Machine learning for high-speed corner detection

Edward Rosten and Tom Drummond

Department of Engineering, Cambridge University, UK
{er258, twd20}@cam.ac.uk

Abstract Where feature points are used in real-time frame-rate applications, a high-speed feature detector is necessary. Feature detectors such as SIFT (DoG), Harris and SUSAN are good methods which yield high quality features, however they are too computationally intensive for use in real-time applications of any complexity. Here we show that machine learning can be used to derive a feature detector which can fully process live PAL video using less than 7% of the available processing time. By comparison neither the Harris detector (120%) nor the detection stage of SIFT (300%) can operate at full frame rate.
Clearly a high-speed detector is of limited use if the features produced are unsuitable for downstream processing. In particular, the same scene viewed from two different positions should yield features which correspond to the same real-world 3D locations[1]. Hence the second contribution of this paper is a comparison corner detectors based on this criterion applied to 3D scenes. This comparison supports a number of claims made elsewhere concerning existing corner detectors. Further, contrary to our initial expectations, we show that despite being principally constructed for speed, our detector significantly outperforms existing feature detectors according to this criterion.

1 Introduction

Corner detection is used as the first step of many vision tasks such as tracking, SLAM (simultaneous localisation and mapping), localisation, image matching and recognition. Hence, a large number of corner detectors exist in the literature. With so many already available it may appear unnecessary to present yet another detector to the community; however, we have a strong interest in real-time frame rate applications such as SLAM in which computational resources are at a premium. In particular, it is still true that when processing live video streams at full frame rate, existing feature detectors leave little if any time for further processing, even despite the consequences of Moore's Law.

Section 2 of this paper demonstrates how a feature detector described in earlier work can be redesigned employing a machine learning algorithm to yield a large speed increase. In addition, the approach allows the detector to be generalised, producing a suite of high-speed detectors which we currently use for real-time tracking [2] and AR label placement [3].

To show that speed can been obtained without necessarily sacrificing the quality of the feature detector we compare our detector, to a variety of well-known detectors. In Section 3 this is done using Schmid's criterion [1], that

when presented with different views of a 3D scene, a detector should yield (as far as possible) corners that correspond to the same features in the scene. Here we show how this can be applied to 3D scenes for which an approximate surface model is known.

1.1 Previous work

The majority of feature detection algorithms work by computing a corner response function (C) across the image. Pixels which exceed a threshold cornerness value (and are locally maximal) are then retained.

Moravec [4] computes the sum-of-squared-differences (SSD) between a patch around a candidate corner and patches shifted a small distance in a number of directions. C is then the smallest SSD so obtained, thus ensuring that extracted corners are those locations which change maximally under translations.

Harris[5] builds on this by computing an approximation to the second derivative of the SSD with respect to the shift The approximation is:

$$\mathbf{H} = \begin{bmatrix} \widehat{I_x^2} & \widehat{I_x I_y} \\ \widehat{I_x I_y} & \widehat{I_y^2} \end{bmatrix}, \tag{1}$$

where $\widehat{\ }$ denotes averaging performed over the image patch (a smooth circular window can be used instead of a rectangle to perform the averaging resulting in a less noisy, isotropic response). Harris then defines the corner response to be

$$C = |\mathbf{H}| - k(\operatorname{trace}\mathbf{H})^2. \tag{2}$$

This is large if both eigenvalues of $\mathbf{H}$ are large, and it avoids explicit computation of the eigenvalues. It has been shown[6] that the eigenvalues are an approximate measure of the image curvature.

Based on the assumption of affine image deformation, a mathematical analysis led Shi and Tomasi[7] conclude that it is better to use the smallest eigen value of $\mathbf{H}$ as the corner strength function:

$$C = \min\ (\lambda_1, \lambda_2). \tag{3}$$

A number of suggestion have [5,7,8,9] been made for how to compute the corner strength from $\mathbf{H}$ and these have been all shown [10] to be equivalent to various matrix norms of $\mathbf{H}$

Zheng et al.[11] perform an analysis of the computation of $\mathbf{H}$, and find some suitable approximations which allow them to obtain a speed increase by computing only two smoothed images, instead of the three previously required.

Lowe [12] obtains scale invariance by convolving the image with a Difference of Gaussians (DoG) kernel at multiple scales, retaining locations which are optima in scale as well as space. DoG is used because it is good approximation for the Laplacian of a Gaussian (LoG) and much faster to compute. An approximation to DoG has been proposed which, provided that scales are $\sqrt{2}$ apart,

speeds up computation by a factor of about two, compared to the striaghtforward implementation of Gaussian convolution [13].

It is noted in [14] that the LoG is a particularly stable scale-space kernel.

Scale-space techniques have also been combined with the Harris approach in [15] which computes Harris corners at multiple scales and retains only those which are also optima of the LoG response across scales.

Recently, scale invariance has been extended to consider features which are invariant to affine transformations [14,16,17].

An edge (usually a step change in intensity) in an image corresponds to the boundary between two regions. At corners of regions, this boundary changes direction rapidly. Several techniques were developed which involved detecting and chaining edges with a view to finding corners in the chained edge by analysing the chain code[18], finding maxima of curvature [19,20,21], change in direction [22] or change in appearance[23]. Others avoid chaining edges and instead look for maxima of curvature[24] or change in direction [25] at places where the gradient is large.

Another class of corner detectors work by examining a small patch of an image to see if it "looks" like a corner. Since second derivatives are not computed, a noise reduction step (such as Gaussian smoothing) is not required. Consequently, these corner detectors are computationally efficient since only a small number of pixels are examined for each corner detected. A corollary of this is that they tend to perform poorly on images with only large-scale features such as blurred images. The corner detector presented in this work belongs to this category.

The method presented in [26] assumes that a corner resembles a blurred wedge, and finds the characteristics of the wedge (the amplitude, angle and blur) by fitting it to the local image. The idea of the wedge is generalised in [27], where a method for calculating the corner strength is proposed which computes self similarity by looking at the proportion of pixels inside a disc whose intensity is within some threshold of the centre (*nucleus*) value. Pixels closer in value to the nucleus receive a higher weighting. This measure is known as the USAN (the Univalue Segment Assimilating Nucleus). A low value for the USAN indicates a corner since the centre pixel is very different from most of its surroundings. A set of rules is used to suppress qualitatively "bad" features, and then local minima of the, SUSANs, (Smallest USAN) are selected from the remaining candidates.

Trajkovic and Hedley[28] use a similar idea: a patch is not self-similar if pixels generally look different from the centre of the patch. This is measured by considering a circle. f_C is the pixel value at the centre of the circle, and f_P and $f_{P'}$ are the pixel values at either end of a diameter line across the circle. The response function is defined as

$$C = \min_P (f_P - f_C)^2 + (f_{P'} - f_C)^2. \tag{4}$$

This can only be large in the case where there corner. The test is performed on a Bresenham circle. Since the circle is discretized, linear or circular interpolation is used in between discrete orientations in order to give the detector a more isotropic response. To this end, the authors present a method whereby the

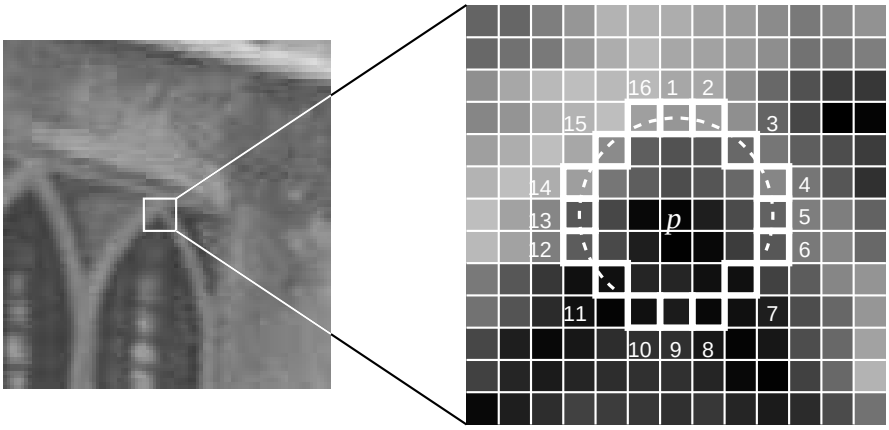

Figure 1. 12 point segment test corner detection in an image patch. The highlighted squares are the pixels used in the corner detection. The pixel at p is the centre of a candidate corner. The arc is indicated by the dashed line passes through 12 contiguous pixels which are brighter than p by more than the threshold.

minimum response function at all interpolated positions between two pixels can be efficiently computed. Computing the response function requires performing a search over all orientations, but any single measurement provides an upper bound on the response. To speed up matching, the response in the horizontal and vertical directions only is checked. If the upper bound on the response is too low, then the potential corner is rejected. To speed up the method further, this fast check is first applied at a coarse scale.

A fast radial symmetry transform is developed in [29] to detect points. Points have a high score when the gradient is both radially symmetric, strong, and of a uniform sign along the radius. The scale can be varied by changing the size of the area which is examined for radial symmetry.

An alternative method of examining a small patch of an image to see if it looks like a corner is to use machine learning to classify patches of the image as corners or non-corners. The examples used in the training set determine the type of features detected. In [30], a three layer neural network is trained to recognise corners where edges meet at a multiple of 45°, near to the centre of an 8×8 window. This is applied to images after edge detection and thinning. It is shown how the neural net learned a more general representation and was able to detect corners at a variety of angles.

2 High-speed corner detection

2.1 FAST: Features from Accelerated Segment Test

The segment test criterion operates by considering a circle of sixteen pixels around the corner candidate p. The original detector [2,3] classifies p as a corner

if there exists a set of n contiguous pixels in the circle which are all brighter than the intensity of the candidate pixel I_p plus a threshold t, or all darker than $I_p - t$, as illustrated in Figure 1. n was chosen to be twelve because it admits a high-speed test which can be used to exclude a very large number of non-corners: the test examines only the four pixels at 1, 5, 9 and 13 (the four compass directions). If p is a corner then at least three of these must all be brighter than $I_p + t$ or darker than $I_p - t$. If neither of these is the case, then p cannot be a corner. The full segment test criterion can then be applied to the remaining candidates by examining all pixels in the circle. This detector in itself exhibits high performance, but there are several weaknesses:

1. The high-speed test does not generalise well for $n < 12$.
2. The choice and ordering of the fast test pixels contains implicit assumptions about the distribution of feature appearance.
3. Knowledge from the first 4 tests is discarded.
4. Multiple features are detected adjacent to one another.

2.2 Machine learning a corner detector

Here we present an approach which uses machine learning to address the first three points (the fourth is addressed in Section 2.3). The process operates in two stages. In order to build a corner detector for a given n, first, corners are detected from a set of images (preferably from the target application domain) using the segment test criterion for n and a convenient threshold. This uses a slow algorithm which for each pixel simply tests all 16 locations on the circle around it.

For each location on the circle $x \in \{1..16\}$, the pixel at that position relative to p (denoted by $p \to x$) can have one of three states:

$$S_{p\to x} = \begin{cases} d, & I_{p\to x} \le I_p - t \quad \text{(darker)} \\ s, & I_p - t < I_{p\to x} < I_p + t \quad \text{(similar)} \\ b, & I_p + t \le I_{p\to x} \quad \text{(brighter)} \end{cases} \tag{5}$$

Choosing an x and computing $S_{p\to x}$ for all $p \in P$ (the set of all pixels in all training images) partitions P into three subsets, P_d, P_s, P_b, where each p is assigned to $P_{S_{p\to x}}$.

Let K_p be a boolean variable which is true if p is a corner and false otherwise. Stage 2 employs the algorithm used in ID3[31] and begins by selecting the x which yields the most information about whether the candidate pixel is a corner, measured by the entropy of K_p.

The entropy of K for the set P is:

$$H(P) = (c + \bar{c}) \log_2(c + \bar{c}) - c \log_2 c - \bar{c} \log_2 \bar{c} \tag{6}$$

$$\text{where} \quad c = \left|\{p | K_p \text{ is true}\}\right| \quad \text{(number of corners)}$$

$$\text{and} \quad \bar{c} = \left|\{p | K_p \text{ is false}\}\right| \quad \text{(number of non corners)}$$

The choice of x then yields the information gain:

$$H(P) - H(P_d) - H(P_s) - H(P_b) \tag{7}$$

Having selected the x which yields the most information, the process is applied recursively on all three subsets i.e. x_b is selected to partition P_b in to $P_{b,d}$, $P_{b,s}$, $P_{b,b}$, x_s is selected to partition P_s in to $P_{s,d}$, $P_{s,s}$, $P_{s,b}$ and so on, where each x is chosen to yield maximum information about the set it is applied to. The process terminates when the entropy of a subset is zero. This means that all p in this subset have the same value of K_p, i.e. they are either all corners or all non-corners. This is guaranteed to occur since K is an exact function of the learning data.

This creates a decision tree which can correctly classify all corners seen in the training set and therefore (to a close approximation) correctly embodies the rules of the chosen FAST corner detector. This decision tree is then converted into C-code, creating a long string of nested if-then-else statements which is compiled and used as a corner detector. For full optimisation, the code is compiled twice, once to obtain profiling data on the test images and a second time with arc-profiling enabled in order to allow reordering optimisations. In some cases, two of the three subtrees may be the same. In this case, the boolean test which separates them is removed.

Note that since the data contains incomplete coverage of all possible corners, the learned detector is not precisely the same as the segment test detector. It would be relatively straightforward to modify the decision tree to ensure that it has the same results as the segment test algorithm, however, all feature detectors are heuristic to some degree, and the learned detector is merely a very slightly different heuristic to the segment test detector.

2.3 Non-maximal suppression

Since the segment test does not compute a corner response function, non maximal suppression can not be applied directly to the resulting features. Consequently, a score function, V must be computed for each detected corner, and non-maximal suppression applied to this to remove corners which have an adjacent corner with higher V. There are several intuitive definitions for V:

1. The maximum value of n for which p is still a corner.
2. The maximum value of t for which p is still a corner.
3. The sum of the absolute difference between the pixels in the contiguous arc and the centre pixel.

Definitions 1 and 2 are very highly quantised measures, and many pixels share the same value of these. For speed of computation, a slightly modified version of 3 is used. V is given by:

$$V = \max \left(\sum_{x \in S_{\text{bright}}} |I_{p \to x} - I_p| - t \ , \ \sum_{x \in S_{\text{dark}}} |I_p - I_{p \to x}| - t \right) \tag{8}$$

Detector	Opteron 2.6GHz		Pentium III 850MHz	
	ms	%	ms	%
Fast $n = 9$ (non-max suppression)	1.33	6.65	5.29	26.5
Fast $n = 9$ (raw)	1.08	5.40	4.34	21.7
Fast $n = 12$ (non-max suppression)	1.34	6.70	4.60	23.0
Fast $n = 12$ (raw)	1.17	5.85	4.31	21.5
Original FAST $n = 12$ (non-max suppression)	1.59	7.95	9.60	48.0
Original FAST $n = 12$ (raw)	1.49	7.45	9.25	48.5
Harris	24.0	120	166	830
DoG	60.1	301	345	1280
SUSAN	7.58	37.9	27.5	137.5

Table 1. Timing results for a selection of feature detectors run on fields (768×288) of a PAL video sequence in milliseconds, and as a percentage of the processing budget per frame. Note that since PAL and NTSC, DV and 30Hz VGA (common for webcams) have approximately the same pixel rate, the percentages are widely applicable. Approximately 500 features per field are detected.

with

$$\begin{aligned} S_{\text{bright}} &= \{x | I_{p \to x} \geq I_p + t\} \\ S_{\text{dark}} &= \{x | I_{p \to x} \leq I_p - t\} \end{aligned} \tag{9}$$

2.4 Timing results

Timing tests were performed on a 2.6GHz Opteron and an 850MHz Pentium III processor. The timing data is taken over 1500 monochrome fields from a PAL video source (with a resolution of 768×288 pixels). The learned FAST detectors for $n = 9$ and 12 have been compared to the original FAST detector, to our implementation of the Harris and DoG (difference of Gaussians—the detector used by SIFT) and to the reference implementation of SUSAN[32].

As can be seen in Table 1, FAST in general offers considerably higher performance than the other tested feature detectors, and the learned FAST performs up to twice as fast as the handwritten version. Importantly, it is able to generate an efficient detector for $n = 9$, which (as will be shown in Section 3) is the most reliable of the FAST detectors. On modern hardware, FAST consumes only a fraction of the time available during video processing, and on low power hardware, it is the only one of the detectors tested which is capable of video rate processing at all.

Examining the decision tree shows that on average, 2.26 (for $n = 9$) and 2.39 (for $n = 12$) questions are asked per pixel to determine whether or not it is a feature. By contrast, the handwritten detector asks on average 2.8 questions.

Interestingly, the difference in speed between the learned detector and the original FAST are considerably less marked on the Opteron processor compared to the Pentium III. We believe that this is in part due to the Opteron having

a diminishing cost per pixel queried that is less well modelled by our system (which assumes equal cost for all pixel accesses), compared to the Pentium III.

3 A comparison of detector repeatability

Although there is a vast body of work on corner detection, there is much less on the subject of comparing detectors. Mohannah and Mokhtarian[33] evaluate performance by warping test images in an affine manner by a known amount. They define the 'consistency of corner numbers' as

$$CCN = 100 \times 1.1^{-|n_w - n_o|},$$

where n_w is the number of features in the warped image and n_o is the number of features in the original image. They also define accuracy as

$$ACU = 100 \times \frac{\frac{n_a}{n_o} + \frac{n_a}{n_g}}{2},$$

where n_g are the number of 'ground truth' corners (marked by humans) and n_a is the number of matched corners compared to the ground truth. This unfortunately relies on subjectively made devisions.

Trajkovic and Hedley[28] define stability to be the number of 'strong ' matches (matches detected over three frames in their tracking algorithm) divided by the total number of corners. This measurement is clearly dependent on both the tracking and matching methods used, but has the advantage that it can be tested on the date used by the system.

When measuring reliability, what is important is if the same real-world features are detected from multiple views [1] This is the definition which will be used here. For an image pair, a feature is 'detected' if is is extracted in one image and appears in the second. It is 'repeated' if it is also detected nearby in the second. The repeatability is the ratio of repeated features detected features. In [1], the test is performed on images of planar scenes so that the relationship between point positions is a homography. Fiducial markers are projected on to the planar scene to allow accurate computation of this.

By modelling the surface as planar and using flat textures, this technique tests the feature detectors' ability to deal with mostly affine warps (since image features are small) under realistic conditions. This test is not so well matched to our intended application domain, so instead, we use a 3D surface model to compute where detected features should appear in other views (illustrated in Figure 2). This allows the repeatability of the detectors to be analysed on features caused by geometry such as corners of polyhedra, occlusions and T-junctions. We also allow bas-relief textures to be modelled with a flat plane so that the repeatability can be tested under non-affine warping.

A margin of error must be allowed because:

1. The alignment is not perfect.

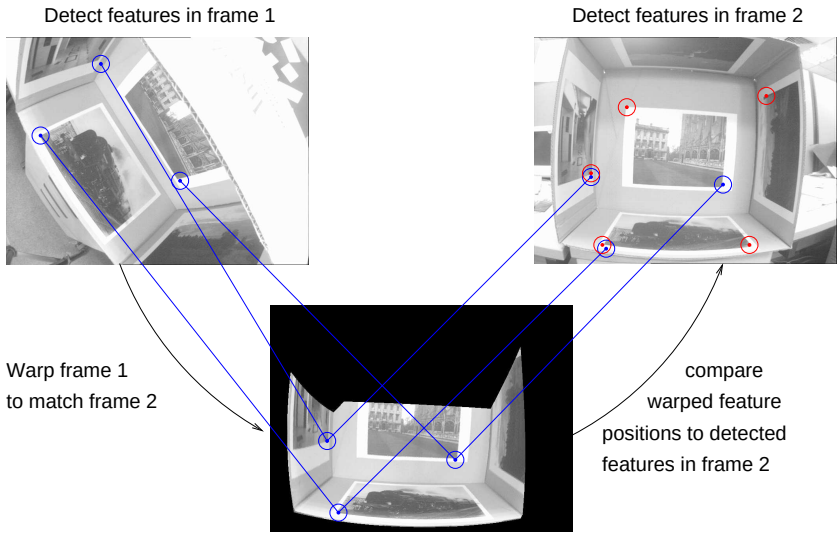

Figure 2. Repeatability is tested by checking if the same real-world features are detected in different views. A geometric model is used to compute where the features reproject to.

2. The model is not perfect.
3. The camera model (especially regarding radial distortion) is not perfect.
4. The detector may find a maximum on a slightly different part of the corner. This becomes more likely as the change in viewpoint and hence change in shape of the corner become large.

Instead of using fiducial markers, the 3D model is aligned to the scene by hand and this is then optimised using a blend of simulated annealing and gradient descent to minimise the SSD between all pairs of frames and reprojections.

To compute the SSD between frame i and reprojected frame j, the position of all points in frame j are found in frame i. The images are then bandpass filtered. High frequencies are removed to reduce noise, while low frequencies are removed to reduce the impact of lighting changes. To improve the speed of the system, the SSD is only computed using 1000 random points (as opposed to every point).

The datasets used are shown in Figure 3, Figure 4 and Figure 5. With these datasets, we have tried to capture a wide range of corner types (geometric and textural).

The repeatability is computed as the number of corners per frame is varied. For comparison we also include a scattering of random points as a baseline measure, since in the limit if every pixel is detected as a corner, then the repeatability is 100%.

To test robustness to image noise, increasing amounts of Gaussian noise were added to the bas-relief dataset. It should be noted that the noise added is in

Figure 3. Box dataset: photographs taken of a test rig (consisting of photographs pasted to the inside of a cuboid) with strong changes of perspective, changes in scale and large amounts of radial distortion. This tests the corner detectors on planar textures.

Figure 4. Maze dataset: photographs taken of a prop used in an augmented reality application. This set consists of textural features undergoing projective warps as well as geometric features. There are also significant changes of scale.

addition to the significant amounts of camera noise already present (from thermal noise, electrical interference, and etc).

4 Results and Discussion

Shi and Tomasi [7], derive their result for better feature detection on the assumption that the deformation of the features is affine. In the box and maze datasets, this assumption holds and can be seen in Figure 6B and Figure 6C the detector outperforms the Harris detector. In the bas-relief dataset, this assumption does not hold, and interestingly, the Harris detector outperforms Shi and Tomasi detector in this case.

Mikolajczyk and Schmid [15] evaluate the repeatability of the Harris-Laplace detector evaluated using the method in [34], where planar scenes are examined. The results show that Harris-Laplace points outperform both DoG points and Harris points in repeatability. For the box dataset, our results verify that this is correct for up to about 1000 points per frame (typical numbers, probably commonly used); the results are somewhat less convincing in the other datasets, where points undergo non-projective changes.

In the sample implementation of SIFT[35], approximately 1000 points are generated on the images from the test sets. We concur that this a good choice for the number of features since this appears to be roughly where the repeatability curve for DoG features starts to flatten off.

Smith and Brady[27] claim that the SUSAN corner detector performs well in the presence of noise since it does not compute image derivatives, and hence, does not amplify noise. We support this claim: although the noise results show

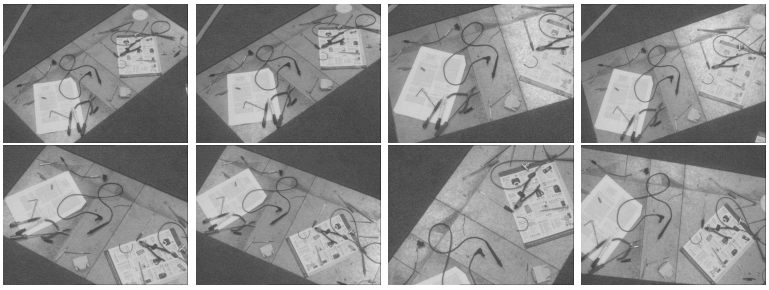

Figure 5. Bas-relief dataset: the model is a flat plane, but there are many objects with significant relief. This causes the appearance of features to change in a non affine way from different viewpoints.

that the performance drops quite rapidly with increasing noise to start with, it soon levels off and outperforms all but the DoG detector.

The big surprise of this experiment is that the FAST feature detectors, despite being designed only for speed, outperform the other feature detectors on these images (provided that more than about 200 corners are needed per frame). It can be seen in Figure 6A, that the 9 point detector provides optimal performance, hence only this and the original 12 point detector are considered in the remaining graphs.

The DoG detector is remarkably robust to the presence of noise. Since convolution is linear, the computation of DoG is equivalent to convolution with a DoG kernel. Since this kernel is symmetric, this is equivalent to matched filtering for objects with that shape. The robustness is achieved because matched filtering is optimal in the presence of additive Gaussian noise[36].

FAST, however, is not very robust to the presence of noise. This is to be expected: Since high speed is achieved by analysing the fewest pixels possible, the detector's ability to average out noise is reduced.

5 Conclusions

In this paper, we have used machine learning to derive a very fast, high quality corner detector. It has the following advantages:

- It is many times faster than other existing corner detectors.
- High levels of repeatability under large aspect changes and for different kinds of feature.

However, it also suffers from a number of disadvantages:

- It is not robust to high levels noise.
- It can respond to 1 pixel wide lines at certain angles, when the quantisation of the circle misses the line.
- It is dependent on a threshold.

We were also able to verify a number of claims made in other papers using the method for evaluating the repeatability of corners and have shown the importance of using more than just planar scenes in this evaluation.

The corner detection code is made available from
http://mi.eng.cam.ac.uk/~er258/work/fast.html
and
http://savannah.nongnu.org/projects/libcvd
and the data sets used for repeatability are available from
http://mi.eng.cam.ac.uk/~er258/work/datasets.html

References

1. Schmid, C., Mohr, R., Bauckhage, C.: Evaluation of interest point detectors. International Journal of Computer Vision **37** (2000) 151–172
2. Rosten, E., Drummond, T.: Fusing points and lines for high performance tracking. In: 10^{th} IEEE International Conference on Computer Vision. Volume 2., Beijing, China, Springer (2005) 1508–1515
3. Rosten, E., Reitmayr, G., Drummond, T.: Real-time video annotations for augmented reality. In: International Symposium on Visual Computing. (2005)
4. Moravec, H.: Obstacle avoidance and navigation in the real world by a seeing robot rover. In: tech. report CMU-RI-TR-80-03, Robotics Institute, Carnegie Mellon University & doctoral dissertation, Stanford University. Carnegie Mellon University (1980) Available as Stanford AIM-340, CS-80-813 and republished as a Carnegie Mellon University Robotics Institue Technical Report to increase availability.
5. Harris, C., Stephens, M.: A combined corner and edge detector. In: Alvey Vision Conference. (1988) 147–151
6. Noble, J.A.: Finding corners. Image and Vision Computing **6** (1988) 121–128
7. Shi, J., Tomasi, C.: Good features to track. In: 9^{th} IEEE Conference on Computer Vision and Pattern Recognition, Springer (1994)
8. Noble, A.: Descriptions of image surfaces. PhD thesis, Department of Engineering Science, University of Oxford. (1989)
9. Kenney, C.S., Manjunath, B.S., Zuliani, M., Hewer, M.G.A., Nevel, A.V.: A condition number for point matching with application to registration and postregistration error estimation. IEEE Transactions on Pattern Analysis and Machine Intelligence **25** (2003) 1437–1454
10. Zuliani, M., Kenney, C., Manjunath, B.: A mathematical comparison of point detectors. In: Second IEEE Image and Video Registration Workshop (IVR), Washington DC, USA (2004)
11. Zheng, Z., Wang, H., Teoh, E.K.: Analysis of gray level corner detection. Pattern Recognition Letters **20** (1999) 149–162
12. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision **60** (2004) 91–110
13. James L. Crowley, O.R.: Fast computation of characteristic scale using a half octave pyramid. In: Scale Space 03: 4th International Conference on Scale-Space theories in Computer Vision, Isle of Skye, Scotland, UK (2003)
14. Mikolajczyk, K., Schmid, C.: An affine invariant interest point detector. In: European Conference on Computer Vision, Springer (2002) 128–142 Copenhagen.

15. Mikolajczyk, K., Schmid, C.: Indexing based on scale invariant interest points. In: 8th IEEE International Conference on Computer Vision, Vancouver, Canada, Springer (2001) 525–531
16. Brown, M., Lowe, D.G.: Invariant features from interest point groups. In: 13th British Machine Vision Conference, Cardiff, British Machine Vision Assosciation (2002) 656–665
17. Schaffalitzky, F., Zisserman, A.: Multi-view matching for unordered image sets, or How do I organise my holiday snaps? In: 7th Euproean Conference on Computer Vision, Springer (2002) 414–431
18. Rutkowski, W.S., Rosenfeld, A.: A comparison of corner detection techniques for chain coded curves. Technical Report 623, Maryland University (1978)
19. Langridge, D.J.: Curve encoding and detection of discontinuities. Computer Vision, Graphics and Image Processing **20** (1987) 58–71
20. Medioni, G., Yasumoto, Y.: Corner detection and curve representation using cubic b-splines. Computer Vision, Graphics and Image Processing **39** (1987) 279–290
21. Mokhtarian, F., Suomela, R.: Robust image corner detection through curvature scale space. IEEE Transactions on Pattern Analysis and Machine Intelligence **20** (1998) 1376–1381
22. Haralick, R.M., Shapiro, L.G.: Computer and robot vision. Volume 1. Adison-Wesley (1993)
23. Cooper, J., Venkatesh, S., Kitchen, L.: Early jump-out corner detectors. IEEE Transactions on Pattern Analysis and Machine Intelligence **15** (1993) 823–828
24. Wang, H., Brady, M.: Real-time corner detection algorithm for motion estimation. Image and Vision Computing **13** (1995) 695–703
25. Kitchen, L., Rosenfeld, A.: Gray-level corner detection. Pattern Recognition Letters **1** (1982) 95–102
26. Guiducci, A.: Corner characterization by differential geometry techniques. Pattern Recognition Letters **8** (1988) 311–318
27. Smith, S.M., Brady, J.M.: SUSAN - a new approach to low level image processing. International Journal of Computer Vision **23** (1997) 45–78
28. Trajkovic, M., Hedley, M.: Fast corner detection. Image and Vision Computing **16** (1998) 75–87
29. Loy, G., Zelinsky, A.: A fast radial symmetry transform for detecting points of interest. In: 7th Euproean Conference on Computer Vision, Springer (2002) 358–368
30. Dias, P., Kassim, A., Srinivasan, V.: A neural network based corner detection method. In: IEEE International Conference on Neural Networks. Volume 4., Perth, WA, Australia (1995) 2116–2120
31. Quinlan, J.R.: Induction of decision trees. Machine Learning **1** (1986) 81–106
32. Smith, S.M.: `http://www.fmrib.ox.ac.uk/~steve/susan/susan2l.c` (Accessed 2005)
33. Cootes, T.F., Taylor, C., eds.: Performenace Evaluation of Corner Detection Algorithms under Affine and Similarity Transforms. In Cootes, T.F., Taylor, C., eds.: 12th British Machine Vision Conference, Manchester, British Machine Vision Assosciation (2001)
34. Schmid, C., Mohr, R., Bauckhage, C.: Comparing and evaluating interest points. In: 6th IEEE International Conference on Computer Vision, Bombay, India, Springer (1998) 230–235
35. Lowe, D.G.: Demo software: Sift keypoint detector. `http://www.cs.ubc.ca/~lowe/keypoints/` (Accessed 2005)
36. Sklar, B.: Digital Communications. Prentice Hall (1988)

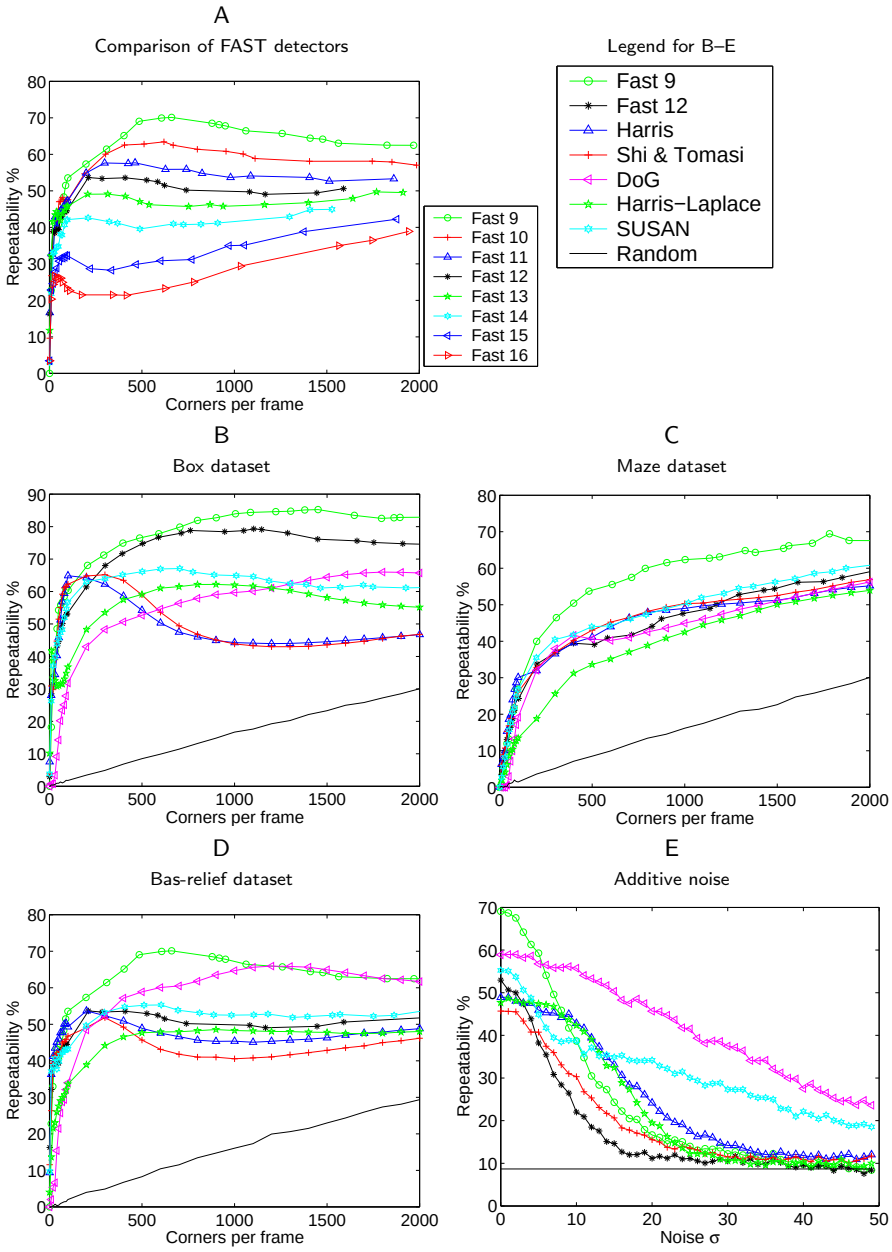

Figure 6. A: A comparison of the FAST detectors shown that $n = 9$ is the most repeatable. For $n \leq 8$, the detector starts to respond strongly to edges. B, C, D: Repeatability results for the three datasets as the number of features per frame is varied. D: repeatability results for the bas-relief data set (500 features per frame) as the amount of Gaussian noise added to the images is varied. For FAST and SUSAN, the number of features can not be chosen arbitrarily; the closest approximation to 500 features per frame achievable is used.