Computer Vision and Image Understanding **84,** 160–178 (2001)
doi:10.1006/cviu.2001.0932, available online at http://www.idealibrary.com on IDEAL®

Comparison of Edge Detector Performance through Use in an Object Recognition Task

Min C. Shin

Department of Computer Science, University of North Carolina at Charlotte, 201 University City Blvd., Charlotte, North Carolina 28223
E-mail: mcshin@uncc.edu

Dmitry B. Goldgof

Department of Computer Science and Engineering, University of South Florida, Tampa, Florida 33620
E-mail: goldgof@csee.usf.edu

and

Kevin W. Bowyer

Department of Computer Science and Engineering, University of Notre Dame, 384 Fitzpatrick Hall, Notre Dame, Indiana 46556
E-mail: kwb@cse.nd.edu

Received September 15, 1999; accepted August 8, 2001

This paper presents an empirical evaluation methodology for edge detectors. Edge detector performance is measured using a particular edge-based object recognition algorithm as a "higher-level" task. A detector's performance is ranked according to the object recognition performance that it generates. We have used a challenging train and test dataset containing 110 images of jeep-like images. Six edge detectors are compared and results suggest that (1) the SUSAN edge detector performs best and (2) the ranking of various edge detectors is different from that found in other evaluations. © 2001 Elsevier Science (USA)

1. INTRODUCTION

The computer vision community has become well aware of the need for sound empirical evaluation methodologies [6, 9, 15]. However, it remains to be seen what sorts of methodologies will be developed and adopted for general use in the research community.

1077-3142/01 $35.00
© 2001 Elsevier Science (USA)
All rights reserved.

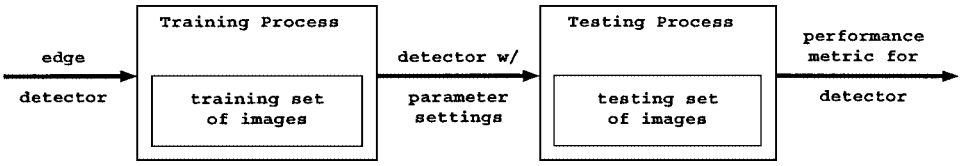

FIG. 1. Overview of "black-box" task-based evaluation.

Important questions to consider in connection with any methodology include: (1) What does the methodology actually measure and evaluate? (2) How much effort is required to apply the methodology? and (3) How well does the result predict performance in some domain of application? We present a methodology for evaluating edge detectors in the context of object recognition. The basic flow of the methodology is depicted in Fig. 1.

The methodology that we present is task-based. It evaluates edge detector performance by the performance obtained on an object recognition task using a particular recognition algorithm [8]. It is possible to evaluate detectors based on pixel-level ground truth [4]. However, edge detection typically is not an end result in itself, but an early step in a sequence of processing. It is not yet clear whether or how pixel-level evaluations might predict the performance on higher-level tasks.

The effort required to apply a methodology can be considered in terms of both operator skill and compute time. The methodology we present is fully automated, using a training stage that selects parameter values for the detector. Thus, operator skill is not a factor in the results. The compute time required to apply the methodology to a new detector is primarily a function of the size of the image dataset. The image content also has some effect, but this will be relatively small for normal images. Applying the methodology to a new edge detector using a current high-end workstation should require only a few days of compute time. The methodology uses a set of 110 intensity images of jeep-like vehicles observed in natural settings. The images were taken with a Kodak DC 25 digital camera. Figure 6 shows a sample of the 100 images, illustrating the variation in content, lighting, background detail, and other factors.

The question of how well the results of the methodology predict performance in some domain of application is harder to answer. The experiments reported here use a particular object recognition algorithm and focus on recognizing a particular type of object. It is impossible to prove by empirical means that the results would not change if a different recognition algorithm were used or if the focus was on recognition of a very different type of object. However, the methodology is automated, the dataset contains a substantial number of images, and the detectors show major differences in an objective performance metric. Thus the methodology does reliably identify performance differences between detectors that are important at least for this style of object recognition algorithm.

The software and image dataset for our comparison methodology are publicly available on the web at http://figment.csee.usf.edu/edge/. The methodology should be readily applicable to the evaluation of detectors other than the ones studied here. It would also be quite easy to substitute other image datasets that focus on different types of objects. It should also be possible to substitute other object recognition algorithms, to investigate whether edge detector ranking changes.

2. EXPERIMENTAL METHODS

2.1. Object Recognition Algorithm

We selected the object recognition algorithm of Huttenlocher and Rucklidge [8] for use in our experiments. It is appropriate for 2-D object recognition or view-based 3-D recognition. This algorithm was selected for several reasons. First, the algorithm directly uses edge maps as its input. There is no requirement to process the edge map to obtain line segments or higher level features, and so recognition performance directly depends on the quality of the edge map. Second, the algorithm is computationally feasible. With 246 $\times$ 186 images, the algorithm converged into a solution (whether an object was detected or not) within 3 min on a Sun Ultra Sparc 1. Third, the algorithm is able to handle relatively complex models and scenes. Lastly, the working implementation was publicly available.

The algorithm uses binary edge images of a *model* image and a *scene* image (refer to Fig. 2). The model image contains an instance of the object to be searched for, without any background. The scene images contains instance(s) of the object to be recognized, with whatever background naturally occurs. This algorithm uses *Hausdorff distance* to measure the similarity between two binary images. Given two sets of points (edges),

$$A = \{a_1, a_2, \ldots, a_m\}, \quad B = \{b_1, b_2, \ldots, b_n\},$$

the Hausdorff distance of sets A and B is measured as

$$H(A, B) = \max(h(A, B), h(B, A))$$

where

$$h(A, B) = \max_{a \in A} \min_{b \in B} \|a - b\|,$$

and $\|a - b\|$ is the Euclidean distance between a and b. The directed Hausdorff distance from A to B ($h(A, B)$) measures the distance to the nearest point in B for each point in A. Then, the largest such distance for any a is $h(A, B)$. If $h(A, B) = d$, any point in A is within the distance of d to some point in B. Note that points are not paired in one-to-one fashion between sets; more than one point in A could have the same b with the closest distance and vice versa.

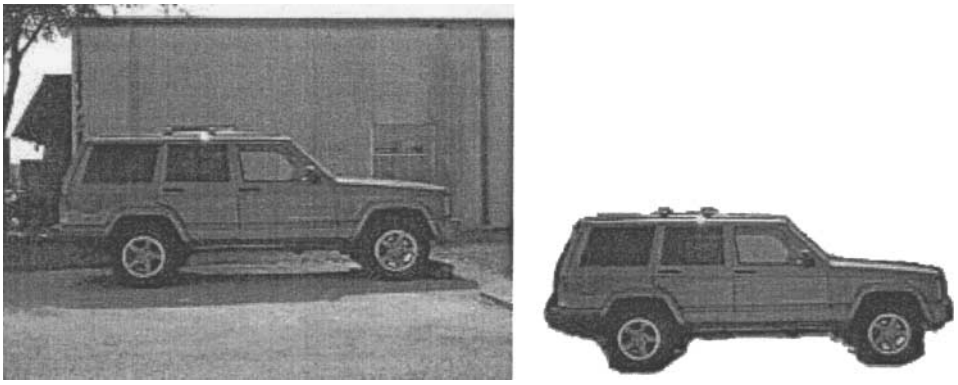

FIG. 2. Scene image vs model image.

The algorithm is able to locate instances of the model with 2D translation and scaling in the scene. The algorithm has the following parameters:

1. Forward Distance (*FD*): The maximum Hausdorff distance from model image edges to scene image edges.
2. Forward Fraction (*FF*): The minimum percentage of pixels of model image edges within *FD*.
3. Reverse Distance (*RD*): The maximum Hausdorff distance from scene image edges to model image edges.
4. Reverse Fraction (*RF*): The minimum percentage of pixels of scene image edges within *RD*.
5. Scaling Range in x and y directions ((SX_{start}, SX_{end}) and (SY_{start}, SY_{end})): The scaling factor is the ratio between the model and the scene in the x and y directions.
6. Skew Factor (*skew*): The maximum ratio between SX and SY and vice versa.

We have fixed these parameters at the following values after many trials: $FD = 2.0$ pixels, $FF = 70\%$, $RD = 2.0$ pixels, $RF = 50\%$, and $skew = 1.001$. Since an instance of the model in the scene image is usually scaled the same in X and Y, the *skew* is set to be near 1.0. The dataset contains 10 model images (Fig. 3) and the scaling range is fixed for each model (refer to Table 1). This implies that for a given model and the range of object to search for, the scaling range can be determined.

To make the task challenging and relevant, it was decided to try to recognize specific models of jeeps in complex scenes with varying background. The algorithm parameters are set to report the one result with the highest FF above a threshold level. Thus, there is zero or one recognition result per image. All images used in our experiments contained either zero or one instance of the target object. The location of a detected object (refer to Fig. 4) is defined by

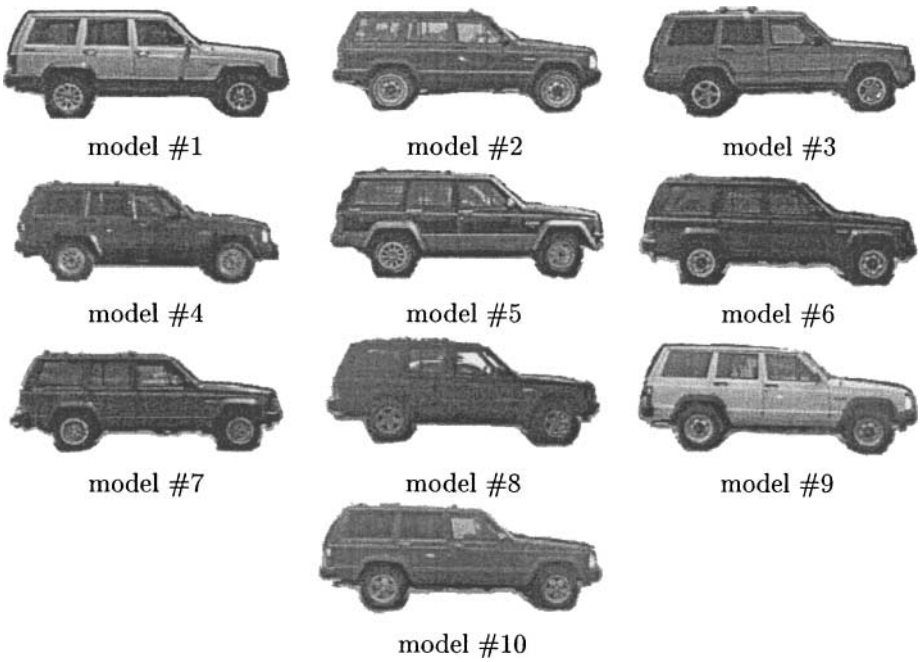

FIG. 3. Model images.

TABLE 1
Scaling Range of Each Model

Model	SX_{start}	SX_{end}	SY_{start}	SY_{end}
1	0.78	1.6	0.78	1.6
2	0.74	1.5	0.74	1.5
3	0.82	1.7	0.82	1.7
4	0.69	1.4	0.69	1.4
5	0.80	1.6	0.80	1.6
6	0.78	1.6	0.78	1.6
7	0.73	1.5	0.73	1.5
8	0.72	1.5	0.72	1.5
9	0.75	1.6	0.75	1.6
10	0.71	1.5	0.71	1.5

1. (T_x, T_y): translation from top-left corner of the image to reach the detected object, and
2. (SX, SY): scaling of model image to the detected object where $SX = width_{model}/width_{scene}$ and $SY = height_{model}/height_{scene}$.

2.2. Edge Detection Algorithms

We tested six edge detectors (refer to Table 2). The robust Anisotropic detector uses a median absolute deviation statistic to smooth the image and selects the outliers of the robust estimation framework as edges [2]. It is used with one parameter, S, in the range (0.0–10.0), which is a scale factor for the σ_e parameter computed by the algorithm. The Bergholm edge detector applies a concept of "edge focusing" to find significant edges [1]. The image is smoothed using a coarse resolution (high smoothing) and the edges exceeding a contrast threshold are identified. Then the locations of these edges are refined by tracking them to a finer resolution. It has three parameters: S_{start} (0.5–5.0), S_{end} (0.5–5.0), T (5.0–60.0). The Canny detector uses Gaussian smoothing, nonmaxima suppression, and hysteresis thresholding [3]. It has three parameters: *sigma* (0.5–5.0), *low* (0.0–1.0), *high* (0.0–1.0). The Heitger detector suppresses the filter responses that do not correspond to a given image curve while it enhances those that do correspond [10]. It has two parameters: *sigma* (0.5–5.0) and T (0.0–50.0). The Sobel detector uses a simple filter for computing a

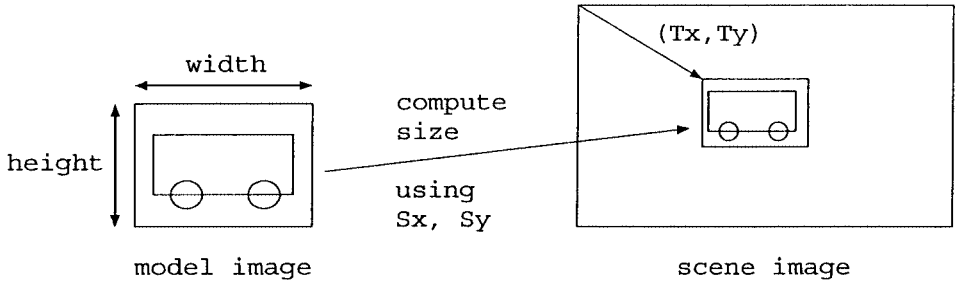

FIG. 4. Location of detected object.

TABLE 2
Parameter Range of Edge Detectors

Edge detector	Parameters		
Anisotropic (98, [2])	S (0.0–10.0)		
Bergholm (87, [1])	S_{start} (0.5–5.0)	S_{end} (0.5–5.0)	T (5.0–60.0)
Canny (86, [3])	*sigma* (0.5–5.0)	*low* (0.0–1.0)	*high* (0.0–1.0)
Heitger (95, [10])	*sigma* (0.5–5.0)	T (0.0–50.0)	
Sobel (70, [14])	*low* (0.0–1.0)	*high* (0.0–1.0)	
SUSAN (97, [13])	T (0.0–50.0)		

gradient and hysteresis for linking edges [14]. It has two parameters: *low* (0.0–1.0) and *high* (0.0–1.0). The SUSAN detector calculates the edge strength using the number of edges with similar brightness within circular masks. The edge direction is calculated using moment calculation and nonmaxima suppression is used to thin edges [13]. It has one parameters: T (0.0–50.0). The robust anisotropic detector was implemented at USF. The implementation of the Bergholm detector was adapted from the software package distributed by the computer vision group at KTH at Stockholm. The implementation of the Canny detector was written at USF. The implementation of the Heitger detector was originally obtained from the web pages at ETH in Zurich. The implementation of the Sobel detector was written at USF. The implementation of the SUSAN detector was obtained from the Web pages at the University of Oxford. The edge detector implementations used in the experiments reported in this paper are the same as used in [4]. The USF web page mentioned earlier should have either the implementation of each detector or a pointer to where the original author's implementation can be obtained.

3. FRAMEWORK

The comparison methodology involves three steps: edges detection, object recognition, and evaluation. The overview of this framework is shown in Fig. 5. Given the model and scene images, an edge detector produces an edge map for each. Using those edge maps, the recognition algorithm outputs (1) whether the target exists in the scene or not, (2) if so, the location and the size of the target. Using a predefined ground truth, the result is evaluated as TP, TN, FP, or FN.

3.1. Dataset

The dataset includes images of nine different types of jeep-like vehicle (refer to Fig. 6). The images were taken with a Kodak DC-25 digital camera in the high-resolution mode (493×373 pixels). The images were acquired as follows. Side views (driver or passenger) are taken. One object appears in approximately the center of the image. The object is horizontally upright. No occlusion (not even the grass covering the wheels) is allowed. Objects could vary in details such as wheel covers, roof racks, and customized add-ons. A total of 110 such images were taken. The vehicle model in each image was noted. Nine different models of vehicle appear in the dataset. Sixty of the images contain an instance of a Jeep Cherokee vehicle. Ten of these were selected for use as different models. This allows us to explore the dependence of the algorithm on selecting a "good" model. The

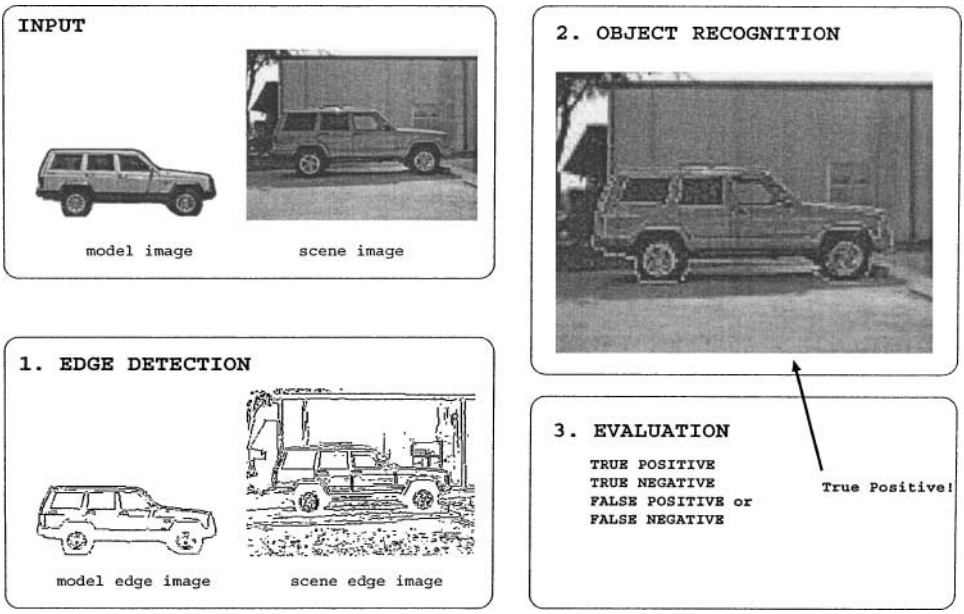

FIG. 5. Overview of the framework. The translated and scaled model edge image is overlaid on the scene image as white pixels in the box "2. Object Recognition."

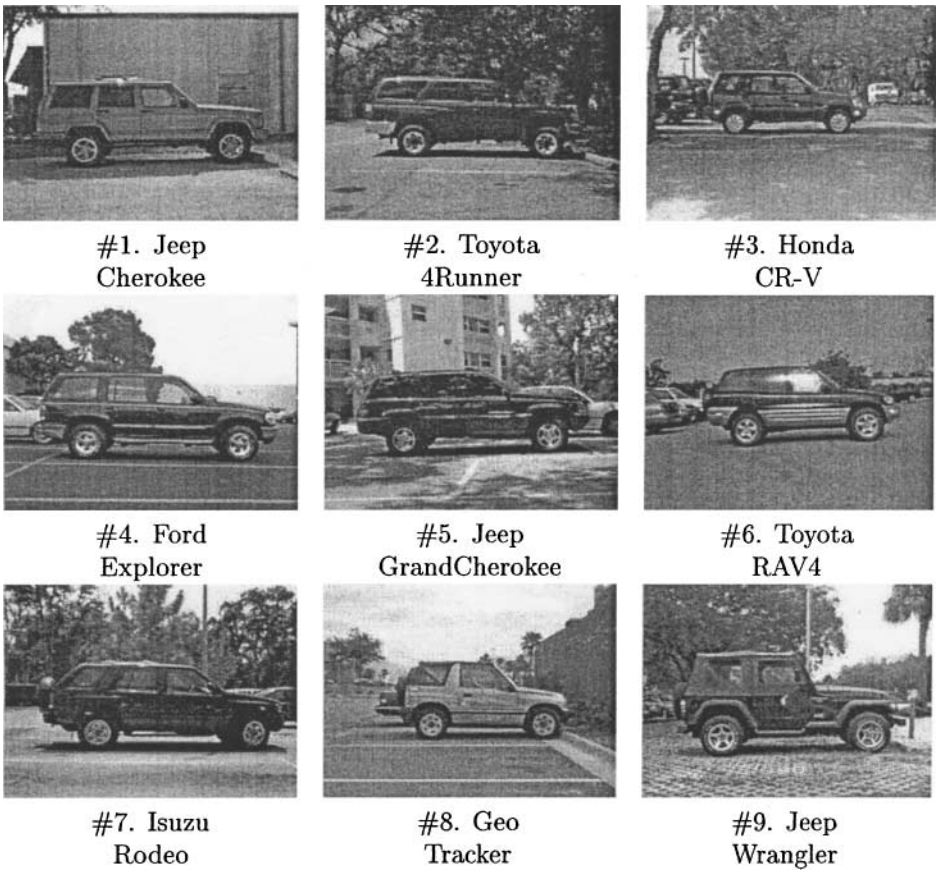

FIG. 6. Examples of the different jeep categories.

remaining 100 images were divided into balanced training and testing sets of 50 images each containing 25 "with-target" (Jeep Cherokee) and 25 "without-target" images.

In order to make the dataset thorough, we attempted to vary the following aspects: (1) The background content was varied by acquiring images at different places. (2) The images were taken with different distance-to-object, so scale will vary. (3) The images were taken from early morning to sunset, so lighting will vary. The images could not be taken at night, since the camera did not provide enough flash lighting to obtain good images. (4) In some instances, the objects were under or beside other objects which added shadows.

Note that (1) obtaining the images of both sides of the jeep was not always possible, and (2) one side of the jeep was usually similar to the one side. So, if the image of one side could not be obtained, we reversed column order to create a mirror image.

It was found that the object recognition algorithm could consume up to 2 days of CPU time on a Sun Ultra Sparc 1 with a 493×373 image. When the image was scaled by half, the compute time was about 3 min. The compute time required clearly scales nonlinearly with image size. Thus, for practical reasons, the original image size was halved by averaging 2×2 blocks, resulting in 246×186 images.

3.2. Model Images

A model image is created by taking one of the "with-target" images and cropping out the background as shown in Fig. 2. We erased any part of the image which did not belong to the model object within several pixels around the boundary of the object. For a given experiment, the scene image that a model image was extracted from is not included. Figure 3 shows 10 models used in this experiment.

3.3. Training and Testing Set

The dataset is organized as 50 train images, 50 test images, and 10 model images. We selected type 1 (Jeep Cherokee) as the "target." The training and testing sets each consist of 50 images $=$ 25 "with-target" $+25$ "without-target." Table 3 explains the distribution of images in the train and testing set. For each model image, we have trained the recognition algorithm on the training set, yielding 10 training performances for each detector (Table 4). For testing, we have selected the best model for each detector during the training process, and tested on the testing set.

3.4. Ground Truth

The ground truth is specified by two values. First, a boolean value represents whether the image contains a "target" or not. Second, the location of the "target" is specified by a

TABLE 3
Train and Test Dataset

	"With-target"	"Without-target"							
Category	1	2	3	4	5	6	7	8	9
Models	10	0	0	0	0	0	0	0	0
Train	25	5	5	5	5	3	2	0	0
Test	25	4	2	5	5	2	2	2	3

Note. The category number corresponds to that in Figure 6.

TABLE 4
Train Performance

	Model									
Edge detector	1	2	3	4	5	6	7	8	9	10
Anisotropic	0.56	0.31	0.50	0.04	0.56	0.31	0.26	0.00	0.39	0.23
Bergholm	0.37	0.15	0.25	0.17	0.37	0.17	0.09	0.05	0.46	0.16
Canny	0.26	0.21	0.07	0.07	0.34	0.29	0.15	0.08	0.17	0.26
Heitger	0.34	0.11	0.16	0.04	0.43	0.17	0.21	0.11	0.42	0.28
Sobel	0.04	0.06	0.06	0.10	0.09	0.06	0.04	0.00	0.08	0.00
SUSAN	0.53	0.47	0.54	0.22	0.55	0.43	0.42	0.08	0.43	0.39

Note. The best performance for each edge detector is indicated in bold.

rectangle that encloses the object. The rectangle is represented using two 2D coordinates: the top-left (LT_{GT}) and bottom-right (RB_{GT}) corners.

3.5. Performance Metrics

3.5.1. Detection Characterization as TP or FP

Since it is possible to have the MD (Machine Detected) object in the scene using (T_x, T_y) and (SX, SY) from the object recognition algorithm, (LT_{MD}, RB_{MD}) is computed. Then, $Area_{overlap}$, the area of the overlap region of two regions is computed (Fig. 7). If $\frac{Area_{overlap}}{Area_{MD}} >= T_{overlap}$ and $\frac{Area_{overlap}}{Area_{GT}} >= T_{overlap}$, then the detection is considered to be a true positive (TP). Otherwise, it is a false positive (FP) (refer to Fig. 8). The $T_{overlap}$ is set to 0.75. When a MD is reported in a "without-target" image, it is also considered to be a FP (refer to Fig. 9).

3.5.2. ROC Curve Analysis

The ROC curve as shown in Fig. 10 consists of a series of performance points. A performance point is computed for each parameter setting of the detector.

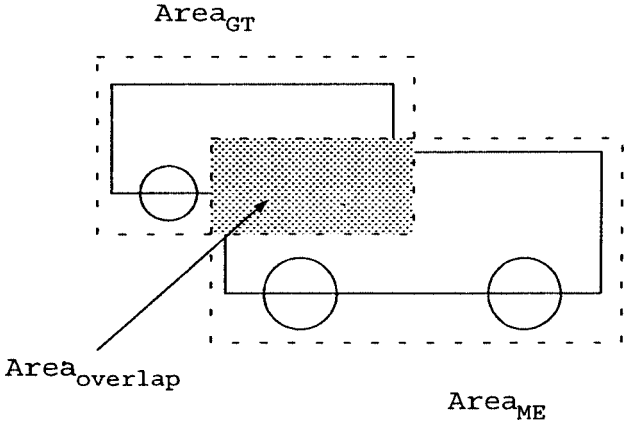

FIG. 7. Overlap of detection.

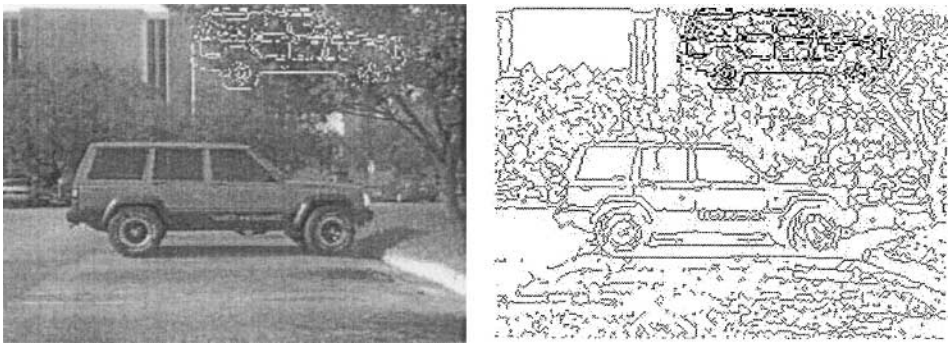

FIG. 8. Example of FP with MD detection at incorrect location in "with-target" image.

FIG. 9. Example of FP with MD detection at correct location in "without-target" image.

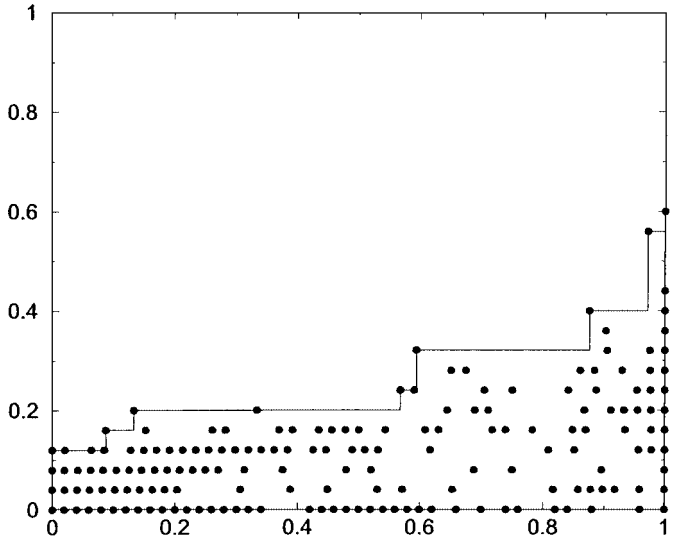

FIG. 10. Example of ROC curve constructed for a set of operating points. (Canny edge detector using Model 1 during training).

For a given set of train or test results, the TP figure is computed as the percent of the 25 with-target images for which there was TP detection. A FP can occur for any of the 25 without-target images and for any of the with-target images which did not result in a TP. The FP figure is calculated as the percent of these images for which a detection was reported. This differs slightly from the standard definition of an ROC curve.

The area of the ROC curve is used to find the performance of an edge detector on a dataset. The points from all parameter settings examined during parameter selection (refer to the next section on Parameter Search) are plotted on the graph. The points that describe the highest TP rate for a given FP rate are connected to generate a stair-step-looking ROC curve (refer to Fig. 10). Effectively, an edge detector parameter setting is discarded if it results in both fewer TPs and more FPs than some other setting. The area under the curve (AUC) is used to describe the performance of an edge detector for a dataset. The ideal AUC would be 1.0 meaning 100% TP recognition with no FP detections.

3.6. Parameter Search

The general sensitivity of edge detector performance to parameter settings has been confirmed by several studies [4, 7, 11]. In this experiment, we have modified the parameter sampling technique used in [4] to adjust the parameter setting objectively and efficiently (refer to Fig. 11).

Each parameter range is evenly divided to create the initial parameter sampling. Then, one parameter's sampling is doubled (the midpoint between consecutive previous sampling points is added). For instance, the initial parameter sampling of $(5 \times 5 \times 5)$ will be expanded to create $(9 \times 5 \times 5)$, $(5 \times 9 \times 5)$, and $(5 \times 5 \times 9)$. The best performing sampling is further expanded if (1) there was at least 10% improvement over the previous best performance, and (2) the maximum performance (1.0 AUC) has not be reached yet.

The parameters of the object recognition algorithm were not trained. With an additional 6 parameters of the object recognition algorithm, training at the same resolution $(5 \times 5 \times \cdots \times 5)$ would require a minimum of 5^6 (15625) times of the current training time taking nearly 342 (8 days/detector $\times$ 15625 $\times$ 1 year/365 days) years to train one edge detector. Therefore, we fixed the parameters at the values described in the Section 2.1.

4. RESULTS

The result analysis is divided into three sections: (1) train, (2) test, and (3) time and parameter settings attempted.

4.1. Train

For all 10 models of the training set, the parameter space of an edge detector is searched using the method described in Section 3.6 to generate an ROC curve. Then, the ROC curves and their AUCs are computed (refer to Table 4 and Fig. 12).

First of all, the Anisotropic detectors performed the best with AUC of 0.56 with Model 1 or 5. The SUSAN detector was only 0.01 away from being with the highest AUC compared to the Anisotropic detector. The highest AUC the Sobel could obtain was 0.10 with the Model 4. The statistical significance of the training performance is shown in Table 5. The "*" is placed whenever the detector in the row significantly out-performed (2/3 or more

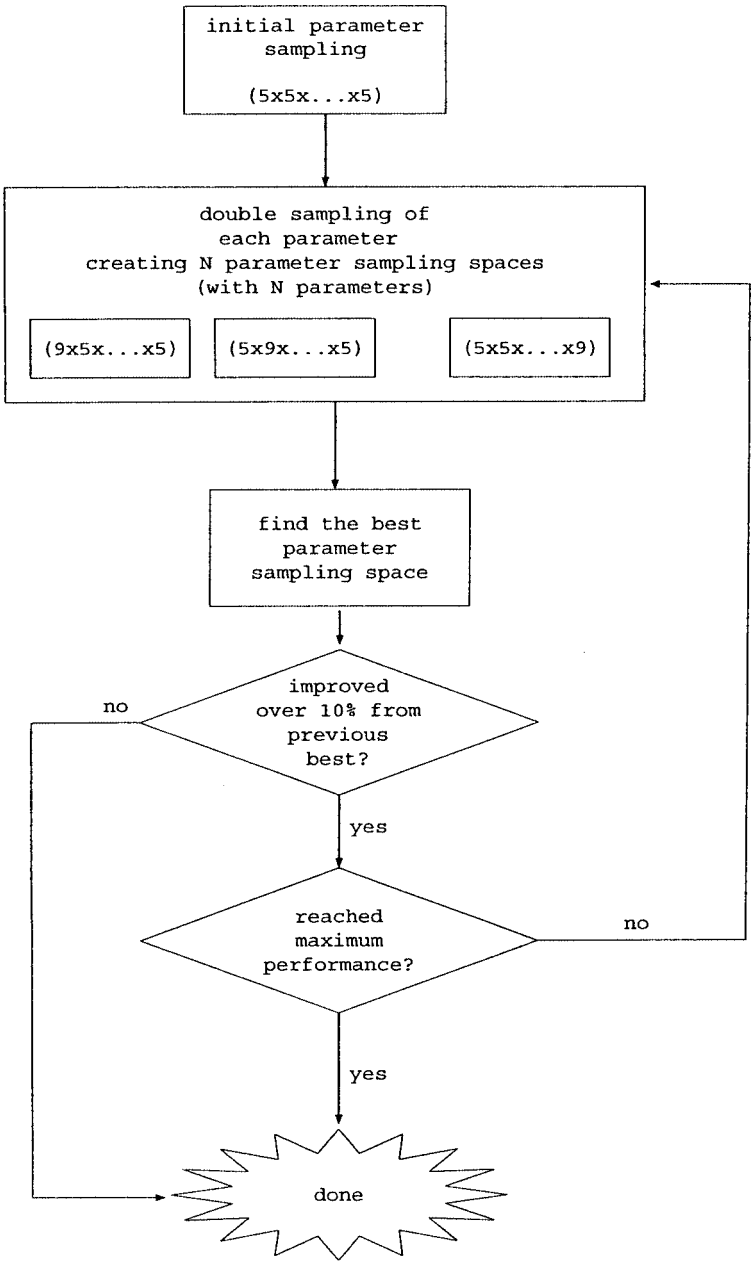

FIG. 11. Parameter searching method.

times) the detector in the column. Note that the SUSAN detector out-performed all other detectors. And the Sobel detector was out-performed by all other detectors.

Second, Model 5 was the best model for four out of six edge detectors. Also, all six edge detectors had Model 4 or 8 as the worst model for training. This result shows that there are specific models that are generally "good" or "bad" across detectors.

Third, surprisingly, the best two edge detectors were "1-parameter" detectors. These detectors were the fastest in the time required to train (since they attempt far fewer parameter settings than others) and the best in the category of trained AUC.

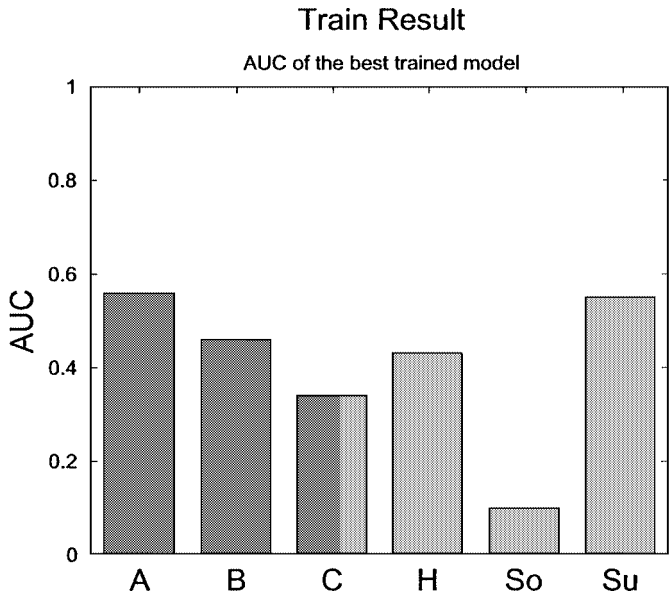

FIG. 12. Train result. A (Anisotropic), B (Bergholm), C (Canny), H (Heitger), So (Sobel), Su (SUSAN) detectors.

4.2. Test

The test results were obtained as follows. The best (highest AUC) target model was selected for each detector. All the parameter settings on this ROC curve were used for the test. If an operating point on the train ROC represented more than one parameter setting, all the parameter settings were used. The selected model and edge detector parameter settings were run on the testing set. The test ROC was constructed, and the test AUC was computed. The Sobel's train performance was poor; its trained AUC was 0.10, which was less than 1/3 of the second worst trained edge detector (Canny). Only with its best performing model (4) was it able to excel over three edge detectors (refer to Table 5). Since the testing with a poorly trained model and the ROC curve would not be meaningful, the testing results of the Sobel are not included.

TABLE 5
Relative Train Performance

	Anisotropic	Bergholm	Canny	Heitger	Sobel	SUSAN
Anisotropic		7*	7*	6	8*	2
Bergholm	3		5	5	10*	1
Canny	3	5		3	9*	0
Heitger	3	4	7*		9*	1
Sobel	1	0	1	1		0
SUSAN	8*	9*	9*	9*	10*	

Note. Each number indicates the number of target models for which the AUC of the edge detector in the row was greater than the AUC of the edge detector in the column. The maximum value is 10. Note that if the detector in the row outperformed the detector in the column 2/3 or more times, the "*" has been placed next to the number.

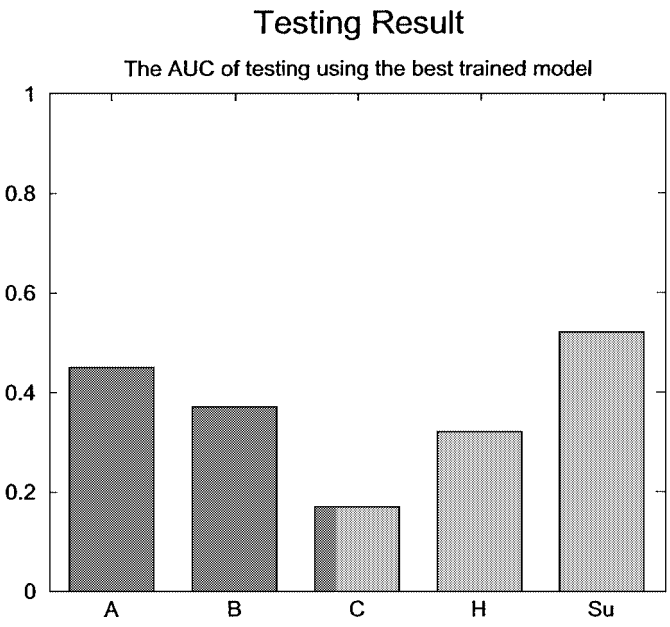

FIG. 13. Test result. A (Anisotropic), B (Bergholm), C (Canny), H (Heitger), Su (SUSAN) detectors. The Sobel is not included since its train performance was so poor that the testing result would not be meaningful.

The SUSAN detector had the highest test AUC, and the Anisotropic detector was second (refer to Fig. 13 and Table 6). This is the reverse of their rank in the training phase, but the AUC differences are very small in the training. The ranking of the rest of the detectors was the same in testing as in training.

4.3. Speed of Edge Detectors & Parameter Searching Complexity

The speed of an edge detector is divided into two numbers: (1) edge detection speed, and (2) edge detection of model + edge detection of scene + object recognition + comparison/evaluation (refer to Table 7). The number is from the average of the time during the training. The Sobel and SUSAN detectors were the fastest edge detectors and they were also the fastest overall. The Anisotropic detector was the slowest edge detector while it also was the slowest overall.

The complexity of the parameter searching/tuning is also studied. Figure 14 shows the average of the number of parameter settings checked during the training of 10 target models. First, as expected, the edge detectors with fewer parameters examined fewer parameter

TABLE 6
Test Performance

Edge detector	Best train model	Test AUC
Anisotropic	1	0.45
Bergholm	9	0.37
Canny	1	0.17
Heitger	5	0.32
SUSAN	5	0.52

TABLE 7
The Speed of Edge Detection and Overall Process

Edge detector	Edge detection	Overall
Anisotropic	6.6 sec	6.2 min
Bergholm	1.2 sec	5.5 min
Canny	0.4 sec	4.2 min
Heitger	2.4 sec	3.8 min
Sobel	0.1 sec	4.0 min
SUSAN	0.1 sec	3.6 min

settings. The SUSAN detector checked fewest (11) while the Canny detector checked the most (3411). These numbers do not necessarily reflect inherent relative difficulty in training. For example, using a larger range of initial values for a parameter might cause the search to take longer to locate the same final parameter settings, and it is problematic to control such factors across detectors.

5. DISCUSSION

5.1. Relationship Between Rating and Edge Maps

This section will attempt to explain how the edge maps change along the ROC curve. In other words, what kind of edge maps scored high on this task?

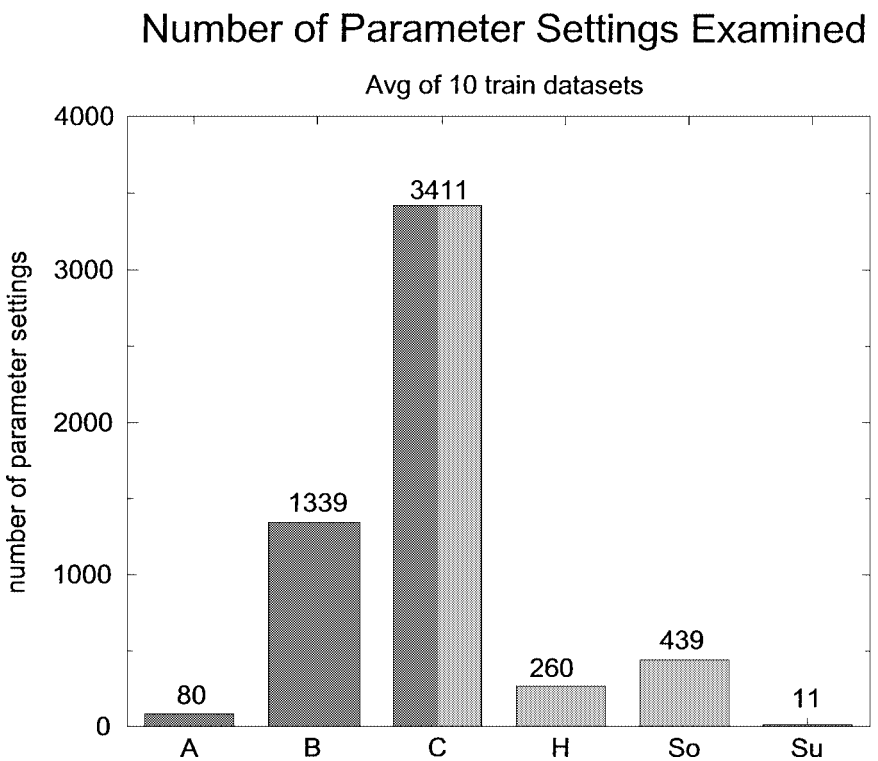

FIG. 14. Parameter searching complexity. A (Anisotropic), B (Bergholm), C (Canny), H (Heitger), So (Sobel), Su (SUSAN) detectors.

We have taken parameter settings (sets of parameter values of an edge detector) at three different locations— (low FP, low TP), (mid TP, and FP), (high TP, high FP)—along the ROC curve from the best trained dataset. The values of low, mid, and high TPs and FPs are all taken relatively for each edge detector. Figure 15 shows the (1) (FP, TP) coordinate on ROC plot, (2) parameter setting, (3) edge maps of the model and the scene images for low, mid, and high (FP, TP). First, all edge maps are very reasonable. In fact, it is difficult for a human being to predict which edge maps would have performed better than others. Second, the edge maps of the low (FP, TP) points tend to have fewer edges than the ones from high

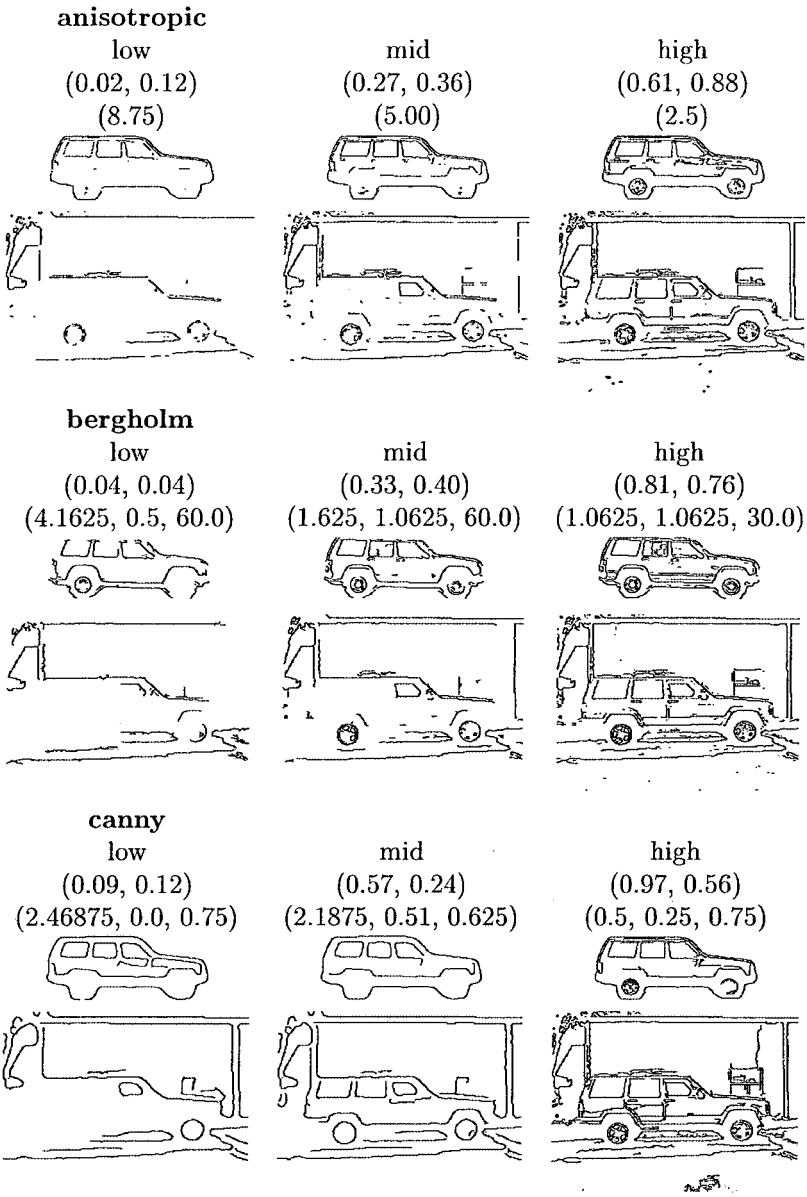

FIG. 15. Edge maps along the ROC curve, part 1. The first number is (FP, TP) location. The second number is the corresponding parameter setting of the edge detector: anisotropic S, bergholm (S_{start}: S_{end}, T), canny (*sigma*, *low*, *high*). Edge maps of the Sobel are not included because the training performance of the Sobel detector was poor.

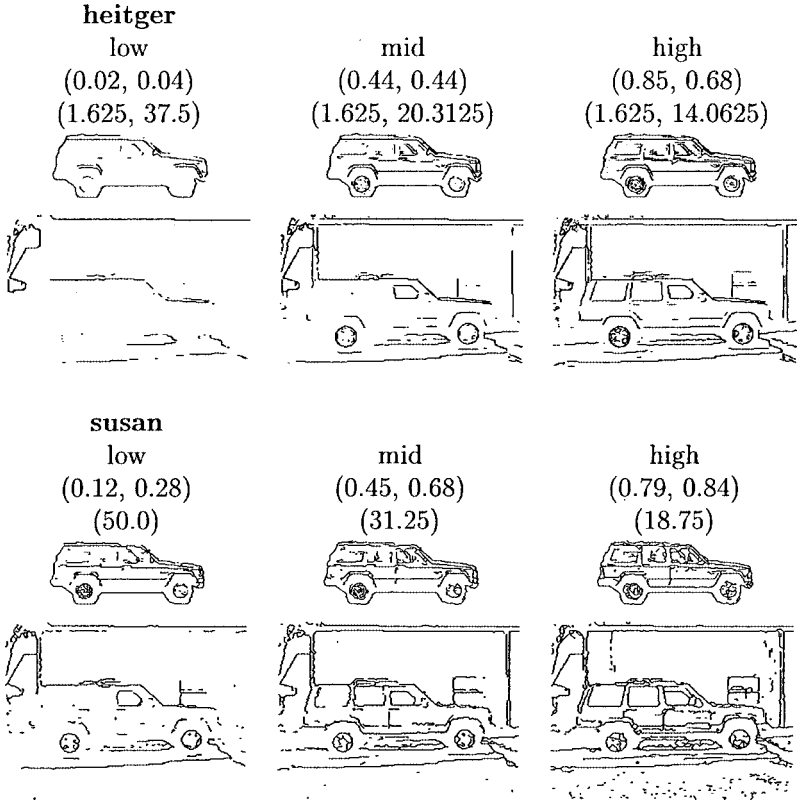

FIG. 15—*Continued*

(FP, TP). Note that if there are too few or too many edges, the recognition performance certainly suffers. Therefore, the object recognition algorithm seems to be utilizing useful additional detail information such as details inside the jeep outline. Note that the edges are incremented as we travel from low, mid, to high (FP, TP) rather than replaced, meaning the edges used in low (FP, TP) are not used in high (FP, TP). More detailed edges can be described by referring to the parameter settings. The sigma (smoothing operator) of the Bergholm detector and the Canny detector have decreased from low and mid to high (FP, TP). Also, thresholding parameters of the Heitger detector and the SUSAN have been decreased to allow more edges.

5.2. Source of FP

This section attempts to examine the source of FPs. As described in Section 3.5.1, there are two cases for FPs: (1) wrong detection (MD detecting a target while GT indicates no target in the scene) and (2) wrong location (MD detecting an object at the wrong location from GT location of the object).

First, there are more wrong detection FPs than wrong location FPs (refer to Table 8). This suggests that the edge maps were more often less detailed. Therefore, there was less difference between “target” and “nontarget” objects. Second, the number of FPs increased as we moved from low to high (FP, TP). Third, the percentage of wrong detection and

TABLE 8
Characterization of FP's along ROC Curve

Edge detector	ROC location	Wrong detect	Wrong location	% of wrong detect	% of wrong location
Anisotropic	low	1	1	50	50
	mid	5	3	63	38
	high	17	4	81	19
Bergholm	low	2	0	100	0
	mid	8	4	67	33
	high	22	10	69	31
Canny	low	4	1	80	20
	mid	13	6	68	32
	high	25	18	58	42
Heitger	low	0	0	n/a	n/a
	mid	16	8	67	33
	high	24	14	63	37
Sobel	low	6	4	60	40
	mid	5	5	50	50
	high	12	5	71	29
SUSAN	low	1	1	50	50
	mid	13	7	65	35
	high	23	8	74	26
mean	low	2	1	68	32
	mid	10	6	63	37
	high	21	10	69	31
	total	11	6	67	33

wrong location FPs did not show any pattern. There were rougly 60% and 40% in all three locations of (FP, TP).

6. CONCLUSION

We have presented results of an experimental evaluation of edge detectors using the task of object recognition. With 110 images and 6 edge detectors, the computing time for train and testing were nearly 48 continuous *days* of computing time for the complete experiment on a Sun Ultra Sparc 1. On average, it requires about 8 CPU days to evaluate a detector. We feel that this is not an impractical amount of effort. Results varied widely between detectors and overall detection performance was not excellent, indicating that the dataset was not too easy or too difficult for the edge detectors.

We found that the Anisotropic detector gave the best train result with AUC of 0.56, while the SUSAN detector gave nearly the best train performance (0.55) and significantly outperformed all detectors in testing. They are both one-parameter edge detectors.

Surprisingly, the Canny detector, which performed nearly the best in three other edge detector evaluation studies [4, 7, 12], ranked as fifth of six in this study for both training and testing. It will certainly be interesting to see (1) why the performance ranking differs among the evaluation methodologies, and (2) what characteristics of edges each of the evaluation methodologies prefers. The Structure From Motion (SFM)-based [11] study suggests that more and longer lines being present in a longer sequence is important for SFM. It is another

research topic to investigate the source of the errors. In fact, one goal of the comparison research should be exploring the error source. According to our analysis, the error seems to come from (1) the lack of details of the objects and (2) the False Positive cases of detecting an object when it is not present in the scene image.

ACKNOWLEDGMENTS

An earlier version of this work was presented at the Computer Vision and Pattern Recognition 99 conference [12]. This work was supported by NSF Grants EIA-9729904 and IIS-9731821.

REFERENCES

1. F. Bergholm, Edge focusing, *IEEE Trans. Pattern Anal. Mach. Intell.* **PAMI-9**(6), 1987, 726–741.
2. M. Black, G. Sapiro, D. Marimont, and D. Heeger, Robust anisotropic diffusion, *IEEE Trans. Image Process.* **7**(3), 1998.
3. J. Canny, A computational approach to edge detection, *IEEE Trans. Pattern Anal. Mach. Intell.* **8**(6), 1986, 679–698.
4. K. W. Bowyer, Christine Kranenburg, and Sean Dougherty, Edge detector evaluation using empirical ROC curves, *Comput. Vision Image Understanding*, to appear.
5. K. I. Chang, K. W. Bowyer, and M. Sivagurunath, Evaluation of texture segmentation algorithms, in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 1999, pp. 294–299.
6. H. Christensen and W. Foerstner, Guest Eds., Special Issue on Performance Evaluation, *Machine Vision and Applications* **9**(5), 1997.
7. M. Heath, S. Sarkar, T. Sanocki, and K. W. Bowyer, A robust visual method for assessing the relative performance of edge detection algorithms, *IEEE Trans. Pattern Anal. Mach. Intell.* **19**(12), 1997, 1338–1359.
8. D. Huttenlocher and W. Rucklidge, A multi-resolution technique for comparing images using the Hausdorff distance, *IEEE Confernce on CVPR 1993*, pp. 705–706.
9. P. J. Phillips and K. W. Bowyer, Special issue on emperical evaluation of computer vision algorithms, *IEEE Trans. Pattern Anal. Mach. Intell.* **21**(4), 1999.
10. L. Rosenthaler, F. Heitger, O. Kubler, and R. von der Heydt, Detection of general edges and keypoints, in *ECCV'92* (G. Sandini, Ed.), pp. 78–86, Springer-Verlag, Berlin/New York, 1992.
11. M. Shin, D. Goldgof, and K. W. Bowyer, An objective comparison methodology of edge detection algorithms for structure from motion task, in *Empirical Evaluation Techniques in Computer Vision* (K. W. Bowyer and P. J. Phillips, Eds.), pp. 235–254, IEEE Comput. Soc. Press, 1998.
12. M. Shin, D. Goldgof, and K. W. Bowyer, Comparison of edge detectors using an object recognition task, in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 1999, pp. 360–365.
13. S. M. Smith and J. M. Brady. SUSAN—A new approach to low level image processing. *Int. J. Comput. Vision* **23**(1), 1997, 45–78.
14. I. E. Sobel, *Camera Models and Machine Perception*, pp. 277–284, Stanford University, 1970.
15. M. A. Viergever, H. S. Stiehl, R. Klette, and K. Vincken, *Performance Characterization and Evaluation of Computer Vision Algorithms*, Kluwer Academic, Dordrecht, 2000.