Learning to Segment Images Using Region-Based Perceptual Features

John Kaufhold and Anthony Hoogs
Visualization and Computer Vision Laboratory, General Electric Global Research Center
kaufhold@research.ge.com, hoogs@research.ge.com

Abstract

The recent establishment of a large-scale ground-truth database of image segmentations [11] has enabled the development of learning approaches to the general segmentation problem. Using this database, we present an algorithm that learns how to segment images using region-based, perceptual features. The image is first densely segmented into regions and the edges between them using a variant of the Mumford-Shah functional. Each edge is classified as a boundary or non-boundary using a classifier trained on the ground-truth, resulting in an edge image estimating human-designated boundaries. This novel approach has a few distinct advantages over filter-based methods such as local gradient operators. First, the same perceptual features can represent texture as well as regular structure. Second, the features can measure relationships between image elements at arbitrary distances in the image, enabling the detection of Gestalt properties at any scale. Third, texture boundaries can be precisely localized, which is difficult when using filter banks. Finally, the learning system outputs a relatively small set of intuitive perceptual rules for detecting boundaries. The classifier is trained on 200 images in the ground-truth database, and tested on another 100 images according to the benchmark evaluation methods. Edge classification improves the benchmark F-score from 0.54, for the initial Mumford-Shah-variant segmentation, to 0.61 on grayscale images. This increase of 13% demonstrates the versatility and representational power of our perceptual features, as the score exceeds published results for any algorithm restricted to one type of image feature such as texture or brightness gradient.

1 Introduction

The recent creation of a ground-truth database for the segmentation of complex images [11] has facilitated the quantitative analysis and learning of segmentation algorithms. Recent work in this area has demonstrated that boundary detection performance can be improved, over traditional edge detection methods such as Canny [3], by a learning approach based on the human database [10]. Martin et al. define local operators to measure gradient and texture, and compare their responses to boundaries delineated by humans. A variety of classifiers are compared in their ability to find salient feature relationships that distinguish true boundaries from significant edges. An interesting result of this work is that boundary detection can be improved with strictly local features computed on fixed windows.

In this work, we present a complementary approach to the same problem of learning boundary detection from this segmentation database. Our primary novel contribution is the use of a set of region-based, perceptual features to characterize boundaries and distinguish true boundaries from other edges. Our hypothesis is that further improvement in boundary detection depends on incorporating knowledge of perceptual image structure – knowledge that is not necessarily evident in any fixed window. We capture this structural information through features computed on the piecewise constant intensity structure in the image, which is estimated through a dense region segmentation. Unlike filter windows, regions have arbitrary shape and size, and can therefore find relationships between pixels at any distance in the image.

Each segmented region edge is either a true boundary or a spurious edge. Our goal is to find a set of features that robustly discriminates between edges that correspond to object boundaries (boundary edges), and those that do not (noise edges).
edges). Since each region edge is a boundary between two regions, we can measure properties of the edge by comparing properties of the regions. For example, edge contrast is the difference in the mean intensities of the regions.

More interestingly, we can capture the Gestalt perceptual characteristics of closure/common region, compactness, collinearity, constant curvature and parallelism. Each edge is naturally associated with the other edges in the same region, and those in neighboring regions. We exploit this relationship to compute perceptual features on edges and regions over arbitrary sizes and shapes in the image. This circumvents the inherent spatial limitation of every filter computed with a fixed mask (including those at multiple scales).

The perceptual features for each edge form a feature vector, and supervised learning methods are used to build a classifier that distinguishes boundary edges (edges marked by humans as boundaries) from noise edges (edges in the over-segmentation that humans did not mark as boundaries). The classifier is trained on 200 ground-truthed training images in the UCB database. To segment a novel image, dense region segmentation is performed, then each edge is classified as boundary vs. noise. In the output image, each edge pixel intensity is computed as a product of its edge probability from the initial segmentation and the class probability output by the classifier. We keep all parameters constant throughout.

Our experimental results indicate that region-based perceptual features are indeed important, and improve overall segmentation performance significantly. We can measure the contribution of perceptual features directly, by comparing the benchmark on the initial segmentation to the segmentation after classification. For instance, given only a dense region segmentation before learning, the benchmark F-score is 0.54 on the 100 test images (distinct from the 200 training images). This improves 13% with learning and some post-processing to 0.61, as discussed below. We present results on grayscale images, and are currently incorporating color.

Our method provides several advantages and novel contributions:

- We quantitatively derive perceptual “rules” for image segmentation in the form of joint relationships of perceptual features that are most salient for the general segmentation problem (as represented in the 200 training images). This is independent of any higher-level grouping method (graph cuts, e.g.). The classifier determines that high region contrast between two large regions indicates a true boundary with 77% accuracy, e.g..

- The method can localize texture boundaries precisely, i.e. down to a local gradient maximum. As mentioned in [10], texture gradients often produce a larger response on either side of a brightness edge than directly on the edge. Hence texture segmentation methods can perform poorly on texture-region boundaries [20]. Since we measure texture as a property of regions, our method is able to determine a single gradient boundary between smaller regions composing a texture, and larger regions that are part of a smooth object.

- We use a single low-level image representation, constant-intensity regions, for both smooth areas and texture. While filter banks have demonstrated great utility in texture analysis, we have shown in previous work that regions are also effective at characterizing coarse-level textures [8]. This representational unification enables the advantages described above.

- We train and evaluate against the UCB segmentation database. This database is the first common quantitative evaluation mechanism for the general segmentation problem that we are aware of. Even at its current size, it is a fundamental contribution and should enable significant advancements in this area.

To illustrate some of these points, we consider the image in Figure 1. The original image (upper left) exhibits multiple boundary types. There is a smooth region/textured boundary between the hair and the background foliage to the left, and a texture/textured boundary between the flowers and the background foliage. The shirt to bush boundary is more difficult to categorize, as the shirt may or may not be “textured” – it depends on the scale and definition of texture. This is an important point for our approach; we do not require a representational switch from smooth region to texture, as they are the same representation. Other approaches such as [17] must explicitly choose one or the other at each image location.

The upper right image shows the human segmentations, where brightness is proportional to the number of segmenters (7 total) that selected a pixel as a boundary. The lower left is our initial dense region segmentation. The three textured areas – shirt, flower, bush and background foliage – have many regions, but their density and shapes are different. The true boundaries are also present, but heavily outnumbered by the noise edges.

The lower right is our final boundary detection image, created from the dense segmentation by filtering edges according to the boundary/noise classifier. No parameter values were tuned for this example, and all parameters were fixed for both training and testing. The edges within each texture are greatly reduced, while the edges between textures are strong and highly localized. Note that this result was produced without explicit grouping, as each edge was classified using properties derived from its two bounding regions, and their neighbors.

Some segmentation errors remain, however. One significant limitation of our approach is that we cannot exceed the representational power of the initial region segmentation. This is analogous, in filter-based methods, to the descriptive power of the filter set. A consequence of this is that we cannot distinguish between fine-grained textures with only region-based features. For texture to be captured in regions, texture elements must be above the gradient threshold to
form an edge in the initial segmentation. For low contrast edges of small texture elements, the edge will be smoothed over. However, this is rarely an issue in natural images, as demonstrated by our results. A second consequence is that all salient edges must be present in the initial segmentation. We compensate for this by segmenting at high sensitivity, and relying on the classifier to discard noise edges. In future work we intend to incorporate higher-level grouping to form boundaries across gaps due to missing edges.

In addition to Martin et al., there are numerous recent efforts in learning how to segment images. Traditionally, perceptual grouping has focused on edges, and recent work has demonstrated that learning and graph partitioning can be effectively combined to form large groups of edge features enhanced with photometric information [15]. Zhu et al. [17, 4] have derived a probabilistic learning framework for segmentation and detection that incorporates various image models, including smooth regions, contours, texture, and classes of objects. The method is also applied to finding a set of image patch exemplars that are fitted to local image areas for a novel form of image description [7]. Similarly, Borenstein and Ullman [2] learn which image fragments are salient for segmenting known classes of objects. Properties of regions similar to those presented here have been used to learn to recognize objects on semi-supervised training data [5]. Our approach differs from these in that we are attempting to find a common set of image features that is optimal for general image segmentation.

The paper is organized as follows. In section 2, Dense Region Segmentation, we describe the initial dense segmentation background and its implementation in this paper. From the dense segmentation, we compute features on pairs of adjacent regions and their neighborhoods—this feature vector computation is described in section 3, Characterizing Edges. In section 4, Learning Boundaries, we use the UCB training database to learn perceptual rules for deciding whether a feature vector corresponding to a novel pair of regions is a real or spurious edge. We then present performance improvement metrics on the classified boundaries in Results and list the major points developed here in Conclusions.

2 Dense Region Segmentation

To compute our perceptual features, we require a partition of the image into regions. To do this, we use a variant of the seminal work by Mumford and Shah [13, 14] on boundary detection in images. The primary motivation for choosing this approach is its ability to join edge fragments into closed edge contours that partition the image. The variational functional originally proposed is shown in equation (1).

\[
E(u, B) = \int \int_{R} \beta^{2}(u - g)^{2} \, dxdy + \int \int_{R/B} \left( |\nabla u|^{2} + \beta \nu |B| \right) \, dxdy \tag{1}
\]

In equation (1), \( \beta \) and \( \nu \) are scalars, \( g(x, y) \) is the intensity at \((x, y)\), \( u(x, y) \) is the piecewise constant field, and \( B \) is the set of boundary curves. The \( B \) term will develop curves to suspend the smoothness constraint where the gradients of \( u \) are high enough.

Although it may be true that the pair of approximating image, \( u(x, y) \), and binary edge process, \( B \), that minimize the energy, \( E(u, B) \), form a theoretically defensible segmentation of the image [12], the mixed nature of \( u \) (continuous field) and \( B \) (curves in image domain) make the minimization of \( E(u, B) \) via descent methods intractable in practice. To address the difficulties in minimizing the original functional, it was shown that limiting solutions of a related functional (where the binary edge process, \( B \), is replaced with a continuously valued edge image, \( v(x, y) \)) are equivalent to solutions of the original functional [1]. We will refer to this alternate functional as the Ambrosio-Tortorelli functional.

In [16], shock formation, links to curve evolution, scale space, and segmentation regularity motivate replacing the Ambrosio-Tortorelli functional’s L2 norms on data fidelity and smoothness with L1 norms [16]. This altered functional, called the Unification functional in this paper, is shown in equation (2).

\[
E_{\rho}(u, v) = \int \int_{R} \left\{ \beta |u - g| + \alpha (1 - v)^{2} ||\nabla u|| + \frac{\rho}{2} ||\nabla v||^{2} + \frac{\nu^{2}}{2\rho} \right\} dxdy \tag{2}
\]

In equation (2), \( \alpha, \beta, \) and \( \rho \) are scalar weights, \( g(x, y) \), and \( u(x, y) \), are as above, and \( v(x, y) \) is the continuously valued \((0,1)\) edge field. A visual argument is made in [16] in favor of segmentations corresponding to minima of the Unification functional over segmentations corresponding to minima of the Ambrosio-Tortorelli functional.

The L1 norms and the cross term in the smoothness constraint make the energy surface in Eq. 2 nonconvex with flat faces. Another energy with flat faces in its energy surface is the total variation energy (L1 penalty on the derivative side constraint). In [18], Vogel shows how to use half-quadratic regularization, as introduced in [6], to obtain a minimum of the total variation energy in a coordinate descent framework. In [9], Kaufhold shows that equation (2) can be minimized in a coordinate descent framework alternatively estimating the
edges holding the smooth field constant and then estimating
the smooth field given the edges estimated in the previous
iteration. The step of estimating the smooth field holding the
dges constant in [9] can be shown to be an extension of the
half-quadratic minimization in the spirit of [18].

The first step in our boundary detection method is to over-
segment all training and testing images in the database by
minimizing equation (2) via half-quadratic regularization in
a coordinate descent loop as described in [9]. The original
gayscale image intensities are linearly rescaled so values of
g(x, y) lie in the range 0-500. The Unification functional
parameter values used to segment all images were: \( \alpha=10, \)
\( \beta=0.04, \) and \( \rho=0.02. \) The algorithmic settings corresponding
to the minimization algorithm in [9] and [18] were as fol-
loows: the half-quadratic regularizer was \( \beta_{HQ}=0.1, \) and the
stopping criterion was \( \frac{|w_{k+1}-w_k|}{|w_k|} > 0.001 \) in the coordinate
descent loop. To arrive at a pair of \( u \) and \( v \) fields from a
single 321x481 pixel image in the database takes approxi-
mately 90 seconds on a 1.7GHz PC running Windows 2000.
Example edge maps, \( v(x, y) \) computed from test images in
the database are shown on the third row of Figure 3. Note
that the edge map in Figure 3 forms predominantly closed
regions around piecewise constant patches in \( g(x, y) \), which
is a desirable property for the subsequent region and feature
vector computations.

3 Characterizing Edges

Once the image is decomposed into piecewise-constant
intensity regions, we consider how true boundaries are
embedded in the region graph. Our approach is based on a few
key observations:

- True boundaries almost always correspond to measur-
able intensity gradients and corresponding edges in our
segmentation. This has been validated quantitatively on
the segmentation ground-truth.

- Boundaries are characterized by the regions bordering
them. While an edge by itself has limited information,
the region graph containing the edge usually provides
important spatial and photometric context for the edge.

The second property is our primary motivation for explor-
ing how to segment images using region-based edge features.
Various filters as in [10] can provide complementary infor-
mation, and should generally be included in a complete sys-
tem, but are not our focus here. In previous work [8] we have
compared perceptual region-based features to filter banks for
texture recognition, and demonstrated that perceptual fea-
tures have significant advantages when the texture categories
are broad, such as man-made objects vs. vegetation.

For boundary detection, we are interested in comparing
the two regions \( R_1 \) and \( R_2 \) that border an edge \( E. \) If \( E \) is a
true boundary, there are three possible relationships for \( R_1 \)
and \( R_2: \) 1) region-region; 2) region-texture; and 3) texture-
texture. We must ensure that our features can capture suffi-
cient information to represent these cases. If \( E \) is not a true
boundary, then it is within a textured area or an object.

One of the main difficulties of segmentation and percep-
tual grouping is that local image information is often insuffi-
cient to distinguish between these cases. Some strong edges
between large, smooth intensity regions may lie within an
object, and humans do not mark these boundaries, as we tend
to describe images at an object level. This motivates learning
on a training set and using image features that capture spa-
tial and photometric relationships at arbitrary, image-specific
scales as our region features do. We do not, however, explicit-
ly attempt to classify \( E \) into one of the three categories.
Instead, the categorization happens naturally when learning
true vs. noise edges, as discussed below.

To characterize \( E, \) we compute properties of its border-
ing regions \( R_1 \) and \( R_2. \) The features we compute on each
region are listed in Figure 2. The single region features com-
puted are similar to those used in other efforts [5, 19]. These
features are found to be useful, particularly for large regions.

Each region \( R \) has a set of neighbors \( N(R) \) that share
edges with \( R. \) To characterize the neighborhood, we com-
pute statistics on the single-region features for \( N_i \in N(R) \)
as shown in the second row of the table.

For smaller regions, the most salient features prove to be
those comparing \( R \) to \( N(R), \) as detailed in the third row of
the table. These differential features are effective for captur-
ting texture on a region-level scale.

Since texture is inherently dependent on spatial scale, it
may not be desirable to include large regions in \( N(R) \), par-
ticularly when area \( A(R) \) is small. This happens frequently
on region-texture boundaries, where the regions embedded
in texture are usually small compared to regions on a smooth
surface. When large regions are included in \( N(R), \) their
statistics can distort the characterization of \( N(R); \) this is
analogous to a filter window including pixels from a smooth
region next to a textured area.

Our approach entails a direct solution, which is to filter
\( N(R) \) by region area. The area threshold is the texture scale
parameter embedded in all texture computations; regions
larger than this are considered to be smooth surfaces. How-
ever, it is not a hard threshold, as we do not explicitly clas-
ify regions as texture or non-texture. Instead, the classifier
may select combinations of any feature types in its attempt
to distinguish true and noise edges. In our experiments, we
exclude regions with area \( > 400 \) pixels from \( N(R), \) impro-
ving performance modestly (1-2% on the UCB benchmark).

The relatively small improvement indicates that the algo-

ithm performs well without any explicit texture scale pa-

rameter, which is an advantage in many applications.

With feature vector \( \mathbf{F}_R \) composed of the features in Fig-
ure 2 for region \( R, \) most of the feature vector \( \mathbf{F}_E \) for edge
\( E \) is computed as \( \mathbf{F}_E = |\mathbf{F}_{R_1} - \mathbf{F}_{R_2}|. \) This feature vector
is augmented with some additional features such as the max
area in the region pair, min area in the region pair, etc. This
feature vector is calculated for each edge in the dense region segmentation, and is the input to training and segmentation. The length of the vector is 148.

4 Learning Boundaries

The database of human segmentations enables the learning of which edges correspond to human-designated boundaries. The UCB database contains as ground truth for each image, a number of human segmentations (typically between 5 and 9). The human segmentations are represented as label maps, \( L_{HS}(x, y) \), one per segmentor, where there are typically on the order of 20 contiguous labels per image. There are human segmentations for 200 training images and 100 test images, which enables supervised learning and quantitative performance evaluation.

In training, each training image is segmented into dense regions and their interposed edges. The feature vector \( \mathbf{F}_E \) is computed for each edge. To obtain ground-truth labels for the computed edge feature vectors, human label maps, \( L_{HS}(x, y) \), are used. Associated with each edge are the centroid locations of \( R_1 \) and \( R_2 \). The labels in each human segmentation for those locations are either the same or different. If the human labels are the same, \( \mathbf{F}_E \) does not correspond to a true boundary and is counted in the “within” label class, \( W \). If the human labels are different, \( \mathbf{F}_E \) corresponds to a boundary identified by that human segmentor and is counted in the “across” label class, \( X \). Each human segmentation is treated independently, resulting in multiple (possibly conflicting) labels for the same \( \mathbf{F}_E \). At the end of the training data gathering stage are two large lists for each image consisting of typically between 50,000-200,000 vectors with \( W \) labels and 4,000-15,000 with \( X \) labels.

There are many possible choices of classifier for this problem. We have chosen a decision tree, because it provides quantitative and intuitive information about feature importance. In fact, its output can be directly interpreted as perceptual “rules” for finding boundaries. It is also very fast, and capable of handling high-dimensional, non-Euclidean feature spaces with hundreds of thousands of training samples. Martin et al. [10] studied various classifiers for the segmentation problem, including decision trees, and determined that all are statistically equivalent.

To train the classifier, we first derive a balanced set of \( X \) and \( W \) \( \mathbf{F}_E \) exemplars from each image’s labeled training data. The large list of \( W \)-labeled \( \mathbf{F}_E \)’s always has many more samples than the \( X \)-labeled \( \mathbf{F}_E \)’s, so for every image, we sample as many \( \mathbf{F}_E \)’s from the \( W \) class as we have training exemplars of the \( X \) class. However, because fine-textured regions will generate disproportionately large numbers of training exemplars for the \( W \) class compared to large region pairs in the \( W \) class, which are arguably more important to learn, we sample nonuniformly. Specifically, for small region pairs (minimum region size in pair < 400 pixels), we sample at approximately 1/4 the frequency we sample the large regions pairs (minimum region size in pair > 400 pixels). There is a smooth increase in sampling frequency from small minimum size to large minimum size. After this nonuniform sampling, we have a balanced set of \( X \) and \( W \) exemplars for each image. All the training data from all the images are then collected into a balanced list of \( W \) and \( X \) exemplars for the set of training images in the database. There are approximately 1.7 million labeled training exemplar feature vectors (850,000 \( W \) and 850,000 \( X \)) for the balanced set of training data.

The balanced training data is gathered and randomly sam-
pled to build a CART decision tree. To prevent overfitting, the minimum number of training samples in a leaf node is set to between 5 and 200. To give some perspective, the largest tree we grew corresponded to a minimum bin size of 5 with 500,000 samples; that tree grew 20,261 leaves scoring 0.55 on the benchmark. A tree of 283 leaves leading to superior benchmark performance (0.59) was grown from 200,000 samples with a minimum bin size of 20, however, indicating that the more complex decision structure of the larger tree is overfitting. In general smaller trees are more desirable, as an accurate small tree indicates its features are highly salient for distinguishing true boundaries.

The three true edge types provide an interesting experiment in feature selection. We expect the classifier to select single-region features to classify edges between two large regions, while neighborhood features will dominate the classification of edges between small regions. Without informing the classifier of these edge types, this is indeed what we observe. At the top of the tree the classifier coarsely segments the joint feature space into the three categories of edges, using region size and centroid distance, then divides each sub-space into true/noise edges using a variety of features.

The UCB benchmark designates 100 ground-truth images for testing. Once the decision tree is generated, for each image in the test set (or any novel image), we compute the dense initial segmentation, and generate image in the test set (or any novel image), we compute the space into true/noise edges using a variety of features.

To compute the final edge image, we use the initial segmentation as a prior, and threshold the boundary edge classification probability, $P_E(X)$:

$$ I_e(x, y) = I_R(x, y) C(E) $$

where

$$ C(E) = \begin{cases} 0 & \text{if } P_E(X) < T_X \text{ or } E = \Lambda \\ P_E(X) & \text{otherwise} \end{cases} $$

$E$ is the edge at location $(x, y)$, if any. This should produce a boundary image with high values at true boundaries, and low values elsewhere. The algorithm is not particularly sensitive to the threshold $T_X$; at $T_X = 0.5$ and $T_X = 0.2$ produces equal benchmark scores. This occurs because the benchmark scoring protocol incorporates multiple sensitivity levels.

In a final post-processing step, we discard short, isolated edges. Specifically, when a summed connected edge group probability is greater than a threshold, we retain those edges; else we reject them. This may induce errors by removing broken boundaries, but results show a net gain. This additional post-processing boost shows that more gains can still be gleaned in our approach after learning. Filtering disconnected segments is only one way of enforcing a high-level rule influencing edges to connect (region grouping is another). Simply, dense segmentation captures low-level properties; learning incorporates mid-level properties; further grouping or post-processing effects high-level decisions. The final edge image is then benchmarked using the UCB Benchmark tool, v1.2.

5 Experimental Results

The UCB benchmark specifies 200 training images, 100 test images, a scoring protocol, and tools to compute the score. Hence it is an effective method for evaluating performance on a standard data set. The benchmark uses precision recall curves to measure performance. The aim of the precision recall curve computation is to describe the agreement between ground-truth boundaries delineated by humans and boundary images computed via boundary detection algorithms. Each computed boundary image between 0 and 1 is thresholded at a number of levels equally spaced between 0 and 1, and at each threshold, the correspondence between human segmentors and the machine-generated boundary is computed. The x-axis of the precision recall curve defines the recall of the curve. For recall values close to 1, very few boundaries are missed; for values close to zero, almost all are missed. Similarly, the y-axis defines the precision of the algorithm on the boundary image. For precision values close to 1, almost all the machine-generated boundaries correspond to human boundaries (no false positives); for precision near 0, almost all detected boundaries are false positives. The aim of a machine boundary detector, then, is to produce precision recall curves that lie in the top right of the plot.

In general, the precision recall curves will cross, making a quantitative comparison of boundary detectors problematic. For this reason, an F-score is computed from the precision recall curve for each image. The aggregate F-score for any given algorithm is computed by aggregating all the precision recall curves for the test images, and from the composite precision recall curve, producing a single F-score. When we refer to algorithm scores below we are describing the scores for this aggregate F-score for the algorithm. UCB’s benchmark web page cites the highest performance boundary detection algorithm as one that fuses a “texture gradient” and a “brightness gradient”. The F-score for that algorithm is 0.63. The component boundary detectors, “brightness gradient” and “texture gradient” score 0.60 and 0.58, respectively. Thus, arguably, the learning used to fuse those boundaries provided a five percent score increase.

In the results presented here, our edge classifier is trained on 200 grayscale training images from the UCB database and is tested on the 100 testing images. Specifically, 200,000 region-region pair feature vector exemplars are chosen from the 1.7 million training exemplars from the UCB database and a CART decision tree with a minimum bin size of 20 exemplars per bin is trained, producing 283 leaves. This decision tree is then used to create boundary images as described above, where each pixel indicates the probability of being on
Figure 3. The 6 images across the top row are representative test images from the UCB benchmark database. The 6 images across the second row show the corresponding human boundaries, the third row shows the dense initial segmentations, and the boundaries found using our approach are shown on the bottom row.

a human-designated boundary.

Representative UCB database test images are shown in Figure 3; across the second row are the human boundaries, across the third row are our dense region segmentations, and across the fourth row are our final boundary estimates.

Figure 3 illustrates a mix of promising results and challenging hurdles not yet overcome. Perhaps the most novel contribution of our perceptual classifier is that region properties are used to classify the edges between each region pair as a true edge or a false edge, on all 3 types of boundaries (region-region, region-texture, texture-texture). Note that the initial dense segmentations contain a preponderance of false positive edges and that the edges after classification are considerably sparser while retaining most of the real edges identified in the dense initial segmentation.

An example of the preservation and localization of a true boundary between two textures is evident in the bridge image (second column). It is particularly difficult to reject high-gradient edges within a texture, particularly without explicit texture recognition, while maintaining texture-texture boundaries. In this image, the true texture-texture boundary between the bridge and the vegetation is preserved, while the strong edges within the textures are filtered out.

In addition, the spatial scale of the texture of the slats on the bridge is quite large (~50 pixels), which would require a very large filter window to detect. Our region-based approach finds the common texture relationship across arbitrary scales determined by the image data, and rejects the edges internal to the bridge.

Note also that even after edge classification, the building window textures are classified by the human segmentors as one object, but split into multiple regions in our final seg-
mentation. Our texture approach fails in this example to reject those edges because each window is composed of a number of smaller regions, whereas the adjoining wall is typically a single region. This combination makes for a strong texture-region boundary and those edges are not rejected.

The precision recall curves for both the initial dense segmentation and our learned boundary detector are shown in Figure 4. Also shown (right) is a comparison with the popular Canny approach to edge detection (scoring 0.57) and the best UCB algorithm (scoring 0.63). The total F-score as computed by the Benchmark (version 1.2) is 0.61 for our boundary detector (WX). Note that the F-score for our initial dense segmentation is only 0.54. Learning alone raises this score to 0.59, and an additional boost to 0.61 is provided by filtering fragmented edge segments. Looking purely at score improvement from learning on regions compared to learning on fixed windows, the region-region pair learning provided a boost of +9% compared to a 5 or 6% boost for fusing boundary detectors on fixed windows. This 0.61 benchmark score is to our knowledge the highest reported score for the benchmark.

The closest single boundary detector performance we know of is the “brightness gradient”, which scores 0.60.

6 Conclusions

We have shown in this paper that perceptual information derived from region-region pairs in a dense initial segmentation can be used to improve boundary detection performance by 13%, as measured by the UCB segmentation benchmark tools. A simple approach to learning true vs. false boundaries in a region-based perceptual feature framework is presented here, and that simple learning approach yields very good results on segmentation performance, including the localization of texture-region boundaries. However, the most motivating aspects of this work are perhaps the experiments that have not yet been done. Other classifiers such as Adaboost may improve performance, and the feature vector should be extended with more information about the edge itself, such as curvature and complexity. Multiple spatial scales should be considered, and higher-level grouping us-

References