Faster and Better: A Machine Learning Approach to Corner Detection

Edward Rosten, Reid Porter, and Tom Drummond

Abstract—The repeatability and efficiency of a corner detector determines how likely it is to be useful in a real-world application. The repeatability is important because the same scene viewed from different positions should yield features which correspond to the same real-world 3D locations [1]. The efficiency is important because this determines whether the detector combined with further processing can operate at frame rate. Three advances are described in this paper. First, we present a new heuristic for feature detection and, using machine learning, we derive a feature detector from this which can fully process live PAL video using *less than 5 percent* of the available processing time. By comparison, most other detectors cannot even operate at frame rate (Harris detector 115 percent, SIFT 195 percent). Second, we generalize the detector, allowing it to be optimized for repeatability, with little loss of efficiency. Third, we carry out a rigorous comparison of corner detectors based on the above repeatability criterion applied to 3D scenes. We show that, despite being principally constructed for speed, on these stringent tests, our heuristic detector significantly outperforms existing feature detectors. Finally, the comparison demonstrates that using machine learning produces significant improvements in repeatability, yielding a detector that is both very fast and of very high quality.

Index Terms—Corner detection, feature detection.

1 Introduction

Corner detection is used as the first step of many vision tasks such as tracking, localization, simultaneous localization and mapping (SLAM), and image matching and recognition. This need has driven the development of a large number of corner detectors. However, despite the massive increase in computing power since the inception of corner detectors, 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.

In the applications described above, corners are typically detected and matched into a database; thus, it is important that the same real-world points are detected repeatedly from multiple views [1]. The amount of variation in viewpoint under which this condition should hold depends on the application.

2 Previous Work

2.1 Corner Detectors

Here, we review the literature to place our advances in context. In the literature, the terms "point feature," "feature," "interest point," and "corner" refer to a small point of interest with variation in two dimensions. Such points often arise as the result of geometric discontinuities, such as the corners of real-world objects, but they may also arise from small patches of texture. Most algorithms are capable of detecting both kinds of points of interest, though the algorithms are often designed to detect one type or the other. A number of detectors described below compute a corner response C and define corners to be large local maxima of C.

2.1.1 Edge-Based Corner Detectors

An edge (usually a step change in intensity) in an image corresponds to the boundary between two regions. At corners, this boundary changes direction rapidly.

Chained edge-based corner detectors. Many techniques have been developed which involved detecting and chaining edges with a view to analyzing the properties of the edge, often taking points of high curvature to be corners. Many early methods used chained curves, and since the curves are highly quantized, the techniques concentrate on methods for effectively and efficiently estimating the curvature. A common approach has been to use a chord for estimating the slope of a curve or a pair of chords to find the angle of the curve at a point.

Early methods computed the smallest angle of the curve over chords spanning different numbers of links. Corners are defined as local minima of angle [2] after local averaging [3]. Alternatively, corners can be defined as isolated discontinuities in the mean slope, which can be computed using a chord spanning a fixed set of links in the chain [4]. Averaging can be used to compute the slope and the length of the curve used to determine if a point is isolated [5]. The angle can be computed using a pair of chords with a central gap, and peaks with certain widths (found by looking for zero crossings of the angle) are defined as corners [6].

• *E. Rosten and T. Drummond are with the Department of Engineering, Cambridge University, Trumpington Street, Cambridge CB2 1PZ, UK. E-mail: {er258, twd20}@cam.ac.uk.*
• *R. Porter is with the Los Alamos National Laboratory, Mail Stop D.436, ISR2: Space and Remote Sensing, Los Alamos, NM 87544. E-mail: rporter@lanl.gov.*

Manuscript received 10 Mar. 2008; revised 29 July 2008; accepted 16 Oct. 2008; published online 10 Nov. 2008.
Recommended for acceptance by M. Lindenbaum.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number TPAMI-2008-03-0143.
Digital Object Identifier no. 10.1109/TPAMI.2008.275.

Instead of using a fixed set of chord spans, some methods compute a "region of support" which depends on local curve properties. For instance, local maxima of chord lengths can be used to define the region of support within which a corner must have maximal curvature [7]. Corners can be defined as the center of a region of support with high mean curvature, where the support region is large and symmetric about its center [8]. The region free from significant discontinuities around the candidate point can be used with curvature being computed as the slope change across the region [9] or the angle to the region's endpoints [10].

An alternative to using chords of the curves is to apply smoothing to the points on the curve. Corners can be defined as points with a high rate of change of slope [11] or points where the curvature decreases rapidly to the nearest minima and the angle to the neighboring maxima is small [12].

A fixed smoothing scale is not necessarily appropriate for all curves, so corners can also be detected at high curvature points which have stable positions under a range of smoothing scales [13]. As smoothing is decreased, curvature maxima bifurcate, forming a tree over scale. Branches of a tree which are longer (in scale) than the parent branch are considered as stable corner points [14]. Instead of Gaussian smoothing, extrema of the wavelet transforms of the slope [15] or wavelet transform modulus maximum of the angle [16], [17] over multiple scales can be taken to be corners.

The smoothing scale can be chosen adaptively. The Curvature Scale Space technique [18] uses a scale proportional to the length and defines corners at maxima of curvature where the maxima are significantly larger than the closest minima. Locally adaptive smoothing using anisotropic diffusion [19] or smoothing scaled by the local variance of curvature [20] have also been proposed.

Instead of direct smoothing, edges can be parameterized with cubic splines and corners detected at points of high second derivative where the spline deviates a long way from the control point [21], [22].

A different approach is to extend curves past the endpoints by following saddle minima or ridge maxima in the gradient image until a nearby edge is crossed, thereby finding junctions [23]. Since the chain code number corresponds roughly to slope, approximate curvature can be found using finite differences, and corners can be found by identifying specific patterns [24]. Histograms of the chain code numbers on either side of the candidate point can be compared using normalized cross correlation and corners can be found at small local minima [25]. Also, a measure of the slope can be computed using circularly smoothed histograms of the chain code numbers [26]. Points can be classified as corners using fuzzy rules applied to measures computed from the forward and backward arm and the curve angle [27].

Edgel-based corner detectors. Chained edge techniques rely on the method used to perform segmentation and edge chaining, so many techniques find edge points (edgels) and examine the local edgels or image to find corners.

For instance, each combination of presence or absence of edgels in a 3×3 window can be assigned a curvature and corners found as maxima of curvature in a local window [28]. Corners can also be found by analyzing edge properties in the window scanned along the edge [29]. A generalized Hough transform [30] can be used, which replaces each edgel with a line segment, and corners can be found where lines intersect, i.e., at large maxima in Hough space [31]. In a manner similar to chaining, a short line segment can be fitted to the edgels, and the corner strength found by the change in gradient direction along the line segment [32]. Edge detectors often fail at junctions, so corners can be defined as points where several edges at different angles end nearby [33]. By finding both edges and their directions, a patch on an edge can be compared to patches on either side in the direction of the contour to find points with low self-similarity [34].

Rapid changes in the edge direction can be found by measuring the derivative of the gradient direction along an edge and multiplying by the magnitude of the gradient

$$C_K = \frac{g_{xx} g_y^2 + g_{yy} g_x^2 - 2 g_{xy} g_x g_y}{g_x^2 g_y^2}, \tag{1}$$

where, in general,

$$g_x = \frac{\partial g}{\partial x}, \quad g_{xx} = \frac{\partial^2 g}{\partial x^2}, \quad \text{etc.},$$

and g is either the image or a bivariate polynomial fitted locally to the image [35]. C_K can also be multiplied by the change in edge direction along the edge [36].

Corner strength can also be computed as the rate of change in gradient angle when a bicubic polynomial is fitted to the local image surface [37], [38],

$$C_Z = -2 \frac{c_x^2 c_{y^2} - c_x c_y c_{xy} + c_y^2 c_{x^2}}{\left(c_x^2 + c_y^2\right)^{\frac{3}{2}}}, \tag{2}$$

where, for example, c_{xy} is the coefficient of xy in the fitted polynomial. If edgels are only detected at the steepest part of an edge, then a score computing total image curvature at the edgels is given by

$$C_W = \nabla^2 I - S|\nabla I|^2, \tag{3}$$

where ∇I is the image gradient [39].

2.1.2 Graylevel-Derivative-Based Detectors

The assumption that corners exist along edges is an inadequate model for patches of texture and point-like features, and is difficult to use at junctions. Therefore, a large number of detectors operate directly on graylevel images without requiring edge detection.

One of the earliest detectors [40] defines corners to be local extrema in the determinant of the Hessian

$$C_{\mathrm{DET}} = |\mathcal{H}[I]| = I_{xx} I_{yy} - (I_{xy})^2. \tag{4}$$

This is frequently referred to as the DET operator. C_{DET} moves along a line as the scale changes. To counteract this, DET extrema can be found in two scales and connected by a line. Corners are then taken as maxima of the Laplacian along the line [41].

Instead of DET maxima, corners can also be taken as the gradient maxima on a line connecting two nearby points of high Gaussian curvature of opposite sign where the gradient direction matches the sign change [42]. By considering gradients as elementary currents, the magnitude of the corresponding magnetic vector potential can be computed. The gradient of this is taken normal and orthogonal to the local contour direction and the corner strength is the multiple of the magnitude of these [43].

Local sum of squared differences (SSD) detectors. Features can be defined as points with low self-similarity in all directions. The self-similarity of an image patch can be measured by taking the SSD between an image patch and a shifted version of itself [44]. This is the basis for a large class of detectors. Harris and Stephens [45] built on this by computing an approximation to the second derivative of the SSD with respect to the shift. This is both computationally more efficient and can be made isotropic. The result is

$$\mathbf{H} = \begin{bmatrix} \widehat{I_x^2} & \widehat{I_x I_y} \\ \widehat{I_x I_y} & \widehat{I_y^2} \end{bmatrix}, \quad (5)$$

where $\widehat{\ }$ denotes averaging performed over the area of the image patch. Because of the wording used in [45], it is often mistakenly claimed that $\mathbf{H}$ is equal to the negative second derivative of the autocorrelation. This is not the case because the SSD is equal to the sum of the autocorrelation and some additional terms [46].

The earlier Förstner [47] algorithm is easily explained in terms of $\mathbf{H}$. For a more recently proposed detector [48], it has been shown [49] that, under affine motion, it is better to use the smallest eigenvalue of $\mathbf{H}$ as the corner strength function. A number of other suggestions [45], [50], [49], [51] have been made for how to compute the corner strength from $\mathbf{H}$, and these have all been shown to be equivalent to various matrix norms of $\mathbf{H}$ [52]. $\mathbf{H}$ can be generalized by generalizing the number of channels and dimensionality of the image [53] and it can also be shown that [47], [49], [54] are equivalent to specific choices of the measure used in [51].

$\mathbf{H}$ can be explained in terms of the first fundamental form of the image surface [55]. From analysis of the second fundamental form, a new detector is proposed which detects points where the probability of the surface being hyperbolic is high.

Instead of local SSD, general template matching, given a warp, appearance model, and pointwise comparison which behaves similarly to the SSD for small differences can be considered [56]. The stability with respect to the match parameters is derived, and the result is a generalization of $\mathbf{H}$ (where $\mathbf{H}$ is maximally stable for no appearance model, linear translation, and SSD matching). This is used to derive detectors which will give points maximally stable for template matching, given similarity transforms, illumination models, and prefiltering.

Laplacian-based detectors. An alternative approach to the problem of finding a scalar value which measures the amount of second derivative is to take the Laplacian of the image. Since second derivatives greatly amplify noise, the noise is reduced by using the smoothed Laplacian, which is computed by convolving the image with the Laplacian of a Gaussian (LoG). Since the LoG kernel is symmetric, one can interpret this as performing matched filtering for features which are of the same shape as an LoG. As a result, the variance of the Gaussian determines the size of features of interest. It has been noted [57] that the locations of maxima of the LoG over different scales are particularly stable.

Scale-invariant corners can be extracted by convolving the image with a Difference of Gaussians (DoG) kernel at a variety of scales (three per octave) and selecting local maxima in space and scale [58]. DoG is a good approximation for LoG and is much faster to compute, especially as the intermediate results are useful for further processing. To reject edge-like features, the eigenvalues of the Hessian of the image are computed and features are kept if the eigenvalues are sufficiently similar (within a factor of 10). This method can be contrasted with (3), where the Laplacian is compared to the magnitude of the edge response. If two scales per octave are satisfactory, then a significant speed increase can be achieved by using recursive filters to approximate Gaussian convolution [59].

Harris-Laplace [60] features are detected using a similar approach. An image pyramid is built and features are detected by computing C_H at each layer of the pyramid. Features are selected if they are a local maximum of C_H in the image plane and a local maxima of the LoG across scales.

Recently, scale invariance has been extended to consider features which are invariant to affine transformations [57], [61], [62], [63]. However, unlike the 3D scale space, the 6D affine space is too large to search, so all of these detectors start from corners detected in scale space. These, in turn, rely on 2D features selected in the layers of an image pyramid.

2.1.3 Direct Graylevel Detectors

Another major class of corner detectors work by examining a small patch of an image to see if it "looks" like a corner. The detectors described in this paper belong to this section.

Wedge model detectors. A number of techniques assume that a corner has the general appearance of one or more wedges of a uniform intensity on a background of a different uniform intensity. For instance, a corner can be modeled as a single [64] or a family [65] of blurred wedges where the parameters are found by fitting a parametric model. The model can include angle, orientation, contrast, bluntness, and curvature of a single wedge [66]. In a manner similar to [67], convolution masks can be derived for various wedges which optimize signal-to-noise ratio and localization error, under the assumption that the image is corrupted by Gaussian noise [68].

It is more straightforward to detect wedges in binary images, and to get useful results, local thresholding can be used to binarize the image [69]. If a corner is a bilevel wedge, then a response function based on local Zernike moments can be used to detect corners [70]. A more direct method for finding wedges is to find points where concentric contiguous arcs of pixels are significantly different from the center pixel [71]. According to the wedge model, a corner will be the intersection of several edges. An angle-only Hough transform [72] is performed on edgels belonging to lines passing through a candidate point to find their angles, and hence, detect corners [73]. Similar reasoning can be used to derive a response function based on gradient moments to detect V, T, and X-shaped corners [74]. The strength of the edgels, wedge angle, and dissimilarity of the wedge regions has also been used to find corners [75].

Self-dissimilarity. The tip of a wedge is not self-similar, so this can be generalized by defining corners as points which are not self-similar. The proportion of pixels in a disk around a center (or *nucleus*) which are similar to the center is a measure of self similarity. This is the univalue segment assimilating nucleus (USAN). Corners are defined as smallest USAN (SUSAN, i.e., local minima) points which also pass a set of rules to suppress qualitatively bad features. In practice, a weighted sum of the number of pixels inside a disk whose intensity is within some threshold of the center value is used [76]. Crosses as Oriented Pair (COP) [77] computes dominant directions using local averages of USANs of a pair of oriented crosses, and defines corners as points with multiple dominant directions.

Self-similarity can be measured using a circle instead of a disk [78]. The SSD between the center pixel and the pixels at either end of a diameter line is an oriented measure of self-dissimilarity. If this is small in any orientation, then the point is not a corner. This is computationally efficient since the process can be stopped as soon as one small value is encountered. This detector is also used by Lepetit and Fua [79] with the additional step that the difference between the center pixel and circle pixels is used to estimate the Laplacian, and points are also required to be locally maximal in the Laplacian.

Small regions with a large range in grayvalues can be used as corners. To find these efficiently, the image can be projected on to the x and y axes and large peaks found in the second derivatives. Candidate corner locations are the intersections of these maxima projected back into the image [80]. Paler et al. [81] proposes self-similarity can be measured by comparing the center pixel of a window to the median value of pixels in the window. In practice, several percentile values (as opposed to just the $50\mathrm{th}$) are used.

Self-dissimilar patches will have a high energy content. Composing two orthogonal quadrature pair Gabor filters gives oriented energy. Corners are maxima of total energy (the sum of oriented energy over a number of directions) [82].

A fast radial symmetry transform is developed in [83] 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 detected points have some resemblance to DoG features.

Machine learning-based detectors. All of the detectors described above define corners using a model or algorithm and apply that algorithm directly to the image. An alternative is to train a classifier on the model and then apply the classifier to the image. For instance, a multilayer perception can be trained on example corners from some model and applied to the image after some processing [84], [85].

Human perception can be used instead of a model [86]: Images are shown to a number of test subjects. Image locations which are consistently fixated on (as measured by an eye tracking system) are taken to be interesting, and a support vector machine is trained to recognize these points.

If a classifier is used, then it can be trained according to how a corner should behave, i.e., that its performance in a system for evaluating detectors should be maximized. Trujillo and Olague [87] state that detected points should have a high repeatability (as defined by [1]), be scattered uniformly across the image, and that there should be at least as many points detected as requested. A corner detector function is optimized (using genetic programming) to maximize the score based on these measures.

The FAST-n detector (described in Section 3) is related to the wedge model style of detector evaluated using a circle surrounding the candidate pixel. To optimize the detector for speed, this model is used to train a decision tree classifier and the classifier is applied to the image. The FAST-ER detector (described in Section 5) is a generalization which allows the detector to be optimized for repeatability.

2.2 Comparison of Feature Detectors

Considerably less work has been done on comparison and evaluation of feature detectors than on inventing new detectors. The tests fall into three broad categories.[1]

1. *Corner detection as object recognition.* Since there is no good definition of exactly what a corner should look like, algorithms can be compared using simplistic test images where the performance is evaluated (in terms of true positives, false positives, etc.) as the image is altered using contrast reduction, warps, and added noise. Since a synthetic image is used, corners exist only at known locations, so the existence of false negatives and false positives is well defined. However, the method and results do not generalize to natural images.
2. *System performance.* The performance of an application (often tracking) is evaluated as the corner detector is changed. The advantage is that it tests the suitability of detected corners for further processing. However, poor results would be obtained from a detector ill matched to the downstream processing. Furthermore, the results do not necessarily generalize well to other systems. To counter this, sometimes part of a system is used, though, in this case, the results do not necessarily apply to any system.
3. *Repeatability.* This tests whether corners are detected from multiple views. It is a low-level measure of corner detector quality and provides an upper bound on performance. Since it is independent of downstream processing, the results are widely applicable, but it is possible that the detected features may not be useful. Care must be used in this technique, since the trivial detector which identifies every pixel as a corner achieves 100 percent repeatability. Furthermore, the repeatability does not provide information about the usefulness of the detected corners for further processing. For instance, the brightest pixels in the image are likely to be repeatable but not especially useful.

In the first category, Rajan and Davidson [88] produce a number of elementary test images with a very small number of corners (one to four) to test the performance of detectors as various parameters are varied. The parameters are corner angle, corner arm length, corner adjacency, corner sharpness,

1. Tests for the localization accuracy are not considered here since, for most applications, the presence or absence of useful corners is the limiting factor.

contrast, and additive noise. The positions of detected corners are tabulated against the actual corner positions as the parameters are varied. Cooper et al. [34], [89] use a synthetic test image consisting of regions of uniform intensity arranged to create L, T, Y, and X-shaped corners. The pattern is repeated several times with decreasing contrast. Finally, the image is blurred and Gaussian noise is added. Chen and Rockett [85] use a related method. A known test pattern is subjected to a number of random affine warps and contrast changes. They note that this is naive, but tractable. They also provide an equivalent to the Receiver Operating Characteristic (ROC) curve. Zhang et al. [90] generate random corners according to their model and plot localization error, false positive rate, and false negative rate against the detector and generated corner parameters. Luo et al. [43] use an image of a carefully constructed scene and plot the proportion of true positives as the scale is varied and noise is added for various corner angles.

Mohanna and Mokhtarian [91] evaluate performance using several criteria. First, they define a *consistent* detector as one where the number of detected corners does not vary with various transforms such as addition of noise and affine warping. This is measured by the "consistency of corner numbers" (CCN),

$$CCN = 100 \times 1.1^{-|n_t - n_o|}, \tag{6}$$

where n_t is the number of features in the transformed image and n_o is the number of features in the original image. This test does not determine the quality of the detected corners in any way, so they also propose measuring the accuracy (ACU) as

$$ACU = 100 \times \frac{\frac{n_a}{n_o} + \frac{n_a}{n_g}}{2}, \tag{7}$$

where n_o is the number of detected corners, n_g is the number of so-called "ground truth" corners, and n_a is the number of detected corners which are close to ground truth corners. Since real images are used, there is no good definition of ground truth, so a number of human test subjects (e.g., 10) familiar with corner detection, in general, but not the methods under test, label corners in the test images. Corners which 70 percent of the test subjects agree on are kept as ground truth corners. This method unfortunately relies on subjective decisions.

Remarkably, of the systems above, only [85], [88], and [86] provide ROC curves (or equivalent): Otherwise, only a single point (without consistency on either axis of the ROC graph) is measured.

In the second category, Trajković and Hedley [78] define stability as the number of "strong" matches, matches detected over three frames in their tracking algorithm, divided by the total number of corners. Tissainayagam and Suter [92] use a similar method: A corner in frame n is stable if it has been successfully tracked from frame 1 to frame n. Bae et al. [77] detect optical flow using cross correlation to match corners between frames and compare the number of matched corners in each frame to the number of corners in the first frame.

To get more general results than provided by system performance, the performance can be computed using only

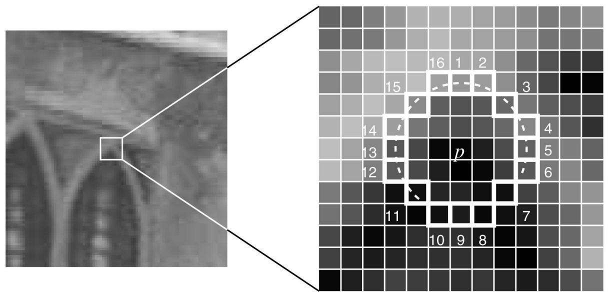


Fig. 1. Twelve-point segment test corner detection in an image patch. The highlighted squares are the pixels used in the corner detection. The pixel at p is the center of a candidate corner. The arc is indicated by the dashed line passing through 12 contiguous pixels which are brighter than p by more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

one part of a system. For instance, Mikolajczyk and Schmid [93] test a large number of interest point descriptors and a small number of closely related detectors by computing how accurately interest point matching can be performed. Moreels and Perona [94] perform detection and matching experiments across a variety of scene types under a variety of lighting conditions. Their results illustrate the difficulties in generalizing from system performance since the best detector varies with both the choice of descriptor and lighting conditions.

In the third category, Schmid et al. [1] propose that, when measuring reliability, the important factor is whether the same real-world features are detected from multiple views. For an image pair, a feature is "detected" if it 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 to detected features. They perform the tests on images of planar scenes so that the relationship between point positions is a homography. Fiducial markers are projected onto the planar scene using an overhead projector to allow accurate computation of the homography. To measure the suitability of interest points for further processing, the information content of descriptors of patches surrounding detected points is also computed.

3 HIGH-SPEED CORNER DETECTION

3.1 FAST: Features from Accelerated Segment Test

The segment test criterion operates by considering a circle of 16 pixels around the corner candidate p. The original detector [95], [96] 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 Fig. 1. n was originally chosen to be 12 because it admits a high-speed test which can be used to exclude a very large number of noncorners. The high-speed test examines pixels 1 and 9. If both of these are within t of I_p, then p cannot be a corner. If p can still be a corner, pixels 5 and 13 are examined. 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. This high-speed test does not reject as many candidates for $n < 12$ since the point can be a corner if only two out of the four pixels are both significantly brighter or both significantly darker than p (assuming the pixels are adjacent). Additional tests are also required to find if the complete test needs to be performed for a bright ring or a dark ring.
2. The efficiency of the detector will depend on the ordering of the questions and the distribution of corner appearances. It is unlikely that this choice of pixels is optimal.
3. Multiple features are detected adjacent to one another.

3.2 Improving Generality and Speed with Machine Learning

Here, we expand on the work first presented in [97] and present an approach which uses machine learning to address the first two points (the third is addressed in Section 3.3). The process operates in two stages. First, to build a corner detector for a given n, all of the 16 pixel rings are extracted from a set of images (preferably from the target application domain). These are labeled using a straightforward implementation of the segment test criterion for n and a convenient threshold.

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

$$S_{p\rightarrow x} = \begin{cases} d, & I_{p\rightarrow x} \leq I_p - t \quad \text{(darker)} \\ s, & I_p - t < I_{p\rightarrow x} < I_p + t \quad \text{(similar)} \\ b, & I_p + t \leq I_{p\rightarrow x} \quad \text{(brighter)} \end{cases} \tag{8}$$

Let P be the set of all pixels in all training images. Choosing an x partitions P into three subsets, P_d, P_s, and P_b, where

$$P_b = \{p \in P : S_{p\rightarrow x} = b\}, \tag{9}$$

and P_d and P_s are defined similarly. In other words, a given choice of x is used to partition the data into three sets. The set P_d contains all points where pixel x is darker than the center pixel by a threshold t, P_b contains points brighter than the center pixel by t, and P_s contains the remaining points where pixel x is similar to the center pixel.

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 [98] 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 total entropy of K for an arbitrary set of corners Q is

$$H(Q) = (c + \bar{c})\log_2(c + \bar{c}) - c\log_2 c - \bar{c}\log_2 \bar{c}, \tag{10}$$

$$\begin{aligned} \text{where} \quad c &= \left|\{i \in Q : K_i \text{ is true}\}\right| \quad \text{(number of corners)} \\ \text{and} \quad \bar{c} &= \left|\{i \in Q : K_i \text{ is false}\}\right| \quad \text{(number of noncorners).} \end{aligned}$$

The choice of x then yields the information gain (H_g)

$$H_g = H(P) - H(P_d) - H(P_s) - H(P_b). \tag{11}$$

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 into $P_{b,d}$, $P_{b,s}$, $P_{b,b}$, x_s is selected to partition P_s into $P_{s,d}$, $P_{s,s}$, $P_{s,b}$, and so on, where each x is chosen to yield maximum information about the set it is applied to. The recursion 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 noncorners. This is guaranteed to occur since K is an exact function of the data. In summary, this procedure 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.

In some cases, two of the three subtrees may be the same. In this case, the Boolean test which separates them is removed. This decision tree is then converted into C code, creating a long string of nested if-else statements which is compiled and used as a corner detector. For highest speed operation, the code is compiled using profile-guided optimizations which allow branch prediction and block reordering optimizations.

For further optimization, we force x_b, x_d, and x_s to be equal. In this case, the second pixel tested is always the same. Since this is the case, the test against the first and second pixels can be performed in batch. This allows the first two tests to be performed in parallel for a strip of pixels using the vectorizing instructions present on many high-performance microprocessors. Since most points are rejected after two tests, this leads to a significant speed increase.

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. In the case of the FAST-n detectors, it is straightforward to include an instance of every possible combination of pixels (there are $3^{16} = 43{,}046{,}721$ combinations) with a low weight to ensure that the learned detector exactly computes the segment test criterion.

3.3 Nonmaximal Suppression

Since the segment test does not compute a corner response function, nonmaximal suppression cannot be applied directly to the resulting features. For a given n, as t is increased, the number of detected corners will decrease. Since $n = 9$ produces the best repeatability results (see Section 6), variations in n will not be considered. The corner strength is, therefore, defined to be the maximum value of t for which a point is detected as a corner.

The decision tree classifier can efficiently determine the class of a pixel for a given value of t. The class of a pixel (for example, 1 for a corner, 0 for a noncorner) is a monotonically decreasing function of t. Therefore, we can use bisection to efficiently find the point where the function changes from 1 to 0. This point gives us the largest value of t for which the point is detected as a corner. Since t is discrete, this is the binary search algorithm.

Alternatively, an iteration scheme can be used. A pixel on the ring "passes" the segment test if it is not within t of the center. If enough pixels fail, then the point will not be classified as a corner. The detector is run and, of all the pixels which pass the test, the *amount* by which they pass is found. The threshold is then increased by the smallest of these amounts, and the detector is rerun. This increases the threshold just enough to ensure that a different path is

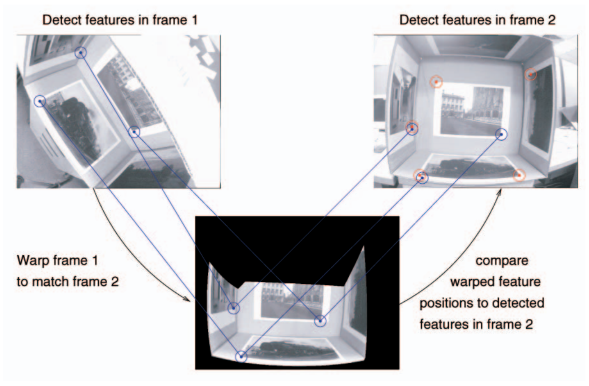

Detect features in frame 1
Detect features in frame 2
Warp frame 1 to match frame 2
compare warped feature positions to detected features in frame 2

Fig. 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.

taken through the tree. This process is then iterated until detection fails.

Because the speed depends strongly on the learned tree and the specific processor architecture, neither technique has a definitive speed advantage over the other. Nonmaximal suppression is performed in a 3×3 mask.

4 Measuring Detector Repeatability

For an image pair, a feature is "useful" if it is extracted in one image and can potentially appear in the second (i.e., it is not occluded). It is "repeated" if it is also detected nearby the same real-world point in the second. For the purposes of measuring repeatability, this allows several features in the first image to match a single feature in the second image. The repeatability R is defined to be

$$R = \frac{N_{\text{repeated}}}{N_{\text{useful}}}, \tag{12}$$

where N_{repeated} and N_{useful} are summed over all image pairs in an image sequence. This is equivalent to the weighted average of the repeatabilities for each image pair, where the weighting is the number of useful features. In this paper, we generally compute the repeatability for a given number of features per frame, varying between zero and 2,000 features (for a 640×480 image). This also allows us to compute the area under the repeatability curve A as an aggregate score.

The repeatability measurement requires the location and visibility of every pixel in the first image to be known in the second image. In order to compute this, we use a 3D surface model of the scene to compute if and where detected features should appear in other views. This is illustrated in Fig. 2. This allows the repeatability of the detectors to be analyzed on features caused by geometry such as corners of polyhedra, occlusions, and junctions. We also allow bas-relief textures to be modeled with a flat plane so that the repeatability can be tested under nonaffine warping.

The definition of "nearby" above must allow a small margin of error (ε pixels) because the alignment, the 3D model, and the camera calibration (especially the radial distortion) are not perfect. Furthermore, 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 optimized using a blend of simulated annealing and gradient descent to minimize 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 1,000 random locations.

The data sets used are shown in Figs. 3, 4, and 5. With these data sets, we have tried to capture a wide range of geometric and textural corner types.

5 FAST-ER: Enhanced Repeatability

Since the segment test detector can be represented as a ternary decision tree and we have defined repeatability, the detector can be generalized by defining a feature detector to be a ternary decision tree which detects points with high

Fig. 3. Box data set: 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 texture.

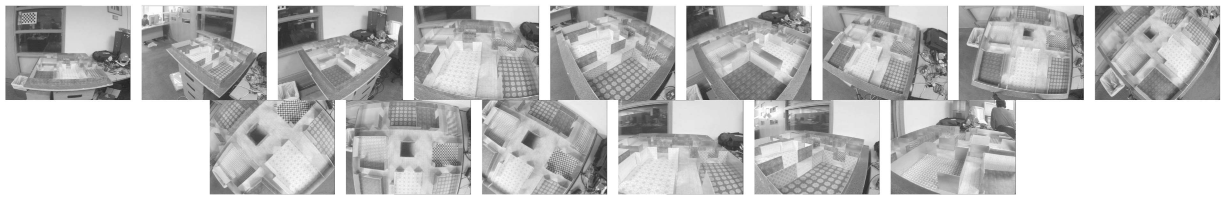


Fig. 4. Maze data set: Photographs taken of a prop used in an augmented reality application. This set consists of textural features undergoing projective warps as well as geometric features. There are also significant changes of scale.

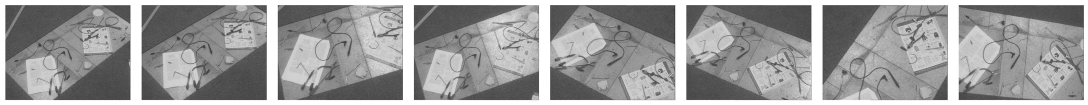


Fig. 5. Bas-relief data set: The model is a flat plane, but there are many objects with significant relief. This causes the appearance of features to change in a nonaffine way from different viewpoints.

repeatability. The repeatability of such a detector is a nonconvex function of the configuration of the tree, so we optimize the tree using simulated annealing. This results in a multiobjective optimization. If every point is detected as a feature, then the repeatability is trivially perfect. Also, if the tree complexity is allowed to grow without bound, then the optimization is quite capable of finding one single feature in each image in the training set which happens to be repeated. Neither of these are useful results. To account for this, the cost function for the tree is defined to be

$$k = \left(1 + \left(\frac{w_r}{r}\right)^2\right)\left(1 + \frac{1}{N}\sum_{i=1}^{N}\left(\frac{d_i}{w_n}\right)^2\right)\left(1 + \left(\frac{s}{w_s}\right)^2\right), \quad (13)$$

where r is the repeatability (as defined in (12)), d_i is the number of detected corners in frame i, N is the number of frames, and s is the size (number of nodes) of the decision tree. The effect of these costs are controlled by w_r, w_n, and w_s. Note that, for efficiency, repeatability is computed at a fixed threshold as opposed to a fixed number of features per frame.

The corner detector should be invariant to rotation, reflection, and intensity inversion of the image. To prevent excessive burden on the optimization algorithm, each time the tree is evaluated, it is applied 16 times: at four rotations, 90 degrees apart, with all combinations of reflection and intensity inversion. The result is the logical OR of the detector applications: A corner is detected if any one of the 16 applications of the tree classifies the point as a corner.

Each node of the tree has an offset relative to the center pixel, x, with $x \in \{0 \dots 47\}$, as defined in Fig. 6. Therefore, $x = 0$ refers to the offset $(-1, 4)$. Each leaf has a class K, with 0 for noncorners and 1 for corners. Apart from the root node, each node is either on a b, d, or s branch of its parent, depending on the test outcome which leads to that branch. The tree is constrained so that each leaf on an s branch of its direct parent has $K = 0$. This ensures that the number of corners generally decreases as the threshold is increased.

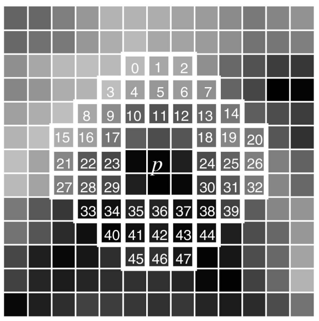


Fig. 6. Positions of offsets used in the FAST-ER detector.

The simulated annealing optimizer makes random modifications to the tree by first selecting a node at random, and then, mutating it. If the selected node is:

- a leaf, then, with equal probability, either:
  - Replace node with a random subtree of depth 1.
  - Flip classification of node. This choice is not available if the leaf class is constrained.
- a node, then, with equal probability, choose any one of:
  - Replace the offset with a random value in $0 \dots 47$.
  - Replace the node with a leaf with a random class (subject to the constraint).
  - Remove a randomly selected branch of the node and replace it with a copy of another randomly selected branch of that node. For example, a b branch may be replaced with a copy of an s branch.

The randomly grown subtree consists of a single decision node (with a random offset in $0 \dots 47$), and three leaf nodes. With the exception of the constrained leaf, the leaves of this random subtree have random classes. These modifications to the tree allow growing, mutation, and mutation and shrinking of the tree, respectively. The last modification of the tree is motivated by our observations of the FAST-9 detector. In FAST-9, a large number of nodes have the characteristic that two out of the three subtrees are identical. Since FAST-9 exhibits high repeatability, we have included this modification to allow FAST-ER to easily learn a similar structure.

The modifications are accepted according to the Boltzmann acceptance criterion, where the probability P of accepting a change at iteration I is

$$P = e^{\frac{\hat{k}_{I-1} - k_I}{T}}, \quad (14)$$

where $\hat{k}$ is the cost after application of the acceptance criterion and T is the temperature. The temperature follows an exponential schedule

$$T = \beta e^{-\alpha \frac{I}{I_{\max}}}, \quad (15)$$

where $I_{\max}$ is the number of iterations. The algorithm is initialized with a randomly grown tree of depth 1, and the algorithm uses a fixed threshold t. Instead of performing a single optimization, the optimizer is rerun a number of times using different random seeds.

Because the detector must be applied to the images every iteration, each candidate tree in all 16 transformations is compiled to machine code in memory and executed directly. Since it is applied with 16 transformations, the resulting detector is not especially efficient. So, for efficiency, the detector is used to generate training data so that a single tree can be generated using the method

TABLE 1
Parameters Used to Optimize the Tree

Parameter	Value
w_r	1
w_n	3,500
w_s	10,000
α	30
β	100
t	35
$I_{\max}$	100,000
Runs	100
ε	5 pixels
Training set	'box' set, images 0–2.

described in Section 3.2. The resulting tree contains approximately 30,000 nonleaf nodes.

5.1 Parameters and Justification

The parameters used for training are given in Table 1. The entire optimization, which consists of 100 repeats of a 100,000 iteration optimization, requires about 200 hours on a Pentium 4 at 3 GHz. Finding the optimal set of parameters is essentially a high-dimensional optimization problem, with many local optima. Furthermore, each evaluation of the cost function is very expensive. Therefore, the values are in no sense optimal, but they are a set of values which produce good results. Refer to [99] for techniques for choosing parameters of a simulated annealing-based optimizer. Recall that the training set consists of only the first three images from the "box" data set.

The weights determine the relative effects of good repeatability, resistance to overfitting and corner density, and therefore, will affect the performance of the resulting corner detector. To demonstrate the sensitivity of the detector with respect to w_r, w_n, and w_s, a detector was learned for three different values of each, $w_r \in \{0.5, 1, 2\}$, $w_n \in \{1{,}750,\ 3{,}500,\ 7{,}000\}$, and $w_s \in \{5{,}000,\ 10{,}000,\ 20{,}000\}$, resulting in a total of 27 parameter combinations. The performance of the detectors is evaluated by computing the mean area under the repeatability curve for the "box," "maze," and "bas-relief" data sets. Since, in each of the 27 points, 100 runs of the optimization are performed, each of the 27 points produces a distribution of scores. The results of this are shown in Fig. 7. The variation in score with respect to the parameters is quite low even though the parameters all vary by a factor of four. Given that, the results for the set of parameters in Table 1 are very close to the results for the best tested set of parameters. This demonstrates that the choices given in Table 1 are reasonable.

6 Results

In this section, the FAST and FAST-ER detectors are compared against a variety of other detectors both in terms of repeatability and speed. In order to test the detectors

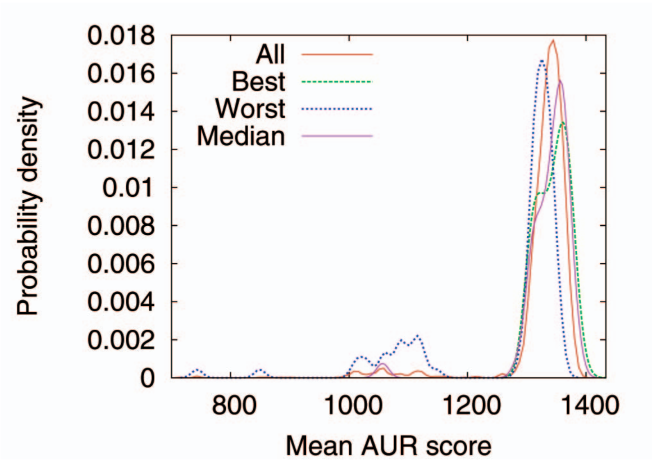


Fig. 7. Distribution of scores for various parameters of (w_r, w_n, w_s). The parameters leading to the best result are $(2.0, 3{,}500, 5{,}000)$ and the parameters for the worst point are $(0.5, 3{,}500, 5{,}000)$. For comparison, the distribution for all 27 runs and the median point (given in Table 1) are given. The score given is the mean value of A computed over the "box," "maze," and "bas-relief" data sets.

further, we have used the "Oxford" data set [100] in addition to our own. This data set models the warp between images using a homography and consists of eight sequences of six images each. It tests detector repeatability under viewpoint changes (for approximately planar scenes), lighting changes, blur, and JPEG compression. Note that the FAST-ER detector is trained on three images (six image pairs) and is tested on a total of 85 images (688 image pairs).

The parameters used in the various detectors are given in Table 2. In all cases (except SUSAN, which uses the reference implementation in [101]), nonmaximal suppression is performed using a 3×3 mask. The number of features was controlled in a manner equivalent to thresholding on the response. For the Harris-Laplace detector, the Harris response was used, and for the SUSAN detector, the "distance threshold" parameter was used. It should be noted that some experimentation was performed on all the detectors to find the best results on our data set. In the case of FAST-ER, the best detector was selected. The parameters were then used without modification on the "Oxford" data

TABLE 2
Parameters Used for Testing Corner Detectors

DoG		SUSAN	
Scales per octave	3	Distance threshld	4.0
Initial blur σ	0.8		
Octaves	4	**Harris-Laplace**	
		Initial blur σ	0.8
Harris, Shi-Tomasi		Harris blur	3
Blur σ	2.5	Octaves	4
		Scales per octave	10
General parameters			
ε	5 pixels		

TABLE 3
Area under Repeatability Curves for 0-2,000 Corners per Frame Averaged over All the Evaluation Data Sets (Except the Additive Noise)

Detector	A
FAST-ER	1313.6
FAST-9	1304.57
DoG	1275.59
Shi & Tomasi	1219.08
Harris	1195.2
Harris-Laplace	1153.13
FAST-12	1121.53
SUSAN	1116.79
Random	271.73

set. The timing results were obtained with the same parameters used in the repeatability experiment.

6.1 Repeatability

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 percent. To test robustness to image noise, increasing amounts of Gaussian noise were added to the bas-relief data set, in addition to the significant amounts of camera noise already present. Aggregate results taken over all data sets are given in Table 3. It can be seen from this that, on average, FAST-ER outperforms all the other tested detectors.

More details are shown in Figs. 8, 9, 10, and 11. As shown in Fig. 8, FAST-9 performs best (FAST-8 and below are edge detectors), so only FAST-9 and FAST-12 (the original FAST detector) are given.

The FAST-9 feature detector, despite being designed only for speed, generally outperforms all but FAST-ER on these images. FAST-n, however, is not very robust to the presence of noise. This is to be expected. High speed is achieved by analyzing the fewest pixels possible, so the detector's ability to average out noise is reduced.

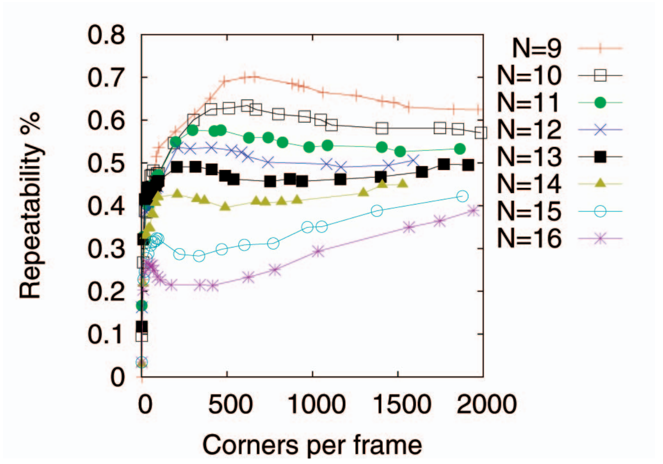


Fig. 8. A comparison of the FAST-n detectors on the "bas-relief" shows that $n = 9$ is the most repeatable. For $n \leq 8$, the detector starts to respond strongly to edges.

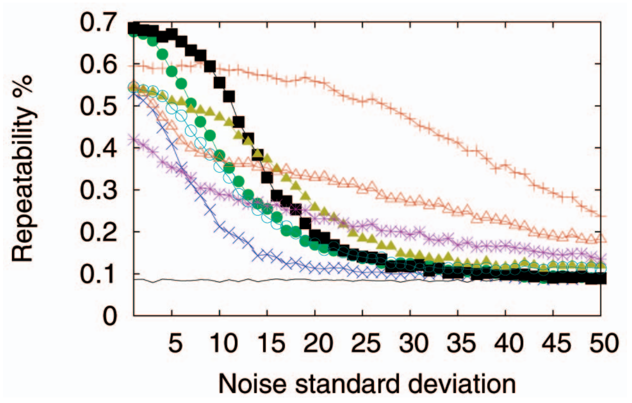


Fig. 9. Repeatability results for the bas-relief data set (at 500 features per frame) as the amount of Gaussian noise added to the images is varied. See Fig. 10 for the key.

The best repeatability results are achieved by FAST-ER. FAST-ER easily outperforms FAST-9 in all but Figs. 10a, 11b, 11c, and 11e. These results are slightly more mixed, but FAST-ER still performs very well for higher corner densities. FAST-ER greatly outperforms FAST-9 on the noise test (and outperforms all other detectors for $\sigma < 7$). This is because the training parameters bias the detector toward detecting more corners for a given threshold than FAST-9. Consequently, for a given number of features per frame, the threshold is higher, so the effect of noise will be reduced.

As the number of corners per frame is increased, all of the detectors, at some point, suffer from decreasing repeatability. This effect is least pronounced with the FAST-ER detector. Therefore, with FAST-ER, the corner density does not need to be as carefully chosen as with the other detectors. This falloff is particularly strong in the Harris and Shi-Tomasi detectors. Shi and Tomasi, derive their result for better feature detection on the assumption that the deformation of the features is affine. Their detector performs slightly better overall, and especially, in the cases where the deformations are largely affine. For instance, in the bas-relief data set (Fig. 10c), this assumption does not hold, and interestingly, the Harris detector outperforms Shi and Tomasi detector in this case. Both of these detectors tend to outperform all others on repeatability for very low corner densities (less than 100 corners per frame).

The Harris-Laplace is a detector that was originally evaluated using planar scenes [60], [102]. The results show that Harris-Laplace points outperform both DoG points and Harris points in repeatability. For the box data set, our results verify that this is correct for up to about 1,000 points per frame (typical numbers, probably commonly used); the results are somewhat less convincing in the other data sets, where points undergo nonprojective changes.

In the sample implementation of SIFT [103], approximately 1,000 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.

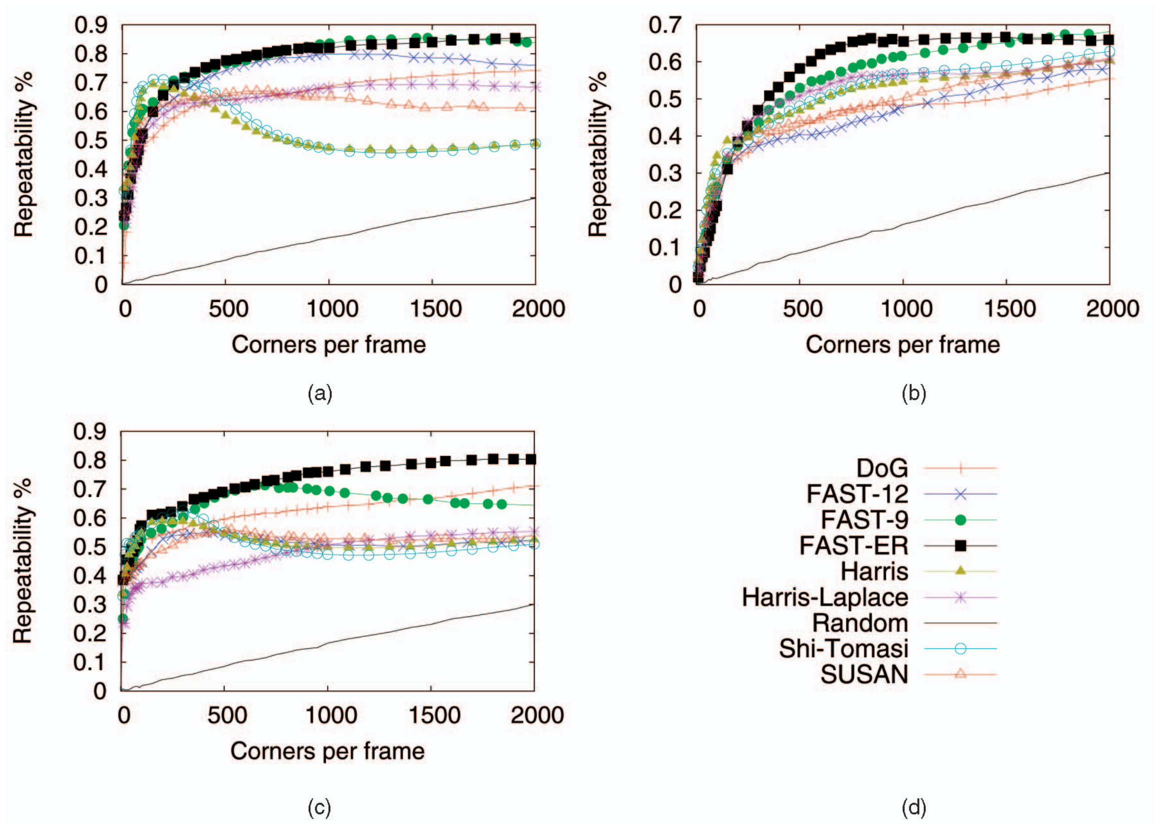


Fig. 10. (a), (b), (c) Repeatability results for the repeatability data set as the number of features per frame is varied. (d) Key for this figure, Figs. 11 and 9. For FAST and SUSAN, the number of features cannot be chosen arbitrarily; the closest approximation to 500 features in each frame is used. (a) Box data set. (b) Maze data set. (c) Bas-relief data set.

Smith and Brady [76] 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 that the performance drops quite rapidly with increasing noise to start with, it soon levels off and outperforms all but the DoG detector. The DoG detector is remarkably robust to the presence of noise. Convolution is linear, so the computation of DoG is equivalent to convolution with a DoG kernel. Since this kernel is symmetric, the convolution 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 [104].

6.2 Speed

Timing tests were performed on a 3.0 GHz Pentium 4D which is representative of a modern desktop computer. The timing tests are performed on two data sets: the test set and the training set. The training set consists of 101 monochrome fields from a high-definition video source with a resolution of 992×668 pixels. This video source is used to train the high-speed FAST detectors and for profile-guided optimizations for all the detectors. The test set consists of 4,968 frames of monochrome 352×288 (quarter-PAL) video.

The learned FAST-ER, FAST-9, and FAST-12 detectors have been compared to the original FAST-12 detector, to our implementation of the Harris and DoG (the detector used by SIFT), and to the reference implementation of SUSAN [101]. The FAST-9, Harris, and DoG detectors use the SSE-2 vectorizing instructions to speed up the processing. The learned FAST-12 does not since using SSE-2 does not yield a speed increase.

TABLE 4
Timing Results for a Selection of Feature Detectors Run on Frames of Two Video Sequences

Detector	Training set		Test set	
	Pixel rate (MPix/s)	%	MPix/s	%
FAST $n = 9$	188	4.90	179	5.15
FAST $n = 12$	158	5.88	154	5.98
Original FAST ($n = 12$)	79.0	11.7	82.2	11.2
FAST-ER	75.4	12.2	67.5	13.7
SUSAN	12.3	74.7	13.6	67.9
Harris	8.05	115	7.90	117
Shi-Tomasi	6.50	142	6.50	142
DoG	4.72	195	5.10	179

The percentage of the processing budget for 640×480 *video is given for comparison. Note that, since PAL, NTSC, DV, and 30 Hz VGA (common for webcams) video have approximately the same pixel rate, the percentages are widely applicable. The feature density is equivalent to approximately 500 features per* 640×480 *frame. The results shown include the time taken for nonmaximal suppression.*

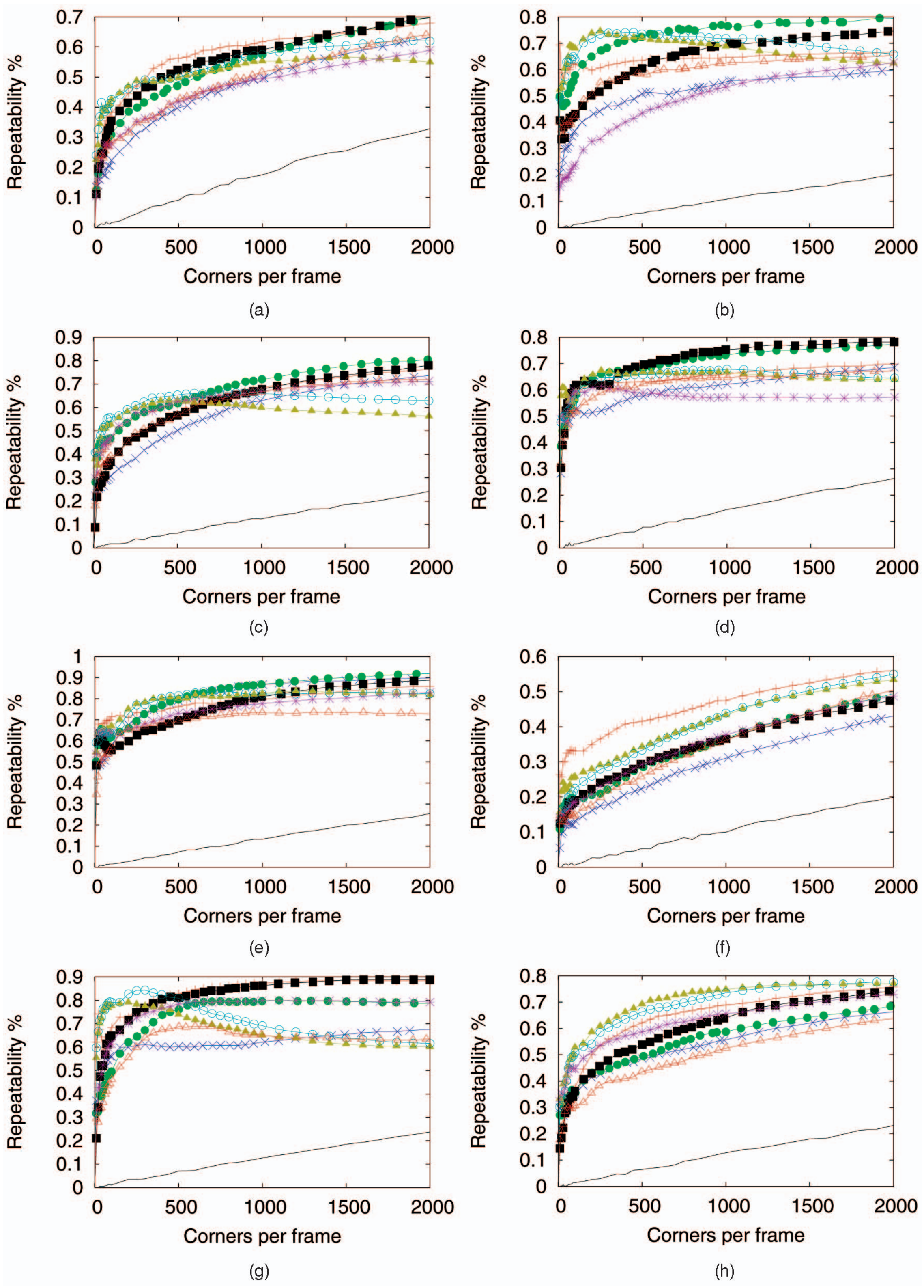


Fig. 11. (a), (b), (c), (d), (e), (f), (g) Repeatability results for the "Oxford" data set as the number of features per frame is varied. See Fig. 10 for the key. (a) Bark data set. (b) Bikes data set. (c) Boat data set. (d) Graffiti data set. (e) Leuven data set. (f) Trees data set. (g) UBC data set. (h) Wall data set.

As can be seen in Table 4, FAST, in general, is much faster than the other tested feature detectors, and the learned FAST is roughly twice as fast as the handwritten version. In addition, it is also able to generate an efficient detector for FAST-9, which is the most reliable of the FAST-n detectors. Furthermore, it is able to generate a very efficient detector for FAST-ER. Despite the increased complexity of this detector, it is still much faster than all but FAST-n. On modern hardware, FAST and FAST-ER consume 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.

7 Conclusions

In this paper, we have presented the FAST family of detectors. Using machine learning, we turned the simple and very repeatable segment test heuristic into the FAST-9 detector which has unmatched processing speed. Despite the design for speed, the resulting detector has excellent repeatability. By generalizing the detector and removing preconceived ideas about how a corner should appear, we were able to optimize a detector directly to improve its repeatability, creating the FAST-ER detector. While still being very efficient, FAST-ER has dramatic improvements in repeatability over FAST-9 (especially in noisy images). The result is a detector which is not only computationally efficient, but has better repeatability results and is more consistent with variation in corner density than any other tested detector.

These results raise an interesting point about corner detection techniques: Too much reliance on intuition can be misleading. Here, rather than concentrating on how the algorithm should do its job, we focus our attention on what performance measure we want to optimize and this yields very good results. The result is a detector which compares favorably to existing detectors.

In the interests of science, we will be making all parts of the experiment freely available as supplemental material,[2] which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPAMI.2008.275, including the data sets, the FAST-ER learning code, and the resulting trees.

References

[1] C. Schmid, R. Mohr, and C. Bauckhage, "Evaluation of Interest Point Detectors," *Int'l J. Computer Vision,* vol. 37, no. 2, pp. 151-172, 2000.

[2] A. Rosenfeld and E. Johnston, "Angle Detection on Digital Curves," *IEEE Trans. Computers,* vol. 22, no. 9, pp. 875-878, Sept. 1973.

[3] A. Rosenfeld and J.S. Weszka, "An Improved Method of Angle Detection on Digital Curves," *IEEE Trans. Computers,* vol. 24, no. 9, pp. 940-941, Sept. 1975.

[4] H. Freeman and L.S. Davis, "A Corner-Finding Algorithm for Chain-Coded Curves," *IEEE Trans. Computers,* vol. 26, no. 3, pp. 297-303, Mar. 1977.

[5] H.L. Beus and S.S.H. Tiu, "An Improved Corner Detection Algorithm Based on Chain-Coded Plane Curves," *Pattern Recognition,* vol. 20, no. 3, pp. 291-296, 1987.

[6] L. O'Gorman, "Curvilinear Feature Detection from Curvature Estimation," *Proc. Ninth Int'l Conf. Pattern Recognition,* pp. 1116-1119, 1988.

[7] C.-H. Teh and R. Chin, "On the Detection of Dominant Points on Digital Curves," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 11, no. 8, pp. 859-872, Aug. 1989.

2. FAST-n detectors are also available in libCVD from: http://savannah.nongnu.org/projects/libcvd Experimental material is also available from: http://mi.eng.cam.ac.uk/~er258/work/fast.html.

[8] H. Ogawa, "Corner Detection on Digital Curves Based on Local Symmetry of the Shape," *Pattern Recognition,* vol. 22, no. 4, pp. 351-357, 1989.

[9] A. Bandera, C. Urdiales, F. Arrebola, and E. Sandoval, "Corner Detection by Means of Adaptively Estimated Curvature Function," *Electronics Letters,* vol. 36, no. 2, pp. 124-126, 2000.

[10] C. Urdiales, C. Trazegnies, A. Bandera, and E. Sandoval, "Corner Detection Based on Adaptively Filtered Curvature Function," *Electronics Letters,* vol. 32, no. 5, pp. 426-428, 2003.

[11] K. Sohn, W.E. Alexander, J.H. Kim, Y. Kim, and W.E. Snyder, "Curvature Estimation and Unique Corner Point Detection for Boundary Representation," *Proc. IEEE Int'l Conf. Robotics and Automation,* vol. 2, pp. 1590-1595, 1992.

[12] X. He and N. Yung, "Curvature Scale Space Corner Detector with Adaptive Threshold and Dynamic Region of Support," *Proc. 17th Int'l Conf. Pattern Recognition,* pp. 791-794, 2004.

[13] N. Ansari and E.J. Delp, "On Detecting Dominant Points," *Pattern Recognition,* vol. 24, no. 5, pp. 441-451, 1991.

[14] A. Rattarangsi and R.T. Chin, "Scale-Based Detection of Corners of Planar Curves," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 14, no. 4, pp. 430-449, Apr. 1992.

[15] J. Lee, Y. Sun, and C. Chen, "Wavelet Transform for Corner Detection," *Proc. IEEE Conf. Systems Eng.,* 1992.

[16] J.-S. Lee, Y.-N. Sun, and C.-H. Chen, "Multiscale Corner Detection by Using Wavelet Transform," *IEEE Trans. Image Processing,* vol. 4, no. 1, pp. 100-104, Jan. 1995.

[17] A. Quddus and M. Fahmy, "Fast Wavelet-Based Corner Detection Technique," *Electronics Letters,* vol. 35, no. 4, pp. 287-288, 1999.

[18] F. Mokhtarian and R. Suomela, "Robust Image Corner Detection through Curvature Scale Space," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 20, no. 12, pp. 1376-1381, Dec. 1998.

[19] P. Saint-Marc, J.-S. Chen, and G. Medioni, "Adaptive Smoothing: A General Tool for Early Vision," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 13, no. 6, pp. 514-529, June 1991.

[20] B.K. Ray and R. Pandyan, "Acord—An Adaptive Corner Detector for Planar Curves," *Pattern Recognition Letters,* vol. 36, no. 3, pp. 703-708, 2003.

[21] D.J. Langridge, "Curve Encoding and Detection of Discontinuities," *Computer Vision, Graphics, and Image Processing,* vol. 20, no. 1, pp. 58-71, 1987.

[22] G. Medioni and Y. Yasumoto, "Corner Detection and Curve Representation Using Cubic B-Splines," *Computer Vision, Graphics, and Image Processing,* vol. 39, no. 3, pp. 279-290, 1987.

[23] D.J. Beymer, "Finding Junctions Using the Image Gradient," *Proc. Sixth IEEE Conf. Computer Vision and Pattern Recognition,* pp. 720-721, 1991.

[24] U. Seeger and R. Seeger, "Fast Corner Detection in Grey-Level Images," *Pattern Recognition Letters,* vol. 15, no. 7, pp. 669-675, 1994.

[25] F. Arrebola, A. Bandera, P. Camacho, and F. Sandoval, "Corner Detection by Local Histograms of Contour Chain Code," *Electronics Letters,* vol. 33, no. 21, pp. 1769-1771, 1997.

[26] F. Arrebola, A. Bandera, P. Camacho, and F. Sandoval, "Corner Detection and Curve Representation by Circular Histograms of Contour Chain Code," *Electronics Letters,* vol. 35, no. 13, pp. 1065-1067, 1999.

[27] L. Li, "Corner Detection and Interpretation on Planar Curves Using Fuzzy Reasoning," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 21, no. 11, pp. 1204-1210, Nov. 1999.

[28] P. Sankar and C. Sharma, "A Parallel Procedure for the Detection of Dominant Points on a Digital Curve," *Computer Graphics and Image Processing,* vol. 7, no. 4, pp. 403-412, 1978.

[29] F.-H. Cheng and W.-H. Hsu, "Parallel Algorithm for Corner Finding on Digital Curves," *Pattern Recognition Letters,* vol. 8, no. 1, pp. 47-53, 1988.

[30] D.H. Ballard, "Generalizing the Hough Transform to Detect Arbitrary Shapes," *Pattern Recognition,* vol. 13, no. 2, pp. 111-122, 1981.

[31] E.R. Davies, "Application of the Generalised Hough Transform to Corner Detection," *Proc. IEE Computers and Digital Techniques,* vol. 135, no. 1, pp. 49-54, 1988.

[32] R.M. Haralick and L.G. Shapiro, *Computer and Robot Vision,* vol. 1, Addison-Wesley, 1993.

[33] R. Mehrotra, S. Nichani, and N. Ranganathan, "Corner Detection," *Pattern Recognition,* vol. 23, no. 11, pp. 1223-1233, 1990.

[34] J. Cooper, S. Venkatesh, and L. Kitchen, "The Dissimilarity Corner Detector," *Proc. Fifth Int'l Conf. Advanced Robotics,* pp. 1377-1382, 1991.

[35] L. Kitchen and A. Rosenfeld, "Gray-Level Corner Detection," *Pattern Recognition Letters,* vol. 1, no. 2, pp. 95-102, 1982.

[36] A. Singh and M. Shneier, "Grey Level Corner Detection: A Generalization and a Robust Real Time Implementation," *Computer Vision, Graphics, and Image Processing,* vol. 51, no. 1, pp. 54-69, 1990.

[37] O. Zuniga and R. Haralick, "Corner Detection Using the Facet Model," *Proc. First IEEE Conf. Computer Vision and Pattern Recognition,* pp. 30-37, 1983.

[38] R. Deriche and G. Giraudon, "A Computational Approach for Corner and Vertex Detection," *Int'l J. Computer Vision,* vol. 10, no. 2, pp. 101-124, 1993.

[39] H. Wang and M. Brady, "Real-Time Corner Detection Algorithm for Motion Estimation." *Image and Vision Computing,* vol. 13, no. 9, pp. 695-703, 1995.

[40] P. Beaudet, "Rotational Invariant Image Operators." *Proc. Fourth Int'l Conf. Pattern Recognition,* pp. 579-583, 1978.

[41] G. Giraudon and R. Deriche, "On Corner and Vertex Detection," *Proc. Sixth IEEE Conf. Computer Vision and Pattern Recognition,* pp. 650-655, 1991.

[42] L. Dreschler and H.-H. Nagel, "Volumetric Model and 3D Trajectory of a Moving Car from Monocular TV Frames Sequence of a Street Scene," *Computer Graphics and Image Processing,* vol. 20, no. 3, pp. 199-228, 1982.

[43] B. Luo, A.D.J. Cross, and E.R. Hancock, "Corner Detection via Topographic Analysis of Vector Potential," *Proc. Ninth British Machine Vision Conf.,* 1998.

[44] H. Moravec, "Obstacle Avoidance and Navigation in the Real World by a Seeing Robot Rover," Technical Report CMU-RI-TR-80-03, Robotics Inst., Carnegie Mellon Univ., 1980, also doctoral dissertation, Stanford Univ., available as Stanford AIM-340, CS-80-813.

[45] C. Harris and M. Stephens, "A Combined Corner and Edge Detector," *Proc. Alvey Vision Conf.,* pp. 147-151, 1988.

[46] E. Rosten, "High Performance Rigid Body Tracking," PhD dissertation, Univ. of Cambridge, Feb. 2006.

[47] W. Förstner, "A Feature-Based Correspondence Algorithm for Image Matching," *Int'l Archive Photogrammetry and Remote Sensing,* vol. 26, pp. 150-166, 1986.

[48] C. Tomasi and T. Kanade, "Detection and Tracking of Point Features," Technical Report CMU-CS-91-132, Carnegie Mellon Univ., 1991.

[49] J. Shi and C. Tomasi, "Good Features to Track," *Proc. Ninth IEEE Conf. Computer Vision and Pattern Recognition,* 1994.

[50] J.A. Noble, "Descriptions of Image Surfaces." PhD dissertation, Dept. of Eng. Science, Univ. of Oxford, 1989.

[51] C.S. Kenney, B.S. Manjunath, M. Zuliani, M.G.A. Hewer, and A.V. Nevel, "A Condition Number for Point Matching with Application to Registration and Postregistration Error Estimation," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 25, no. 11, pp. 1437-1454, Nov. 2003.

[52] M. Zuliani, C. Kenney, and B. Manjunath, "A Mathematical Comparison of Point Detectors," *Proc. Second IEEE Image and Video Registration Workshop,* 2004.

[53] C. Kenney, M. Zuliani, and B. Manjunath, "An Axiomatic Approach to Corner Detection," *Proc. 18th IEEE Conf. Computer Vision and Pattern Recognition,* pp. 191-197, 2005.

[54] K. Rohr, "On 3D Differential Operators for Detecting Point Landmarks," *Image and Vision Computing,* vol. 15, no. 3, pp. 219-233, 1997.

[55] J.A. Noble, "Finding Corners," *Image and Vision Computing,* vol. 6, no. 2, pp. 121-128, 1988.

[56] B. Triggs, "Detecting Keypoints with Stable Position, Orientation and Scale under Illumination Changes," *Proc. Eighth European Conf. Computer Vision,* vol. 4, pp. 100-113, 2004.

[57] K. Mikolajczyk and C. Schmid, "An Affine Invariant Interest Point Detector," *Proc. European Conf. Computer Vision,* pp. 128-142, 2002.

[58] D.G. Lowe, "Distinctive Image Features from Scale Invariant Keypoints," *Int'l J. Computer Vision,* vol. 60, no. 2, pp. 91-110, 2004.

[59] J.L. Crowley, O. Riff, and J.H. Piater, "Fast Computation of Characteristic Scale Using a Half Octave Pyramid," *Proc. Fourth Int'l Conf. Scale-Space Theories in Computer Vision,* 2003.

[60] K. Mikolajczyk and C. Schmid, "Indexing Based on Scale Invariant Interest Points," *Proc. Eighth IEEE Int'l Conf. Computer Vision,* vol. 1, pp. 525-531, 2001.

[61] M. Brown and D.G. Lowe, "Invariant Features from Interest Point Groups." *Proc. 13th British Machine Vision Conf.,* pp. 656-665, 2002.

[62] F. Schaffalitzky and A. Zisserman, "Viewpoint Invariant Texture Matching and Wide Baseline Stereo," *Proc. Eighth IEEE Int'l Conf. Computer Vision,* pp. 636-643, 2001.

[63] F. Schaffalitzky and A. Zisserman, "Multi-View Matching for Unordered Image Sets, or How Do I Organise My Holiday Snaps?" *Proc. Seventh European Conf. Computer Vision,* pp. 414-431, 2002.

[64] A. Guiducci, "Corner Characterization by Differential Geometry Techniques," *Pattern Recognition Letters,* vol. 8, no. 5, pp. 311-318, 1988.

[65] K. Rohr, "Recognizing Corners by Fitting Parametric Models," *Int'l J. Computer Vision,* vol. 9, no. 3, pp. 213-230, 1992.

[66] P.L. Rosin, "Measuring Corner Properties," *Computer Vision and Image Understanding,* vol. 73, no. 2, pp. 291-307, 1999.

[67] J. Canny, "A Computational Approach to Edge Detection," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 8, no. 6, pp. 679-698, Nov. 1986.

[68] K. Rangarajan, M. Shah, and D. van Brackle, "Optimal Corner Detection," *Proc. Second IEEE Int'l Conf. Computer Vision,* pp. 90-94, 1988.

[69] S.-T. Liu and W.-H. Tsai, "Moment-Preserving Corner Detection," *Pattern Recognition,* vol. 23, no. 5, pp. 441-460, 1990.

[70] S. Ghosal and R. Mehrotra, "Zernike Moment-Based Feature Detectors," *Proc. First Int'l Conf. Image Processing,* vol. 1, pp. 934-938, 1994.

[71] F. Shen and H. Wang, "Real Time Gray Level Corner Detector," *Proc. Sixth Int'l Conf. Control, Automation, Robotics, and Vision,* 2000.

[72] R.O. Duda and P.E. Hart, "Use of the Hough Transformation to Detect Lines and Curves in Pictures," *Comm. ACM,* vol. 15, no. 1, pp. 11-15, 1972.

[73] F. Shen and H. Wang, "Corner Detection Based on Modified Hough Transform," *Pattern Recognition Letters,* vol. 32, no. 8, pp. 1039-1049, 2002.

[74] B. Luo and D. Pycock, "Unified Multi-Scale Corner Detection," *Proc. Fourth IASTED Int'l Conf. Visualisation, Imaging and Image Processing,* 2004.

[75] X. Xie, R. Sudhakar, and H. Zhuang, "Corner Detection by a Cost Minimization Approach," *Pattern Recognition,* vol. 26, no. 8, pp. 1235-1243, 1993.

[76] S.M. Smith and J.M. Brady, "SUSAN—A New Approach to Low Level Image Processing," *Int'l J. Computer Vision,* vol. 23, no. 1, pp. 45-78, 1997.

[77] S.C. Bae, I.S. Kweon, and C.D. Yoo, "Cop: A New Corner Detector," *Pattern Recognition Letters,* vol. 23, no. 11, pp. 1349-1360, 2002.

[78] M. Trajković and M. Hedley, "Fast Corner Detection," *Image and Vision Computing,* vol. 16, no. 2, pp. 75-87, 1998.

[79] V. Lepetit and P. Fua, "Keypoint Recognition Using Randomized Trees," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 28, no. 9, pp. 1465-1479, Sept. 2006.

[80] Z.-Q. Wu and A. Rosenfeld, "Filtered Projections as an Aid in Corner Detection," *Pattern Recognition,* vol. 16, no. 1, pp. 31-38, 1983.

[81] K. Paler, J. Föglein, J. Illingworth, and J. Kittler, "Local Ordered Grey Levels as an Aid to Corner Detection," *Pattern Recognition,* vol. 17, no. 5, pp. 535-543, 1984.

[82] B. Robbins and R. Owens, "2D Feature Detection via Local Energy," *Image and Vision Computing,* vol. 15, no. 5, pp. 353-368, 1997.

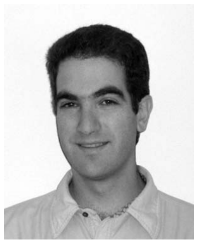
[83] G. Loy and A. Zelinsky, "A Fast Radial Symmetry Transform for Detecting Points of Interest," *Proc. Seventh European Conf. Computer Vision,* pp. 358-368, 2002.

[84] P. Dias, A. Kassim, and V. Srinivasan, "A Neural Network Based Corner Detection Method," *Proc. IEEE Int'l Conf. Neural Networks,* vol. 4, pp. 2116-2120, 1995.

[85] W.-C. Chen and P. Rockett, "Bayesian Labelling of Corners Using a Grey-Level Corner Image Model," *Proc. Fourth Int'l Conf. Image Processing,* pp. 687-690, 1997.

[86] W. Kienzle, F.A. Wichmann, B. Schölkopf, and M.O. Franz, "Learning an Interest Operator from Human Eye Movements," *Proc. 18th IEEE Conf. Computer Vision and Pattern Recognition Workshop,* 2005.

[87] L. Trujillo and G. Olague, "Synthesis of Interest Point Detectors through Genetic Programming," *Proc. Eighth Ann. Conf. Genetic and Evolutionary Computation*, pp. 887-894, 2006.
[88] P. Rajan and J. Davidson, "Evaluation of Corner Detection Algorithms," *Proc. 21st Southeastern Symp. System Theory,* pp. 29-33, 1989.
[89] J. Cooper, S. Venkatesh, and L. Kitchen, "Early Jump-Out Corner Detectors," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 15, no. 8, pp. 823-828, Aug. 1993.
[90] X. Zhang, R. Haralick, and V. Ramesh, "Corner Detection Using the Map Technique," *Proc. 12th Int'l Conf. Pattern Recognition,* vol. 1, pp. 549-552, 1994.
[91] F. Mohannah and F. Mokhtarian, "Performance Evaluation of Corner Detection Algorithms under Affine and Similarity Transforms." *Proc. 12th British Machine Vision Conf.,* T.F. Cootes and C. Taylor, eds., 2001.
[92] P. Tissainayagam and D. Suter, "Assessing the Performance of Corner Detectors for Point Feature Tracking Applications," *Image and Vision Computing,* vol. 22, no. 8, pp. 663-679, 2004.
[93] K. Mikolajczyk and C. Schmid, "A Performance Evaluation of Local Descriptors," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 27, no. 10, pp. 1615-1630, Oct. 2005.
[94] P. Moreels and P. Perona, "Evaluation of Features Detectors and Descriptors Based on 3D Objects," *Int'l J. Computer Vision,* pp. 263-284, 2007.
[95] E. Rosten and T. Drummond, "Fusing Points and Lines for High Performance Tracking," *Proc. 10th IEEE Int'l Conf. Computer Vision,* vol. 2, pp. 1508-1515, 2005.
[96] E. Rosten, G. Reitmayr, and T. Drummond, "Real-Time Video Annotations for Augmented Reality," *Proc. Int'l Symp. Visual Computing,* 2005.
[97] E. Rosten and T. Drummond, "Machine Learning for High Speed Corner Detection," *Proc. Ninth European Conf. Computer Vision,* vol. 1, pp. 430-443, 2006.
[98] J.R. Quinlan, "Induction of Decision Trees," *Machine Learning,* vol. 1, pp. 81-106, 1986.
[99] W.H. Press, S.A. Teukolsky, W.H. Vetterling, and B.P. Flannery, *Numerical Recipes in C.* Cambridge Univ. Press, 1999.
[100] http://www.robots.ox.ac.uk/~vgg/data/data-aff.html, 2007.
[101] S.M. Smith, http://www.fmrib.ox.ac.uk/~steve/susan/susan2l.c, 2005.
[102] C. Schmid, R. Mohr, and C. Bauckhage, "Comparing and Evaluating Interest Points," *Proc. Sixth IEEE Int'l Conf. Computer Vision*, pp. 230-235, 1998.
[103] D.G. Lowe, "Demo Software: Sift Keypoint Detector," http://www.cs.ubc.ca/~lowe/keypoints/, 2005.
[104] B. Sklar, *Digital Communications.* Prentice Hall, 1988.



Edward Rosten received the MEng degree from Oxford in 2002 and the PhD degree from Cambridge in 2006. He is a postdoctoral research associate in the Department of Engineering at Cambridge University, having recently moved from Los Alamos National Laboratory. His research interests include efficient computer vision, robotics, machine learning, and the application of computer vision to basic science research.

Reid Porter received the PhD degree in electrical engineering from the Queensland University of Technology in 2001. He completed his thesis while working as a graduate research assistant at Los Alamos National Laboratory from 1998 to 2001. He is now a technical staff member at Los Alamos National Laboratory, where he pursues research interests in computer architecture, signal processing, and machine learning.

Tom Drummond received the BA degree in mathematics from Cambridge University in 1988 and the PhD degree in computer science from Curtin University, Western Australia, in 1998. He is a university senior lecturer in the Department of Engineering at Cambridge University. In 2001, he was appointed to a university lectureship at Cambridge. His research interests include real-time computer vision, robust methods, robotics, and augmented reality.