, would take $O(N^3)$ time, which is too slow for practical use. Fortunately, by judiciously characterizing the intrinsic structure of the problem, we are able to improve the running time by a factor of $O(N^{1.5})$.

We evaluate our algorithm on the geometric class scene labeling problem [4], where the goal is to assign each pixel a rough geometric label, such as "sky", "ground", "surface above ground", etc. Experiment shows our algorithm runs faster and more robustly on average than the order-preserving moves method [7]. The standard deviation of the execution times over hundreds of test images with the same size is almost 0 for our method, while it is comparable to the mean execution time for the order-preserving moves method. By intentionally adding Gaussian noise, we observe little effect on the execution time of our algorithm, while a big deterioration is observed for the order-preserving moves method. The average labeling accuracy of the two methods is highly comparable over all the test datasets, though we do find some image datasets, for which our algorithm obtains clearly superior labeling.

Related Work. Felzenszwalb and Veksler recently proposed a tiered scene model which is more general than the five-parts model [3]. In this model, the image is first divided by two horizontal curves into the top, middle and bottom regions, and the middle region is further subdivided vertically into subregions. They give a dynamic programming based algorithm which runs in $O(N^{1.5}K)$ time for a Potts-like model and in $O(N^{1.5}K^2)$ time for more general models, in which N is the size of the image, and K is the number of possible labels in the middle region. However, the space complexity of the algorithm is $O(N^{1.5})$, which could be problematic in practical use for a large image. For example, for an 500×500 image, an $O(N^{1.5})$ memory algorithm will require hundreds of times more memory than an $O(N)$ memory algorithm. In addition, only the 4-connected neighborhood setting case is presented in [3]. Another closely related work is Strekalovskiy and Cremers' multi-labeling framework with generalized ordering constraints based on spatially continuous optimization [8]. This framework includes both the five-parts model and the tiered scene model as special cases, and can deal with even more complex ordering constraints. However, this algorithm does not guarantee global optimality and does not run fast enough in practice. The execution time reported in [8] is 90 seconds for solving the five-parts labeling problem on a 640×480 image using CUDA parallel implementation; while our algorithm takes only about 17 seconds with no parallel implementation.

2. The Model

Given an image $\mathcal{I}$ with a set $\mathcal{P}$ of $N = n \times n$ pixels and a set $\mathcal{L}$ of labels, the pixel labeling problem seeks a labeling f that assigns a label $f_p \in \mathcal{L}$ to each pixel $p \in \mathcal{P}$, such that the energy function of the following form is minimized.

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

In Eq. (1), $\mathcal{N}$ is a neighboring system defined on $\mathcal{P}$. In this paper, we demonstrate our approach using a 4-connected neighborhood. But our approach can be easily extended to 8-connected neighborhood. $D_p(f_p)$ is the data term, which reversely measures the likelihood of assigning label f_p to the pixel p. $V_{pq}(f_p, f_q)$ is the smoothness term, which is the penalty we pay for assigning labels f_p and f_q to neighboring pixels p and q, respectively.

Due to the intractability of the general labeling problem, most of previous work assumes V_{pq} is of a particular form (e.g, convex or metric). Here in the five-parts labeling problem, we make no assumptions on either D_p or V_{pq}. Instead we consider the problem of minimizing $\mathcal{E}(f)$ over a restricted class of labelings. Specifically, D_p and V_{pq} can be arbitrary.

The label set $\mathcal{L}$ includes five labels, $'L'$, $'R'$, $'C'$, $'T'$, $'B'$, which represent "left", "right", "center", "top" and "bottom", respectively. This model enforces the ordering constraints by letting $V_{pq}(f_p, f_q) = \infty$ if f_p and f_q are not allowed. Because we're minimizing the energy, such labeling will not be feasible in a minimized energy. For example, we set $V_{(x,y)(x,y+1)}('B', 'C') = \infty$ to prevent a pixel (x, y) labeled as "bottom" from appearing above a pixel $(x, y+1)$ labeled as "center". If $f_p = f_q$, then $V_{pq}(f_p, f_q) = 0$. Finally, if f_p and f_q satisfy the ordering constraints, $V_{pq}(f_p, f_q) = w_{pq} > 0$.

The complete set of ordering constraints is described in Table.1 and 2. An example labeling satisfying the 5 label ordering constraint model is presented in Fig. 1(a).

3. The Method

This section presents our $O(N^{1.5})$ time algorithm for solving the five-parts labeling problem by dynamic pro-

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

Table 1. Ordering constraints penalty table for vertical neighbors $p(x,y)$ and $q(x,y+1)$ with p being on top of q.

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

Table 2. Ordering constraints penalty table for horizontal neighbors $p(x,y)$ and $q(x+1,y)$ with q being to the right of p.

gramming.

A key observation for our algorithm is as follows. For a fixed center rectangle M, one can extend the four sides of M to divide the image $\mathcal{I}$ into nine regions, denoted by $NW, N, NE, W, M, E, SW, S, SE$, as shown in Figure 1(a). Due to the ordering constraints, the label for each of the regions N, W, M, E, and S is determined, and each of the four corner regions NW, NE, SW, and SE is labeled with at most two different labels, more precisely, regions NW, NE, SW, and SE are labeled with 'L' and 'T', 'T' and 'R', 'L' and 'B', and 'B' and 'R', respectively. Clearly, the energy on each of the regions N, W, M, E, and S is well defined if the center rectangle M is fixed. Thus, the problem is reduced to computing an optimal two-labeling for each of those corner regions. We further observe that, in each of those corner regions, the boundary between the two labeled parts forms a monotone path with respect to both horizontal and vertical directions (Fig. 1(a)). Our main idea is to optimally solve the two-labeling problem for each corner region by computing a shortest monotone path, which takes $O(N)$ time. Note that there are $O(N^2)$ possible center rectangles in total. It thus takes $O(N^3)$ time for solving the five-parts labeling problem. Interestingly, we are able to batch the computation of all $O(N)$ shortest paths in $O(N)$ time. Hence, the running time of our algorithm can be reduced to $O(N^2)$. Furthermore, by judiciously exploring the intrinsic structure of the problem, we can further improve the running time to $O(N^{1.5})$.

Sections 3.1 and 3.2 show that after $O(N)$ time preprocessing, the minimum energy of the five-parts labeling can be computed in $O(1)$ time for each possible center rectangle, and the speedup of the algorithm is presented in Section 3.3.

3.1. Computing energy for non-corner regions

Given a center rectangle M specified by its two diagonal corner points, (x_1, y_1) and (x_2, y_2) with $x_1 \le x_2, y_1 \le y_2$, we show that the energy for each of the regions N, W, M, E and S, can be computed in $O(1)$ time after $O(N)$ time of preprocessing.

The idea is to first pre-compute the integral data cost image [9] $\mathcal{C}^l_{data}(x,y)$ of each label $l \in \mathcal{L}$ for the data term of the energy function, with $\mathcal{C}^l_{data}(x,y) = \sum_{1\le i\le x, 1\le j\le y} D_{p(i,j)}(f_p = l)$. Note that $\mathcal{C}^l_{data}(\cdot,\cdot)$ can be computed in $O(N)$ time. Then we compute the integral row-smoothness cost $\mathcal{C}^{(T,C)}_{row\_sm}(x,y) = \sum_{1\le i\le x} V_{p(i,y),q(i,y-1)}(f_p = C, f_q = T)$ for the label transition from 'T' on Row $y-1$ to 'C' on Row y. Similarly, we can compute $\mathcal{C}^{(C,B)}_{row\_sm}(x,y)$ for the label transition from 'C' to 'B'. In addition, we define the integral column-smoothness cost $\mathcal{C}^{(L,C)}_{col\_sm}(x,y) = \sum_{1\le j\le y} V_{p(x-1,j),q(x,j)}(f_p = L, f_q = C)$ for the label transition from 'L' on Column $x-1$ to 'C' on Column x. Similarly, $\mathcal{C}^{(C,R)}_{col\_sm}(x,y)$ can be computed. Note that all these tables can be computed in $O(N)$ time. Now we can compute the energy for each of the regions N, W, M, E and S in $O(1)$ time, as follows.

$$\mathcal{E}_N = \mathcal{C}^T_{data}(x_2, y_1-1) - \mathcal{C}^T_{data}(x_1-1, y_1-1); \quad (2)$$

$$\mathcal{E}_W = \mathcal{C}^L_{data}(x_1-1, y_2) - \mathcal{C}^L_{data}(x_1-1, y_1-1); \quad (3)$$

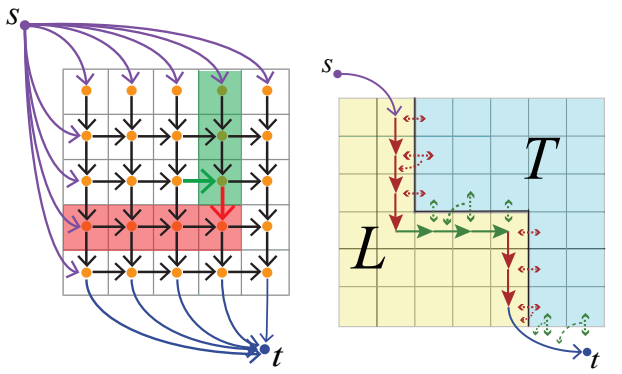
$$\begin{aligned}\mathcal{E}_E = {} & \mathcal{C}^R_{data}(n, y_2) - \mathcal{C}^R_{data}(x_2, y_2) \\ & -\mathcal{C}^R_{data}(n, y_1-1) + \mathcal{C}^R_{data}(x_2, y_1-1); \end{aligned} \quad (4)$$

$$\begin{aligned}\mathcal{E}_S = {} & \mathcal{C}^B_{data}(x_2, n) - \mathcal{C}^B_{data}(x_1-1, n) \\ & -\mathcal{C}^B_{data}(x_2, y_2) + \mathcal{C}^B_{data}(x_1-1, y_2); \end{aligned} \quad (5)$$

$$\begin{aligned}\mathcal{E}_M = {} & \mathcal{C}^C_{data}(x_2, y_2) - \mathcal{C}^C_{data}(x_1-1, y_2) \\ & -\mathcal{C}^C_{data}(x_2, y_1-1) + \mathcal{C}^C_{data}(x_1-1, y_1-1) \\ & +\mathcal{C}^{(T,C)}_{row\_sm}(x_2, y_1) - \mathcal{C}^{(T,C)}_{row\_sm}(x_1-1, y_1) \\ & +\mathcal{C}^{(C,B)}_{row\_sm}(x_2, y_2) - \mathcal{C}^{(C,B)}_{row\_sm}(x_1-1, y_2) \\ & +\mathcal{C}^{(L,C)}_{col\_sm}(x_1, y_2) - \mathcal{C}^{(L,C)}_{col\_sm}(x_1, y_1-1) \\ & +\mathcal{C}^{(C,R)}_{col\_sm}(x_2, y_2) - \mathcal{C}^{(C,R)}_{col\_sm}(x_2, y_1-1) \end{aligned} \quad (6)$$

3.2. Computing min energy for corner regions

For each of the corner regions NW, NE, SW and SE (Fig. 1(a)), we essentially need to solve an optimal 2-labeling problem given a fixed center rectangle M. The idea is to compute a shortest monotone path which completely separates the two parts with different labels. Now we illustrate on the corner region NW that after $O(N)$ preprocessing, given a fixed center rectangle M, each of those 2-labeling problem can be solved in $O(1)$ time.



(a) (b)

Figure 2. (a) Graph construction for solving the 2-labeling problem on Region NW. The horizontal edge (green) incorporates the data term of part of the current column (green block). The vertical edge (red) incorporates the data term of part of the current row (red block). (b) Distribution of the smoothness penalties to the edges. The dotted double-arrows represent the smoothness penalties between two pixels with different labels. The dotted single-arrows shows how the smoothness penalties are assigned to the edges. The smoothness penalties indicated by the red (green) double-arrows are distributed to the red (green) edges.

Assume the lower-right corner of the NW region is (x_0, y_0). We construct the following directed acyclic graph (DAG) $G_{(x_0,y_0)}$ to compute a shortest path minimizing the energy function $\mathcal{E}_{NW}(x_0, y_0)$. The node set consists of two dummy nodes, a *source* s and a *sink* t, and N pixel nodes $v_{(x,y)}$ with each corresponding to exactly one pixel $\mathcal{I}(x, y)$ in the image $\mathcal{I}$.

Now we define directed edges in $G_{(x_0,y_0)}$. Note that the boundary between the L-part (whose pixels are labeled as "left") and the T-part (whose pixels are labeled as "top") of the NW region is monotone to both horizontal and vertical directions. Thus, for every node $v_{(x,y)}$ with $1 \le x \le x_0$ and $1 \le y < y_0$, one vertical edge to $v_{(x,y+1)}$ is introduced. For every node $v_{(x,y)}$ with $1 \le x < x_0$ and $1 < y \le y_0$, one horizontal edge to $v_{(x+1,y)}$ is introduced. We define the *boundary path* in $G_{(x_0,y_0)}$ as a path whose nodes corresponding to the upper envelop of an L-part of the NW region. We further notice that a boundary path could start from any node in the first row or first column, and may end at any node in the last row. Hence, we add one directed edge from s to every node in the first row and the first column, and a directed edge from each node in the last row to the sink t (Fig. 2(a)).

We assign edge costs to encode the energy function in $G_{(x_0,y_0)}$. For notation convenience, denote by $rPreSum(L; x, y) = \sum_{1 \le i \le x} D_{p(i,y)}(f_p = L)$ the total sum of the data cost of the first x pixels of Row y, which are labeled as "left"; and by $cPreSum(T; x, y) = \sum_{1 \le j \le y} D_{p(x,j)}(f_p = T)$ the total sum of the data cost of the first y pixels of Column x, which are labeled as "top".

For any two vertically neighboring pixels $p(x, y)$ and $r(x, y + 1)$, $(1 \le x \le n, 1 \le y < n)$, there is a downward edge from p to r. If both v_p and v_r are on the boundary path (i.e. p and r are labeled as "left"), then pixel $q(x+1, y)$ is labeled as "top". Hence, a smoothness penalty $V_{pq}(f_p = L, f_q = T)$ needs to be enforced. In addition, all pixels from the leftmost pixel of Row $y+1$ to pixel r are labeled as "left". We thus assign a cost $c_e(v_p, v_r)$ to the edge $e(v_p, v_r)$, with

$$c_e(v_p, v_r) = V_{pq}(f_p = L, f_q = T) + rPreSum(L; x, y+1) \tag{7}$$

Specifically, the edge from s to each node in the first row $v_r(x, 1)$ can be treated as a special case of vertical edges, with cost:

$$c_e(s, v_{r(x,1)}) = rPreSum(L; x, 1) \tag{8}$$

For any two horizontal neighbor pixels $p(x, y)$ and $r(x+1, y)$, there is a rightward edge from p to r. If both v_p and v_r are on the boundary path (i.e. p and r are labeled as "left"), then pixel $q(x + 1, y - 1)$ is labeled as "top". Hence, a smoothness penalty $V_{rq}(f_r = L, f_q = T)$ needs to be enforced. In addition, all pixels in Column y starting from the topmost pixel to pixel q are labeled as "top", and pixel r is labeled as "left". Thus the cost of edge $e(v_p, v_r)$ is

$$\begin{aligned} c_e(v_p, v_r) =& V_{qr}(f_q = T, f_r = L) + D_r(f_r = L) \\ &+ cPreSum(T; x + 1, y - 1) \end{aligned} \tag{9}$$

Specifically, the edge from s to each node in the first column $v_{r(1,y)}$ can be treated as a special case of horizontal edges, with cost:

$$\begin{aligned} c_e(s, v_{r(1,y)}) =& V_{q(1,y-1),r}(f_q = T, f_r = L) + D_r(f_r = L) \\ &+ cPreSum(T; 1, y - 1) \end{aligned} \tag{10}$$

Finally, we need to set the costs for the edges connected to the sink t. For a pixel $p(x, y_0)$ in the last row of the NW region, if v_p is the last node on a boundary path, then p is labeled as "left", and each pixel $q(i, y_0)$ right after p in the same row (i.e. $x < i \le x_0$) is labeled as "top". However, the pixel $r(i, y_0 + 1)$ immediately below $q(i, y_0)$ is labeled as "left", as r is in Region W. Thus, we assign a cost $c_e(v_p, t)$ to the edge $e(v_p, t)$ to enforce the smoothness penalty for those label changes. In addition, the data cost for those columns after Column x also need to be enforced. Hence, we have

$$\begin{aligned} c_e(v_p, t) \;=\; & V_{p,q(x+1,y_0)}(f_p = L, f_q = T) \\ &+ \sum_{x < i \le x_0} V_{q(i,y_0),r(i,y_0+1)}(f_q = T, f_r = L) \\ &+ \mathcal{C}^T_{data}(x_0, y_0) - \mathcal{C}^T_{data}(x, y_0). \end{aligned} \tag{11}$$

This completes the construction of $G_{(x_0,y_0)}$ for computing $\mathcal{E}_{NW}(x_0, y_0)$. A shortest s-to-t path can be computed in $O(N)$ time using topological ordering of this DAG, which specifies an optimal 2-labeling for the region NW. However, this is far from good enough to achieve our goal to compute $\mathcal{E}_{NW}(x_0, y_0)$ in $O(1)$ time after an $O(N)$ preprocessing.

Observe that for $x_0 \leq x_0'$ and $y_0 \leq y_0'$, the induced graph of $G_{(x_0,y_0)}$ after removing its sink is a subgraph of the induced graph of $G_{(x_0',y_0')}$ after removing the sink. Thus we can compute all $\mathcal{E}_{NW}(x_0, y_0)$ for $1 \leq x_0 \leq n$ and $1 \leq y_0 \leq n$, as follows. First, construct the graph $G_{(n,n)}$, and compute a *shortest path tree* from the source s in $O(N)$ time. Then, for each node $v_{(x_0,y_0)}$, we introduce the sink $t(x_0, y_0)$ and its incident edges, as we do for the construction of $G_{(x_0,y_0)}$. Thus, it take additional $O(N^{0.5})$ time to find a shortest path from s to $t(x_0, y_0)$ from the computed shortest path tree, rather than from scratch. In this way, it takes $O(N^{1.5})$ time to compute all $\mathcal{E}_{NW}(x_0, y_0)$ for $1 \leq x_0 \leq n$ and $1 \leq y_0 \leq n$.

Interestingly, we can further improve our algorithm. Consider all $\mathcal{E}_{NW}(x, y_0)$ (for $1 \leq x \leq n$) in the same Row y_0. Define the *ending point* of $\mathcal{E}_{NW}(x, y_0)$ as the last node that is on the shortest path from s to the sink $t(x, y_0)$. We have the following lemma.

Lemma 1. *If the ending point of $\mathcal{E}_{NW}(x, y_0)$ is $v_{(x',y_0)}$ ($x' \leq x$), then the ending point of $\mathcal{E}_{NW}(x + 1, y_0)$ is either $v_{(x',y_0)}$ or $v_{(x+1,y_0)}$.*

The proof of the lemma is in the Supplementary Material. Based on Lemma 1, we can compute all $\mathcal{E}_{NW}(x, y_0)$, $(1 \leq x \leq n)$ for Row y_0 in $O(N^{0.5})$ time from the computed shortest path tree. Hence, all $\mathcal{E}_{NW}(x, y)$ for $1 \leq x \leq n$ and $1 \leq y \leq n$ can be computed in $O(N)$time. Similarly, one can compute the table $\mathcal{E}_{NE}(\cdot, \cdot)$, $\mathcal{E}_{SW}(\cdot, \cdot)$ and $\mathcal{E}_{SE}(\cdot, \cdot)$ in $O(N)$ time.

At this point, given a center rectangle M specified by its two diagonal corner points, (x_1, y_1) and (x_2, y_2), we can compute an optimal five-parts labeling with minimized energy $\mathcal{E}_f(x_1, y_1; x_2, y_2)$ in $O(1)$ after an $O(N)$ preprocessing. That is,

$$\begin{aligned}\mathcal{E}_f(x_1, y_1; x_2, y_2) = &\sum_{g\in\{N,W,M,E,S\}} \mathcal{E}_g \\ &+\mathcal{E}_{NW}(x_1 - 1, y_1 - 1) + \mathcal{E}_{NE}(x_2 + 1, y_1 - 1) \\ &+\mathcal{E}_{SW}(x_1 - 1, y_2 + 1) + \mathcal{E}_{SE}(x_2 + 1, y_2 + 1) \end{aligned} \quad (12)$$

Since there are $O(N^2)$ possible center rectangles, we are able to optimally solve the five-parts labeling problem in $O(N^2)$ time. During the preprocessing, we only need to compute $O(1)$ tables each with a size of $O(N)$. Thus, the space complexity is $O(N)$.

3.3. Speedup from $O(N^2)$ to $O(N^{1.5})$

The key idea of the speedup is: given two rows y_1 and y_2, $y_1 \leq y_2$, we return the best possible solution with its upper leftmost corner resides in Row y_1 and its lower rightmost corner resides in Row y_2, in $O(N^{0.5})$ time. In another word, find $\min_{x_1,x_2} \mathcal{E}_f(x_1, y_1, x_2, y_2)$ in $O(N^{0.5})$ time.

Applying Eqn. (12) results in $\mathcal{E}_f(x_1 + 1, y_1, x_2, y_2) - \mathcal{E}_f(x_1, y_1, x_2, y_2) = H(x_1, y_1; y_2)$. Note $H(x_1, y_1; y_2)$ is independent of x_2. This property of $H(\cdot, \cdot; \cdot)$ is crucial to the speedup (proof can be found in supplementary materials). According to definition of $H(\cdot, \cdot; \cdot)$, we have $\mathcal{E}_f(x_1, y_1, x_2, y_2) = \mathcal{E}_f(1, y_1, x_2, y_2) + \sum_{i=1}^{x_1-1} H(i, y_1; y_2)$. As a result,

$$\arg_{x_2} \min_{x_2 \geq x_1} \mathcal{E}_f(x_1, y_1, x_2, y_2) = \arg_{x_2} \min_{x_2 \geq x_1} \mathcal{E}_f(1, y_1, x_2, y_2) \quad (13)$$

for fixed y_1, y_2. In another word, we only need to compute $\min \mathcal{E}_f(1, y_1, \cdot, y_2)$ for $x_1 = 1$, and it could be used to compute $\min_{x_2 \geq x_1} \mathcal{E}_f(x_1, y_1, \cdot, y_2)$ for $x_1 \neq 1$. Define the following running min and running sum:

$$rMin_{(y_1,y_2)}(x) = \min_{i \geq x} \mathcal{E}_f(1, y_1, i, y_2) \quad (14)$$

$$hMin_{(y_1,y_2)}(x) = \sum_{i=1}^{x-1} H(i, y_1; y_2) \quad (15)$$

For fixed y_1 and y_2, $rMin_{(y_1,y_2)}(\cdot)$ and $hMin_{(y_1,y_2)}(\cdot)$ can be computed within $O(N^{0.5})$ time. Let x_2^* be the optimal x_2 that achieves optimal energy $\mathcal{E}_f(x_1, y_1, x_2^*, y_2)$ for fixed x_1, y_1, y_2, then

$$\mathcal{E}_f(x_1, y_1, x_2^*, y_2) = rMin_{(y_1,y_2)}(x_1) + hMin_{(y_1,y_2)}(x_1) \quad (16)$$

Note for fixed x_1, y_1, y_2 this only takes constant time, given that $rMin_{(y_1,y_2)}(\cdot)$ and $hMin_{(y_1,y_2)}(\cdot)$ have been computed.

This accomplish our goal of finding optimal solution for fixed Row y_1 and y_2 in $O(N^{0.5})$ time. Directly repeating this process for all $1 \leq y_1 \leq y_2 \leq n$ results in a $O(N^{1.5})$ algorithm. Note $rMin_{(y_1,y_2)}$ and $hMin_{(y_1,y_2)}$ does not need to be remembered for different y_1, y_2, so memory consumption for them is just $O(N^{0.5})$.

Theorem 1. *Given an image of $N = n \times n$ pixels, the five-parts labeling problem can be solved in $O(N^{1.5})$ time and $O(N)$ space.*

4. Experiment–Geometric Class Labeling

We used 300 indoor images and 42 outdoor images, which are the same as the test images used in [6]. All indoor images are 640*480. But outdoor images have various sizes.

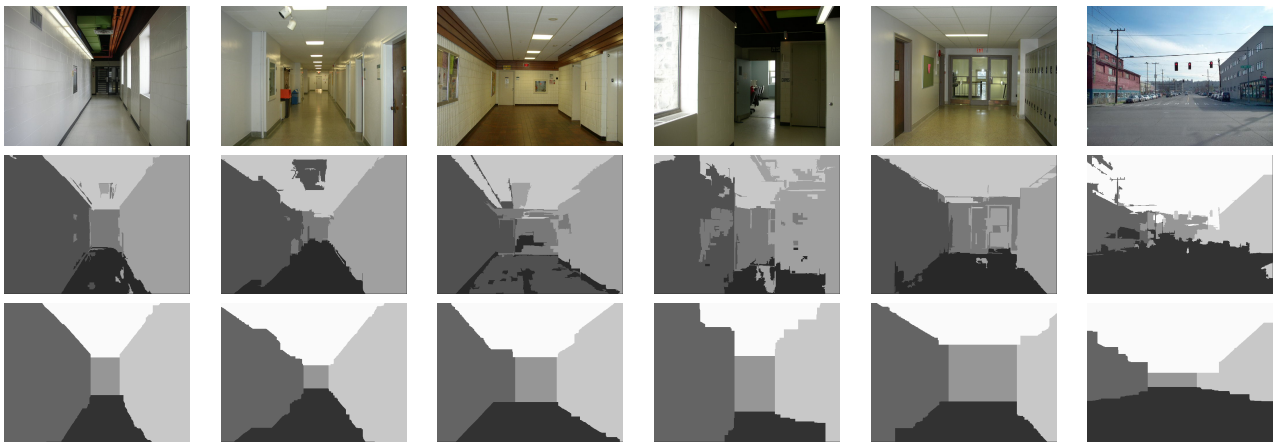


Figure 3. Some labeling results. Top row: original images; second row: SVM classifier results using data term only; last row: our results

Table 3. Average accuracy rate (%)

Image sets	OPM	Our alg.
Indoor images	84.9 ± 14.9	85.1 ± 14.5
Outdoor images	85.7 ± 7.0	85.7 ± 6.9

OPM: the order-preserving moves method.

4.1. Cost images

We used the same data term costs as in [6]. First, the images are partitioned into "superpixels", which are homogeneous regions within each region and heterogeneous between different regions, using the algorithm by Felzenszwalb *et al*. [2]. Similar to Hoeim *et al*.'s method [4], an SVM classifier is then trained with a wide variety of selected features, such like location, color, texture, geometry, and edges. Finally, a probability for each "superpixel" to be assigned a label $l \in \mathcal{L} = \{L, R, T, B, C\}$ is computed. All pixels within this "superpixel" are assigned a cost according to the probability of the "superpixel" it belongs to. This completes the data term generation.

The smoothness term is generated simply using Sobel operator along the horizontal and vertical directions.

4.2. Results

Example results are shown in Fig. 3.

Define the accuracy rate as the ratio of the number of correctly labeled pixels over the total number of pixels. The performance on the accuracy of our algorithm and the order-preserving moves method is shown in Table.3.

Our algorithm does not show significant improvement in the accuracy rate. The difference of minimized energy is 0.10% for indoor images and 0.16% for outdoor images on average for both methods, although our method always obtained an energy no worse than the order-preserving moves method. The marginal difference indicates that the order-preserving moves works pretty well in practice.

Although for most of the test cases, there are little difference between the order-preserving moves and our algorithm, we do observe significant difference on some cases, as shown in Fig.4. Our algorithm captures the door in the image in the first column (last row), and the rectangular space between the door and the box in the second column. The cost image for the image in the third column is poor, from which it is very difficult to distinguish the "left" region from the "center" region. However, our algorithm still can produce a reasonable labeling, while the order-preserving moves is trapped into a local minima with a long execution time of 244.18 seconds.

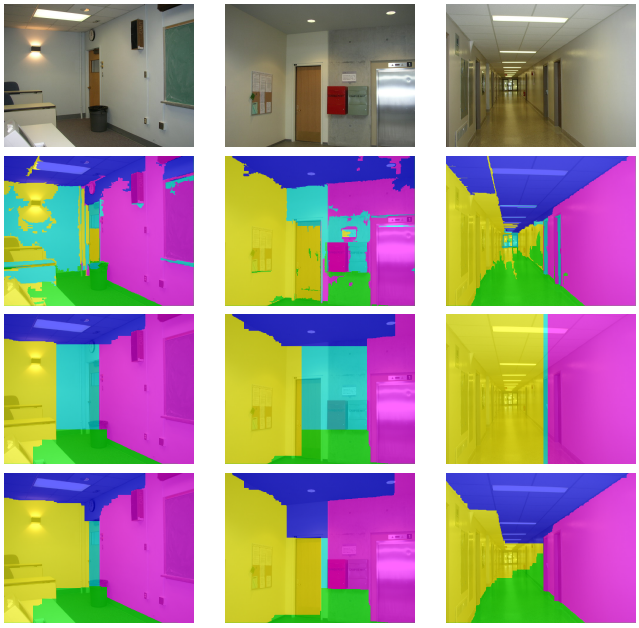


Figure 4. Example images on which our algorithm output quite different labeling results from the order-preserving moves. First row: original image; second row: SVM classifier results, *i.e*. results using only data term; third row: results by OPM; fourth row: results by our algorithm.

Table 4. Average execution time (s)

Image sets	OPM	Our alg.
Indoor images	26.6±22.6	17.1±0.1
27 outdoor images with size of 640×480	20.6±12.5	16.4± 0.3
Overall outdoor images*	20.2	14.1

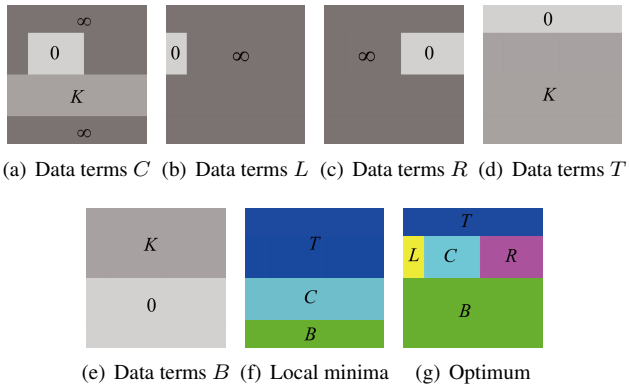
*: The overall outdoor image datasets include varying image sizes. Thus, no standard deviations are reported.

Table 5. Average execution time comparison (s)

Methods	No noise	$\sigma = 0.17$	$\sigma = 0.29$	$\sigma = 0.58$
OPM	20.2	43.8	40.4	81.4
Our alg.	14.1	14.7	14.5	15.0

Table 6. Max execution time comparison (s)

Methods	No noise	$\sigma = 0.17$	$\sigma = 0.29$	$\sigma = 0.58$
OPM	136.4	396.4	267.7	1408.8
Our alg.	33.8	35.2	35.2	41.1



(a) Data terms C (b) Data terms L (c) Data terms R (d) Data terms T

(e) Data terms B (f) Local minima (g) Optimum

Figure 5. Illustrating the lack of global optimality for the order-preserving moves method. All smoothness penalties are 0 in this example. The energy of the global optimum is 0. While the energy of the local minima is a multiple of K. Note K can be arbitrarily large, which will make the local minima arbitrarily far away from the optimum.

Table.4 shows the average execution time of the order-preserving moves and our algorithm. Our algorithm outperforms the order-preserving moves significantly while guaranteeing the global optimality. Note that our execution time is much better than that (90s) reported in [8] by Strekalovskiy and Cremers despite their use of CUDA for parallel implementation.

The execution time reported in [3], for tiered scene labeling, is 9.4 seconds on images approximately 300×250; while the execution time of our algorithm is 2.2 seconds on 320×240 images. Note the $O(N^{1.5})$ memory consumption of [3] might make it problematic to process large images, which is overcome by our algorithm. In addition, our algorithm can be easily parallelized for GPU, which may bring more significant speedups. This will be discussed in Sec. 5

Moreover, the running time of our method only depends on the image size. The standard deviation of the execution times over hundreds of test images with the same size is almost 0 for our method, while it is comparable to the mean execution time for the order-preserving moves method, as in Table. 4. By intentionally adding Gaussian noise to cost images, we observe little effect on the execution time of our algorithm, while a big deterioration is observed for the order-preserving moves method, as shown in Table 5 and Table 6. The mean value of the Gaussian noise is 0 and σ is normalized with respect to the maximum intensity value of the cost image.

5. Discussion

5.1. Global optimality

Global optimization is important for the labeling problems. Although the order-preserving moves method works well for test image datasets we used, it may get trapped in a local minima very far away from the global optimal solution, and fail to find an acceptable solution.

Consider the given costs for each label shown in Fig. 5, and all smoothness penalties are set to be 0. Start from an initial labeling with all pixels labeled as 'C' [7]. A horizontal move results in a labeling of an energy of ∞ , since a vertical strip across the whole image must be labeled as 'C' in this horizontal move. Hence, only a vertical move is possible to return a finite energy by labeling the horizontal strip in which all pixels have a cost of K for label 'C'. Unfortunately, the order-preserving moves method gets stuck here. Any further order-preserving move will results in ∞ energy.

However, the energy of the global optimal solution is 0. Note the value of K could be arbitrarily large, which indicates that even this strong order-preserving moves method gets trapped in a local minima arbitrarily far from the optimal solution.

5.2. Parallelization of the algorithm

The most time-consuming part of this algorithm is the optimal center rectangle searching process. In a typical running on a 640×480 image, this process takes about 16 seconds, while the average total execution time for such an images is just about 17 seconds.

However, this process is highly parallelizable. We can view each row pair of y_1 and y_2 as a unit for parallelization. As indicated in Sec. 5.1, the computations between different y_1, y_2 row pairs are totally independent of each other. This makes our algorithm straightforward to parallelize on multi-core CPUs and high-end GPUs. There are $O(N)$ different y_1, y_2 pairs in total. This number of threads should be able to saturate the current high-end CPUs, which only

have hundreds of cores.

5.3. Extension to 8-connectivity

Our method can be easily extended to the 8-neighborhood setting. Note that the algorithm in [3] is at least nontrivial to do the extension.

It introduces additional smoothness terms using the 8-connectivity. The smoothness penalty between the center region and the other four non-corner regions is easy to handle. We next show how to handle the smoothness penalty for the corner regions.

We again illustrate our idea on the NW region. In Fig. 6, the red and green dotted double-arrows show the smoothness penalties on the diagonally neighboring pixels we need to enforce into our shortest path model. We basically want to distribute those penalties to the edges on the boundary path. Note that the smoothness penalties indicated by the green double-arrows correspond one-by-one to the edges on the boundary path as follows. For the downward edge from $v_{(x,y)}$ to $v_{(x,y+1)}$, we add an additional cost of $V_{p(x,y+1),q(x+1,y)}(f_p = L, f_q = T)$ to the edge. While for the rightward edge from $v_{(x,y)}$ to $v_{(x+1,y)}$, an additional cost of $V_{p(x+1,y),q(x+2,y-1)}(f_p = L, f_q = T)$ is added to the edge.

Similarly, smoothness penalties indicated by the red double-arrows can be accounted by adding following costs: for the downward edge from $v_{(x,y)}$ to $v_{(x,y+1)}$, add $V_{p(x,y),q(x+1,y+1)}(f_p = L, f_q = T)$, and for the rightward edge from $v_{(x,y)}$ to $v_{(x+1,y)}$, add $V_{p(x+1,y),q(x,y-1)}(f_p = L, f_q = T)$. The only problem is that the sum of the two brown edges incident at $(\bar{x}, \bar{y})$ in Fig. 6 overestimates by an amount of $V_{p(\bar{x},\bar{y}-1),q(\bar{x}+1,\bar{y})}(f_p = L, f_q = T) + V_{p(\bar{x}+1,\bar{y}),q(\bar{x},\bar{y}-1)}(f_p = L, f_q = T)$. To solve this problem, we introduce a diagonal edge $e(v_{(\bar{x},\bar{y}-1)}, v_{(\bar{x}+1,\bar{y})})$, whose cost equals $c_e(v_{(\bar{x},\bar{y}-1)}, v_{(\bar{x},\bar{y})}) + c_e(v_{(\bar{x},\bar{y})}, v_{(\bar{x}+1,\bar{y})}) - V_{p(\bar{x},\bar{y}-1),q(\bar{x}+1,\bar{y})}(f_p = L, f_q = T) - V_{p(\bar{x}+1,\bar{y}),q(\bar{x},\bar{y}-1)}(f_p = L, f_q = T)$. If we assume all smoothness penalties are nonnegative, then this edge is always preferable than the "detour" of the two brown edges.

6. Conclusion

In this paper, we present an algorithm optimally solving the five-parts labeling problem, which, to the best of our knowledge, is the first algorithm that guarantees globally optimal solution to that labeling problem with linear space complexity. The theoretical running time is $O(N^{1.5})$, with N being the number of pixels in the image. In practice, it runs much faster than the method reported in [7]. Moreover, it can easily be parallelized for GPU, and it is extensible to the 8-neighborhood setting without affecting the theoretical running time.

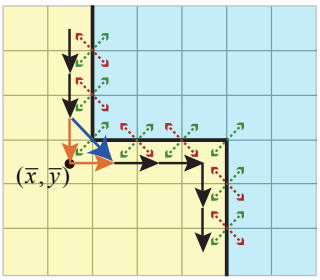


Figure 6. The green and red double-arrows indicate the additional smoothness penalties we need to enforce when extended to the 8-connectivity.

Acknowledgments

This work was supported in part by NSF grants CCF-0830402 and CCF-0844765; and the NIH grant K25-CA123112, as well as NSERC and ERA grants.

References

[1] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE Transactions on pattern analysis and machine intelligence*, pages 1222–1239, 2001. 1

[2] P. Felzenszwalb and D. Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59(2):167181, 2004. 6

[3] P. Felzenszwalb and O. Veksler. Tiered scene labeling with dynamic programming. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 3097–3104. IEEE. 1, 2, 7, 8

[4] D. Hoiem, A. Efros, and M. Hebert. Geometric context from a single image. In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, volume 1, page 654661, 2005. 2, 6

[5] H. Ishikawa. Exact optimization for markov random fields with convex priors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1333–1336, 2003. 1

[6] X. Liu, O. Veksler, and J. Samarabandu. Graph cut with ordering constraints on labels and its applications. In *IEEE Conference on Computer Vision and Pattern Recognition, 2008. CVPR 2008*, pages 1–8. IEEE, June 2008. 1, 5, 6

[7] X. Liu, O. Veksler, and J. Samarabandu. Order-Preserving moves for Graph-Cut-Based optimization. *IEEE transactions on pattern analysis and machine intelligence*, page 11821196, 2010. 1, 2, 7, 8

[8] E. Strekalovskiy and D. Cremers. Generalized ordering constraints for multilabel optimization. In *IEEE International Conference on Computer Vision (ICCV)*. 2011. 1, 2, 7

[9] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001.*, volume 1, pages I–511. IEEE, 2001. 3

[10] X. Wu and D. Chen. Optimal net surface problems with applications. *Automata, Languages and Programming*, pages 775–775, 2002. 1