Tiered Scene Labeling with Dynamic Programming

Pedro F. Felzenszwalb
University of Chicago
pff@cs.uchicago.edu

Olga Veksler
University of Western Ontario
olga@csd.uwo.ca

Abstract

Dynamic programming (DP) has been a useful tool for a variety of computer vision problems. However its application is usually limited to problems with a one dimensional or low treewidth structure, whereas most domains in vision are at least 2D. In this paper we show how to apply DP for pixel labeling of 2D scenes with simple "tiered" structure. While there are many variations possible, for the applications we consider the following tiered structure is appropriate. An image is first divided by horizontal curves into the top, middle, and bottom regions, and the middle region is further subdivided vertically into subregions. Under these constraints a globally optimal labeling can be found using an efficient dynamic programming algorithm. We apply this algorithm to two very different tasks. The first is the problem of geometric class labeling where the goal is to assign each pixel a label such as "sky", "ground", and "surface above ground". The second task involves incorporating simple shape priors for segmentation of an image into the "foreground" and "background" regions.

1. Introduction

In a pixel labeling problem the goal is to assign a label to each pixel in an image. Pixel labeling is a very general framework that can be used to solve a variety of problems in computer vision [14]. Pixel labeling problems in 2D are usually NP-hard [4], but can be solved efficiently in some special cases [8, 10, 18, 12]. If the domain is 1D or of low treewidth, then optimal labelings can be found using dynamic programming (DP). A simple example involves labeling each scanline of an image independently [17].

In this paper, we consider a new class of pixel labeling problems in 2D domain that can be solved efficiently and exactly with dynamic programming.

We consider labelings that have simple *tiered* structure. We focus on the structure shown in Figure 1. In this case, an image is partitioned into a top, bottom and middle part. Pixels in the top receive the "top" label, pixels in the bottom receive the "bottom" label, and pixels in each column of the middle region receive a label from a finite set.

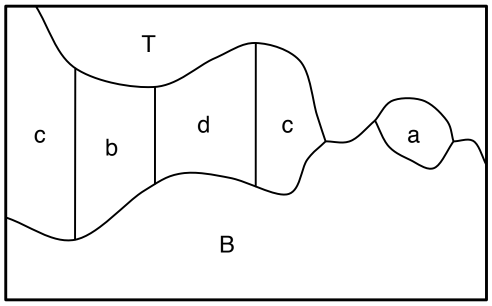


Figure 1. A *tiered* labeling. The labeling is defined by two horizontal curves α and β that partition the image into a top, bottom and middle region. The middle region is subpartitioned by vertical boundaries. The top region is labeled T, the bottom region is labeled B and the middle regions take labels from M.

Our main technical contribution is a DP algorithm for computing a globally optimal tiered labeling of an $n \times m$ image in $O(mn^2K^2)$ time. Here K is the number of possible labels for pixels in the middle region. This algorithm is quite fast in practice, taking just a few seconds to find an optimal labeling of fairly large images.

Our method works by reducing the 2D labeling problem to a 1D optimization problem, equivalent to finding a shortest path in a very large state space. Standard DP methods for this 1D problem would take $O(mn^4K^2)$ time, which is too slow for practical use. We exploit special structure from the 2D problem to obtain a much faster solution.

Global optimization for 2D labeling is quite rare, but its importance cannot be understated. Even the relatively strong, widely used expansion algorithm [4] gets trapped in a local minima very far from the global one, and fails to find an acceptable solution in our segmentation application.

Our work is the only method we know of that uses DP for optimization of a 2D labeling, except for the simple case in [1]. The problem in [1] involves binary segmentation where the foreground object is constrained to be a connected region bounded by two horizontal curves. The objective function in [1] simply penalizes regions with high variance and does not have any smoothness terms.

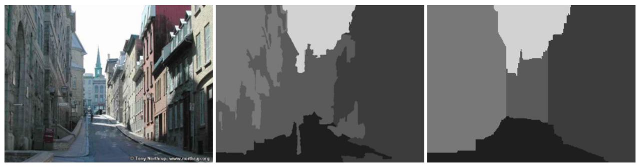


Figure 2. Geometric labeling application. Each intensity represents a different geometric class.

We evaluate our algorithm on two very different applications. One is the geometric class labeling problem [9], where the goal is to assign each pixel a label such as “sky”, “ground” and “surface above ground”. In this case the tiered constraints arise naturally. The sky and ground correspond to the top and bottom region, and different labels model different kinds of surfaces above ground in the middle region.

We use [9] as a baseline, and show how the geometric constraints imposed by tiered labelings improve the accuracy of solutions. An example is shown in Figure 2.

Our second application is binary segmentation, where the goal is to classify each pixel into “foreground” or “background”. Here the usefulness of a tiered structure is not immediately obvious. We use the tiered structure to incorporate a novel type of shape prior for the foreground. Thus, in addition to showing the application of our labeling algorithm for segmentation, we develop a novel approach for shape priors in segmentation, which is of interest in itself.

The idea behind our shape prior is as follows. A shape is partitioned vertically into several parts, each part corresponds to a “middle” label. The top and bottom regions in a labeling correspond to the background. We encourage (or enforce using hard constraints) the boundary between the background labels and the foreground parts to follow certain paths, for example a diagonal slanted to the right, etc.

We demonstrate several shape priors that are tuned for generic classes of objects. An example is in Figure 3. The input image (top row) has several objects with identical intensity histograms. There are four shape priors: a plane, leaf, bow, and cross. Depending on which shape prior is used, we get the results in the bottom four rows. To help visualize the constraints between different parts, we display them with different intensity. We illustrate results in both unsupervised and interactive settings and show that using shape priors helps to obtain more accurate segmentations.

Related Work Optimization algorithms for pixel labeling problems is a very active area of research [19]. The majority of the work is concerned with approximate solutions for NP-hard energies, which is not our case. The closest method, and also the one that motivated our work is [15].

The method in [15] is a graph-cut based algorithm for a five-label energy function with certain restrictions on geometric relationships between labels. The restrictions lead to

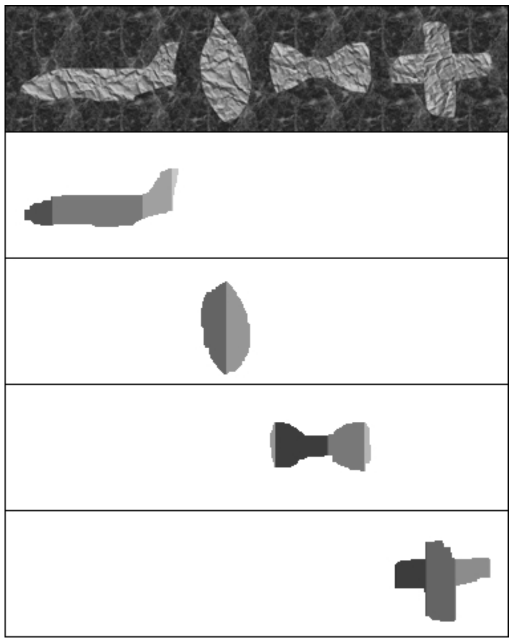


Figure 3. Segmentation with shape prior. Top row: the input image. Next rows show segmentation results with a “plane”, “leaf”, “bow, and “cross” shape priors, respectively.

a special case of tiered labelings where there are only three possible labels for the middle region. Moreover, in [15] the middle region is constrained to be partitioned into 3 sequential subregions with the second one in the shape of a rectangle. Furthermore, since [15] uses graph-cuts, they require submodular [12] V_{pq} (see Section 2). While the problem is polynomially solvable, the algorithm in [15] finds only an approximate solution. The authors note that a global minimum could be found using $O(n^2)$ graph-cuts (n is the height of the image), but this would be impractical.

For a 4-connected neighborhood our model is a strict generalization of [15]. We find an optimum solution to a more general problem with a much smaller complexity. The only advantage of [15] is that they can handle more general neighborhoods, but these have rarely been used in practice.

One of our applications is the geometric class labeling problem popularized by [9]. Since the original paper several advances were made, most of them addressing how to better handle the problem in an energy optimization framework [11, 7, 15]. Except for [15] these methods can offer only approximate optimization, since they rely on NP-hard energies. In contrast, we compute a globally optimal solution within the class of tiered labelings. The disadvantage, is, of course, limitation to tiered scenes, which are nevertheless much more general than those considered in [15].

Our other application is segmentation with a shape prior, which is a natural approach for getting more robust segmentation results [13, 5]. Segmentation with shape priors usually leads to NP-hard problems. In rare cases [6, 21, 7] when a global minimum can be computed, the shape prior is very restrictive. We model a shape by partitioning it into several regions and imposing (soft) constraints on the boundaries between different regions and the background. Most closely related is the work of [22], who also decompose an object into parts and model geometric relationships between them. However, our models are more generic, while [22] is based on a deformable model framework. Another related idea is in [15] who model a shape prior by geometric relationships between a single object label and four background labels. Having only a single label for the object severely restricts the possible shape priors, in fact, they experiment only with a rectangle and a trapezoid prior.

2. The Model

In a pixel labeling problem we have an image I with pixels P and a set of labels L. A labeling f assigns a label $f_p \in L$ to each pixel $p \in P$. The goal is to find a labeling minimizing an energy function of the form,

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

Here $D_p(f_p)$ and $V_{pq}(f_p, f_q)$ are the data and smoothness terms, and N is a neighborhood system on P. We assume N is the 4-connected neighborhood with ordered pairs (p, q) such that p is directly to the left or above q.

The data term $D_p(a)$ specifies a cost for assigning label a to pixel p. This usually depends on the value of I at p. The smoothness term $V_{pq}(a, b)$ specifies a cost for assigning labels a and b to pixels p and q. This can be independent of p and q but often depends on the value of I at p and q.

Most work on pixel labeling assumes V_{pq} is of a particular form (e.g. metric or semi-metric [4]). Here we make no assumptions on D_p or V_{pq}. Instead we consider the problem of minimizing E over a restricted class of labelings. In particular we can handle arbitrary V_{pq}, including asymmetric, $V_{pq}(a, b) \neq V_{pq}(b, a)$, and repulsive, $V_{pq}(a, a) > V_{pq}(a, b)$.

Let L be a set of labels with two distinguished labels T and B. Let $M = L \backslash \{T, B\}$. In a *tiered labeling* of I each column is partitioned into 3 regions corresponding to the top, bottom and middle parts. Here we consider labelings where pixels in the top are labeled T, pixels in the bottom are labeled B, while the pixels in the middle are labeled ℓ for some $\ell \in M$. Figure 1 illustrates an example.

More formally we define tiered labelings as follows. Let I be an image with n rows and m columns. We index rows and columns starting at zero. A tiered labeling is defined by a sequence of m triples (i_k, j_k, ℓ_k), one per column of I. For each column k we have a triple (i, j, ℓ) with $0 \leq i \leq j \leq n$ and $\ell \in M$. This triple defines labels for all pixels in column k. Pixels in rows $0, \ldots, i-1$ are labeled T, pixels in rows $j, \ldots, n-1$ are labeled B and pixels in rows $i, \ldots, j-1$ are labeled ℓ. See Figure 4.

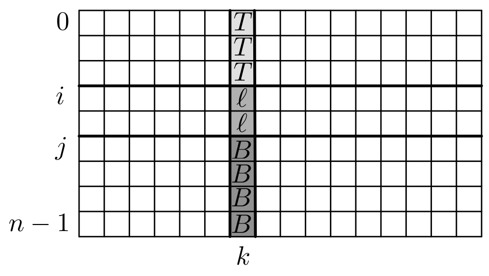


Figure 4. The labeling in each column is defined by 2 indices $0 \leq i \leq j \leq n$ and a label ℓ for the middle region.

Intuitively the sequence of indices $\alpha = (i_0, \ldots, i_{m-1})$ defines a boundary between the top and middle region. Similarly $\beta = (j_0, \ldots, j_{m-1})$ defines a boundary between the middle and bottom region. The middle region is subpartitioned as each column can take a different label.

We can think of α and β as "horizontal curves" in the sense that they go between the left and right boundaries of I, going through each column exactly once. The two curves do not cross because $i_k \leq j_k$. Conversely, tiered labelings can be defined by two non-crossing horizontal curves plus labels for the middle region in each column.

3. Dynamic Programming

Let $Z = \{(i, j) \,|\, 0 \leq i \leq j \leq n\}$.
Let $S = Z \times M$.
A tiered labeling is defined by a sequence of m triples

$$s_k = (i_k, j_k, \ell_k) \in S, \quad 0 \leq k \leq m-1. \qquad (2)$$

We can see a labeling as a path in $S \times \{0, \ldots, m-1\}$, going through points (s_k, k). This path simultaneously traces the two boundaries α and β in the image, moving from one column to the next in each step.

When f is a tiered labeling, the energy function in equation (1) can be expressed as

$$E(f) = \sum_{k=0}^{m-1} U_k(s_k) + \sum_{k=0}^{m-2} H_k(s_k, s_{k+1}). \qquad (3)$$

This form is obtained by putting all data terms and vertical smoothness terms within column k into U_k, while the horizontal smoothness terms between columns k and $k+1$ go into H_k. We note that $U_k(s)$ can be computed in $O(1)$ time if we precompute cumulative sums of data terms and vertical smoothness terms up to each row in each column of the image. Similarly $H_k(s, t)$ can be computed in $O(1)$ time using cumulative sums of the horizontal smoothness terms leaving each column. We describe this approach for computing H_k in equation (13).

The problem of finding a sequence of states minimizing (3) can be solved via standard dynamic programming techniques. It is equivalent to computing a MAP estimate for an HMM, which can be solved using the Viterbi algorithm. However, a standard implementation of the Viterbi algorithm is too slow in our setting because it takes $O(m|S|^2)$ time. Since for us $|S| = O(n^2K)$ the standard algorithm runs in $O(mn^4K^2)$ time. In a square image with n^2 pixels this is more than quadratic in the number of pixels.

Our algorithm exploits the special form of the pairwise terms H_k in our problem. This makes it possible to speed up the standard dynamic programming method to run in $O(mn^2K^2)$ time. Note that in the case of a square image with n^2 pixels our algorithm runs in $O(n^3K^2)$ time.

Before introducing our algorithm, we review the standard DP for equation (3). The method builds m tables, E_k, indexed by states $s \in S$. The value in $E_k[s]$ is the cost of an optimal sequence of $k+1$ states ending in state s.

Since we are essentially solving a shortest path problem, if $(s_0, \ldots, s_k)$ is an optimal sequence ending in s_k then $(s_0, \ldots, s_{k-1})$ is an optimal sequence ending in s_{k-1}. This allows us to compute E_k in order of increasing k using a simple recurrence,

$$E_0[s] = U_0(s), \tag{4}$$

$$E_k[s] = U_k(s) + \min_{\bar{s}}(E_{k-1}[\bar{s}] + H_{k-1}(\bar{s}, s)). \tag{5}$$

When computing $E_k[s]$ we also record the optimum previous state in a table $P_k[s]$. After all tables are computed a solution is obtained by selecting $s_{m-1} = \text{argmax}_s\, E_{m-1}[s]$ and tracing the optimal path using P_k.

Note that when computing E_k, for each state $s \in S$ we need to search over all possible previous states $\bar{s} \in S$. This means it takes $O(|S|^2)$ time to compute each table. Since we need to compute m tables the runtime of the algorithm is $O(m|S|^2) = O(mn^4K^2)$.

3.1. Fast Algorithm

We can speed up the DP algorithm to run in $O(mn^2K^2)$ time in our setting. The method works by speeding up the computation of each table E_k from $O(n^4K^2)$ to $O(n^2K^2)$.

Let g be an array of size n. The running-min of g is an array h, also of size n, defined by

$$h[i] = \min_{i' \leq i} g[i']. \tag{6}$$

There is a simple algorithm for computing h in $O(n)$ time. First we set $h[0] = g[0]$. We then sequentially set $h[i] = \min(g[i], h[i-1])$. We use this algorithm below.

Consider the standard DP algorithm specified in the last section. We will speed it up by considering the basic computation that propagates information from one stage to the next. In each stage the basic computation is of the form

$$F[s] = \min_{\bar{s} \in S}(E[\bar{s}] + H(\bar{s}, s)). \tag{7}$$

Here E specifies the cost of solutions up to the previous column, and F can be used to compute solutions up to the current column. Both E and F have $|S| = O(n^2K)$ entries.

For each $s \in S$ we need to search over $\bar{s} \in S$. If these searches are done independently it takes $O(n^4K^2)$ time to compute F. In our algorithm we break the search for $\bar{s}$ into different sub-cases so that we can do all searches much faster. First, for each state $s = (i, j, \ell)$ we will separately minimize over choices for $\bar{\ell}$. Suppose ℓ and $\bar{\ell}$ are fixed. In this case we need to compute quantities of the form,

$$F_{\ell,\bar{\ell}}[i, j] = \min_{(\bar{i},\bar{j}) \in Z}(E_{\ell,\bar{\ell}}[\bar{i}, \bar{j}] + H_{\ell,\bar{\ell}}((\bar{i}, \bar{j}), (i, j))). \tag{8}$$

The subscripts on E, F and H indicate we are working on subproblems defined by fixed ℓ and $\bar{\ell}$. After computing $F_{\ell,\bar{\ell}}$ for each choice of ℓ and $\bar{\ell}$ we can simply pick the best $\bar{\ell}$ for each state (i, j, ℓ).

Our algorithm works by breaking the minimization in equation (8) into 6 cases. Let $Z_1(i, j), \ldots, Z_6(i, j)$ be subsets of Z such that $Z = Z_1(i, j) \cup \cdots \cup Z_6(i, j)$. Then we can compute $F_{\ell,\bar{\ell}}$ as follows

$$F_t[i, j] = \min_{(\bar{i},\bar{j}) \in Z_t(i,j)}(E_{\ell,\bar{\ell}}[\bar{i}, \bar{j}] + H_{\ell,\bar{\ell}}((\bar{i}, \bar{j}), (i, j))), \tag{9}$$

$$F_{\ell,\bar{\ell}}[i, j] = \min_{1 \leq t \leq 6} F_t[i, j]. \tag{10}$$

The sets $Z_t(i, j)$ are defined in terms of the positions of $\bar{i}$ and $\bar{j}$ relative to i and j. Note that $\bar{i}$ and $\bar{j}$ can each be less than or equal to i, between i and j, or greater than or equal to j. There are 3 possible choices for each, but since $\bar{i} \leq \bar{j}$ there are only 6 choices total. This leads to a partition of Z into 6 subsets. For example, we have one subset for $i \leq j \leq \bar{i} \leq \bar{j}$ and another for $i \leq \bar{i} \leq \bar{j} \leq j$.

The speedup For each t, computing F_t via brute force search takes $O(n^4)$ time, which is no better than computing $F_{\ell,\bar{\ell}}$ directly. Now we describe an algorithm for computing F_t in $O(n^2)$ time for the case

$$Z_t(i, j) = \{(\bar{i}, \bar{j}) \mid \bar{i} \leq i \leq \bar{j} \leq j\}. \tag{11}$$

The other cases are analogous.

Our method relies on two key ideas. First, we note that we can decouple the search for $\bar{i}$ from the search for $\bar{j}$. Second, we note that the resulting searches can be done quickly using running-min computations.

Let $I(a, b)[x]$ be the sum of the horizontal smoothness terms for labels a and b between the previous and current columns in rows 0 through $x-1$. That is, $I(a, b)$ is the sum of $V_{pq}(a, b)$ for p in the previous column, q in the current column, and both p and q in rows 0 through $x-1$. We can express the smoothness terms in $H((\bar{i}, \bar{j}), (i, j))$ using these

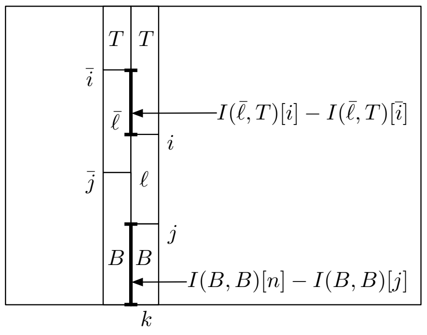


Figure 5. Computing $H((\bar{i},\bar{j}),(i,j))$ when $\bar{i} \le i \le \bar{j} \le j$. The horizontal V_{pq} can be defined by integral arrays.

cumulative sums, see Figure 5.

$$H((\bar{i},\bar{j}),(i,j)) = d1 + d2 + d3 + d4 + d5 \tag{12}$$

$$\text{where}\quad d1 = I(T,T)[\bar{i}] \tag{13}$$

$$d2 = I(\bar{\ell},T)[i] - I(\bar{\ell},T)[\bar{i}] \tag{14}$$

$$d3 = I(\bar{\ell},\ell)[\bar{j}] - I(\bar{\ell},\ell)[i] \tag{15}$$

$$d4 = I(B,\ell)[j] - I(B,\ell)[\bar{j}] \tag{16}$$

$$d5 = I(B,B)[n] - I(B,B)[j] \tag{17}$$

Note that we can re-group the terms so that H is a sum of functions of a single index plus a constant.

$$H((\bar{i},\bar{j}),(i,j)) = \mathcal{I}(i) + \mathcal{J}(j) + \bar{\mathcal{I}}(\bar{i}) + \bar{\mathcal{J}}(\bar{j}) + \mathcal{C}, \tag{18}$$

Now we have $F_t[i,j]$

$$= \min_{\bar{i}\le i\le \bar{j}\le j} (E[\bar{i},\bar{j}] + \mathcal{I}(i) + \mathcal{J}(j) + \bar{\mathcal{I}}(\bar{i}) + \bar{\mathcal{J}}(\bar{j}) + \mathcal{C})$$
$$= \mathcal{I}(i) + \mathcal{J}(j) + \mathcal{C} + \min_{i\le\bar{j}\le j}(\bar{\mathcal{J}}(\bar{j}) + \min_{\bar{i}\le i}(E[\bar{i},\bar{j}] + \bar{\mathcal{I}}(\bar{i})))$$

Since neither $E[\bar{i},\bar{j}]$, or $\bar{\mathcal{I}}(\bar{i})$ depend on j we can solve for the optimum $\bar{i}$ as a function of i and $\bar{j}$,

$$E'[i,\bar{j}] = \min_{\bar{i}\le i}(E[\bar{i},\bar{j}] + \bar{\mathcal{I}}(\bar{i})). \tag{19}$$

Now note that we can compute E' by doing n running-min computations, one for each choice of $\bar{j}$. For each $\bar{j}$ let g be an array of size $\bar{j}+1$ (since $\bar{i} \le \bar{j}$) with

$$g[i] = E[i,\bar{j}] + \bar{\mathcal{I}}(i). \tag{20}$$

Then $E'[i,\bar{j}] = h[i]$ where h is the running-min of g. Since it takes $O(n)$ time to compute each running-min we can compute E' in $O(n^2)$ time. Once E' is computed we have

$$F_t[i,j] = \mathcal{I}(i) + \mathcal{J}(j) + \mathcal{C} + \min_{i\le\bar{j}\le j}(\bar{\mathcal{J}}(\bar{j}) + E'[i,\bar{j}]). \tag{21}$$

Once again we can use running-mins. For each i let g be an array of size $n-i+1$ (since $j \ge i$) with

$$g[j-i] = \bar{\mathcal{J}}(j) + E'[i,j]. \tag{22}$$

Then $F_t[i,j] = \mathcal{I}(i) + \mathcal{J}(j) + \mathcal{C} + h[j-i]$ where h is the running-min of g.

This procedure will compute F_t in $O(n^2)$ time for a particular t. As discussed above the other cases are very similar and we omit their details here.

Now we can compute $F_{\ell,\bar{\ell}}$ in $O(n^2)$ time, and by searching over ℓ and $\bar{\ell}$ we get an $O(n^2K^2)$ algorithm for computing F, and thus E_k in each stage of the Viterbi algorithm. This leads to an $O(mn^2K^2)$ labeling algorithm.

3.2. Generalizations

Our algorithm can be generalized in a number of interesting ways. One direction involves using more sophisticated data costs that take into account statistics of the pixels in each column of a region. We can optimize equation (3) for any choice of U_k as long as $U_k(s)$ can be computed in $O(1)$ time. We could penalize labelings if the variance of pixel values in each column of a region is high. In this case $U_k(s)$ can be computed in $O(1)$ using cumulative sums of x and x^2 where x is the value of a pixel in column k.

We can also generalize the algorithm to handle several possible labels for the top and bottom region, just like it handles several labels for the middle region. One could also consider labelings with more than 3 tiers, but this would significantly increase the complexity of the algorithm.

Another generalization involves requiring the middle region to form a single component. We can modify the algorithm so the top and bottom boundaries separate only once. We do this for segmentation in Section 5.

4. Geometric Class Labeling

Geometric class labeling [9], is an application particularly suitable for tiered labelings because of naturally arising geometric constraints on labels. The goal is to obtain a coarse 3D structure from a single image by labeling each pixel with a geometric label that represents a rough surface orientation. We use five labels, namely T (sky), B (ground) and three labels for the middle region, namely L (facing left), R (facing right), C (front facing). The tiered scene restriction implies that any single image column can have only one of L, R or C labels. While this is somewhat restrictive, it is much more general than the model in [15]. In fact [15] is not applicable to general outside scenes, they report a significant worsening of results on the dataset of [9] (compared to [9]), whereas we get an improvement.

We use the dataset of 250 outdoor images with ground truth from [9] (the ground truth is somewhat subjective). We do not model classes “porous” and “solid”, used in [9]. Therefore we do not count errors for pixels with these labels in the ground truth. Here we are not concerned with the particular form of the data terms $D_p(f_p)$, since our goal is to test whether our tiered labeling algorithm is useful for

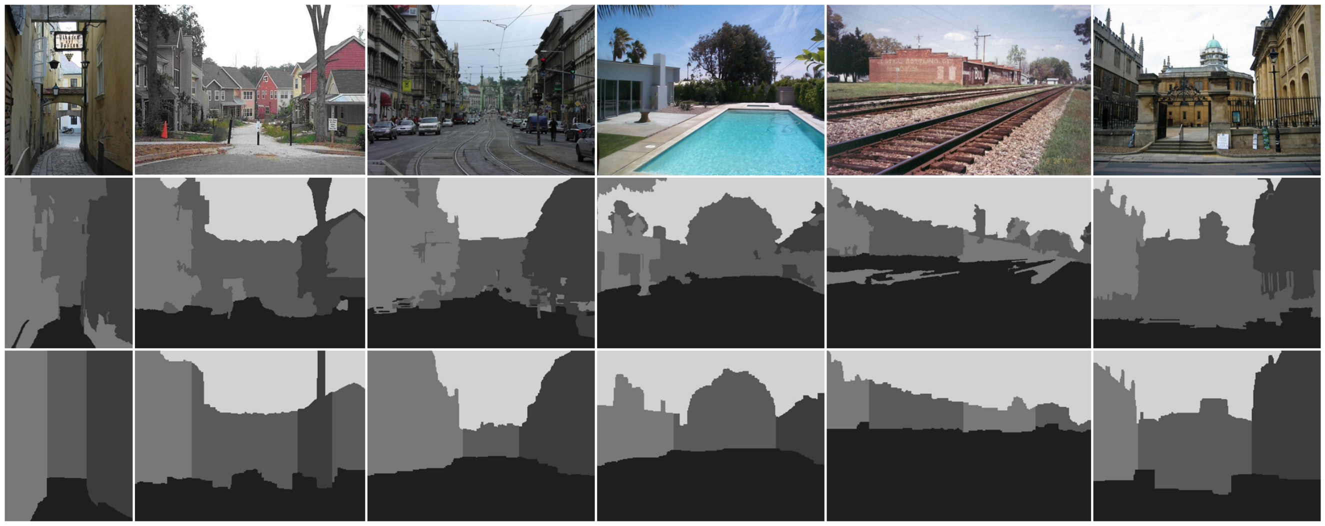


Figure 6. Some results on the dataset from [9]. Top row: original images, second row: confidence only results [9], last row: our results.

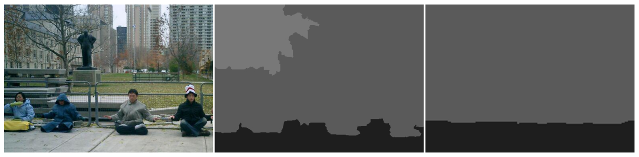


Figure 7. A failure case. Left: original image, middle: confidence [9] only result, last: our result.

improving on the results that do not use global optimization. We use the per-pixel class confidences provided by the authors of [9], generated from five-fold cross validation. We refer to those results as *confidence only*. Since we omit classes "porous" and "solid", the confidence only error rates shown here are better than those in [9].

We use the following observations for V_{pq}: labels L and R are less often adjacent than L and C or C and R. The boundary between R and B is more likely to be slanted up to the right, while between L and B it is more likely to be up to the left. The boundary between T and L, C, R regions is less predictable, because the skyline can be highly irregular. As in [4, 15] we encourage label changes to align with changes in image intensity. We use $V_{pq}(a,b) = w_{pq} \cdot f_{pq}(a,b)$, where w_{pq} is inversely proportional to image gradient. For vertical neighbors $f_{pq}(a,b) = \mathbb{1}(a \neq b)$, while when p is to the left of q $f_{pq}(a,b)$ is given by:

$p\backslash q$	B	L	C	R	T
B	0	1	1	3	2
L	3	0	1	4	1
C	1	1	0	1	1
R	1	4	1	0	1
T	2	1	1	1	0

Figure 6 shows some results. As expected, our labelings (last row) are regularized with smoother boundaries, compared to the confidence only labelings. Of course, if an image violates tiered constraints our results can be worse than when using confidence only, see Figure 7. In this case our algorithm smooths out the right-facing (R) part in the left corner of the image.

The regularization imposed by the tiered labeling algorithm improves the overall per-pixel accuracy in this application. Using confidence only, the accuracy is 78.1%. Our accuracy is 81.4%. The confusion matrices for these two cases are shown below.

confidence only

	B	L	C	R	T
B	85	1	12	2	0
L	10	42	40	9	0
C	10	7	67	15	1
R	4	3	39	52	2
T	0	2	7	1	90

tiered labeling

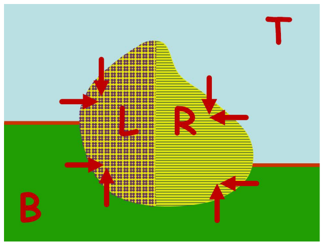
	B	L	C	R	T
B	94	0	4	0	0
L	16	38	36	9	1
C	17	4	62	13	4
R	10	4	35	48	3
T	1	1	5	1	94

The average time to find an optimal labeling (for images approximately 300 by 250 and $K = 5$) was 9.4 sec.

5. Binary Segmentation

We now discuss the application of our tiered labeling algorithm to segmentation with a shape prior. Our approach to shape prior is novel. The prior captures geometric relationships between different parts of an object and the background. The background is divided into two parts (T and B) and the object is divided into as many parts as needed. The availability of a global optimization algorithm lets us use V_{pq}'s that are most appropriate, without worrying about local minima in the energy function, a rare freedom.

We illustrate the idea on a simple example of a "pear" shape in Figure 8. The object is divided into two parts, L and R. The background is divided into T and B. The L part is encouraged to have a boundary with T that is slanted up and to the right, and boundary with B that is slanted down and to the right, etc. As before, $V_{pq}(a,b) = w_{pq} \cdot f_{pq}(a,b)$,



p \ q	B	L	R	T
B	0	1	20	100
L	20	0	0	20
R	1	100	0	1
T	100	1	20	0

Figure 8. "Pear" shape prior illustration. Left: shows the encouraged boundary directions between the two object labels (L and R) and the two background labels (T and B). Right: table of penalties $f_{pq}(a, b)$ when pixel p is to the left of pixel q.

where w_{pq} is inversely proportional to the image gradient. If p and q are horizontal neighbors, $f_{pq}(a, b)$'s are summarized in the table in Figure 8. If p and q are vertical neighbors, then $V_{pq}(a, b) = w_{pq} \cdot \mathbb{1}(a \neq b)$, i.e. the Potts model. If we wanted to encourage a thinner (or wider) shape, we could make vertical w_{pq}'s more (or less) expensive than the horizontal w_{pq}'s. Notice that the penalties listed in Figure 8 are finite, thus the shape encouragement is "soft". Hard constraints can be incorporated too with infinite penalties.

We can use similar ideas to encode various generic shape priors, see Figures 3, 9, 10. We show the recovered object parts with a unique constant intensity. The number of parts needed to model an object depends on the complexity of its shape. In Figure 3 we use four parts for the airplane and bow, seven for the cross (not all parts are clearly visible at the given image resolution), and two for the leaf.

While each object part can have its own data model D_p, for experiments in this section we used the same model for each part, as well as the same model for the two background labels (T and B). The data models were obtained either interactively from the user (Figure 9), as in [3], or set as a mixture of Gaussians (Figures 3 and 10), as in [2].

The middle column in Figures 9 and 10 shows segmentation results without shape prior, i.e. the object consists of a single part. All other parameter settings between the middle column and the last column are equal. As expected, shape prior helps to achieve a more accurate segmentation.

A limitation of our shape prior is the tiered labeling structure. That is each column must contain only one object part label. The class of such shapes is still rather rich, since each object part has several choices of slants to encourage with respect to the background boundary, and the number of object parts to combine is unlimited, although the complexity does increase with K.

If we prefer one object part to be larger than another, we can bias the data terms D_p accordingly. If D_p is smaller for a part we have a bias towards making that part bigger. This was done for the middle part of the shape in the second row of Figure 10, to encourage a flat, rather than triangular roof.

Our shape priors are invariant to translations and they can represent objects of a variety of sizes, since the slant of a boundary does not change if we scale an object. There is also some amount of invariance to rotations. Consider the third row in Figure 9. In general there is a trade-off between being able to tolerate rotations and the "softness" of the prior. If we reduce penalties for unlikely slants, the shape prior becomes more forgiving of rotations, but at the cost of becoming weaker (less specific).

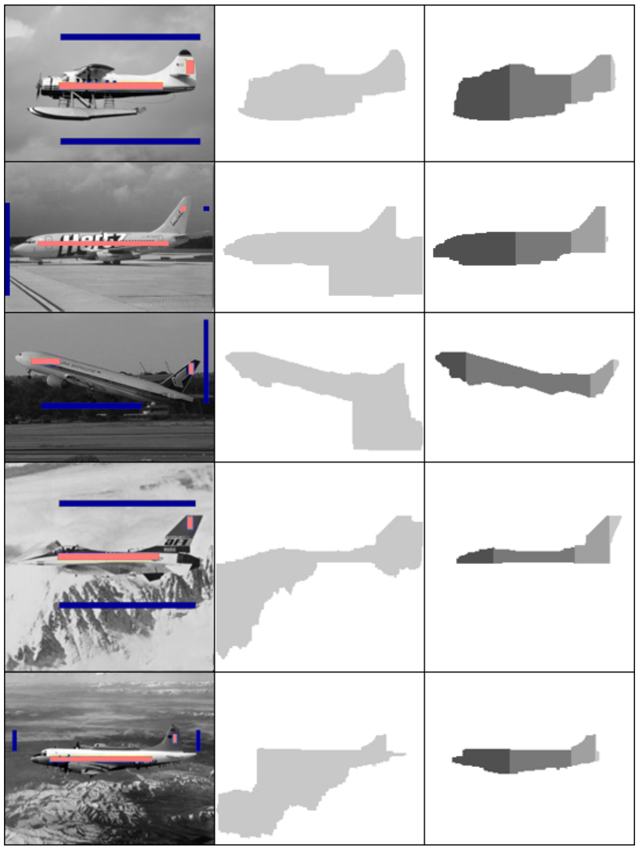


Figure 9. Interactive segmentation with a "plane" shape prior. Middle column: segmentation without shape prior. Last column: segmentation with shape prior. Foreground and background seeds are in black and white respectively.

We tried the expansion algorithm [4] to optimize our energy on the examples in Figure 10. We used the truncation trick of [16], since our energy is not a metric. This attempt was a total failure. Initializing with a constant labeling, the expansion algorithm gets stuck immediately. Initialized at random, it moves to a totally unreasonable local minimum.

The running time depends on the image size and the number of parts in the shape prior. The range for the examples shown here was between 3 and 30 seconds. Our algorithm would be trivial to parallelize for GPU.

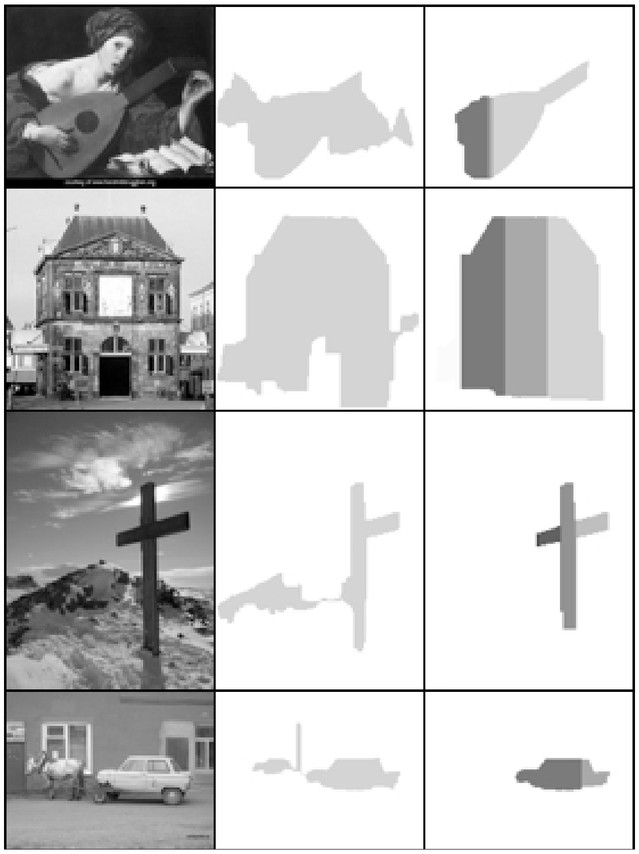


Figure 10. Segmentation without interaction. Middle column: segmentation without shape prior. Last column: segmentation with shape prior. We used a different prior for each example.

6. Conclusion

Our main technical result is an efficient DP algorithm for finding labelings that satisfy tiered constraints. DP can be used because a tiered labeling can be seen as a path in a large state space. In contrast to currently popular optimization methods, such as graph cuts, we find globally optimal solutions with arbitrary smoothness terms V_{pq}. In particular, this allows for modeling fairly interesting shape priors for segmentation and will likely have other applications as well. We also outlined several possible generalizations of our algorithm that may be useful in different applications.

The shape priors introduced here are novel and appear to be quite powerful. One direction for future work involves learning shape priors from examples. Learning such models can be done using standard techniques for learning 1D (hidden) Markov models. Discriminative learning from fully labeled examples can be done using structured SVMs [20]. This would be applicable in the geometric class labeling application. In the case of binary segmentation, discriminative learning from examples that are not pre-segmented into parts can be done using latent structured SVMs [23].

Acknowledgements

We would like to thank Derek Hoiem for providing data for the geometric class labeling experiments. Pedro Felzenszwalb was supported in part by NSF award 0746569. Olga Veksler would like to acknowledge support provided by NSERC, CFI and ERA grants.

References

[1] T. Asano, D. Chen, N. Katoh, and T. Tokuyama. Efficient algorithms for optimization-based image segmentation. *IJCGA*, 11(2):145–166, 2001.

[2] A. Blake, C. Rother, M. Brown, P. Perez, and P. Torr. Interactive image segmentation using an adaptive GMMRF model. In *ECCV*, 2004.

[3] Y. Boykov and G. Funka Lea. Graph cuts and efficient n-d image segmentation. *IJCV*, 69(2):109–131, 2006.

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

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

[6] P. Das, O. Veksler, S. Zavadsky, and Y. Boykov. Semiautomatic segmentation with compact shapre prior. In *CRV*, 2006.

[7] A. Delong and Y. Boykov. Globally optimal segmentation of multi-region objects. In *ICCV*, 2009.

[8] D. Greig, B. Porteous, and A. Seheult. Exact maximum a posteriori estimation for binary images. *Journal of the Royal Statistical Society, Series B*, 51(2):271–279, 1989.

[9] D. Hoiem, A. Efros, and M. Hebert. Recovering surface layout from an image. *IJCV*, 75(1):151–172, 2007.

[10] H. Ishikawa. Exact optimization for markov random fields with convex priors. *PAMI*, 25(10):1333–1336, 2003.

[11] P. Kohli, L. Ladický, and P. H. Torr. Robust higher order potentials for enforcing label consistency. *IJCV*, 82(3):302–324, 2009.

[12] V. Kolmogorov and C. Rother. Minimizing nonsubmodular functions with graph cuts - a review. *PAMI*, 29(7):1274–1279, 2007.

[13] M. Leventon, W. Grimson, and O. Faugeras. Stat. shape influence in geodesic active contours. In *CVPR*, 2000.

[14] S. Li. *Markov Random Field Modeling in Computer Vision*. Springer-Verlag, 1995.

[15] X. Liu, O. Veksler, and J. Samarabandu. Order preserving moves for graph cut based optimization. *PAMI*, to appear.

[16] C. Rother, S. Kumar, V. Kolmogorov, and A. Blake. Digital tapestry. In *CVPR*, 2005.

[17] D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *IJCV*, 47(1-3):7–42, 2002.

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

[19] R. Szeliski, R. Zabih, D. Scharstein, O. Veksler, V. Kolmogorov, A. Agarwala, M. F. Tappen, and C. Rother. A comparative study of energy minimization methods for markov random fields with smoothness-based priors. *PAMI*, 30(6):1068–1080, 2008.

[20] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Support vector machine learning for interdependent and structured output spaces. In *ICML*, 2004.

[21] O. Veksler. Star shape prior for graph-cut image segmentation. In *ECCV*, 2008.

[22] J. Winn and J. Shotton. The layout consistent random field for recognizing and segmenting partially occluded objects. In *CVPR*, 2006.

[23] C.-N. Yu and T. Joachims. Learning structural svms with latent variables. In *ICML*, 2009.