Algorithm Performance Contest

Selim Aksoy, Ming Ye, Michael L. Schauf, Mingzhou Song, Yalin Wang, Robert M. Haralick
Intelligent Systems Laboratory, University of Washington, Seattle, WA, 98195-2500, USA

Jim R. Parker, Juraj Pivovarov, Dominik Royko
University of Calgary, Dept. of Computer Science, Calgary, Canada

Changming Sun
CSIRO Mathematical and Information Sciences, Locked Bag 17, North Ryde, NSW 1670, Australia

Gunnar Farnebäck
Computer Vision Lab., Dept. of Electrical Eng., Linköping University, SE-581 83 Linköping, Sweden

Abstract

This contest involved the running and evaluation of computer vision and pattern recognition techniques on different data sets with known groundtruth. The contest included three areas; binary shape recognition, symbol recognition and image flow estimation. A package was made available for each area. Each package contained either real images with manual groundtruth or programs to generate data sets of ideal as well as noisy images with known groundtruth. They also contained programs to evaluate the results of an algorithm according to the given groundtruth. These evaluation criteria included the generation of confusion matrices, computation of the misdetection and false alarm rates and other performance measures suitable for the problems. This paper summarizes the data generation for each area and experimental results for a total of six participating algorithms.

1. Introduction

The contest home page and the packages are available at http://isl.ee.washington.edu/IAPR/ICPR00. All the software was written in C and developed in the Unix environment. The participants were allowed to use any set of parameters to generate test images for algorithm development. A specific set of parameters were supplied to generate data for the final experiments. The experiments were run by the participants themselves and the final results that were the output of the evaluation algorithms were submitted to the contest organizers.

The following sections describe the data generation and experimental results for the binary shape recognition, symbol recognition and image flow estimation areas.

2. Binary Shape Recognition

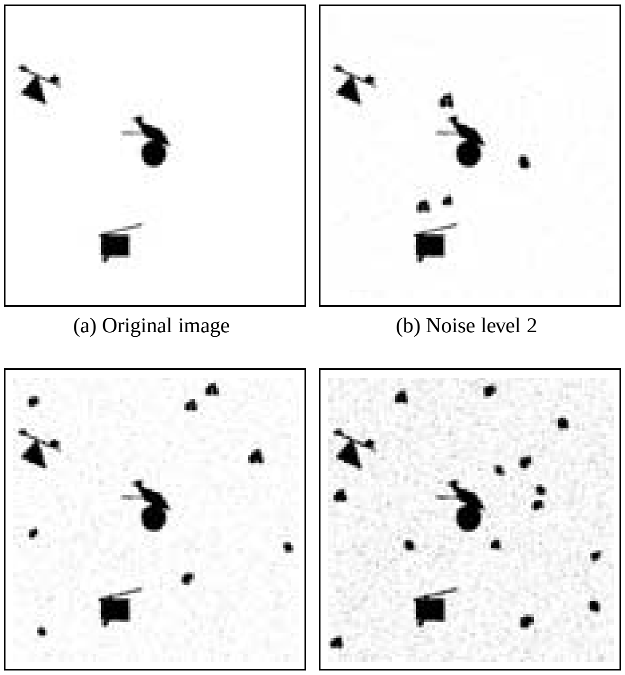
2.1. Data Generation

This package was prepared by Michael L. Schauf and Selim Aksoy. It was intended to provide a test data set with known groundtruth to evaluate binary shape recognition algorithms. It included code for generation of primitives and shape prototypes as the groundtruth model set, and perturbed images containing translated and scaled prototypes as the test data set.

The program started with the generation of shape models. A shape model was composed of a set of primitives. Each primitive was mildly constrained so that its digital image bore a reasonable resemblance to the ideal continuous primitive. The primitives for this data set were lines, circles, triangles, sectors, and quadrilaterals. Each primitive had some restriction on its free parameters in order to retain its general properties. The different primitives were randomly selected, generated and combined to form the different shape models. Each shape model was constrained so that each primitive slightly overlapped another.

Once the shape models were generated, they were randomly selected to be placed in an image. Each selected shape model was placed in the image at a random location with a random scale with the only constraint that its bounding box did not overlap with any other shape model's bounding box that was already in the image. Since we knew the locations and scales of the models in all images, we had the complete groundtruth.

Testing the robustness of recognition algorithms also requires the design of images containing varying levels of noise. For the addition of noise, the Document Degradation Model by Kanungo [4] was used. This model added pepper noise in such a way that pixels around the borders of the shape models had a higher probability of switching to opposite values than those pixels farther away from the border. Additional noise was added by generating non-relevant shape models and scaling them smaller than the smallest scale of the relevant models. Some example noisy images are given in Figure 1. Model and image sizes, overlap between primitives in a model, range for the number of primitives in a model, range for the number of models in an image, range for the scales of models in an image, and noise level were some of the parameters that were allowed to be changed in the code.


(a) Original image (b) Noise level 2

(c) Noise level 3 (d) Noise level 5

Figure 1: Example shape images.

The package also included a program for performance evaluation on a database of images with their groundtruth and recognition algorithm results. The output of the program was a confusion matrix and a success score for the recognition algorithm. The program read the groundtruth information for each image and compared it with the recognition results. The success value was a linear combination of the correct detection rate, incorrect detection rate, misdetection rate, false alarm rate, and the accuracy of the detected location and scale.

2.2. Experiments

2.2.1. Schauf, Aksoy and Haralick Algorithm

The only result available for this package is the results of the binary shape recognition algorithm by Michael L. Schauf, Selim Aksoy and Robert M. Haralick from the Intelligent Systems Laboratory, University of Washington, USA. The algorithm [6] introduces a size invariant method to recognize complex two-dimensional shapes using multiple generalized recursive erosion transforms. The method accomplishes the same kind of recognition that templates of each shape at multiple scales would do, but the method takes constant time per pixel regardless of the scale of the prototype. The method workes on noisy images without requiring noise removal as preprocessing.

Experiments were run on 100 noisy images having a total of 254 randomly translated and scaled models plus a number of extraneous small shapes that might appear like a model. The algorithm had 5 misdetections and 13 false alarms. The confusion matrix is given in Table 1.

3. Symbol Recognition

3.1. Data Generation

This package was prepared by Ming Ye, Mingzhou Song, Yalin Wang and Selim Aksoy. It was intended to provide a test data set with known groundtruth to evaluate binary symbol recognition algorithms. The symbol library consisted of electrical symbols as the model set and noisy versions of randomly translated and scaled symbols as the test data set. The symbol library contained 25 electrical symbols where each ideal symbol image was 512x512 pixels. The symbols were all line drawings with a 30 pixel line width.

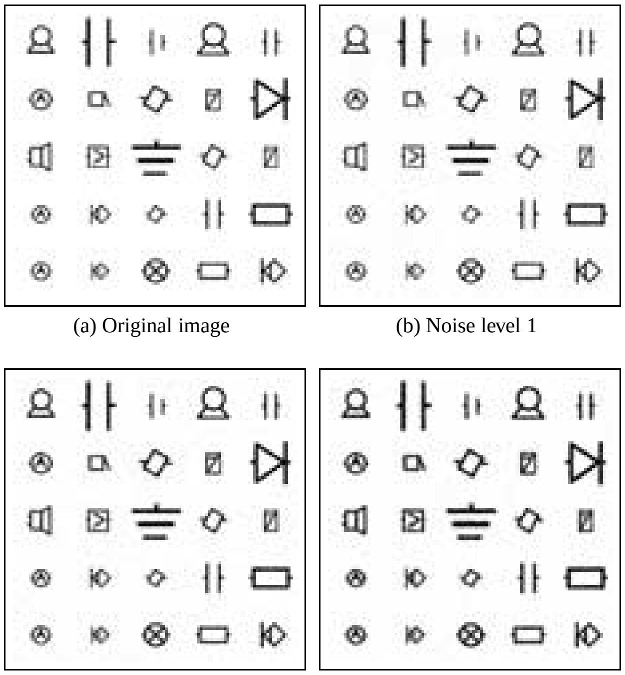
An instantiation of a symbol was obtained by scaling the symbol image down to a certain size between 40 and 160 pixels. Because of the high resolution of the library symbols, the observed symbol had a thick enough line width so that it was very unlikely that broken lines existed after denoising. The observed symbols might also have a small rotation angle.

In order to generate test data, an empty image was first partitioned into square patches, each of which had the same size. A randomly selected symbol was randomly scaled and put into each of the patches, with the centroid of the symbol lying at the patch center. No symbol occluded others. Since we knew the locations and scales of the symbols in all images, we had the complete groundtruth. It is worth mentioning that such symbol arrangement was just for synthesis convenience, but not for the ease of recognition.

There were three major types of noise perturbing the observations: quantization error, replacement noise, and salt-and-pepper noise. Quantization error came from scaling the library symbol, because the scaling factor could be any real value while the pixel positions had to be integers. Salt-and-pepper noise flipped the pixel values by a certain probability. Replacement noise flipped the pixel values as to a more complex probability model. The chance of a pixel being flipped increased as the pixel was closer to areas of the opposite value and as such areas increased. Replacement noise was common to documents which have been manipulated quite a few times by facsimiling, copying and so on. Some noisy images are given in Figure 2.

Table 1: Confusion matrix for Schauf *et al.*'s binary shape recognition algorithm. Rows represent correct models and columns represent detected models. Performance measures include misdetection (MD), average location error (ALE), average scale error (ASE) and false alarm (FA).

	Assigned Models										MD	ALE	ASE
Original Models	32	0	0	0	0	0	0	0	0	0	0	2.6939	0.0109
	0	28	0	0	0	0	0	0	0	0	0	2.2733	0.0081
	0	0	30	0	0	0	0	0	0	0	3	3.3781	0.0124
	0	0	0	30	0	0	0	0	0	0	0	3.1621	0.0120
	0	0	0	0	16	0	0	0	0	0	0	4.1056	0.0191
	0	0	0	0	0	22	0	0	0	0	0	2.7724	0.0111
	0	0	0	0	0	0	20	0	0	0	0	2.1297	0.0117
	0	0	0	0	0	0	0	21	0	0	0	1.4366	0.0054
	0	0	0	0	0	0	0	0	27	0	1	1.9564	0.0073
	0	0	0	0	0	0	0	0	0	28	1	2.5694	0.0119
FA	0	0	0	0	0	11	2	0	0	0			


(a) Original image (b) Noise level 1

(c) Noise level 2 (d) Noise level 3

Figure 2: Example symbol images.

The package also included a program for performance evaluation. The program read the groundtruth information for each image, compared it with the recognition results and output a confusion matrix. For convenience of comparison, we summarized the confusion matrices of all sizes and noise levels by the number of wrong assignments, misdetections, false alarms, average location errors and average scale errors.

3.2. Experiments

First, 50 images that contain 25 symbols with sizes "all 75x75", "all 50x50" and "all random" were generated. This made up 150 images with 25 symbols in each image. Then each image was perturbed with three levels of noise separately. This made up an additional of 450 images.

3.2.1. Ye and Haralick Algorithm

Two algorithms were submitted to this contest. The first one is by Ming Ye and Robert M. Haralick from the Intelligent Systems Laboratory, University of Washington, USA. This is a segmentation-free symbol recognition algorithm [8] purely using mathematical morphology. Given a page of symbols, the algorithm simultaneously determines the positions and scaling factors of all the symbols which come from the given symbol library. Each symbol has a feature set composed of the relative maxima of the recursive erosion transforms from a few structural elements. The feature set has the property that the distance between any two features within the set is proportional to their scales. Given an observed feature, the algorithm decides whether it has come from a hypothesized symbol by examining if it establishes a similar proportional relationship with its neighboring observations to that of the symbol. For a given image, the algorithm starts from the feature observation with the largest value and works till all feature points are assigned to certain symbols. Before extracting feature points from the image, morphological closing and opening operations are conducted to reduce noise impact.

A very simple version of this recognition system was implemented. The overall confusion matrix is given in Table 2. The errors are mainly due to two reasons. First, as mathematical morphology operations are fragile to holes, this system breaks down when the noise level is high. Second, because only three features are used for each symbol, this system is not discriminative enough. The first problem can be alleviated by using non-morphological denoising methods and the second can be improved by developing more

Table 2: Confusion matrix for Ye and Haralick's symbol recognition algorithm when the results for all noisy images are combined. Rows represent correct models and columns represent detected models. Performance measures include misdetection (MD), average location error (ALE), average scale error (ASE), false alarm (FA) and wrong scale detection (WSD).

	Assigned Models																									MD	ALE	ASE
	230	0	37	0	0	0	0	0	0	0	0	0	19	14	0	8	64	0	0	0	0	13	0	0	29	66	7.4208	2.0471
	5	202	134	0	0	0	0	0	0	0	0	38	93	0	0	0	9	0	0	0	0	4	0	0	1	129	1.8423	1.0724
	1	0	334	0	0	0	0	0	0	0	0	0	25	0	0	0	0	0	0	0	0	0	0	0	0	129	3.1368	1.3732
	0	0	0	8	0	0	0	0	0	0	0	0	56	0	0	0	0	0	0	24	0	0	0	0	5	126	0.0358	0.0882
	18	0	56	0	1	0	0	0	0	0	0	0	46	0	0	5	7	15	0	0	42	9	0	0	9	245	0.1633	0.0267
	14	0	105	0	0	45	0	0	5	0	0	0	5	0	0	2	0	0	0	0	0	1	0	0	10	206	0.9131	0.2900
	0	0	0	1	5	0	44	5	0	0	0	0	133	0	0	0	0	0	0	6	0	0	0	0	0	259	0.3733	0.2990
	35	0	8	0	0	0	0	66	0	0	0	0	0	0	0	0	15	0	0	0	0	0	0	0	0	371	0.7509	0.3584
	10	0	153	0	0	0	0	0	133	0	0	0	17	0	0	0	11	0	0	0	0	0	0	0	0	150	0.9366	0.4326
	72	0	34	0	0	0	0	1	0	112	0	0	23	2	5	10	5	0	0	0	0	2	0	0	0	175	1.5702	0.8116
	2	0	18	0	0	0	0	0	0	41	57	0	106	0	0	0	3	0	0	0	0	0	0	0	4	72	0.9324	0.1622
	89	0	14	0	0	0	0	0	0	0	0	184	25	2	0	0	24	0	0	0	0	0	0	0	4	216	2.2608	0.8462
	6	0	28	0	0	0	0	0	0	0	0	0	238	0	0	19	15	0	0	0	0	102	0	0	0	276	2.8707	1.3403
	5	0	7	0	0	0	0	0	0	0	0	0	0	102	1	0	0	0	0	0	0	0	0	0	0	71	4.2794	1.1383
	0	0	21	0	0	0	0	0	0	0	0	0	0	0	85	0	0	0	0	0	0	0	0	0	0	158	4.0596	1.6531
	47	0	26	0	0	0	0	0	0	0	0	0	16	3	0	73	35	0	0	0	0	77	0	0	4	142	4.0397	1.5101
	162	0	36	0	0	0	0	5	0	0	0	0	36	5	0	0	183	0	0	0	0	13	0	0	96	19	7.1570	2.7056
	33	0	59	0	0	9	0	0	0	0	5	0	22	5	0	15	10	48	0	0	3	3	0	0	33	244	2.4146	0.7739
	7	0	24	0	0	14	0	1	6	15	4	0	6	0	0	0	8	0	42	0	0	8	0	20	0	145	0.7890	0.2737
	14	0	64	0	0	0	0	5	0	0	0	0	76	0	0	0	14	0	0	237	0	16	0	0	5	160	2.1797	1.1420
	8	0	14	0	0	0	0	0	0	0	0	0	41	0	0	0	0	0	0	0	60	0	0	0	4	59	0.8312	0.2647
	55	4	30	0	0	0	0	0	0	0	0	0	0	0	0	0	63	0	0	0	0	37	0	0	30	255	0.4408	0.0913
	6	2	190	0	0	0	0	0	87	5	5	0	13	0	0	0	9	0	0	0	0	0	166	0	6	243	1.8791	1.1622
	26	0	33	0	0	0	0	3	0	0	0	0	102	0	0	18	55	4	0	0	0	10	0	160	17	199	1.9390	0.8208
	111	0	19	0	0	0	0	0	0	0	0	0	22	0	0	12	60	0	0	0	0	17	0	0	39	86	1.9237	0.4960
FA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
WSD	20	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	14	0	0	3	0	0	0	0	0			

refined features.

3.2.2. Parker, Pivovarov and Royko Algorithm

The second submission is by J. R. Parker, J. Pivovarov and D. Royko from the Department of Computer Science, University of Calgary, Canada. The algorithm [5] first extracts the unknown symbol into a bilevel image which is called image A. Then a scaled vector template is plotted into a blank image with the same size bounding box as the unknown image, which is called image B. Thick lines are plotted in this image, based on the measured "estimated pen width" of image A. Next, the algorithm iterates through all pixels of A and measures how far the nearest matching pixel is in B, i.e. if $A(i,j)$ is a foreground pixel, finds the 8-distance to the nearest foreground pixel in B, if $A(i,j)$ is a background pixel, finds the 8-distance to the nearest background pixel in B. This gives a distance map, $M = \mathrm{map}(A,B)$ where $M(i,j)$ is the distance from pixel $A(i,j)$ to the nearest matching pixel in B. Then, the squares of all the entries in $\mathrm{map}(A,B)$ and $\mathrm{map}(B,A)$ are summed. This is the measure of goodness-of-fit of the particular orientation of the template plotted in image B to the object in image A. A small set of orientations, $-3°,\ldots,+3°$, are used and the one with minimum distance is found. This is stored as the distance from A to that particular template. This can be implemented efficiently through the use of dynamic programming, and requires a constant (small) number of image passes. Two passes of noise removal consisting of an averaging filter followed by an edge-cleaning filter are performed. A final smoothing is performed along the bounding boxes of each extracted symbol.

This algorithm achieved a recognition rate of 100% on all test data with an unoptimized execution time of approximately 5 symbols per second on an Intel Celeron 400MHz PC. The confusion matrix is given in Table 3.

4. Image Flow Estimation

4.1. Data Generation

This package included synthetic and real image sequences and their optic flow groundtruth generation for performance evaluation in terms of false alarm rate, misdetection rate and average error vector magnitude. The two synthetic image sequences were rot and div. They were generated using a ray-tracing method which traced each ray passing through the camera and an image pixel, and found out if it touched the surface of the 3D object. If it did, the surface intensity was recorded for that pixel, otherwise a background value was recorded. For accuracy of the inten-

Table 3: Confusion matrix for Parker *et al.*'s symbol recognition algorithm when the results for all noisy images are combined.

	Assigned Models																									MD	ALE	ASE
	480	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.7248	0.7846
	0	615	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.0456	1.3737
	0	0	489	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.9917	0.6351
	0	0	0	219	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.8746	2.1794
	0	0	0	0	453	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.9401	2.0193
	0	0	0	0	0	393	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.9213	3.0666
	0	0	0	0	0	0	453	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.8852	2.5840
	0	0	0	0	0	0	0	495	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.0261	0.9972
	0	0	0	0	0	0	0	0	474	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.9627	1.5840
	0	0	0	0	0	0	0	0	0	441	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.0177	1.6244
	0	0	0	0	0	0	0	0	0	0	303	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.8461	1.1549
	0	0	0	0	0	0	0	0	0	0	0	558	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.8632	1.2364
	0	0	0	0	0	0	0	0	0	0	0	0	684	0	0	0	0	0	0	0	0	0	0	0	0	0	0.8734	0.7166
	0	0	0	0	0	0	0	0	0	0	0	0	0	186	0	0	0	0	0	0	0	0	0	0	0	0	0.8547	3.3598
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	264	0	0	0	0	0	0	0	0	0	0	0	0.7557	2.4719
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	423	0	0	0	0	0	0	0	0	0	0	1.2268	0.2076
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	555	0	0	0	0	0	0	0	0	0	0.6594	0.9158
	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	488	0	0	0	0	0	0	0	0	1.3944	0.6332
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	300	0	0	0	0	0	0	0	1.1690	0.4996
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	591	0	0	0	0	0	0	0.8583	0.4768
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	186	0	0	0	0	0	1.3291	0.3490
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	474	0	0	0	0	1.0470	3.6406
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	732	0	0	0	0.9582	0.7132
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	627	0	0	0.8403	0.6434
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	366	0	0.7517	0.4322
FA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
WSD	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			

sity values, a sinusoidal function was used as the 3D surface pattern. Example frames and flow field groundtruth for the synthetic image sequences are given in Figure 3. Three real image sequences, Taxi, Rubik and SRI were obtained from Barron *et al.* [1]. Examples are given in Figure 4.

4.2. Experiments

4.2.1. Ye and Haralick Algorithm

Three algorithms were submitted to this contest. The first one is by Ming Ye and Robert M. Haralick from the Intelligent Systems Laboratory, University of Washington, USA. The algorithm [9] forms a set of constraint equations from the first and second order derivatives for each pixel, and solves a combined set of equations from a neighborhood for the image flow at the central pixel by assuming a constant local motion model. The derivatives are estimated from a 3D cubic facet model. This image flow estimation scheme has shown generally good results. Besides it provides a covariance matrix with each estimate as a reliable error measurement, which can help subsequent applications make judicious use of the image flow estimates. The covariance matrix is obtained from propagating image noise through the facet model and the image flow constraint equation, considering the correlation of the constraints. Its effectiveness has been verified by a successful χ^2 hypothesis testing based selection scheme.

The false alarm rate, misdetection rate and the average absolute error vector magnitude for the synthetic image sequences are given in Table 4. The optical flow fields for the real image sequences are given in Figure 4.

4.2.2. Sun Algorithm

The second submission is by Changming Sun from CSIRO Mathematical and Information Sciences, Australia. The algorithm [7] uses fast area cross correlation and 3D shortest-path techniques to obtain a dense optical flow field. Fast correlation is achieved by using the box filtering technique which is invariant to the size of the correlation window. The motion for each scan line of the input image is obtained from the correlation coefficient volume by finding the best 3D path using dynamic programming techniques rather than simply choosing the position that gives the maximum cross correlation coefficient. Sub-pixel accuracy is achieved by fitting the local correlation coefficients to a quadratic surface. Currently only two images are used for the optical flow estimation. Typical running time for a 256x256 image is in the order of a few seconds.

The correlation window sizes were in the range of 15x15 and 19x19. The matching search range was from -3 to +3 pixels in both the x and the y directions. The algorithm only used two frames in an image sequence for the flow estimation (the middle ones were used, usually frames 9 and 10). The running time of the algorithm was 14.35s for an image of size 316x252 on a 85MHz SUN SPARCserver1000. The false alarm rate, misdetection rate and the average absolute error vector magnitude for the synthetic image sequences are given in Table 4. The optical flow fields obtained for the real image sequences are given in Figure 4.

4.2.3. Farnebäck Algorithm

The third submission is by Gunnar Farnebäck from the Computer Vision Laboratory, Linköping University, Sweden. The algorithm [2, 3] starts by computing 3D spatiotemporal orientation tensors from the image sequence. This is done by a method based on carefully weighted least squares approximations of signal neighborhoods by quadratic polynomials. Then the orientation tensors are combined under the constraints of a parametric motion model, in this case just a translation, to produce velocity estimates. This is done locally in each neighborhood, without regard to possible discontinuities in the velocity field, but with a Gaussian weighting of the points in the neighborhood. Computationally the weighted least squares approximations are most demanding, but since these can be computed efficiently by a hierarchical scheme of separable convolutions, the algorithm is very fast.

The algorithm used filters of effective size 9x9x9 and the tensors were combined over local neighborhoods of size 15x15. On a Sun Ultra 30, this took 2 seconds for each sequence (velocities were computed only for the frame where we had groundtruth). The algorithm was implemented as Matlab m-files, except for the convolutions which were computed by a Matlab mex-file implemented in C. The false alarm rate, misdetection rate and the average absolute error vector magnitude for the synthetic image sequences are given in Table 4. The optical flow fields obtained for the real image sequences are given in Figure 4.

5. Conclusions

Because it takes a considerable amount of effort to prepare ground truthed data sets and evaluation software for pattern recognition processing algorithms, it was hoped that there would be many researchers who would participate in the contests. Unfortunately, this was not the case. We will keep the data sets and evaluation software out on the web so that by next ICPR more researchers will have tried their hand at these tasks and a more comprehensive discussion and comparison of techniques can be made.

References

[1] J. L. Barron, S. S. Beauchemin, and D. J. Fleet. Performance of optical flow techniques. *International Journal of Computer Vision*, 12(1):43–77, 1994. ftp://csd.uwo.ca/pub/vision/TESTDATA/.

[2] G. Farnebäck. Spatial Domain Methods for Orientation and Velocity Estimation. Lic. Thesis LiU-Tek-Lic-1999:13, Dept. EE, Linköping University, SE-581 83 Linköping, Sweden, March 1999. Thesis No. 755, ISBN 91-7219-441-3, ftp://ftp.isy.liu.se/pub/bb/Theses/LicTheses/G_Farneback_lic.ps.gz.

[3] G. Farnebäck. Fast and accurate motion estimation using orientation tensors and parametric motion models. In *Proceedings of 15th IAPR International Conference on Pattern Recognition*, Barcelona, Spain, September 2000.

[4] T. Kanungo, R. M. Haralick, and H. S. Baird. Power functions and their use in selecting distance functions for document degradation model validation. In *Proceedings of the Third International Conference on Document Analysis and Recognition*, volume 2, pages 734–739, 1995.

[5] J. R. Parker. Vector templates and handprinted symbol recognition. Technical Report 95/559/11, University of Calgary, Department of Computer Science, Calgary, Canada, 1995.

[6] M. Schauf, S. Aksoy, and R. M. Haralick. Model-based shape recognition using recursive mathematical morphology. In *Proceedings of 14th IAPR International Conference on Pattern Recognition*, volume 1, pages 202–204, Brisbane, Australia, August 16–20 1998.

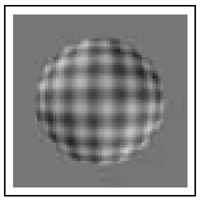
[7] C. Sun. Fast optical flow using cross correlation and shortest-path techniques. In *Proceedings of Digital Image Computing: Techniques and Applications*, pages 143–148, Perth, Australia, December 7–8 1999. http://extra.cmis.csiro.au/IA/changs/doc/motion_dicta99.ps.gz.

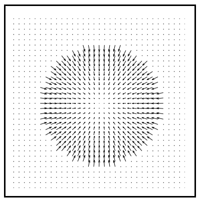
[8] M. Ye and R. M. Haralick. Recognizing symbols using mathematical morphology. Technical report, Intelligent Systems Laboratory, 1998.

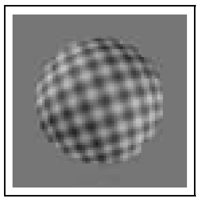
[9] M. Ye and R. M. Haralick. Image flow estimation using facet model and covariance propagation. In M. Cheriet and Y. H. Yang, editors, *Vision Interface: Real World Applications of Computer Vision*, volume 35 of *MPAI*, pages 209–241. World Scientific Pub Co., 2000.

Table 4: Image flow quantitative comparison on div and rot sequences. The false alarm rate (FA), misdetection rate (MD) and the average absolute error vector magnitude (AEVM) are given for each algorithm.

	Div			Rot		
	FA	MD	AEVM	FA	MD	AEVM
Ye	4.76	1.26	0.0388	3.39	0.67	0.0931
Sun	40.6	0	0.0784	31.12	0	0.1254
Farnebäck	50.66	0	0.0270	40.45	0	0.0481


(a) Div frame


(b) Div flow field


(c) Rot frame

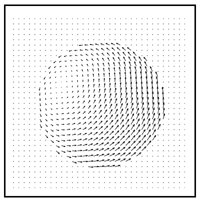
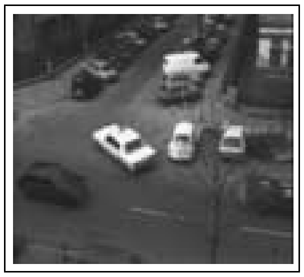
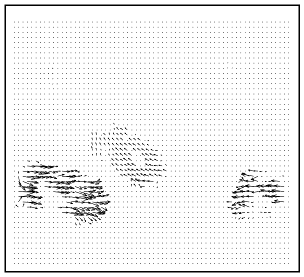
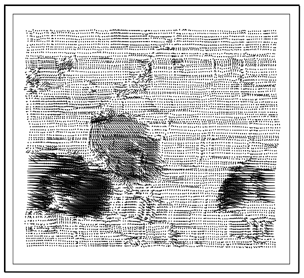
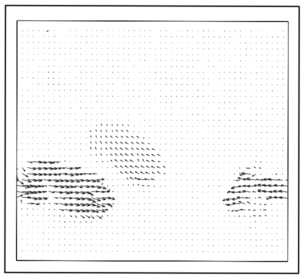

(d) Rot flow field

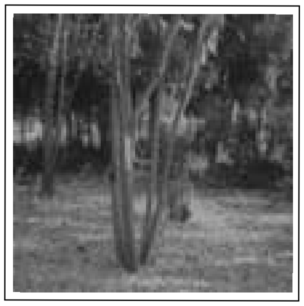
Figure 3: Central frames of both div and rot sequences, and their true flow fields.

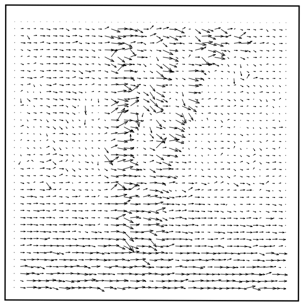

(a) Taxi

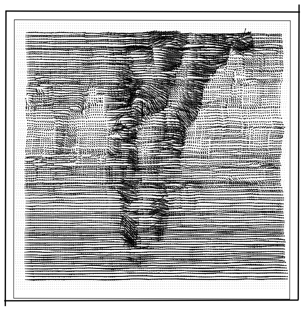

(b) Taxi – Ye

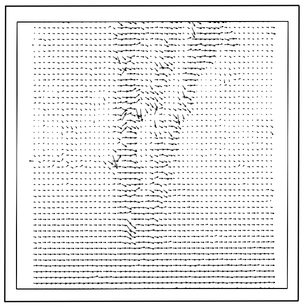

(c) Taxi – Sun

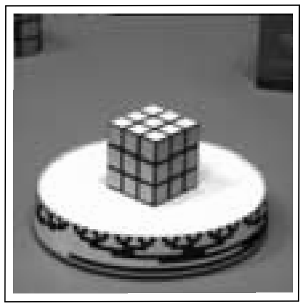

(d) Taxi – Farnebäck

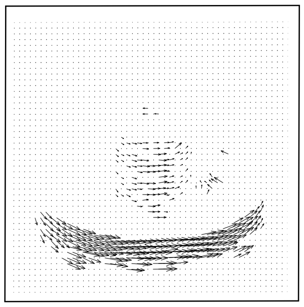

(e) SRI

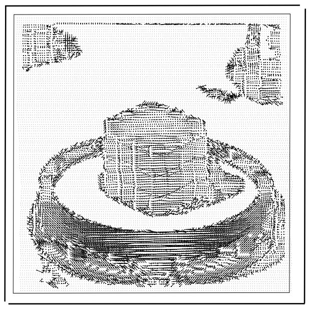

(f) SRI – Ye


(g) SRI – Sun


(h) SRI – Farnebäck


(i) Rubik


(j) Rubik – Ye


(k) Rubik – Sun

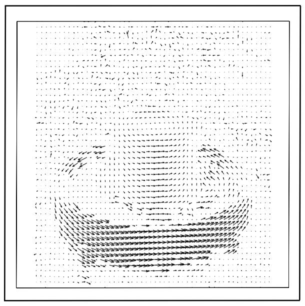

(l) Rubik – Farnebäck

Figure 4: Image flow test sequences and flow field results.