Fast Corner Detection

Article *in* Image and Vision Computing · February 1998

DOI: 10.1016/S0262-8856(97)00056-5 · Source: DBLP

2 authors:

Miroslav Trajkovic
Zebra Technologies Corporation
16 PUBLICATIONS **403** CITATIONS

SEE PROFILE

Mark Hedley
The Commonwealth Scientific and Industrial Research Organisation
121 PUBLICATIONS **1,648** CITATIONS

SEE PROFILE

ELSEVIER

Image and Vision Computing 16 (1998) 75–87

image AND vision COMPUTING

Fast corner detection

Miroslav Trajković, Mark Hedley

Department of Electrical Engineering, Bldg J03, University of Sydney, Sydney, NSW 2006, Australia

Received 21 March 1996; revised 20 March 1997; accepted 16 July 1997

Abstract

This paper describes a new corner detection algorithm, based on the property of corners that the change of image intensity should be high in all directions. Consequently, the corner response function (CRF) is computed as a minimum change of intensity over all possible directions. To compute the intensity change in an arbitrary direction an interpixel approximation is used. A multigrid approach is employed to reduce the computational complexity and to improve the quality of the detected corners. This algorithm, and other popular corner detectors, were evaluated and compared on the basis of their consistency, accuracy and speed using a range of images and video sequences. It was found that our algorithm performs well compared to the other algorithms, but it is significantly faster to compute. © 1998 Elsevier Science B.V.

Keywords: Feature detection; Multigrid algorithm; Corner detector

1. Introduction

The detection of feature points in images is essential for many tasks in machine vision, including optical flow computation [1,4,11], structure from motion [7,17,20], object tracking [15,17], 3D scene reconstruction from stereo image pairs [2], etc. The reason that this approach is so popular is that feature points provide a sufficient constraint to compute image displacements reliably, and that by processing feature points the data is reduced by orders of magnitude compared to the original image data, which is particularly important for application that must run in real time.

One of the most intuitive types of feature point is the corner. Corners are image points that show a strong two-dimensional intensity change, and are therefore well distinguished from the neighbouring points. Corner detectors have been widely used as feature point detectors because corners correspond to image locations with a high information content, and they can be matched between images (e.g. temporal sequence or stereo pair) reliably. These matched feature point locations are then taken as an input to high-level computer vision tasks.

To be useful for feature point matching a corner detector should satisfy the following criteria: (1) *consistency*, detected positions should be insensitive to the variation of noise and, more importantly, they should not move when multiple images are acquired of the same scene; (2) *accuracy*, corners should be detected as close as possible to the correct position; and (3) *speed*, even the best corner detector is useless for the real time tasks if it is not fast enough.

In this paper we introduce a new corner detector and compare its performance, based on the above criteria, with other popular detectors [6,18,22]. The new corner detector uses a corner response function (CRF) that gives a numerical value for the corner strength at a pixel location based on the image intensity in the local neighbourhood. The CRF is computed over the image and corners are local maxima of the CRF. A multigrid technique is employed which both increases the computational speed of the algorithm and also acts to suppress false corners being detected in textured regions of an image.

Besides our own corner detector, we have implemented the Harris [6], SUSAN [18] and Wang [22,23] corner detectors. The four corner detectors were tested and compared on a range of images and image sequences. We also propose a modification to the Harris corner detector that decreases the computational time of the algorithm by a factor of two to three without compromising the results.

The main contributions of this paper are: (1) to present a new algorithm that overcomes some limitations of currently used corner detectors; (2) to provide a thorough and consistent comparison between our corner detector and three other widely used detectors based on consistency, accuracy and speed; and (3) to show how the computational cost of existing corner detectors may be reduced.

PII S0262-8856(97)00056-5

The organisation of the paper is as follows. First an overview of the previous work on corner detectors is presented. We then introduce and verify a new corner response function, and show how to overcome the problem of false corner detection. After that we discuss the application of a multigrid algorithm corner detection to reduce the computational cost of the corner detector. Finally, we present experimental results based on several real images and image sequences for the four tested algorithms and compare the results.

2. Previous work

One of the first approaches to finding corners was to segment the image into regions, extracting the boundaries as a chain code, then identify corners as points where directions changes rapidly (see [16] for a review of those techniques). This approach has been largely abandoned as it relied on the previous segmentation step (which is a complex task itself) and is also computationally expensive.

Subsequent corner detectors may be divided into two classes: (1) curvature based; and (2) 'interest operators', or feature point detectors.

Most of the existing corner detectors [4,9,14,22–24] belong to the first class and exploit an intuitive definition of a corner as a point on the boundary of two image regions where the boundary curvature is sufficiently high. Essentially, these methods define 'cornerness' as the product of the magnitude of the gradient of image intensity and the rate of change of gradient direction at the point under consideration. This measure has the form [9]:

$$C=\frac{I_{xx}I_y^2+I_{yy}I_x^2-2I_{xy}I_xI_y}{(I_x^2+I_y^2)}$$

As this 'cornerness' depends of the second-order derivatives, these corner detectors are generally sensitive to noise. In addition, they can not handle other types of 2D features such as **X**, **Y** and **T** junctions.

One of the curvature-based corner detectors that has been used in some applications [3,15,17] is the Wang and Brady corner detector [22,23]. To improve the stability of detected corners they convolved the original image with a Gaussian filter ($\sigma = 0.5$ pixels) and then computed the total image surface curvature. They showed that for points with a strong gradient, the total curvature κ may be approximated by:

$$\kappa=\frac{1}{|\nabla F|}\frac{\partial^2 F}{\partial t^2},\quad |\nabla F|\gg 1$$

where $\partial^2 F/\partial t^2$ is a directional derivative along direction perpendicular to the image gradient **n**. This derivative is computed directly using a linear approximation for the pixel values along the line. Corners are defined as points where κ is high and is a local maxima. After some refinement, corners were defined as points where the following conditions are fulfilled:

$$\Gamma=\left(\frac{\partial^2 F}{\partial t^2}\right)^2-S|\nabla F|^2=\text{maximum},\quad \frac{\partial^2 F}{\partial n^2}=0$$
$$|\nabla F|^2>T_1,\ \Gamma>T_2 \tag{1}$$

where F is intensity image after Gaussian smoothing, Γ is a CRF, S is a measure of image curvature specified by user and T_1 and T_2 are user-defined thresholds on edge and corner strengths.

The second class of corner detectors is the so-called feature point detectors [5,6,10,13,18]. These operators deviate from the intuitive definition of corners and define corners as points that are sufficiently different from their neighbours, or [5,6,10] as points where the local autocorrelation of the image intensity is high.

Moravec [10] defined 'points of interest' as points where there is a large intensity variation in every direction. The corner response function (CRF) is found by computing an un-normalised local autocorrelation in four directions and taking the lowest result as a measure of the CRF. As the variation was computed along four directions only, this operator is sensitive to strong edges under certain directions.

Similar ideas were used by Harris and Stephens [6], but the measure of autocorrelation was estimated from the first-order derivatives. At each pixel location they computed a 2×2 autocorrelation matrix, $A=w^*[\ (\nabla I)(\nabla I)^T]$, where w is a Gaussian smoothing mask and if both eigenvalues are large the pixel is flagged as a corner. To avoid eigenvalue decomposition of A, they defined the CRF as $\det(A)-k(\text{trace}(A))^2$ where k is a given constant (0.04). The algorithm is consistent, but computationally expensive, mainly due to the Gaussian filter that is applied three times. It was also found [12], that the algorithm works reliably only for L junctions. This algorithm is probably the most widely used one, and it is interesting to note that, in one form or another, it has been 'reinvented' by at least three different authors. Förstner and Gülch [5] first described a method that used the same measure of 'cornerness' as the Harris corner detector. They used a more complicated multiscale implementation and obtained better localisation than Harris, but the computational complexity was even higher. Tomasi and Kanade [19] obtained the same equation by analysing the optical flow equation proposed by Lucas and Kanade [8]

$$\begin{bmatrix}\sum wI_x^2 & \sum wI_xI_y\\ \sum wI_xI_y & \sum wI_x^2\end{bmatrix}\begin{bmatrix}u\\ v\end{bmatrix}=-\begin{bmatrix}\sum wI_xI_t\\ \sum wI_yI_t\end{bmatrix}.$$

This equation is stable (i.e. the optical flow can be reliably computed) if both eigenvalues are high. Corners are chosen as image locations where this condition is satisfied and this condition is seen to be the same as the one used by Harris corner detector.

Smith and Brady [18] introduced the SUSAN algorithm for low-level image processing, and this will be briefly

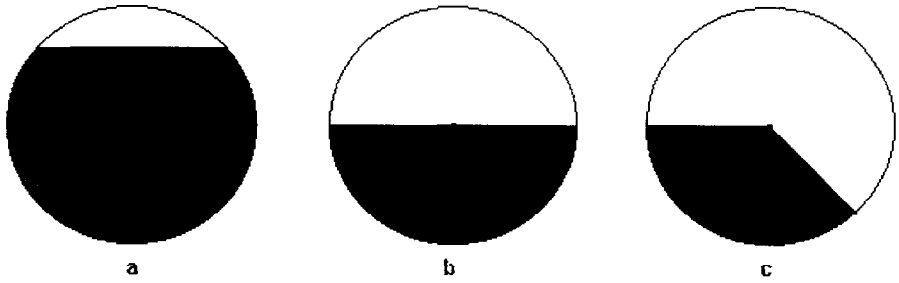


Fig. 1. Representative shapes of USAN: (a) the nucleus is within the USAN; (b) the nucleus is an edge point; (c) the nucleus is a corner point.

explained here, as we will use the same notation. Consider an arbitrary pixel in the image and a corresponding circular window around it (the central pixel shall be called the 'nucleus'). Provided the image is not textured, there is a compact region within the window whose pixels have similar brightness to the nucleus and this area will be called USAN, standing for 'Univalue Segment Assimilating Nucleus' (see Fig. 1 for the representative shapes of USAN). To find corners they computed the area and the center of gravity of the USAN, and developed a corner detector based on these parameters. Their algorithm is theoretically sound and can handle all types of junctions. The accuracy and speed of the algorithm are reasonable, but our experiments have shown that it has poor stability.

In this paper we present a new corner detector that provides an accuracy and stability that is comparable with the best of these algorithms, but which is much faster to compute. The algorithm is based on the variation of image intensity along arbitrary lines passing through the point of investigation within a neighbourhood of the point. A corner is detected if the variation of image intensity along such lines is high for all line orientations. The variation is found using only first derivatives and therefore the corner detector is less sensitive to noise than curvature-based methods, which use second-order derivatives. Our algorithm employs linear interpolation to compute the directional first-order derivatives. It makes no assumption about image structure in the vicinity of corners and it can detect any type of junction. The algorithm is very fast as it effectively rejects points with small intensity variations. A further increase in speed is achieved through the use of a multigrid approach, which also largely eliminates the detection of false corners in textured regions of the image. We refer to this algorithm as **MIC**, standing for *minimum intensity change*, as the points whose minimal intensity change over all directions is high are declared corners.

2.1. Modified Harris algorithm

In this paper we propose a modified version of Harris algorithm that achieves almost the same performance as the original algorithm, but has a much lower computational cost. In addition to a high CRF, we also require a pixel to have high image gradient in order to be a corner candidate. For each pixel we first compute the image gradient, and if it is lower than some threshold it is not necessary to evaluate the computationally expensive CRF. Since for most real images only 10–20% of image pixels have a high gradient, we do not need to compute the CRF for the majority of pixels. The drawback is that the Gaussian convolution now cannot be fully decomposed. Nonetheless, a speed up factor of two to three is obtained, depending of content of the image.

3. Corner response function

We will consider here three representative shapes of the USAN, which correspond to a point in the uniform area, on the edge and on the corner, as shown on Fig. 1. Our goal is to develop a CRF which will distinguish between a corner point (c) and a point which belongs to an edge or a uniform area (a, b).

Let us now consider an arbitrary line l containing the nucleus and intersecting the boundary of the circular window at two opposite points P and P', and the following CRF:

$$R_N = \min((f_P - f_N)^2 + (f_{P'} - f_N)^2), \tag{2}$$

where N is the central point and f_P refers to the image intensity at the point P.

Three cases can occur, corresponding to cases a, b and c.

Case a: The nucleus is within the uniform area. There is at least one line l, so that both P and P' belong to the USAN. Therefore, the response is low.

Case b: The nucleus is the edge point. There is exactly one line (tangential to the edge), so that both P and P' belong to the USAN and the CRF is again low.

Case c: The nucleus is a corner point. For every line l at

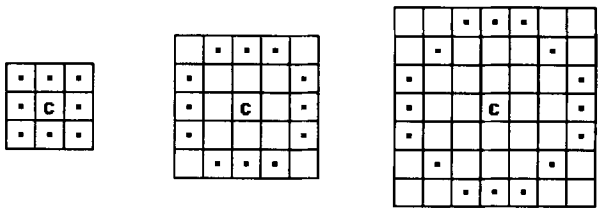


Fig. 2. Digital circles of diameter 3, 5 and 7 (S_3, S_5 and S_7).

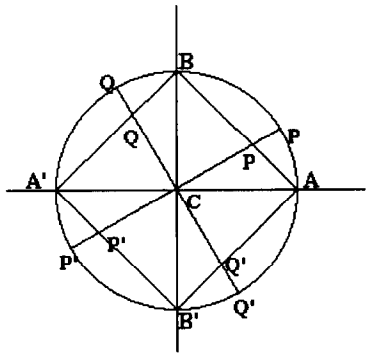


Fig. 3. First-order neighbourhood of nucleus C ($ABA'B'$) showing linear and circular interpixel positions (P, P', Q and Q').

least one of points P and P' does not belong to the USAN. Hence the CRF is high.

In practice, to compute the CRF we use a discrete approximation of the circular window, as shown in Fig. 2. Hence Eq. (2) becomes

$$R_N = \min_{P,P' \in S_n} ((f_P - f_N)^2 + (f_{P'} - f_N)^2) \tag{3}$$

where N is the nucleus and P and P' are opposite with respect to N.

4. Interpixel approximation

The problem with Eq. (3) is that a strong edge with a direction different to those examined can cause a false corner response. This can be partly resolved by using bigger window (more directions are considered) but then we usually have a worse localisation of the corner pixel. To overcome this problem, we use an interpixel approximation that can be linear or circular. To show how the interpixel approximation is used we will consider the simplest case, a window of diameter three, containing four neighbours only (Fig. 3). The extension to higher order neighbourhoods is straightforward and will not be considered here.

First we compute horizontal (r_A) and vertical (r_B) intensity variation, defined as

$$r_A = (f_A - f_C)^2 + (f_{A'} - f_C)^2,$$
$$r_B = (f_B - f_C)^2 + (f_{B'} - f_C)^2. \tag{4}$$

Then, the CRF is computed as

$$R = \min(r_A, r_B). \tag{5}$$

If R is less than a given threshold, the nucleus is not a corner point and no further computation is necessary. However, if R is greater than a given threshold, the interpixel approximation is applied to check for diagonal edges.

4.1. Linear interpixel approximation

The CRF is computed along the square $ABA'B'$ as

$$R = \min_{x \in (0,1)} (r_1(x), r_2(x)) \tag{6}$$

where x is a parameter which determines position of the point on the square. The response functions are given as

$$r_1(x) = (f_P - f_C)^2 + (f_{P'} - f_C)^2,$$
$$r_2(x) = (f_Q - f_C)^2 + (f_{Q'} - f_C)^2. \tag{7}$$

P (i.e., Q) and P' (i.e., Q') are opposite regarding to C and as shown on Fig. 3.

The intensity at interpixel locations is computed as a linear combination of the corresponding endpoint intensities. Hence.

$$f_P = (1-x)\cdot f_A + x\cdot f_B,\ f_{P'} = (1-x)\cdot f_{A'} + x\cdot f_{B'},$$
$$f_Q = (1-x)\cdot f_{A'} + x\cdot f_B,\ f_{Q'} = (1-x)\cdot f_A + x\cdot f_{B'}. \tag{8}$$

Note that $r_1(0) = r_2(0) = r_A$ and $r_1(1) = r_2(1) = r_B$. Substituting Eq. (8) in Eq. (7) we get

$$r_1(x) = A_1 x^2 + 2B_1 x + C,\ r_2(x) = A_2 x^2 + 2B_2 x + C \tag{9}$$

where

$$C = r_A;$$

$$B_1 = (f_B - f_A)(f_A - f_C) + (f_{B'} - f_{A'})(f_{A'} - f_C);$$

$$B_2 = (f_B - f_{A'})(f_{A'} - f_C) + (f_{B'} - f_A)(f_A - f_C);$$

$$A_1 = r_B - r_A - 2B_1;$$

$$A_2 = r_B - r_A - 2B_2.$$

If we define $B = \min(B_1, B_2)$ and $A = r_B - r_A - 2B$ then the CRF has a minimum on the square $ABA'B'$ iff

$$B < 0 \quad \text{and} \quad A + B > 0, \tag{10}$$

and the value of minimum is

$$R = C - \frac{B^2}{A}; \tag{11}$$

If Eq. (10) is not satisfied, then we use Eq. (5) to compute R.

4.2. Circular interpixel approximation

In this case the CRF is computed along the circle $ABA'B'$ as

$$R = \min_{\alpha \in (0, \pi/2)} (r_1(\alpha), r_2(\alpha))$$

As before

$$r_1(\alpha) = (f_P - f_C)^2 + (f_{P'} - f_C)^2$$
$$r_2(\alpha) = (f_Q - f_C)^2 + (f_{Q'} - f_C)^2 \tag{12}$$

The intensity at interpixel locations is computed using the following equations:

$$f_P - f_C = (f_A - f_C)\cdot \cos\alpha + (f_B - f_C)\cdot \sin\alpha;$$
$$f_{P'} - f_C = (f_{A'} - f_C)\cdot \cos\alpha + (f_{B'} - f_C)\cdot \sin\alpha;$$
$$f_Q - f_C = (f_{A'} - f_C)\cdot \cos\alpha + (f_B - f_C)\cdot \sin\alpha;$$

$$f_{Q'} - f_C = (f_A - f_C)\cdot \cos\alpha + (f_{B'} - f_C)\cdot \sin\alpha; \quad (13)$$

As before $r_1(0) = r_2(0) = r_A$ and $r_1(\pi/2) = r_2(\pi/2) = r_B$. Substituting Eq. (13) in Eq. (12) yields

$$r_1(\alpha) = A\cos 2\alpha + B_1 \sin 2\alpha + C;$$

$$r_2(\alpha) = A\cos 2\alpha + B_2 \sin 2\alpha + C \quad (14)$$

where

$$A = \frac{r_A - r_B}{2},$$

$$B = \frac{r_A + r_B}{2},$$

$$B_1 = (f_A - f_C)\cdot(f_B - f_C) + (f_{A'} - f_C)\cdot(f_{B'} - f_C),$$

$$B_2 = (f_{A'} - f_C)\cdot(f_B - f_C) + (f_A - f_C)\cdot(f_{B'} - f_C).$$

Defining $B = \min(B_1, B_2)$ it can be easily shown that the CRF has a maximum iff $B < 0$ and that value of the CRF is

$$R = C - \sqrt{A^2 + B^2} \quad (15)$$

As before, if $B \geq 0$ we use Eq. (5) to compute R.

5. Multigrid algorithm

A multigrid algorithm is used to compute the feature points. The advantages of this approach are: (1) To decrease the computational time; and (2) To improve the quality of the detected corners.

To show how these goals are achieved, we first classify all corners into two categories—geometrical and texture corners.

Geometrical corners belong to the boundaries of the objects in the scene. Since these objects are expected to be of reasonable size and few in number, the number of geometrical corners is small, rarely exceeding 1% of the number of all pixels in the image. Because objects in the scene do not vanish or change shape on lower scales, geometrical corners are relatively invariant to the scale (they will disappear at sufficiently low scale) or the window used to compute the CRF. Therefore these corners can be detected at any scale, with the greater precision at the higher scales.

Texture corners are associated with small or textured objects in the scene (e.g. grass or woven materials), and usually do not correspond to the physical corners of objects. Hence they are usually not good for trying to match in two images (either stereo or in time sequence). Also, there are usually more texture corners than geometrical corners in a scene, thus taking a lot of computational time to try to match between images. For these reasons we usually do not want to find texture corners. As they are created by small regions of intensity variations, these corners will disappear on lower scales, unlike geometrical corners, giving us a criteria for separating the two.

Another issue, which seems to attract little attention in the literature, is the quality and computational complexity of the CRF used. Since the number of corners usually does not exceed a few percent of the image pixels, it does not make sense to apply the same (usually expensive) CRF to each pixel, because the response is likely to be low, no matter which CRF we use. It is, therefore, much more economical to use a simple (computationally inexpensive) CRF. In case of high response, a more sophisticated CRF can be applied to verify the existence of the corner. Since we expect that most pixels will give a low response, the more sophisticated CRF will be applied only to a small number of the image locations, greatly reducing the computational time compared to using this CRF over the entire image.

The three-step algorithm used to find the corners is presented below.

Step 1. In a low resolution image compute the simple CRF (Eq. (5)) at every pixel location. Classify pixels with a response higher than a given threshold (T_1) as 'potential corners'.

Step 2. Using the full resolution image, for each potential corner pixel:

(2a) compute the CRF using Eq. (5). If the response is lower than another threshold (T_2) then the pixel is not a corner, and do not perform (2b).

(2b) Use the interpixel approximation and compute a new response as explained before. If the response is lower than threshold T_2 then the pixel is not a corner.

Step 3. Find pixels with a locally maximal CRF and mark them as corners. This step is necessary since in the vicinity of a corner more than one point will have high CRF, and only the largest one is declared to be a corner point—this is called non-maximum suppression (NMS).

6. Experimental results

In this section we examine the performance of five corner detectors, ours (MIC), SUSAN [18], Harris [6], Wang [22,23], and the modified Harris algorithm presented earlier. The algorithms were tested and compared on the basis of their:

Accuracy: this is subjectively evaluated using images presenting a range of corner types.

Stability: to test the stability of the corner detection we video sequences. This test also relies on the corner-matching algorithm that we use, which here is correlation matching over a rectangular window. The correlation matching is widely used, but can have problems in regions with little texture or features, particularly when there is a non-integer pixel displacement. Better results could be obtained using sub-pixel matching techniques, but with a greatly increased computational cost.

Computational cost: this is determined theoretically and confirmed experimentally.

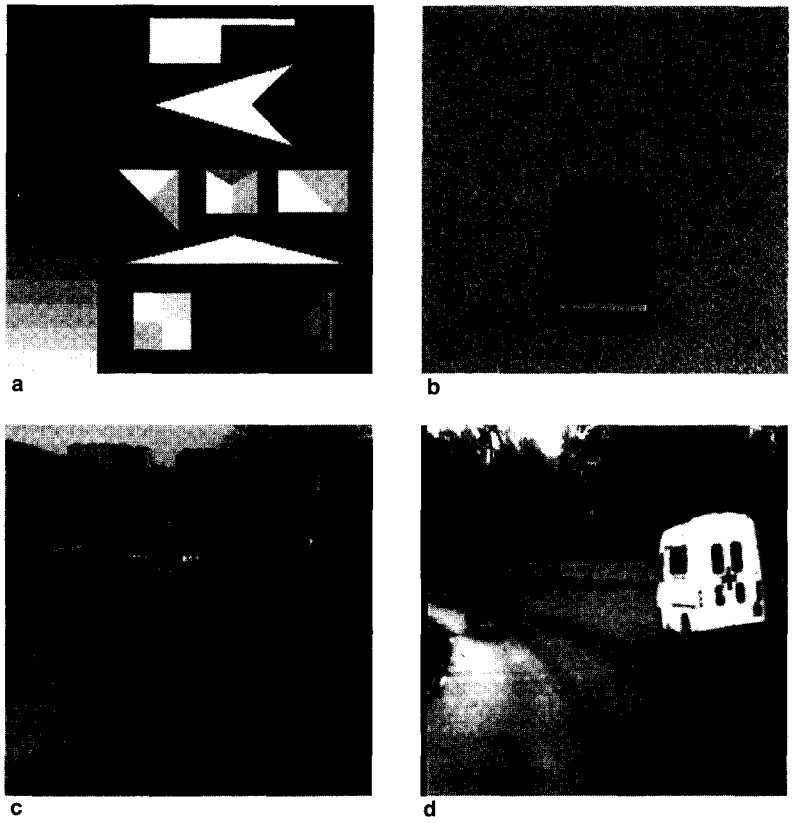


Fig. 4. Images and video sequences used to evaluate the performance of the corner detection algorithms: (a) synthetic image; (b) book image; (c) car sequence; and (d) ambulance sequence.

6.1. Test images

In this section we describe the test images and video sequences that were used to evaluate the performance of the corner detector algorithms.

6.1.1. Synthetic image

The image in Fig. 4a consists of most types of junctions (L, T, X and Y) and is widely used [18,23] to test how accurately an algorithm responds to different types of junctions.

6.1.2. Book image

The image in Fig. 4b shows a book on a carpet. It consists of nine geometric corners on the book and many texture corners on the carpet. It is used to show the ability of the algorithm to distinguish the geometrical corners from the texture corners.

6.1.3. Car sequence

Fig. 4c shows an image from the 'car' sequence, which was taken from a static camera. This image consists of three parts: a road which is mainly weakly textured, with some geometrical structure on the left side; a moving car which mainly has geometrical corners; and distant buildings with obvious geometrical corners. This video sequence was used to test the stability of the algorithms. The moving car is the most difficult part of the image due to the problem of corner matching with non-integer displacements between frames. Another difficulty with the sequence is the illumination changes between frames.

6.1.4. Ambulance sequence

Fig. 4d shows one of the images from the 'ambulance' sequence. This sequence was taken by a moving observer and consists of two vehicles, a road (weakly textured area) and trees and a bush in the background (highly textured area). Geometrical corners are present in both vehicles, and at the border of the trees, while texture corners mainly appear in the trees and the bush. This tests the ability of the algorithm to distinguish geometrical corners from texture, and the stability test has the same motion problem as the previous sequence, except now the entire image moves (due to the moving camera), rather than just a region of the image.

6.2. Selection of parameters

Every corner detector computes the CRF for each pixel in the image and from this needs to classify each pixel as either a corner or not. One part of this test is that the CRF must exceed some threshold for a pixel to be classified as a corner, and the value of threshold is usually content dependent. Some algorithms will employ other thresholds or

parameters. In all cases these parameters must be selected before the algorithm may be executed. It should be noted that for all algorithms the choice of thresholds is not critical. The range of threshold values given below will provide reasonable response for a wide range of situations. Generally, for the low contrast images, lower thresholds are more appropriate and vice-versa.

The *MIC corner detector* requires two thresholds to be chosen. The first threshold, T_1, controls the number of texture corners, and we have found experimentally that values in the range from 0 to 200 (default 50) are suitable for most of the applications. The higher T_1 is, the fewer corners will be reported. The second threshold, T_2, determines the minimum variation of brightness around the pixel, for the pixel to be a corner candidate. As this value is reduced more subtle structures in the image will be reported as corners, and more corners will be found, but the algorithm will have a higher sensitivity to noise. We have found that values ranging from 200 to 800 work well (default 500).

The original *Harris corner detector* used only one threshold and it is of the same nature as T_2 for the MIC corner detector. We have found that a suitable range for this threshold is 10 000–1 000 000 depending of the image content (default 80 000).

The *modified Harris algorithm* first checks the strength of the square of the magnitude of the image gradient and if this is higher than threshold T_1, it then computes the CRF as for the original Harris corner detector. The value for T_1 is not critical and values from 0 to 400 (default 100) were found to be suitable.

The *SUSAN corner detector* has two types of threshold [18]. The geometric threshold that controls the area of USAN was set by Smith to be half of the size of the window, and we use the same value. We found that suitable values for the brightness threshold, which depends of the contrast in the image, are in the range from 5 to 30 (default 20).

The *Wang algorithm* requires two thresholds and one constant parameter S (see Eq. (1)). S is a measure of image curvature used for the suppression of the false corners and in [22,23] they recommended values in the range from 0.0 to 0.5 (default 0.1). T_1 defines the minimum edge gradient that each other candidate must have, and it affects the number of corners and the speed of the algorithm. We found that suitable values range from 0 to 400 (default 100). T_2 is a threshold for the minimum CRF and we found suitable values range from 500 to 2000. The choice of suitable thresholds for this algorithm requires more attention, as their 'roles' are not well separated as in other algorithms. The increase of any of the parameters will reduce the number of reported corners, so if T_1 is high T_2 has to be lower in order to preserve the same number of corners, and different choices of parameters can give same number of corners, but not an identical response. However, we have found that if the parameters are kept in above-mentioned ranges, the corner maps will be similar in terms of accuracy and stability, so that choice of threshold is not critical.

6.3. Algorithm performance

Fig. 5 shows the corner maps obtained from the MIC, original Harris, modified Harris, SUSAN and Wang corner detectors applied to an image from the ambulance sequence.

The initial corner map obtained by the first step of the MIC algorithm ('potential corners') is shown in Fig. 5a (black pixels are potential corners). This map was computed on the lowest resolution, using threshold $T_1 = 80$, for which 14 864 potential corners (22.68% of all image pixels) were selected for further processing. To test the sensitivity of this parameter we computed the potential corners with $T_1 = 50$ and $T_1 = 200$, and we found they numbered 19 792 (30.2% of pixels) and 8800 (13.4% of pixels), respectively.

For the second step we computed the CRF using: (a) S_7 neighbourhood (see Fig. 2) without interpixel approximation; (b) first-order neighbourhood with linear interpolation; and (c) first-order neighbourhood with circular interpolation. For a range of images, we found that linear interpolation gave the best results and only this is presented here. The threshold T_2 was set to 500. The corners obtained using the linear interpixel approximation are shown in Fig. 5b. Strong edges have almost no effect on the final corner map. Most of the corners have been accurately found, although a few corners have not been reported on the small car and on the lower right window of the ambulance.

The corners obtained using the Harris (original and modified), SUSAN and Wang algorithms are shown in Fig. 5c–f, respectively. All the thresholds were chosen so that each corner map has about 150 corners. For both versions of Harris algorithm, threshold T_2 was set to 50 000, and for the modified version T_1 was set to 100. For the SUSAN algorithm the threshold was set to 24, and for the Wang algorithm we used $S = 0.1$, $T_1 = 100$ and $T_2 = 800$. Generally, all the corner detectors perform reasonably well, but some differences may be found. Both of the Harris corner detectors achieved almost identical results, so we will comment only on the original version. This corner detector found the lowest number of false responses, just a few on the bend of the road, but they are due to specularity of the surface. On the other hand, it has missed just a few corners, e.g. on the lower windows of the ambulance and on the back right wheel. It has found most of the corners on the small vehicle, more than any other algorithm. It may be noted that many corners found by the Harris corner detector have the same location as those found by the MIC corner detector. Compared to other corner detectors, the SUSAN corner detector performed worse on this image. It picked some subtle corners (e.g. back right wheel of the ambulance), but missed some strong corners on the car, and had a poor accuracy for most of the corners on the ambulance.

The Wang corner detector has a good accuracy and roughly the same distribution of detected corners as the MIC corner detector, although they are detected on slightly different positions. This is for the following reason. If we model the image intensity around as a cliff (e.g. convolution

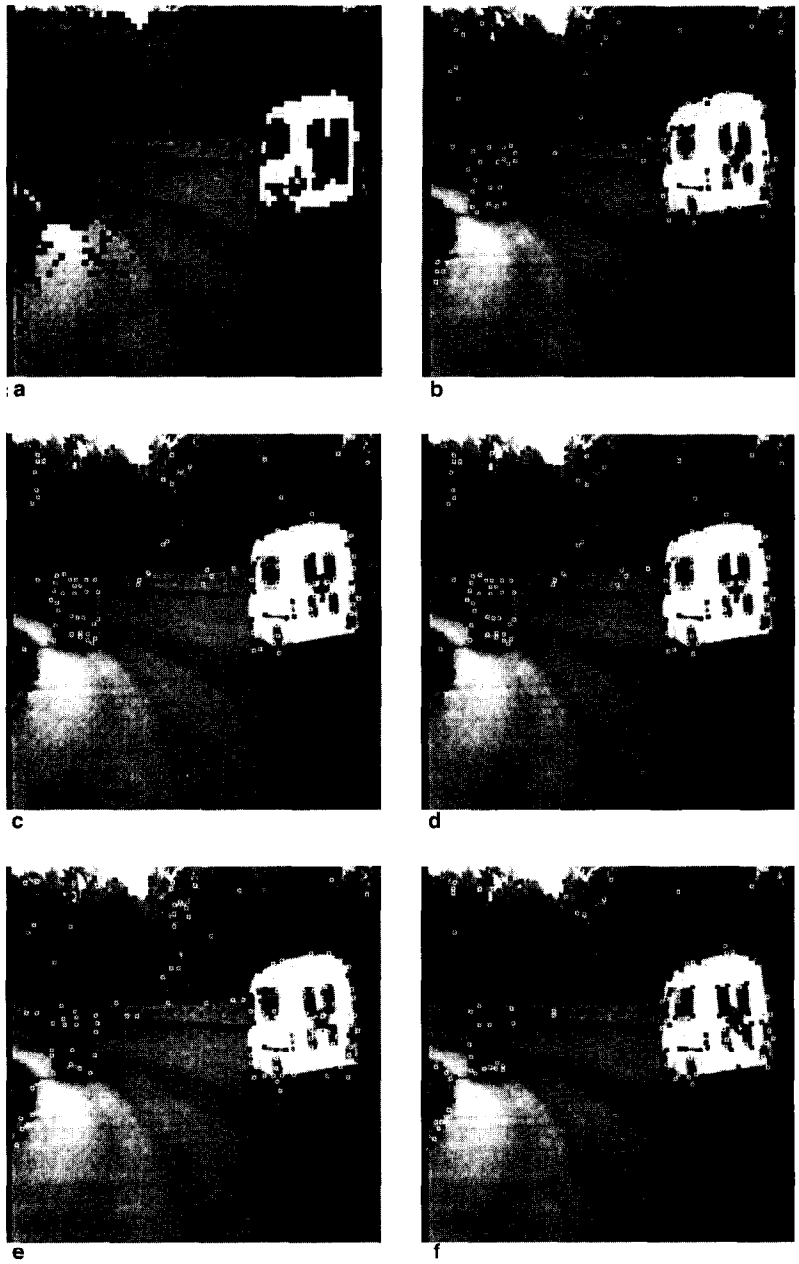


Fig. 5. Potential corners for ambulance sequence (a), and corner maps obtained using: (b) MIC algorithms; (c) the Harris corner detector; (d) the modified Harris corner detector; (e) the Susan corner detector; and (e) the Wang and Brady algorithm.

of two-dimensional step function end Gaussian), the MIC corner detector will detect a corner in the middle of the cliff, where the intensity change is the highest. the Wang corner detector should detect a corner close the top of the cliff, but due to high symmetry and noise, it may report a corner on the bottom as well (This is clearly seen on the right higher (or lower) window of the ambulance). This may be a problem when corners are tracked over the time as the detected corner can flicker between the top and the bottom.

For the Wang and Harris algorithm, the size of the Gaussian mask was 5×5, while for the SUSAN algorithm we used a '37-pixel' circular mask. For all detectors a 5×5 mask was used for the non-maximum suppression.

We also tested the MIC algorithm on the synthetic image shown on Fig. 4a, which has been widely used for measuring the accuracy of corner detectors [18,23]. Its corner map is shown in Fig. 6. All junctions have been correctly detected, as we would expect because the algorithm checks the direction of lowest contrast which will reveal the junction, unlike the derivative-based methods. Only one corner was missed, at the obtuse angle of the triangle, and this is because a small mask was used (3×3) and the angle is very close to 180°, so it appears as a line, not a corner. Also, one false corner was detected at the diagonal edge in the middle square. This is because the edge is strong and synthetic, i.e. contains a microstructure

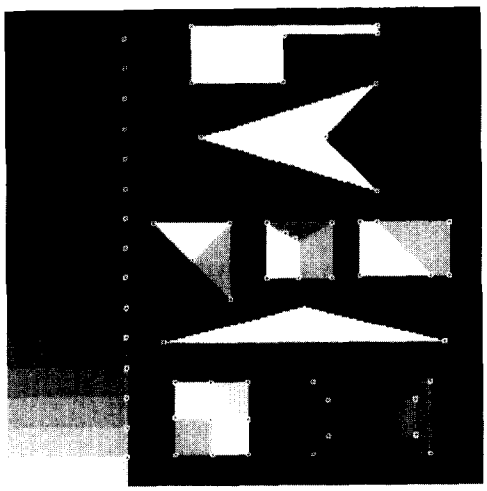


Fig. 6. Corner map of synthetic image obtained using the MIC algorithm.

that was locally detected as a corner. It should be noted that of the other three algorithms, only SUSAN gives better results, namely it detects all the corners correctly. As shown in [23], the Wang algorithm gave a similar result to the MIC, achieving good accuracy, while the Harris algorithm [6] performed the worst, achieving poor accuracy on all T junctions.

The algorithms were also tested using the 'Book' image, and the results are shown in Fig. 7, which shows the corner maps detected by the MIC and the Wang algorithms. All thresholds were manually set to give the best results. Our algorithm removed all texture corners and detected all geometric corners precisely. On the other hand, the Wang algorithm was incapable of detecting only geometric corners, and we could not get a better result using any of the other algorithms.

Comparing algorithms in terms of accuracy, we can see from the presented results that the MIC and Wang algorithms achieved the best accuracy over a range of images. The SUSAN algorithm has a good accuracy for all kind of junctions, but it seems to be vulnerable to the blurring in the image. The Harris algorithm detects accurately L junctions, but has poor accuracy for the other types.

6.4. Computational complexity

In all derivations below we assume that 75% of pixels have a low gradient (i.e. belong to uniform areas), and of the remaining 25% of pixels 20% are assumed to be edge pixels and 5% are assumed to be corners.

6.4.1. MIC algorithm

The computational superiority of our algorithm lies in the fact that we rarely check intensity changes in all possible directions. Namely, we check intensity changes by direction (investigating horizontal and vertical directions and diagonals first and then employing interpixel approximation), and once a low intensity change has been found the pixel is

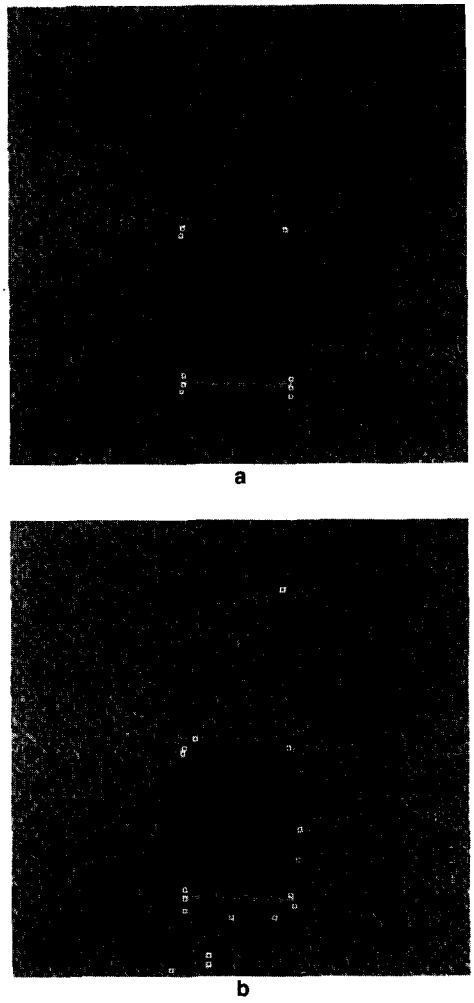


Fig. 7. Corner maps of the book image obtained using: (a) the MIC algorithm; and (b) the Wang algorithm.

rejected as a potential corner. As the majority of pixels have small intensity change (lower than T_2) in all directions, they will be disregarded in the first step, employing only three additions and two multiplications. In the worst case (high intensity change in all directions), the full check must be employed with the computational complexity $7(n + 1)$ additions and $4(n + 1)$ multiplications, where n refers to the diameter of window used.

For edge pixels, the computational complexity varies between the lowest and the highest, depending on the edge angle and strength. Assuming a uniform distribution of edge angles, the average complexity of the edge point elimination may be taken as $2 + 3.5n$ additions and $3 + 2n$ multiplications. Using the given image assumptions, the average computational complexity of our algorithm is $3.3 + 1.05n$ additions and $2.3 + 0.6n$ multiplications. For $n = 3$, which is used in our implementation, there are 10.5 operations per pixel (it is usually even lower in practice, as the number of low intensity pixels is usually higher than 75%).

When the multigrid approach is used, the computational

Table 1
Computational times of the five algorithms, over a range of images, and computational complexity in arithmetic operations

Algorithm	MIC	Mod. Harris	Wang	SUSAN	Harris
Time (ms)	70–140	400–500	350–400	320–360	800–1000
Complexity	10	30	32	49	55

complexity is even lower, and assuming 20% of the pixels are flagged as potential corners the computational cost is $2.60 + 0.875n$ additions and $1.12 + 0.5n$ multiplications, or in total for $n = 3$, 7.8 operations per pixel.

6.4.2. Wang algorithm

Due to the use of Gaussian smoothing ($\sigma = 0.5$, i.e., a 5×5 window is used) and the Sobel operator (3×3 window) the computational cost for each pixel is at least 17 additions and seven multiplications. The additional cost of computing the directional derivatives is $4n + 2$ additions and $2n$ multiplications.

Given the image assumptions, the average computational cost is $17.5 + n$ additions and $7 + 0.5n$ multiplications. For $n = 5$ (as the authors used), the total cost is 32 operations per pixel.

6.4.3. SUSAN corner detector

The SUSAN corner detector has a computational cost $\sim n^2$, but with a rather small constant. The expression for the computational complexity is rather complicated, but for $n = 7$ (that is value that authors used in his implementation) the average cost is 49 operations per pixel (48.2 additions and 0.8 multiplications, see Appendix A for more details).

6.4.4. Harris corner detector

This is the only corner detector for which the computation is independent of the image, and it required $6n$ additions and $3n + 10$ multiplications which for $n = 5$ gives a total of 55 operations per pixel. The modified Harris algorithm that we propose here first checks if intensity gradient is higher than threshold, and if so, performs the full detection. The computational complexity for $n = 5$ can be shown to be 30 operations per pixel, see Appendix A for more details.

To experimentally verify that the above analysis is approximately correct the computational times of five algorithms executed on a Pentium-based PC (90 Mhz) under the Windows 95 operating system were found (see Table 1). Note that these times are indicative only, as the actual execution time varies from image to image and also depends upon the selected parameters. It may be noted that experimental times roughly correspond to the computational complexity. The highest difference is for the SUSAN corner detector. According to the computational complexity, this algorithm should be among the slowest, but as this algorithm requires almost exclusively additions (all integer operations) it is in fact the second fastest.

6.5. Stability

Our interest in corner detection comes from its use in tracking and structure from motion estimation. As previously mentioned, a corner detector can be successfully used for these tasks if it has a good stability over time, i.e., if it can track corners reliably over a sequence of images. The minimum requirement that any corner detector must fulfil is that for each moving segment it can reliably track at least four corners over three frames [15].

As there is no standard procedure to measure stability of corner detector we have performed the following test to compare the stability of different corner detectors. For each corner detector we choose threshold(s), so that each of them detects a similar number of corners in the first frame. Then, we find corners in the next three frames and perform matching using a cross-correlation-based procedure as described in [21]. Basically, between each two consecutive frames, we find mutually best matches, and only corners that are reliably matched over three consecutive pairs are used (we call them 'strong matches'). Ideally, if there is no occlusion, the number of strong matches will be same as the number of corners in the first frame. However, due to variation of noise and imperfection of corner detectors this number will be lower. As a measure of stability we can define $\kappa = N_m/N_c$, where N_m and N_c denote number of strong

Table 2
Results of stability tests for the five algorithms over two, three and four frames using the car sequence: all thresholds were chosen to give a similar number of corners in the first frame

Algorithm	Thresholds $T_1/T_2/S$	Initial corners	Matches over (frames)		
			2	3	4
MIC	20/600	276	236	213	208
Harris (orig.)	20 000	278	245	228	219
Harris (mod.)	100/20 000	277	245	227	218
SUSAN	19	276	178	132	111
Wang	80/300/0.1	273	218	185	175

Table 3
Results of stability tests for the five algorithms over two, three and four frames using the ambulance sequence: all thresholds were chosen to give a similar number of corners in the first frame

Algorithm	Thresholds $T_1/T_2/S$	Initial corners	Matches over (frames)		
			2	3	4
MIC	0/410	250	154	103	83
Harris (orig.)	10000	250	186	142	118
Harris (mod.)	100/10000	250	185	141	118
SUSAN	19	256	135	58	29
Wang	60/240/0.1	249	150	92	68

matches and number of corners in the first frame respectively. In terms of stability, a corner detector is better if κ is higher.

The results of the stability test for all corner detectors are presented for both the 'car' and the 'ambulance' sequences. The results for all algorithms applied to both sequences are given in Table 2 and Table 3.

As can be seen from Table 2, all algorithms performed well for this sequence except SUSAN. The Harris algorithm (both modified and original) has the best performance, while our algorithm has slightly lower stability over all frames. The Wang algorithm also has good stability for this sequence (about 10% lower than ours), while the SUSAN algorithm has the lowest stability.

The second sequence is more difficult because the entire scene is moving, so it is not surprising that all the algorithms performed worse on this sequence, as seen in Table 3.

This sequence gives a clearer insight into the difference among corner detectors. The Harris detector is the most reliable with stability over four frames of about 50%. Our algorithm achieved around 33%, while the Wang algorithm stability was about 25%. The SUSAN corner detector had the lowest stability, of only around 12%.

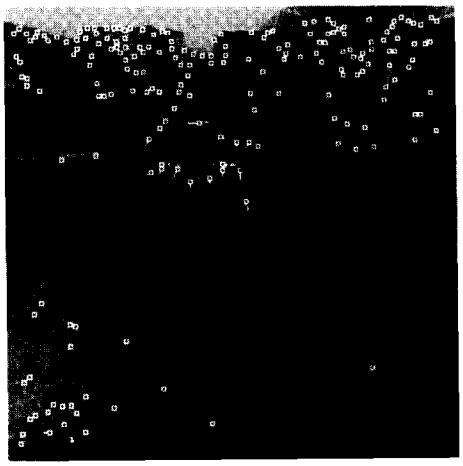
The results for corner matching are presented for our algorithm only and are shown in Fig. 8. The figure shows the strong matches over four frames for the two sequences. A 5×5 mask was used for the cross-correlation matching, the diameter for the search space was set to 11, and the correlation threshold was set to 0.5 (see [21] for more details).

6.6. Comparison among algorithms

The properties of the five algorithms are summarised in Table 4. The accuracy is a subjective assessment based on the results presented earlier in this section (see Section 6.3).

7. Conclusion

In this paper a novel corner detector was introduced. The new corner response function (CRF) operates in a 3×3 window, and to overcome the problem of lines at certain orientations being detected as corners an interpixel approximation was used. Two types of approximation were tested, and the linear approximation was found to perform better over a range of images. A multiguard approach was employed to reduce the sensitivity of the algorithm to false corners in textured regions of the images, and to increase the computation speed of the algorithm.



a

b

Fig. 8. The images show strong corner matches obtained from a sequence of four images using the MIC algorithm on: (a) the car sequence; and (b) the ambulance sequence.

Table 4
Stability, accuracy and speed of five corner detectors

Algorithm	Stability	Accuracy	Speed
MIC	Good	Good	Excellent
Harris (orig.)	Excellent	Good for L junctions, poor for all other types	Very slow
Harris (mod.)	Excellent	As for original	Good
SUSAN	Poor	Bad for blurred images, very good otherwise	Good
Wang	Good	Good	Good

Five algorithms were implemented and compared, ours (MIC), Harris [6], modified Harris, SUSAN [18] and Wang [22,23]. These were evaluated on the basis of their accuracy, consistency and speed. It was found that for accuracy the performance of our algorithm was among the best and for consistency our algorithm performed well, just behind the best which is the Harris algorithm, but our algorithm was significantly faster than any of the other algorithms—which is important for real time machine vision applications.

Acknowledgements

We wish to thank S. Smith from DRA Chartsey, UK, for providing the ambulance sequence, and the University of Sydney Research Grant Scheme for financial support.

Appendix A Computational complexities of the SUSAN and modified Harris corner detectors

In the derivations in this appendix we assume that 75% of pixels have a low gradient (i.e. belong to uniform areas), and of the remaining 25% of pixels 20% are assumed to be edge pixels and 5% are assumed to be corners.

Appendix A.1 SUSAN corner detector

From the computational point of view, the SUSAN corner detection algorithm can be divided in three steps. The first step involves computation of the USAN area for half of the pixels from the circular neighbourhood and this step must be performed for each pixel. This step involves $\pi(n) - 1$ additions, where π is a function which gives the number of points in a digital circle of diameter n, e.g., $\pi(3) = 9$, $\pi(5) = 21$ and $\pi(7) = 37$.

The second step involves computation of the USAN area for the second half of circular neighbourhood, but after value for each pixel is added, the size of USAN is compared with a threshold and if the point is not a corner no further computation is performed. Because of the comparison with threshold, the computational cost of this step is $1.5(\pi(n) - 1)$. This step is not performed for pixels in uniform areas, as they will have large USAN after the first step. According to the above assumption this step will be performed for roughly 25% of pixels (edge and corner pixels). After this step another 20% of the pixels should be rejected, and as they can be rejected in various phases of the algorithm, we can take the average time of processing each edge pixel to be $0.75(\pi(n) - 1)$.

The remaining 5% of pixels will go to the third step, which checks for false corners and has a computational complexity of $2\pi(n) - 2n + 3$ additions and $2n + 2$ multiplications.The total cost of the SUSAN algorithm can now be evaluated as $1.325\pi(n) - 0.1n - 0.05$ additions and $0.1n + 0.1$ multiplications, giving 49 operations for $n = 7$.

Appendix A.2 Modified Harris corner detector

The computational complexity of the modified Harris corner detector can be found in a similar manner. For all pixels we have to compute I_x^2, I_y^2 and I_xI_y and to compare the gradient with a threshold and this requires three multiplications and four additions. The second step is performed on 25% of pixels having a computational cost of $3n^2 + 4$ additions and $3\mu(n) + 7$ multiplications where $\mu(n) = (n+1)(n + 3)/8$. The average computational cost per pixel can be easily found as $2.5 + 0.75n^2$ additions and $4 + 0.75\mu(n)$ multiplications, giving 30 operations per pixel for $n = 5$.

References

[1] S.T. Barnard, W.B. Thomson, Disparity analysis of images, IEEE Trans. PAMI 2 (4) (1980) 333–340.
[2] R.J. Blisset, Retrieving 3D information from video for robot control and surveillance, Electr. Commun. Eng. J. (1990) 155–163.
[3] M. Brady, H. Wang, Vision for mobile robots, Phil. Trans. Royal Soc. London B 337 (1992) 341–350.
[4] L. Dreschler, H. Nagel, Volumetric model and 3D trajectory of a moving car derived from monocular TV-frame sequence of a street scene, CVGIP 20 (3) (1982) 199–228.
[5] W. Förstner, E. Gülch, A fast operator for detection and precise location of distinct points, corners and centers of circular features, ISPRS Intercommision Workshop, Interlaken, June 1987, pp. 149–155.
[6] C. Harris, M. Stephens, A combined corner and edge detector, Proc. 4th Alvey Vision Conf., 1988, pp. 147–151.
[7] C. Harris, Structure from motion under orthographic projection, Proc. 1st ECCV, 1990, pp. 118–123.
[8] B. Lucas, T. Kanade, An iterative registration technique with an application to stereo vision, Proc. 7th IJCAI, vol. 2, 1981, pp. 674–679.
[9] L. Kitchen, A. Rosenfeld, Grey-level corner detection, Pattern Recognition Lett. 1 (2) (1982) 95–102.
[10] H. Moravec, Towards automatic visual obstacle avoidance, Proc. IJCAI, 1977, p. 584.
[11] H. Nagel, On the estimation of optical flow: Relations between different approaches and some new results, Artif. Intell. 33 (1987) 299–324.
[12] A. Noble, Finding corners, Image Vision Comput. 6 (2) (1988) 121–128.
[13] K. Paler, J. Föglein, J. Illingworth, J. Kittler, Local ordered grey levels as an aid to corner detection, Pattern Recognition 17 (5) (1984) 535–543.

[14] K. Rangarajan, M. Shah, D.V. Brackle, Optimal corner detector, CVGIP 48 (1989) 230–245.
[15] I.D. Reid, D.W. Murray, Active tracking of foveated feature clusters using affine structure, Int. J. Comp. Vision 18 (1996) 41–60.
[16] W.S. Rutkowski, A. Rosenfeld, A comparison of corner detection techniques for chain-coded curves, Technical report 623, Computer Science Center, University of Maryland, 1977.
[17] L. Shapiro, Affine Analysis of Image Sequences, Cambridge University Press, 1995.
[18] S. Smith, M. Brady, SUSAN—a new approach to low level image processing, DRA Technical Report TR95SMS1, 1994.
[19] C. Tomasi, T. Kanade, Shape and motion from image streams: a factorization method—Part 3, Detection and Tracking of Point Features, Technical Report, CMU-CS-91-132, 1991.
[20] C. Tomasi, T. Kanade, Shape and motion from image streams under orthography: a factorization, Int. J. Comp. Vision 9 (2) (1992) 137–154.
[21] M. Trajković, M. Hedley, Fast feature detection and matching for machine vision, Proc. 7th BMVC, vol. 1, 1996, pp. 93–102.
[22] H. Wang, M. Brady, A practical solution to corner detection, Proc. 5th ICIP, vol. 1, 1994, pp. 919–923.
[23] H. Wang, M. Brady, Real-time corner detection algorithm for motion estimation, Image Vision Comput. 13 (9) (1995) 695–703.
[24] O. Zuniga, R. Haralick, Corner detection using the facet model, Proc. IEEE Conf. CVPR, 1983, pp. 30–37.