A Systematic Approach to Learning Object Segmentation from Motion

Michael G. Ross and Leslie Pack Kaelbling
{*mgross,lpk*}*@csail.mit.edu*
MIT Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
Cambridge, MA 02139, USA

Abstract

This paper 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. A video camera and a background subtraction algorithm can automatically produce a large database of motion-segmented images. 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.

1 Introduction

This work addresses the problem of object segmentation, the task of discovering the object boundaries present in a single, static image. Object segmentation is clearly related to the image segmentation task of discovering salient regions in images, a human visual ability first identified by Gestalt psychologists [10] and more recently the focus of computer vision research.

In order to apply the techniques of machine learning to the object segmentation problem, a reliable source of training data is required. Similar projects [9] have used a database of manually created segmentations, but collecting such a database is expensive and requires a subjective determination of which boundaries are important. On the other hand, motion is a reliable indicator of object boundaries in the real world, and it is possible to use background subtraction algorithms to locate the boundaries of moving objects in videos. Human infants' ability to segment objects by motion properties precedes their ability to segment them using cues such as shape, color, or texture [14]. Therefore, the algorithm described in this work learns to detect object boundaries in static images by training on the object segmentations indicated by motion in videos.

2 Related work

Recent results in learning segmentation and edge detection include Feng et al.'s work, which combined belief and neural network techniques [3]. 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. [8] and Martin et al. [9] surpass standard edge detectors by training on a human-labeled database.

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 [7], 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 [4] 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. [17] performed unsupervised learning of object class models by assuming that the class examples are the most prevalent element in their image set.

The use of belief propagation in this system is very similar to the work of Shashua and Ullman [13], which described a hand-built saliency network that combined incomplete contours to minimize an error function. Geman and Geman [6] first applied Markov random fields to image segmentation problems.

3 Object Boundary Model

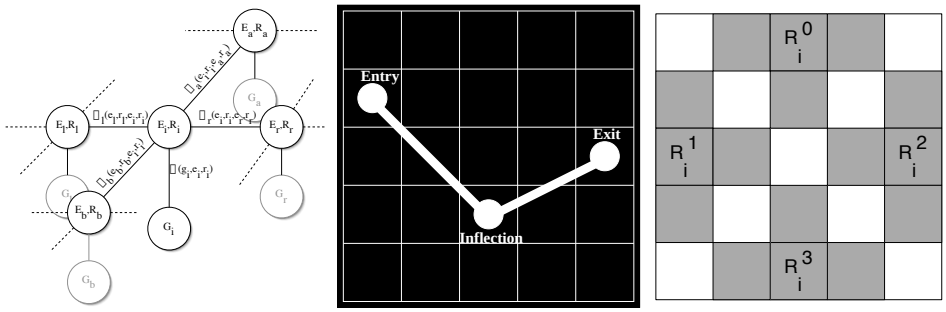

Figure 1: Left: A piece of the Markov random field model, with variables indicating boundary (E), region (R), and local edge detection (G) at each image patch. Center: A sample E variable assignment, parameterized by entry, inflection, and exit points in a 5x5 pixel patch. Right: The grayscale pixel values that are averaged to provide an $R_i = \{R_i^0, R_i^1, R_i^2, R_i^3\}$ assignment.

The object boundary model is inspired by Freeman et al.'s work on learning super-resolution [5]. The MRF model (Figure 1) partitions the image into a grid of 5x5 pixel, non-overlapping patches. Every patch, i, is associated with two random variables, a local gradient variable G_i, and a segmentation variable (E_i, R_i). The division of an image into regions is based on the shape properties of boundaries and the interior properties of regions. For instance, experience might bias a system into picking out square regions (boundary property) that contain only white pixels (region property). The E, or edge, variables represent the object boundary (or lack of boundary) through a particular patch, while the R

variables represent local pixel information and the G variables the response of several edge detectors at the center of each patch. In training, all variables are observed. In inference, the E values are hidden and must be estimated.

The MRF model represents the probability distribution of object boundary edges and image data. Every segmentation node, (E_i, R_i), is connected to a gradient node, G_i, and to its left (l), right (r), above (a), and below (b) segmentation node neighbors. The value e_i is an assignment to E_i, and a vector $\bar{e}$ is an assignment to every edge node; g_i and $\bar{g}$ and r_i and $\bar{r}$ are analogously defined.

After training, segmenting new images requires knowledge of the distribution $\Pr(\bar{e}|\bar{r}, \bar{g}) = \Pr(\bar{r}, \bar{g}|\bar{e}) \Pr(\bar{e}) / \Pr(\bar{r}, \bar{g})$. Given an image, $\bar{r}$ and $\bar{g}$ are fixed, so the desired boundary is the $\bar{e}$ that maximizes $\Pr(\bar{r}, \bar{g}|\bar{e}) \Pr(\bar{e}) = \Pr(\bar{e}, \bar{r}, \bar{g})$. The MRF represents $\Pr(\bar{e}, \bar{r}, \bar{g})$ with two sets of compatibility functions. The $\phi_i(g_i, e_i, r_i)$ functions represent the compatibility between pairs of gradient values and segmentation values, and $\psi_{ij}(e_i, r_i, e_j, r_j)$ functions represent the compatibility between assignments to neighboring segmentation nodes.

In learning, it is easiest to summarize the training data by the neighboring-node marginal probabilities, but there is no closed-form method for translating these values into equivalent compatibility functions. An analysis by Wainwright et al. [16] provides

$$\phi_i(g_i, e_i, r_i) = \Pr(g_i, e_i, r_i)$$

and

$$\psi_{ij}(e_i, r_i, e_j, r_j) = \frac{\Pr(e_i, r_i, e_j, r_j)}{\Pr(e_i, r_i) \Pr(e_j, r_j)}$$

as approximate maximum-likelihood estimates of the compatibility functions.

It is easy to generate training data via background subtraction, so it is a simple matter to estimate any discrete probability function by frequency counting. The set of potential edges in a 5x5 patch is discrete, but too large to use since a 5x5 binary image can take on 2^{25} possible values. Therefore, the model uses a parameterization (Figure 1) that represents any object boundary passing through a patch by its entry point, inflection point, and exit point. The "empty" edge value is included as a special case. This makes every E_i a discrete random variable with 2717 possible values. The filter responses in the G variables are discretized using 1000-bin histograms.

It is difficult to employ similar tactics with the R variables, so they are represented as continuous variables. Each R_i value (Figure 1) is a vector of four grayscale averages from four patch regions. The probability density functions for R variables are represented with mixtures of Gaussians, fit to the training data using expectation maximization. The compatibility estimate above and the belief propagation algorithm below are only valid for networks of discrete variables. However, since the R values are always observed, and the hidden E values are discrete, using a continuous representation for R is possible, and appears to do no harm.

4 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. The Stauffer and Grimson background subtraction algorithm is simple to compute, and is robust to non-motion variability [15]. For every frame in a video, it returns a binary image (Figure 2) indicating which pixels belong to moving objects.

Motion only provides data about the moving object. Therefore, the training procedure discards all data except the areas containing the moving object and its immediate surroundings. After post-processing to remove small noise regions, the cropped foreground-background

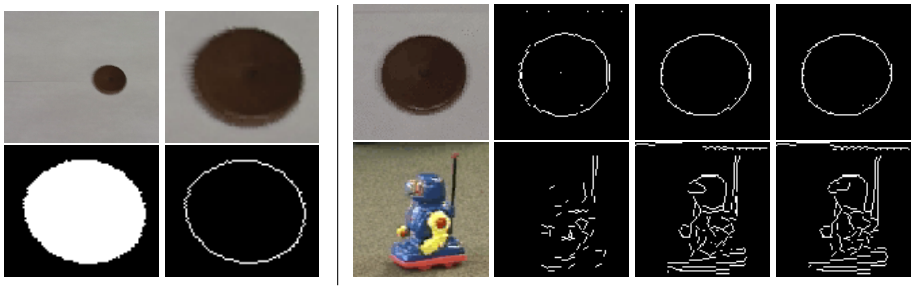

Figure 2: Left: The background subtraction algorithm detects the moving object, which is cropped from the frame. The binary background-foreground image is processed to produce an edge image for the object. Right: MAP estimates after 0, 5, and 10 iterations on a sample disc image and after 0, 10, and 20 iterations on a sample robot image.

image is processed into an edge image by a simple scanning algorithm. Each edge image and its associated cropped video frame provide a complete assignment to all the nodes in an instance of our object boundary model.

The training algorithm constructs representations of the ϕ and ψ functions by learning a few sets of probability distribution functions, which can be used to compute the compatibility functions described in Section 3 using Bayes' rule. The edge pixels at any location are represented by their best-fit edge in the parameterized edge model. The model is translationally invariant, so all the ϕ functions are equal, and the ψ functions fall into four classes: $\psi_l(e_i, r_i, e_l, r_l)$, $\psi_r(e_i, r_i, e_r, r_r)$, $\psi_a(e_i, r_i, e_a, r_a)$, and $\psi_b(e_i, r_i, e_b, r_b)$.

To learn the ϕ function, the algorithm assumes that $\Pr(g_i|e_i, r_i) = \Pr(g_i|e_i)$ and therefore $\phi(g_i, e_i, r_i) = \Pr(g_i|e_i) * \Pr(e_i, r_i)$. The ψ functions can be divided into four groups, one for each neighboring relationship (left, right, below, and above) and the probabilities can be factored into

$$\psi_{ij}((e_i, r_i), (e_j, r_j)) = \frac{\Pr(e_i, e_j)\Pr(r_i, r_j|e_i, e_j)}{\Pr(e_i)\Pr(e_j)\Pr(r_i|e_i)\Pr(r_j|e_j)}.$$

This factorization corresponds with the expected interaction between region properties and boundaries. If an edge assignment e_i divides an image patch, it will place some of the r_i elements in one region and some in another. Depending on the observed values of these pixels, the division might be more or less likely. Similarly, two neighboring edge assignments are more or less compatible depending on both the joint shape properties of their paired edges, and on the joint compatibilities of the pixels that they jointly group or separate.

In order to save memory and maximize the use of training data, the training assumes that object boundaries are invariant to rotation and reflection. Note that since each patch contains a piece of an object boundary and not just a point on it, this is not equivalent to making the four classes of ψ functions equivalent.

Data from multiple edge filters must be combined into a single single $\Pr(g_i|e_i)$ value. 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 $\Pr(g_i^1, g_i^2, \ldots, g_i^n|e_i)$ 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 $\prod_j \Pr(g_i^j|e_i)$. This assumption is incorrect, but it is useful, especially in discriminative applications [12]. 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.

5 Inference Algorithm

There is no efficient algorithm for exact inference on a Markov random field containing cycles, so we employ the belief propagation algorithm [11, 18]. 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. Our network has loops, but belief propagation still works well in practice, as it does in similar vision problems [5].

As in the super-resolution algorithm described by Freeman et al. [5], 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 with the largest $\Pr(e_i|g_i)$. 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 might border it.

On every iteration of the belief propagation algorithm, every segmentation node is visited and its messages are updated using the max-product belief propagation algorithm. After a user-selected number of iterations, the MAP estimate of E_i at each node can be locally computed, and the union of all of the local MAP estimates are used as the object segmentation estimate.

6 Results

Three models were trained, each on a particular video sequence. The sequences were a dark disc moving against a white background, a toy robot traveling across a highly textured carpet, and cars driving along a highway. In each case, the training set contained 200 frames.

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.

The G variables were the responses of four oriented derivative of Gaussian filters, oriented to 0, 45, 90, and 135 degrees. Canny [2] demonstrated the good edge-detection properties of derivative of Gaussian filters. These filters were computed on grayscale image values.

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

Figure 3 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 brightness

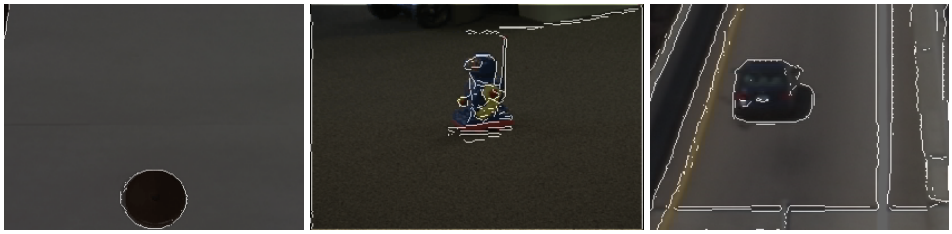

Figure 3: Sample results from three different data sets.

gradients 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. A more sophisticated model of shape, that captures longer-range dependencies, and the use of color and texture features might eliminate these undesired edges.

It is important to note that the information provided by the edge detection operators is very sparse because only the response at the center of each patch is used. As demonstrated in Figure 2, satisfactory results require the integration of sources of information across the image.

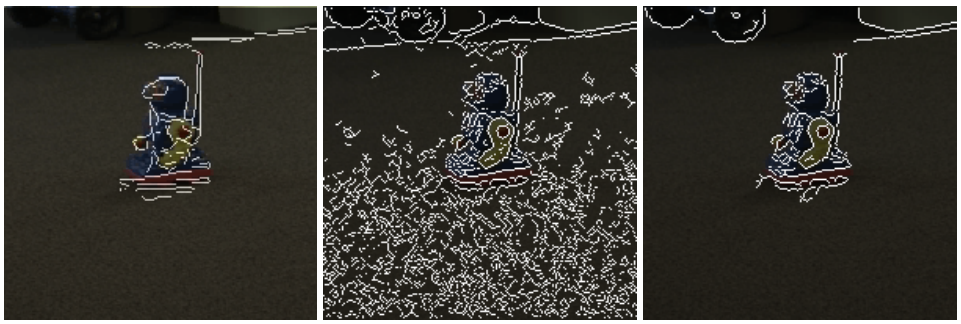

Figure 4: Top: From left to right, boundary detection with our algorithm, with the default Canny edge detector, and with a hand-tuned Canny edge detector.

Figure 4 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.

Figure 5 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 low image gradients at the borders of some of the cars, and the incompatibilities caused by the intersection of car and

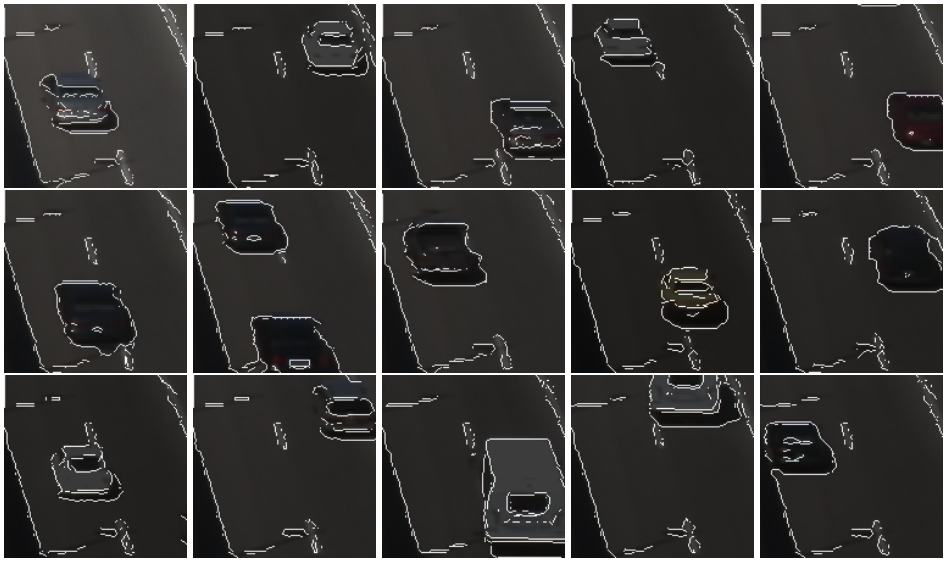

Figure 5: These results were inferred by the car model on images drawn from another road's traffic. The results required 40 candidates per node and 20 iterations of belief propagation.

road contours.

The running time of of our inference algorithm can be divided into two phases, constructing an MRF model for a given image and performing belief propagation. On a 150x150 pixel image, allowing 40 initial candidate edges at each node, building the model required approximately 13 minutes, but the 20 iterations of belief propagation that followed required less than 30 seconds. The current model-creation implementation performs many expensive marginalization calculations that are difficult to reuse, but a better implementation might significantly improve that aspect of the running time. Results were produced using a Java implementation of the algorithm on a dual 1.8 GHz Apple Power Macintosh G5.

7 Future Work

Our future work is focused on expanding the features used for the R variables, adding color and local texture information in order to achieve better results. We also intend to provide numerical comparisons of our algorithm to other methods using the motion information to provide ground truth for a test data set.

Acknowledgments

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

The authors thank William T. Freeman, Chris Stauffer, Yair Weiss, Martin Wainwright, and Erik Sudderth.

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] 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.

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

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

[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*, 25(1), 2003.

[9] 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.

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

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

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

[13] 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.

[14] 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.

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

[16] 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.

[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.