Inconsistencies in Edge Detector Evaluation

Lance A. Forbes and Bruce A. Draper
Computer Science Department
Colorado State University
Fort Collins, CO 80523

Abstract

In recent years, increasing effort has gone into evaluating computer vision algorithms in general, and edge detection algorithms in particular. Most of the evaluation techniques use only a few test images, leaving open the question of how broadly their results can be interpreted.

Our research tests the consistency of the receiver operating characteristic (ROC) curve, and demonstrates why consistent edge detector evaluation is difficult to achieve. We show how easily the framework can be manipulated to rank any of three modern edge detectors in any order by making minor changes to the test imagery. We also note that at least some of the inconsistency is the result of the erratic nature of the algorithms themselves, suggesting that it is still possible to create better edge detectors.

1 Introduction

Several frameworks have been proposed to evaluate edge detectors.[1, 2, 5, 7, 8, 11] A good edge detector evaluation framework should measure the progress of research, direct future research, or select the best detector for an application. The recently proposed methods use a variety of approaches, but none have tested the consistency of their results. A good evaluation framework must provide a consistent evaluation across the scene variability expected for a task; otherwise, the evaluation will depend entirely on the sample of images instead of the performance of the edge detector.

Several properties of edge detection make designing an evaluation framework difficult. First, evaluating a result implies there is a definitive answer. For an edge detector this means there is a definition of the "true" edges in an image. This is seldom the case. Even a simple stack of blocks (refer to Figure 1) demonstrates the ambiguity of the definition of edges. The perimeters of shadows or reflections create intensity differences that may or may not be considered edges. Since edge detectors are used within larger systems, a functional definition of an edge can be determined by the next step in the process. For example, if the larger system is performing model-based object recognition, the edges are the projection of the object's geometric boundaries onto the image plane (refer to Figure 2). In this case, the perimeters of shadows and highlights are classified as clutter. The task specific definition of an edge provides a necessary definition of ground truth for the evaluation framework.

Edge detector parameters create another problem for evaluation frameworks. Parameters fundamentally change the performance of most edge detectors, and the best settings are dependent on the image. An evaluation framework must either select parameter settings based on a criteria that is fair to all edge detectors, or evaluate the edge detectors across a broad range of settings.[1, 4] The evaluation framework must address these issues and provide a consistent score for any image in the task domain. In other words, minor variations in images that may be seen when performing the task should not cause the edge detector evaluation to change.

The issues of ground truth and parameter selection have been addressed by several evaluation frameworks, but consistency has not. For example, the Receiver Operating Characteristic (ROC) Curve Evaluation Framework uses a three-valued ground truth system to address the ground truth problem and parameter space search to address the parameter problem. However, the framework's consistency has not been tested. To test the consistency of the ROC Curve Evaluation Framework we decided to incrementally change a single property of a scene consistent with the task of model-based object recognition and re-perform the evaluation after each change. If the framework is consistent across the expected range of images, the relative score of the edge detectors should not change. We tested the ROC Curve Evaluation Framework because it is one of the leading edge detector evaluation frameworks proposed and the authors claimed observations provided by the framework were remarkably

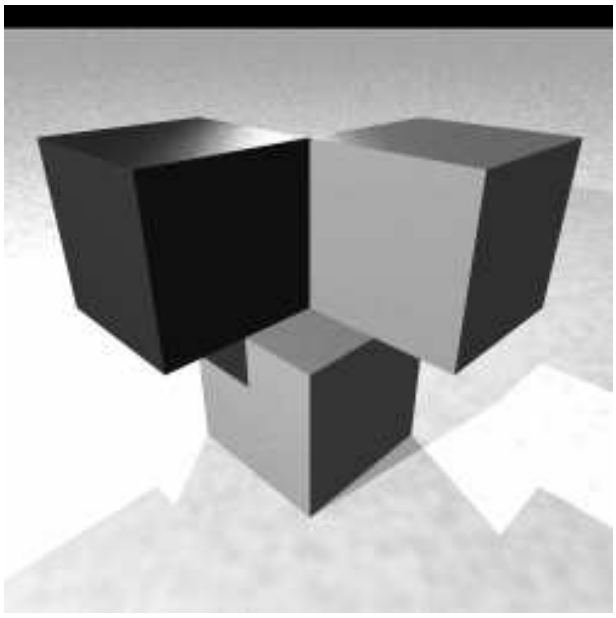

Figure 1: Test Image

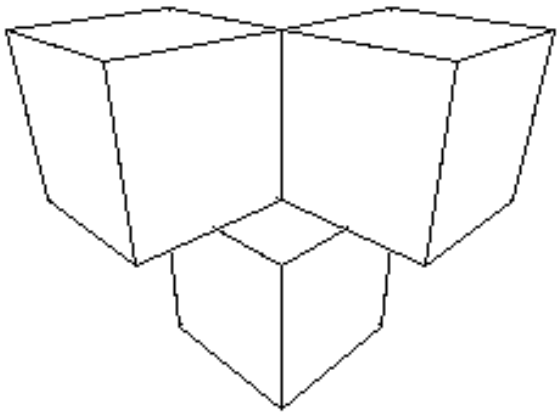

Figure 2: Ground Truth Edge Image

similar across image categories.[4] An equally interesting candidate for future research is the global measures of coherence framework proposed in [1]. The Canny, Susan, and Rothwell edge detectors were selected for the test because they represent competitive, but different, methods of edge detection. Canny uses non-maximal suppression and hysteresis edge following.[3] Susan uses circular masks to find pixel regions with the same relative brightness.[9] Rothwell uses topologically-based edge thinning to perform dynamic non-maximal suppression.[6] The methods reported here could be used to test any of the referenced frameworks using any combination of competitive edge detectors.

1.1 ROC Curve Evaluation Framework

The ROC Curve Evaluation Framework is designed to objectively and repeatably evaluate edge detectors by fairly sampling an edge detector's parameter space and accurately rating edge detectors consistent with a specific task. The evaluation framework uses real imagery and manually specified ground truth to evaluate each edge detector. Due to the inherent ambiguity of edge pixels commonly found in real imagery, the ROC curve framework uses a three-value ground truth. Each pixel in the image is marked as either edge, non-edge, or don't-count. Pixels marked as don't-count are the pixels where the edge status appears ambiguous to the operators specifying the ground truth.[2, 4]

Once the three-valued ground truth image has been manually created, each edge detector is run on the image and the result is compared to the ground truth. To fairly sample the parameter settings for each detector, the parameters are equally partitioned. Each parameter is partitioned into four equal parts creating a uniform sampling of the parameter space. For example, an edge detector with two parameters where parameter A has the range 0 to 1 and parameter B has the range 3 to 4 would be run with the parameter settings (0, 3), (0, 3.25), (0, 3.5), (0, 3.75), (0, 4), (0.25, 3), (0.25, 3.25), (0.25, 3.5) ... (1, 3.5), (1, 3.75), and (1, 4). This parameter space partitioning represents a 4x4 partition of the parameter space. To find the parameter space partitioning that generates the best edge detection for a given edge detector, the space partitioning is searched in the manner of a tree graph. The level of the graph below the 4x4 partition has one branch for the 7x4 partition and one branch for the 4x7 partition. The edge detector is evaluated using both partitions and the best is selected for the next partitioning of the space—a basic steepest descent search. If the 7x4 partition has the best evalua-

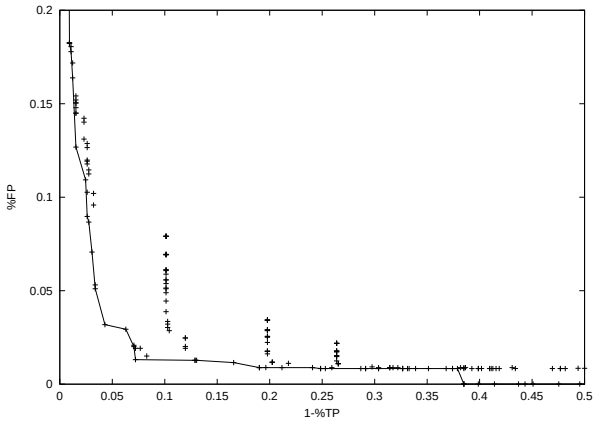

Figure 3: Canny ROC Data and Curve

tion, the next choice would be either the 13x4 or 7x7 branch of the graph.

The ROC framework uses the true positive and false positive edge pixel counts to generate the ROC curve (refer to Figure 3). For each edge image generated by a particular parameter setting the true positive and false positive edge pixels are counted and a point on the ROC curve is plotted. For example, when an edge detector with two parameters is evaluated using a 4x4 partition of the parameter space, sixteen points are plotted on the ROC plot. A cloud of points is plotted as the tree-graph of possible partitionings is searched and the space is further divided. The ROC curve is the edge of the points closest to the true positive and false positive axes. The evaluation function for a particular partitioning of the space is simply the area under the ROC curve. The depth-first search of the parameter space continues and the shape of the ROC curve is refined until the area under the curve (AUC) fails to improve by more than five percent. This final AUC is the score for the edge detector. Figure 3 shows a cloud of points and ROC curve obtained by evaluating the Canny edge detector on an image similar to Figure 1.

The data and conclusions presented in [4] indicate the ROC curve evaluation framework meets the authors criteria of minimizing evaluator involvement, repeatability, fairly sampling the parameter space, and accurately rating edge detectors consistent with a specific task. It also measures the difficulty of setting the parameters for each edge detector. The framework presents a positive step toward the difficult goal of task specific edge detector evaluation. However, the ROC curve evaluation framework is simply a sophisticated benchmark and benchmarks can be misled.

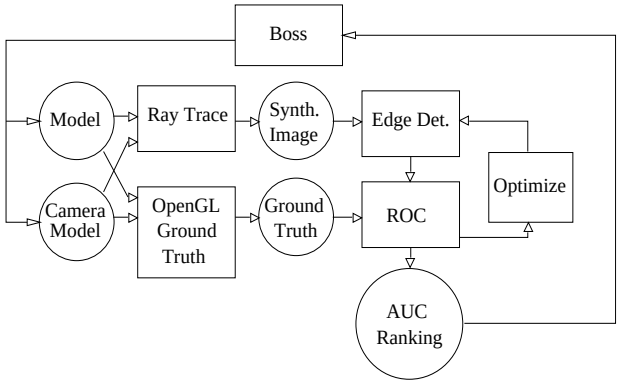

Figure 4: Data-flow diagram

2 Experiment Set-up

To test our hypothesis that small changes to an image can cause inconsistent evaluations, we implemented a supervisory program to automate the ROC curve evaluation framework using synthetic imagery. The program, called Boss, is shown in Figure 4. Boss generates a single scene file describing the test image. The model file is used to generate the test image and the ground truth image (refer to Figure 1 and 2). The test image and the ground truth image are used as inputs to run the ROC framework on each of three edge detectors. After recording the scores, the scene property is incrementally changed and the process of generating the test image and evaluation is repeated. The results are shown as a graph of the AUC score versus the property for each edge detector. Figure 5 shows an example where image resolution was changed. Several scene properties were tested, including surface reflectivity, light position, lens aperture and resolution. This paper focuses on resolution because incrementally changing this property provided the most dramatic results we have seen so far.

To facilitate automatic control of the test imagery and a precise two-value definition of ground truth, we generated the test imagery using an open-source graphics engine[1] and the ground truth imagery using Mesa OpenGL. To generate the most realistic imagery feasible, the maximum realism setting was used with adaptive oversampling set to 1% variability and every object was textured including the ground plane (refer to Figure 1). We used OpenGL's polygon offset feature described in [10] to create an edge image with hidden line removal as shown in Figure 2. Because OpenGL only uses polygons to model objects, the ground truth image could only accurately repre-

[1] The Persistence of Vision Ray-tracer is an open source program available from www.povray.org.

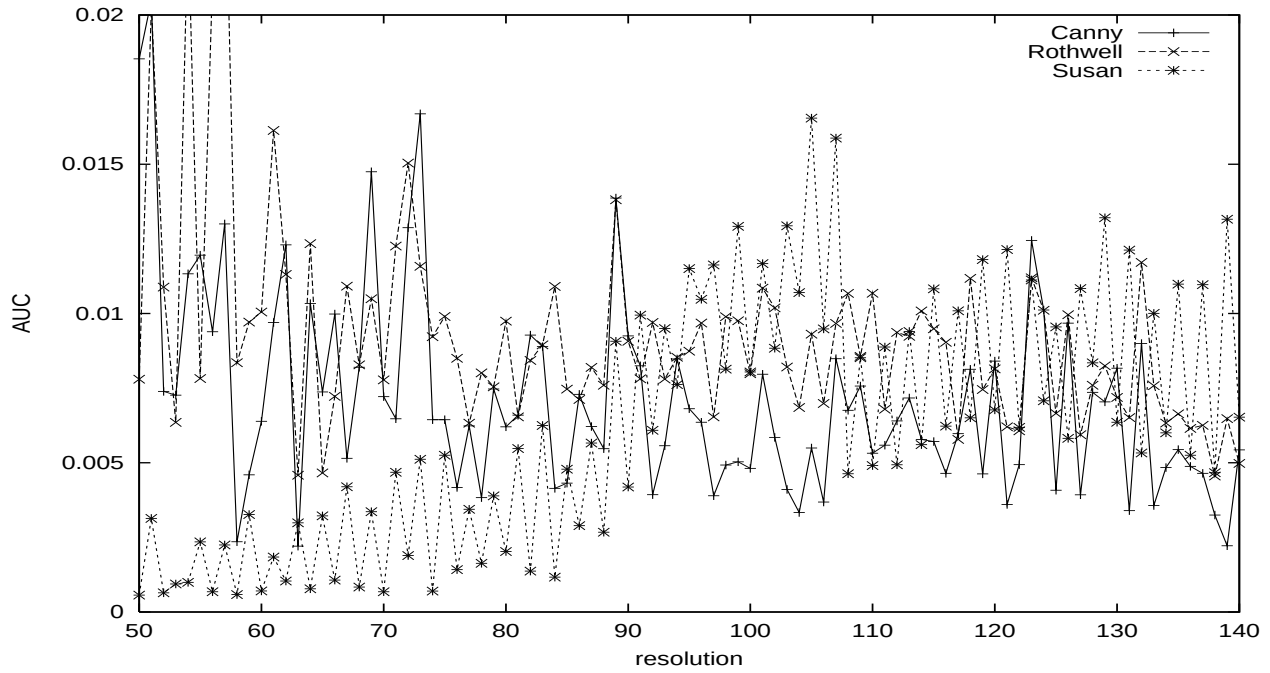

Figure 5: AUC v. Resolution

sent polygonal objects.

One of the weaknesses of the original ROC Curve Evaluation Framework is the three-valued ground truth. Much of the image is arbitrarily excluded from evaluation when the delineation of edge and non-edge pixels is not clear. It is unclear to us whether eliminating parts of the image biases the evaluation process. Using synthetic imagery and assuming the task of model-based object recognition allowed us to use a less arbitrary two-valued ground truth. Every pixel in the image is either edge or non-edge.

The test image from POV-Ray and the ground truth edge image from OpenGL were used to evaluate each edge detector using the ROC framework. The only difference from the original evaluation process in [2] was the use of two-valued ground truth rather than three-valued. Otherwise, the parameter search process was identical, including starting with four partitions per parameter and stopping based on a five-percent improvement criteria. The range of parameter values used for each edge detector is shown in Table 1. To verify that the edge detectors were operating correctly, edge images were generated using images from the University of South Florida's ROC Curve Evaluation web site and compared to edge images also on the web site.[2]

Detector	Param 1	Param 2	Param 3
Canny	σ=0.5-5.0	Tl=0.0-1.0	Th=0.0-1.0
Rothwell	σ=0.5-5.0	T=0-60	α=0.0-1.0
Susan	T_{brt}=1-100		

Table 1: Detector Parameter Value Range

Once the entire process of generating a test image, generating the corresponding ground truth image, running each edge detector within the ROC curve evaluation process, and recording the results is complete, the parameter representing a scene property is incremented to the next value and the process is run again. Several properties were evaluated independently, including reflection, oversampling, resolution, and lens aperture. In this paper we focus on resolution because it provides the most dramatic results. To change the test image resolution, the scene shown in Figure 1 was rendered with a different number of pixels on the image plane. The corresponding ground truth image was rendered with the matching number of pixels in the image buffer. For all images the number of vertical and horizontal pixels was always the same.

[2]The test images and edge detectors are available from: marathon.csee.usf.edu/edge/edgecompare_main.html

Detector	Rank Count 1st	2nd	3rd	Rank Changes
Canny	138	42	20	63
Rothwell	6	131	63	94
Susan	56	27	117	85

Table 2: Number of times each edge detector was ranked 1st, 2nd, 3rd and the number of times the rank changed between 50 and 250 pixels of resolution

3 Results

Incrementally increasing the image resolution by one pixel frequently changed the edge detector evaluation results. Figure 5 shows the results for a range of image resolutions from 50 to 140 pixels for the three edge detectors. In several cases, changing the resolution makes the best edge detector become the worst. For example, Susan consistently received the best (lowest AUC) score from 50 to about 90 pixels of resolution. However, from about 130 to 200 pixels of resolution Susan consistently has the worst score. In another case, Canny received the worst score at 123 pixels of resolution, then received the best score at 125 pixels of resolution. Table 2 shows the number of times an edge detector was ranked first, second, or third and the number of times each edge detector's rank changed. To better understand the inconsistency, we examined several of the ROC curves that generated the AUC values and the points that generated the ROC curves. The following example illustrates how an edge image can dramatically change due to a resolution change of only one pixel.

The Canny detector was evaluated on two nearly identical images; the only difference is rendering one image at 203 pixels of resolution and the other at 204 pixels of resolution. The two images are identical to Figure 1 except for resolution. The Canny edge detector was run on the images with the parameter settings ($\sigma = 0.5, T_{low} = 0, T_{high} = 1$). Although the images appear nearly identical to our eye; Canny, using the parameter setting above, responds very differently, as shown in Figures 6 and 7.

Canny's dramatically different response is due to the small changes in the gradient caused by the resolution change and the unconventional parameters. To create an edge image, Canny performs Gaussian blurring, creates a gradient image, creates a non-maximal suppression (NMS) image from the gradient image, and finally performs hysteresis edge path following based on the NMS image. The gradient values for the 203 and 204 pixel images are slightly different and, as a result, the NMS and hysteresis images are slightly

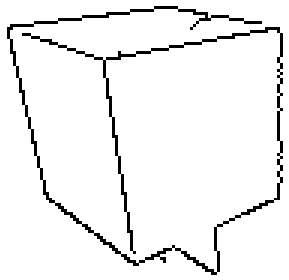

Figure 6: Canny Edge Image from 203 Pixel Image

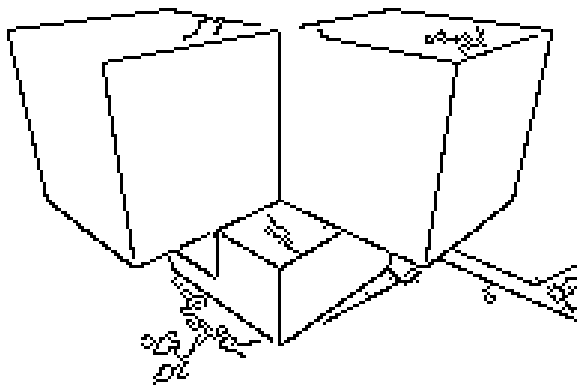

Figure 7: Canny Edge Image from 204 Pixel Image

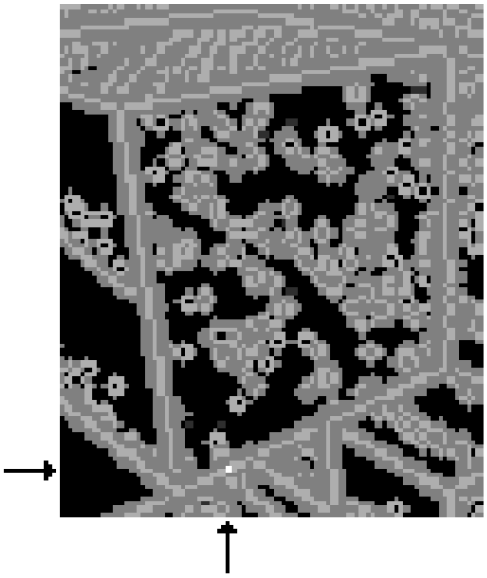

Figure 8: Canny NMS Image from 203 Pixel Image

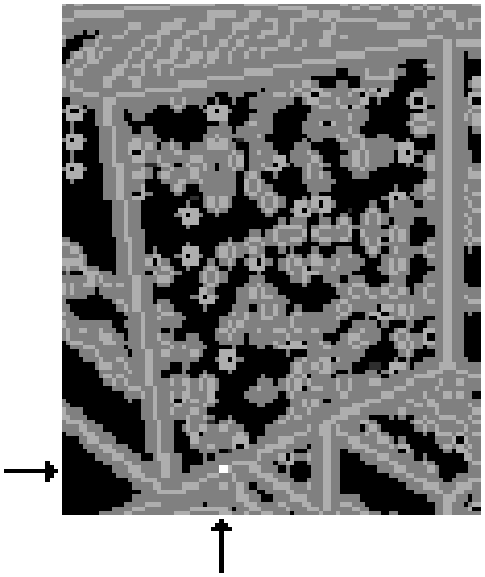

Figure 9: Canny NMS Image from 204 Pixel Image

different. This sensitivity of the gradient image was previously reported by Rothwell et al.[6] Figures 8 and 9 show an enlarged portion from the center of the combined NMS and hysteresis image for the 203 and 204 pixel images. The small change in the NMS image has a more pronounced effect due to the high setting for the parameter T_{high}.

T_{high} is used to determine the minimum gradient magnitude where a pixel is labeled an edge. If T_{high} is 0.9, the pixels in the highest 90th percentile of the gradient values are considered edge pixels. T_{low} is used in the same manner, but the pixels between T_{low} and T_{high} are labeled probably-edge pixels. All pixels below the T_{low} percentile gradient magnitude are labeled non-edge pixels. Once all the pixels are labeled either edge, probably-edge, or not-edge, a recursive routine is called for each pixel labeled edge. If a neighbor of the edge pixel is labeled probably-edge and was not suppressed in the NMS image, then it is changed to the label edge and the recursive routine is called again for the new edge pixel. In a sense, a flood-fill is performed starting at all the edge pixels and filling into all the neighboring non-suppressed probably-edge pixels.

The effect of the resolution change is dramatic because T_{high} is set to 1. Only the few pixels with the largest measured gradient magnitude are set to edge pixels. In this example, both the 203 and 204 pixel image have only one pixel labeled edge. The arrows at the perimeter of Figures 8 and 9 indicate the location of the white edge pixel, the light grey pixels are the NMS non-suppressed pixels, the dark grey pixels are probably edge pixels and the black pixels are not-edge pixel. Starting at the edge pixel and following all the probably-edge non-suppressed pixels (light grey) leads to a very different set of edges because the single 204 edge pixel is connected to most of the edge pixels in the image while the single 203 edge pixel is not. The critical missing pixels in Figure 8 can be identified by looking carefully at the vertices where the edges meet. In [6], Rothwell et al noted this particular sensitivity of Canny's gradient images at vertices. This sensitivity is seen in the NMS images in Figures 8 and 9. At each of these locations a few missing pixels create a gap. This sensitivity of the Canny edge detector can have a dramatic effect on the ROC curve and the relative rankings.

Setting T_{high} to 1 is unusual when using the Canny edge detector. The unusual setting is due to the way the framework partitions and searches the parameter space. The extreme settings for every parameter are always tested because the initial 4x4 partitioning always includes the minimum and maximum values, as described earlier. One might be tempted to restrict the range of parameters sampled to prevent such an unusual parameter setting, but this would exclude the relatively accurate result shown in Figure 7 and introduce an arbitrary and poorly understood bias in the evaluation process.

This example only explains why one point in the data used to construct the ROC curve for Canny moved during one particular resolution change. This

example does not explain all the inconsistencies for Canny and it does not explain any of the inconsistencies we observed for either the Rothwell or Susan edge detectors. We provided this particularly dramatic example only to highlight the types of sensitivities an edge detector may have that the other edge detectors do not share. These different sensitivities create inconsistent scores.

4 Conclusions

Our results indicate the ROC curve evaluation framework rankings of edge detectors is dependent on the resolution of the image. Other properties of the image such as aperture setting and lighting provided similar, but less dramatic results. As a result, depending on the contents of the scene, the rank of an edge detector can be selected by changing these properties. In a sense, any of the edge detectors can be called the best by simply changing properties of the scene. Given this sensitivity, can the ROC curve evaluation framework be used to evaluate edge detectors? Yes, but only if a statistically representative set of images that span the expected variability of the task can be collected and used to provide an average ROC curve evaluation. Then, on average, the evaluation would predict the best edge detector. Since a large number of images are required to represent each property of the scene, collecting the images and corresponding ground truth can only practically be accomplished using synthetic imagery.

5 Future Work

This research indicates the ROC curve edge detector evaluation framework is inconsistent, but our conclusions are based entirely on synthetic imagery. To ensure these results are not biased by the differences between synthetic and real imagery, we intend to reproduce these results with real images. A set of simple images intended to mimic the previous synthetic images of blocks will be used with manually defined two-value ground truth. We believe the same inconsistencies will be observed simply by acquiring images of the blocks at different resolutions while holding the other properties of the image constant. This future work should eliminate the unlikely possibility that the inconsistent results from the evaluation framework were caused purely by the differences between synthetic and real imagery.

Acknowledgments

We thank Kevin Bowyer and his research group at the University of South Florida who provided the edge detector implementations and patiently answered our questions.

This research was sponsored in part by the United States Air Force. The opinions and conclusions in this paper are those of the author and are not intended to represent the official position of the DOD, USAF, or any other government agency.

References

[1] Simon Baker and Shree K. Nayar, "Global Measures of Coherence for Edge Detector Evaluation", Proceedings IEEE CVPR, pp. 373-379, 1999.

[2] Kevin Bowyer, Christine Kranenburg, and Sean Dougherty, "Edge Detector Evaluation Using Empirical ROC Curves", Proceedings IEEE CVPR, pp 354-359, 1999.

[3] John Canny, "A computational approach to edge detection", PAMI, 679-698, 1986

[4] Sean Dougherty and Kevin W. Bowyer, "Objective Evaluation of Edge Detectors Using a Formally Defined Framework" in *Empirical Evaluation Techniques in Computer Vision*, IEEE Computer Society, pp 211-234, 1998.

[5] L. Kitchen and A. Rosenfeld, "Edge evaluation using local edge coherence", pp 597-605, IEEE Trans. SMC 11 (9), 1981.

[6] Charlie Rothwell et al., Driving Vision by Topology, *International Symposium on Computer Vision*, 395-400, 1995.

[7] Min C. Shin, Dmitry Goldgof, and Kevin W. Bowyer, "An Objective Comparison Methodology of Edge Detection Algorithms Using a Structure from Motion Task," in *Empirical Evaluation Techniques in Computer Vision*, IEEE Computer Society, pp 235-254, 1998.

[8] Min C. Shin, Dmitry Goldgof, and Kevin W. Bowyer, "Comparison of Edge Detectors Using an Object Recognition Task", Proceedings of the IEEE CVPR Conference, pp 360-365, 1999.

[9] S. M. Smith and J. M. Brady, "SUSAN - A New Approach to Low Level Image Processing", Technical Report TR95SMS1c, Oxford University, 1995.

[10] Mason Woo, Jackie Neider, and Tom Davis, *OpenGL Programming Guide*, pp 247-250, 2nd Edition, Addison-Wesley, 1997.

[11] Q. Zhu, "Efficient evaluations of edge connectivity and width uniformity", *Image and Vision Computing* 14, pp 21-34, 1996.