SpatialBoost: Adding Spatial Reasoning to AdaBoost

Shai Avidan

Mitsubishi Electric Research Labs,
201 Broadway, Cambridge, MA, 02139
avidan@merl.com

Abstract. SpatialBoost extends AdaBoost to incorporate spatial reasoning. We demonstrate the effectiveness of SpatialBoost on the problem of interactive image segmentation. Our application takes as input a tri-map of the original image, trains SpatialBoost on the pixels of the object and the background and use the trained classifier to classify the unlabeled pixels. The spatial reasoning is introduced in the form of weak classifiers that attempt to infer pixel label from the pixel labels of surrounding pixels, after each boosting iteration. We call this variant of AdaBoost — SpatialBoost. We then extend the application to work with "GrabCut". In GrabCut the user casually marks a rectangle around the object, instead of tediously marking a tri-map, and we pose the segmentation as the problem of learning with outliers, where we know that only positive pixels (i.e. pixels that are assumed to belong to the object) might be outliers and in fact should belong to the background.

1 Introduction

Image segmentation is an ill-posed problem and automatic image segmentation is still an illusive target. This led to the development of interactive image segmentation algorithms that allow the user to intervene in the segmentation process with minimal effort.

Image segmentation can be categorized into "soft" and "hard" segmentation. In "soft" segmentation one is interested in recovering both the color and the transparency of the pixels, so that mixed pixels such as hair or fur could be handled, while in "hard" segmentation one is only interested in segmentation the pixels of the object from the background, without recovering their transparency. Here we focus on the latter and note that it can be used in its own right or as an initial guess for soft segmentation.

We treat image segmentation as a binary classification problem, where a classifier is trained on pixels of the object and pixels of the background and then used to classify the unlabeled pixels. We introduce *SpatialBoost* as our classifier. SpatialBoost extends the standard AdaBoost classifier to handle spatial reasoning. This is done by defining two types of weak classifiers. One type is the usual weak classifier that works on each pixel independently. The other type works on the predicted labels of a neighborhood of pixels, after each round of boosting. This allows SpatialBoost to learn spatial arrangements of pixels that can improve the overall quality of the classification. Both types of weak classifiers optimize the same target function and the implementation of SpatialBoost involves just a slight modification of the AdaBoost algorithm.

The three types of pixels (*object*, *background* and *unlabeled*) are defined in a tri-map that the user draw manually. This is often a laborious work and an easier user interface was recently suggested where the user casually marks a rectangle around the object and let the application take care of the rest. We show that our approach can be extended to handle this type of input as well by considering it as learning with outliers. That is, we assume that part of the pixels that are marked as positive (i.e. belong to the object) are actually outliers and should belong to the background.

2 Background

Our work brings together two lines of research. One focused on image segmentation and the other focused on extending AdaBoost to handle spatial information.

Interactive image segmentation has been studied extensively in the past. The Magic Wand [1] allows the user to pick pixels and then automatically cluster together pixels with similar color statistics. Other algorithms take as input a tri-map image. Their goal is to learn from the labeled object and background pixels enough information to correctly label the unlabeled pixels. Ruzon & Tomasi [12] and Chuang *et al.* [4] learn the local statistics of color distribution to predict the label of the unknown pixels. Because color does not carry spatial information they break the region of unlabeled pixels into many sub-regions, in ad-hoc fashion, and process each sub-region independently. In contrast, Boykov & Jolly [3] and later Blake *et al.* [2] use graph-cut algorithms that rely on color and contrast information, together with strong spatial prior to efficiently segment the image. This approach works on the entire image at once and there is no need to process multiple sub-regions separately. Finally, Rother *et al.* [11] eliminated the need for the creation of a tri-map by introducing GrabCut, where the user casually draw a rectangle around the object and the algorithm takes it from there. These methods are generative methods that seek to learn the likelihoods of the colors of the object and background and then, given the unlabeled pixels, determine to which color distribution they belong. We, on the other hand, take a discriminative approach where a classifier is trained on the labeled pixels and then applied to the unlabeled ones.

Efforts to extend AdaBoost to handle spatial reasoning were reported by Fink and Perona [8] who termed their method “Mutual Boost”. They consider the problem of mutual detection of multiple objects in images and use the spatial relationship of AdaBoost classifiers during the detection iterations to improve overall performance. However, they use it for object detection and not for image segmentation. Torralba *et al.* [13] suggested “Boosted random fields” to combine AdaBoost and Belief propagation to handle interaction between neighboring pixels, for the purpose of using context to improve object detection.

3 SpatialBoost: AdaBoost with Spatial Reasoning

We pose image segmentation as a binary classification problem where a classifier is trained on the labeled pixels of the object and the background and then applied to the unlabeled pixels of the border region. In particular, the user constructs a tri-map image that defines pixels that are part of the *object*, part of the *background* or are *unlabeled*. We will term pixels that belong to the object as positive examples and pixels that belong to the background as negative examples. We can train a classifier on the labeled pixels and then apply the classifier to the unlabeled pixels. Recall that AdaBoost training, and testing, is done on each pixel independently, without any spatial interaction between neighboring pixels. Extending the feature vector of every pixel to capture some local image statistics can give a partial solution to the problem but can also pose several new problems. First, the dimensionality of the data grows, which in turn might require additional training data. Second, the interaction between neighboring pixels is limited to the particular image statistics selected. Finally, the information can not be *propagated* beyond the extent of the local image patch that was used to compute the local image statistics.

3.1 SpatialBoost

Within the context of AdaBoost, we give a simple extension that can incorporate spatial reasoning automatically. Given a collection of N data points and their labels, denoted $\{\mathbf{x_i}, y_i\}_{i=1}^{N}$, AdaBoost minimizes the exponential loss function

$$J(H) = E(e^{-yH(\mathbf{x})}) \tag{1}$$

as a way to minimize the zero-one loss function, where $H(\mathbf{x})$, termed the “strong” classifier, is a linear combination of T “weak” classifiers $h_i(\mathbf{x})$.

$$H(\mathbf{x}) = \sum_{i=1}^{T} h_i(\mathbf{x}) \tag{2}$$

We will denote the weak classifiers $h_i(\mathbf{x})$ as *data* classifiers because they operate solely on the data point and do not model spatial interaction between the data points. However, the goal of AdaBoost is to minimize $J(H)$ and every weak classifier that helps the minimization can, and should, be used. In particular, we can use the current labels of the *neighbors* of the pixel to predict its label, in the next iteration of AdaBoost. That is, after each iteration of AdaBoost training we have, in addition to the feature vector of every pixel, the predicted labels of its neighbors. This is the additional information we want to capture and we do that by introducing a new “weak” classifier, that we term *spatial* classifier. In each iteration of AdaBoost training we now train two classifiers. A “data” classifier that was trained on each pixel independently and a “spatial” classifier that was trained on the predicted label of neighborhoods of pixels. AdaBoost now gets to choose the “weak” classifier that minimizes the classification error, be it

Algorithm 1. SpatialBoost - Training

```
Input:   Training set $\{\mathbf{x_i}, y_i\}_{i=1}^N$
         Number of iterations $T$
Output:  A strong classifier $H(\mathbf{x})$

1. Initialize weights $\{w_i\}_{i=1}^N$ to $\frac{1}{N}$
2. Initialize estimated margins $\{\hat{y}_i\}_{i=1}^N$ to zero
3. For $t = 1...T$
   (a) Make $\{w_i\}_{i=1}^N$ a distribution
   (b) Set $\mathbf{x'_i} = \{\hat{y}_j | \mathbf{x_j} \in Nbr(\mathbf{x_i})\}$
   (c) Train weak *data* classifier $h_t$ on the data $\{\mathbf{x_i}, y_i\}_{i=1}^N$ and the weights $\{\mathbf{w_i}\}_{i=1}^N$
   (d) Train weak *spatial* classifier $h'_t$ on the data $\{\mathbf{x'_i}, y_i\}_{i=1}^N$ and the weights $\{\mathbf{w_i}\}_{i=1}^N$
   (e) Set $\epsilon = \sum_{i=1}^N w_i |h_t(\mathbf{x_i}) - y_i|$
   (f) Set $\epsilon' = \sum_{i=1}^N w_i |h'_t(\mathbf{x'_i}) - y_i|$
   (g) Set $\lambda_t = \begin{cases} 1 \text{ if } \epsilon < \epsilon' \\ 0 \text{ otherwise} \end{cases}$
   (h) Set $err = \lambda_t \epsilon + (1 - \lambda_t)\epsilon'$
   (i) Set weak classifier weight $\alpha_t = \frac{1}{2} log \frac{1-err}{err}$
   (j) Update examples weights

         $w_i = w_i e^{(\alpha_t(\lambda_t |h_t(\mathbf{x_i}) - y_i| + (1-\lambda_t)|h'_t(\mathbf{x_i}) - y_i|)}$

   (k) Update margins $\hat{y}_i$ to be

         $\hat{y}_i = \hat{y}_i + \alpha_t(\lambda_t h_t(\mathbf{x_i}) + (1 - \lambda_t) h'_t(\mathbf{x'_i}))$

4. The strong classifier is given by $sign(H(\mathbf{x}))$ where $H(x) = \sum_{t=1}^T \alpha_t(\lambda_t h_t(\mathbf{x}) +$
   $(1 - \lambda_t) h'_t(\mathbf{x}))$
```

the "data" classifier or the "spatial" classifier. As a result, the strong AdaBoost classifier might be a weighted sum of weak *data* and *spatial* classifiers where both types of classifiers work in concert to improve the same objective function. For the weak *spatial* classifiers we actually use the estimated margin of each data point, after each boosting round, instead of the label (which is the sign of the margin).

The SpatialBoost training algorithm is given in Algorithm 1. It takes as input a collection of labeled data points $\{\mathbf{x_i}, y_i\}_{i=1}^N$ and a function $Nbr(\mathbf{x_i})$ that returns the list of neighbors of the point $\mathbf{x_i}$. Once the strong classifier has been trained we can apply it to the unlabeled pixels of the image using Algorithm 2.

3.2 GrabCut – Learning with Outliers

Creating a tri-map image is time consuming and hence, Rother *et al.* [11] suggested GrabCut. In GrabCut the user merely draws a rectangle around the object and the system automatically takes care of the rest. Within the context of SpatialBoost this means nothing more than outlier rejection. Given the rectangle we know that all the pixels outside the rectangle are negative examples, while

Algorithm 2. SpatialBoost - Testing

```
Input:   Unlabeled pixels $\{\mathbf{x_i}\}_{i=1}^{N}$
         The strong classifier $H(\mathbf{x})$
Output:  Labels $\{y_i\}_{i=1}^{N}$

1. Initialize estimated margins $\{\hat{y}_i\}_{i=1}^{N}$ to zero
2. For $t = 1...T$
   (a) Set $\mathbf{x'_i} = \{\hat{y}_j | \mathbf{x_j} \in Nbr(\mathbf{x_i})\}$
   (b) Update margins $\hat{y}_i$ to be
           $\hat{y}_i = \hat{y}_i + \alpha_t(\lambda_t h_t(\mathbf{x_i}) + (1 - \lambda_t)h'_t(\mathbf{x'_i}))$
3. Output $sign(\hat{y}_i)$
```

part of the positive pixels (i.e. pixels inside the rectangle) might be negative examples. Hence, we modify SpatialBoost to handle outliers. A simple approach to outlier rejection is to run SpatialBoost for several iterations and then mark the positive pixels with large weights (i.e. weights larger than a predefined threshold) as outliers, change their label to negative and repeat. In our case, we run SpatialBoost for several iterations (typically, 10 iterations), then take all the positive pixels that are still wrongly classified and have weight greater than $\frac{3}{N}$ (where N is the number of labeled pixels), flip their sign to be negative and restart SpatialBoost. We repeat this procedure for several times (typically, 5 times). Alternatively, one can adopt the BrownBoost algorithm [9].

3.3 The Feature Space

We are also interested in finding what is a good feature space to represent every pixel. Clearly, one can use the (R,G,B) color of every pixel as its feature vector but color carries no spatial information. This can be fixed in one of two ways. One way is to add spatial smoothness assumption, for instance by introducing a penalty term if two neighboring pixels disagree on their label. The second way is to consider more complicated feature spaces that capture both color and spatial information. In our experiments, we use feature vectors that capture the local HoG of every pixel, in addition to the color. This combined feature vector help disambiguate pixels. Working in a high-dimensional space makes it hard to model the distribution of the data points, as is done in generative methods. On the other hand, discriminative methods, such as AdaBoost or SpatialBoost, can give better results.

Also, since our feature space encodes both color and spatial information we do not have to break the image into multiple sub-regions and process each sub-region independently. Breaking the image into sub-region could improve our results, but we prefer to show the advantages of SpatialBoost without ad-hoc improvements.

4 Experiments

We show experiments on synthetic and real images.

4.1 Synthetic Images

To gain some intuition as to how SpatialBoost work we first present experiments on synthetic images where we compare SpatialBoost and AdaBoost on a toy problem of noise removal from binary images. Given a noisy binary image we wish to infer the original "clean" image. To do so, we take a pair of clean/noisy training images, that have the same local image statistics as our test image, and train our classifier on them. We then apply the classifier to the noisy test image. In our case we take the feature vector to be the 3×3 window around every pixel, in the noisy image, and the label of each such data point is taken to be the label of the center pixel of the window, in the corresponding clean image. The neighborhood used by the function $Nbr()$ in the SpatialBoost algorithm is taken to be a 5×5 window around every pixel. Figure 1 compare AdaBoost and SpatialBoost. The size of the images is 100×100 pixels and the amount

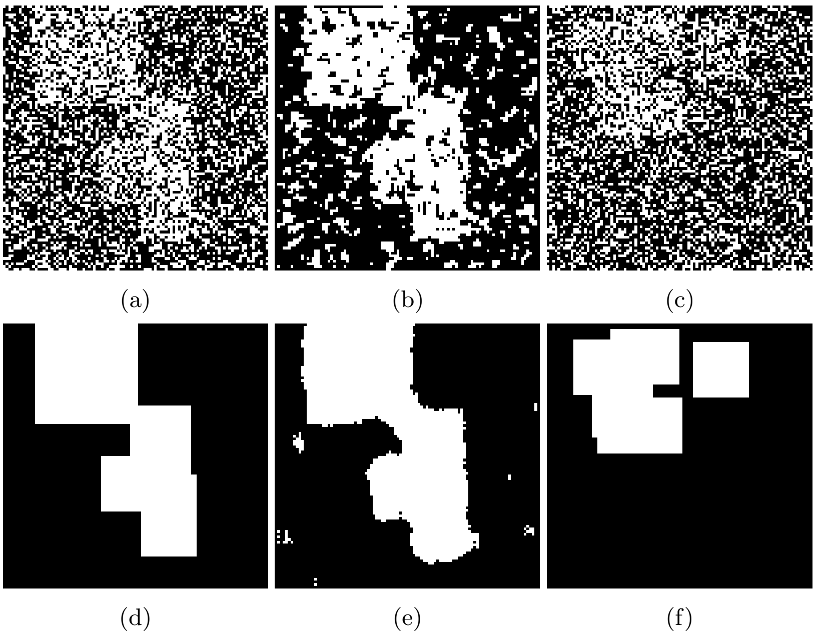

Fig. 1. Noise removal with 35% random noise. Given image (a) we want to infer image (d). We show the results of two methods: AdaBoost (b) and SpatialBoost (e). For training we used images (c) and (f). The "data" classifier takes every 3×3 window in image (c) as a data point whose label is the value of its central pixel in image (f). The "spatial" classifier takes every 5×5 window of the predicted labels in image (c) as a data point.

of noise is 35%, that is we randomly flipped the sign of 35% of all the pixels in the image. One can see that SpatialBoost does a much better job in removing the noise then AdaBoost. This is because SpatialBoost allows information to *propagate* over time (i.e. iterations), whereas in AdaBoost the classification is much more localized.

4.2 Real Images

We now turn our attention to the problem of image segmentation and use the database published by [2].

Pre-processing. We compared two feature spaces. The first one is simply the (R,G,B) values of every pixel. The second feature space consists of color and local Histogram of Oriented Gradients (HoG). HoG is reminiscent of the SIFT detector [7] and has been used in several object detection and recognition applications [5, 6]. In particular, we compute it as follows.

We convert the color image into a gray scale image and compute its x and y derivatives, we then clip pixels whose x and y derivative are below a threshold (5 intensity values, in our case) and create an 8 bin Histogram of Oriented Gradients (HoG) in the neighborhood of each pixel. The feature vector contains both the (R,G,B) values of the pixel, as well as two 8-bin HoGs, on 3×3 and 5×5 windows, centered at the pixel. To improve the weak classifiers, we store several powers of the feature vector elements. Let $\mathbf{f} = [f_1, ..., f_n]$ denote the original feature vector, then we store the feature vector $[\mathbf{f}, \mathbf{f}^2, \mathbf{f}^3]$, that is, we raise every element to all the powers in the range one through three. In total, our feature vector consists of $57 = 3 * (3 + 8 + 8)$ elements. This is a cheap way of gaining kernel power, a-la kernel-SVM, for the weak classifier without implementing an SVM as our weak classifier.

For the *spatial* weak classifier we set the $Nbr()$ function to return a neighborhood of 5×5 pixels around every pixel.

Image Segmentation Results. In figure 2 we compare the different feature spaces (Color Vs. Color+HoG) and the different classifiers (AdaBoost Vs.

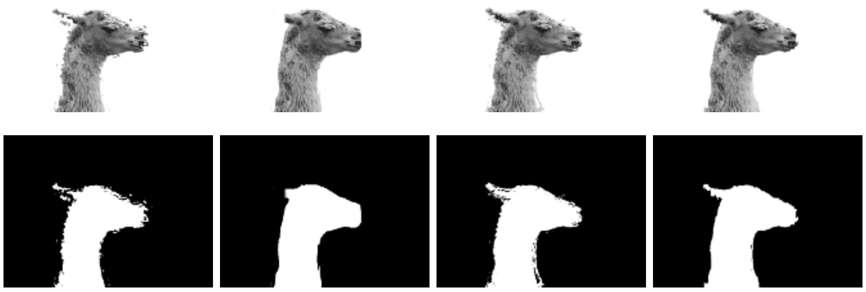

Fig. 2. Comparing feature space as well as AdaBoost Vs. SpatialBoost. First column: AdaBoost + RGB, Second column: SpatialBoost + RGB, Third column: AdaBoost + RGB + HoG, Fourth column: SpatialBoost + RGB + HoG.

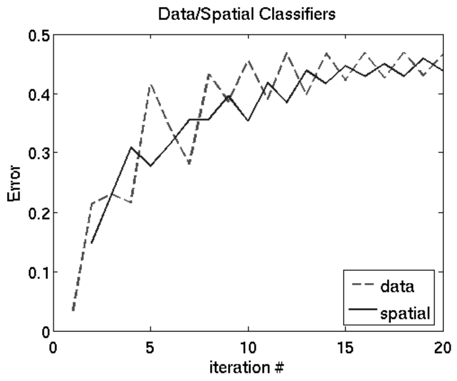

Fig. 3. Roles of *data* and *spatial* classifiers. The x-axis show the iteration number, the y-axis show the error rate of the *data* and *spatial* weak classifiers. In each iteration SpatialBoost chooses the weak classifier with the lowest error rate. As can be seen, In the first iteration the weak *data* classifier gives the lowest error rate, after that, the two types of weak classifier play interleaving roles.

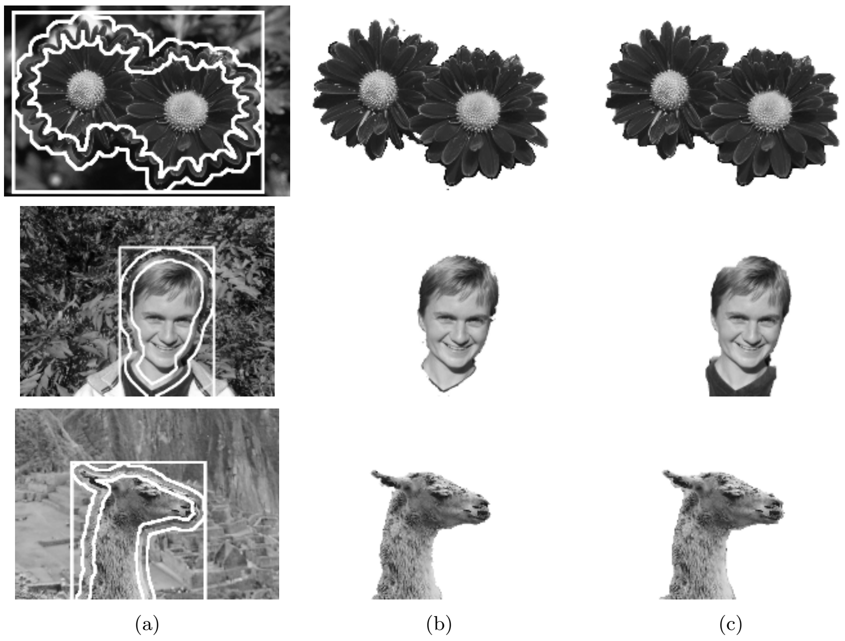

Fig. 4. Experiments on real data. Each row correspond to one example. Column (a) show the input image, column (b) show the results on the tri-map data and column (c) show the results on the GrabCut input. The tri-map and GrabCut inputs are overlaid on the original image. In the tri-map case, the inner most region is marked as positive, the outer most region is marked as negative and the region between the two is the test pixels to be classified. In the GrabCut method all the pixels outside the rectangle are marked as negative and all the pixels inside are marked as positive. The algorithm must determine which of the "positive" pixels is an outlier and should in fact belong to the background.

SpatialBoost). As expected, the combined Color+HoG conveys additional information that improves the results of both AdaBoost and SpatialBoost. Of the two, SpatialBoost produces better looking, and more accurate, segmentation results. In both cases we used weighted least squares as our weak learner.

Next, we measured the role *data* and *spatial* classifiers play in SpatialBoost. Figure 3 shows a plot of the error rate of each of these classifiers when trained on the llama image (shown in figure 2). In each round SpatialBoost picks the classifiers with the lowest error rate and as can be seen from the graph, the two types of classifiers play interleaving roles. At the first iteration, SpatialBoost picks a *data* classifier, but in the second iteration it picks a *spatial* classifiers because it has a lower error rate, and so on.

Figure 4 show some results of running SpatialBoost on some real images. We show results of running SpatialBoost with tri-map and GrabCut, as they appear in the database. No morphological post-processing operations are performed to enhance the results. We ran SpatialBoost on all 50 images in the database and found the average error rate to be 8.00% for the set of 30 training images and 8.23% for the set of 20 test images. The best results, for the tri-map input

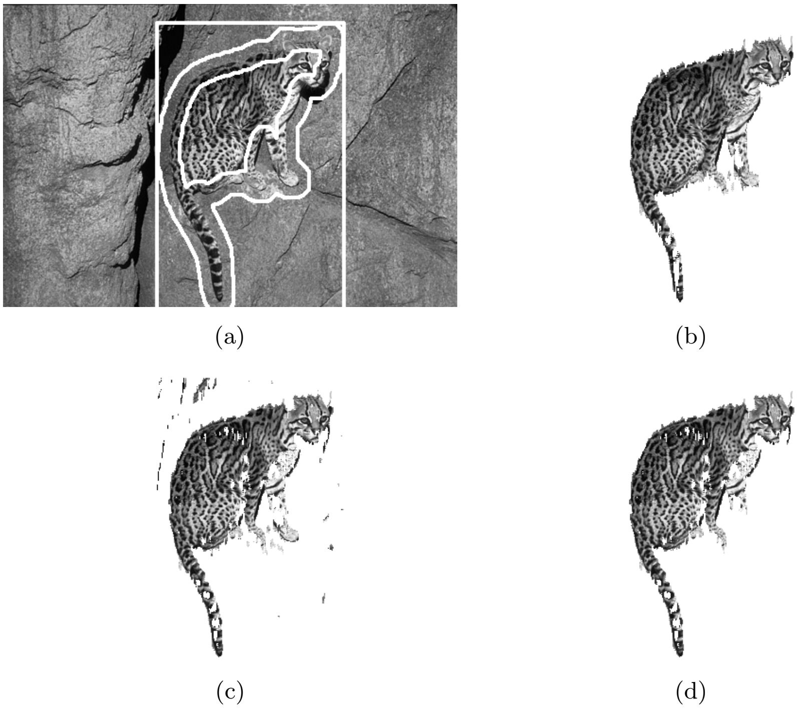

Fig. 5. Experiments on real data. Adding morphological post-processing can further improve results. Image (a) show the input image (with the tri-map and GrabCut boundaries overlaid), image (b) show the results with the tri-map input, image (c) show the results with the GrabCut input and image (d) show only the largest connected component of image (c).

reported by [2] are 7.9% on a test set of 20 out of the 50 images. They do not report results for the GrabCut input. We obtain an error rate of 8.84% for the 30 training images and 11.96% for the 20 test images, in the GrabCut case. In their follow-up work [11], where GrabCut was introduced, the authors show examples with multiple user inputs and so direct comparison is no longer possible.

A couple of comments are in order. First, we found that choosing small neighborhood windows gave better results, this is because larger neighborhood windows lead to blur that degrades segmentation performance. Second, we found that a large number of iterations actually help the segmentation as it allows the propagation phase to spread the information. Finally, the method takes a couple of seconds to run on a non-optimized MATLAB implementation.

In figure 5 we show results of combining SpatialBoost with basic morphological operations. In this case we cleaned the result of SpatialBoost in the case of GrabCut by detecting and keeping the largest connected component. The results on the tri-map input usually do not require the use of morphological post-processing operations.

SpatialBoost will automatically default to the standard AdaBoost algorithm in case there is no spatial information to be used for classification. Indeed, we tested spatialBoost on some of the UCI ML repostiroy [10] datasets (Ionosphere and Glass) and found that no “spatial” classifier was chosen. Specifically, the “spatial” classifier was trained on the predicted label of the 3 nearest examples, but apparently this information was not useful.

5 Conclusions

We give a simple extension to AdaBoost to handle spatial information. In addition to the usual weak *data* classifiers, we introduce weak *spatial* classifiers that work on the labels of the data points, after each iteration of the boosting algorithm. In each SpatialBoost iteration the algorithm chooses the best weak classifier (either *data* or *spatial* classifier) to be added. Results on synthetic and real images show the superiority of SpatialBoost over AdaBoost in cases that involve spatial reasoning.

References

1. Adobe System, 2002. Adobe Photoshop User Guide.
2. Blake, A. and Rother, C. and Brown, M. and Perez, P. and Torr, P. Interactive Image Segmentation using an adaptive GMMRF model. In *Proc. European Conf. Computer Vision*, 2004.
3. Boykov, Y. and Jolly, M.-P. Interactive graph cuts for optimal boundary and region segmentation of objects in N-D images. In *Proc. IEEE International Conference on Computer Vision*, 2001.
4. Chuang, y.-y., Curless, B., Salesin, D. and Szeliski, R. A Bayesian approach to digital matting. In *Proc IEEE Conf. on Computer Vision and Pattern Recognition*, 2001.

5. N. Dalal and B. Triggs. Histograms of Oriented Gradients for Human Detection. In *IEEE Conference on Computer Vision and Pattern Recognition* (CVPR), 2005.
6. W. T. Freeman and M. Roth. Orientation histograms for hand gesture recognition. In *Intl. Workshop on Automatic Face and Gesture Recognition*, 1995.
7. D. G. Lowe. Distinctive image features from scale-invariant keypoints. In *International Journal of Computer Vision* (IJCV, 60(2):91-110, 2004.
8. Fink, M. and Perona, P. Mutual Boosting for Contextual Inference. In *Adv. in Neural Information Processing Systems* (NIPS), 2003.
9. Freund, Y. An adaptive version of the boost by majority algorithm. In *Machine Learning*, 43(3):293-318, June 2001.
10. D.J. Newman, S. Hettich, C.L. Blake and C.J. Merz, UCI Repository of machine learning databases, url = "http://www.ics.uci.edu/~mlearn/MLRepository.html", University of California, Irvine, Dept. of Information and Computer Sciences, 1998.
11. Rother, C. and Kolmogorov, V. and Blake, A. GrabCut - Interactive Foreground Extraction using Iterated Graph Cuts, In *Proc. ACM Siggraph*, 2004.
12. Ruzon, M. and Tomasi, C. Alpha estimation in natural images. In *Proc IEEE Conf. on Computer Vision and Pattern Recognition*, 2000.
13. A. Torralba, K. P. Murphy and W. T. Freeman. contextual Models for Object Detection using Boosted Random Fields. In *Adv. in Neural Information Processing Systems* (NIPS), 2004.