# Learning Static Object Segmentation from Motion Segmentation

by

Michael Gregory Ross

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

August 2005



Author .....
Department of Electrical Engineering and Computer Science
August 16, 2005

Certified by .....
Leslie Pack Kaelbling
Professor
Thesis Supervisor

Accepted by .....
Arthur C. Smith
Chairman, Department Committee on Graduate Students

# Learning Static Object Segmentation from Motion Segmentation

by

Michael Gregory Ross

Submitted to the Department of Electrical Engineering and Computer Science
on August 16, 2005, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

## Abstract

This thesis describes the SANE (Segmentation According to Natural Examples) algorithm for learning to segment objects in static images from video data. SANE uses background subtraction to find the segmentation of moving objects in videos. This provides object segmentation information for each video frame. The collection of frames and segmentations forms a training set that SANE uses to learn the image and shape properties that correspond to the observed motion boundaries. Then, when presented with new static images, the model infers segmentations similar to the observed motion segmentations.

SANE is a general method for learning environment-specific segmentation models. Because it is self-supervised, it can adapt to a new environment and new objects with relative ease. Comparisons of its output to a leading image segmentation algorithm demonstrate that motion-defined object segmentation is a distinct problem from traditional image segmentation. The model outperforms a trained local boundary detector because it leverages the shape information it learned from the training data.

Thesis Supervisor: Leslie Pack Kaelbling
Title: Professor

# Acknowledgments

My wife, Sophie Delano, has encouraged, supported, and endured me ever since we met at Dartmouth College. Her love makes every obstacle surmountable.

My parents, Carol and Greg Ross, were my first teachers and supporters and I can never thank them enough for the values and the love they gave me. My sister, Jessica Ross, was my partner in playing "Star Wars" on the big rocks in our backyard, an experience that started the path to this thesis.

My cats, Mal and Thisbe, provided lots of reassurance.

I've always been able to count of the support of friends that I've met at every stage of life, especially Ravi Nanavati, Joe Schatz, Dave Schaefer, Jeff Steeves, Pat Murray, Lon Setnik, Amy Murray, Andy Schoenhard, Sharon Stanley, Christine Alvarado, Mark Foltz, Mike Oltmans, and John Winn.

At MIT, I've had the good fortune to spend a lot of time working and not working with my officemates: Huizhen Yu, James McLurkin, Luke Zettlemoyer, Sarah Finney, Nick Matsakis, Meg Aycinena, Natalia Hernandez-Gardiol, Kurt Steinkraus, Sam Davies, Terry Koo, and Yu-Han Chang (my road office-mate). I also thank Erik Learned-Miller, Joshua Migdal, Erik Sudderth, John Fisher, Chris Stauffer, Luis Ortiz, John Barnett, and Hanna Pasula for their help with this thesis and other projects. I also relied on Mark Pearrow for uncounted hours of computer support and Teresa Coates Cataldo for her fearless navigation of bureaucracy.

My algorithm relied on background subtraction code supplied by Chris Stauffer and Joshua Migdal. The comparisons in this thesis would not have been possible without software provided by David Martin and Charless Fowlkes and by Jianbo Shi and his students.

This thesis is the culmination of my formal education, which began in 1980. During the 18 years before I began graduate school I was fortunate to have many teachers who helped and influenced me profoundly. I especially thank Carol Buteau, Sam Slarskey, Dorothy Johnson, Niki Veley, Neil Murphy, Bill Noeth, Tom Cormen, and Daniela Rus.

I also thank my thesis committee: Tomás Lozano-Pérez, Bill Freeman, and Alan Yuille. Their questions and challenges lead to some of the most significant parts of this work.

The last seven years at MIT were the most challenging of my educational experience. Without the support of my advisor, Leslie Pack Kaelbling, I would not have made it to this point. I hope that I honor her kindness and integrity by following her example.

I dedicate this thesis to my grandmother, Ellinor Orpen, who died this past Christmas. Her contributions to this work include: encyclopedias that helped me pass uncounted elementary school writing assignments, a winter weekend home and grilled cheese sandwiches that made me the academic and basketball success that I am today, and the patience to watch *2001: A Space Odyssey* twice.

# Contents

# List of Figures

"Stephen pointed to a basket which a butcher's boy had slung inverted on his head.

-Look at that basket, he said.

-I see it, said Lynch.

-In order to see that basket, said Stephen, your mind first of all separates the basket from the rest of the visible universe which is not the basket. The first phase of apprehension is a bounding line drawn about the object to be apprehended."

James Joyce, *A Portrait of the Artist as a Young Man*, 1916 [20]

"The simple things you see are all complicated"

Pete Townshend, *Substitute*, 1966 [45]

# Chapter 1

# Introduction

## 1.1 Vision requires grouping

A picture is worth a thousand words, but a pixel is nearly worthless. A single pixel in an image carries very little useful data about the observed scene. Its brightness and color might be due to an object surface, lighting conditions, or instrument sensitivity. Its value may be corrupted by sensor noise. It provides no depth information. There are few useful vision tasks that can be performed with a single pixel. Thus, computer vision necessarily requires the aggregation of information from many pixels. Grouping pixels is an essential visual process.

Although grouping is essential, it need not be explicit. Consider the classical optical flow equation of Horn and Schunck [18, 17]:

$$\iint (E_x u + E_y v + E_t)^2 + \lambda((u_x^2 + u_y^2) + (v_x^2 + v_y^2))\,\mathrm{d}x\,\mathrm{d}y.$$

In order to overcome the aperture problem, that only motion perpendicular to the brightness gradient can be detected locally (Figure 1-1), a regularization term that encourages flow vectors to vary smoothly across an image ($(u_x^2 + u_y^2) + (v_x^2 + x_y^2)$) is added to the flow error term ($(E_x u + E_y v + E_t)^2$). This embodies a weak pixel-grouping assumption—that neighboring pixels are part of a coherent object motion. The popularity of Markov random field models in computer vision is entirely due to their utility in grouping the properties of multi-

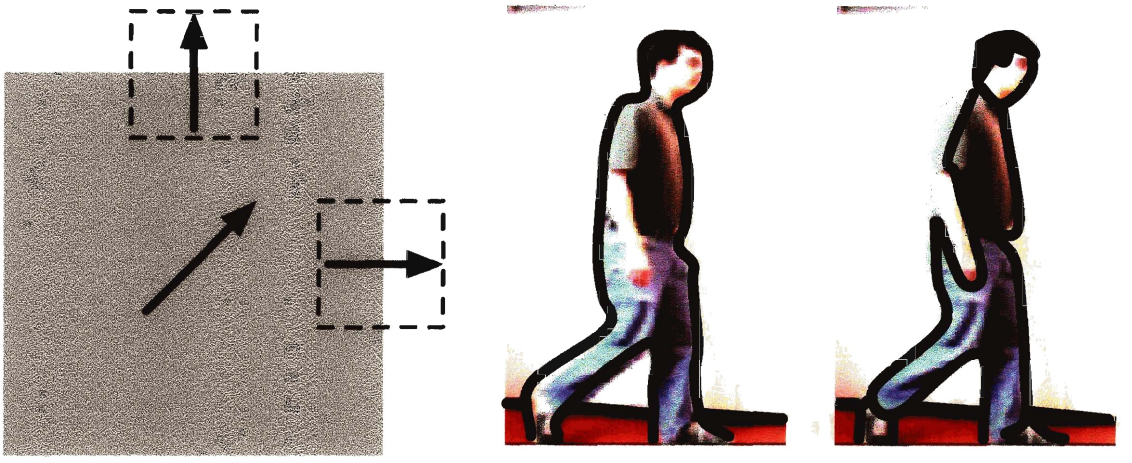


Figure 1-1: Left: Determining the motion of a moving object requires grouping information from multiple locations. Local information can only provide motion components that are parallel to the local image gradient. Right: The segments useful for computing stereo depth might differ from those useful for texture discrimination or compression.

ple image pixels to make useful calculations, such as denoising [14], super-resolution [12], texture modeling [27], or stereo depth [5].

Even in the absence of physical requirements, grouping can provide computational benefits. Searching for the position of the upper-left hand corner of an object in a 200 pixel by 200 pixel image might require the evaluation of 40,000 possible locations. If all possible rotations need to be considered, in 5 degree increments, that produces 2,880,000 comparisons. Increasing the degrees of freedom further arbitrarily inflates the computational requirements. A reasonable scheme of grouping the pixels into locally similar regions might reduce the search space from 40,000 pixels to 100 regions.

The ubiquity of grouping makes it as fundamental to the study of vision as the physics of light or the mathematics of projection geometry. Grouping covers a range of processes, from the relatively weak regularization constraints of the Horn and Schunck equation to the explicit division of image pixels into multiple disjoint sets. This later extreme is usually termed "image segmentation" and has been studied by both psychologists and computer vision scientists since at least the early twentieth century [36].

## 1.2 What segments?

The first task in the study of image segmentation is specifying a proper criterion for the image regions. Are regions simply neighboring pixels that are nearly similar in color or brightness? Do they encompass entire physical objects? Can they contain high-variation internal textures? If so, how is textural similarity measured?

Although many segmentation algorithms address these questions by choosing practically computable optimizations of region statistics that produce visually appealing regions, the relationship between segmentation and regularization provides a better answer. Just as regularizers are chosen for their utility in accurately fitting desired functions, segmentation algorithms should be chosen for their utility to other computations in an agent's visual and non-visual systems (Figure 1-1).

Because the term "image segmentation" often refers to optimizing purely image-based region properties, or even the production of regions that are simply visually appealing, the external utility approach to segmentation is better described as "object segmentation." Just as object-oriented programming marshals code and data into objects, the primitives that are most useful for a given task, object segmentation seeks to create pixel groupings that correspond to task-appropriate objects. If the task is purely image-based, such as compression, a segmentation based on the principle of minimum description length, such as LeClerc's algorithm [25], might be appropriate. Discovering the materials present in a scene, on the other hand, might be better served by a segmentation that explicitly respected surface texture and reflectivity properties.

For any given object definition, it is important to determine what information is necessary to distinguish desirable object boundaries from all the other visual boundaries in the images. In many cases, local image information might not be sufficient and a successful object segmentation might require shape information or other "high-level" knowledge.

The most principled way to develop an object segmentation algorithm is to integrate it into another system. A visual robotic navigation system, for example, might need to locate the boundaries of obstacles. Using the success or failure of the navigator as reinforcement and the output of bump sensors as supervision, a segmentation algorithm could

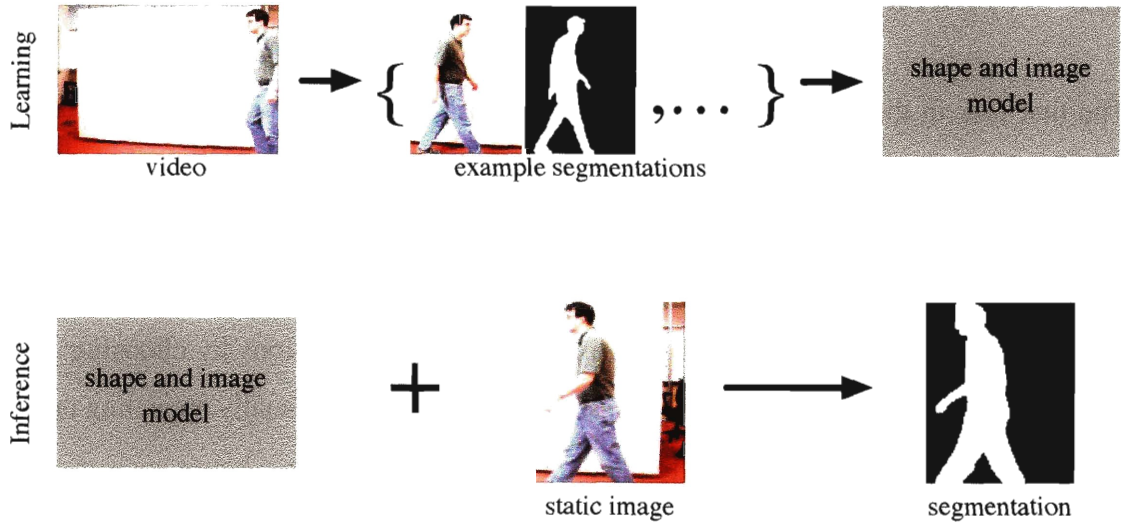



Figure 1-2: The SANE algorithm learns the shape and image properties of moving objects in videos. Then it uses this information to infer the segmentation of objects in static images.

be trained to maximize performance on the high level task, which might require balancing computational requirements against segmentation correctness. An alternative to this fully integrated approach is to put forth a generally useful object definition and a method for providing training and testing ground-truth object segmentations. Segmentation performance is measured by the match between inferred and ground-truth segmentations. The measurements reflect the algorithm's utility to any system that requires the segmentation of the specified class of visual objects.

## 1.3 Learning to segment motion-defined objects

Motion provides a useful, general-purpose object definition for segmentation. Defining an object as a collection of elements that undergo coherent motion corresponds well to intuition. All the visually distinct elements of a car can be considered a single object because they frequently move together, but a pile of leaves is composed of many objects that move independently on a windy day. A robot manipulator must know whether a collection of pixels represents a single entity or something that will disintegrate at the slightest touch.

Apart from their utility, motion-defined objects are desirable because their test and training example sets can be automatically generated. A video camera on a street corner can

record hundreds of moving cars, people, or animals every hour, and software can use their motion to separate them from their surroundings. Thus motion-defined objects combine utility and practicality.

This thesis describes the SANE (Segmentation According to Natural Examples) algorithm for learning to segment objects in static images from video data. SANE uses background subtraction to find the segmentation of moving objects in videos. This provides object segmentation information for each video frame. The collection of frames and segmentations forms a training set that SANE uses to learn the image and shape properties that correspond to the observed motion boundaries. Then, when presented with new static images, the model infers segmentations similar to the observed motion segmentations.

SANE is a general method for learning environment-specific segmentation models. Because it is self-supervised, it can adapt to a new environment and new objects with relative ease. Comparisons of its output to a leading image segmentation algorithm demonstrate that motion-defined object segmentation is a distinct problem from traditional image segmentation. The model outperforms a trained local boundary detector because it leverages the shape information it learned from the training data.

# Chapter 2

# Related work

## 2.1 Image segmentation

Modern scientific interest in image segmentation derives from the work of Max Wertheimer and other Gestalt psychologists, who studied the visual properties that cause humans to perceive multiple image elements as unified groups. They identified grouping factors such as proximity, color, orientation, and common motion [36]. Many of the features used in computer image segmentation algorithms correspond to these classical principles.

The work of Spelke et al. on the development of image-segmentation abilities in children is a particular inspiration for the work in this thesis. They determined that the ability to distinguish object boundaries by motion and depth perception developmentally precedes the ability to segment based on cues such as color, brightness, and texture [41]. These results suggest that segmentation with these cues can be learned from the more primitive modes of segmentation.

In computer vision, image segmentation is considered an ill-defined problem because of the difficulty in quantitatively specifying the properties of a good segmentation. David Marr despaired "that the difficulties in trying to formulate what should be recovered as a region from an image are so great as to amount almost to philosophical problems [30]." While acknowledging this criticism, Shimon Ullman noted the existence of psychological evidence for segmentation processes that precede object recognition [46]. In addition to these difficulties, the problem of finding a segmentation that obeys specified region prop-

erties with the fewest possible regions is known to be NP-complete [7].

As a result, there are multiple approaches to the image segmentation problem. Some algorithms model the properties of region boundaries. The Canny edge detector [6] defines a local operator for detecting likely edges and sought to produce well-connected edges through the use of multiple thresholds—effectively allowing borderline cases to be classified as edges if they were connected to locations that were above the higher threshold. However, this hysteresis technique does not completely eliminate the possibility of broken edges, which prevent the Canny algorithm from being a true segmentation method. Kass et al.'s snakes algorithm [21] solves this problem by specifying the existence, initial size, and location of a set of complete image boundaries, and then uses gradient ascent to maximize their match to local image boundary indicators, such as high-gradient regions, and boundary-shape smoothness requirements. The initialization requirements of snakes are alleviated in the more modern level-set segmentation algorithms [29]. Taking a different approach to the boundary-modeling problem, Shashua and Ullman developed a network of variables that can infer boundaries by passing information that allow the completion of contours across image regions that lack local edge information [39].

The boundary algorithms typically only model region borders and not their internal properties. Unsurprisingly, they are complimented by a group of region algorithms that operate by assigning segment labels to each pixel in an image. The best known of these is the Normalized Cuts algorithm developed by Shi and Malik [40]. This algorithm creates a weighted graph in which each pixel is connected to every other, and the weights represent the similarity between them. A cut of the graph is a set of edges whose removal divides the pixels into two groups. A minimum cut is the cut whose total edge weights are the smallest, which is biased towards separating small regions from the remainder of the image. Normalized Cuts corrects this bias by dividing the cut value by associativity factors that penalize small partitions.

While Normalized Cuts divides an initially fully-connected graph into smaller regions, Felzenszwalb and Huttenlocher's algorithm [9] is an example of the region-growing approach, which starts with each pixel in an individual segment and iteratively merges the pixels into larger regions. Regions in this algorithm can be either high-variance, as in the

case of a highly textured surface, or low-variance, such as a region of constant color, and the merging procedure automatically adjusts its criteria to maintain these properties.

Just as the boundary models largely ignore region interiors, these region models forego shape modeling. Although their methods tolerate noise in measurements of pixel associativity, it is difficult to make a model based on pairwise pixel similarity encode strong priors of region shape, which is probably why boundary-based methods appear to dominate domains such as medical imaging, where noise is high, but a great deal of prior shape knowledge is available.

Yet another approach to image segmentation is based on the principle of minimum description length, a criterion drawn from information theory, which casts segmentation as a data-compression problem. LeClerc developed a language that represents region interior variation and boundaries and finds segmentations that result in efficient image descriptions [25]. Zhu and Yuille demonstrated that snakes, region-growing, and the LeClerc method can be combined in a general framework [56].

The Geman and Geman image-restoration algorithm [14] can be viewed as a segmentation method that combines some boundary and region properties. The algorithm seeks to recover the true pixel brightnesses in a noise-corrupted image, but these uncorrupted values can also be considered segmentation labels when the regions are defined as areas of constant brightness. The Geman and Geman model represents the relationship between these labels with two Markov random fields (for more details on Markov random fields, see Section 2.2). One field enforces the constraint that neighboring pixels should be given the same label. The other field infers the presence or absence of boundaries between each pair of pixels. The presence of a boundary at a particular location makes the presence of nearby boundaries more likely and also makes it more likely that pixels on either side of the boundary will be given different labels. The segmentation model described in Chapter 3 has many similarities to this formulation, except that it works on patches rather than pixels, which allows it to capture more shape information and more complex region properties than this pixel-based model.

If example image segmentations are available, supervised machine learning can be applied to the problem. In recent years, there have been an increasing number of publications

in this and related areas. Two important recent algorithms use human-segmented test and training data to learn boundary detection models. Martin et al. [32] studied the performance of brightness, color, and texture gradient-based detectors on the task of replicating human-determined boundaries from the Berkeley Segmentation Database [33]. By fitting combinations of features to the training data using logistic regression, Martin developed a detector that matched human performance more closely than well-known alternatives, such as the Canny edge detector [6]. The algorithm utilizes complex texture features extremely effectively, but classifies each pixel independently. Lacking a boundary-shape model, it's unable to complete boundaries across regions that lack local information. Konishi et al. [23] trained on the Sowerby and South Florida human-labeled boundary data sets and developed a new method for combining multiple features in their detector. They employed an adaptive histogram and decision trees to represent the joint probability of the features, using the Chernoff information to determine which divisions are most useful for the edge-detection task. Although a version of the algorithm learned equivalents of the Canny techniques for extending edges across low-information regions, the results were improved only slightly, perhaps indicating the desirability of stronger shape information. Neither the Konishi and Martin algorithms ensure that all detected edges are part of closed boundaries, so neither is a true segmentation algorithm.

Hayman and Eklundh [15] have contributed an algorithm that learns to combine multiple video cues on-line, bootstrapping the process from an initial motion-only segmentation algorithm. However, the algorithm does not learn a static-image segmentation model, it just utilizes static cues to improve its motion segmentation.

Using a top-down approach, Borenstein and Ullman's class-specific segmentation algorithm [3] fits a stored database of object parts, derived from a manually segmented training set, to a new image to determine the appropriate segmentation. A more recent extension [4] requires two sets of images for training, one containing examples of the object class and the other consisting of out-of-class images, but does not need prespecified segmentations. An iterative technique determines the examples' figure-ground segmentations and constructs the appropriate object-part database. This technique assumes images in which the foreground objects are less variable across the set than the surrounding background regions,

which may not be the case in some data, such as overhead pictures of cars on a highway.

Among these learning algorithms, none provide a model for segmenting objects in static images that can capture useful shape and region information and provide a reasonable expectation of generalizability to larger domains. The Konishi and Martin algorithms focus on local boundary detection, and the class-specific fragment databases used in the Borenstein and Ullman methods seem unlikely to generalize well. The Hayman and Eklundh algorithm uses motion as a starting point for learning, but only segments moving, not static, images. The other algorithms require human-supplied training data. All but the second Borenstein and Ullman algorithm require example human segmentations for training, which means that they are learning to match human-perceived regions of uncertain significance. Although Martin et al. note that human segmentations demonstrate remarkable consistency [33], that still does not address the question of the underlying region definition and whether they are useful inputs to other algorithms. The second Borenstein and Ullman algorithm relaxes the human-segmentation requirement, but the robustness of the learning technique and the practicality of gathering sufficient in-class images are unknown, and, ultimately, the learned models still reflect an unknown human object definition.

Medical-imaging segmentation, such as the work of Leventon et al. [26], frequently attempts to duplicate the manual segmentation of organs or other body structures of interest. The use of human-provided training examples is ideal for these applications because the goal is to remove the need for humans to segment the images, and organ boundaries seem to be more precisely specified than human-segmented regions in typical natural images.

Example segmentations of a particular object class could also be used to learn an object-detection model. In object detection, the goal is to discover image regions that contain examples of a particular object or object class. Some notable recent object detection algorithms include the Viola and Jones face detector [47] and Lowe's SIFT algorithm [28]. Both of these algorithms, like many other object-detection methods, draw bounding boxes around detected objects and do not find their boundaries in any detail. Differentiating between a face and a car is done more effectively by detecting key features such as tires or eyes than by finding boundaries and comparing shapes. But many tasks, such as obstacle avoidance or optical flow, are much more sensitive to precise boundary location than they

are to the types of objects present. Even object detection can benefit from a process that separates objects from their surroundings without requiring specific detection. Eliminating unwanted background variance can make detectors simpler and more effective. It's also useful to consider that object segmentation is founded on the notion that some boundary image and shape properties are broadly useful — an algorithm that learned to segment farm animals and then successfully segmented tractors would be considered a huge success. Object detection, on the other hand, focuses on discrimination. An object detector that is trained on farm animals and then detects a tractor would be a failure. It's now just a low-resolution segmentation algorithm.

Fitzpatrick developed an algorithm that allows a robot to collect segmented object examples by poking possible objects and observing their motion [10], a strikingly similar approach to the method described in Chapter 3 for extracting example segmentations by passively observing motion. Fitzpatrick used the examples to learn to detect and segment specific objects, while the algorithms in this thesis learn to segment large classes of objects.

## 2.2 Markov random fields

The segmentation models introduced in Chapter 3 are based on Markov random fields (MRFs), a class of probabilistic graphical models that encode the relationships between multiple variables. To introduce Markov random fields properly, some notational conventions are required. The remainder of this thesis will adopt the convention that variables are denoted by capital letters ($V$) and their assignments by lower-case letters ($V = v$). The elements of vector-valued variables are indicated by subscripts ($V_i$). Sets are upper-case ($C$) and their elements are lower-case ($c$). Sums and products over all possible assignments to a variable are indicated by the variable name ($\sum_V$, $\prod_V$), sums and products over set elements are indicated by the set name ($\sum_C$, $\prod_C$).

Markov random fields have been widely used since their introduction to computer vision in Geman and Geman's work on image restoration [14]. A Markov random field is an undirected graph whose nodes represent variables of interest. If a graph contains variables $V$, and the neighbors of a particular variable $V_i$ are denoted $N(V_i)$, the Markov property

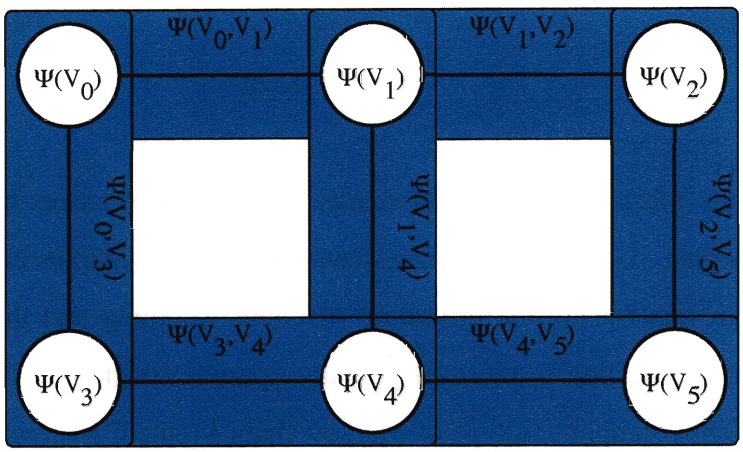



Figure 2-1: A Markov random field consisting of 6 variables arranged in 6 one-element and 7 two-element cliques.

holds that $P(V_i|V \setminus V_i) = P(V_i|N(V_i))$, a variable is independent of the remainder of the graph given the values of its neighbors. A Markov chain is a one-dimensional MRF. Figure 2-1 shows a simple Markov random field.

Unlike Bayesian networks, the joint distribution of the variables in an MRF cannot be found by multiplying local conditional probability functions. Instead, the Hammersley-Clifford theorem [1] states that every Markov random fields is a Gibbs random field, in which

$$P(V = v) = \frac{\prod_C \Psi_c(V_c = v_c)}{Z},$$

where $C$ is the set of cliques, or mutual neighbors, in the graph[1]; $\Psi_c$ is a positive-valued compatibility function; and

$$Z = \sum_V \prod_C \Psi_c(V_c = v_c)$$

is a normalization constant called the "partition function." For a particular assignment to the variables in clique $c$, the larger the value of $\Psi_c$, the more likely the combination.

[1] Readers unfamiliar with mathematical terms such as "cliques" can find excellent definitions on the Mathworld web site at http://mathworld.wolfram.com/.

## 2.3 MRF inference

The necessity of normalizing the Markov random field energy function by Z makes inference in these models challenging. Geman and Geman demonstrated that samples can be drawn from Markov random fields using the procedure of Gibbs sampling and this technique can be combined with annealing to discover globally optimal assignments to a model [14]. Unfortunately, Gibbs sampling, a type of Markov chain Monte Carlo technique, requires many iterations to produce useful samples in a large MRF, and the annealing procedure requires a slow cooling schedule to ensure optimality. Therefore, it is not surprising that much of the research in Markov random fields has focused on discovering alternatives and approximations that make their use more practical.

Besag [2] noted that simple hill-climbing produces globally adequate assignments to MRF variables in many instances. His iterative conditional modes (ICM) algorithm takes an initial assignment and scans through all the variables, making any local change that increases the probability of the overall assignment. Since individual modifications only affect a few clique potentials, the algorithm is extremely efficient, although it is most useful in cases where the initial assignment is nearly optimal. Some other well-known methods are the pseudo-likelihood approach, which approximates the joint MRF distribution with the product of all the local conditional probabilities, and the related mean-field approximation, which assumes that the influence of non-neighboring nodes can be represented by their mean assignments [27].

Currently, a new class of algorithms based on the belief propagation procedure, are ascendent. Discovered by Pearl, belief propagation [37] is a pair of related algorithms that allow for efficient calculation of local marginal probabilities and global maximum *a posteriori* (MAP) estimates in tree-structured MRFs and Bayesian networks. Every variable $V_i$ has a set of possible assignments $\{v_i\}$ and a message vector $M_{i \leftarrow j}$ for each neighbor $V_j \in N(V_i)$. Belief propagation is an iterative algorithm with the message vectors initialized to 1. Given a pairwise MRF, which contains only one and two-element cliques, an

iteration of the *sum-product* algorithm for computing marginal probabilities is

$$M_{i \leftarrow j}(v_i) = \sum_{v_j} \Psi_j(v_j)\Psi_{i,j}(v_i, v_j) \prod_{V_k \in \{N(V_j) \setminus V_i\}} M_{j \leftarrow k}(v_j).$$

An iteration of the *max-product* algorithm, which computes joint MAP estimates, only substitutes max for $\sum$, producing

$$M_{i \leftarrow j}(v_i) = \max_{v_j} \Psi_j(v_j)\Psi_{i,j}(v_i, v_j) \prod_{V_k \in \{N(V_j) \setminus V_i\}} M_{j \leftarrow k}(v_j).$$

In a tree-structured MRF with $|V|$ variables, either algorithm converges after at most $|V| - 1$ iterations. Then, the sum-product messages can be used to calculate the single node marginals

$$P(v_i) = \Psi_i(v_i) \prod_{V_j \in N(V_i)} M_{i \leftarrow j}(v_i)$$

or the max-product messages can be used to calculate the joint MAP estimate by selecting

$$V_i = \arg\max_{v_i} \Psi_i(v_i) \prod_{V_j \in N(V_i)} M_{i \leftarrow j}(v_i)$$

at every node.

Although these results are only guaranteed for tree-structured MRFs, there is nothing in the algorithm that prevents its application to graphs containing loops. Commonly called "loopy belief propagation," this approach has found great favor among computer-vision researchers because it often produces useful approximate inference [51]. The effectiveness of loopy belief propagation led to further study which revealed its relationship to Bethe free-energy models in statistical physics and the existence of more accurate physics-based approximate inference techniques [55].

To solve the possible non-convergence of loopy belief propagation, researchers have investigated altering the "step size" of the algorithm. Instead of replacing old messages with new ones after each iteration, a combination of the old and new messages is stored. Murphy et al. found that using a linear combination of the old and new messages improved

convergence in some cases, but note that "in several cases the beliefs to which the algorithm converged were quite inaccurate" [35]. Heskes described "damping" the steps of loopy belief propagation by linearly combining the logarithms of new and old messages [16]. In this formulation, if $M_{i \leftarrow j}^{t}$ is a message that regular belief propagation would compute at time $t$ and $\hat{M}_{i \leftarrow j}^{t}$ is the damped message,

$$\hat{M}_{i \leftarrow j}^{t} = \exp(\lambda \log M_{i \leftarrow j}^{t-1} + (1 - \lambda) \log \hat{M}_{i \leftarrow j}^{t-1})$$

with damping factor $\lambda$.

A competing inference approach that has found favor with some vision researchers uses solutions to the minimum-cut graph problem to find MRF MAP estimates [5, 22]. These methods impose some restrictions on the form of the MRF potentials and if the unknown variables are not binary they require multiple iterations of expensive graph construction and cutting to compute an approximate estimate. Currently, there is no consensus over whether these graph-cut methods are generally superior or inferior to loopy belief propagation on typical vision problems. Tappen and Freeman compared the two methods' performance on stereo problems and found that while graph cuts found higher-scoring assignments, those assignments were not necessarily more accurate compared to ground truth. This indicates that modeling choices were more important than inference algorithms on these problems [44].

## 2.4 MRF parameter estimation

Creating a Markov random field model for a set of variables requires choosing values for the clique potential functions. There is no universal relationship between clique potential functions and the joint probability distributions of the clique's component variables, because the normalizing partition function Z is necessary to all marginalizations and is influenced by all of the MRF's potentials. Certain graph structures can be decomposed, which results in a simple, closed-form relationship between clique potentials and marginals [19]. Unfortunately, large, grid-structured MRFs such as those described in Chapter 3 are not de-

composable. Alternative maximum-likelihood approaches require computing the Z function, which is intractable. As a result, approximate algorithms based on pseudo-likelihood, mean-field, or other approximations are frequently used.

Another parameter-fitting procedure is iterative proportional fitting (IPF), which selects parameters that match the marginal probabilities of the model to a set of observed marginal probabilities [19]. For example, if a clique consisting of variables A and B has an observed probability $\hat{P}(A, B)$, and the current MRF parameters produce a marginal $P(A, B)$, IPF updates

$$\Psi^t(A, B) = \Psi^{t-1}(A, B)\frac{\hat{P}(A, B)}{P(A, B)}.$$

These updates are guaranteed to converge and monotonically increase the log-likelihood of the data. However, they do require inferring clique marginals, which is intractable due to the partition function. One solution is to use approximate inference to efficiently computer the necessary marginals. Wainwright and Sudderth have proven that in an MRF with single and pairwise clique potentials, $\Psi_i(V_i) = P(V_i)$ and $\Psi_{ij}(V_i, V_j) = \frac{P(V_i, V_j)}{P(V_i)P(V_j)}$ are fixed points of IPF using belief propagation inference [50]. Subsequently, Wainwright et al. demonstrated that these parameter settings are also approximate maximum-likelihood estimates [49] and noted that they are exact compatibilities on a tree-structured MRF [48]. The two-element compatibility functions have a useful intuitive interpretation. If $\Psi_{ij}(V_i = v_i, V_j = v_j) = 1$, the assignments are independent of each other, if $\Psi > 1$, they are positively correlated and if $\Psi < 1$, they are negatively correlated. These approximations will be the bases for the compatibility functions used in this thesis work. These compatibilities have been used in MRF models prior to Wainwright and Sudderth's theoretical justification. For example, they are equivalent to the conditional probability compatibilities used by Freeman et al. [12].

# Chapter 3

# The SANE Algorithm

## 3.1 Algorithmic development

The Segmentation According to Natural Examples (SANE) algorithm is an image segmentation algorithm that is trained via supervised machine learning to segment objects in individual static images. The training examples are automatically generated by a background subtraction algorithm that segments moving objects from their surroundings in videos. Taken together the full system is a self-supervised learning procedure for training static image segmentation models from videos of moving objects.

The SANE algorithm developed from a boundary-detection method to a full segmentation algorithm in three distinct stages. Each stage directly extended its predecessor, so the final algorithm is best described by its developmental history. The first version of SANE only modeled boundaries, the second version added more region-compatibility information, and the final version added explicit segmentation labels. After the main algorithmic discussion, a multiresolution extension, parameter training, and inference methods are described.

## 3.2 Boundary model

Segmentation and boundary detection are identical problems. Detecting the boundary of a region is equivalent to assigning that region's label to every interior pixel. By definition,

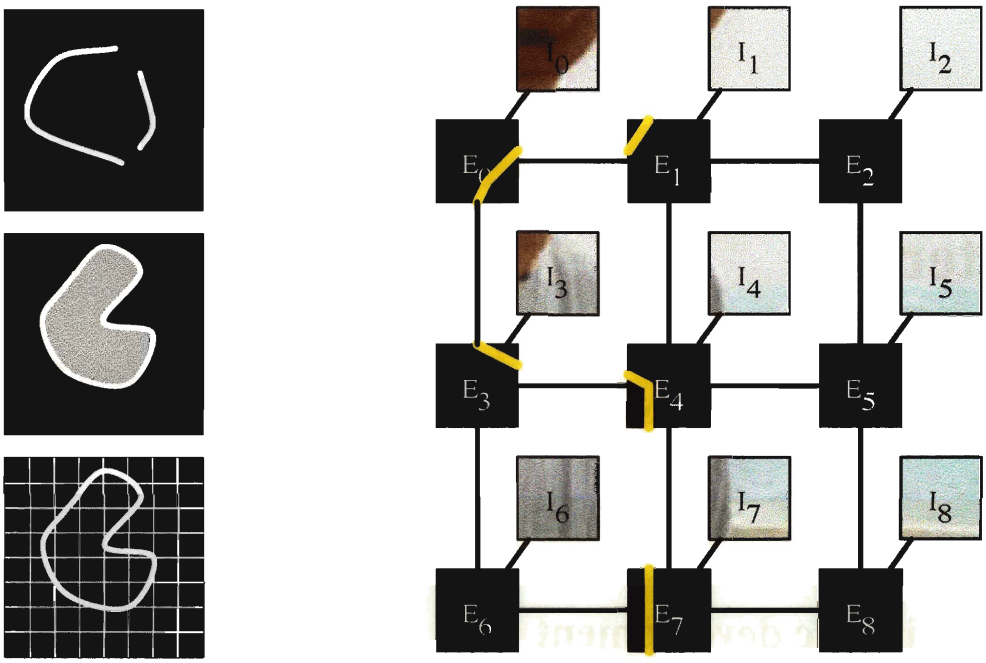



Figure 3-1: Left: Boundaries are closed and separate pixels into segments, while open edge segments do not. Boundaries can be represented as a grid of local edge assignments. Right: The first boundary model on a small sample image, with edge nodes E and image data nodes I.

boundaries are closed. An edge detector, such as the well-known Canny algorithm [6], which allows open edges, is not a boundary detector (Figure 3-1). The first SANE model attempted to learn the properties of object boundaries and perform segmentation by detecting complete, closed boundaries in each image.

Boundaries can be represented by a collection of connected local boundary segments (Figure 3-1), which we will refer to as edges. Given an image, the task is to infer the correct assignments to the hidden edge variables, denoted $E$, from the available image information, $I$. Inspired by Freeman et al.'s [12] use of a Markov random field on image patches to solve the super-resolution problem, the first boundary model (Figure 3-1) divides an input image into a lattice of 5 pixel by 5 pixel, non-overlapping patches. For each patch $i$[1] there is an MRF node $I_i$ representing the image data and an MRF node $E_i$ indicating the shape of the object boundary segment passing through that region or representing the absence of a boundary segment at a location with the "empty edge" value.

[1]When lattice location is important, patches and nodes will be indexed by their Cartesian lattice coordinates (i,j).

The edge nodes are connected to their edge-node neighbors in a first-order neighborhood and to their corresponding image node: $N(E_{i,j}) = \{E_{i+1,j}, E_{i,j+1}, E_{i-1,j}, E_{i,j-1}, I_{i,j}\}$ (Figure 3-1). This makes each image node independent of the remainder of the graph given the value of the edge node at that location. This conditional independence assumption, common in computer-vision MRF models, significantly limits the power of the model and eventually lead to the development of the second boundary model in Section 3.3.

Some simple mathematical manipulation will illuminate the parameters of this model. Because the elements of $I$ are conditionally independent,

$$P(E, I) = P(E)P(I|E) = P(E)\prod_i P(I_i|E_i).$$

The $P(E)$ term is a Markov random field with one and two-element cliques, $C_1$ and $C_2$. Therefore,

$$P(E, I) = Z^{-1}\prod_{E_i \in C_1} \Psi_i(E_i) \prod_{(E_i,E_j)\in C_2} \Psi_{i,j}(E_i, E_j)\prod_i P(I_i|E_i).$$

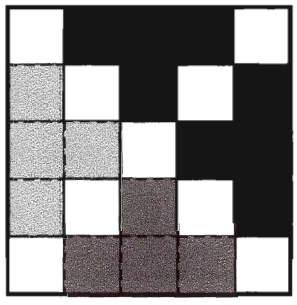
Applying the compatibility approximations from Section 2.4 produces

$$\Psi_{i,j}(E_i, E_j) = \frac{P(E_i, E_j)}{P(E_i)P(E_j)}$$

and

$$\Psi_i(E_i) = P(E_i).$$

The image nodes are real vector-valued variables representing image-patch features that are useful for distinguishing boundaries. In this thesis, all models use the average brightness of four patch regions as image features, as shown in Figure 3-2. Color models use the average red, green, and blue values of all the patch's pixels in addition to these local brightnesses. Earlier implementations represented each patch with the value of oriented derivative features evaluated at the center of each patch. The derivative features were replaced by average brightness and color values because those were more successful in the boundary and image model described in Section 3.3.



$b_{i0}=\Sigma\blacksquare/4 \quad b_{i1}=\Sigma\square/4 \quad b_{i2}=\Sigma\square/4 \quad b_{i3}=\Sigma\blacksquare/4$

brightness: $I_i=(b_{i0}, b_{i1}, b_{i2}, b_{i3})$

color: $I_i=(b_{i0}, b_{i1}, b_{i2}, b_{i3}, red_i, green_i, blue_i)$

Figure 3-2: Brightness models use average brightnesses of four patch regions as features. Color models add the average red, green, and blue values of all the patch's pixels as extra features.

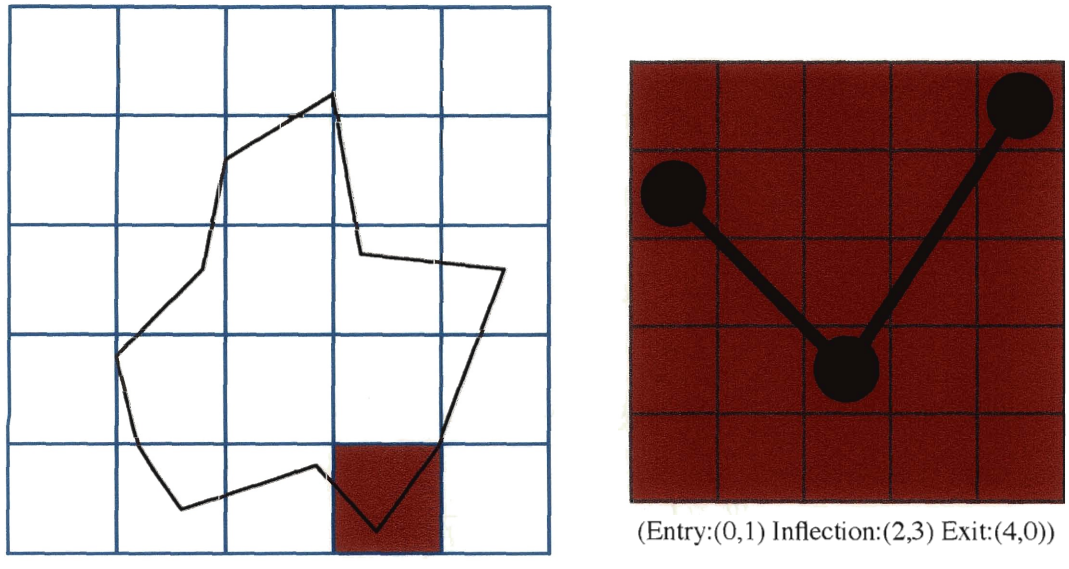



Figure 3-3: Any region boundary can be represented as a collection of edge patches.

Edge variables, on the other hand, are discrete and range over a parameterization of all possible boundary edges that can pass through a 5 by 5 patch. Because region boundaries are closed, their constituent edges cannot terminate in the middle of a patch. Therefore, local edges are represented as an entry point on the border of the patch, an optional inflection point inside the patch, and an exit point on the border of the patch. The permitted coordinates for these locations are the integer-valued pixel locations in the patch. Figure 3-3 shows an example boundary and one of its constituent edge values. This parameterization produces 2717 possible local edge assignments, including the empty edge as a special case.

Because this boundary model is supposed to produce segmentations, it is important to

investigate its behavior with respect to the continuity of neighboring edge assignments. If an edge's exit location borders another patch, the model should guarantee that the assignment to that neighbor continues the edge, that it has an entry or exit point near to that location.

Consider a function $\text{cont}(e_i, e_j)$, which returns 1 if $e_i$ continues $e_j$ or if the two assignments do not have entries or exits along their common border, and 0 otherwise. Entries and exits are only considered to be continued if they are no more than 1.5 pixels distant from an entry or exit on a neighboring patch. The *cont* function does not penalize unmatched entries and exits through patch corners because they could continue onto two different neighboring patches. This ambiguity leads to problems which are discussed and solved in Section 3.4.

Ideally, $\Psi(e_i, e_j) = \epsilon$ if $\text{cont}(e_i, e_j) = 0$, where $\epsilon > 0$ is a small value indicating low compatibility. Clearly, errors in probability estimation, caused by noisy training data, could prevent this from occurring. Unfortunately, so can perfect training data that consists of nothing but closed, continuous boundaries.

A well-established technique of estimating discrete probabilities is to count the occurrences of the values and then normalize the counts. In order to ensure that 0 probabilities do not occur as a result of sparse data, a Dirichlet prior distribution can be used, and the expectation of the posterior can provide the probability estimates [13]. This is equivalent to initializing all of the value counts to 1. However, this can cause problems with highly improbable neighboring edge pairs. Consider two edge assignments, $e'_i$ and $e'_j$, which never occur in the data, neither singly nor in combination. Assuming that the data points are pairs of neighboring assignments, after N data points are counted,

$$P(e'_i) = \frac{1}{2N}, \quad P(e'_j) = \frac{1}{2N}, \quad \text{and} \quad P(e'_i, e'_j) = \frac{1}{N}.$$

This produces a compatibility value of

$$\Psi(e'_i, e'_j) = \frac{\frac{1}{N}}{\frac{1}{2N}\frac{1}{2N}} = \frac{4N^2}{N} = 4N.$$

As N increases, the model makes these unobserved neighboring assignments more and

more compatible with each other, the opposite of the desired behavior. This failure demonstrates the danger of combining Dirichlet priors with these approximate compatibilities in low-probability cases.

The solution is to modify the compatibility function so that non-continuous neighbors are given value $\epsilon$. For this thesis, $\epsilon = 10^{-10}$, a value chosen by trial and error. This addition makes the neighbor compatibilities

$$\hat{\Psi}_{i,j}(E_i, E_j) = \text{cont}(E_i, E_j)\Psi_{i,j}(E_i, E_j) + (1 - \text{cont}(E_i, E_j))\epsilon.$$

## 3.3 Boundary and image model

In the boundary model, the relationship of the image nodes to the edge nodes limits the algorithm's ability to make some important distinctions. Consider a model trained on a cartoon world in which all objects are uniformly-colored regions. With sufficient training, the image patch likelihood would associate large color derivatives with edges, regardless of the particular patch colors. The edge compatibilities would define the expected rules of edge continuation—edges tend to neighbor similarly-oriented edges. But, although the training regions are all uniformly-colored, the boundary model cannot learn that an edge assignment should never leave green and red pixels grouped in the same segment (Figure 3-4).

In this case, the problem could be solved by expanding the space of edges to differentiate green-blue edges from red-blue edges. But that requires the user to manually choose the correct edge annotations for a data set, and each annotation vastly increases the number of potential assignments at each node.

A superior solution is to combine the edge and patch variable at each location into a single node, as shown in Figure 3-4. This modifies the MRF probability function to

$$P(E, I) = Z^{-1} \prod_{i \in C_1} \Psi_i(E_i, I_i) \prod_{(i,j) \in C_2} \Psi_{i,j}(E_i, I_i, E_j, I_j),$$

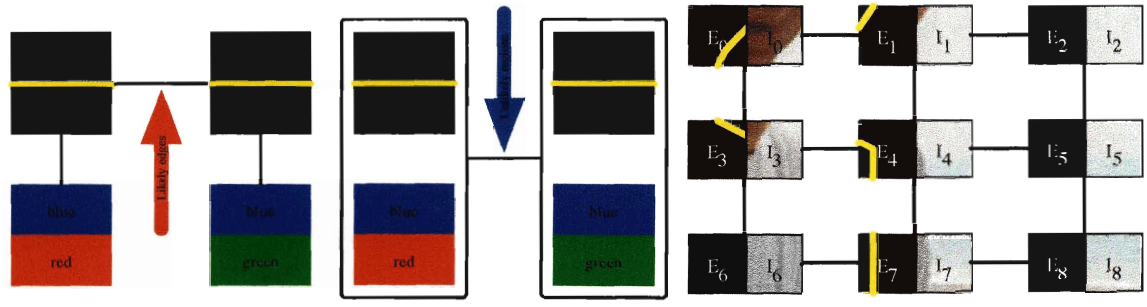



Figure 3-4: Left: Conditionally independent image patches result in the merging of two incompatible regions, but merging image and edge variables allow the model to reject the match based on the region properties. Right: An example of the boundary-image model, combining E and I variables into the same nodes.

the per-node compatibility function to

$$\Psi_i(E_i, I_i) = P(E_i, I_i) = P(I_i|E_i)P(I_i),$$

and the neighbor compatibility function to

$$\Psi_{i,j}(E_i, I_i, E_j, I_j) = \frac{P(E_i, I_i, E_j, I_j)}{P(E_i, I_i)P(E_j, I_j)} = \frac{P(I_i, I_j|E_i, E_j)P(E_i, E_j)}{P(I_i|E_i)P(E_i)P(I_j|E_j)P(E_j)}.$$

The terms that only involve edge variables are equivalent to the original edge compatibility formulation, and the new terms encode information about the image patches conditioned on the edge assignment. Just as in the first model, adding the continuity term

$$\hat{\Psi}_{i,j}(E_i, I_i, E_j, I_j) = \text{cont}(E_i, E_j)\Psi_{i,j}(E_i, I_i, E_j, I_j) + (1 - \text{cont}(E_i, E_j))\epsilon$$

encourages the formation of complete edges.

With these additions, the model can select boundaries depending on the qualities of the regions that they would encode. For example, in the situation shown in Figure 3-4, both neighboring edge assignments are $e_h = (0, 2, 2, 2, 4, 2)$, while the two image patches are $i_{br}$ and $i_{bg}$. Therefore

$$\Psi(e_h, e_h) = \frac{P(e_h, e_h)}{P(e_h)P(e_h)}$$

is likely to be a high value since horizontal edges frequently co-occur. In the new model,

$$\Psi(e_h, i_{br}, e_h, i_{bg}) = \frac{P(e_h, e_h)P(i_{br}, i_{bg}|e_h, e_h)}{P(e_h)P(e_h)P(i_{br}|e_h)P(i_{bg}|e_h)}.$$

If the model has never seen red and green pixels in the same region (i.e. unseparated by a boundary), $P(i_{br}, i_{bg}|e_h, e_h)$ will be nearly 0 and the model will reject the assignment to the edges as unlikely.

It is important to note that although merging the nodes increases the modeling burden because more probability distributions need to be estimated, it does not increase the computational complexity of inference. Since the image patches are visible, there is no increase in the number of hidden variables, and the hidden variables still range over the same sets of possible edge values. Incorporating the image data into the pairwise compatibilities is similar to the construction of the conditional random field (CRF) models proposed by Lafferty et al. [24]. CRFs assume that the distribution of hidden variables conditioned on visible data form an MRF, as opposed to the traditional MRF approach which models a joint distribution of all variables. The MRFs described in this section can also be considered CRFs of the boundary nodes conditioned on the image data.

## 3.4 Segmentation model

Although detecting closed boundaries is equivalent to segmenting an image into regions, representing boundaries with local edge patches produces a solution space containing many non-segmentations; for example, any discontinuities between neighboring edge assignments precludes their use as a region boundary. Ideally, the penalty on discontinuities discussed in the previous models should prevent such assignments from occurring. However, in practice they still occur. There are two possible sources of these errors: loopy belief propagation and the impossibility of enforcing continuity across image corners in our model.

The first source of error is well-known. Belief propagation is guaranteed to produce exact MAP estimates in acyclic MRFs [37], but its application to cyclic graphical models

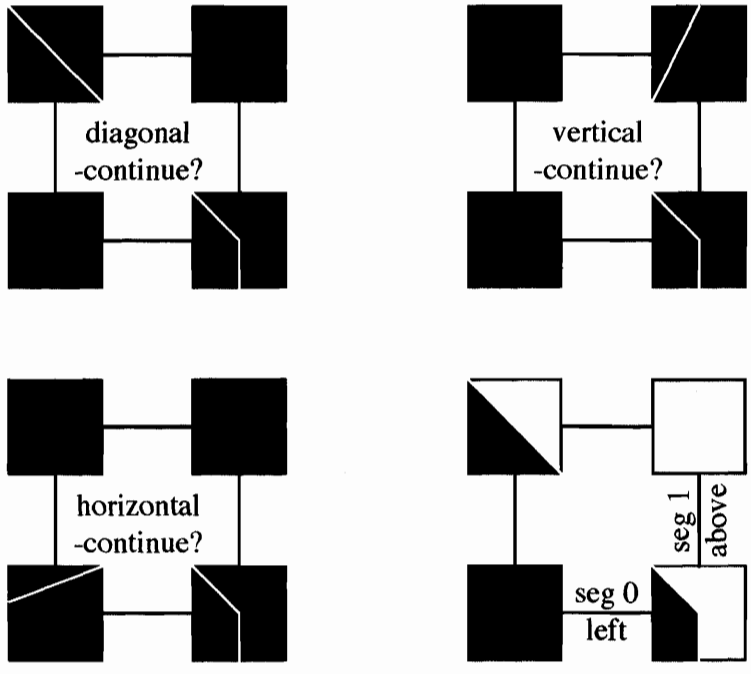



Figure 3-5: With pairwise cliques, it's impossible to predetermine which nearby patch is responsible for continuing an edge that exits at a corner. If the pixels are explicitly assigned labels and label continuity is enforced, the problem is solved.

is known to produce errors in some cases, including algorithmic non-convergence [35]. When it converges, loopy belief propagation finds locally maximal assignments, but even that property is not guaranteed in non-convergent cases [52].

The corner-continuity problem is particular to our model and its reliance on two-element clique potentials. Consider an edge exiting a patch at coordinate (0,0), the upper-left pixel. The edge should be continued, but onto which neighboring patch? As shown in Figure 3-5, it would be acceptable for the boundary to extend onto the patch directly above, the patch to the left, or diagonally. The continuity of this edge is a joint property of the assignment to all four edge patches. Individually penalizing the failure of continuity between the edge in question and any one of its neighbors with a near-zero compatibility ignores the possibility that one of the other neighbors has continued the edge.

Four-element clique potential functions would solve the corner problem, but not the problem of non-segmentation output due to inference errors. Fortunately, it is possible to simultaneously enforce corner continuity and transform the solution space so it only contains segmentations. The training and testing ground truths, produced by background

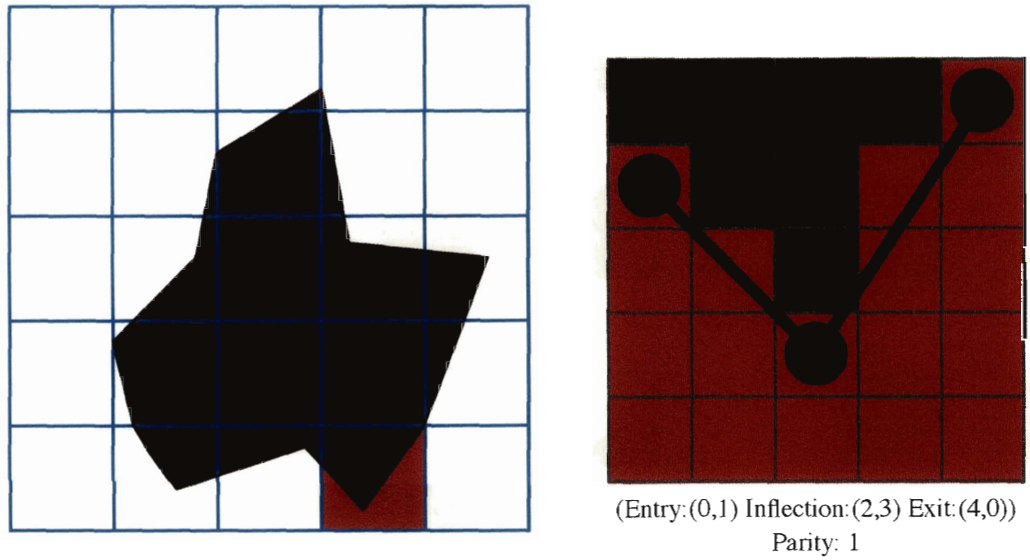



Figure 3-6: Segmentations of isolated objects can be represented by edges representing their boundaries combined with one-bit parities that indicate which segment lies on which side of the boundary.

subtraction, label every pixel as "0" (background) or "1" (foreground). These two pixel labels are sufficient to separate any number of non-overlapping image regions from their surroundings. Therefore, as demonstrated in Figure 3-6, the pixel regions in the training and test data can be represented with our existing edge model, enhanced by assigning a parity to each edge—a single bit indicating which side of the edge is given which region label.

Clearly, this allows any model output to be interpreted as a valid segmentation because all of the pixels are explicitly assigned a region label. It also solves the corner continuity problem. As shown in Figure 3-5, enforcing a continuity rule that neighboring pixels must have the same region label unless they are separated by an edge allows continuity information about the corner-exit to propagate correctly. Since the lower-right edge has a region of white, region 1, pixels above it and a region of black, region 0, pixels below and to the left of it, the requirement that some edge above and/or to the left of it exists to continue the separation is propagated correctly.

In this model, a segment variable, $S_i$, is composed of an edge variable and a one-bit parity value, so $S_i = (E_i, P_i)$. The $E_i$ element is the same type of edge variable introduced

in the previous sections. Adding parity is relatively simple. Unlike the previous modifications, the parity information does not have to be learned, it only affects the continuity of neighboring assignments. The neighbor compatibility function becomes

$$\hat{\Psi}_{ij}(S_i, I_i, S_j, I_j) = \text{cont}(S_i, S_j)\Psi_{ij}(E_i, I_i, E_j, I_j) + (1 - \text{cont}(S_i, S_j))\epsilon.$$

The new version of the *cont* function accepts two neighboring segment assignments and ensures that the pixels along their common boundary have identical labels unless they are separated by edges. If the same possible edges are considered at each patch, parity doubles the number of values to consider at each node, and squares the per-neighboring-pair overhead of the belief propagation algorithm. In practice, this expansion is computationally manageable.

The parities only affect the compatibilities between neighbors; they carry no semantic information indicating which segment label indicates the object and which the background. Therefore, the model described above always has two equivalent MAP assignments, both with the same edge assignments, but with opposite parity bits at every node. This causes trouble with belief propagation inference because the algorithm is unable to break ties between joint map assignments [54], which can result in inconsistent assignments to neighboring nodes. To avoid this problem, when creating an MRF for an image, SANE eliminates all of the parity 1 assignments from the assignment set of the upper-left node. Since the parity at this node is fixed, and its neighbors are forced to be compatible with it, there is no longer a MAP tie and the parity assignments propagate correctly across the MRF during inference.

## 3.5 Multiresolution segmentation

Many computer vision algorithms use multiresolution representations to combine the long-distance image relationships best captured at a low resolution with the precision possible at full resolution. In this vein, the multiresolution version of SANE consists of two linked MRFs, one defined on the full resolution image and the other on a half resolution rendering.

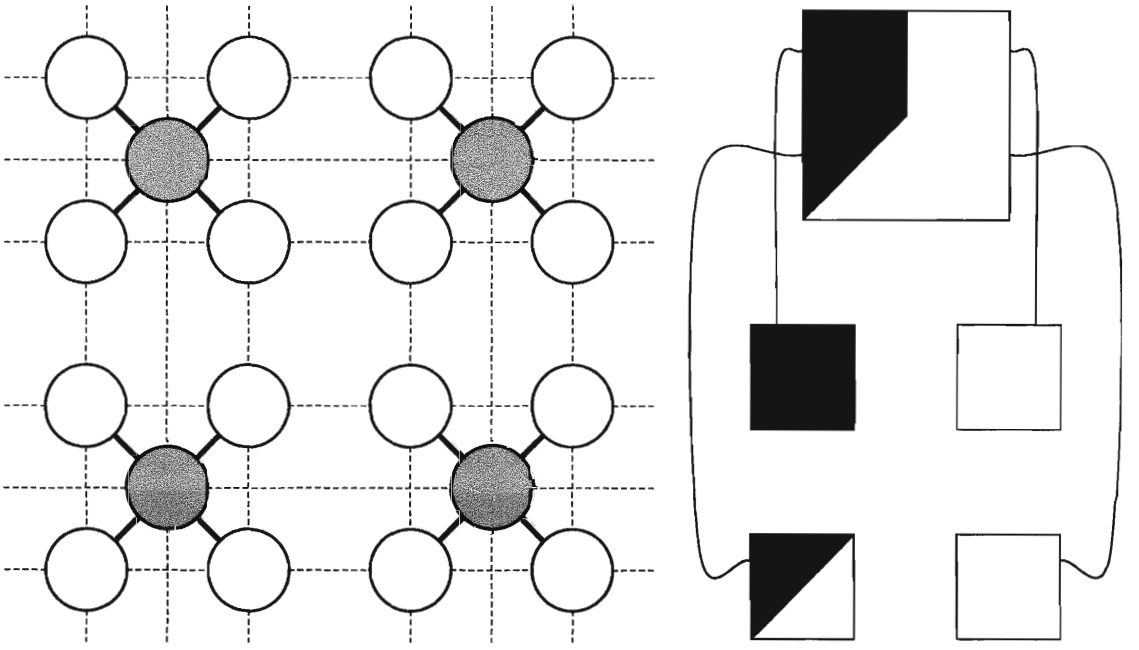


Figure 3-7: Left: In a multiresolution SANE MRF, every half-resolution node (gray) is connected to the four full-resolution nodes that cover the equivalent area in the full-resolution image. Right: The compatibilities are set to require the full-resolution assignments to be compatible with the neighboring half-resolution assignment.

Constructing these two MRFs requires two segmentation models, one trained on full-scale and one trained on half-scale images.

As seen in Figure 3-7, the nodes of the two Markov random fields are intraconnected within each level as described previously and, additionally, each low resolution node is connected to the four full resolution nodes that are responsible for the equivalent area of the full-resolution image. The intralevel compatibilities are the same as they would be for two independent MRFs, and the interlevel compatibilities enforce consistency between each low-resolution assignment and the associated full-resolution assignments.

Figure 3-7 demonstrates that for each half-resolution segmentation assignment, there is a full-resolution node that is responsible for matching each of its four corners. For example, an assignment to the full-resolution upper-left variable $S_{ul}$ implies a 5x5 array of pixel labels $L_{ul}$ and it is attached to a half-resolution node that assigns $L_h$ to its pixels. Consistency demands that the sum of the squared differences between overlapping full and half-resolution labels, $M(L_{ul}, L_h) = \sum_{i,j}(L_{ul}(i, j) - L_h(i/2, j/2))^2$ is minimized. In

order to avoid the complications of learning this relationship, and to allow flexibility for the inevitable minor variations between full and half-resolution assignments, multiresolution SANE assigns the compatibility between the nodes as

$$\Psi(S_{ul}, I_{ul}, S_h, I_h) = \begin{cases} 1 & : \quad M(L_{ul}, L_h) <= 4 \\ \epsilon & : \quad M(L_{ul}, L_h) > 4 \end{cases} .$$

The threshold of 4 was determined by trial and error. The compatibilities for the other three multiresolution relationships, upper-right, lower-left, and lower-right, are defined analogously.

An alternative to these binary compatibilities would have been soft compatibilities similar to the intralevel compatibilities. In order for soft compatibilities to correctly balance the influence of multiresolution information with the other model parameters, they should be learned from training data just as the other pairwise compatibilities were estimated.

## 3.6 Training

Training the segmentation model requires a video of moving objects against a still background. The video is processed with the Stauffer and Grimson [42] or Migdal and Grimson [34] background-subtraction algorithms, which produces a binary image for every video frame in which "1" pixels indicate a moving object and "0" pixels indicate background. Both algorithms work by modeling the appearance of each pixel in each frame with a mixture of Gaussians distributions. Every pixel has a unique distribution whose means and covariances are determined from the statistics of the pixel's color over the previously observed video frames. After a period of initialization, the pixel distributions model the expected color of the background at that location and its expected variance, providing some robustness to noise and lighting changes. Outlier pixels are then labeled as belonging to foreground objects. The Migdal extension uses a Markov random field to improve the output, recognizing that moving objects are spatially and temporally adjacent groups of pixels and modeling this dependence.

The background subtraction output only provides information about moving objects,

so it produces a partially labeled dataset. For example, in a video of moving cars on a highway, the segmentation of disabled cars along the roadside will not be detected because static objects become part of the background. Clearly, training could be negatively affected by presenting the boundaries of such static objects as internal parts of a region and labeling them as empty edges. To alleviate this, a bounding box around the foreground pixels is computed and the remainder of the image is excluded from the training set. If multiple moving objects are present, a single bounding box containing all of them is computed. In any future work employing videos that frequently contain multiple simultaneously moving objects in environments containing many static objects, separate bounding boxes for each object would be more appropriate.

Next, the regions inside the bounding box are extracted from the original video frame and the binary background subtraction image. The background subtraction image is converted into a binary edge image by scanning every row and column and labeling any transition between "1" and "0" as a boundary point.

The segmentation model requires estimates of discrete and continuous probability functions. The edge values are discrete, with 2717 possible values, while the image data are continuous, with a dimensionality and range dependent on the image features used. The edge-value distributions are computed by counting observed instances. The image feature distributions are represented using Gaussian kernel density estimates.

The first task is estimating the edge probabilities $P(E_i)$ and $P(E_i, E_j)$ from the examples. The video processing steps described above result in a binary edge image. Dividing any edge image into 5x5 patches produces a set of edge and edge pair samples. Shifting the patch boundaries right or down by 1-4 pixels produces an additional set of samples from the same image. Rotating each example patch and its immediate surroundings through 360 degrees (by 22.5 degree increments) further increases the number of sample edge and edge pairs harvested from each edge image and imparts rotational invariance to the resulting model.

Because the model should be translationally invariant, the individual edge samples are pooled into a single set without regard to their image locations. The edge pairs can be similarly pooled without regard to location, but they divide into two separate sets—vertical

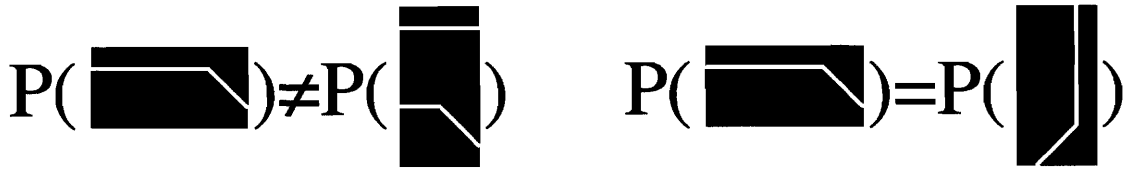



Figure 3-8: The probability of a horizontal pairing of two edges is not necessarily equal the probability of their vertical pairing. Assuming rotational invariance, the probability of a horizontal pairing is equal to the probability of the vertical, rotated pair.

and horizontal neighbors. As demonstrated in Figure 3-8, the statistics of these groups are different, but can be related through a -90 degree rotation of any horizontal pair into an equivalent vertical relationship if the model is rotationally invariant. Transforming all of the horizontal pairs in this manner represents all the pair samples as vertical relationships, concentrating them into fewer categories and making more efficient use of their information.

After marshaling the training examples into their proper orientations, the problem of matching them to the appropriate edge parameterization remains. A 5x5 binary image patch can represent $2^{25}$ possible values, but the parameterization described in Section 3.2 only contains 2717 distinct edges. Therefore, each raw binary edge sample is represented by the most similar parameterized edge. The best match is found by comparing a rendered version of each parameterized edge to the binary image patch in question. The rendering of edge $E$ is a 5x5 matrix $S(E)$ where location $S(E)(i, j) = \exp(-d(E, (i, j)))$ and $d(E, (i, j))$ is the distance between location $(i, j)$ and the nearest location on edge $E$. For these calculations, the integer pixel and edge coordinates are considered to be in the center of pixels. For example, (0, 0) represents the center of the top-left pixel in the patch. Given a binary edge image patch, $B$, the error for parameterized edge $E$ is $\sum_{i,j}(S(E)(i, j) - B(i, j))^2$. In order to discourage bending the local boundary to slightly reduce the error, an edge containing an inflection point is only chosen if its error is at least a 30% improvement over the lowest-error uninflected edge.

In some cases, a 5x5 binary image patch provides too little support to select the correct parameterized edge. Because edges must be continuous with their neighbors in inference, all edge-fittings occur in pairs, choosing the pair of continuous neighboring edges that best

fit the binary edge image data. So, at each location $(i, j)$ the software finds the lowest-scoring continuous matches for $((i, j), (i + 5, j))$, $((i, j), (i - 5, j)$, $((i, j), (i, j + 5))$, and $((i, j), (i, j - 5))$, if they are not off the edge of the image.[2] Each pair is stored as an example of neighboring edge assignments and each $(i, j)$ value as a single edge sample. Any overcounting that results should be relatively evenly distributed and is not a cause for concern.

Given sets of sample individual edges, the next step is to estimate the probability distributions $P(E_i)$ and $P(E_i, E_j)$. Assuming that both of these distributions are multinomial, the counts of the edge values observed in the training are sufficient to estimate them. The sets $C_s = \{c_e\}$ and $C_p = \{c_{e_a,e_b}\}$ contain the counts of all of the observed edges and edge pairs. Excluding edges that were never observed, the posterior estimates are

$$P(E_i = e_i) = \frac{c_{e_i} + 1}{|C_s| + \sum_{C_s} c_e}$$

and

$$P(E_i = e_i, E_j = e_j) = \frac{c_{e_i,e_j} + 1}{|C_s|^2 + \sum_{C_p} c_{e_a,e_b}}.$$

The marginal probability of edge assignment $e_i$ is the number of times it has been observed plus one, divided by the number of possible edge assignments plus the total number of observed edge examples of any type. The marginal probability of an edge pair is also one more than the observed count divided by the number of possible edge pairs plus the total number of edge-pair observations.

Two techniques are used in these estimates to handle the problem of data sparsity. First, the set of possible edge values is restricted to edges actually observed in the training data, which has the effect of reducing the number of edge pair values from $2717^2 = 7382089$ to $|C_s|^2$. Secondly, the counts of each possible single or pair value are initialized to 1, as discussed in Section 3.2. This prevents unobserved edge combinations from receiving a 0 probability. In practice, 0 probabilities often make inference difficult and are disallowed in MRF models by the Hammersley-Clifford theorem (see Chapter 2).

[2]Due to a coding error, the models used to produce the results in Chapter 4 do contain some discontinuous edge-pair samples. However, they are infrequent and should not have a significant impact on the results.

The image probabilities, $P(I_i|E_i)$ and $P(I_i, I_j|E_i, E_j)$ are continuous and estimated by collecting samples using the same procedures described for collecting edge examples. The only differences are that image patches are collected rather than edge values and a separate store of samples is necessary for each edge and edge pair observed. The number of available samples is enhanced by observing multiple rotations of the underlying image data at each image location. Just as horizontal edge pairs can be rotated -90 degrees and counted as a vertical edge pair, their associated image information can be rotated and stored as image samples associated with the appropriate vertical edge pair. Because the image feature space is continuous and multi-dimensional, additional steps are taken to increase the density of estimates. The image pair probabilities are made to be reflection invariant, $P(I_i, I_j|E_i, E_j) = P(R(I_i, I_j)|R(E_i, E_j))$ where $R$ is a horizontal or vertical reflection applied to both the edges and image data. This allows the concentration of samples into a smaller number of edge-pair groups, which produces more densely sampled data and better estimates.

The single and pair distributions are related by

$$P(I_i|E_i) = \sum_{E_j} \int_{I_j} P(I_i, I_j|E_i, E_j) P(E_j|E_i).$$

The two probability distributions could be estimated separately, which would result in a computationally more efficient model because this calculation is relatively expensive. However, the compatibility approximations SANE uses (Section 2.4) appear to be sensitive to mismatches between the pair distributions and their marginals, so this explicit integration and summation is necessary.

Although collecting enough image patches to provide a conditional image distribution for every edge and edge pair is difficult, in some cases there is too much data rather than too little. For example, the empty-edge assignment appears far more often than any other value because it is used to label all non-boundary image regions. Extracting and storing all the empty-edge image patches would quickly overwhelm available memory resources and make the resulting model cumbersome and slow. Therefore, the stored examples for each edge pair are downsampled and collected at a reduced rate if their number exceeds

a specified threshold. In these experiments, if more than 8000 example image patches are stored for a particular edge-pair, 25% are randomly discarded and future examples are only accepted with 0.75 probability. Each time the threshold is passed, samples are discarded and the sampling rate for that combination is reduced to 0.75 of its previous value. Because the training data are so dominated by the empty edge and a few other highly probable assignments, the downsampling procedure is never applied to most edge-pair examples.

The image-pair probabilities are represented by Gaussian kernel density estimates, as described by Russell and Norvig [38]. $F$ is a function that converts an image patch into a real-valued feature vector of dimension $d$, in this case a vector of brightness and, possibly, color averages (see Section 3.2). For each edge pair $(e_i, e_j)$, the associated image pair samples are divided into two sets—the kernel set $I_{k_{ij}}$ and the validation set $I_{v_{ij}}$. The probability function defined by the kernel set is

$$P(I_i = i_i, I_j = i_j | E_i = e_i, E_j = e_j) = \frac{1}{|I_{k_{ij}}| \sigma^d (2\pi)^{d/2}} \sum_{i_k \in I_{k_{ij}}} \exp\left(-\frac{|F(i_i, i_j) - F(i_k)|^2}{2\sigma^2}\right).$$

The only free parameter is $\sigma$, the standard deviation of the Gaussian kernels. That is set during training by choosing an initial value of $\sigma = 1$ and repeatedly reducing it as long as the log likelihood of the elements of the validation set increases and $\sigma$ stays above the minimum value of $\frac{\sqrt{10}}{16}$, which was chosen by trial and error. After $\sigma$ is selected, all the points from both the kernel and validation sets are used as kernel centers in the final estimator. All the image features are preprocessed to have zero mean and unit variance.

## 3.7 Inference

Once all the appropriate probability functions have been estimated from training data, the segmentation of new images can be performed by constructing an MRF and using the belief propagation inference algorithm.

Given an input image $I$ with height $h$ and width $w$, the algorithm divides it into a lattice of non-overlapping 5x5 patches $\{I_{i,j} | i = 0, \ldots, h; j = 0, \ldots, w\}$. Each patch is assigned to a node $N_{i,j} = (S_{i,j}, I_{i,j})$ as described previously. The computational costs of

the belief propagation inference algorithm, described in Section 2.3, are $O(|e|^2)$ where $|e|$ is the maximum number of possible edge assignments at each node. Therefore, using the full set of 2717 possible edge assignments is impractical and each node is instead given a set of the locally most likely edge assignments.

For each node $N_{i,j}$, the algorithm selects the 20 edge assignments that maximize $P(E_{i,j} = e_{i,j}|I_{i,j})$. After these initial sets of possible assignments are selected at each node, the assignments at every neighbor ($N_{i-1,j}$, $N_{i+1,j}$, $N_{i,j-1}$, $N_{i,j+1}$) and every pair of neighbors ($\{N_{i-1,j}, N_{i+1,j}\}$, $\{N_{i-1,j}, N_{i,j-1}\}$, $\{N_{i-1,j}, N_{i,j+1}\}$, $\{N_{i+1,j}, N_{i,j-1}\}$, $\{N_{i+1,j}, N_{i,j+1}\}$, $\{N_{i,j-1}, N_{i,j+1}\}$) are examined to discover neighboring edge assignments that cannot be continued correctly by any currently available assignment at $N_{i,j}$. For each uncontinuable neighbor or paired-neighbor assignment, the most locally likely edge that continues it is added to the set of possible edges. Unlike the *cont* function, continuity of edges that enter and exit through patch corners is considered because it is necessary to provide completing assignments at both of the bordering neighbors. This enhancement process is repeated until it converges and no more edges are added at any node. Finally, the edges (except at $N_0$, see Section 3.4) are paired with each of the two possible parity values which doubles the size of the final set of possible assignments at each node.

The empty edge is always included in the original group of 20 edge assignments chosen at each location, because the algorithm should always be able to infer that no edge is present, even if local data strongly suggests an edge. Additionally, if the procedure described above does not result in the inclusion of the four side (((0,0) (0,4)), ((0,4) (4,4)), ((4,4) (4,0)), ((4,0) (0,0))) and four corner (((0,0) (0,4) (4,4)), ((0,4) (4,4) (4,0)), ((4,4) (4,0) (0,0)), ((4,0) (0,0) (0,4))) edges, they are added to the list. In some cases, their availability might help to close a segmentation boundary in an area with poor local information.

The approximate maximum *a posteriori* (MAP) estimates of the $S$ variables are calculated via the max-product belief propagation algorithm described in Section 2.3. There is no guarantee of convergence for loopy MRFs. Therefore the segmentation algorithm stops belief propagation after 200 iterations and takes the approximate MAP estimate available at that point.

Due to the errors that can result from applying loopy belief propagation, the results

of this procedure are not necessarily the most probable joint variable assignment nor are they necessarily consistent. For example, it is possible that two neighboring nodes might be given assignments that do not obey edge and segment continuity. In the cases where belief propagation diverges, the results can sometimes be a checkerboard pattern of segmentation labels that alternate parity with each iteration. Therefore, results can sometimes be improved by adding Besag's iterative conditional modes (ICM) algorithm [2] as a post-processing step. ICM improves an existing assignment by repeatedly scanning across the nodes and making any change that improves the joint assignment's probability. After belief propagation ends, a restricted ICM is applied that only modifies the parity labels to improve their consistency given the selected edges.

Another method of combating divergence or suboptimal estimates is to alter the belief propagation step size, as described in Section 2.3. In Chapter 4, the results using a standard step size of 1 are reported, along with the results produced by trying ten Heskes [16] step sizes ranging from 0.1 to 1 and choosing the result with the maximum MRF configuration value.

# Chapter 4

# Experiments

## 4.1 Experimental purpose

The experiments in this chapter illuminate the performance of SANE on four data sets, and demonstrate its advantages over the standard Normalized Cuts segmentation algorithm [40] and the Martin trained boundary detection algorithm [32]. They demonstrate the relative value of different versions of the algorithm, and measure its ability to infer segmentations on a test set with different lighting or environment than its training set.

## 4.2 Data sets

SANE was tested on four data sets (Figure 4-1). The *traffic* data consists of a single video recording of cars traveling down two roads. The original video was spatially split to produce two videos, each covering traffic on a different road. The *walking* video shows the author walking back and forth in front of a whiteboard. Two other videos of many people individually walking across a room form the *mwalk1500A* and *mwalk1500C* data sets. These sequences were both filmed in the same room, with the same camera position and nearly the same subjects, but each has different lighting conditions.

For the traffic videos, the background subtractions, which provide "ground truth" segmentations for training and testing were produced by the Stauffer and Grimson algorithm [42]. The background subtractions for the other videos were produced by the Migdal and Grim-

Figure 4-1: Sample images and background subtraction masks from each of the four data sets. From left to right: mwalk1500A, mwalk1500C, walking, and traffic.

son [34] algorithm. These two methods are described in Section 3.6.

Each data set was divided into training and testing examples. On the traffic data, the video of the left road became the training set, and the video of the right road became the testing set. The other videos were split temporally into training and testing sets — all frames prior to a certain time index were for training and the remainder for testing.

Discarding the frames in which background subtraction detected no moving objects, the traffic data contains 1432 training frames and 2285 testing frames, the walking data contains 401 training and 200 testing frames, and both the mwalk1500A and mwalk1500C data sets contain 1201 training and 1200 testing frames. Three random subsets, each containing 200 training frames and 40 testing frames, were uniformly drawn from each data set. SANE and Martin models were trained on each of the training subsets and then were tested on the corresponding testing subsets, as was Normalized Cuts. The reported results

are the average performance over the three subsets, and error bars in the graph represent the sample standard deviation over the three subsets, calculated as the square root of the unbiased sample variance.

Throughout the results section, precision, recall, and the f-measure are used to measure the quality of the results. Precision (P) is the fraction of inferred object pixels that match the background subtraction, recall (R) is the fraction of background subtraction pixels that are detected, and f-measure, $F = 2PR/(P + R)$, combines both values. A perfect f-measure is 1 and getting every detection wrong produces an f-measure of 0. Randomly assigning "object" and "background" labels with equal probability would produce a recall of 0.5, but the precision would depend on the relative frequency of the two labels in the background subtraction. For example, if the background subtraction was evenly divided between the two labels, random assignment would produce a precision and f-measure of 0.5. But if only 1 pixel in a large image was truly labeled "object" the random assignment's precision and f-measure would be nearly 0.

The background subtraction is corrupted by noise, so even an apparently perfect segmentation might not score perfectly. In the background subtraction images, "1" indicates an object and "0" the background. SANE also outputs "1" and "0" segmentation labels, but does not determine which one belongs to the object. When measuring error, the label that achieves the highest f-measure is considered the "object" label. SANE's goal is a segmentation that separates objects from their surroundings, so this method accurately measures its success in that mission.

Both per-image and per-pixel statistics are reported. Per-image statistics are computed by finding the precision, recall, and f-measures for each image in a test set and then reporting their average value across the set. They are useful because they measure the performance that can be expected given a random image drawn from the testing set, and they reflect that the segmentations of different images are computed independently. However, it is also often true that larger images are more difficult to segment correctly because they are more likely to contain multiple objects or potentially distracting non-object boundaries. Per-pixel statistics emphasize performance on larger images. These values are computed for a set by cumulatively counting all of the pixel-wise successes and errors across all test

images. Per-pixel statistics are particularly useful for comparing SANE to the Normalized Cuts algorithm. Apart from the initial presentation of SANE results, all figures report per-image statistics unless the per-pixel results are useful for illustrating a particular point.

## 4.3 SANE results

There are four types of SANE model and four inference algorithms. Figures 4-2, 4-3, and 4-4 show the f-measures, precisions, and recalls for all sixteen combinations on the four data sets. The four model types are:

- brightness-only, single resolution (bright)
- brightness-only, multi-resolution (bright,mres)
- color, single-resolution (color)
- color, multiresolution (color,mres).

The brightness and color features are discussed in Section 3.2 and the multiresolution models are described in Section 3.5. The four inference algorithms are:

- standard belief propagation (bp)
- belief propagation followed by parity-flipping iterative conditional modes (bp,picm)
- belief propagation using step-size search (bp,ss)
- the combination of all three methods (bp,ss,picm).

Details about the algorithms can be found in Section 3.7. Note especially that the bp,ss algorithm chooses the best step size by maximizing the MRF's configuration value, not by comparing its outputs to ground truth.

The algorithm was more successful at segmenting the walking and traffic data sets than the mwalk sets. This is not surprising since the walking and traffic sets feature objects against relatively uncluttered, untextured backgrounds. The cars demonstrate less internal variation than walking people, and it's easier to learn to segment a single walking person

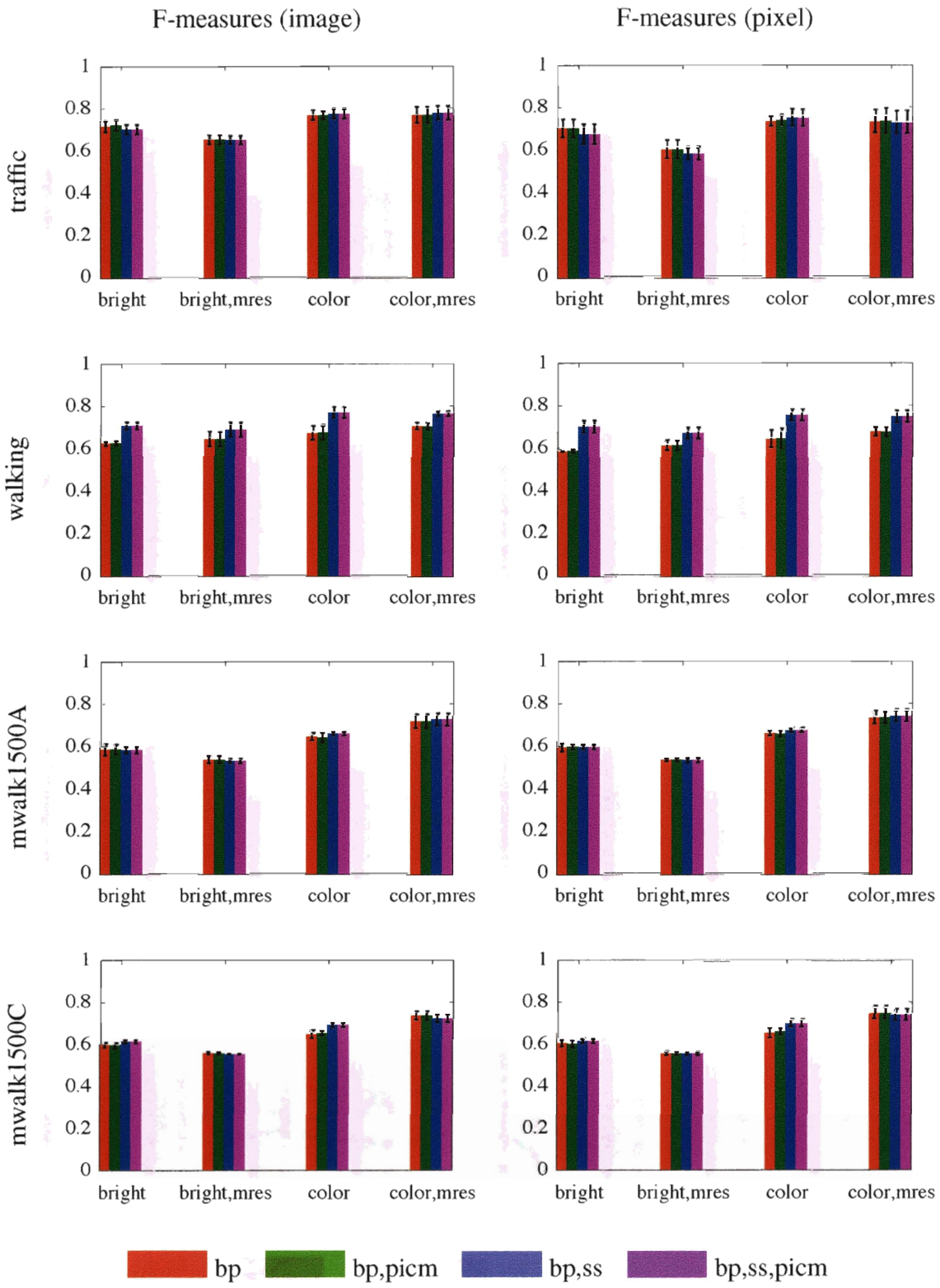



Figure 4-2: The f-measures of SANE models and inference methods on the data sets.

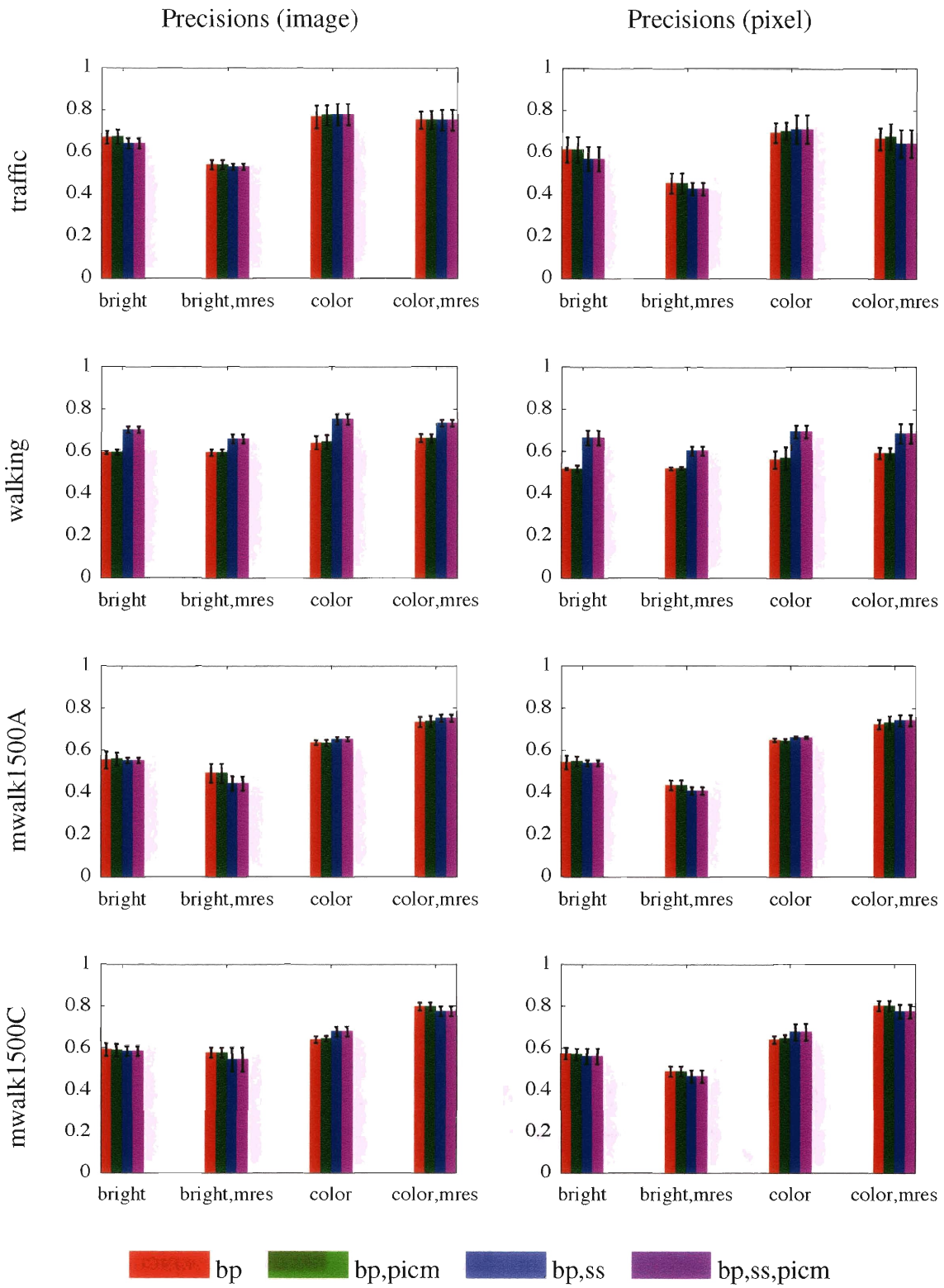



Figure 4-3: The precisions of SANE models and inference methods on the data sets.

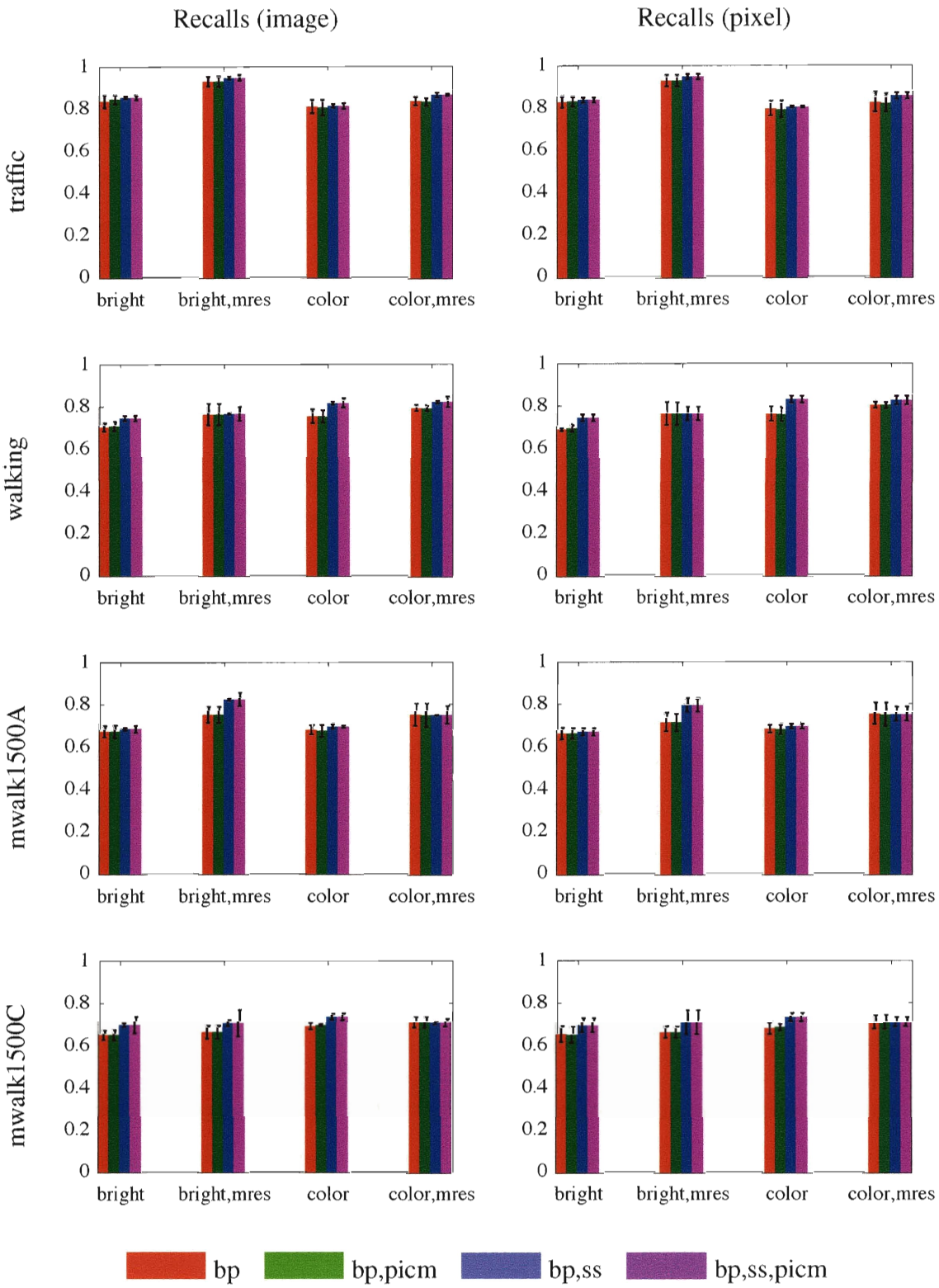



Figure 4-4: The recalls of SANE models and inference methods on the data sets.

(walking) than multiple people (mwalk), some of whom do not appear in the training data. Despite these variations, performance was remarkably consistent. The highest average per-image f-measure on each data set ranged from 0.728 (mwalk1500A) to 0.780 (traffic). In almost all cases, recall values exceeded precision. This indicates that it is more common for object pixels to be overdetected than underdetected. In some cases, the extra pixels might constitute non-moving objects, such as the highway lane lines in the traffic images, that the performance measure marks as incorrect. In other cases, including some seen in Figure 4-5, part of the object's boundary is not detected, possibly due to poor local data or background clutter, which causes the object label to bleed across the image. Both types of errors harm precision without affecting recall. Multiresolution models using only brightness features almost universally underperformed their single resolution counterparts, trading too much precision for increased recall. The half-resolution layer's patches cover greater image area, and perhaps that reduces the effectiveness of the brightness features, since their average distances from an object boundary will subsequently increase. Color multiresolution models were slightly better or roughly equal to color single resolution models; they improved precision on the mwalk sets, and provided some benefit on the walking data when step-size search was not used. Figure 4-6 contains several instances in which multiresolution fixed the label bleeding problem that can lead to low precision results. At coarse resolutions, the patch color averages might serve as a useful regional property—representing, for example, "this is a red region, and should not be merged with a blue region"—more robustly than the finer-grained brightness averages. Single resolution color models also tended to improve precision—in all of the sequences at least some objects, such as the red cars, had strong color differences from their surroundings.

None of the four inference algorithms consistently beat the others. Step-size search made a significant improvement on all the walking results, but not on the other data sets. It never appears to make results significantly worse, so it can be considered the best inference algorithm. The parity-flipping ICM step, which was originally added to fix some of the "checkerboard" patterns that appear when belief propagation diverges (for examples, see Figure 4-6) never made a substantial difference. This is not surprising, since it can easily make a checkerboard worse by making a half-wrong area completely mislabeled if it picks


Figure 4-5: The five best and five worst color, multiresolution SANE results on the four data sets. All the results were computed using the step-size search (bp,ss) method.

the wrong square for its first flip. Perceptually, a checkerboard might appear to be a larger error than mislabeling a region, particularly if the region boundaries match well with an object boundary. The checkerboard results are also frequently unstable—continuing to run belief propagation causes the squares to oscillate between the two labels. A future error measure might penalize this instability directly because it truly represents a failure to make either classification.

Figure 4-5 presents the five best and five worst results for the color, multiresolution models of each data set, inferred using the step-search method. Unsurprisingly, clear foreground-background separations, such as a single dark car or an individual surrounded by white pixels, appear in the best collection, while multiple objects and greater clutter produced some of the poorer output. Many of the bad examples contain objects at the edge of the frame, or only small object pieces, such as an arm or leg. These cases might indicate that the absence of part of the object boundary makes inference difficult, perhaps because that situation is not well-represented by the learned shape model or because a small input image make the creation of the half-resolution MRF level difficult. Weiss analyzed the performance of loopy belief propagation by considering the number of times evidence is over-counted [51]. Every message emitted by a node on a loop edge travels through the network, is multiplied by other messages, and returns to its source. The size and number of loops attached to a node at the edge of an MRF are different than they are for a central node, which changes the distribution of message over-counting and might cause some of these problems with objects at the edge of frames. Only the traffic data contains multiple moving objects in some example frames, and all of the five worst examples contain two full or partial cars. Because there are only two labels, segmenting multiple objects correctly requires a relatively unforgiving alternation between foreground and background labels. Failing to close the boundary of one car may make it extremely difficult to correctly label the other. The multi-label extensions described in Section 5.3 may alleviate this problem.

Discarding ICM, which did not significantly affect the results, the color, multiresolution and step-size search variations all fixed some checkerboard patterns, the most visible effect of belief propagation non-convergence. In Figure 4-6, the five most improved examples from each technique contain at least one example of a fixed checkerboard. In other cases,

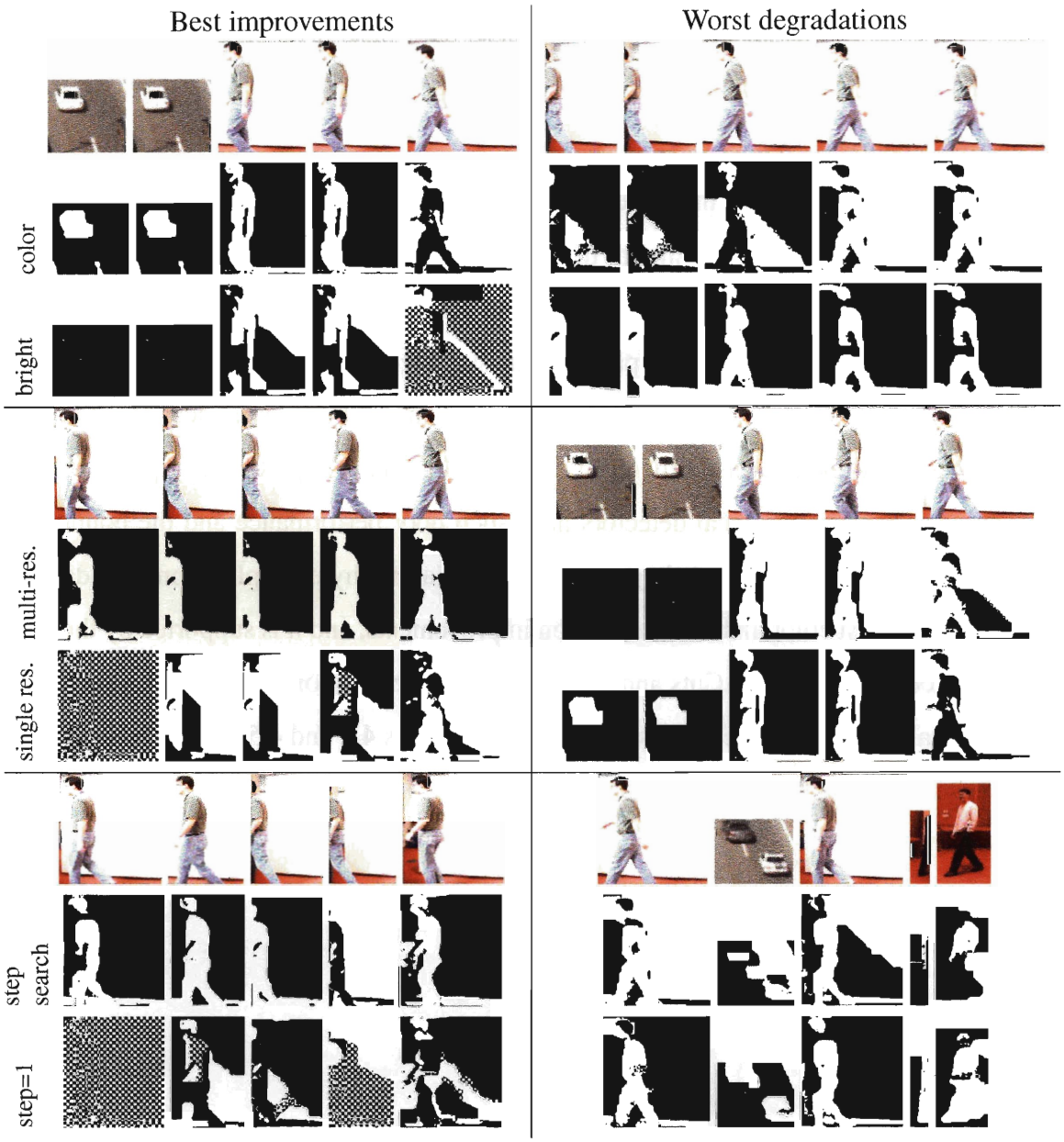



Figure 4-6: Across all the test examples, examples demonstrating some effects of applying color models, multiresolution models, and step-size search. For each variation, the five most improved results are shown, along with the five most degraded results across all test examples.

the changes improved the detection of object boundaries and the ability to distinguish them from internal borders. However, the modifications also made some of these boundary determinations worse, as can be seen in the degraded examples. The model might benefit from more sophisticated local features, which could reduce reliance on belief propagation accuracy, or a higher-level shape model, which could use stronger priors to overcome uncertain image information.

Unsurprisingly, many of the largest effects from each of these techniques occurred on the walking data set, corresponding to the statistics in Figure 4-2. All the best step-search improvements came from examples in this data set in which a fixed step size of 1 produced at least a small checkerboard pattern. The step-size search was designed to eliminate these types of divergence. The walking data set, which exhibits a great deal of saturation, making the figure boundaries difficult to detect locally, may be particularly prone to non-convergence because the local detectors have such poor performance and the boundaries are frequently washed out. This hypothesis also explains why examples from this data set are among the best color and multiresolution improvements, and it is supported by the poor performance of Normalized Cuts and the Martin edge detector on this data set, which both rely on local boundary detection, as described in Sections 4.4 and 4.5.

## 4.4 Comparing SANE and Normalized Cuts

Comparing SANE to Normalized Cuts reveals that general-purpose segmentation does not always solve the object segmentation problem well. Despite designing the comparison to favor Normalized Cuts, the best it can do is match SANE's performance on some of the data sets. In some significant situations it fails to solve the object segmentation problem as well as SANE. Although Normalized Cuts is an excellent generic image segmentation algorithm, the object segmentation task presents unique challenges that SANE handles better.

Normalized Cuts [40] is a well-known general-purpose segmentation algorithm. It computes a matrix that represents the similarity between all pairs of image pixels and then segments the image by solving a related eigenproblem. The implementation used for these experiments, developed by Jianbo Shi and his students [8], measures the difference between image pixels by searching for the presence of brightness contours found by the Canny edge detector [6] between every pair of pixels. It was not trained or parameter-tuned for the test data set.

Normalized cuts breaks an image into a user-specified number of regions. Comparing the output to an object-background segmentation, which only uses two labels, requires transforming the output into that form. In this comparison, the goal is to determine if the Normalized Cuts algorithm has produced segments that match the object segments well. If an image contained only one object, as in Figure 4-7, and Normalized Cuts produced three regions, the region that produced the best f-measure compared to the motion segmentation would be selected as the object region. The other two would be merged to represent the background.

No test image contains more than 3 moving objects, so the subset of 3 or fewer Normalized Cuts regions that maximizes the f-measure are chosen to represent the foreground, and all other segments will be taken to represent the background. Dividing an object into many pieces violates the goals of object segmentation, so the foreground subset must only contain segments that do not touch one another. No test images contain touching moving objects.

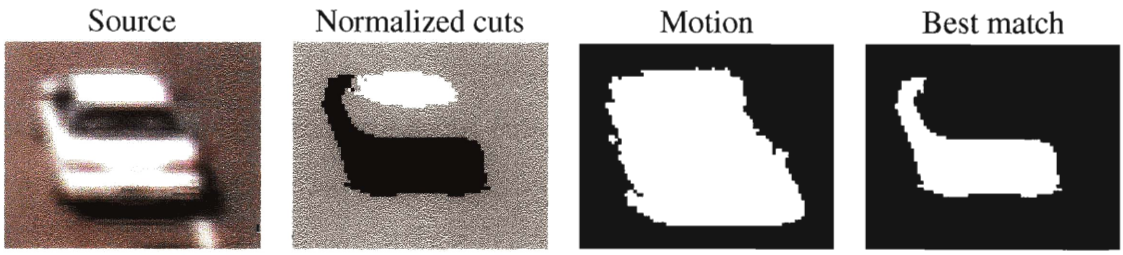



Figure 4-7: A source image, a Normalized Cuts segmentation of that image, the ground-truth motion segmentation, and the selection of the foreground and background segments that best match the motion segmentation.

This comparison scheme is very favorable to Normalized Cuts. It allows Normalized Cuts use of the ground truth to pick the best segments, which would be impossible in a real application where segmentations are unknown and inaccessible. For a setting of two regions, this corresponds to SANE's ability to pick which label best matches the motion objects, but for more than two regions it constitutes a large advantage for Normalized Cuts. Furthermore, it does not penalize the algorithm for subdividing the background, while SANE is punished for doing the same thing. Because the Normalized Cuts code only uses grayscale images, SANE is also restricted to brightness features. Since multiresolution only harms the performance of brightness-only models, only comparisons to the single-resolution brightness model are presented. Color features improved SANE's output on all the data sets, but it would have been unfair to graft crude color features onto the publicly available Normalized Cuts code for the purposes of this comparison. Presumably color features could improve Normalized Cuts performance too, but those are best added by researchers with expertise in that code, since badly-designed color features could unfairly tip the scales towards SANE.

In Figure 4-8, it is clear that Normalized Cuts closely matches SANE's performance on the two mwalk data sets. On the traffic data, the per-image and per-pixel comparisons diverge in interesting ways. With two regions, Normalized Cuts matches or slightly improves on SANE's per-image performance, but it performs much worse than SANE in the per-pixel statistics. Matching the best SANE per-pixel performance requires making 4 or 5 regions, at which point the per-image performance starts to decline. Figures 4-9 and 4-10 reveal that this improved per-pixel f-measure is only achieved at the expense of a large

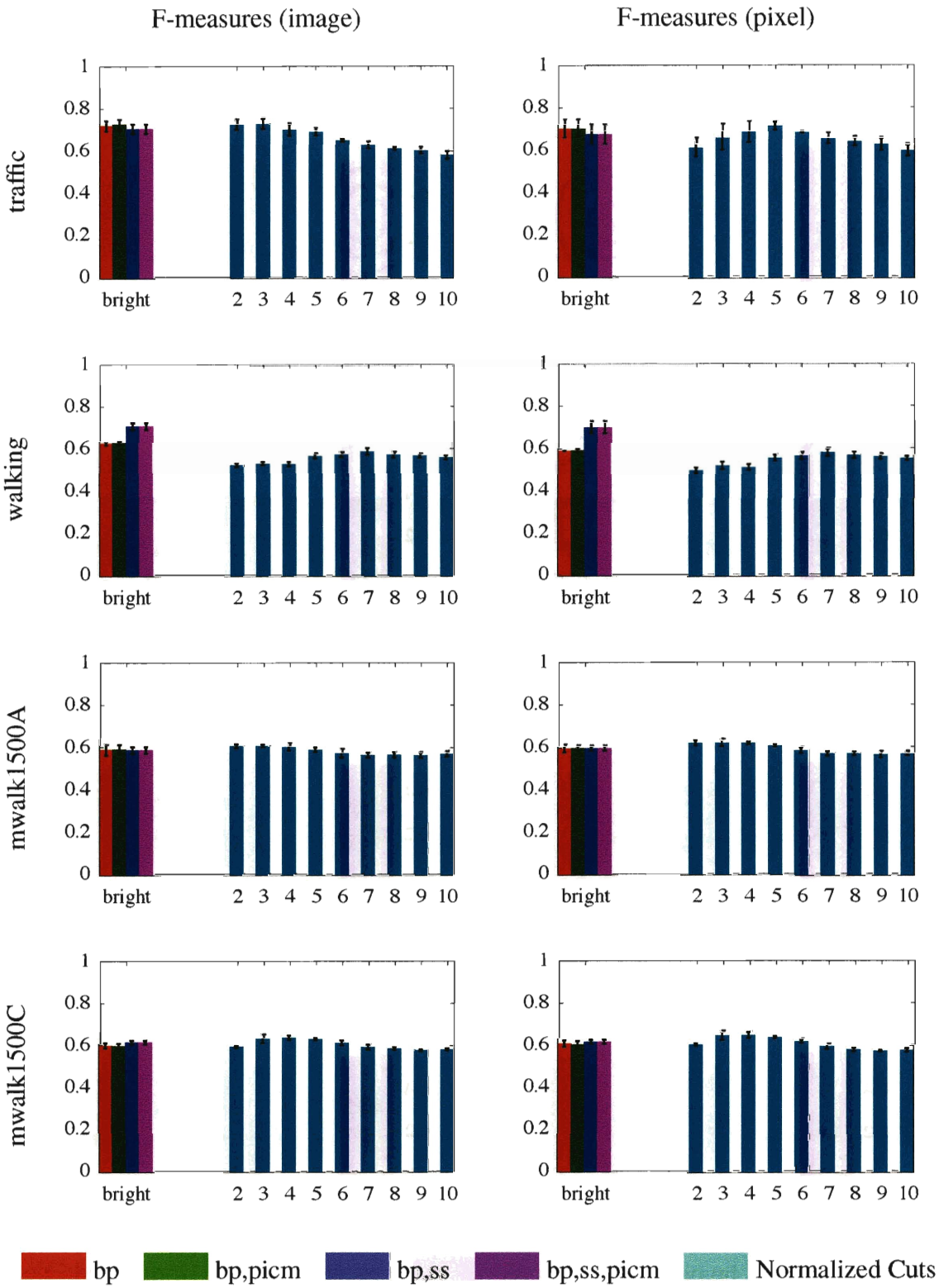



Figure 4-8: The f-measures of brightness, single-resolution SANE and Normalized Cuts set for different numbers of segments.

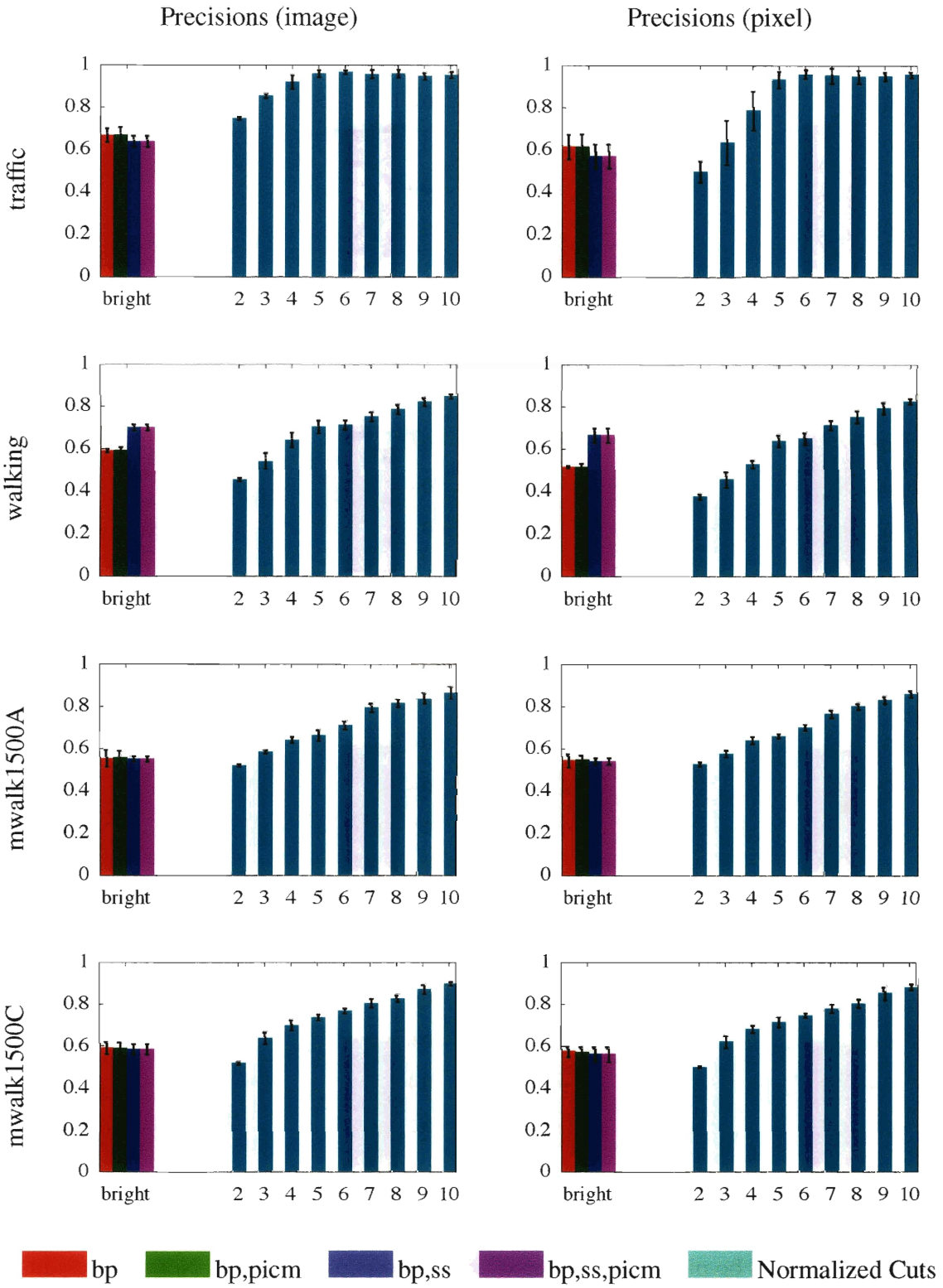



Figure 4-9: The precisions of brightness, single-resolution SANE and Normalized Cuts set for different numbers of segments.

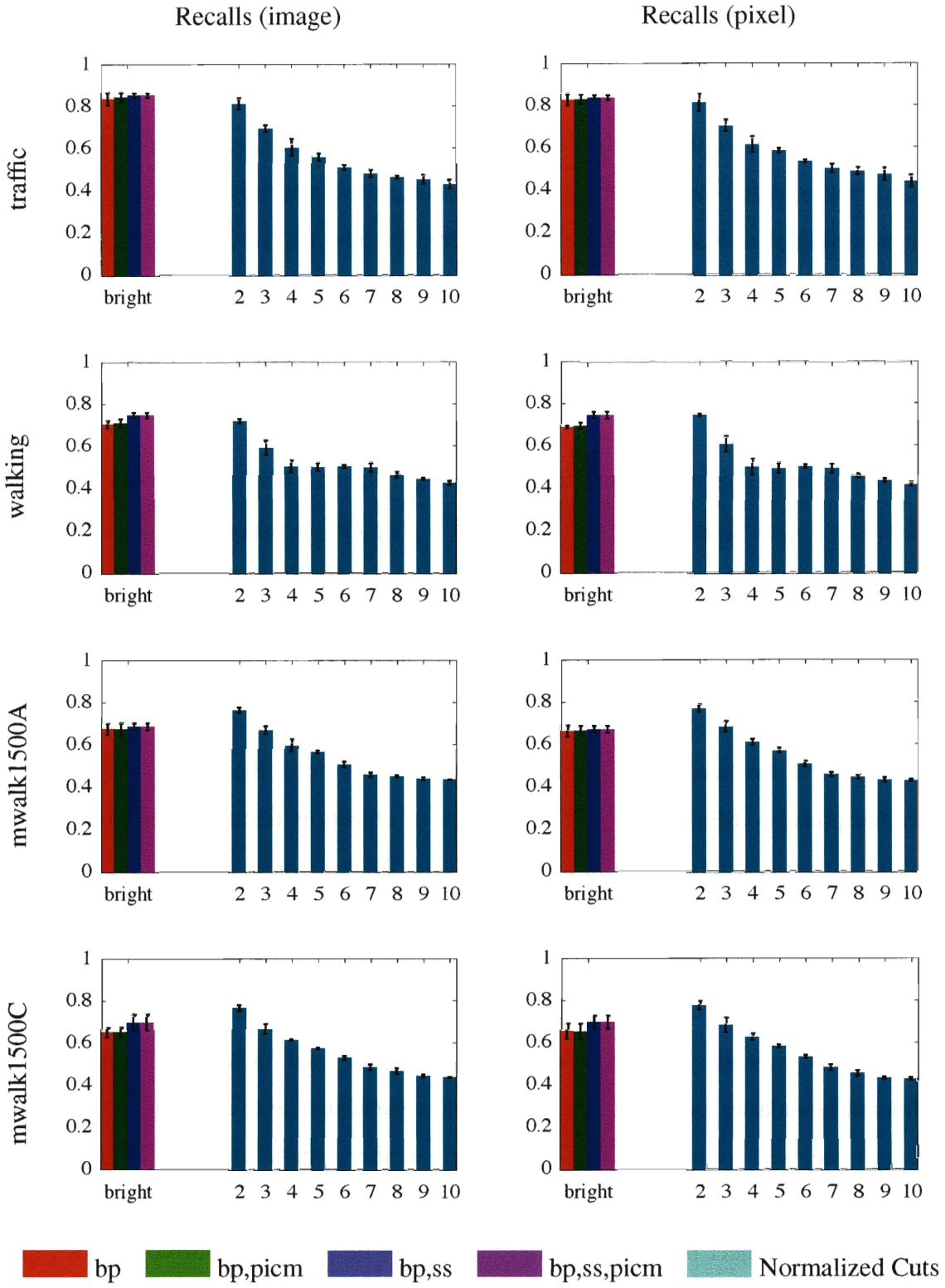



Figure 4-10: The recalls of brightness, single-resolution SANE and Normalized Cuts set for different numbers of segments.

reduction of recall. With 3 regions, Normalized Cuts's recall is roughly 0.1 worse than SANE and it decreases to about 0.2 worse than SANE with 5 regions. At the same time, its precision soars towards 1.

The examples in Figure 4-11 reveal that with two Normalized Cuts regions, SANE does better on many larger, multi-object examples, which, not surprisingly, do not segment well with just a single cut. As the number of cuts increases, Normalized Cuts improves relative to SANE on these larger examples, but its recall numbers suffer because small, single object examples are subdivided. At the same time, over-segmentation improves precision by allowing the performance measure to select segments that have minimal overlap with non-object pixels.

On the walking data, no Normalized Cuts setting matched SANE's performance. Figure 4-12 makes it clear that Normalized Cuts does a poor job on many examples because it considers the external boundaries of the person to be less significant than his internal visual boundaries, or because it fails to detect pieces of the boundary, perhaps due to the saturation of the pixels bordering the whiteboard.

These data indicate several limitations of using Normalized Cuts on the object detection problem. First, the need to prespecify parameters that control the number of segments generated presents difficulties since the number of objects is unknown and might vary, requiring different settings for different input. Secondly, both traffic and walking examples demonstrate that choosing the number of segments that matches the number of objects can still lead to poor performance if non-object boundaries are considered to be more visually significant. The first problem demands an algorithm with some ability to self-determine the number of possible objects and the second indicates a need for learning models of the significant boundaries in an environment or for a particular task; a manually specified, generic model will not always suffice. Although another generic segmentation algorithm might select a reasonable number of regions automatically, the problem of distinguishing object from non-object boundaries can probably only be practically solved by a learned model, as in the SANE algorithm.

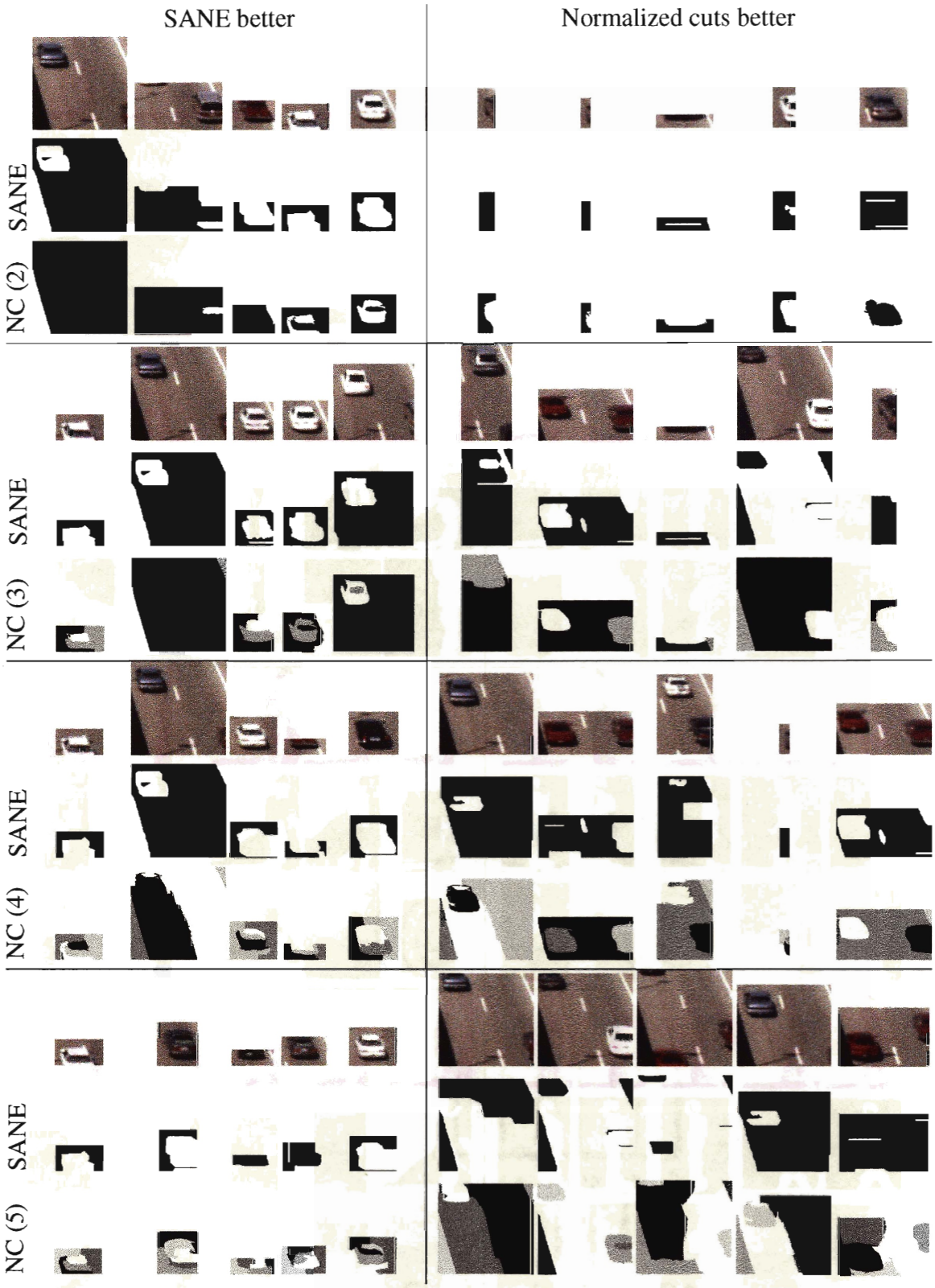



Figure 4-11: Traffic examples in which SANE most outperformed Normalized Cuts, and vice-versa, for different numbers of Normalized Cuts regions.


Figure 4-12: Walking examples in which SANE most outperformed Normalized Cuts, and vice-versa, for different numbers of Normalized Cuts regions.

## 4.5 Comparing SANE and Martin's edge detector

Martin et al.'s learned boundary detector [32] is a good contrast to the SANE approach. Like SANE, the Martin algorithm can be trained on a database of example segmentations, but unlike SANE it attempts to discover the best local boundary detector without learning a shape model or propagating information to neighboring sites. Martin detectors attempt to classify each point as "boundary" or "non-boundary" based on the image data in that point's immediate vicinity. Classification decisions are made independently at each image location. Thus the trained Martin detectors, which outperform many standard edge detection algorithms on the Berkeley Segmentation Database, rely on a strong local detection model, while SANE uses weak local detectors and relies on shape information for joint boundary determination. The Martin algorithm also strongly contrasts to Normalized Cuts because it focuses on modeling local boundary properties while Normalized Cuts is concerned with the joint similarities and differences of entire pixel regions.

The comparison is difficult because the Martin algorithm outputs a map of boundary detections, rather than a segmentation. Each pixel in the boundary map has a value between 0 and 1 that indicates the likelihood of it being a boundary pixel; the map must be thresholded to produce a binary boundary detection. SANE outputs are converted to binary boundary images for the purpose of these comparisons by labeling all horizontal and vertical transitions between segment labels as boundary locations.

The Martin and Fowlkes implementation of the edge detection training and testing algorithms [31] was used to compare the Martin detectors to SANE. Five detector classes were trained on the SANE data sets:

- brightness gradient (bg)
- color gradient (cg)
- texture gradient (tg)
- brightness and texture gradient (bgtg)
- color and texture gradient (cgtg).

The cg detector contains no brightness features, but the cgtg detector does. Each detector uses image data from a fixed, prespecified radius around the location to be classified.

For these experiments, the Martin detectors were trained on each SANE training subset, just as the SANE models were, and tested on the corresponding testing subsets. In the original implementation, the detector radius varied according to the size of the input image. In the Berkeley Segmentation Database, the image size indicates the scale of the image boundaries. This is not consistently true in the SANE data sets, and fixed radii performed better in early experiments. The bg, cg, and tg detectors were trained with radii ranging from 1 to 10. The original Martin code used a 1:2 ratio of bg radius to tg radius in the bgtg detector, so these detectors ranged from 1:2 to 10:20. For the same reason, the brightness, color, and texture radii in the cgtg detector ranged from 1:2:2 to 10:20:20.

Comparing boundary images is a difficult task. Simply overlapping two binary boundary maps and counting matches is not useful because a small misalignment can cause a complete mismatch, and the resulting score could indicate that two very similar boundaries are extremely different from one another. Martin introduced a method of comparing two boundaries by solving a bipartite graph matching problem. Given a binary detected boundary image and binary ground-truth boundary image, each boundary pixel in each image is considered as a graph node. The goal is to match each detected node to a unique ground-truth node. In turn, each ground-truth node can match at most one detected node. If a detected node matches a ground-truth node, it is considered to be an accurate detection. If it cannot be matched, it is inaccurate. The distance between two matched nodes cannot exceed a specified maximum. As with the detector radii, the original implementation varied these match distances relative to image size, but the modified implementation does not. Again, the reason is that the sizes of our images do not indicate the scale of the boundaries.

In order to transform the soft boundary maps of the Martin detectors into binary boundary maps, the Martin error-measurement code chooses the threshold that maximizes the f-measure across the testing set. In a deployed application that required binary boundary detection, this threshold would need to be chosen during training. This extra degree of freedom benefits the Martin detectors in any comparison to SANE.

The threshold also interacts with the maximum match size to confer another potential

advantage. Consider an infinite maximum matching distance. Given two binary boundary images, the error between them simply depends on the number of boundary pixels in each. If they both contain the same number, the algorithm scores the detection perfect. The Martin algorithm outputs a soft boundary map and chooses the threshold that produces the best score. If the threshold increments are fine enough, and no two detections are given identical values, it is possible to pick a threshold such that the binary detection map contains exactly the same number of pixels as the ground truth map. This does not hold true when the threshold is chosen across a set of maps, but it suggests that larger match maximums favor the Martin algorithm over SANE.

The results in Figures 4-13, 4-14, 4-15, 4-16, 4-17, and 4-18 make it clear that SANE outperforms Martin by virtue of its higher precision on these data sets. While the Martin detectors have little trouble finding object boundaries, they also detect many other non-object boundaries in the process. In all four data sets, the color, multiresolution SANE f-measures outperform all Martin detectors at all feature radii using the strict matching distance of 1. As the match maximum distance relaxes to 3 and 5, the Martin results improve, largely due to improved recall. Only on the traffic data at match radii 3 and 5 do the Martin f-measures match SANE's performance. Figure 4-19 contains examples in which SANE's performance was superior to cgtg with feature radius 5 (one of the overall best-performing Martin detectors) using match maximum distance 3. Most of the Martin results are riddled with non-object boundaries compared to the SANE output. Similarly, the examples in Figure 4-20, demonstrate that extreme SANE recall failures are responsible for many of the cases in which cgtg is superior.

Much of the performance disparity could derive from the data sets that each algorithm was originally designed for. Although the Martin detectors were retrained on the SANE data sets, the Berkeley Segmentation Database contains low-noise, high-resolution images, and much of the success of the Martin detectors on that set appears to be due to the use of the sophisticated texture gradient features [32]. The SANE sample images have much lower resolution and much poorer quality. There is less texture information available in each image, and boundaries are not as sharp. SANE's local edge detectors are poor, but the shape model compensates. Also, the segmentations in the Berkeley database frequently include

visually significant internal object regions, and it's unsurprising that a detector designed to detect all significant image boundaries will have difficulty learning to ignore those instances in which they do not correspond to object boundaries. The shape information stored in the SANE model appears to be more useful on this data than the sophisticated local Martin detectors.

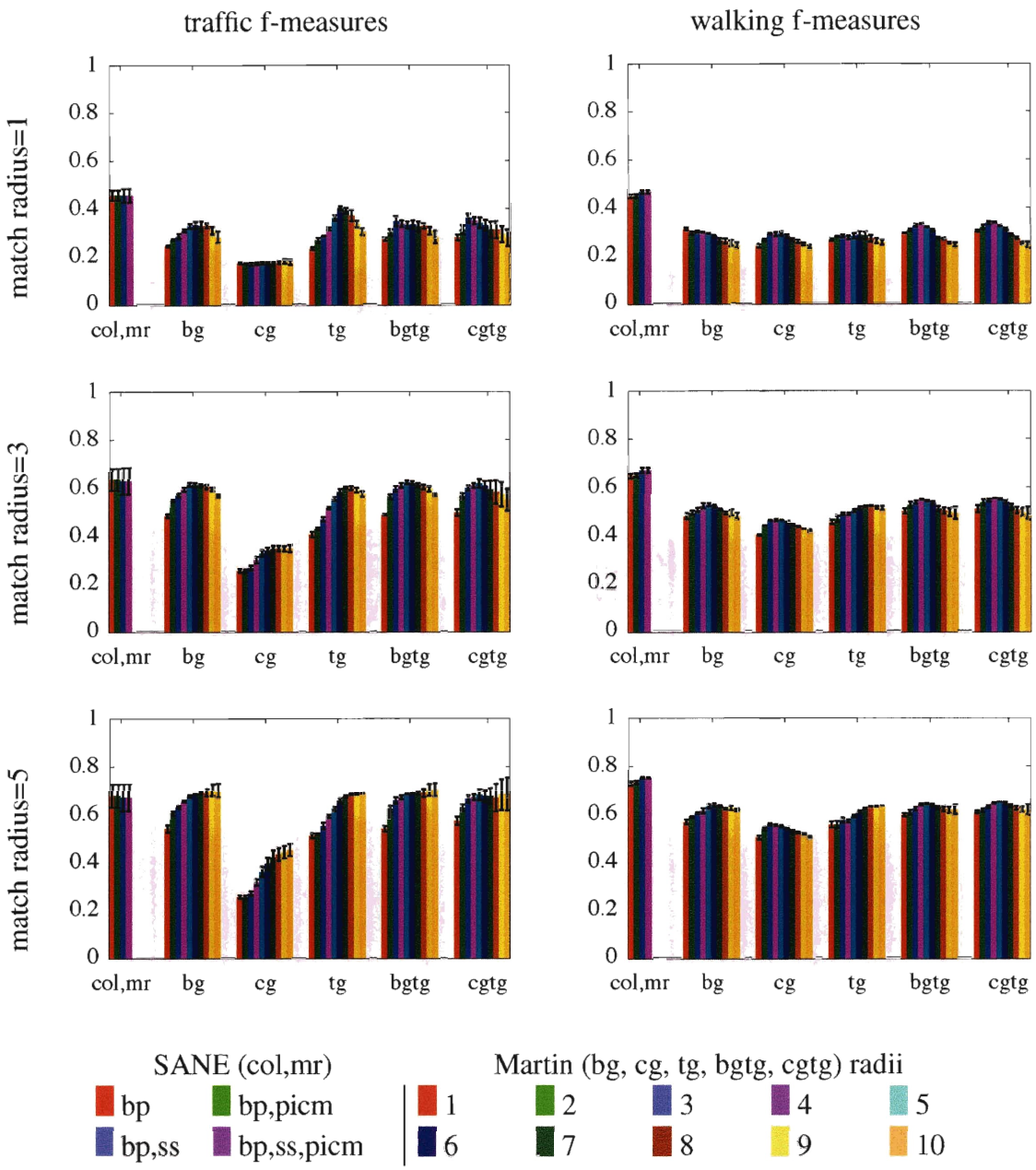



Figure 4-13: The f-measures of color, multiresolution SANE and Martin models on traffic and walking data.

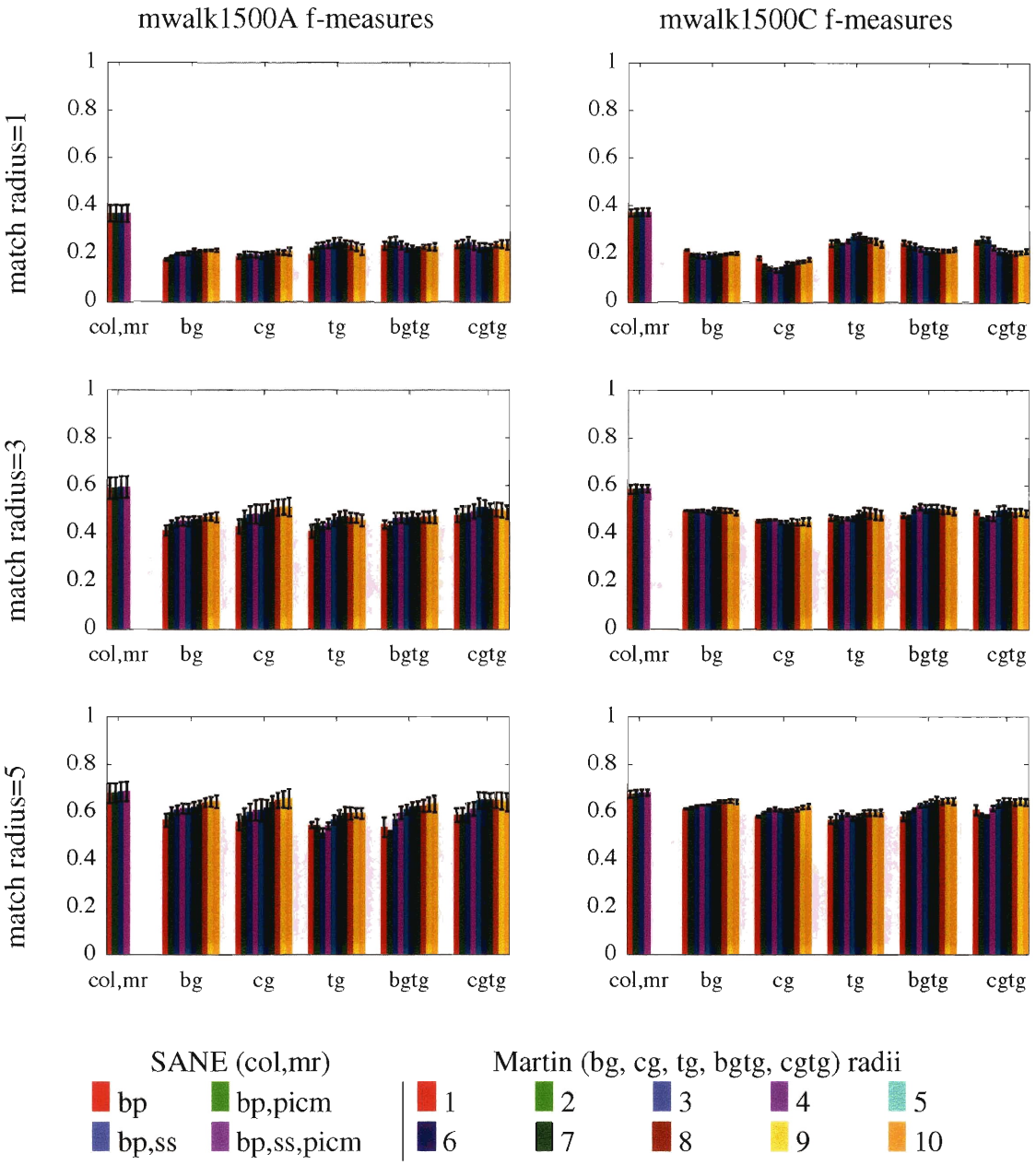



Figure 4-14: The f-measures of color, multiresolution SANE and Martin models on mwalk1500A and mwalk1500C data.

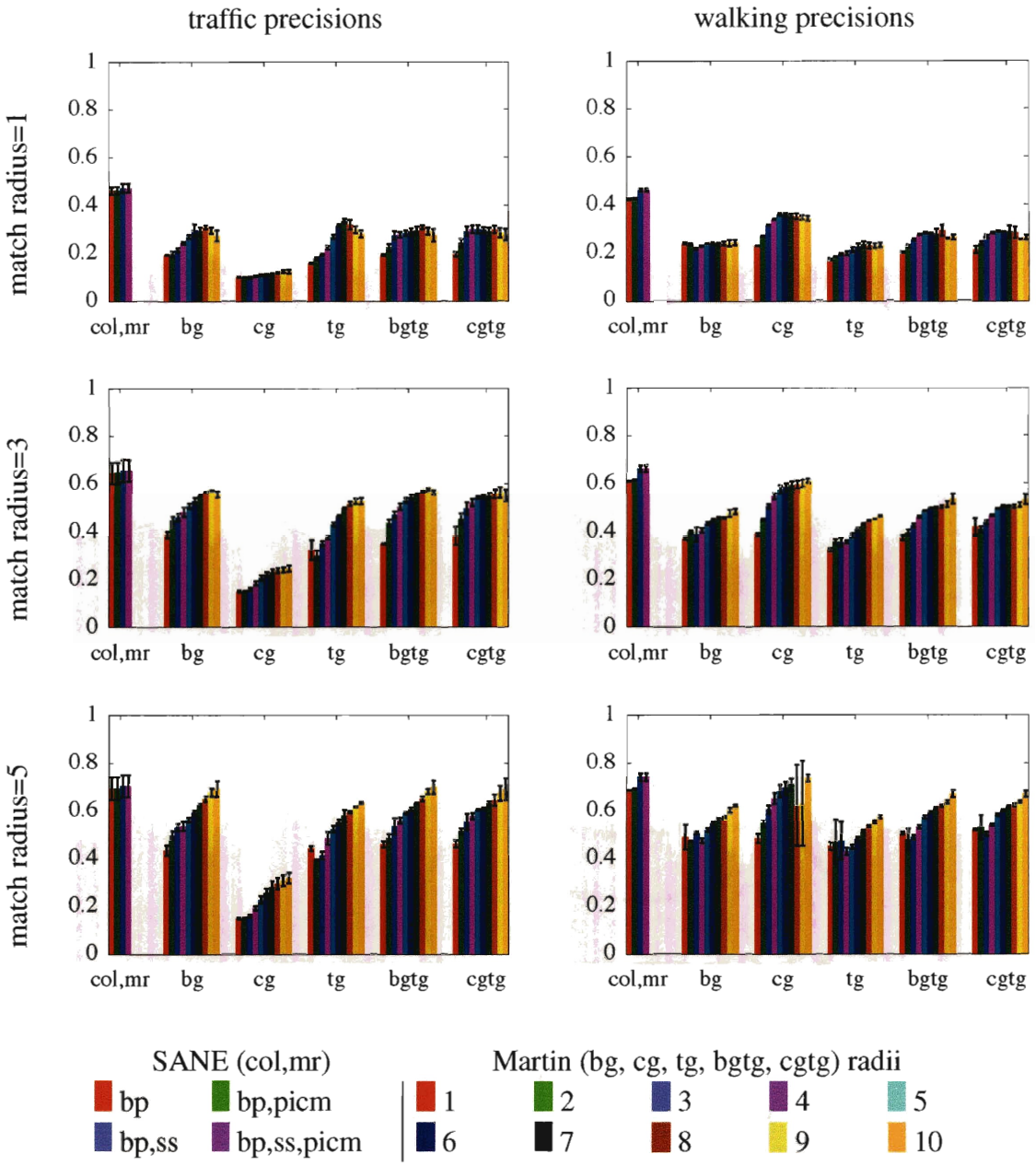



Figure 4-15: The precisions of color, multiresolution SANE and Martin models on traffic and walking data.

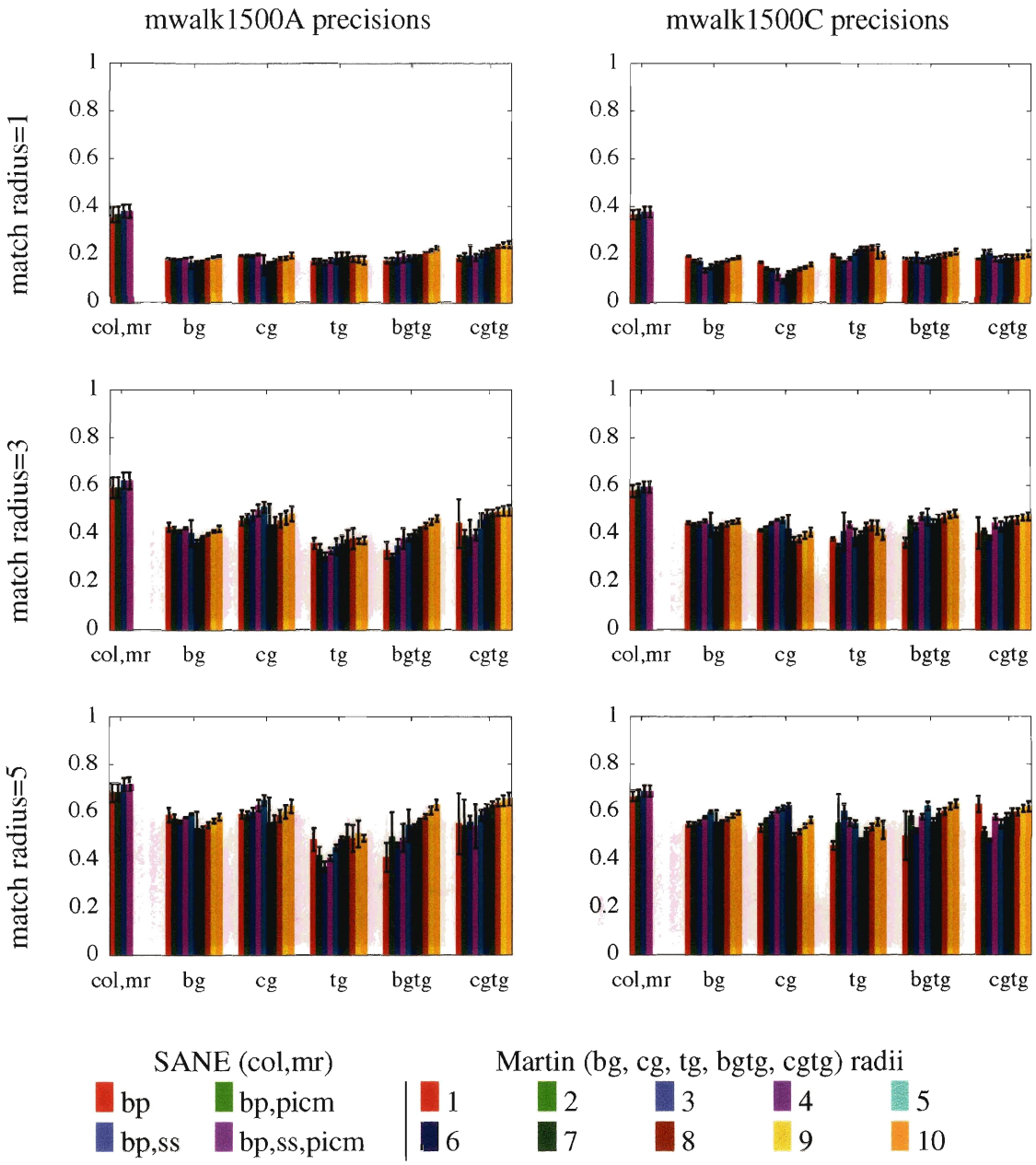



Figure 4-16: The precisions of color, multiresolution SANE and Martin models on mwalk1500A and mwalk1500C data.

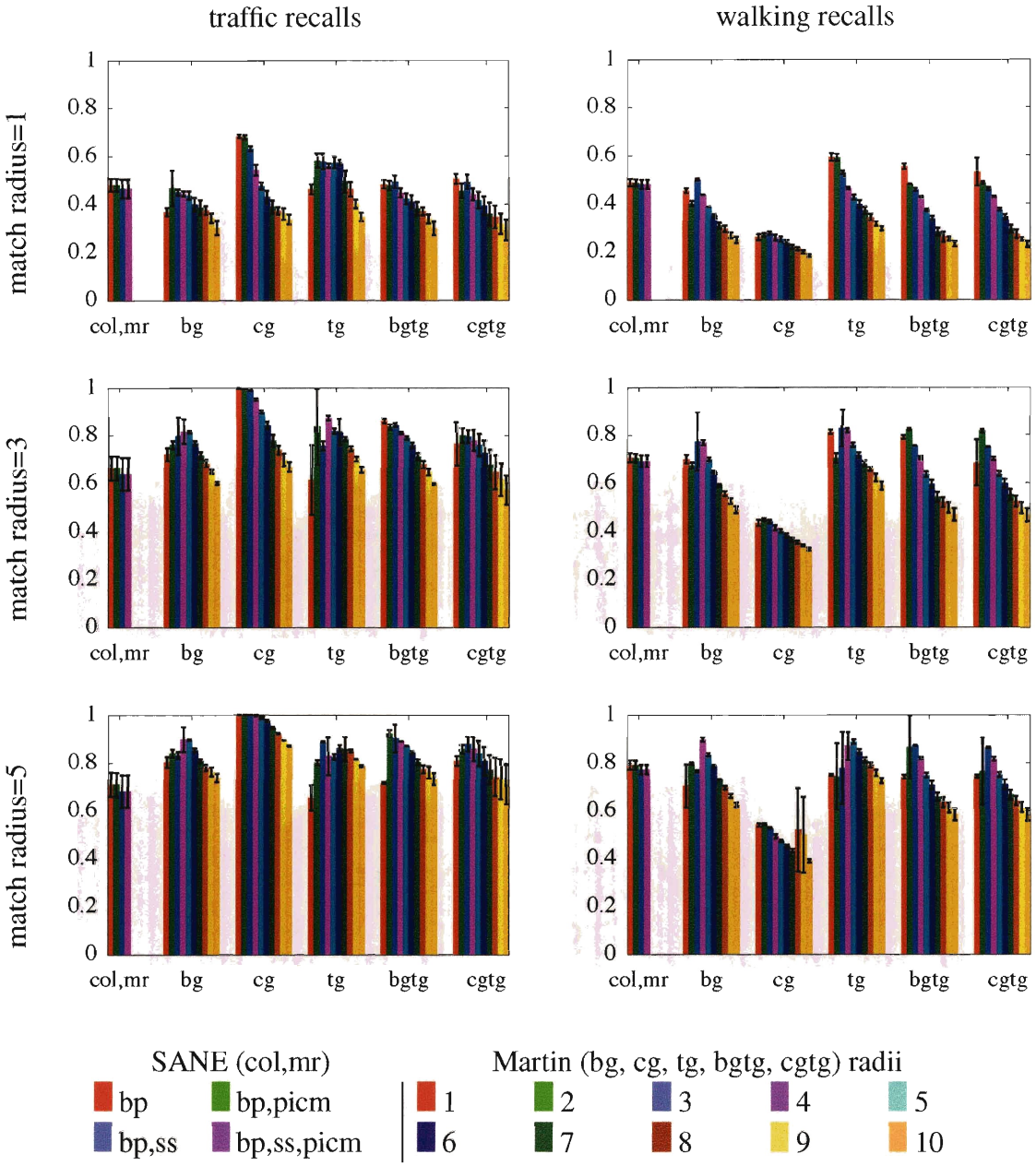



Figure 4-17: The recalls of color, multiresolution SANE and Martin models on traffic and walking data.

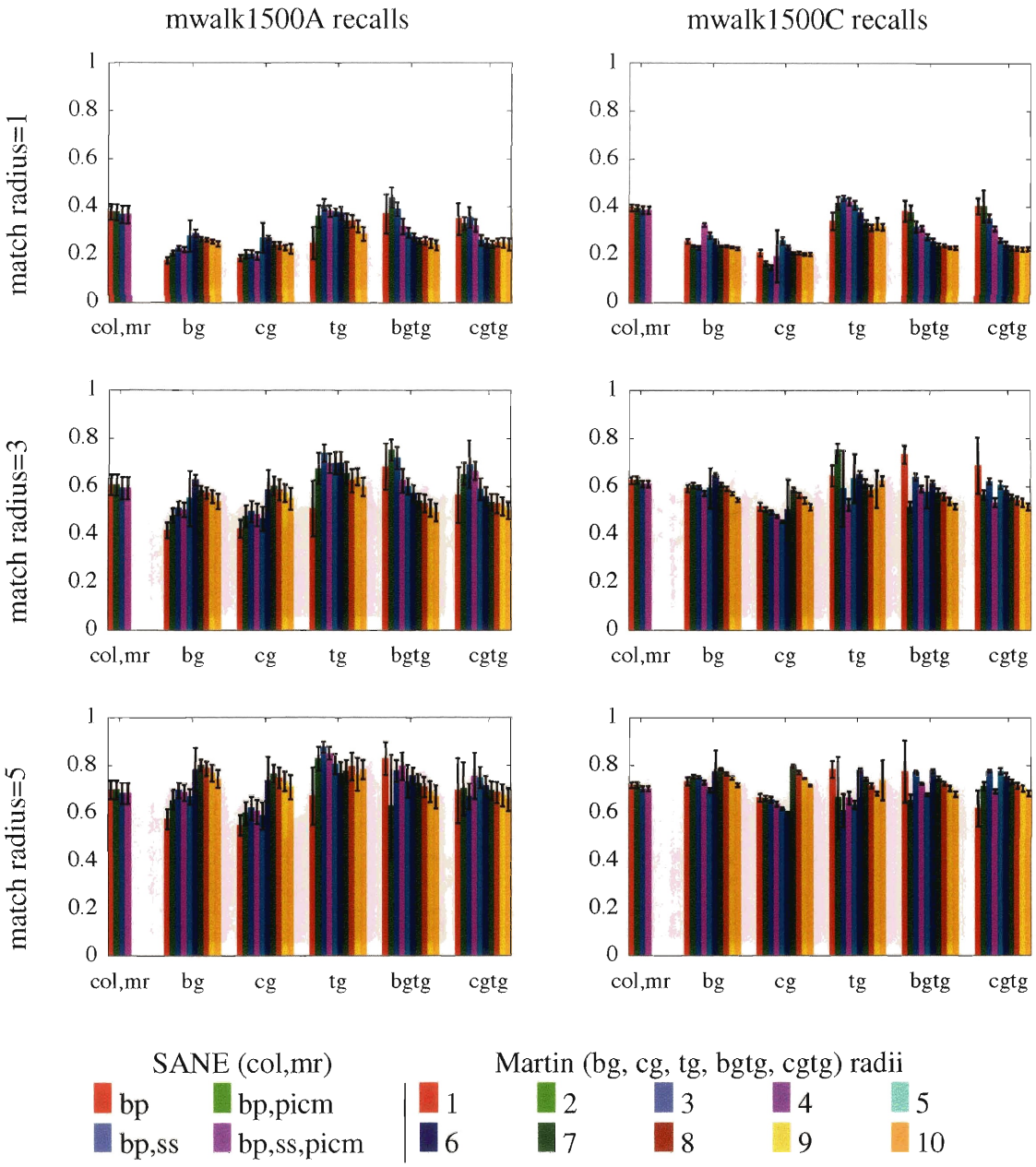



Figure 4-18: The recalls of color, multiresolution SANE and Martin models on mwalk1500A and mwalk1500C data.

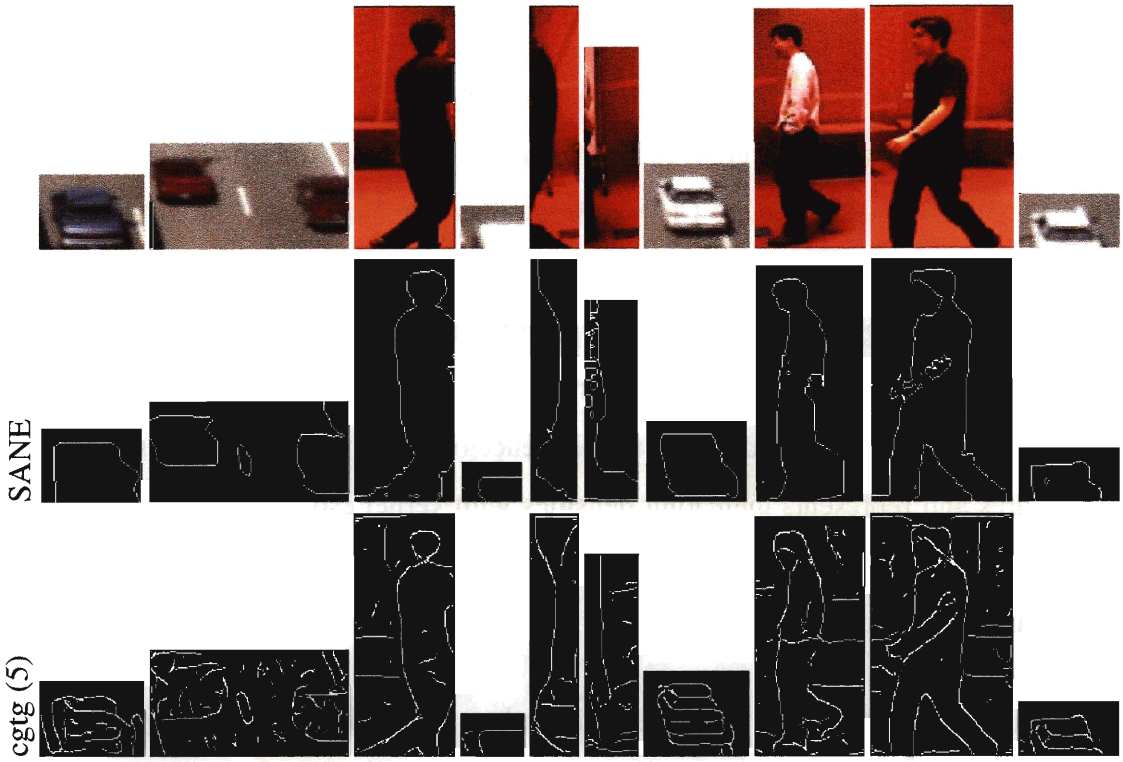



Figure 4-19: The examples in which color, multiresolution SANE most outperforms cgtg Martin with feature radius 5 using match maximum distance 3.

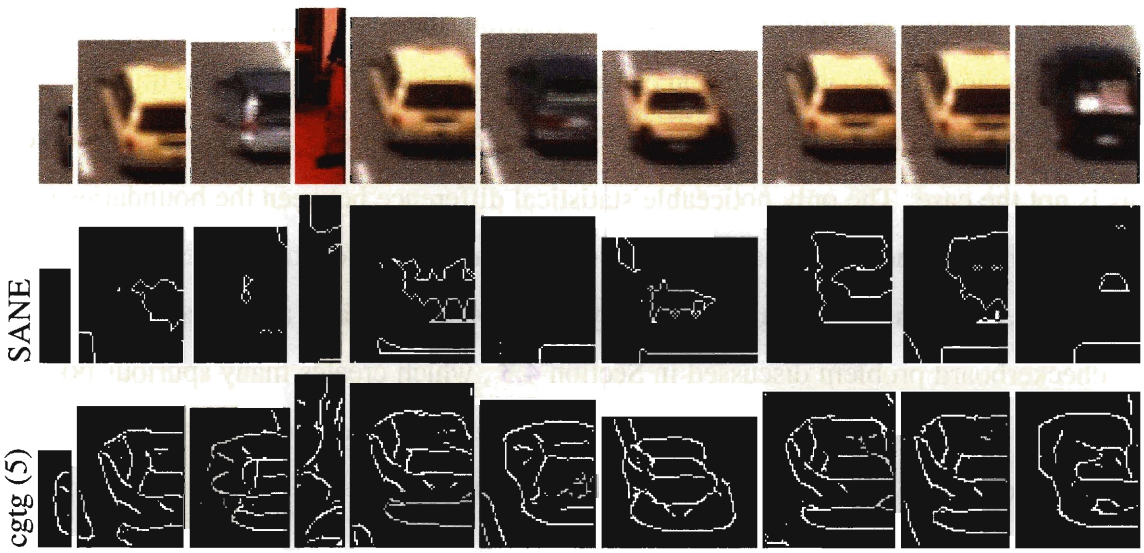



Figure 4-20: The examples in which cgtg Martin with feature radius 5 most outperforms color, multiresolution SANE using match maximum distance 3.

## 4.6 SANE segmentation compared to SANE boundary

To understand the value of different model features, it is useful to remove them and observe the changes in performance. Chapter 3 described the evolution of the SANE model in three steps: boundary-only, boundary and image, and the final segmentation model. Unfortunately, recovering a working version of the first model is impractical. It used simple oriented edge detectors on each patch and represented the relevant probability densities with histograms, assuming conditional independence of detectors. The current model uses pixel features and represents their joint densities with kernel estimates. Using it to construct MRFs in the boundary-only style produced extremely poor results, far worse than the original model's output.

Comparing the second type of model, a boundary model that uses image information in pairwise compatibilities, to the full segmentation model, reveals no appreciable difference in boundary detection performance. Figure 4-21 uses the Martin error metric with a matching radius of 3 to compare the boundary detections produced by both models. Intuition suggests that the segmentation models should produce better boundaries because the parity-matching requirements spreads information (the segmentation label) perpendicularly to the boundaries, which should cause better boundary completion and improve precision by eliminating low-probability internal boundaries from consideration. However, this is not the case. The only noticeable statistical difference between the boundaries produced by the two models is that step-size search is necessary for the segmentation model to achieve equivalent scores on the boundary detection task. This is almost certainly due to the checkerboard problem discussed in Section 4.3 , which creates many spurious boundaries that can be largely eliminated with step-size search. The boundary model, lacking segmentation labels, does not suffer from this problem.

Even if the segmentation labels do not improve the boundary performance, they are still valuable because they ensure the production of closed boundaries and valid segmentations, which the boundary model does not.

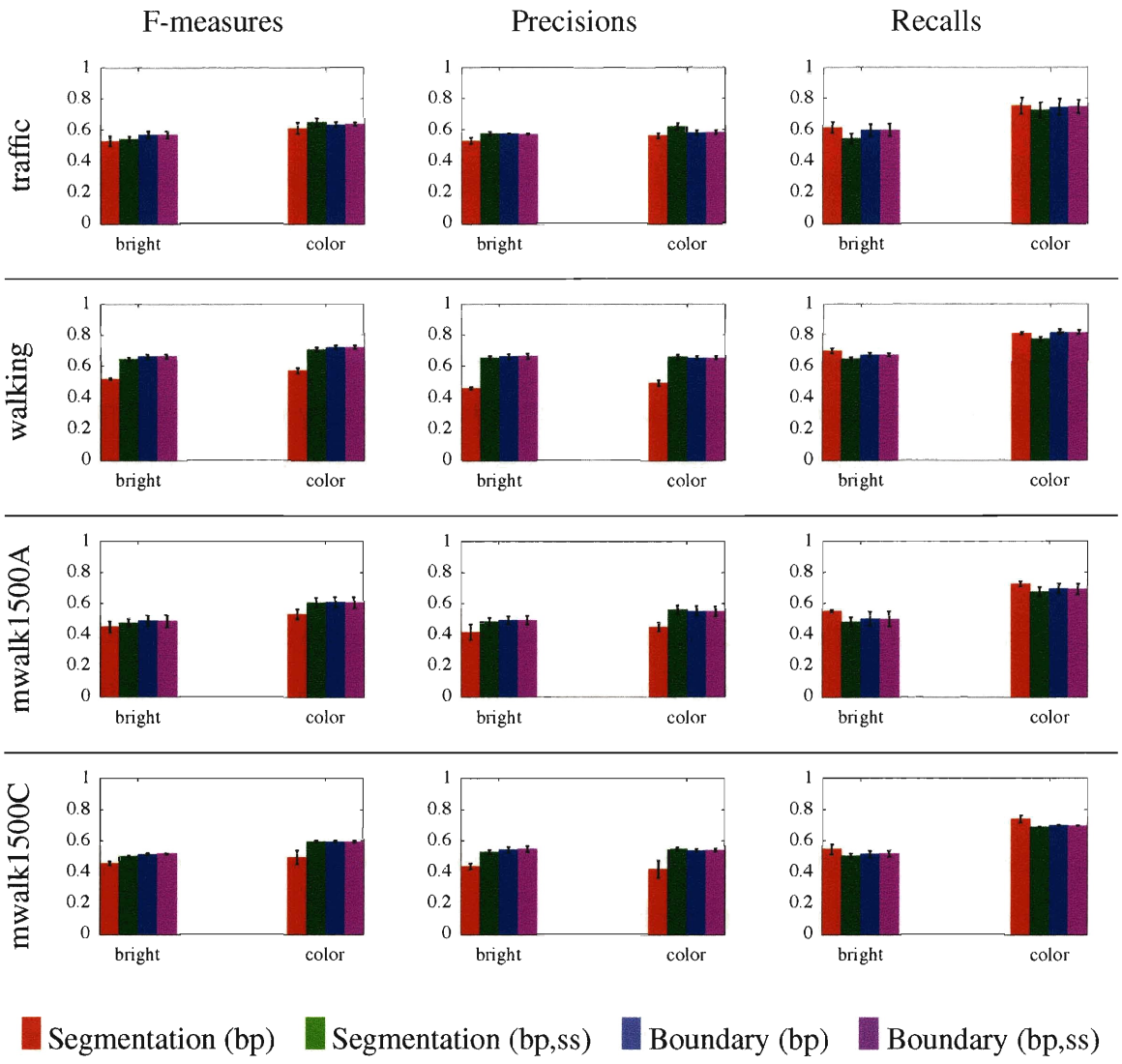



Figure 4-21: The boundary-detection results of running with and without segmentation labels.

## 4.7 Generality and capacity

The previously discussed experiments have all involved testing and training data drawn from the same location and environmental conditions. Trying the models on other environments and conditions explores their generality.

In these cross-over experiments, models trained on the traffic training subsets segmented examples from the walking testing subsets, and vice-versa. Just as with the other experiments there were three training and three testing subsets per data set. The mwalk1500A and mwalk1500C models were crossed in the same manner.

Unsurprisingly, running the walking models on the traffic data and vice-versa produced significantly worse results than those reported in Section 4.3, as seen in Figures 4-22 and 4-23. Given the differences in the training data, it is unsurprising that a loss of precision produced most of the additional error. Crossing the mwalk1500A and mwalk1500C models produced roughly no change in the brightness results and a relatively small decline in color model performance (Figures 4-24 and 4-25). The two sequences came from the same environment and contained many of the same people, but were filmed under different lighting conditions. Given the well-known sensitivity of color values to lighting, it is unsurprising that the color results were affected more by the swap. Again, lower precision accounted for the decline in color model performance. Since the shape of the objects in the sequences were very similar, the edge distributions in the models are probably responsible for a significant part of the lighting robustness.

The capacity of a model to learn to segment multiple environments is another measure of SANE's generalizability. Models trained on traffic and walking subsets were merged together and tested on both testing subsets, as were the mwalk1500A and mwalk1500C models. The mixed models substantially outperformed the cross-over models on the traffic data, but only made a small improvement on the cross-over walking results. The mixed color model improved more on the walking data than the brightness model did. The color information might have helped to differentiate the two data sets from each other and allowed the merged model to switch into a "walking mode." Models trained on both mwalk sequences performed roughly the same as the crossed-over models on each sequence, which

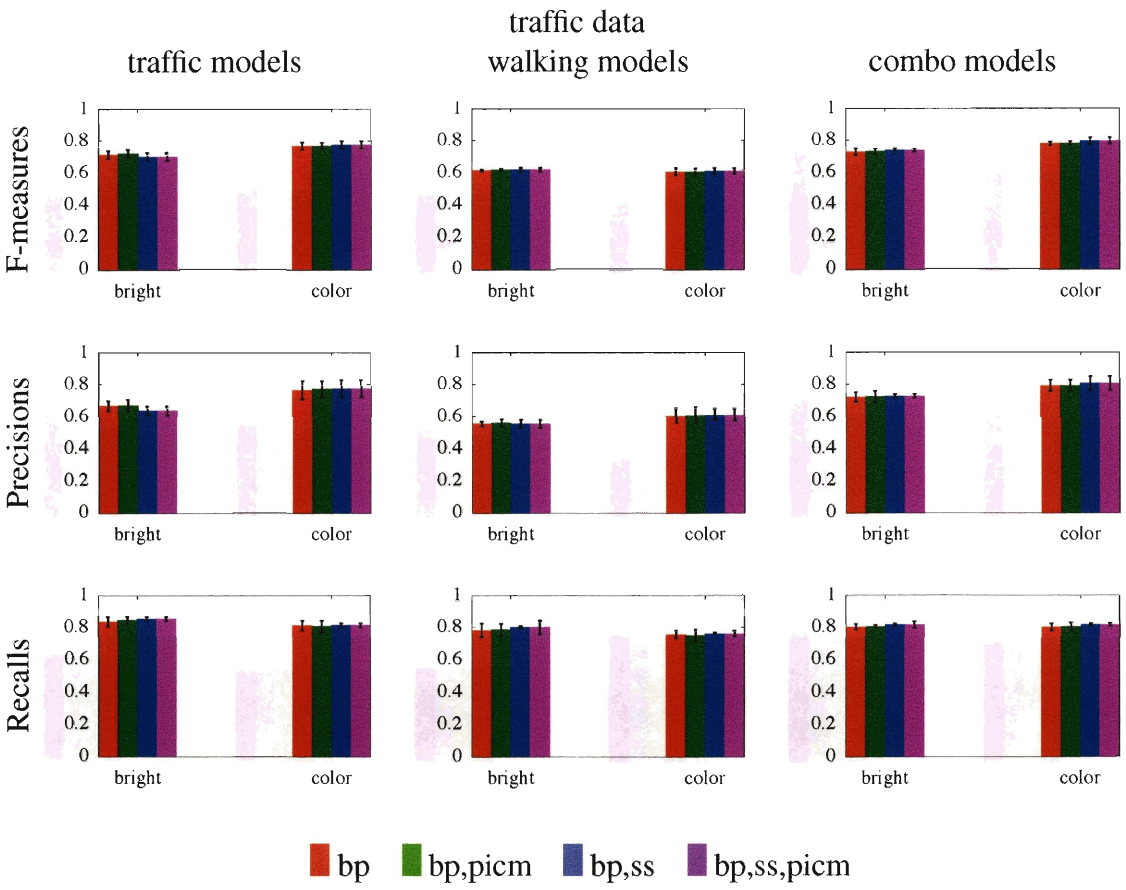



Figure 4-22: The results of running traffic-trained, walking-trained, and traffic and walking-trained models on the traffic data.

is unsurprising since the cross-over models themselves performed so similarly to the native models on each set. The mixed color models performed more poorly than the mixed brightness models or the cross-over color models on the mwalk1500C data. It is not clear why the mwalk1500A color models would outperform color models that contained training examples from mwalk1500C. Perhaps the minimum variance of the Gaussian kernels used to model the image patch conditional probabilities (see Section 3.6) were too large to model samples from such similar sequences with adequate resolution. The result of adding the mwalk1500C samples to the mwalk1500A samples might have been to effectively noise-corrupt the original estimates, resulting in the observed performance decline.

Although more study is needed, the brightness SANE models have shown some robustness to lighting conditions, and little generalizability to radical environmental change. Given SANE's reliance on shape information, it is not surprising that a model trained on

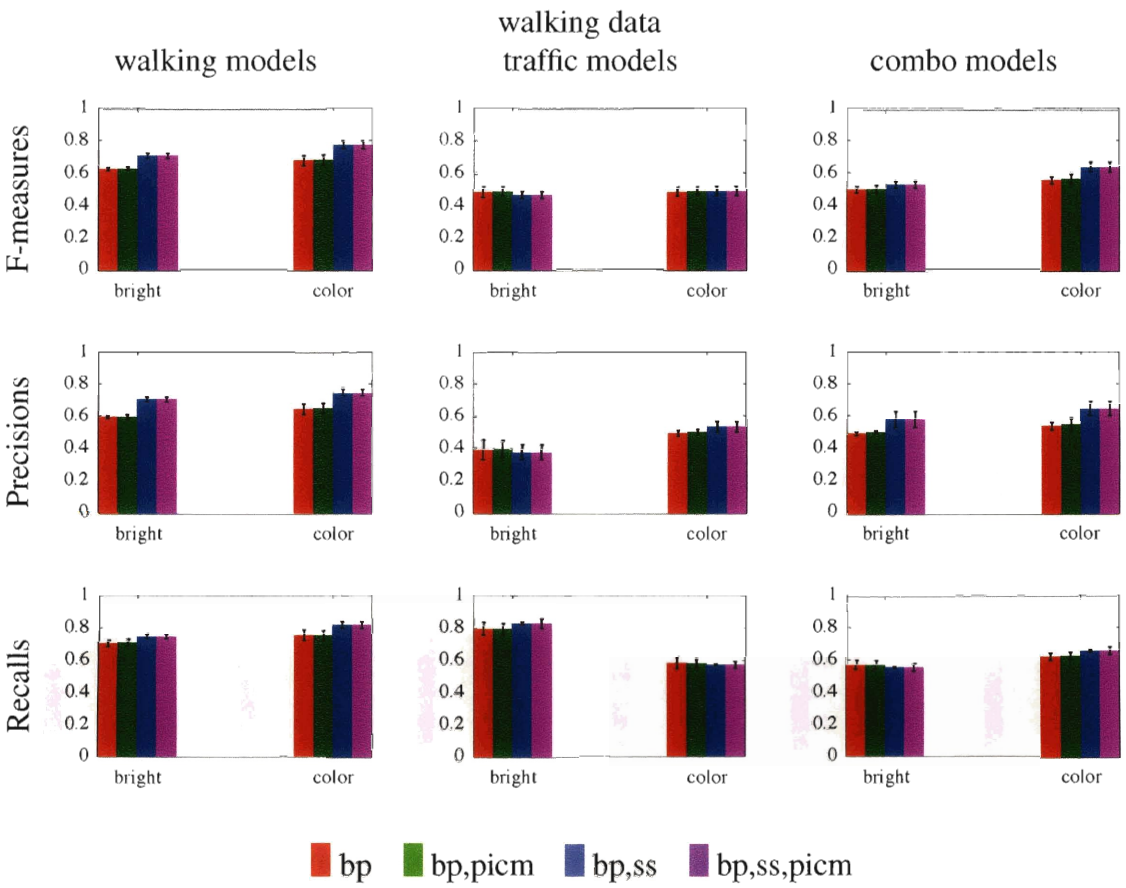



Figure 4-23: The results of running walking-trained, traffic-trained, and traffic and walking-trained models on the walking data.

cars fails to segment people well. The failure of mixed training sets to improve the walking results more could indicate a need for more robust local features or changes to the shape model, two possible areas for future work discussed in Chapter 5.

As discussed in Section 2.1, segmentation differentiates itself from object detection both by the higher resolution of its boundaries and by its generality. The most desirable result of this research would be a version of SANE that, exposed to enough different objects, could learn a model that at least encompassed all of them and hopefully generalized beyond them to unseen object classes. The results have demonstrated this phenomenon to a limited degree. There is some ability for SANE to encompass both the traffic and walking data, but it appears to be somewhat limited. Also, in many of the car results, we found that other "objects," such as the sides of the road or highway lane markers were frequently segmented. However these do not fit the motion definition of object described in Chapter 1

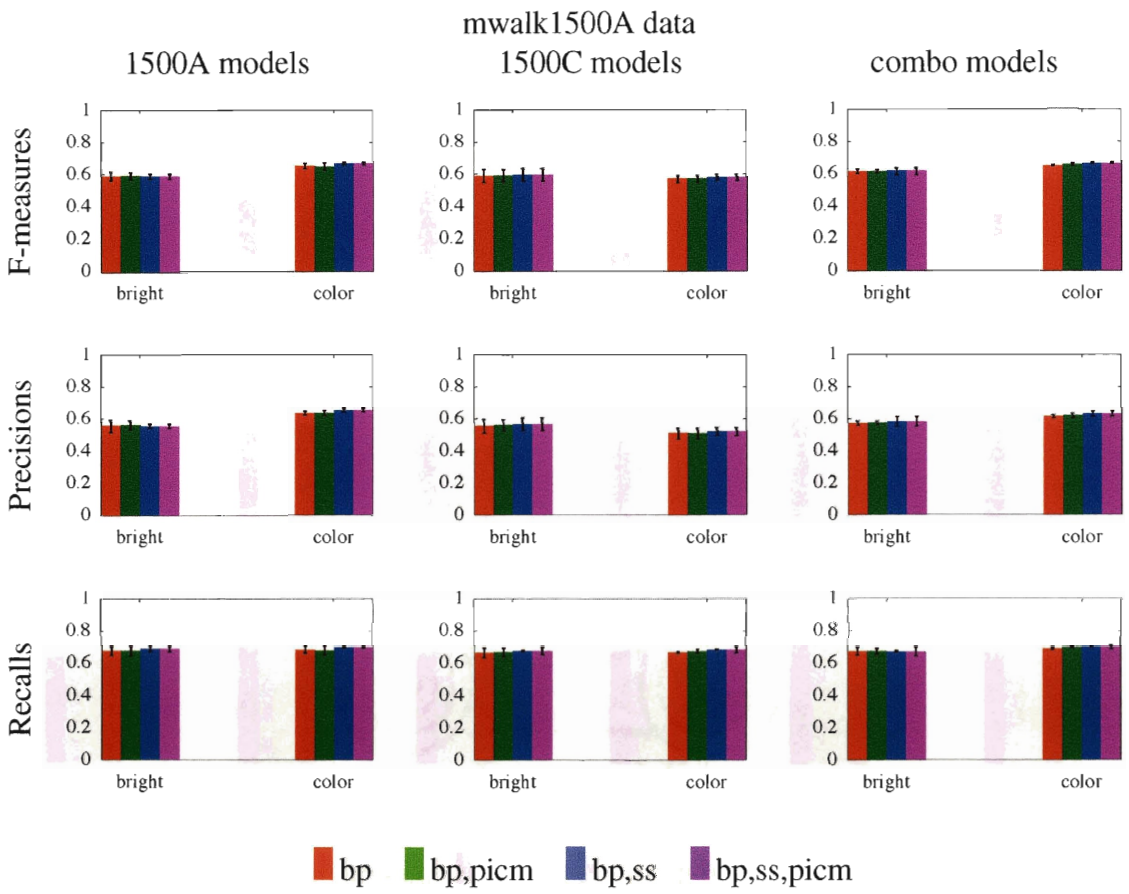



Figure 4-24: The results of running mwalk1500A-trained, mwalk1500C-trained, and mwalk1500A and mwalk1500C-trained models on the mwalk1500A data.

so they can also be considered significant failures. Future work should further distinguish SANE from object detection along the generality axis, and it will undoubtedly require data containing more diverse sets of moving objects.

## 4.8 Computational requirements

Although there are no carefully measured statistics of SANE's resource consumption, it is expensive both in space and time. These experiments employed a multithreaded Java implementation running on dual-processor Apple Xserve G5 computers with 2 or 2.5 gigahertz processors. In inference, SANE typically required over 1 gigabyte of main memory for execution. Training required on the order of 6 to 12 hours per single-resolution model and resulted in model files ranging roughly between 400 and 700 megabytes. Training time

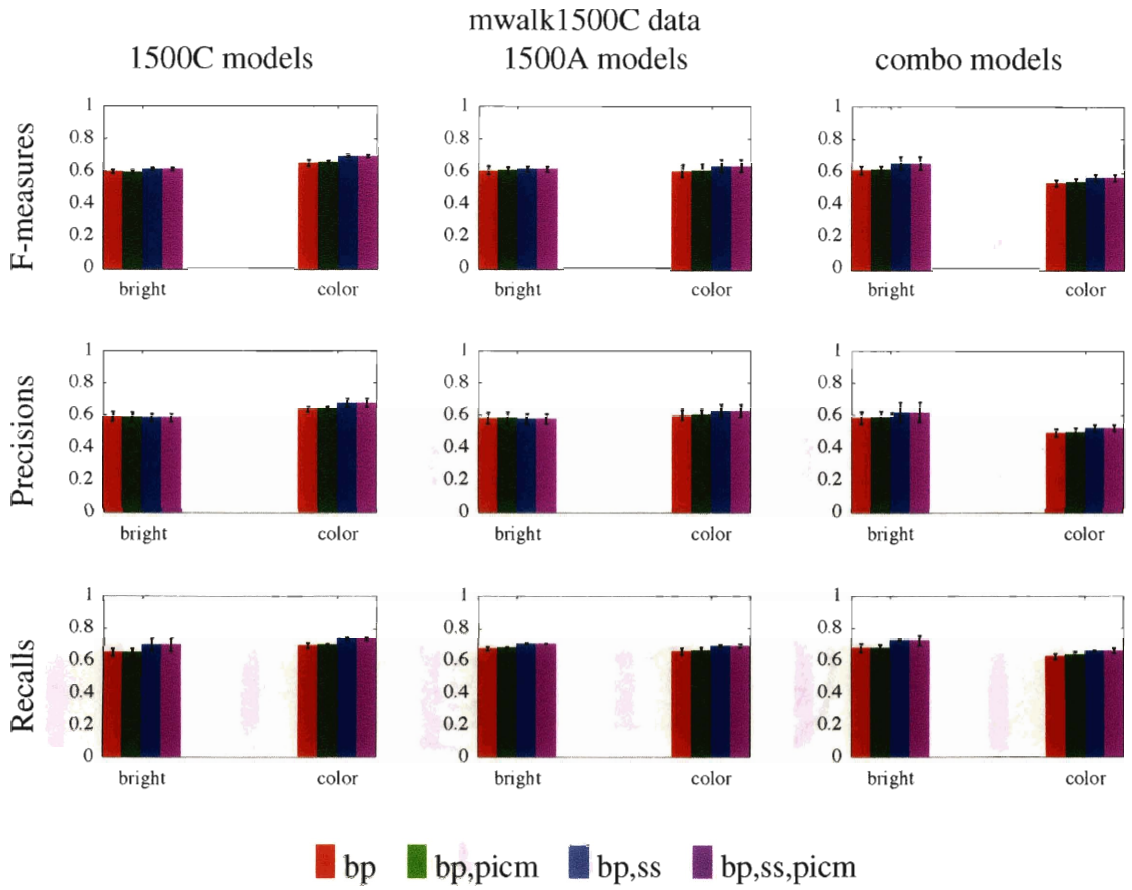



Figure 4-25: The results of running mwalk1500C-trained, mwalk1500A-trained, and mwalk1500A and mwalk1500C-trained models on the mwalk1500C data.

is dominated by fitting kernel density estimates. The total time required to perform both step=1 and step-search segmentation ranged for a few minutes for images containing a few thousand pixels to a few hours for images containing tens of thousands of pixels.

The segmentation running time is dominated by the construction of the MRF, particularly in selecting the candidate scenes and setting the compatibilities. The latter step is expensive partially due to the marginalization discussed in Section 3.6. Constructing kernel density estimates of the single-node marginals that are guaranteed to match that calculation would eliminate the need to constantly recompute these values and greatly speed MRF construction. It might be possible to create such estimates by drawing samples from the distributions involved in the marginalization. Additionally, finding sets of possible edges at each node such that neighboring assignments can be completed requires a great deal of searching. It seems likely that equivalent results might be obtained by only ensuring that

the most probable neighboring assignments can be completed; this could lead to equivalent output, but require much less computation.

It is unlikely that translating the code to C or C++ would make a significant difference in running time. Although many perceive Java code to be slow, this impression often arises from the high startup costs of the Java Virtual Machine (JVM). In these experiments, the SANE code ran for hours and days in a single JVM instance, so the startup penalties were insignificant and the code fully benefitted from just-in-time compilation, dynamic profiling, and other JVM benefits. However, the code relies heavily on object-oriented programming features and reimplementing it with more primitive data structures, using any reasonable language, might make it faster.

It is certainly possible to construct a more memory-efficient SANE. The current implementation relies on aggressive caching to improve speed, so much of the memory can be reclaimed at the cost of extra computation.

## 4.9 Further questions

The experiments have demonstrated that SANE can be successfully trained to segment a class of objects in a specific environment, and that it has a limited ability to generalize to multiple environments, which can probably be greatly enhanced by future extensions. The experiments do not address the question of what class of segmentation algorithms can be learned from motion data. The current SANE procedure requires a fixed camera and learns about objects that move in the observed world, but it seems unlikely that the set of moving objects is an adequate sampling of the set of all objects. According to the motion-based object definition, first discussed in Chapter 1, even things that seldom move, like rocks or desks or mountains, can be considered objects because they would move coherently if force were applied to them. However, it seems obvious that seldom-moving objects, like rocks or lichens, may have many visual properties that distinguish them from frequent movers, such as cats. They might lack legs or wheels. They might have complex branching shapes or highly irregular boundaries. Their surface textures might be different reflecting their mineral or vegetable nature. All of these factors make it likely that their visual properties

will be underrepresented in any database of moving objects.

Truly discovering how much segmentation can be automatically learned from the environment will probably require the ability to extract motion-defined objects by means of depth perception and manipulation. Disparity information, captured by two cameras simultaneously (stereo) or by a single camera across time (structure from motion), can illuminate the boundaries of many seldom-moving objects, and the ability to apply forces via manipulation can test boundary hypotheses when other information is unavailable. The use of manipulation to provide segmentation knowledge is similar to recent work by Fitzpatrick [10] (see Section 2.1), but with the goal of learning better segmentation algorithms rather than focusing on detecting specific objects and their boundaries.

Manipulation and depth perception also provide the opportunity to overcome the problem of shadows. The background subtraction algorithm detected moving shadows and labeled them as part of a moving object. In the traffic images, for example, most cars have a shadow and SANE frequently detects them without penalty because they are considered to be "foreground." This does not correspond well with intuition, but it is difficult to avoid without adding special shadow-suppression machinery to the background subtracter. The ability to judge depth and to verify segmentation hypotheses by manipulation would be invaluable in solving this problem because they can both provide data distinguishing shadow from object without requiring shadow-specific engineering.

The experiments did not address the issue of scale-dependence. Apart from natural variations in the scale of car and people examples, the scales of the training and testing data were constant and identical. Because the shape model is local and has no overall object scale information, it seems likely that SANE would be robust to scale until the scale change significantly affected the local patch statistics. For example, a large change in scale could alter the prior probability of encountering boundary inflections or the distributions of inflection angles and could have a similar impact on pairwise edge assignment statistics. It is likely that making SANE robust to scale would require multi-scale training, just as it was made rotation invariant by training on multiple rotations of the data.

Another unexplored topic is SANE's ability to adapt to situations with similar objects to the training data, but substantially different backgrounds. The brightness models, which

demonstrated the most lighting invariance, would probably perform best in such a situation, but for strong performance it's likely that diverse training backgrounds would be necessary.

Finally, the dreaded checkerboard patterns indicate that some of SANE's errors might be the result of inference errors and not the fault of the segmentation model. The step-size search method's relative success on the walking data support this conclusion since it discovers solutions that have higher MRF scores than those found by standard belief propagation. The next step in this exploration would be to try exact inference algorithms, such as Gibbs sampling [14], on some examples to determine the magnitude of the error attributable to inference approximation. Unfortunately, exact inference on a SANE MRF would be extremely computationally expensive, so picking worthwhile examples might be crucial to the experiment. Another possible source of approximation error is the compatibility functions, which could also be better estimated with exact inference, as discussed in Section 2.4.

# Chapter 5

# Future work and conclusions

## 5.1 Improved local features

SANE's biggest weakness is its lack of sophisticated local edge detectors. The Martin detectors demonstrate that a great deal can be achieved by using optimal local detectors, even though they consider each point as an independent detection problem. In contrast, SANE's detectors perform extremely poorly when employed separately, frequently detecting no edges at all. This may be because they are attempting to determine a 5x5 edge segment rather than the boundary status of a particular pixel, which is more difficult to do without non-local information, but it's also likely related to the use of pixel-value features rather than employing image derivatives, textons, or other traditional boundary-detection features.

Multiple experiments attempted to mate better image features to the SANE model, all without success. The output of the Martin edge detectors, derivative of Gaussian filters (as in the Canny detector [6]), and features derived from principal components analysis (for a description see Forsyth and Ponce [11]) all failed to produce usable results. It seems likely that the use of kernel density estimates impeded the success of these alternate features. The Gaussian kernels used for the pixel features were spherical for simplicity. If the samples' variance is strongly aligned along a particular axis, elliptical kernels that can match that orientation might fit them better. Results using the output of the Martin detectors appeared to improve with the use of elliptical kernels.

Konishi et al. [23] developed an adaptive approach for representing local image-feature probabilities for edge detection. Probabilities are represented with either adaptive histograms or decision trees; bin boundaries and decision cuts are selected to maximize the ability to distinguish between edge and non-edge image locations. The resulting representations are relatively compact and avoid over-fitting. Compared to the kernel density estimator, the Konishi approach might be less sensitive to the particular image features used, and it would focus modeling effort on discriminating between different edges. Because the adaptive estimators were originally designed to perform a binary classification, while the SANE model has to choose between nearly 3000 edge classes, it might require substantial work to generalize the approach.

The Martin detectors employ logistic regression, a much simpler and more restrictive estimator than kernel density estimates, and found it equal or superior to more complex estimation methods, such as support vector machines [32]. An early version of SANE used the differences between pixel brightnesses as image features, and modeled the probabilities of their absolute value with Gaussian distributions. It never performed very well, but it was a much smaller model with lower computational requirements. It seems likely that with the right image features, the overhead of the kernel density estimates would be superfluous and much simpler models would suffice.

## 5.2 Continuous edge models

SANE's edge model is discretized into 2717 possible assignments. This simplifies some of the estimation problems, but makes the model extremely sensitive to the training procedure that extracts parameterized edges from raw binary image patches. Consider two similar edges: (Entry:(3,0) Exit:(4,1)) and (Entry:(3,0) Inflection:(4,1) Exit:(4,2)). The procedure for matching edges only chooses to use inflection points to fit data if the result fits substantially better than the best uninflected boundary, so the second possibility might be chosen much less frequently than the first and be assigned a much lower prior probability. This also results in fewer supporting samples for computing image likelihoods related to the second edge. However, it could be argued that the two edges are very similar and should

have very similar local probabilities.

If the edge model was continuous, there would be six continuous variables representing the entry, exit, and inflection coordinates. Estimates of related probabilities and likelihoods could use functional forms that would naturally smooth over the parameters. The result would be that the particular details of the edge-fitting procedure would have less influence and training data would be used more effectively.

The main difficulty is specifying a reasonable probability model for continuous edges. If the edges are represented by Gaussian compatibilities, the multiplicative updates of belief propagation pose no problem because the product of Gaussian functions is also Gaussian. However, naively multiplying non-Gaussian compatibilities can cause the functional form of the local beliefs to change with every update. Fortunately, a new method of nonparametric belief propagation developed by Sudderth et al. [43] solves this problem by constructing tractable approximations of the messages after each update.

## 5.3 T-junctions and multi-label segmentations

The background subtraction method described in Section 3.6 can separate objects from a static background, but is not capable of separating two overlapping moving objects from each other. Doing this would require extra information, such as optical flow vectors or tracking data, that disambiguate the overlapping objects and the introduction of more pixel labels. The introduction of more sophisticated motion-segmentation techniques would be a highly useful extension that would allow training in more cluttered environments.

The issue of overlapping objects can still be important in environments where moving objects never overlap one another during training. Although the partially-labeled training data only contains well-separated objects, a typical image might contain overlapping objects, some of them static. The potential presence of static objects is the reason for limiting training to the bounding box around moving objects, but this is not possible during inference in a deployed system. Success might depend on the ability to model multiple object segments that touch one another, despite the fact that the training data contained no such instances.

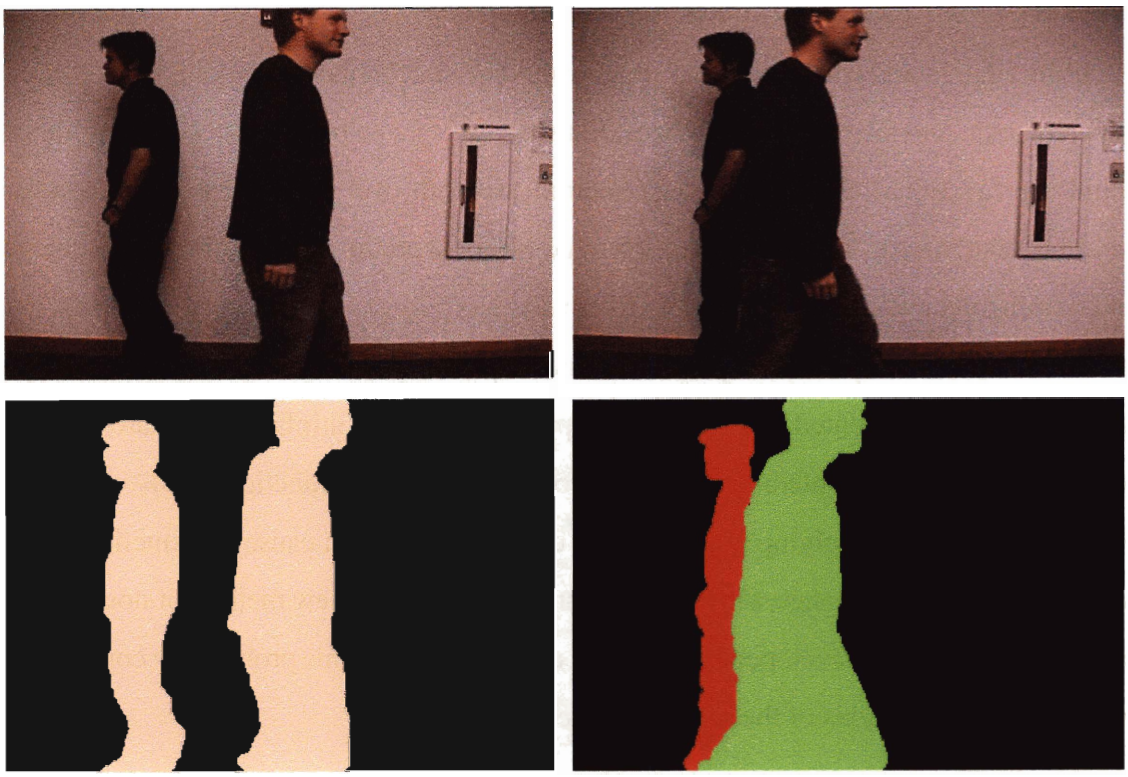


Figure 5-1: Segmenting overlapping or touching objects from each other and the background requires more than 2 segmentation labels.

In order to address multiple touching object segments that need to be distinguished from each other and from the common background, two innovations are required. First, the current model only infers binary pixel labels, each image location is assigned a "0" or "1." This is sufficient if objects never touch one another, but more labels are required in other circumstances, as seen in Figure 5-1. According to the four-color theorem, an image can be divided into an arbitrary number of regions, whatever their configuration, with only four pixel labels [53]. So, the first expansion to the model is to move to four segment labels, "0" through "3."

But adding more labels is insufficient without the ability to form more complex region intersections. The current model only allows a single edge to pass through an image region. To be generally useful, the extra labels must be combined with the ability to represent the intersection of three or more regions in a single patch. This is difficult because the background subtraction data only contains two or fewer regions in any patch. How can the

conjunction of more regions be added if they are totally unsupported by the training data?

Assuming that the touching object segments are independent of one another, a conjunction of three or more regions can be modeled by the presence of two independent edge assignments at the same image location. Now the segmentation assignment at a node is $S_i = (E_{i0}, E_{i1}, P_i)$, where $E_{i0}$ and $E_{i1}$ are the two edge variables and $P_i$ is an expanded parity value. Each $E$ separates a particular object from its surroundings. Each edge defines two regions, so two overlapping edges can define as many as four independent regions. Each region needs one of the four segment labels, so the new $P_i$ variables will range over 24 (4!) values rather than 2.

Implementing this algorithm in an efficient manner requires some restrictions on the values of $E_{i0}$ and $E_{i1}$. Allowing any combination increases the (unpruned) state space at each node from $2717 \cdot 2 = 5434$ to $2717^2 \cdot 24 = 177,170,136$. Additionally, there are a number of important special cases that govern the relationship between a node's edge assignments. The two edges must not be identical, or else finding the MAP assignment would necessarily choose identical edge pairs at every node. It's also important to handle the representation of a solitary edge. The most obvious way to represent a single edge is to assign one of the two edges to the empty edge, but in a boundary region the empty edge is extremely improbable, which will bias the model towards making undesirable multi-edge conjunctions. One solution is to allow one of the two edge assignments at each location to be empty edge without penalty.

This extension has been implemented and undergone extremely preliminary testing on artificial data. Further testing and refinement to make its use practical on real images may be a focus of future work.

## 5.4 Beyond Markov random fields

The largest drawback of employing Markov random fields is the inability to capture truly global shape information. The variables in the SANE model are all tied to explicit image locations, so modeling a property such as "all objects are ellipses with a minor to major axis ratio of 2:1" is impractical. A node verifying that property would need to be connected

to all image locations and would be have to check this property for all possible segments. It is difficult to model global shape properties with collections of locally-connected parts, but most graphical model formalisms naturally lead to that style of representation. The other extreme is represented by techniques such as principal components analysis which capture the globally significant features and variations in data. A technique spanning this spectrum would be invaluable for future segmentation work.

## 5.5 Conclusions

The primary contributions of the Segmentation According to Natural Examples algorithm described in this thesis include: the formulation of the problem of learning a static image segmentation model from motion training data, a new Markov random field segmentation model, and learning algorithms for training the model to solve the object segmentation problem.

The problem of learning image segmentation from motion-segmented training data is significant because it provides an objective method of generating training and testing data sets and a simple method for evaluating success and failure. It also fits into a larger framework of redefining image segmentation in terms of extracting objects from images, where the objects are defined to be useful to other systems, not only as visually significant regions.

The MRF segmentation model uses variables that correspond to relatively large image patches and the boundaries that pass through them in order to capture shape, regional, and boundary information well. It can be viewed as a descendent of the original Geman and Geman [14] line and pixel-process model, which operated on the pixel scale, but using its more complex variables to capture more shape and image information. The success of the SANE model demonstrates the practicality of these larger, more sophisticated models given modern computer power and modern approximate inference and learning algorithms.

The comparison of SANE's performance to the Normalized Cuts algorithm showed that object segmentation does not necessarily correspond to a traditional image segmentation problem because a generic model of visual significance does not apply equally well to all objects. The comparison to the Martin detectors demonstrated the value of a strong shape

model, showing that it can outperform sophisticated local detectors in a real-world case with low resolution and high noise, and when shape is an important cue.

Finally, the SANE model demonstrated limited ability to generalize to new lighting conditions and a limited capacity to be trained to cover multiple environments. This is a clearly important area for future work, probably focused on improved local features and the incorporation of more and better shape information.

The most direct applications for SANE would be to use it as a grouping step in problems such as optical flow, as discussed in Chapter 1, or as a pre-processing step before applying object detection algorithms. In fact, a reliable object segmentation can eliminate the need for object detection, which typically requires searching across image locations, and allow the substitution of pure object recognition approaches in which the object has been localized, but not yet identified. Furthermore, by separating object and background pixels, SANE could reduce the modeling burden on recognition algorithms, eliminating the need for them to account for background variance.

Ultimately, the widest use for segmentation lies in the domain of robotics. Navigation and manipulation can benefit from accurate division of the image into objects, even if the objects are unclassified. Furthermore, ego-motion and manipulation can provide new sources of data about motion objects, as described in Section 4.9.

Apart from the specifics of the algorithm, this thesis demonstrates the interesting and important problems in the area of finding task-optimal segmentations. Although a great deal of outstanding work has focused on purely image-based segmentation performance, the move towards finding segmentations that infer useful properties, such as potential motion boundaries, will hopefully lead to a larger and more robust role for image segmentation in real-world systems.

# Bibliography


[1] J. Besag. Spatial interaction and the statistical analysis of lattice systems. *Journal of the Royal Statistical Society, Series B(Methodological)*, 36(2), 1974.

[2] J. Besag. On the statistical analysis of dirty pictures. *Journal of the Royal Statistical Society, Series B(Methodoloical)*, 48(3), 1986.

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

[4] E. Borenstein and S. Ullman. Learning to segment. In *European Conference on Computer Vision*, 2004.

[5] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. In *International Conference on Computer Vision*, 1999.

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

[7] M.C. Cooper. The tractability of segmentation and scene analysis. *International Journal of Computer Vision*, 30(1), 1998.

[8] T. Cour, S. Yu, and J. Shi. Matlab normalized cuts segmentation code. http://www.cis.upenn.edu/~jshi/software/, 2004.

[9] P. Felzenszwalb and D. Huttenlocher. Efficiently computing a good segmentation. Technical report, Cornell University, 1999.

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

[11] D.A. Forsyth and J. Ponce. *Computer Vision: A Modern Approach.* Prentice Hall, 2003.

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

[13] A. Gelman, J.B. Carlin, H.S. Stern, and D.B. Rubin. *Bayesian Data Analysis.* Chapman & Hall/CRC, 2003.

[14] 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.

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

[16] T. Heskes. On the uniqueness of loopy belief propagation fixed points. *Neural Computation*, 16, 2004.

[17] B.K.P. Horn. *Robot Vision.* The MIT Press, 1986.

[18] B.K.P. Horn and B.G. Schunck. Determining optical flow. *Artificial Intelligence*, August 1981.

[19] M.I. Jordan. *An Introduction to Probabilistic Graphical Models.* To be determined, 2006.

[20] J. Joyce. *A Portrait of the Artist As a Young Man.* Penguin Classics, 2003.

[21] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. In *International Conference on Computer Vision*, 1987.

[22] V. Kolmogorov and R. Zabih. What energy functions can be minimized via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(2), 2004.

[23] 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.

[24] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *International Conference on Machine Learning*, 2001.

[25] Y.G. LeClerc. Constructing simple stable descriptions for image Partitioning. *International Journal of Computer Vision*, 3:73–102, 1989.

[26] M.E. Leventon, O. Faugeras, W.E.L. Grimson, and W.M. Wells. Level set based segmentation with intensity and curvature priors. In *Mathematical Methods in Biomedical Image Analysis*, 2000.

[27] S.Z. Li. *Markov Random Field Modeling in Image Analysis*. Springer-Verlag, 2001.

[28] D.G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2), 2004.

[29] R. Malladi, J.A. Sethian, and B.C. Vemuri. Shape modeling with front propagation: A level set approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(2), 1995.

[30] D. Marr. *Vision*. W.H. Freeman and Company, 1982.

[31] D. Martin and C. Fowlkes. Matlab boundary detection code. http://www.cs.berkeley.edu/projects/vision/grouping/segbench/, 2004.

[32] 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*, 26(5), 2004.

[33] 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.

[34] J. Migdal and W.E.L. Grimson. Background subtraction using markov thresholds. In *IEEE Workshop on Motion and Video Computing*, 2005.

[35] K.P. Murphy, Y. Weiss, and M.I. Jordan. Loopy belief propagation for approximate inference: An empirical study. In *Uncertainty in Artificial Intelligence*, 1999.

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

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

[38] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2003.

[39] 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.

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

[41] 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.

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

[43] E.B. Sudderth, A.T. Ihler, W.T. Freeman, and A.S. Willsky. Nonparametric belief propagation. In *Computer Vision and Pattern Recognition*, 2003.

[44] M.F. Tappen and W.T. Freeman. Comparison of graph cuts with belief propagation for stereo, using identical mrf parameters. In *International Conference on Computer Vision*, 2003.

[45] P. Townshend. Substitute, 1966.

[46] S. Ullman. *High-Level Vision.* MIT Press, 1995.

[47] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition*, 2001.

[48] M.J. Wainwright, T.S. Jaakkola, and A.S. Willsky. Tree-based reparameterization framework for analysis of sum-product and related algorithms. *IEEE Transactions on Information Theory*, 49(5), May 2003.

[49] M.J. Wainwright, T.S. 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.

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

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

[52] Y. Weiss and W.T. Freeman. On the optimality of solutions of the max-product belief propagation algorithm in arbitrary graphs. *IEEE Transactions on Information Theory*, 47(2), 2001.

[53] E. Weisstein. Four-color theorem. In *Mathworld–A Wolfram Web Resource*. Wolfram Research Inc., 1999-2005. http://mathworld.wolfram.com/Four-ColorTheorem.html.

[54] C. Yanover and Y. Weiss. Finding the m most probable configurations using loopy belief propagation. In *Advances in Neural Information Processing Systems*, 2003.

[55] J.S. Yedidia, W.T. Freeman, and Y. Weiss. Understanding belief propagation and its generalizations. Technical Report TR-2001-22, Mitsubishi Electric Research Laboratories, January 2002.

[56] 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.