Segmentation of Rectangular Objects Lying on an Unknown Background in a Small Preview Scan Image

Michael Guerzhoy^{*}

University of Toronto guerzhoy@cs.toronto.edu

Abstract

We describe a method to segment rectangular objects that lie on a slightly textured background of an a-priori unknown colour. Our contribution consists of a fast and accurate background colour approximation method, a set of heuristics for accurate detection of rectangle sides, and procedures to generate imprecise hypotheses of rectangles, adjust hypotheses to fit the rectangles in the image, and verify or reject the hypotheses. Our algorithm is capable of detecting overlapping and touching objects such as photos, receipts, and business cards on a very small-sized preview scan image (79 by 109 pixels) on a coloured/textured background

1. Introduction

Automatic cropping of objects in a scanned image considerably speeds up the process of scanning for the user. Ideally, a cropping is proposed by the graphical user interface after obtaining only a low-resolution preview image since the preview image is faster to obtain(in our experiments, we use a 79x109 pixels preview image). While the colour of the background of the scanned image is usually known, some scenarios call for online background colour estimation.

We are motivated by the task of extracting approximately rectangular documents from the preview image. One application of the algorithm is speeding up the process of digitizing photographs by allowing the user to put the photographs on the flatbed scanner in an arbitrary manner and on an arbitrary background and still be able to store each photo automatically in a separate file (Figure 1).

Ordinarily, the background of the scanned image is white. This makes segmentation of mostly white objects, such as business cards or receipts, challenging.

Hui Zhou^{*}

ViXS Systems, Inc hzhou@vixs.com

In fact, many scanner utilities don't provide such functionality. Our algorithm allows the user to automatically crop their rectangular documents by providing their own non-white background (Figure 2). Again, being able to scan multiple cards or receipts at once is useful.

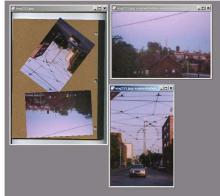


Figure 1. Photo segmentation in preview

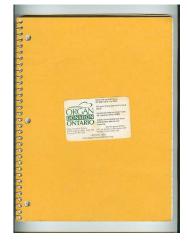


Figure 2. Scanning in a card using our algorithm

^{*} The authors were with Epson Edge, Epson Canada Ltd., Toronto, ON when they worked on this project

In our algorithm, we first estimate the background colour, and then proceed to segment the rectangular objects in the image using the estimated background colour.

Background colour estimation is particularly challenging when most of the background is occluded and a single colour dominates the foreground objects (for instance, this would happen when scanning several photos with snow scenes lying on a white background). We use the assumption that background-coloured pixels will appears contiguously to generate background colour hypotheses, and choose a hypothesis based on the edge statistics of the image.

We detect the background-non background edges in the image and use perceptual grouping to find the edges of the objects and then the objects themselves. Throughout, we use our large synthetically-generated dataset to learn to classify edges and, later, rectangle hypotheses, and verify or reject them.

Popular approaches to the problem of segmentation of rectangular objects include the Generalized Hough Transform that detects rectangles directly following edge detection [9]; Hough transform of the edge space and subsequent search for rectangle hypotheses in Hough space [6]; distance transforms may be used to detect corners and group them into rectangles [11]. The methods above can be directly applied to the problem of segmentation of rectangular objects in a scanned image; however, they do not make use of all of the available domain knowledge. Namely, we can assume that the background is at least approximately uniformly coloured and that the objects are not. We use this assumption to explicitly detect the colour of the background.

Herley [3, 4, 5] addresses the problem of detecting rectangular objects on a scanned image as well as background colour detection under the assumption that the objects are separable by grid-aligned line-segments. Our approach applies when these assumptions don't hold.

Our algorithm is outlined in Section 2. The details of the algorithm are given in Sections 3 through 6. We provide experimental results in Section 7.

2. Algorithm outline

The algorithm proceeds as outlined in Figure 3.

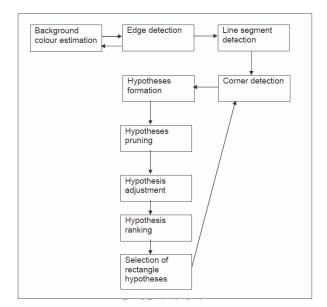


Figure 3. Algorithm outline

3. Background colour estimation

Widely used general background/foreground segmentation methods include pixel clustering using graph-cuts [1], mean-shift and variations, edge detection, and combinations of the above [2]. These methods are of non-linear complexity in the number of pixels being clustered, and do not make strong model assumptions about the input image. The straightforward colour histogram-based method takes the peak of the colour histogram of the image as the background colour. This simple method, however, often fails for our task. Herley [4] proposes a linear-time background colour detection scheme that uses the assumption that the foreground objects are separable by grid-aligned line-segments. Herley constructs a histogram of those colours that account for a majority of pixels in at least k rows or columns. This method, however, sometimes fails in cases where the rectangular objects are not separable by grid-aligned line-segments. A more aggressive approach is therefore needed.

We use the following observations for the background colour **B**:

- a. Since the background is approximately constant, we will observe uniformly coloured contiguous line-segments along image rows and columns that are of colour B
- b. The edges that occur near pixels of colour B will tend to be stronger than those that occur near pixels of other colours
- c. Given the size, shape, and number of the rectangular objects in the image, assuming

some fixed resolution, it is possible to predict the number of edge points near pixels of colour B that will occur in the image.

Our idea is to select the colour C that accounts for the plurality of uniformly-coloured line-segments in the image (observation a) subject to constraints on the strength and quantity of edge points on background/non-background boundaries that were detected by assuming that the background colour is C(observations b and c).

3.1. Segmenting lines into uniformly-coloured segments

We first segment the rows and columns of the image into segments that are roughly the same colour. To do this, we propose a one-pass method, which is explained below.

Consider a line with pixels numbered 0, 1, 2, ... Let *colour(k)* be the RGB colour vector of pixel *k* on the line. Suppose the line contains two differently-coloured line segments, $s_1 = [k_1, k_2]$ and $s_2 = [k_2 + 1, k_3]$. Assume that the colours of the pixels on the line-segments are normally distributed around a mean colour vector C_i , and that the standard deviation of the colour σ is significantly smaller than the difference between the mean colour of s_1 and the mean colour

of
$$s_2$$
. Formally,

color
$$(k) \sim N(C_1, \sigma_1), k \in [k_1, k_2]$$

color $(k) \sim N(C_2, \sigma_2), k \in [k_2 + 1, k_3]$
 $k_1 \leq k_2 \leq k_3$
 $C_1 \neq C_2$
max $(\|\sigma_1\|_{\infty}, \|\sigma_2\|_{\infty}) \ll \|C_1 - C_2\|_{\infty}$

We then define

mean
$$([i, j]) := (\sum_{k=i}^{j} color(k))/(j-i)$$

It's easy to see that mean $([k_1, i+1]) - mean ([k_1, i])$ is symmetrically distributed around $(0 \ 0 \ 0)^T$ for $i < k_2$, and is symmetrically distributed around (approximately) $k_1(C_2 - C_1)/i^2$ for $i \ge k_2$.

Therefore, we can detect that we passed point k_2 if the vectors $D_i = (mean ([k_1, i+1]) - mean ([k_1, i]))$ point in the same direction (roughly $C_2 - C_1$) for i = [j, j+1, ..., j+N-1] (N can be adjusted). We can therefore obtain an approximate segmentation of a line into uniformlycoloured segments by sequentially finding the end point of the line-segments that comprise the line. This method, with N = 3, turns out to work well enough for our purposes for image resolution 79x109. Note that the chosen optimal N will vary with resolution and, to a much lesser extent, the expected typical size of the rectangular objects to be detected, since different run lengths of the same colour would be expected for different resolutions and sizes of objects.

3.2. Selecting candidate background colours

In order to select colours that are likely to be background colours, we assume that there are many long background-coloured line-segments in the image.

Two voting arrays of size 16x16x16 are set up, one for colour means and one for colour variances. The cell (i, j, k) in the means array stores the number of votes for the colour $16^*(i, j, k)$, and the cell (i, j, k) in the variance array stores the (diagonal) variance that corresponds to the colour candidate $16^*(i, j, k)$. The variance will tend to be larger both when there is a genuine variation in background colour in the image and when the estimate $16^*(i, j, k)$ is imprecise.

When a line segment votes, we increment the appropriate cell in the colour means voting array by 1, and the appropriate cell in the colour variances array with the (diagonal) variance of the segment.

The voting array is then smoothed to avoid aliasing, the variances array is normalized, and the colour with the highest vote count is considered to be the most likely background colour. Background colour candidates with high vote counts are then used in edge detection, as described below.

3.3. Edge detection

Given a potential background colour $C = (r \pm \sigma_r, g \pm \sigma_g, b \pm \sigma_b)$, we can attempt to detect edge points near the boundary of the background. For a pixel of colour *C* that lies on the boundary of *C*-coloured and non-*C*-coloured regions, edge strength is calculated. For other pixels, edge strength is set to 0. More specifically, a pixel *p* is of colour *C* if

$$|c_i - p_i| < \max(\sigma_i, \delta), i \in \{r, g, b\}$$

 δ can be chosen to optimize performance.

The pixel *p* is on the boundary given the background colour C if $(C(p) \oplus C(L(p))) \lor (C(p) \oplus C(B(p)))$, where C(p) indicates that *p* is of colour *C*, L(p) is the pixel to the left of *p*, and B(p) is the pixel to the bottom of *p*.

This ensures that we only detect edge pixels on the boundaries of background-coloured and nonbackground-coloured regions in the image.

Edge strength, which will be used in next step, is calculated as follows:

 $EdgeStrength = \delta(U(p), B(p)) + \delta(L(p), R(p)),$ where

 $\delta(a,b) = \min((\max_{i} (|a_i - b_i|), i \in \{r, g, b\}), \Delta), \Delta = 127$

, U(p) is the pixel to the top of p, and R(p) is the pixel to the right of p.

3.4. Detecting the background colour

When the image consists mostly of foreground objects, the background colour determined by only using line-segment information is not reliable. The reason is that the background line-segments tend to be shorter and there are fewer of them for an image with more foreground objects.

We observe that: 1) the average edge strength is higher when the true background colour is used; 2) when the true background is used, the number of background/non-background edge pixels is correlated with the number of foreground objects in the image.

These two observations allow us to heuristically pick the true background colour by combining the edge statistics and the results of the voting by line-segments.

Here follows the procedure to determine the background colour:

- 1. Determine the most likely background colour candidates using line segmentation by only considering line-segments longer than α *Length(scanline) for several α between 0 and 1.
- 2. Among the different colour candidates, choose ones that occur most frequently
- 3. Select the colour that has edge point count within reasonable range whose average edge strength is the largest (we tune the parameters here using our training dataset)

Note that this is effective because we need only arbitrate between different background colour candidates when the background is mostly occluded, since otherwise there is a clear winner in the vote. Since the edge count statistics for this case are more constrained (we know that there are several objects in the image), it is easier to tune the parameters in (3).

4. Feature Grouping

We group edge points into line-segments, which are then grouped into right-angle corners that form our rectangle hypotheses.

Grouping edge points into line-segments proceeds as follows: a small group of spatially 8-connected edge pixels is selected, and is fitted to a line-segment using weighted Total Least Squares, i.e., we compute the major eigenvector of the weighted covariance matrix of the coordinates of the edge pixels, with the weight corresponding to the edge strength. We add neighboring pixels that lie in proximity and along the line-segment until the minor eigenvalue of the weighted variance-covariance matrix exceeds a threshold. This is similar to the UpWrite method [10].

If two line-segments are nearly perpendicular and their ends are sufficiently close, their intersection is marked as a right-angle corner, and the angle of the bisector of the corner is recorded.

Groups of right-angle corners form a rectangle hypothesis if their configuration forms a plausible rectangle, i.e., if two oriented right-angle corners can be seen as lying on the diagonal of a hypothesized rectangle.

Rectangle hypotheses that nearly coincide are merged. This is done to avoid adjusting duplicate hypotheses, since the adjustment operation described below is relatively expensive.

5. Rectangle side detection

Given an inaccurate estimation of the endpoints of a side s of a rectangle and an estimate of the background colour, we need to find the accurate endpoints of s. This is required for the hypothesis adjustment stage immediately after rectangle hypothesis formation.

We devise a score function that reaches maximum when the estimated edge is a true edge, and perform local search to maximize this function: we compute four edge statistics and then learn a function that combines them for optimal detection. Learning-based edge detection in a somewhat different context has been explored in [7].

The score function takes as inputs the image, the end points of the rectangle side *AB*, the center coordinates of the rectangle, and the estimate of the background colour $(r \pm \sigma_r, g \pm \sigma_g, b \pm \sigma_b)$.

We compute the following for points uniformly spaced out across the interval *AB*:

(1) The colour difference in RGB computed along the normal to AB pointing outside the rectangle, where the colour edge between $a = (r_1, g_1, b_1)$ and $b = (r_2, g_2, b_2)$ is $\delta(a, b) = \min((\max(|a_i - b_i|), i \in \{r, g, b\}), 127).$

(2) The difference in "non-backgroundedness" computed along the normal to *AB* pointing outside the rectangle, where the "non-backgroundedness" of pixel $a = (r_1, g_1, b_1)$ given background estimate $c \pm \sigma = (r \pm \sigma_r, g \pm \sigma_g, b \pm \sigma_b)$ is $nbg = \min(4, \max_i (|(a(i) - c(i)) / \sigma_i|))$

We then compute the medians and means of (1) and (2) above for a total of four statistics. We learn an optimal score function that combines these four statistics.

We learn a function of the form $f(s_1, s_2, s_3, s_4) = as_1 + bs_2 + cs_3 + ds_4$ such that f is maximized when applied to a true edge.

For this purpose we generate 100,000 synthetic samples with photos lying on different backgrounds, and quantized each parameter variable into 40 values. For each ground-truth rectangle side in the images, we compute the edge statistics for it and for the edges that are parallel to it and lie close to it. We then proceed iteratively as follows:

• Set (a,b) to initial values, and find the value of (c,d) s.t. the edges localized by the algorithm coincide most closely with the ground truth

• Having found (c, d), search for the optimal (a, b)

- Search for the optimal (*a*,*c*)
- Do the same for other combinations

This appears to be a good way to find f, and the function is fast to compute and not memory intensive. The learning procedure, while non-standard, appears to work well enough for our purposes.

Having found a good edge score function, it is a trivial matter to find the accurate endpoints of the rectangle side given by its approximate endpoints AB by using hill-climbing.

The method described here is also used to refine the cropping in higher resolutions given a cropping in the preview image as an input.

5. Rectangle hypotheses

5.1. Rectangle scoring

We obtain several rectangle hypotheses, and then select the true hypotheses and reject the false ones. The rectangle score function is obtained in the following way:

• We generate rectangle hypotheses in 100,000 synthetic samples as in Section 4.

• For each hypothesis, we adjust it (see below), and calculate the four statistics defined in Section 4 for side. well as the average each as nonbackgroundedness score of the pixels inside the rectangle, with the non-backgroundedness of a pixel given background (p_r, p_b, p_o) $(b_r \pm \sigma_r, b_b \pm \sigma_b, b_g \pm \sigma_g)$ defined as max $(|(p_i - b_i)/\sigma_i|)$.

• We normalize each of the statistics to [0, 1] using the maximum and minimum obtained in each sample for each particular statistic

• We train a decision-stumps-boosted-by-discrete-Adaboost classifier using these statistics and the ground truth data. That is, we set the target to 1 if the rectangle hypothesis corresponds to a true rectangle; and 0 otherwise

The output of the discrete Adaboost classifier serves as the rectangle score for a rectangle hypothesis. The greater the score, the more likely it is that the rectangle hypothesis corresponds to a true rectangle.

5.2. Rectangle adjustment

The statistics defined in Section 4 and used here for scoring the rectangle are sensitive to shifts. We therefore perform for a hypothesis H a local search for each side, or for the whole rectangle when searching for a correct angle (see Section 4), to find the hypothesis H' in the local neighborhood of H such that the sum of the scores of its sides is maximal.

6. Rectangle Detection

We accept rectangle hypotheses iteratively. The adjusted rectangle hypothesis with the highest score is accepted at each iteration unless it overlaps with previously accepted hypotheses by more than 30% of either rectangle's area. The sides of the accepted hypothesis are added into the line-segment list, and we backtrack to corner detection (Section 4) until no new hypotheses are accepted.

We output the list of accepted hypotheses when the loop terminates. The termination condition is as follows: the loop terminates when at least 90% of pixels classified as non-background lie within accepted hypotheses. A pixel (p_r, p_b, p_s) is classified as non-

background	given	background
$(b_r \pm \sigma_r, b_b \pm$	$\sigma_{\scriptscriptstyle b}, b_{\scriptscriptstyle g} \pm \sigma_{\scriptscriptstyle g}$)	if

max $(|(p_i - b_i) / \sigma_i|)$ is greater than a threshold.

7. Results

Table 1 shows the results of running the algorithm on our sample set of a total of 90 images (79x109 pixels) of multiple photos obtained by EPSON Stylus CX5400 in professional mode, with colour and textured backgrounds. The runtime of the algorithm is around 50 milliseconds per image on a 2GHz CPU. See Figure 4 for samples from the test set.

In Table 1, each correctly detected photo counts as a true positive, each detection that does not correspond to a photo counts as a false positive, photos with no corresponding detections count as false negatives, and photos for which the boundaries obtained aren't accurate are counted as incorrect detects.

The main failure modes are:

- Objects where two opposite corners cannot be detected because of the absence of the necessary edges in the image because of similar colours in the foreground and the background
- Pairs of objects which form rectangles, so that the pair is detected as a single object

Our background colour detection algorithm works correctly on 89 of the 90 test samples.



Figure 4. Samples from our test set for 2, 3 and 4 photos in an image

Table 1. Experimental results	Table	1. Ex	perimenta	l results
-------------------------------	-------	-------	-----------	-----------

# of Photos	True	False	False	Incorrect
in image	positives	positives	negatives	detects
1	10	0	0	0
2	30	0	3	1
2 (touching)	20	0	0	2
3	26	0	0	1
3 (touching)	38	2	5	2
4	36	0	0	0

8. Conclusions

We have introduced an approach for rectangular object segmentation in preview of scanned images, including:

- a background colour estimation method;
- an algorithm that uses perceptual organization to detect rectangles on an image given a background colour; and
- a method to adjust detected rectangles to better fit the objects.

The approach can deal with overlapping and touching, and thus can be used for streamlining the scanning of multiple rectangular documents. Because it only requires an image with small resolution, it can be integrated into scanning software for segmentation at the pre-scan stage.

We described an algorithm that relies in part on counting background-non background edge points given a hypothesized background colour to select the correct background colour, and that uses one-pass scanline segmentation in order to use the fact that background patches are contiguous in the image in the background colour detection process. These ideas could conceivably be adapted to other settings.

Our algorithm uses a synthetically generated sample set in order to learn the parameters for edge detection and the procedure that accepts or rejects hypotheses.

Finally, we described applications for rectangular objects segmentation on an unknown background, which include scanning multiple mostly-white documents by placing them on a non-white background.

9. References

[1] Y. Boykov, 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, Sept. 2004

[2] C. M. Christoudias, B. Georgescu, P. Meer, "Synergism in low level vision", 16th International Conference on Pattern Recognition., Quebec City, Canada, volume IV, pages 150–155, 2002

[3] C. Herley, "Recursive method to extract rectangular objects from scans," International Conference on Image Processing, vol.3, no. pp. III- 989-92, 14-17 Sept. 2003

[4] C. Herley, "Recursive Method to Detect and Segment Multiple Rectangular Objects in Scanned Images," technical report MSR-TR-2004-01, Microsoft Research, 2004

[5] C. Herley, "Efficient inscribing of noisy rectangular objects in scanned images," International Conference on Image Processing, Vol.4, 2399- 2402, 24-27 Oct. 2004

[6] C. Jung, R. Schramm, "Rectangle Detection based on a Windowed Hough Transform," Proceedings of the XVII Brazilian Symposium on Computer Graphics and Image Processing 1530-1834, 2004

[7] S. Konishi, A. L. Yuille, J. M. Coughlan, S. C. Zhu, "Statistical Edge Detection: Learning and Evaluating Edge Cues," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 25, no. 1, pp. 57-74, Jan., 2003.

[8] C. Rothwell, "The Importance of Reasoning about Occlusions during Hypothesis Verification in Object Recognition," Rapport de recherche N.2673, INRIA-Sophia Antipolis, Team ROBOTVIS, 1995

[9] Y. Zhu et al., "Automatic Particle Detection through Efficient Hough Transforms," IEEE Transactions on Medical Imaging 22(9): 1053-1062, 2003

[10] R. A. McLaughlin, M. D. Alder "The Hough Transform Versus the UpWrite," IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 20, No. 4, April 1998

[11] Z. Yu, C. Bajaj, "Detecting circular and rectangular particles based on geometric feature detection in electron micrographs," Journal of Structural Biology 145 (2004) 168–180