Computer Science and Artificial Intelligence Laboratory

Technical Report

MIT-CSAIL-TR-2003-018
AIM-2003-022

September 8, 2003

Learning object segmentation from video data

Michael G. Ross and Leslie Pack Kaelbling

massachusetts institute of technology, cambridge, ma 02139 usa — www.csail.mit.edu

Abstract

This memo describes the initial results of a project to create a self-supervised algorithm for learning object segmentation from video data. Developmental psychology and computational experience have demonstrated that the motion segmentation of objects is a simpler, more primitive process than the detection of object boundaries by static image cues. Therefore, motion information provides a plausible supervision signal for learning the static boundary detection task and for evaluating performance on a test set. A video camera and previously developed background subtraction algorithms can automatically produce a large database of motion-segmented images for minimal cost. The purpose of this work is to use the information in such a database to learn how to detect the object boundaries in novel images using static information, such as color, texture, and shape.

This work was funded in part by the Office of Naval Research contract #N00014-00-1-0298, in part by the Singapore-MIT Alliance agreement of 11/6/98, and in part by a National Science Foundation Graduate Student Fellowship.

1 Introduction

This work addresses the problem of object segmentation, a subproblem of the image segmentation problem described in the computer vision literature. Image segmentation is the discovery of salient regions in static images, traditionally by optimizing functions of the image data. Image segmentation research continues the work of the Gestalt psychologists, who first described the principles of organization and grouping in human vision [13]. More recently, segmentation algorithms have been derived from graph cut metrics [18] and information criteria [27]. Despite decades of research, it is unclear which approach is optimal because their definitions of region saliency are incompatible. Therefore, the algorithms are frequently judged by qualitative comparisons of their results.

Object segmentation is the process of detecting regions that correspond to input entities for higher-level intelligent systems. A robot navigation system, for example, requires information about walls, doors, and chairs, not the values of millions of pixels. These useful agglomerations are objects, and dividing a single static image according to the object boundaries it contains is the goal of object segmentation.

In order to be useful to external systems, objects must represent properties that are useful to such systems. While higher-level visual systems, such as object classifiers, might be primarily concerned with grouping pixels according to similarity in color or texture, a robot manipulator requires objects that will not disintegrate when they are picked up. In this work, we will consider objects defined by the property of common motion. A group of elements will be considered an object if and only if they are undergoing coherent motion in the world, or if they would undergo coherent motion in response to typical forces.

Why is this definition generally useful? Consider an agent in a world without common-motion objects, a Sandworld in which no particle was attached to its neighbors. Interaction with this world is difficult, because the motion of every element must be separately considered. A world with common-motion objects, on the other hand, groups elements into motion-dependence clusters that enable much more powerful modeling. Optical flow measurements, for example, can use object grouping to overcome the aperture problem. Operating at the level of objects, rather than the level of particles, simplifies planning, prediction, and navigation based on visual input. All of these advantages derive from the common motion assumption, and that is why it is the basis of our definition of object.

Apart from this utility argument, there is evidence to suggest that humans' notions of image segmentation derive from predictions of common motion. Psychological studies by Spelke et al. [19] 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. Sample human segmentations in the Martin et al. database [12] suggest that common-motion boundaries are more commonly selected by subjects than other textural or color boundaries.

This developmental evidence suggests that humans 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 [12], but this is inherently expensive, and in many images the identity of the salient boundaries is unclear, even to a human. Are the windows of a car separate regions, 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 car from the road?

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.

2 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 [18] makes globally optimal divisions based on a matrix of similarity measures. Most similar to our work is Geman and Geman's image restoration algorithm [7], 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 similar to the line process, but it is learned from data and captures more shape information. Poggio et al. [15] 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. [9] have investigated the statistical optimality of common local edge detectors and Martin et al. [11] improved on standard edge detectors by learning detector parameters from a human-labeled database. These methods rely on manually segmented training data, requiring a time-consuming process that may produce subjective results.

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. Hayman and Eklundh learn additional figure-ground segmentation cues from motion detection and prediction [8], but they are concerned with improving motion segmentation performance on video sequences by adding color and contrast cues, not with learning to perform the static segmentation task.

Fitzpatrick [5] developed an object-recognition system for Cog, a humanoid robot, that acquired examples by moving objects and observing them using background subtraction. Weber et al. [24] performed unsupervised learning of object class models by assuming that the class examples are the most prevalent element in their image set.

Our use of belief propagation to detect and reinforce image contours is very similar to the work of Shashua and Ullman [17], which described a hand-built saliency network that combined incomplete contours to minimize an error function.

3 Current progress

We have developed an initial learning and segmentation framework that has some of the properties we desire. The initial results demonstrate that it is possible to learn a useful model of object boundaries with a very simple shape and feature model. Future extensions and modifications of the current work, combined with a clearer understanding of the object segmentation problem should lead us toward models that better capture the problem and its solutions, and therefore provide improved, and more general, results.

We have created a new object-boundary detection algorithm that is trained on motion segmentations output by an algorithm developed by Stauffer and Grimson [20]. The data is used to construct a probabilistic model that captures information about the spatial properties of the observed objects. After training, belief propagation inference [14, 25] can efficiently find boundaries in novel static images. Our model is based on work by Freeman et al. [6] on training Markov random field models for vision problems. This work contributes a solution to the segmentation database problem, a low-dimensional, discrete object-edge representation, and the ability to construct high-resolution object boundaries from noisy, low-resolution data.

3.1 Object Boundary Model

The object boundary model is inspired by Freeman et al.'s work on learning super-resolution [6]. The boundary model is a Markov random field (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.

The MRF model (Figure 1) represents the probability distribution of object boundary edges given visible scene data. Every edge node, E_i, is connected to an image signal node, S_i, and to it's left, right, upper, and lower edge node neighbors. The value of an edge node represents a 5 pixel by 5 pixel segment of an object boundary (or the absence of a boundary), and its associated signal node represents the convolution of several image filters at the corresponding image location. Each image and edge node pair is responsible for a single 5x5 location in the image, and neighboring nodes are responsible for adjacent, but non-overlapping areas. The value e_i is an assignment to E_i, and a vector $\bar{e}$ is an assignment to every edge node; s_i and $\bar{s}$ are analogously defined for signal nodes.

After training, segmenting new images requires knowledge of the distribution $\Pr(\bar{e}|\bar{s}) = \Pr(\bar{s}|\bar{e})\Pr(\bar{e})/\Pr(\bar{s})$. Given an image, $\bar{s}$ is fixed, so the desired boundary is the $\bar{e}$ that

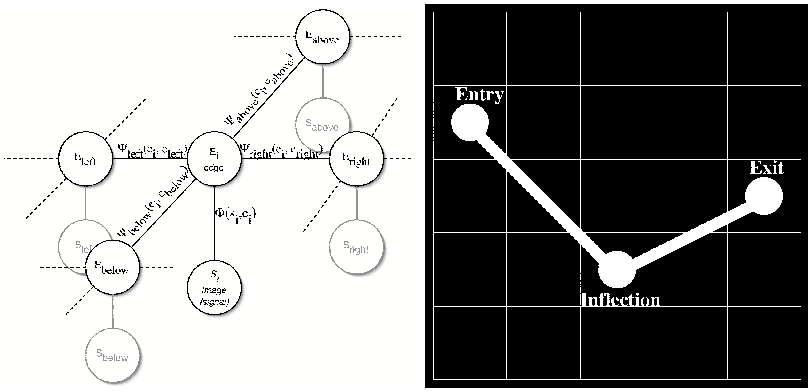

Figure 1: Each edge node in our Markov random field (left) is attached to a visible data input node and to the edge nodes for neighboring image locations. The edge scene values (right) are represented by three parameters. Each edge enters a scene patch at some border pixel, may change direction at an inflection point, and exits at another border pixel.

maximizes $\Pr(\bar{s}|\bar{e})\Pr(\bar{e}) = \Pr(\bar{e}, \bar{s})$. The MRF represents $\Pr(\bar{e}, \bar{s})$ with two sets of compatibility functions. The $\phi_i(s_i, e_i)$ functions represent the compatibility between pairs of signal values and edge values, and $\psi_{ij}(e_i, e_j)$ functions represent the compatibility between assignments to neighboring edge nodes. If N is the set of neighboring edge node indices,

$$\Pr(\bar{e}, \bar{s}) = Z^{-1} \prod_i \phi_i(s_i, e_i) \prod_{(i,j)\in N} \psi_{ij}(e_i, e_j)$$

where

$$Z = \sum_{\bar{s}', \bar{e}'} \prod_i \phi_i(s_i', e_i') \prod_{(i,j)\in N} \psi_{ij}(e_i', e_j') \; .$$

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

In general, there is no known closed-form solution for specifying compatibility functions that will produce 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. A tree reparameterization analysis [22] of this algorithm by Wainright and Sudderth reveals that it has a simple fixed point [23]:

$$\phi_i(s_i, e_i) = \Pr(s_i, e_i)$$

and

$$\psi_{ij}(e_i, e_j) = \frac{\Pr(e_i, e_j)}{\Pr(e_i)\Pr(e_j)} \; .$$

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 [6]. They also appear as approximate maximum likelihood parameter estimates in the tree reparameterization framework [21].

In Section 3.2, 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 discarding images that could never represent an edge fragment, such as an all-white image, noise and natural variability produce tens of thousands of edge images. Therefore, we have developed a simple edge parameterization (Figure 1) 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.

3.2 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. This work used Stauffer and Grimson's background subtraction algorithm [20], 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 2).

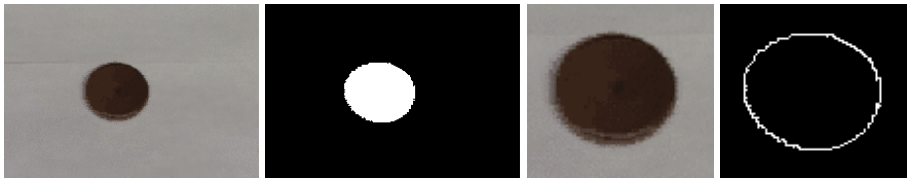

Figure 2: From left: The background subtraction algorithm learns the background colors at each pixel and returns a binary image indicating the location of the moving object. Then, the moving object is cropped out of the image and the binary image is processed to produce the object edge image.

Motion can only give us data about the object edges immediately around the moving object. Therefore, we discard 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.

The training algorithm constructs representations of the ϕ and ψ functions by learning three sets of probability distribution functions, which can be used to compute the compatibility values described in Section 3.1 using Bayes' rule. The edge pixels at any location are represented by their best-fit edge in the parameterized edge model (Figure 1). The model is translationally invariant, so all the ϕ functions are equal, and the ψ functions fall into four classes: $\psi_{\text{left}}(e_i, e_{\text{left}})$, $\psi_{\text{right}}(e_i, e_{\text{right}})$, $\psi_{\text{above}}(e_i, e_{\text{above}})$, and $\psi_{\text{below}}(e_i, e_{\text{below}})$. Learning $\Pr(e_i)$, $\Pr(s_i|e_i)$, and the conditional probability functions $\Pr(e_n|e_i)$ for each neighboring relation ($n = \text{left}, \text{right}, \text{above}, \text{below}$) will provide the necessary information to specify each compatibility function. All the probabilities are discrete, and data is plentiful, so it is possible to learn them by storing the value frequencies. The nodes in the model are assumed to be homogeneous, so only one set of distributions is learned.

The algorithm must combine the data from multiple filters into a single $\Pr(s_i|e_i)$ 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. The model requires the joint probability distribution $\Pr(s_i^1, s_i^2, \ldots, s_i^n|e_i)$ 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 $\prod_j \Pr(s_i^j|e_i)$. This assumption is clearly incorrect, but the results it gives are still useful in practice, especially in discriminative applications [16]. In the future, we hope to employ new representations that will better approximate the full joint probability of the features. Konishi et al. [10] 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 $\psi_n(e_n, e_i)$ 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.

3.3 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 [25]. Belief propagation produces exact MAP estimates in loopless MRFs [14, 26]. Our network has loops, but belief propagation still works well in practice [6].

As in the super-resolution algorithm described by Freeman et al., the edge-inference algorithm begins by selecting a set of locally likely candidate edges at each location. It first visits each edge node i and selects 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, the algorithm visits each node a second time, and adds 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 edge node is visited and its messages are updated. An edge node E_i can be described by the signal associated with it, s_i, a set of candidate scenes, $\mathrm{Cand}(i)$, a set of neighbors, $\mathrm{Neigh}(i)$, and an array of messages from each neighbor indexed by scene, $m_{i \leftarrow j}$ with $j \in \mathrm{Neigh}(i)$. 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:

$$m_{i \leftarrow j}(e_i \in \mathrm{Cand}(i)) = \max_{e_j \in \mathrm{Cand}(j)} \phi(\boldsymbol{s}_j, e_j) \psi_{r(i,j)}(e_j, e_i) \prod_{k \in \{\mathrm{Neigh}(j) \setminus i\}} m_{j \leftarrow k}(e_j).$$

Each message $m_{i \leftarrow j}(e_i)$ represents the compatibility between node E_i having assignment e_i and the information from its neighboring node E_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 E_j. After sufficient propagation (convergence is not guaranteed in loopy networks), the approximate MAP estimate for an MRF node is

$$\mathrm{MAP}_{E_i} = \arg \max_{e_i \in \mathrm{Cand}(i)} \phi(s_i, e_i) \prod_{j \in \mathrm{Neigh}(i)} m_{i \leftarrow j}(e_i)$$

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

3.4 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

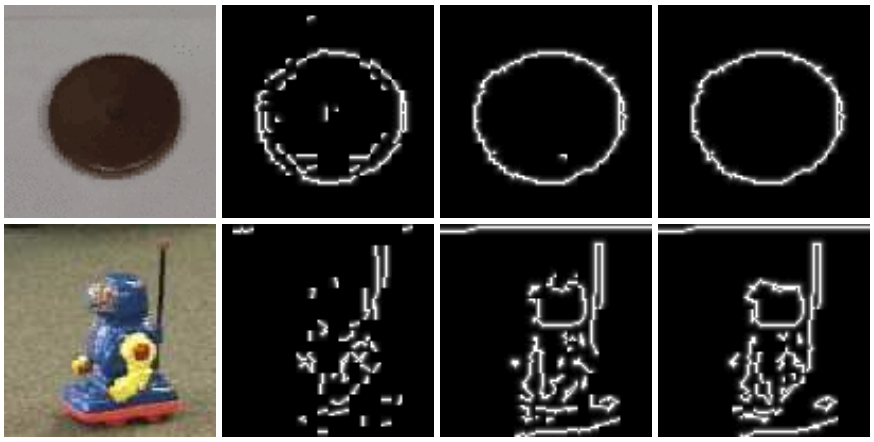

Figure 3: Top: MAP estimates after 0, 5, and 10 iterations on a sample disc image. Bottom: MAP estimates after 0, 10 and 20 iterations on a sample robot image.

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 of the underlying data. Because the algorithm selects an initial set of N possible values at each edge node, and then augments 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. The number of initial candidates and belief propagation steps were manually selected. Selecting larger values in each instance should produce equivalent results at an increased computational cost. In future work, we hope to determine these parameters automatically from the data and models.

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 3 demonstrates the progress of belief propagation on samples from the disc and robot sequences.

Figure 4 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

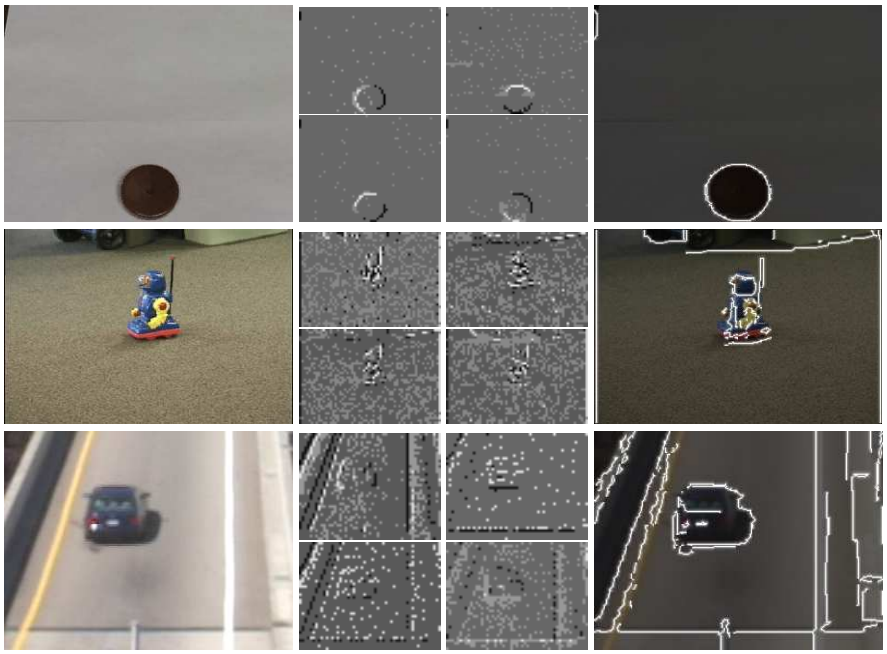

Figure 4: Sample results from three different data sets. Each row shows, from left to right, an image outside the training set, the contrast-enhanced outputs of the four coarse derivative filters, and the result of using our MRF model and belief propagation to find object edges in the scene.

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 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.

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 4 are 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.

Figure 5: From left to right: boundary detection with our algorithm, with the default Canny edge detector, and with a hand-tuned Canny edge detector.

Figure 5 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 6 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.

Although the MRF model is very simple, it is clear that it learns a great deal of contextual information during training. Figure 7 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.

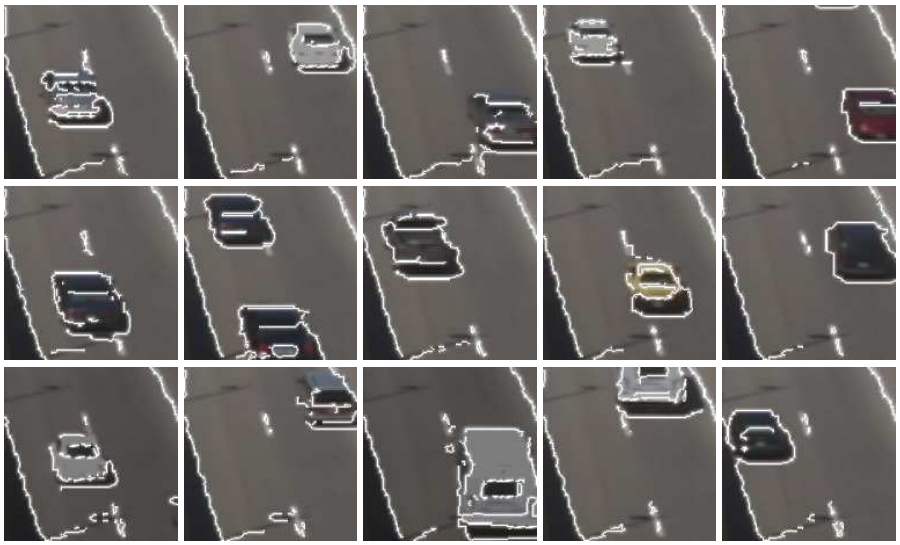

Figure 6: These results were inferred by the car model on images drawn from another road's traffic. The results required 40 candidates per node and 20 iterations of belief propagation. The model does a good job of extracting the car boundaries, except where the car colors blend into the road too much, or where there is competition between car contours and strong road contours.

4 Future Work

The work described is a strong starting point, but it is clearly incomplete in several respects. Our experimental results and a better understanding of the self-supervised object segmentation learning problem indicate that better image features and a more sophisticated approach to shape representation are promising areas for future work. Demonstrating progress on the segmentation problem requires a sensible metric, and we plan to report the match between inferred object boundaries and observed motion boundaries. Additionally, we wish to better account for the fact that our training data is partially labeled, in contrast to the completely labeled Berkeley database.

Acknowledgements

The authors greatly benefitted from the assistance and advice of William T. Freeman. Special thanks are awarded to Yair Weiss, Martin Wainwright, and Erik Sudderth for theoretical belief propagation assistance, to Kinh Tieu for a good shape-modeling suggestion, to Chris Stauffer for data and background subtraction software, and to Kevin Murphy for reading an earlier draft. Finally, they thank Huizhen Yu, Erik Miller, John Winn, and John Fisher for many discussions.

Figure 7: Results of running the robot model on a disc image, and vice-versa.

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] P. Fitzpatrick. *From First Contact to Close Encounters: A developmentally deep perceptual system for a humanoid robot.* PhD thesis, Masschusetts Institute of Technology, 2003.

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

[7] 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.

[8] E. Hayman and J.-O. Eklundh. Probabilistic and voting approaches to cue integration for figure-ground segmentation. In *European Conference on Computer Vision*, 2002.

[9] 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.

[10] 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*, 25(1), 2003.

[11] D. Martin, C. Fowlkes, and J. Malik. Learning to detect natural image boundaries using local brightness, color, and texture cues. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, In press, 2003.

[12] 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.

[13] S.E. Palmer. *Vision Science: Photons to Phenomenology*. The MIT Press, 1999.

[14] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.

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

[16] H. Schneiderman and T. Kanade. A statistical method for 3d object detection applied to faces and cars. In *International Conference on Image Processing*, 2000.

[17] 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.

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

[19] 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.

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

[21] M.J. Wainwrigh, T. Jaakkola, and A.S. Willsky. Tree-reweighted belief propagation and approximate ml estimation by pseudo-moment matching. In *Workshop on Artificial Intelligence and Statistics*, 2003.

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

[23] M.J. Wainwright and E. Sudderth. Personal communication, 2002.

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

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

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

[27] S.C. Zhu and A. Yuille. Region competition: Unifying snakes, region growing, and bayes/mdl for multi-band image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(9), 1996.