Learning object boundary detection from motion data

Michael G. Ross and Leslie Pack Kaelbling
M.I.T. Artificial Intelligence Lab

Abstract

A significant barrier to applying the techniques of machine learning to the domain of object boundary detection is the need to obtain a large database of correctly labeled examples. Inspired by developmental psychology, this paper proposes that boundary detection can be learned from the output of a motion tracking algorithm that separates moving objects from their static surroundings. Motion segmentation solves the database problem by providing cheap, unlimited, labeled training data. A probabilistic model of the textural and shape properties of object boundaries can be trained from this data and then used to efficiently detect boundaries in novel images via loopy belief propagation.

I. Introduction

Psychological studies by Spelke et al. [13] indicate that infants use motion as their primary mechanism for grouping visual perceptions into objects. The ability to detect objects via two-dimensional spatial cues, such as color, texture, or shape, seems to develop later. The discovery of static object boundaries, commonly referred to as image segmentation in computer vision literature, is a complex and ill-defined process. Typical segmentation algorithms rely on optimizing statistical criteria derived from information theory or Gestalt psychology, but this psychological evidence suggests a new approach, grounded in motion segmentation and machine learning.

The developmental evidence suggests that humans might use their initial knowledge of motion segmentation to learn image segmentation. The most obvious problem in applying machine learning techniques to image segmentation is the need for a large set of pre-segmented data for training. One approach to this problem is to produce a large human-labeled segmentation database [9], but this is inherently expensive and in many images the identity of the object boundaries is unclear, even to a human. Are the drawers of a desk separate objects, or parts of the whole? Does detecting these types of boundaries diminish a model's ability to make more important distinctions, such as separating the desk from the rug it sits on?

Using motion information removes this difficulty. Just as infants join visual elements together by their common motion, a video camera and computer can automatically distinguish between moving objects and their immediate surroundings and provide a learning algorithm with cheap, unlimited training data. Furthermore, the training set will only contain object boundaries, so the model will not waste its explanatory power on the uncertain subdivision of an object into parts. With appropriate data, machine learning can capture the necessary contextual and environmental information for a particular segmentation task. A learned boundary detection algorithm can adapt and optimize its performance for different situations without the need for extensive manual tuning.

We have created a new object boundary detection algorithm that is trained on motion segmentations output by an algorithm developed by Stauffer and Grimson [14]. The data is used to construct a probabilistic model that captures information about the spatial properties of the observed objects. After training, a loopy belief propagation algorithm developed by Freeman et al. [5] can efficiently find boundaries in novel static images. This work contributes a solution to the segmentation database problem, a low-dimensional, discrete object edge representation, and the ability to construct high dimensional object boundaries from noisy, low resolution data.

II. Related Work

Image segmentation algorithms are based on a variety of models. Some methods, such as Felzenszwalb and Huttenlocher's algorithm [3] are based on local models of texture and region size, while Shi and Malik's normalized cuts method [12] makes globally optimal divisions based on a matrix of similarity measures. Most similar to our work is Geman and Geman's image restoration algorithm [6], which uses two linked Markov random fields (MRFs) to denoise images. One of these fields, the line process, divides images into regions based on local image gradients and contour properties, such as edge continuity. Our model is very similar to the line process, but it is learned from data, while their model was hand-crafted. Poggio et al. [10] described the use of MRFs to combine different image features.

Recent work in learning segmentation and edge detection include Feng et al.'s work, which combined belief and neural network techniques [4]. This work is closer to region or texture modeling than pure segmentation: their goal is to apply a set of predetermined labels (e.g. sky, vegetation) to images. Konishi et al. [7] have investigated the statistical optimality of common local edge detectors. Both methods rely on manually segmented training data, requiring a time consuming process that may produce subjective results. Weber et al. [17] perform unsupervised learning of object class models by assuming that the class examples are the most prevalent element in their image set. In many ways, this is the opposite of our approach to data extraction, in which we assume that the object samples are the novel, dynamic elements in a scene.

Borenstein and Ullman have developed a model of class-specific segmentation that learns to perform figure-ground

segmentations for a particular class of objects by building a database of fragments that can be assembled like puzzle pieces [1]. They hypothesize that motion could be a source of training data for their algorithm, which combines segmentation and classification.

The use of belief propagation to detect and reinforce image contours is very similar to the work of Shashua and Ullman [11], which described a hand-built saliency network that combined incomplete contours to minimize an error function.

III. Object Boundary Model

Our object boundary model is inspired by Freeman et al.'s work on learning super-resolution [5]. We model the object edges as an MRF with two sets of variables: the visible "signal" nodes representing image data, and the hidden "scene" nodes that represent the underlying object edges. In the following description, the possibility that no edge is present at a location is included among the set of edge scenes.

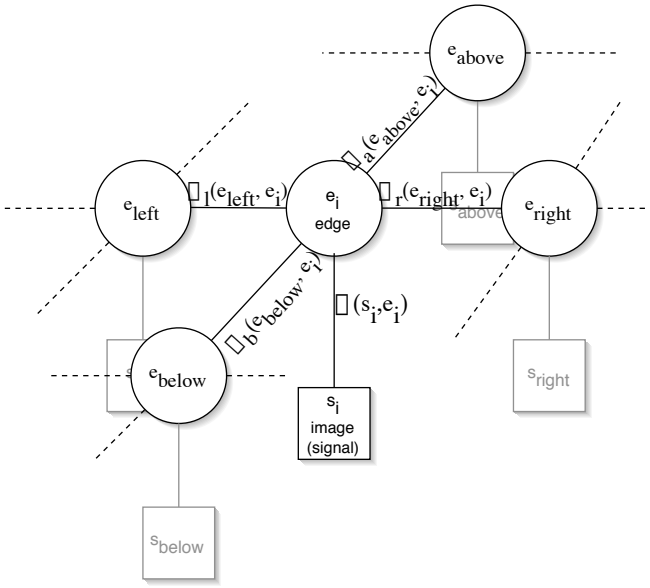

Fig. 1. Each edge node in our Markov random field is attached to a visible data input node and to the edge nodes for neighboring image locations.

The MRF model (Figure 1) posits that the probability of an edge assignment e_i at location i is dependent on its compatibility with the local image data s_i ($\phi(s_i, e_i)$) and its compatibility with the assignments to the neighboring scene nodes ($\psi_l(e_{\text{left}}, e_i)$, $\psi_r(e_{\text{right}}, e_i)$, etc.). The value of an edge node represents a 5 pixel by 5 pixel segment of an object boundary (or the absence of a boundary) at a particular image location. Its associated signal node represents the value of the convolution of several image filters at that location. The image areas covered by neighboring nodes are adjacent, but non-overlapping.

If N is the set of neighbors (i, j), and $r(i, j)$ is their neighboring relationship, the joint probability of an assignment $(\bar{s}, \bar{e})$ to the nodes is

$$\Pr(\bar{s}, \bar{e}) = Z^{-1} \prod_i \phi(s_i, e_i) \prod_N \psi_{r(i,j)}(e_i, e_j) \quad (1)$$

$$Z = \sum_{\bar{s}', \bar{e}'} \prod_i \phi(s'_i, e'_i) \prod_N \psi_{r(i,j)}(e'_i, e'_j). \quad (2)$$

Z is the partition function, which normalizes the probability, but whose computation is intractable for all but the smallest examples.

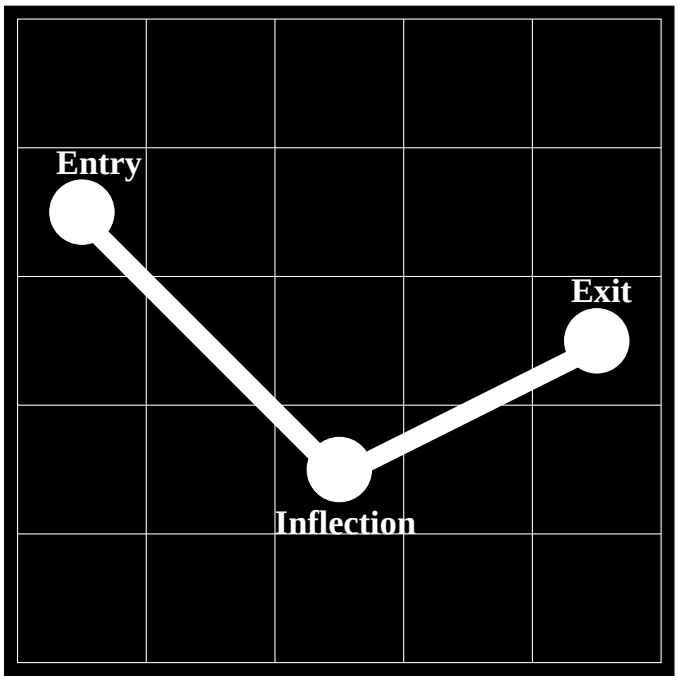

Fig. 2. The edge scene values are represented by three parameters. Each edge enters a scene patch at some border pixel, may change direction at an inflection point, and exits at another border pixel.

In the general case, there is no known closed-form solution for specifying the compatibility functions associated with particular marginal probabilities in an MRF. Iterative proportional fitting (IPF) is a gradient descent method that repeatedly adjusts compatibility functions to match the network's marginal probabilities to match an empirically observed set. Unfortunately, IPF requires us to perform inference on the MRF after each descent, which is prohibitively expensive. Instead, we can substitute belief propagation for the exact inference step. An analysis of the resulting algorithm in the context of Wainwright's tree reparameterization [16] view of belief propagation reveals that it has a simple fixed point[15]. For an edge node i with signal node i_s and neighbors $n = l, r, a, b$:

$$\phi(s, e) = \Pr(i_s = s, i = e)$$

and

$$\psi_n(e_n, e) = \frac{\Pr(n = e_n, i = e)}{\Pr(n = e_n) \Pr(i = e)}.$$

These compatibility settings are intuitively sound because they give high compatibilities to pairs that co-occur frequently,

and they exactly correspond to Freeman et al.'s conditional probability message passing algorithm [5].

In Section IV, we will demonstrate that it is easy to generate training data, so it is a simple matter to estimate any discrete probability function by frequency counting. Therefore, we discretize filter responses by counting them in 1000-bin histograms. The set of potential edges is already discrete, but it is too large to handle via simple counting. A 5x5 binary image can take on 2^{25} possible values. Even if we discard images that could never represent an edge fragment, such as an all-white image, noise and natural variability still leaves us with tens of thousands of edge images. Therefore, we have developed a simple edge parameterization (Figure 2) that reduces the space to 2717 possible edges and provides excellent coverage of the examples in our training set. Each edge is represented by the boundary pixel at which it enters the 5x5 patch, a possible interior inflection point, and a patch boundary exit point. The "empty" edge, representing the absence of a boundary, is included as a special case. During training, each scene value is represented by its best-fit parameterized edge.

IV. Training Algorithm

Just as simple neurons can detect motion due to their tendency to habituate to static input, computer algorithms can detect motion by background subtraction. By modeling the values observed at every image location across a video sequence, areas containing moving objects are highlighted as outliers. In this work we used Stauffer and Grimson's background subtraction algorithm [14], which models every image pixel with a small set of Gaussian distributions. This algorithm is easy to compute and is robust to non-motion changes, such as lighting variations, that we wish to discard. The background subtracter works on a stream of images; for each image it returns a binary image that labels every pixel as either foreground (moving) or background (static) (Figure 3).

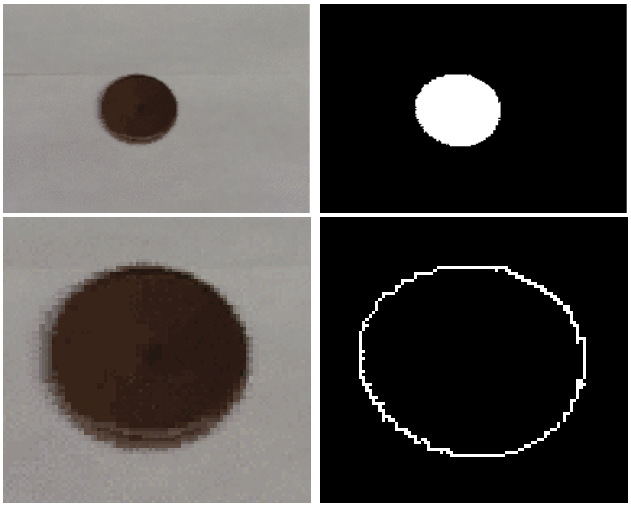

Fig. 3. Top row: The background subtraction algorithm learns the background colors at each pixel and returns a binary image indicating the location of the moving object. Bottom row: The moving object is cropped out of the image and the binary image is processed to produce the object edge image.

Motion can only give us data about the object edges immediately around the moving object. Therefore, we disregard all of the original image and the binary foreground-background image, except the areas containing the moving object and its immediate surroundings. Given the cropped foreground-background image, we scan across all rows and columns and label every location at which there is a transition from foreground pixels to background pixels as an edge. The output is a new binary image of edge and non-edge pixels. Our goal is to learn a model that will generate an equivalent object edge image from a novel, static image. Each image and edge image pair provide a complete set of assignments for an MRF object boundary model of the image.

During training, we need to construct representations of the ϕ and ψ functions. We will do this by learning three sets of probability distribution functions from the training data, which can be used to compute the compatibility values described in Section III using Bayes' rule and marginalization. The edge pixels at any location are represented by their best-fit edge in the parameterized edge model (Figure 2). Learning $\Pr(e)$, $\Pr(s|e)$, and the conditional probability functions $\Pr(e_n|e)$ for each neighboring relation (above, below, left, right) will provide the necessary information. All the probabilities are discrete, and data is plentiful, so it is possible to learn them by storing the value frequencies.

The algorithm must combine the data from multiple filters into a single $\Pr(s|e)$ value. Experience shows that no single filter will be an adequate edge detector. In some simple images, local horizontal and vertical image gradients respond strongly to object boundaries, but in highly textured examples they might respond more strongly to non-edge regions. One solution is to probabilistically combine many features. The underlying signal for each scene is an n-tuple of the values of a set of filters at a particular location. We need to represent the joint probability distribution $\Pr(s^1, s^2, \ldots, s^n|e)$ for each possible edge. Incorporating more features requires exponentially more data to estimate, and memory to store, the resulting model. Therefore, we make the naive Bayes assumption that the features are conditionally independent given the underlying edge and approximate the joint likelihood as $\Pr(s^1|e) \cdot \Pr(s^2|e) \cdot \ldots \cdot \Pr(s^n|e)$. This assumption is clearly incorrect, but the results it gives are still useful in practice. In the future, we hope to employ new representations that will better approximate the full joint probability of the features. Konishi et al. [8] have discovered an adaptive histogramming algorithm that efficiently combines edge detection features.

If the training data were noiseless, from a perfect background subtraction algorithm, the learning algorithm would only experience closed contours, and for any neighboring pair of candidate edges whose endpoints did not match the compatibility $\psi_n(e_n, e)$ would be zero. Unfortunately, this is not the case, so we encourage the formation of closed contours during inference by setting the compatibilities for non-matching neighbors to be nearly zero. Setting the compatibilities to exactly zero would violate the MRF definition, which forbids zero probability states. This produces cleaner contours

and fewer spurious edges, but does not completely rule out incomplete contours, because our edge parameterization does not allow multiple contours to combine via T-junctions.

V. Inference Algorithm

There is no known efficient algorithm for exact inference on a Markov random field, so we employ the belief propagation algorithm, a speedy approximation that works well in practice. Once the model is trained, belief propagation can compute an approximate maximum *a posteriori* (MAP) estimate of the object edges present in a static scene. Belief propagation produces exact MAP estimates in loopless MRFs [19][18]. Our network has loops, but belief propagation still works well in practice [5].

The belief propagation algorithm, as described by Freeman et al., begins by selecting a set of locally likely candidate edges at each location. In our implementation, we first visit each edge node i and select the empty edge and the $N-1$ edges (where we have experimented with $20 < N < 100$) with the largest $\Pr(s_i, e_i)$. Because the edge candidates at a node have only been selected based on local information, it is possible that the node may have no assignments compatible with some of the potential values of its neighbors. Therefore, we visit each node a second time, and add additional scenes so that the node can continue any edge that enters and exits it from its neighbors.

On every iteration of the algorithm, every node is visited and its messages are updated. A node i can be described by the signal associated with it (s_i), a set of candidate scenes (E_i), a set of neighbors (N_i), and an array of messages from each neighbor indexed by scene ($m_{i \leftarrow j}$, $j \in N_i$). All messages are initialized to 1. The index $r(i,j)$ indicates the position of j relative to i (left, right, above, below). On every iteration, we simultaneously update the messages at each node by the following equation:

$$m_{i \leftarrow j}(e \in E_i) = \max_{e_j \in E_j} \phi(s_j, e_j)\psi_{r(i,j)}(e_j, e) \prod_{k \in \{N_j \setminus i\}} m_{j \leftarrow k}(e_j). \quad (3)$$

Each message $m_{i \leftarrow j}(e)$ represents the compatibility between node i having assignment e and the information from its neighboring node j. The message is updated by maximizing a function over the neighboring assignments which combines their fit to the local data at j and their match to information from the other neighbors of j. After sufficient propagation (convergence is not guaranteed in loopy networks), we can approximate the MAP estimate for the MRF by computing

$$\mathrm{MAP}_i = \arg\max_{e \in E_i} \phi(s_i, e) \prod_{j \in N_i} m_{i \leftarrow j}(e) \quad (4)$$

at each node i. This selects the edge e that is maximally compatible with local evidence and the information propagated from the remainder of the graph.

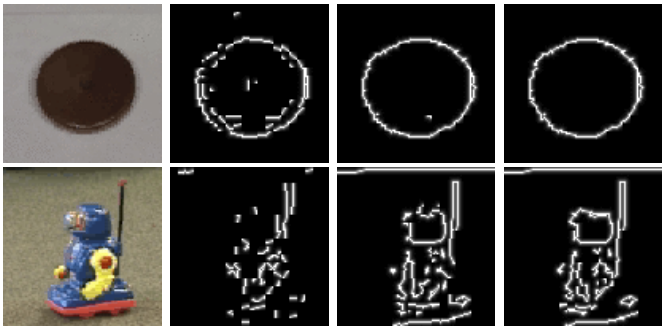

Fig. 4. Top: MAP estimates after 0, 5, and 10 iterations on a sample disc image. Bottom: MAP estimates after 0, 10 and 20 iterations on a sample robot image.

VI. Results

We trained models on three video sequences: a dark disc moving against a white background, a toy robot traveling across a highly textured carpet, and cars driving along a highway. The first two sequences contained approximately 1200 frames, and the third sequence contained 7000 frames. In each case the first 200 frames were used to initialize the background subtraction algorithm and were not included in the training set. The detection results presented are all drawn from these discarded video frames or from other non-training sets.

Different numbers of candidate scenes and iterations were required for each result, depending on the complexity of the object and the quality underlying data. Because we select an initial set of N possible values at each edge node, and then augment them with extra possibilities to allow for contour completion, each node in a particular MRF may consider a different number of possible edges. Disc results used N=20 candidates and 10 belief propagation iterations. The robot results used 100 candidates and 20 iterations, due to the robot's irregular shape and the "noise" provided by the textured carpet. The cars required 40 candidates and 20 iterations.

Although we have experimented with texture-sensitive filters, such as Gabor functions, all of the results presented here were computed using four local gradient operators, oriented to 0, 45, 90, and 135 degrees, as the input signals. These filters were computed on the grayscale image values; color was not used. We trained a four-neighbor model in which each node is connected to its first-order (above, below, left, and right) neighbors.

In a typical run, the initial MAP estimate, made before belief propagation occurs, contains approximate object edges, which are improved by enforcing local edge continuity and learned shape information. Figure 4 demonstrates the progress of belief propagation on samples from the disc and robot sequences.

Figure 5 displays a sample result from each trained model. Unsurprisingly, the simple disc case was the most successful, due to its highly regular shape and the strong spatial derivatives along its boundaries. The robot was the most difficult, given its irregular shape and the fact that the carpet produced spurious image gradients that the model had to learn to ignore. The

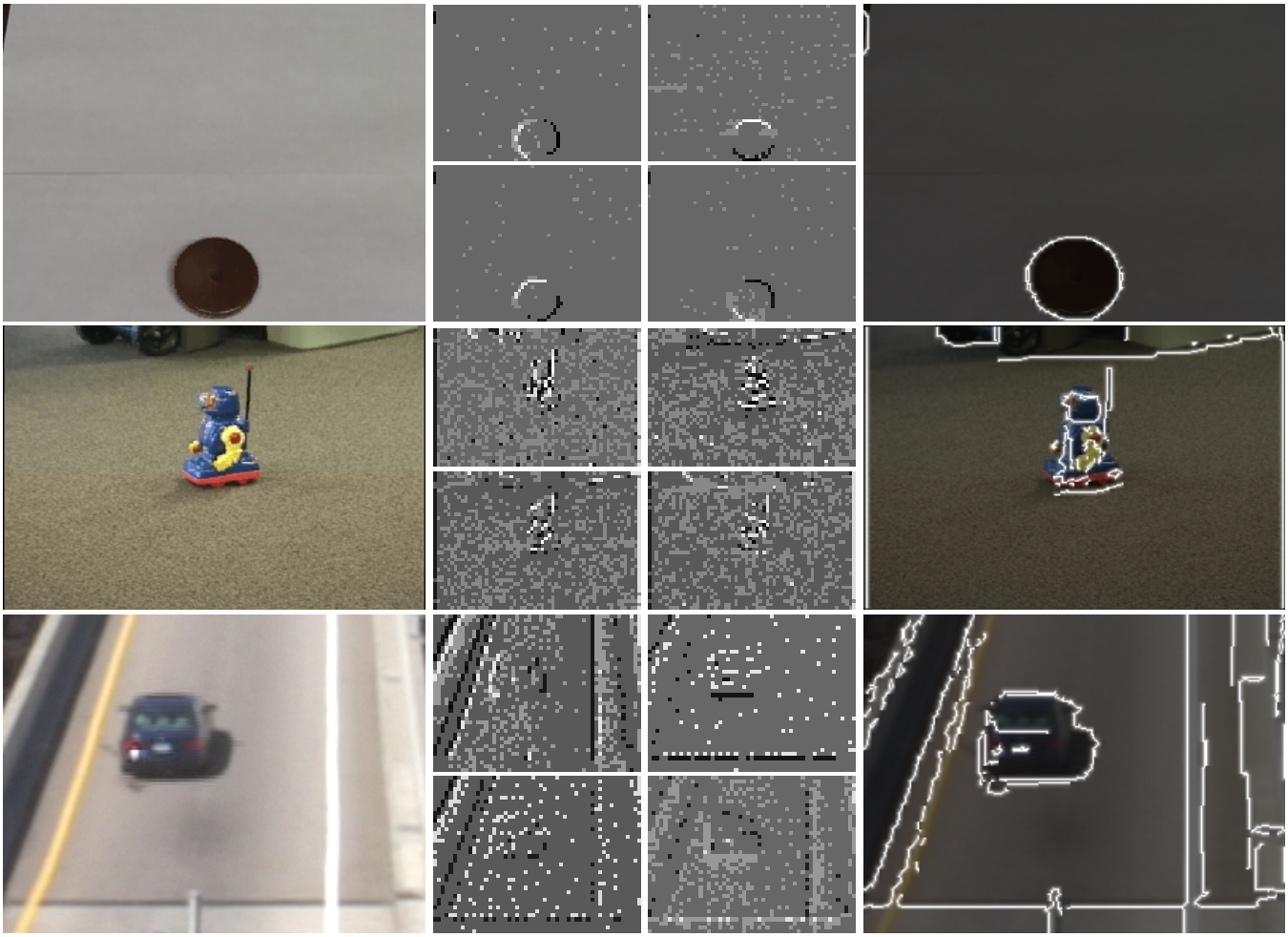

Fig. 5. Sample results from three different data sets. Each row shows, from left to right, an image outside the training set, the contrast-enhanced outputs of the four coarse derivative filters, and the result of using our MRF model and belief propagation to find object edges in the scene. The disc border was detected with approximately 20 candidates per location and 10 rounds of belief propagation. The robot required approximately 100 candidates per location and 20 rounds, and the car used 40 candidates and 20 rounds.

car was very successful, especially considering that the car shadows were included as moving objects during training. The model segmented the car and its shadow from the road, and also detected other object and non-object edges in the image.

In both the car and robot examples, non-object edges, such as the lines on the road and internal color changes on the robot, were detected. In the robot, these extra edges apparently prevented some of the robot object contours from closing properly. Because our edge model only allows one entry and one exit point in each patch, it is impossible to represent contour intersections properly. This clearly needs to be addressed in the future. In the case of the car output, a more sophisticated model of shape would be necessary to eliminate the road lines as potential objects, if we desired to do so. It is generally accepted, however, that some degree of oversegmentation is preferable to undersegmentation.

In all three cases, it is important to note that the contours were detected remarkably well given the sparse, imperfect information that was available. Next to each image in Figure 5 is the four input signals used to produce the object edge outputs. Gradients were only computed at the image locations associated with the center of each edge node's patch, so an image of height h and width w is represented only by $\frac{hw}{25}$ inputs in each filtered image, and these filters were then combined suboptimally by the naive Bayes assumption. The shape model that inter-relates neighboring edge patterns provides much of the output accuracy.

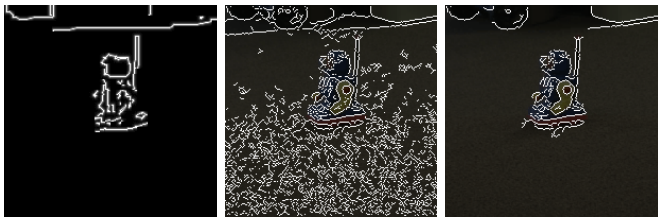

Fig. 6. From left to right: boundary detection with our algorithm, with the default Canny edge detector, and with a hand-tuned Canny edge detector.

Figure 6 compares our performance on the robot image to the output of the Canny edge detector [2] included in the

Matlab Image Toolbox. Our detector significantly outperforms the results using the default threshold and smoothing settings, and approaches the output of the Canny detector with manually chosen parameters (threshold = 1, sigma = 0.2). Our algorithm has learned many of the boundary rules that are hand-coded into the Canny algorithm, and is able to adapt itself to the requirements of the visual environment without the need for manual parameter tuning. The Canny algorithm also has the advantage of higher resolution gradient information than that available to our algorithm.

Figure 7 demonstrates that the model trained on the car data sequence can be successfully applied to other similar situations. The images in this test set come from another road which was observed in the same wide-angle video as the training data. The model does a good job at detecting the car boundaries. The errors arise from extremely low image gradients at the borders of some of the cars, and the incompatibilities caused by the intersection of car and road contours.

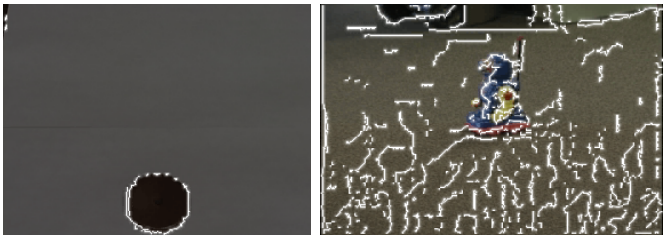

Fig. 8. Results of running the robot model on a disc image, and vice-versa.

Although the MRF model is very simple, it is clear that it learns a great deal of contextual information during training. Figure 8 demonstrates the results of applying the disc-trained model to a sample from the robot sequence and the robot model to a disc image. In the first case, the disc model erroneously detects a number of long contours along the carpet. In the disc training sequence, nearly all the image gradients were associated with object edges, so the model does not know to avoid the weak, random carpet gradients. On the other hand, the robot model appears to detect the boundaries of the disc adequately, but it lacks the shape experience to complete the simple circular contour.

The algorithm is very efficient, completing all the calculations for 20 iterations of belief propagation on a 150x150 pixel image with 40 candidates at each location in less than 30 seconds on a dual 1.4 GHz Pentium 3 machine in our Java implementation. This is due to the efficiency of belief propagation, the preselection of a small set of potential scenes at each edge node, and the relatively large size of the edge patches, which allow us to cover the image with only 900 edge nodes.

VII. Future Work

Our future work will focus on improving the approximations in our model and in capturing more shape and feature information. A clear area for improvement is the use of the conditional independence assumption in computing the scene likelihood function. Our experience with larger, more complex feature sets suggests that their distributions are not well approximated by this assumption and will require better modeling techniques.

We also need to capture more information about the shapes of objects in the training set and expand the features used to detect potential edges. It is common to acquire shape information with a multiresolution model, but our experiments indicate that these models do not provide better results than uniresolution models in belief propagation. We need to explore other methods of capturing and enforcing high-level boundary properties, such as closed contours and global shape. Expanding our feature set to include color and texture-sensitive features such as Gabor filters should also help to improve our results.

Acknowledgments

This research was supported by the Singapore-MIT Alliance, the National Science Foundation, and the Office of Naval Research.

The authors thank Bill Freeman for continuing input and assistance in this research. They also wish to especially thank Yair Weiss and Martin Wainwright for discussions about belief propagation, Chris Stauffer for his background subtraction software and the cars data set, Kinh Tieu for suggestions about shape modeling, and Kevin Murphy for comments on the draft. Finally, we thank Huizhen Yu, Erik Miller, John Winn, and John Fisher for many discussions.

References

[1] E. Borenstein and S. Ullman. Class-specific, top-down segmentation. In *European Conference on Computer Vision*, 2002.

[2] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6), November 1986.

[3] P. Felzenszwalb and D. Huttenlocher. Efficiently computing a good segmentation. In *DARPA Image Understanding Workshop*, 1998.

[4] X. Feng, C.K.I. Williams, and S.N. Felderhof. Combining belief networks and neural networks for scene segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(4), 2002.

[5] W.T. Freeman, E.C. Pasztor, and O.T. Carmichael. Learning low-level vision. *International Journal of Computer Vision*, 40(1), 2000.

[6] S. Geman and D. Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(6), November 1984.

[7] S.M. Konishi, J.M. Coughlan, A.L. Yuille, and S.C. Zhu. Fundamental bounds on edge detection: An information theoretic evaluation of different edge cues. In *Computer Vision and Pattern Recognition*, 1999.

[8] S.M. Konishi, A.L. Yuille, J.M. Coughlan, and Song Chun Zhu. Statistical edge detection: Learning and evaluating edge cues. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, In Press, 2002.

[9] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *International Conference on Computer Vision*, 2001.

[10] T. Poggio, E.B. Gamble, and J.J. Little. Parallel integration of vision modules. *Science*, 242, 1988.

[11] A. Shashua and S. Ullman. Structural saliency: The detection of globally salient structures using a locally connected network. In *International Conference on Computer Vision*, 1988.

[12] J. Shi and J. Malik. Normalized cuts and image segmentation. In *Computer Vision and Pattern Recognition*, June 1997.

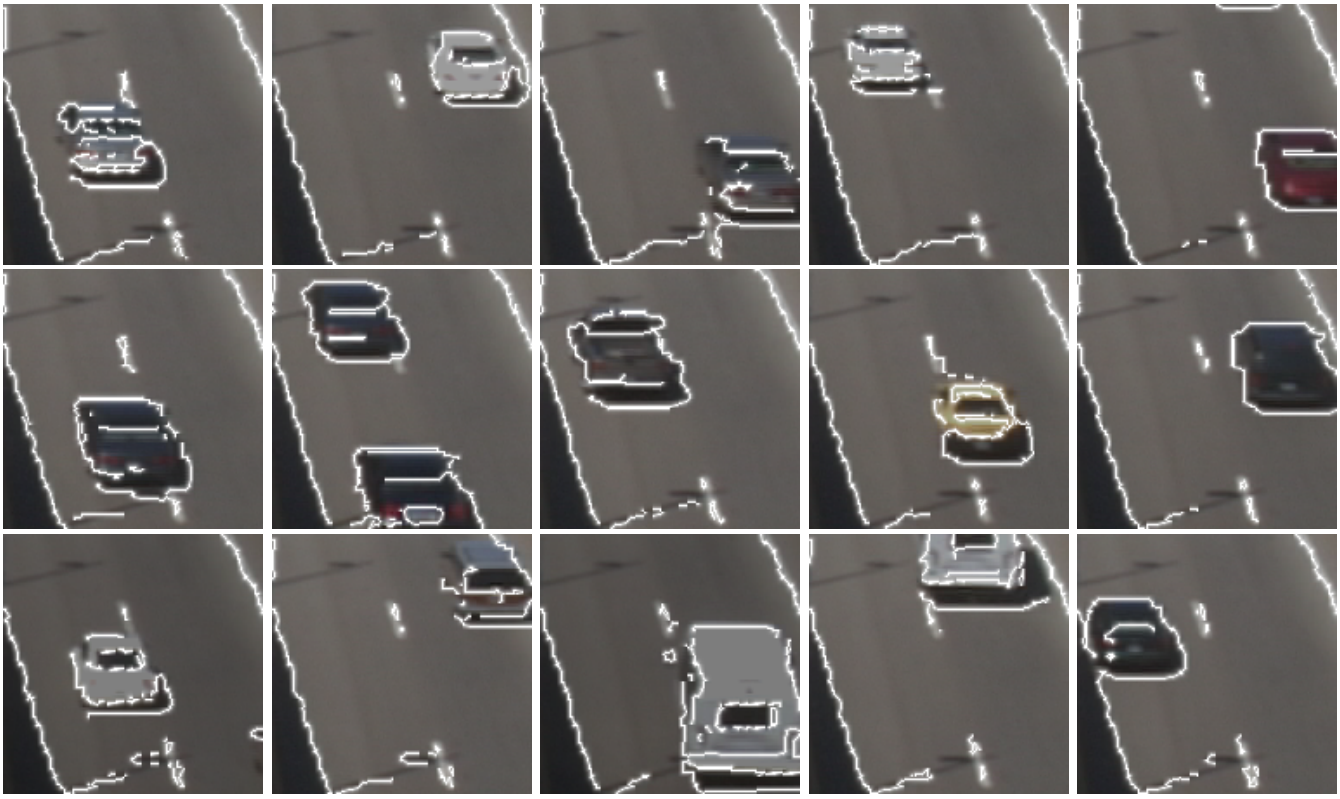

Fig. 7. These results were inferred by the car model on images drawn from another road's traffic. The results required 40 candidates per node and 20 iterations of belief propagation. The model does a good job of extracting the car boundaries, except where the car colors blend into the road too much, or where there is competition between car contours and strong road contours.

[13] E.S. Spelke, P. Vishton, and C. von Hofsten. Object perception, object-directed action, and physical knowledge in infancy. In *The Cognitive Neurosciences*, pages 165–179. The MIT Press, 1994.

[14] C. Stauffer and W.E.L. Grimson. Adaptive background mixture models for real-time tracking. In *Computer Vision and Pattern Recognition*, 1999.

[15] M.J. Wainwright. Personal communication, 2002.

[16] M.J. Wainwright, T. Jaakkola, and A.S. Willsky. Tree-based reparameterization for approximate estimation on loopy graphs. In *Neural Information Processing Systems*, 2001.

[17] M. Weber, M. Welling, and P. Perona. Unsupervised learning of models for recognition. In *European Conference on Computer Vision*, 2000.

[18] Y. Weiss. Belief propagation and revision in networks with loops. Technical Report 1616, MIT Artificial Intelligence Laboratory, November 1997.

[19] Y. Weiss. Interpreting images by propagating bayesian beliefs. In *Advances in Neural Information Processing Systems*, 1997.