Shape Agnostic Pedestrian Segmentation Using Joint Kernel Support Estimation

Ethan A. Benjamin
eb2947@columbia.edu

Abstract

Person segmentation, the task of identifying pixels in an image corresponding to a person, is an important problem in computer vision, and has applications to image editing, surveillance, and autonomous vehicles[1]. One particularly important application of this task is segmenting imagery of pedestrians.

Many existing approaches to pedestrian segmentation are based on generative shape models [1] [2] [3]. These generative models need to account for a sufficiently diverse array of shapes , and each rely upon large training sets of shape exemplars outside of the data sets which they are evaluated on.

In this project I adopt an algorithm which applies the technique of Joint Kernel Support Estimation [9] to the problem of supervised object segmentation. In particular, I implement the algorithm developed by Manfredi et al [4] (which derives from the work of Bertelli,Yu, et al [5]) and adapt it for use on the Penn-Fudan pedestrian data set[6].

I demonstrate that by using this technique I obtain foreground and background accuracy comparable to state-of-the-art approaches for pedestrian segmentation while using only a small number of training examples taken from the Penn-Fudan data set.

1 Introduction

My paper is organized as follows: in Section 2, I review the supervised object segmentation algorithm of Manfredi and discuss details of my implementation, particularly concerning ambiguous aspects of Manfredi's paper. In Section 3 I discuss the adaptations I made to this algorithm for use on the Penn-Fudan pedestrian data set. In Section 4 I compare my results to other state of the art algorithms on this data set. Finally, I conclude by discussing potential areas for future work.

2 Algorithm

2.1 Image-Mask Kernels

I follow the algorithm proposed by Manfredi et al, which builds on the work of Bertelli et al. Both approaches rely on a sophisticated kernel between pairs of images and their corresponding binary object masks.

Given images x_i and x_j and their corresponding foreground masks y_i and y_j we define:

$$K((x_i, y_i), (x_j, y_j)) = \theta(x_i, x_j) \cdot \Omega(x_i, x_j, y_i, y_j) \quad (1)$$

The object similarity kernel $\theta(x_i, x_j)$ is a Gaussian kernel between image features $\phi(x_i)$ and $\phi(x_j)$:

$$\theta(x_i, x_j) = \exp(-\frac{||\phi(x_i) - \phi(x_j)||^2}{2\sigma^2}) \quad (2)$$

This term serves to filter out pairs for which there is low similarity between images/objects. It is particularly important at classification time, when we must weigh the similarity of an unlabeled test image to existing support vectors. Just as Manfredi, I use HOG descriptors for image features.

The image-mask kernel Ω is a weighted sum of component image-mask kernels:

$$\Omega(x_i, x_j, y_i, y_j) = \sum_{k=1}^{3} \beta_k \Omega_k(x_i, x_j, y_i, y_j) \quad (3)$$

The "mask similarity" kernel $\Omega_1(y_i, y_j)$ depends only on the binary masks, and simply measures the percentage of pixel labels which are equal in both masks:

$$\Omega_1(y_i, y_j) = \frac{1}{P} \sum_{p=1}^{P} \delta(y_{ip}, y_{jp}) \quad (4)$$

Where δ is the indicator function.

The remaining image-mask kernels Ω_2 and Ω_3 each rely on color histograms built from the input RGB images. I create a histogram as follows: assuming integer channels and given per-channel bin count C as well as channel maximum value M, I set the bin index of a color with channels r,g, and b as $\left\lfloor \frac{r}{M+1} C \right\rfloor + \left\lfloor \frac{g}{M+1} C^2 \right\rfloor + \left\lfloor \frac{b}{M+1} C^3 \right\rfloor$. Thus to account for every possible color I create a histogram with C^3 bins. As Manfredi suggests I set $C = 16$.

Define Count(x_p, H) as the bin count of color x_p in histogram H, and Count(H) as the sum of all bins in H. Let $P(x_p|H)$ be the probability of color x_p in histogram H. Manfredi leaves this expression undefined, but I simply set it as $P(x_p|H) =$ Count(x_p, H)/Count(H). Define $L(x_p|H)$ as negative log probability , $L(x_p|H) = -\log(P(x_p|H))$. Note that this is undefined if Count$(x_p) = 0$. I avoid this by adding 1 to all histogram bin counts. Finally, given foreground histogram F and background histogram B define:

$$L(x_p|y_p, F, B) = \begin{cases} L(x_p|B) & \text{if } y_p = \text{“}object\text{”} \\ L(x_p|F) & \text{if } y_p = \text{“}background\text{”} \end{cases}$$

Intuitively, $L(x_p|y_p, F, B)$ rewards foreground pixels which are unlikely in the background, and background pixels which are unlikely in the foreground. Let F_i^j be the histogram of foreground pixels obtained applying mask y_j to image x_i, and B_i^j be the same

for background pixels. Then we define the "local color model kernel" Ω_2 as:

$$\Omega_2(x_i, y_i, x_j, y_j) = \frac{1}{P}\sum_{p=1}^{P} L(x_{ip}|y_{ip}, F_i^j, b_i^j) \tag{5}$$

Intuitively, Ω_2 rewards masks y_j which are able to best distinguish color profiles in foreground and background regions when applied x_i.

Next, define F_G and B_G as the foreground and background color histograms taken across all training images. We define the "global color model kernel" Ω_3 as

$$\Omega_3(x_i, x_j, y_i, y_j) = \Omega_{3i} \cdot \Omega_{3j} \tag{6}$$

Where

$$\Omega_{3i} = \frac{1}{P}\sum_{p=1}^{P} L(x_{ip}|y_{ip}, F_G, B_G) \tag{7}$$

$$\Omega_{3j} = \frac{1}{P}\sum_{p=1}^{P} L(x_{jp}|y_{jp}, F_G, B_G) \tag{8}$$

Note that Ω_3 does not directly compare images or masks. Rather, it has a general penalizing effect on image-mask pairs which do not conform to the global color properties of the data set.

2.2 Joint Kernel Support Estimation

In Bertelli's original work[5], he trains a structural SVM over image-mask pairs using kernels very similar to those described above. After training, the structural SVM gives the following decision function:

$$F(x, y) = \sum_{\bar{y}\in\mathcal{W}} \alpha_{\bar{y}}^* \left(\frac{1}{n}\sum_{i=1}^{n}[K((x,y),(x_i,y_i)) - K((x,y),(x,\bar{y}_i))]\right) \tag{9}$$

Here, each $\bar{y} \in \mathcal{W}$ is one of the most violated constraints found in the course of training. For a given test image x, we find the optimal mask by obtaining $y^* = \arg\max F(x, y)$.

Instead of using structural SVMs, Manfredi uses a structured prediction technique called Joint Kernel Support Estimation[9]. Given observation-label pairs $(x_1, y_1), ..., (x_n, y_n)$, JKSE models the support of density function $p(x, y)$ and uses $f(x) = \arg\max p(x, y)$ for prediction. It does this by training a one-class SVM over training examples consisting of observation-label pairs. A parameter $\nu \in (0, 1]$ to one-class SVM is an upper bound to the number of outliers; the higher the value of ν, the more freedom the method has to discard training samples. We determine the optimal value of ν for a given data set through cross validation.

Because one-class SVM is kernalizable, JKSE is capable of modeling any hilbert space induced by a joint kernel function such as the one we have defined. After training our one-class SVM, we obtain the following decision function:

$$F(x) = \arg\max_{y\in Y} \sum_{i=1}^{n} \alpha_i K((x, y), (x_i, y_i))$$

Where each α_i is the dual coeficient of its corresponding support vector. Note that here, each support vector is an image-mask pair. I use the Python library *scikit-learn* for one-class SVM.[11]

A significant advantage of JKSE compared to structural SVM is that we do not need compute arg max solutions during training. Rather, we must simply provide a gram matrix. This makes JKSE far faster than structural SVM. Manfredi shows that segmentation accuracy using JKSE is virtually identical to that structural SVM. I further speed up computation of the gram matrix by splitting it into chunks and computing each chunk in a separate process.

2.3 Graphcut For Argmax Solution

To find the arg max solution $y^* = F(x_j)$ for a test image x_j, Manfredi performs energy minimization using s-t graphcut[7].

Based on the definition of our image-mask kernels, we can formulate each pixel's unary potential as a sum of pixel-specific kernel terms weighted by support vector dual coefficients. In particular, the foreground and background unary potentials x_{fp} and x_{bp} for a pixel x_p in image x_j are given by:

$$x_{fp} = \sum_{i=1}^{n} \alpha_i \theta(x_j, x_i)(\beta_1 y_{ip} + \beta_2 L(x_p|B_j^i) + \beta_3 x_{fp}^3) \tag{10}$$

$$x_{bp} = \sum_{i=1}^{n} \alpha_i \theta(x_j, x_i)(\beta_1 (1 - y_{ip}) + \beta_2 L(x_p|F_j^i) + \beta_3 x_{bp}^3) \tag{11}$$

Where we define:

$$x_{fp}^3 = L(x_p|B_G) \cdot \frac{1}{P}\sum_{p=1}^{P} L(x_{ip}|y_{ip}, F_G, B_G)$$

$$x_{bp}^3 = L(x_p|F_G) \cdot \frac{1}{P}\sum_{p=1}^{P} L(x_{ip}|y_{ip}, F_G, B_G)$$

Additionally, Manfredi introduces a binary smoothness term $B(y)$ over the output binary mask which encourages neighboring pixels with similar colors to share the same label:

$$B(y) = \sum_{p,q\in\mathcal{N}} \delta(y_p, y_q) \frac{1}{\text{dist}(p, q)} \exp(-\frac{||x_p - x_q||^2}{2\sigma^2})$$

Here, $\mathcal{N}$ contains all pairs of neighboring pixels, $\text{dist}(p, q)$ is the distance between pixels, and σ is the expected value of $||x_p - x_q||^2$ across the image.

Let $R(y)$ express the sum of unary potentials we have defined across all pixels. Then our problem can be formulated as a MAP estimation for a markov random field in which we seek to minimize $R(y) + \lambda B(y)$. Parameter λ weighs the importance of our binary smoothness term, and is tuned by cross-validation. I used the Python library *PyMaxflow*[12] to perform s-t graph cuts.

2.4 Replicating Previous Results

To ensure that my implementation of Manfredi's algorithm was correct, I tested it on the Weizmann horse dataset [8], which contains 328 images of horses in a wide variety of backgrounds and poses.

I used parameters β, λ, and ν which Manfredi found to work well for this data set. On each experiment, I randomly chose 200 images for training and 64 for testing.

To evaluate test performance on a given image, I obtain a solution y^* and compared it to the ground truth mask y using the same two metrics given by Manfredi. The first, S_α, measures the percentage of pixels for which $y_p^* = y_p$. The second, S_o, is defined as the intersection of object pixels between masks over their union, or:

$$S_o = \frac{\sum_{p=1}^{P} y_p = \text{“object”} \wedge y_p^* = \text{“object}}{\sum_{p=1}^{P} y_p = \text{“object”} \vee y_p^* = \text{“object}} \tag{12}$$

I conducted 10 total experiments. My mean result for S_α was 92.82 with a standard deviation of .60. My mean result for S_o was 75.26 with a standard deviation of 1.46. These are each close to the results reported by Manfredi, who obtained $S_\alpha = 93.04$ and $S_o = 76.32$.

3 Adpating To Pedestrians

3.1 The Penn-Fudan Data Set

I adapted Manfredi's algorithm to the Penn-Fudan data set [6]. It contains 170 color images with 345 labeled images of pedestrians from which 169 labels are used in [1][2][3]. I use this same subset of data.

3.2 Enhancing Training Data

The Penn-Fudan data set is constrained by its limited size. I added variety to training data by flipping each training image horizontally and adding the flipped image to the training set. I found that this had a small beneficial impact on performance, and seemed to increase the odds that a test image would have high unary potentials on difficult areas such as the lower extremities of the human body, which tend to exhibit higher variety than the torso.

3.3 Image Pre-Processing

The cropped images in the Penn-Fudan data set came in a variety of sizes. In order to compute kernels between images I needed to ensure every image could be compared in an equivalent dimension. To this end, I resized all images to 100 by 270 pixels, which conforms closely to the mean width and aspect ratio of images in the data set.

3.4 Feature Selection

I used HOG descriptors as image features. Initially, following Manfredi, I used square cells and blocks with enough pixels in a cell to fit 5 cells in a window width-wise. Since my canonical image dimension was 2.7 times higher than wide, this resulted in a greater vertical than horizontal cell span, and meant that there were more HOG features than would be produced in a square image.

After trial and error, I found I achieved better test performance by setting both horizontal and vertical cell size to fill one fifth of the image window in their respective dimensions. This reduced the number of HOG features for each image from 729 to 324. I believe that due to the small size of training data, fine grained HOG descriptors result in overfitting, and reducing feature size alleviates this.

After making this adjustment, I found that the image similarity kernel between test and training images was consistently high for image pairs which exhibited similar pose and zoom, and thus similar optimal segmentation. This is illustrated in Figure 1.

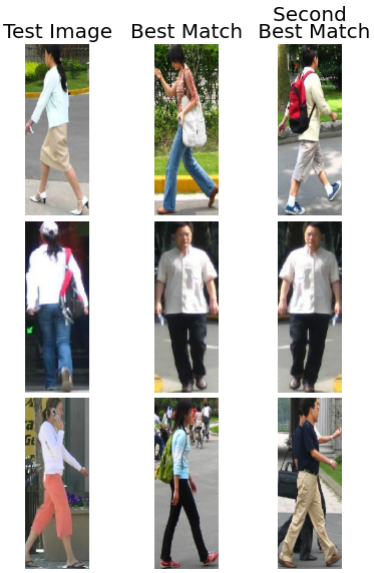

Figure 1. Left column: three test images x_i. Middle and right columns: training images x for which $\theta(x_i, x)$ is highest.

3.5 Cross-Validation of Parameters

In order to optimize our algorithm for the Penn-Fudan data set I searched over parameters β, λ, and ν which minimized test error. Initially, I used a process of trial and error to find ranges of parameters which tended to produce strong results.

I found that the optimal value of β_1 was far higher for this data set than the reported optimal values other data sets. Optimal values for β_1 ranged from roughly .8 to 1.5, while the optimal values for β_1 in horse and flower data sets were reported at .28 and .20 respectively. This likely reflects the fact that there is less variation in pedestrian masks than in masks from the horse and flower data sets. This is particularly true for the Penn-Fudan data set, in which pedestrians are centered, and images do not extend beyond the bounding box of the pedestrian.

I also found that the optimal β_3 was lower than in horse or flower data sets. This may be because few colors in this data set are unique to pedestrians or to the background. The few exceptions include strong shades of red (seen in clothing and hardly

ever in the background) and dark green (seen in many backgrounds including grass but not in pedestrians).

My procedure for cross-validating over β,λ, and ν works as follows: after finding rough ranges of optimal values for each parameter through trial and error, I chose 5 search values for each parameter to iterate through on each round of cross-validation. On a given round of cross-validation, for each parameter, for each search value, I train my model with the parameter set to this search value and all other parameters set to their median search value. Then I evaluate on cross-validation data. I do this for all parameters and all search values. Then, for each parameter, I select the search value which performed best on cross-validation data.

This procedure assumes that parameters do not work synergystically, which is not entirely true. However, by making this simplification I am able to perform trials of cross validation in parallel and greatly reduce training time.

4 Results

I ran 10 experiments on the Penn-Fudan test data. On each experiment, I randomly chose 100 images for training, 34 for cross-validation, and 35 images for testing.

I evaluated foreground (FG) accuracy and background (BG) accuracy using the intersection/union criterium of the PASCAL VOC challenge, just as in [1][2][3].

	Mine	Flohr[1]	Eslami[2]	Bo[3]
FG	76.47	78.5	71.6	73.3
BG	78.60	81.5	73.8	81.1
Average	77.54	80.0	72.7	77.2

As we can see, my algorithm performs better on average than all but one algorithm.

In Figure 2 I illustrate results of my algorithm,unary potentials induced by each of the image-mask kernels Ω_k (foreground minus background), and full unary potential. We can see that by favoring the masks of training images close to the test image, Ω_1 is able to capture information about body width and leg spread. The local color kernel Ω_2 is invaluable when the color profile of a pedestrian is distinct from the background, which tends to be the case. Finally, we can see that Ω_3 tends to convey useful energy only for the few foreground colors that are rare in the background and vice versa.

5 Conclusion

By adapting Manfredi's algorithm for structured object segmentation to the Penn-Fudan data set, I was able to achieve performance comparable with other state of the art pedestrian segmentation algorithms. Unlike other pedestrian segmentation algorithms which have been evaluated on the Penn-Fudan data set, my algorithm does not attempt to create a generative shape model, and does not rely on any external training data.

Future work may wish to explore different classes of image features for the image-image kernel. Bertelli and Manfredi's kernel is flexible in that it is amenable to a wide variety of feature types. It is possible that features such as GIST or a discriminative part based model such the one Bertelli used for horses could perform better for pedestrians.

References

[1] Fabian, and Gavrila. "PedCut: an iterative framework for pedestrian segmentation combining shape models and multiple data cues." Proc. BMVC. 2013.

[2] Eslami and Williams. "A generative model for parts-based object segmentation." Advances in Neural Information Processing Systems. 2012.

[3] Bo and Fowlkes. "Shape-based pedestrian parsing." Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on. IEEE, 2011.

[4] Manfredi, Grana, and Cucchiara. "Learning Graph Cut Energy Functions for Image Segmentation." Pattern Recognition (ICPR), 2014 22nd International Conference on. IEEE, 2014.

[5] Bertelli, Yu, et al. "Kernelized structural SVM learning for supervised object segmentation." Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on. IEEE, 2011.

[6] Wang, Liming, et al. "Object detection combining recognition and segmentation." Computer VisionACCV 2007. Springer Berlin Heidelberg, 2007. 189-199.

[7] Boykov and Jolly. Interactive graph cuts for optimal boundary & region segmentation of objects in ND images." Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on. Vol. 1. IEEE, 2001.

[8] E. Borenstein, E. Sharon, and S. Ullman, "Combining top-down and bottom-up segmentation," in *IEEE International Conference on Computer Vision and Pattern Recognition WOrkshops, 2004* CVPRW '04., vol 4, 2004, pp. 46-54.

[9] Lampert and Blaschko. "Structured prediction by joint kernel support estimation." Machine Learning 77.2-3 (2009): 249-269.

[10] Schlkopf, Bernhard, et al. "Estimating the support of a high-dimensional distribution." Neural computation 13.7 (2001): 1443-1471.

[11] scikit-learn: Machine Learning in Python. http://scikit-learn.org/stable/. Accessed May 6th, 2015.

[12] PyMaxflow. https://github.com/pmneila/PyMaxflow. Accessed May 6th, 2015.

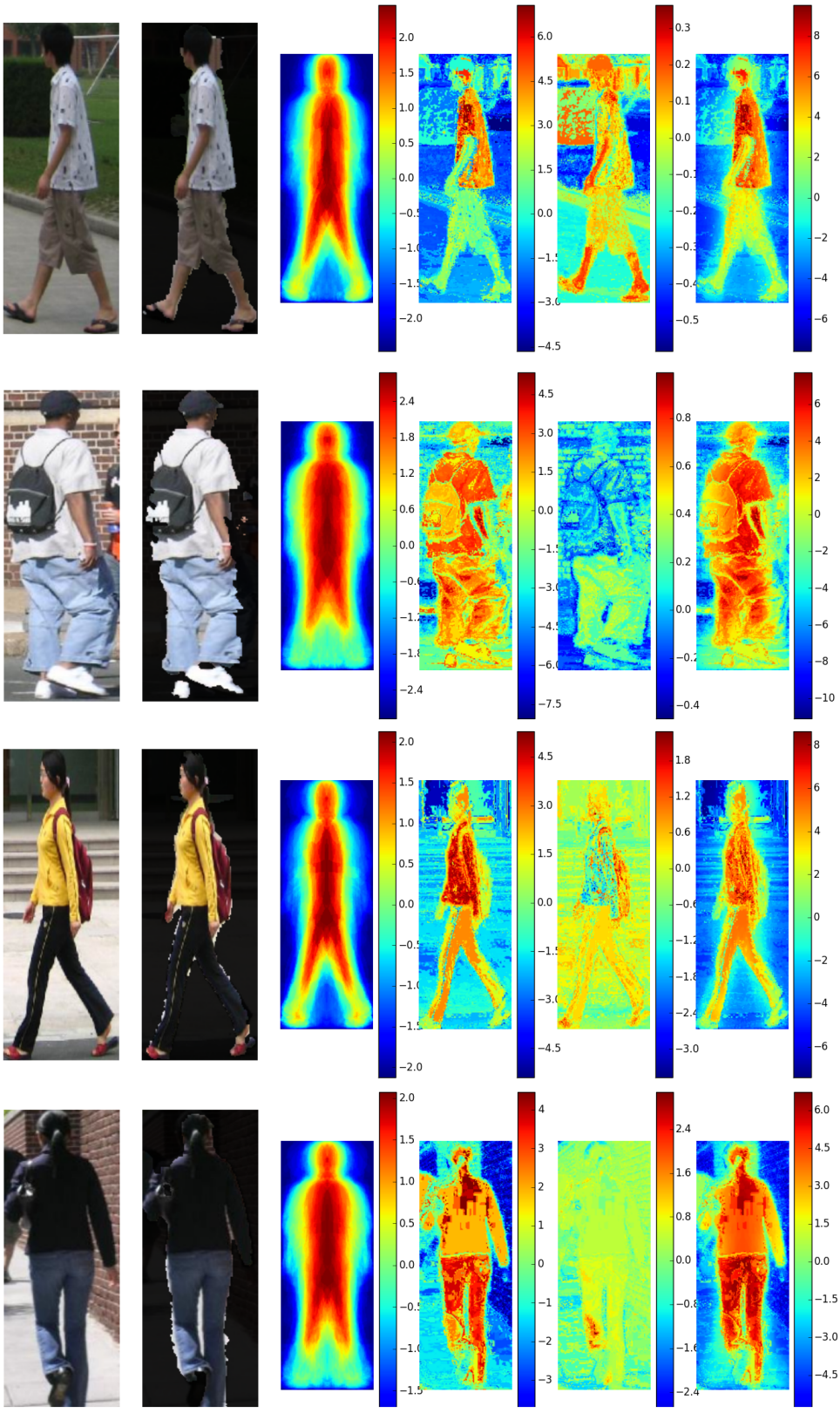

Figure 2. Column 1: test image. Column 2: output of my algorithm. Columns 3-5: unary potentials (foreground minus background) induced by $\Omega_1, \Omega_2, \Omega_3$. Column 6: Full unary potential (foreground minus background)