

Corner Detection

D.Parks, J.P. Gravel

Introduction

Corners as Interest Points

Many [applications](#) require relating two or more images in order to extract information from them. For example, if two successive frames in a video sequence taken from a moving camera can be related, it is possible to extract information regarding the depth of objects in the environment and the speed of the camera. The brute force method of comparing every pixel in the two images is computationally prohibitive for the majority of applications. Intuitively, one can image relating two images by matching only locations in the image that are in some way interesting. Such points are referred to as interest points and are located using an interest point detector. Finding a relationship between images is then performed using only these points. This drastically reduces the required computation time.

Many different interest point detectors have been proposed with a wide range of definitions for what points in an image are interesting. Some detectors find points of high local symmetry, others find areas of highly varying texture, while others locate corner points. Corner points are interesting as they are formed from two or more edges and edges usually define the boundary between two different objects or parts of the same object. Many corner detectors have been developed and this website investigates some of the more important ones.

Applications of Corner Detectors

The use of interest points (and thus corner detectors) to find corresponding points across multiple images is a key step in many image processing and computer vision applications. Some of the most notable examples are:

- stereo matching
- image registration (of particular importance in medical imaging)
- stitching of panoramic photographs
- object detection/recognition
- motion tracking
- robot navigation

Figure 1.1 shows part of a hypothetical system to illustrate how a corner detector might be used in an automated assembly line. This assembly line fills triangle gift boxes with four different chocolates. However, the boxes must be positioned properly on the

conveyor belt to ensure the chocolates are packed properly into the boxes. An overhead camera is used to capture a picture of each box as it passes under it and a computer compares it to a stored image of a properly aligned box. By finding the corners of each image, how much the box needs to be rotated can easily be computed.

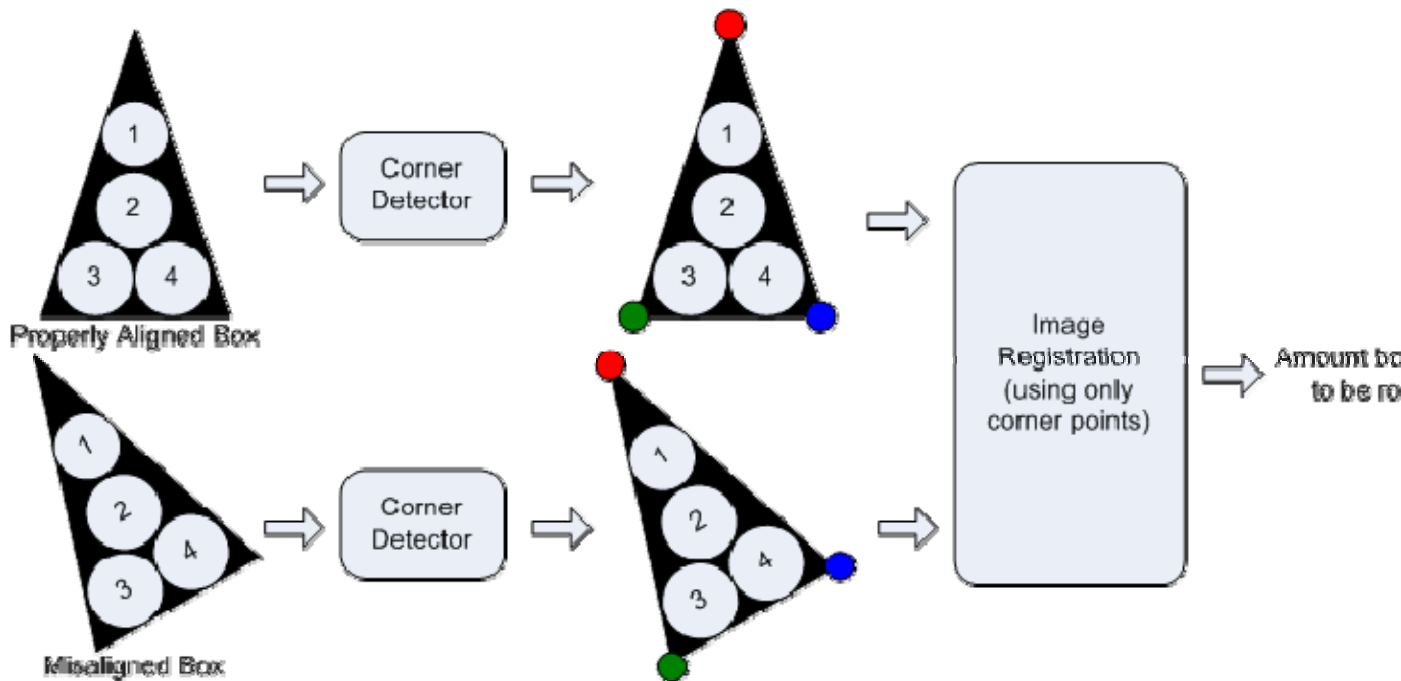


Figure 1.1: Part of a systems for aligning boxes on an assembly line

Requirements of a Corner Detector

It is desirable for a corner detector to satisfy a number of criteria:

1. All "true corners" should be detected.
2. No "false corners" should be detected.
3. Corner points should be well localized.
4. Detector should have a high repeatability rate (good stability).
5. Detector should be robust with respect to noise.
6. Detector should be computationally efficient.

The detection of all true corners with no false corners is application (interpretation) dependent since there is no well defined definition of a grayscale corner. However, in many images the corners are intuitively clear and such images can be used to evaluate the performance of different corner detectors (see [Evaluating and Comparing Corner Detectors](#)).

Localization refers to how accurately the position of a corner is found. This is critical in applications requiring the precise alignment of multiple images (for example, in registration of medical images). In Figure 1.2 the reported position of the corner is illustrated with a red circle. The corner detector on the right has good localization whereas the corner detector on the left has poor localization. Although good localization is desirable for all applications, it is not critical for all applications (for example, an object detection algorithm may simply require the approximate location of all the object's corners).

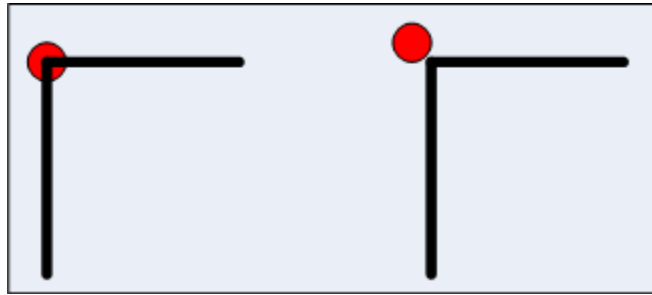


Figure 1.2: Illustration of good and poor localization, respectively

Think of a robot wandering down a hallway with a single camera. Many approaches in mobile robotics require analyzing the frames captured by the camera in order to interpret the robot's environment. A first step in many of these approaches requires finding corresponding points between frames. The two consecutive frames will be similar, but may differ due to slight geometric, illumination, or viewpoint transformations. A corner detector that is robust against these transformations is said to have a high repeatability rate. The repeatability rate is the percentage of the total number of corner points which are repeated between two images. Figure 1.3 shows a corner detector that fails to detect one of the triangle's corners after it is rotated, so has a repeatability rate of $2/3$ for this transformation.

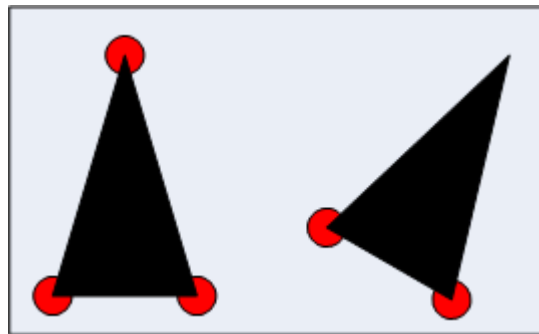


Figure 1.3: Corner detector failing to detect one of the triangle's corners after rotation

Since noise in images is unavoidable in most applications, a corner detector must be robust against it. The corner detector should not incorrectly identify noise as corner points and noise should have minimal affect on the localization ability of the detector.

Many applications require real-time processing so the corner detector must be computationally efficient. Such applications will often elect to use a simpler (and less

robust or precise) corner detector in order to minimize the time spent finding the corner. Many corner detectors have been designed with this criteria in mind.

Evaluating and Comparing Corner Detectors

Each of the corner detectors are evaluated on an artificial test image containing different corner types and two real world images (see the "Results on Test Images" section for each [Corner Detection Algorithm](#)). The section [Evaluation Summary](#) gives a side-by-side comparison of each of these corner detector on these test images. It is important to remember that the intended application of the corner detector must be kept in mind when selecting a corner detector. For example, in an image alignment application the repeatability rate is critical, but detecting all true corners is of little importance as long as a sufficient number of interest points are available to allow accurate alignment of the images. However, in an object recognition application failing to detect a corner may result in a different description of the object being generated leading to a misclassification of the object.

The image in Figure 1.4 contains many corner types (L-Junction, Y-Junction, T-Junction, Arrow-Junction, and X-Junction as depicted in Figure 1.5) and is widely used to evaluate how a corner detector responds to each of these corner types. This test images also contains corners formed at a range of different grayscale values to test operation on corners of varying intensity.

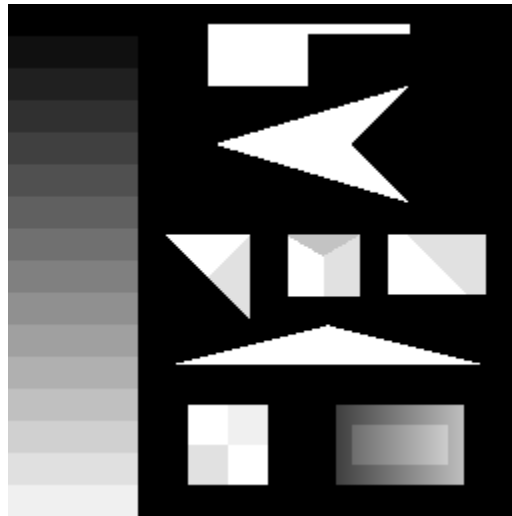


Figure 1.4: Artificial test image containing different corner types

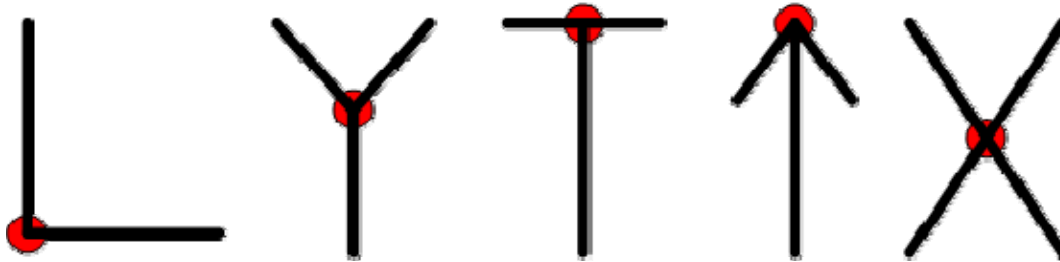


Figure 1.5: Example of L-junction, Y-junction, T-junction, Arrow-junction, and X-junction corner types

Since the majority of applications operate on real-world images, each of the corner detectors are evaluated on the two real images in Figure 1.6. The "block" image tests the operation of the corner detectors on a real image where the location of the corners is intuitively clear, the background is uniform, and each object has a nearly uniform colour and texture. An image such as this, is representative of the type of images required in many assembly line or manufacturing type applications. The "house" image is a more difficult image due to the texture on the side and roof of the house and the presence of many different types of corners. This examples illustrates why a strict definition for a corner has not been established - it is unlikely different individuals would agree on exactly what parts of the image have a corner and would certainly not agree on the precise location of a given corner.



Figure 1.6: Real test images of a block and house scene

Background

Since the first corner detectors were developed in the late 1970's, dozens of corner detectors have been proposed. This section highlights a few of the more prevalent detectors in order to give the reader a feel for the variety of approaches that have been

developed and to provide a historical context for the corner detectors discussed in this website. Perhaps most importantly, this section illustrates that the seemingly simple problem of corner detection is still an open problem as no universally "good" corner detector exists. Figure 2.1 shows a timeline of the corner detectors discussed in this section with the corner detectors explored on this website highlighted in red.

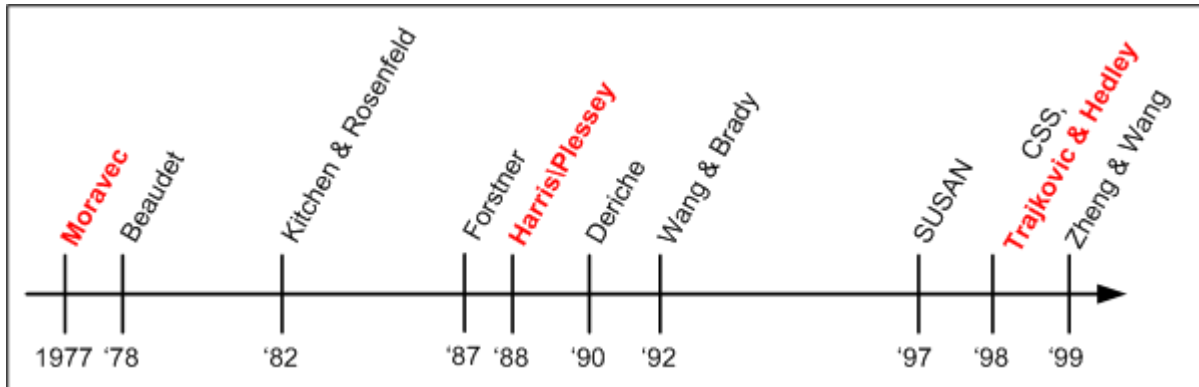


Figure 2.1: Timeline of select corner detectors

There are a significant number of different approaches for detecting corners. However, there are three main trends for detection of corners in gray scale images: edge-relation methods, topology methods, and autocorrelation methods. Most of the corner detectors discussed here can be placed into one of these categories. However, it should be made clear that the classification is somewhat arbitrary as most of the detectors have multiple interpretations and thus could be classified in more than one way. The detectors that do not fit into one of these categories are discussed in a separate section at the end of this page.

Edge-Relation Methods

Kitchen and Rosenfeld were the first to apply differential geometry operators to corner detection. They proposed a cornerness measure for each pixel based on the change of gradient direction (second-order derivatives) along an edge contour weighted by the local gradient magnitude. Corners are then identified by the local maximum of this measure, which can be done in a computationally efficient manner by applying non-maximum suppression to the gradient magnitudes before weighting the second-order derivatives. This corner detector suffers from sensitivity as it relies on second-order derivative terms and has been shown to have a poor repeatability rate and localization.

A popular corner detector designed for motion estimation was proposed by Wang and Brady. They applied differential geometry operators to detect corners based on the measurement of surface curvature, but derived a simplified cornerness measure suitable for real-time applications while improving performance relative to the Kitchen and Rosenfeld operator.

Topology Methods

Beaudet developed one of the first corner detectors and the insights gained from his approach are exploited by many other corner detectors. The Beaudet operator is a rotationally invariant measure of cornerness given by the determinant of the [Hessian matrix](#). Since the Hessian matrix involves the computation of second-order derivatives this operator is sensitive to noise. In addition, it has been shown to be unstable in [scale space](#). This approach can be viewed as looking for high curvature edges (i.e. [saddle points](#) in the [image surface](#)) by calculating image Gaussian curvature (i.e. the product of two principle curvatures).

Deriche extended Beaudet's operator by applying it at multiple scales. Lines are drawn between the corresponding corners at each scale and the intersection of these lines with the nearest zero crossing of the [Laplacian](#) image are defined as the position of the corner. This method improves the localization relative to the Beaudet approach, but the Deriche operator still suffers from sensitivity to noise.

Autocorrelation Methods

The Moravec operator [\[1,2\]](#) is of importance as it inspired the widely used Plessey operator developed by Harris et al. and as Moravec's paper introduced the concept of "points of interest". His operator considers a local window in the image and determines the average change of intensity resulting from shifting the window by a small amount in various directions. This operation is repeated for each pixel position which is assigned an *interest value* equal to the *minimum* change produced by these shifts. Points of interest are the local maximum of the interest values. Since corners exhibit a large intensity variation in every direction, this operator is a corner detector - although, with a more relaxed definition for corner. Moravec implemented this approach by computing the unnormalized local [autocorrelation](#) in the 4 principle directions, which results in an [anisotropic](#) response. This operator has been shown to be sensitive to noise along strong edges because only the minimum intensity change is considered for each pixel position (as opposed to the variation between them).

Many of the weaknesses of the Moravec operator are addressed by the Plessey operator [\[3\]](#) (often referred to as the Harris operator). Harris estimates the measure of local autocorrelation using first-order derivatives which is suggested by performing an analytic expansion of the Moravec operator. The response is isotropic as the variation of the autocorrelation over different orientations can be calculated from the principle curvatures of the local autocorrelation (as discussed in the [Harris/Plessey Operator](#) section, the response is theoretically isotropic, but is often calculated in such a manner as to make it anisotropic). The Plessey operator is generally considered the best operator with respect to detecting true corners, but has poor localization and is expensive to compute.

To address the computational complexity of the Plessey operator, Zheng and Wang have proposed a computationally simplified cornerness measure justified by analyzing the

Plessey operator and identifying the key aspects responsible for corner detection. As expected, this corner detector has slightly degraded performance in terms of detecting corners, but reduced the computational complexity and makes some additions to improve localization.

The Forstner operator uses a similar measure of cornerness to the Plessey operator (although, how this measure is calculated differs greatly) and uses local statistics to calculate the selection threshold. The result is better localization at the cost of increased computation. In practice, the Forstner operator is often used as it is easily extended to detect the center of circular features along with corners.

Alternative Methods

The Curvature Scale Space (CSS) operator detects corners by directly looking for local maxima of absolute curvature. That is, it detects corners using the intuitive notion of locating when the contour of an object makes a sharp turn. The following is an outline of the steps involved in the CSS corner detector:

1. Extract the edge contours from the input image using any good edge detector (e.g. Canny).
2. Fill small gaps in edge contours. When the gap forms a T-junction, mark it as a T-corner.
3. Compute curvature on the edge contours at a high scale.
4. The corner points are defined as the maxima of absolute curvature that are above a threshold value.
5. Track the corners through multiple lower scales to improve localization.
6. Compare T-corners to the other corners found using the CSS procedure and remove very close corners.

Experimental results indicate approximately 80% of the time is spent performing edge detection. Applications requiring faster corner detection can trade off other performance criteria for speed by using a simpler corner detector. Steps 2 and 6 concern themselves with T-junctions in order to avoid assigning two corners to pixels which are close together. As illustrated in Figure 2.2, as the contour is being followed it will pass by a T-junction twice and thus come across the same corner twice.

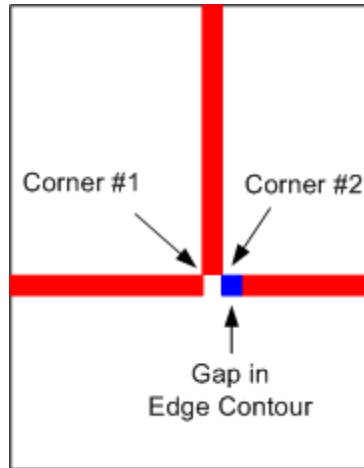


Figure 2.2: CSS operator detecting the same corner twice at a T-junction

A novel corner detector algorithm based on brightness comparisons within a circular mask is proposed by Smith and Brady [4]. SUSAN (Smallest Univalve Segment Assimilating Nucleus) assumes that within a relatively small circular region pixels belonging to a given object will have relatively uniform brightness. The algorithm computes the number of pixels with similar brightness to the pixel at the center of the mask (the nucleus of the mask). These pixels are called the USAN (Univalve Segment Assimilating Nucleus) of the mask. Corners are detected by applying the mask to all pixels in the image and then finding the local minima in this new USAN map. Figure 2.3 illustrates this circular mask applied to different positions of a black rectangle with the USAN shown in red. Notice that the USAN becomes smaller as it approaches an edge and this reduction is stronger at corners. SUSAN can thus be used for both line and edge detection. This corner detector is robust to noise (note that no spatial derivatives are computed), fast to compute, but only has an average repeatability rate.

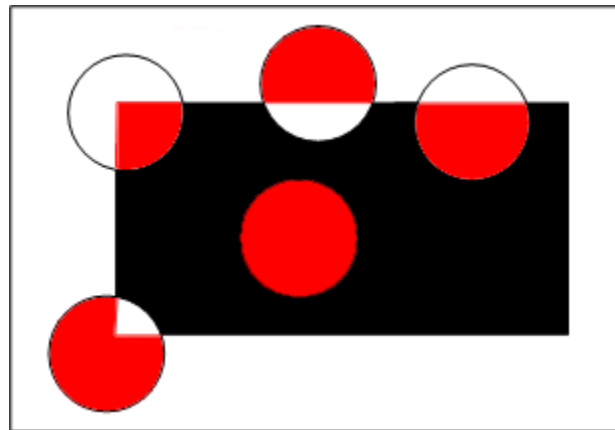


Figure 2.3: USAN for different circular masks on a uniform rectangle

Trajkovic and Hedley [5] have proposed a corner detector that uses the same intuition used in the SUSAN operator. For a given point in the image, the variation in brightness along all lines passing through the point are considered. At corners the variation in

brightness will be high for all lines. The repeatability rate of this algorithm is not as strong as the Plessey operator, but it is one of the fastest available corner detectors.

Comparison of Select Corner Detectors

Table 3.1 provides a comparison of selected corner detectors based on the criteria given in the [Introduction](#). The results of Table 3.1 were compiled based on the comparisons performed in [\[5\]-\[8\]](#) along with the literature search done while compiling the [Background](#) section. It should be noted that no standard evaluation method has been developed for these criteria so results vary based on how the different criteria is tested. This comparison can aid in selecting a corner detector for a given application and emphasizes that no optimal corner detector is available.

Detection rate is a measure of the number of true corners found relative to the number of false detections and missed corners.

Operator	Detection Rate	Localization	Repeatability Rate	Robustness to Noise	Speed
Beaudet	Fair	Fair	Poor for scaling, good for affine transformations	Poor	Good
Moravec	Fair	Good	Fair	Fair	Good
Kitchen & Rosenfeld	Fair	Fair	Fair	Fair	Poor
Forstner	Good	Good	Excellent for affine transformations, fair for scaling	Good	Poor
Plessey	Good	Good for L-junctions, poor for all other types	Excellent for affine transformations if isotropic gradient calculations are used, fair for scaling	Fair	Poor
Deriche	Fair (?)	Good	Good	Poor	Good
Wang & Brady	Good	Good	Good	Fair	Good
SUSAN	Good	Bad for blurred images,	Good for scaling, poor for affine transformations	Excellent	Good

		very good otherwise			
CSS	Good	Good	Excellent	Good	Highly dependent on edge detector used
Trajkovic & Hedley (4- neighbours)	Poor	Good	Fair (not rotationally invariant)	Poor	Excellent
Trajkovic & Hedley (8-neighbours)	Fair	Good	Fair+ (not rotationally invariant)	Good	Excellent
Zheng & Wang	Good	Good for L- junctions, fair for all other types	Excellent for affine transformations, fair for scaling	Fair	Fair

Table 3.1: Comparison of select corner detectors

Corner Detection Algorithms

Three corner detection algorithms are explored in detail in this section. They have been chosen because they are historically significant, widely used, or well suited for a particular application (i.e. real-time). In addition, all these detectors can be considered interest point corner detectors as they assign a measure of cornerness to all pixels in an image. The majority of corner detectors fall into this interest point category, although how they calculate the cornerness measure varies significantly as discussed in the [Background](#) section. This is in contrast to corner detectors which find corners by tracing the contours of objects and look for local maxima of absolute curvature or approaches using morphological operators.

The following corner detection algorithms are considered here:

- [Moravec](#) (1977)
- [Harris/Plessey](#) (1988)
- [Trajkovic and Hedley](#) (4-Neighbours) (1998)
- [Trajkovic and Hedley](#) (8-Neighbours) (1998)

All of these algorithms follow the same general steps for detecting corners. Figure 4.1 shows a flowchart of these steps:

1. *Apply Corner Operator*: This step takes as input the image and typically a few parameters required by the corner operator. For each pixel in the input image, the corner operator is applied to obtain a *cornerness measure* for this pixel. The cornerness measure is simply a number indicating the degree to which the corner operator believes this pixel is a corner. Interest point corner detection algorithms differ on how the corner operator makes this measurement, but all algorithms consider only pixels within a small window centered on the pixel a measurement is being made for. The output of this step is a *cornerness map*. Since for each pixel in the input image the corner operator is applied to obtain a cornerness measure, the cornerness map has the same dimensions as the input image and can be thought of as a processed version of the input image.
2. *Threshold Cornerness Map*: Interest point corner detectors define corners as local maximum in the cornerness map. However, at this point the cornerness map will contain many local maximum that have a relatively small cornerness measure and are not true corners. To avoid reporting these points as corners, the cornerness map is typically thresholded. All values in the cornerness map below the threshold are set to zero. Choosing the threshold is application dependent and often requires trial and error experimentation. The threshold must be set high enough to remove local maximum that are not true corners, but low enough to retain local maximum at true corners. In practice there is rarely a threshold value that will remove all false corners and retain all true corners so a trade-off must be made based on the requirements of the application.
3. *Non-maximal Suppression*: The thresholded cornerness map contains only non-zero values around the local maximums that need to be marked as corner points. To locate the local maxima, *non-maximal suppression* is applied. For each point in the thresholded cornerness map, non-maximal suppression sets the cornerness measure for this point to zero if its cornerness measure is not larger than the cornerness measure of all points within a certain distance. After non-maximal suppression is applied, the corners are simply the non-zero points remaining in the cornerness map.

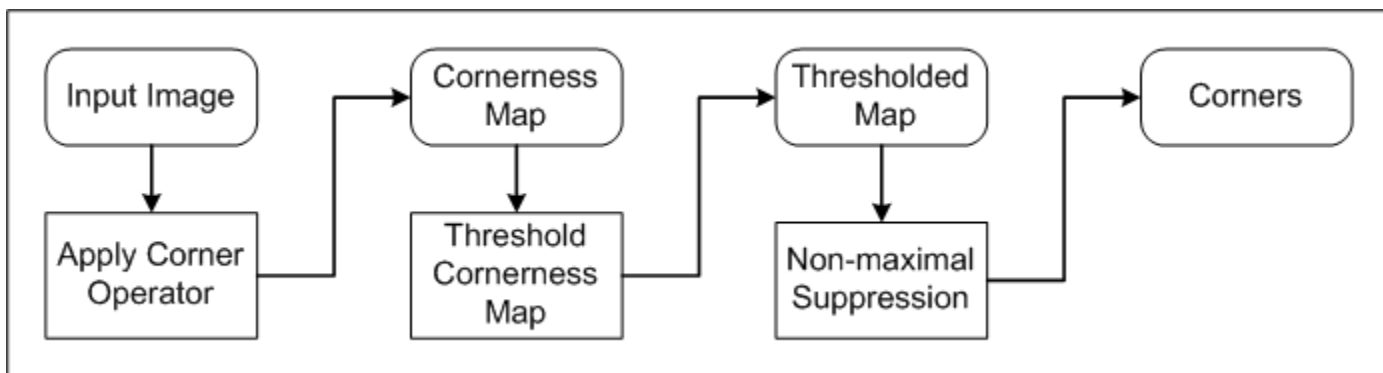


Figure 4.1: Flowchart for interest point corner detectors

The results of applying each of these steps is illustrated in Figure 4.2.

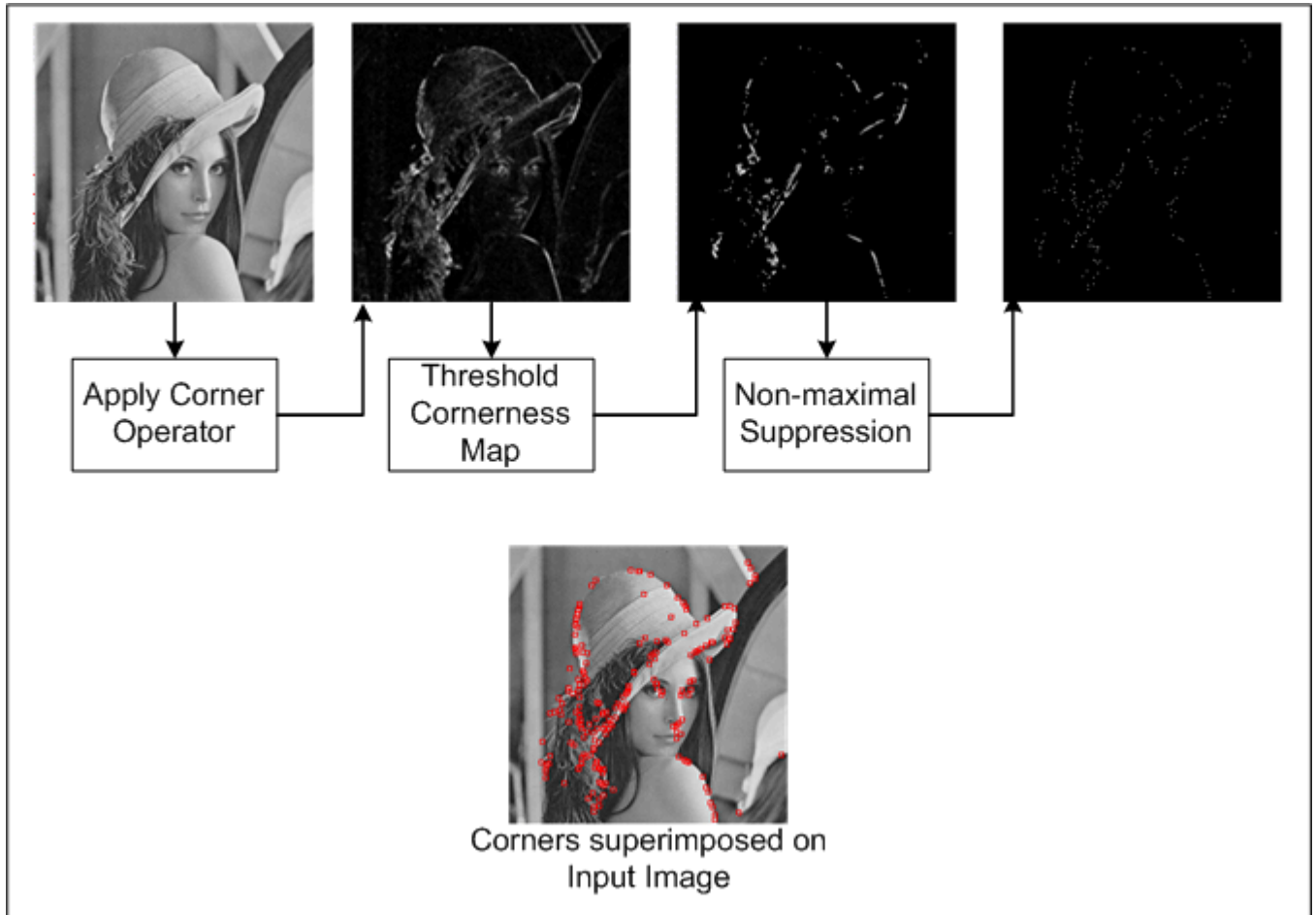


Figure 4.2: Example of steps in a typical interest point corner detector

Moravec Operator

Introduction

This operator was developed by [Hans P. Moravec](#) in 1977 for his research involving the navigation of the Stanford Cart [\[1,2\]](#) through a clustered environment. Moravec defined the concept of "points of interest" as being distinct regions in images and concluded these interest points could be used to find matching regions in consecutive image frames. This was a vital low-level processing step that allowed him to determine the existence and location of objects in the vehicle's environment.

The Moravec operator is considered a corner detector since it defines interest points as points where there is a large intensity variation in every direction. This is the case at

corners. However, Moravec was not specifically interested in finding corners, just distinct regions in an image that could be used to register consecutive image frames. Many have commended this relaxation in the "definition" of what a corner is, since the concept of a corner is not well-defined for gray scale images.

Discussion

How is intensity variation V measured at a particular position P in the image? Moravec proposed measuring the intensity variation by placing a small square window (typically, 3x3, 5x5, or 7x7 pixels) centered at P and then shifting this window by one pixel in each of the eight principle directions (horizontally, vertically, and four diagonals). The intensity variation for a given shift is calculated by taking the sum of squares of intensity differences of corresponding pixels in these two windows. Figure 5.1 shows this calculation for a diagonal shift on an isolated black pixel (intensity equal to 0) on a white background (intensity equal to 255) and on an ideal corner. The red square indicates the original window and the blue box indicates the shifted window. Intensity variation at P is the *minimum* intensity variation calculated over the eight principle directions.

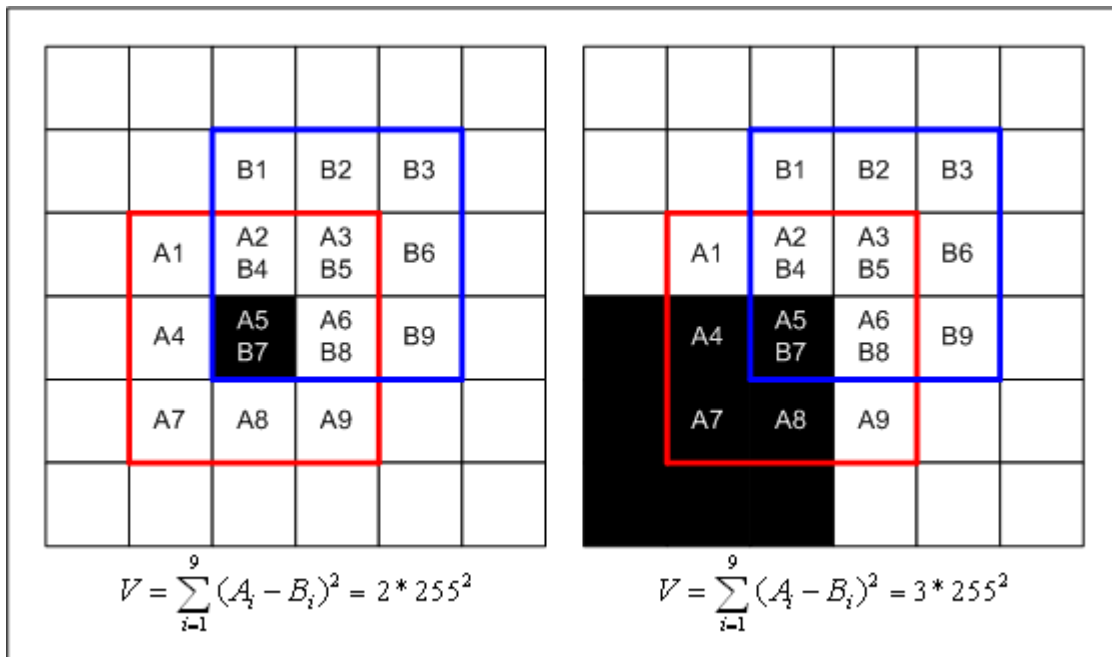


Figure 5.1: Calculations of intensity variation for a 3x3 window in the upper right diagonal direction

To understand why the Moravec operator is a corner detector consider Figure 5.2. Figure 5.2 shows windows at four different types of positions. Position A is interior to an object (or on the background) where it is assumed the image intensity will be relatively constant within the window, so shifting the window in any direction results in only a small intensity variation. For a window straddling an edge, as in position B, shifting the window perpendicular to the edge will result in a large intensity variation, but shifting it

along the edge will result in only a small intensity variation. Both positions C and D, which respectively correspond to a corner and an isolated pixel, will give a large intensity variation for all shift directions. This shows that the Moravec operator is indeed a corner detector, but that it may be sensitive to detecting isolated pixels as corners. Note that having a large intensity variation in every direction is equivalent to the shift direction giving the minimum intensity variation being large.

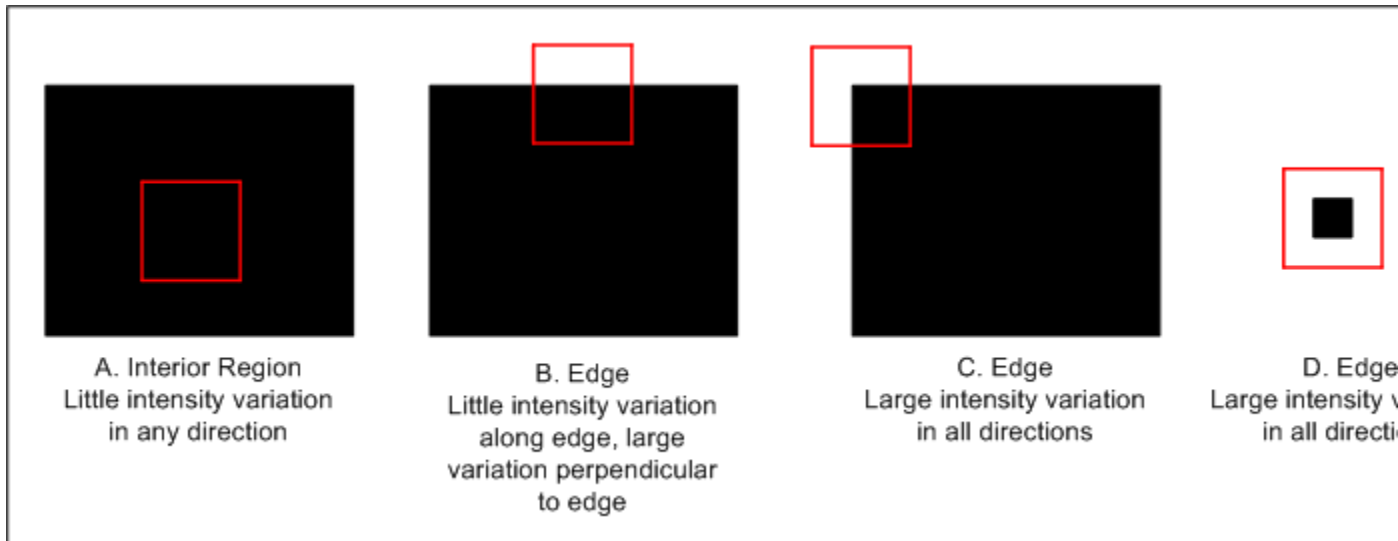


Figure 5.2: Different cases for the Moravec operator

The Moravec operator can be used to give a measure of cornerness to each pixel in the image. This measure is the minimum intensity value found over the eight shift directions. Applying the Moravec operator to each pixel in an image creates a *cornerness map*. Figure 5.3 shows the cornerness map for a simple image when a 3x3 window is used. This cornerness map illustrates several important points of the Moravec operator (the cornerness values shown in Figure 5.3 have all been divided by 255^2 so a value of 2 actually represents $2 \cdot (255^2)$):

1. the corner is a local maximum
2. the cornerness value of the isolated pixel is the same as the cornerness value of the corner
3. there is a region around the border of the image where the Moravec operator can not be directly applied (indicated by X's in Figure 5.3)

X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
X	X	0	0	0	0	0	0	0	0	0	1	1	1	X	X
X	X	0	0	0	0	0	1	1	0	0	1	2	1	X	X
X	X	0	0	0	0	0	2	1	0	0	1	1	1	X	X
X	X	0	0	0	0	0	0	0	0	0	0	0	0	X	X
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Figure 5.3: Cornerness map produced from Moravec operator

The first point indicates that corners are the local maximum in the cornerness map. Local maxima can be identified using non-maximal suppression. However, this will result in the isolated pixel being detected as a corner. For this reason, the Moravec corner detector is consider sensitive to noise. Using a larger window size makes the algorithm more robust to noise as the true corner will have a larger intensity variation than the isolated pixel. However, the isolated pixel will still be a local maximum. Since every pixel is assigned a cornerness measure, it is easy to imagine that in a grayscale image there will be many local maxima that do not correspond to corners. This problem is overcome by setting all cornerness values below a certain threshold to zero. Choosing this threshold is difficult as it must be set high enough to avoid these false corners, but low enough to retain as many true corners as possible. Finally, it is typical to ignore pixels around the border where the Moravec operator can not be applied (since one or more of the shifted windows would "fall" off the image) by simply setting their cornerness value to 0.

Algorithm

The Moravec corner detector is stated formally below:

Denote the image intensity of a pixel at (x, y) by $I(x, y)$.

Input: grayscale image, window size, threshold T

Output: map indicating position of each detected corner

1. For each pixel (x, y) in the image calculate the intensity variation from a shift (u, v) as:

$$V_{u,v}(x, y) = \sum_{\forall a, b \text{ in the window}} \left(I(x+u+a, y+v+b) - I(x+a, y+b) \right)^2$$

where the shifts (u, v) considered are:

$$(1, 0), (1, 1), (0, 1), (-1, 1), (-1, 0), (-1, -1), (0, -1), (1, -1)$$

2. Construct the cornerness map by calculating the cornerness measure $C(x, y)$ for each pixel (x, y) :

$$C(x, y) = \min(V_{u,v}(x, y))$$

3. Threshold the interest map by setting all $C(x, y)$ below a threshold T to zero.
4. Perform non-maximal suppression to find local maxima.

All non-zero points remaining in the cornerness map are corners.

Results on Test Images

The Moravec operator was applied to the three test images discussed in the [Introduction](#).

Figure 5.4 shows the Moravec operator with a 3x3 window and a threshold near zero applied to the artificial test image. The Moravec operator does detect all of the corners, but also responds too readily to diagonal edges. It should not be surprising that the Moravec operator assigns a large cornerness measure to diagonal edges. Along an edge, there is relatively large intensity variation in all directions except in the direction parallel to the edge. Since the Moravec operator only looks for the minimum intensity variation in the eight principle directions, any edge not in one of these eight directions will be assigned a relatively large cornerness measure.

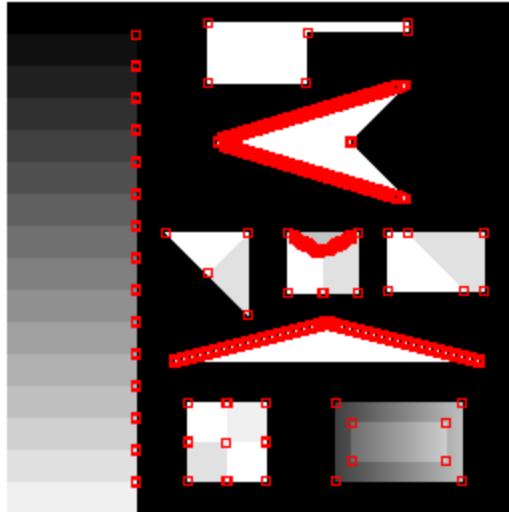


Figure 5.4: Moravec operator applied to Artificial Test Image

Figure 5.5 shows the Moravec operator applied to the blocks test image. A 3x3 window was used and the threshold was chosen in order to detect most of the corners while trying to minimize the number of false corners detected. The Moravec operator does do a reasonable job of finding the majority of true corners, but again it can be seen that the operator responds readily to diagonal edges.

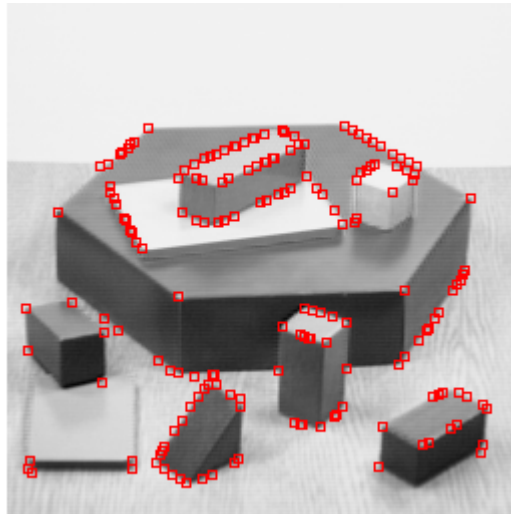


Figure 5.5: Moravec operator applied to Blocks Test Image

Figure 5.6 shows the Moravec operator applied to the house test image. A 3x3 window was used and the threshold was manually set high enough to avoid detecting corners due to the texture of the house. Again, the most notable aspect of this test image is the large number of 'corners' detected along diagonal edges.

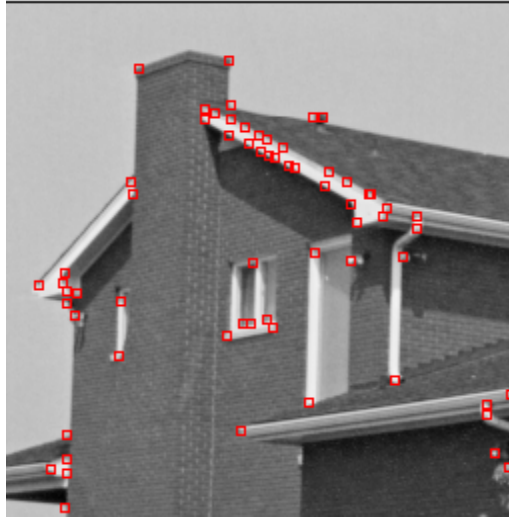


Figure 5.6: Moravec operator applied to House Test Image

The above results were obtained using a 3x3 window and manually picking a threshold for each image that gave desirable results. To see the effect of different window sizes and threshold values use the [Corner Detection Applet](#).

Limitations

As indicated by its performance on the test images, the Moravec operator suffers from a number of problems. The most significant of these are explained here in detail as they provide the motivation for the [Harris/Plessey corner detector](#).

Anisotropic Response of Operator

The response is [anisotropic](#) as the intensity variation is only calculated at a discrete set of shifts (namely, in the eight principle directions). Therefore, the operator is not rotationally invariant as shown in Figure 5.7. This will cause the corner detector to have a poor repeatability rate.

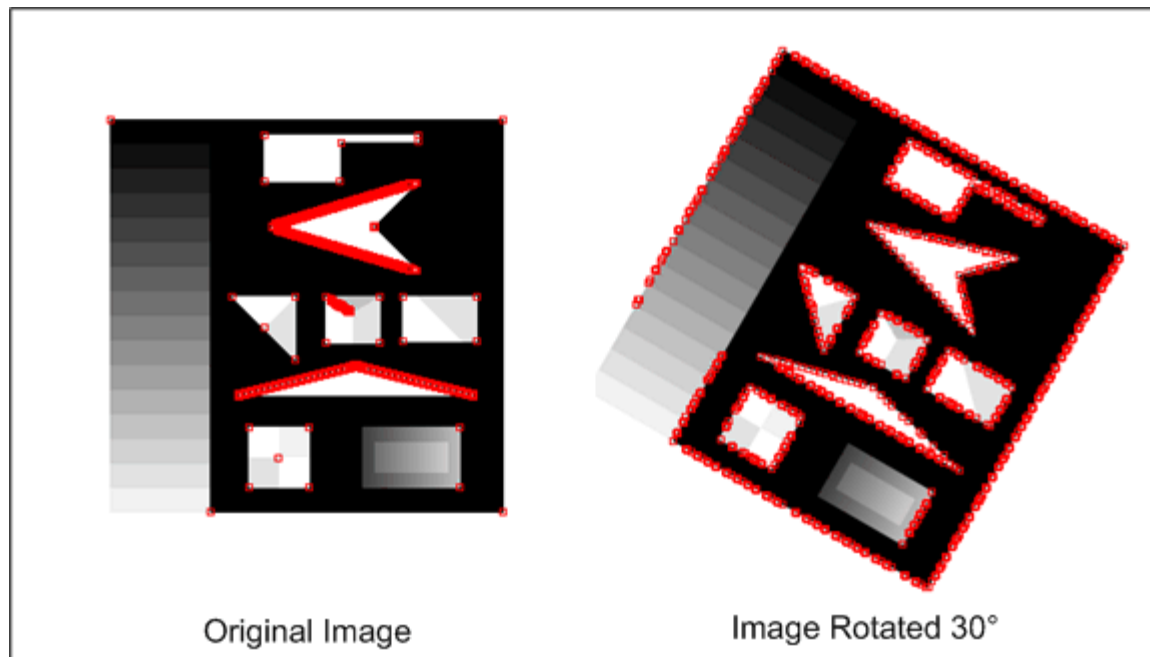


Figure 5.7: Rotational instability of Moravec operator due to anisotropic response

Noisy Response

The window used by Moravec is square and binary. To achieve a more accurate estimate of the local intensity variation, a circular window is desirable so that the Euclidean distance from the center pixel to the edge of the window is the same in all directions. Even on a discrete grid, the use of a circular window will result in an improved estimate of the local intensity variation. Figure 5.8 illustrates a 7x7 square window and a circular window with a diameter of 7 and shows the maximum discrepancy in the Euclidean distances for both of these windows.

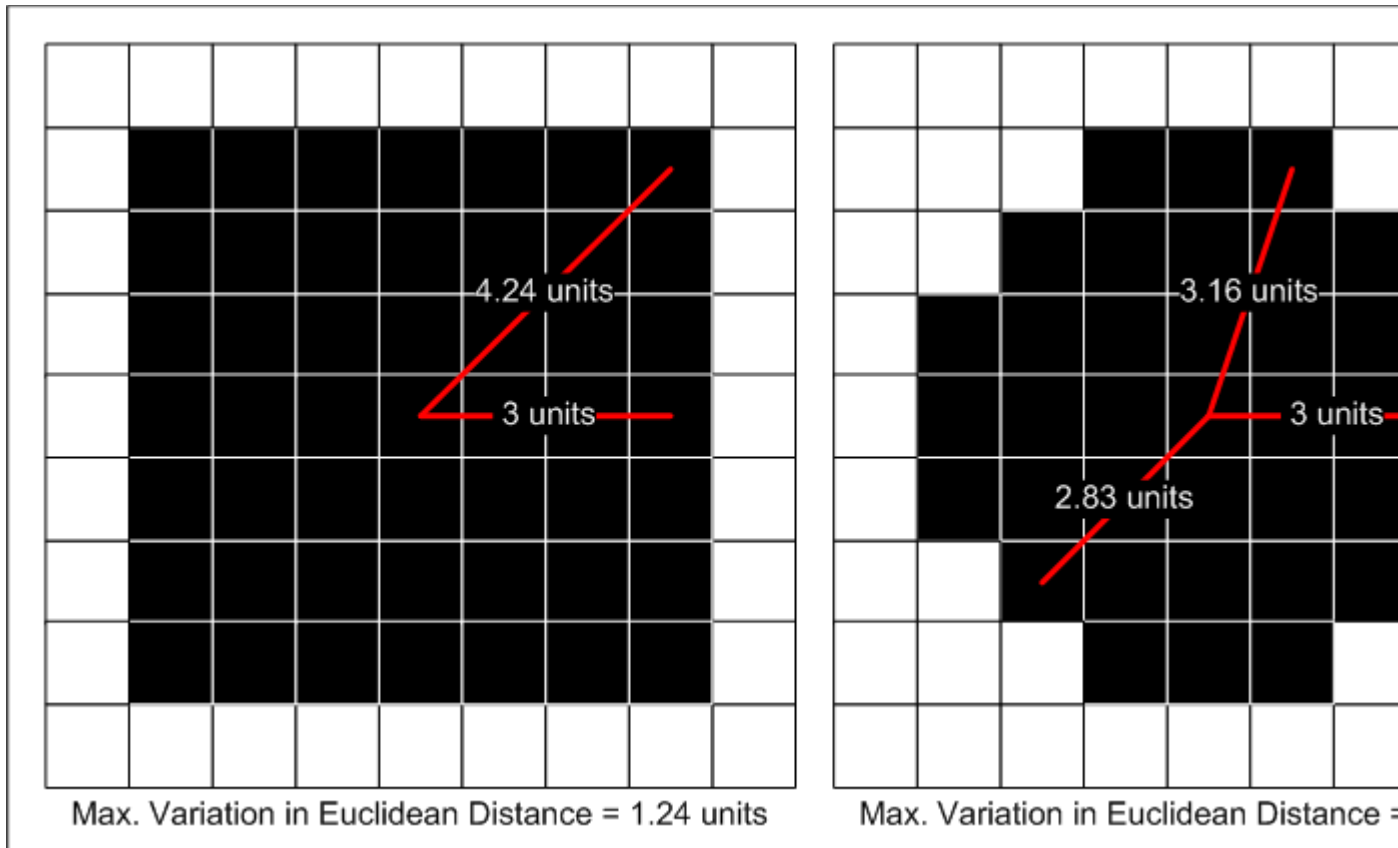


Figure 5.8: Reduced discrepancy in the Euclidean distance from the center of the window to the edge for a circular window

Estimation of the local intensity variation can be further improved by placing more weight on the intensity differences of pixel pairs near the center of the window. Intuitively, the difference of pixel pairs near the center are a better indication of the local variance so should have more effect on the estimate.

These two desirable properties - having a circular window and higher weights for the intensity differences of pixel pairs near the center - suggest the use of a [Gaussian window](#). Figure 5.9 shows the values commonly used for a 5x5 Gaussian window (it is square, but values at the corners have very small weight).

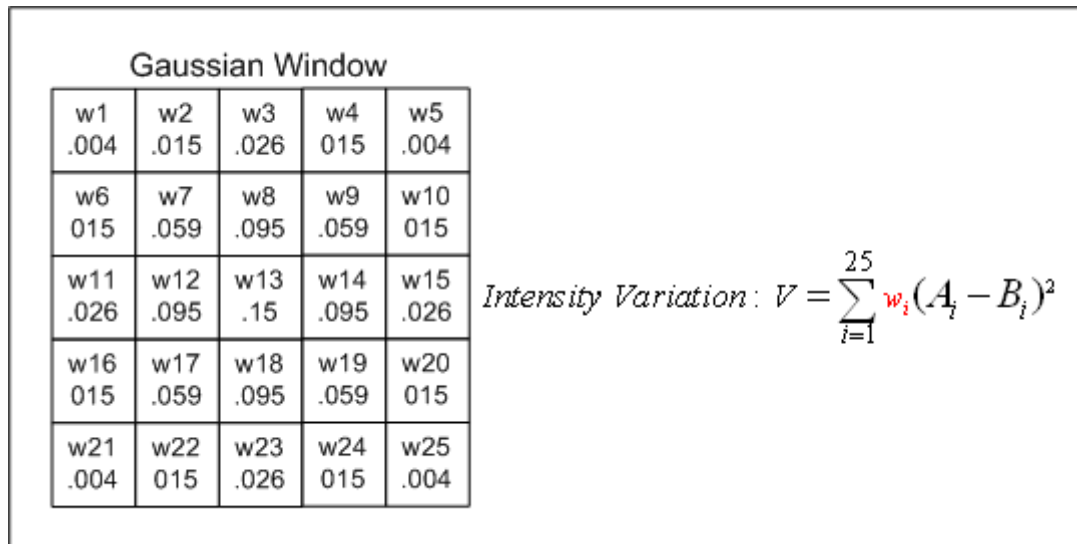


Figure 5.9: Common values for a discrete 5x5 Gaussian window

Large Response to Edges

By definition edges in gray scale images are places where the magnitude of the gradient is large. That is, edges are places where the intensity varies rapidly in some direction. It has been found that the Moravec operator responds too readily to edges since any imperfections in an edge due to noise, pixilation, or intensity quantization may result in the local minimum intensity variation over all shift directions being relatively large. As shown in the [Results on Test Images](#) section, edges at certain orientations will also respond too readily due to the intensity variation only being calculated in the eight principle directions. Figure 5.10 gives the same simple image as in Figure 5.3 except in this image one of the pixels along the edge is white (say, due to noise). The cornerness measure of pixels adjacent to the corrupt pixel are as large as at the true corner and therefore cannot be corrected by thresholding.

X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
X	X	0	0	0	0	0	0	0	0	0	1	1	1	X	X
X	X	1	2	1	0	0	1	1	0	0	1	2	1	X	X
X	X	2	2	2	0	0	2	1	0	0	1	1	1	X	X
X	X	1	1	1	0	0	0	0	0	0	0	0	0	X	X
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Figure 5.10: False detection of a corner along an edge due to noise

Conclusions

Moravec's corner detector is a relatively simple algorithm that was used by Moravec and others, but is now generally considered obsolete. It is not rotationally invariant (a property prevalent even in more modern corner detectors) as the response is [anisotropic](#), is considered to have a noise response, and is susceptible to reporting false corners along edges and at isolated pixels so is sensitive to noise. However, it is computationally efficient which was critical for Moravec as he was interested in a real-time application and had minimal computational power at his disposal.

Due to the increase in computational power over the last few decades, most applications now employ corner detectors with better performance although at the cost of increased computation. One of the most widely used corner detectors today is the [Harris/Plessey corner detector](#). This corner detector is computationally demanding, but directly addresses many of the limitations of the Moravec corner detector.

Harris/Plessey Operator

Introduction

This operator was developed by Chris Harris and Mike Stephens in 1988 [\[3\]](#) as a low-level processing step to aid researchers trying to build interpretations of a robot's environment based on image sequences. Specifically, Harris and Stephens were

interested in using motion analysis techniques to interpret the environment based on images from a single mobile camera. Like [Moravec](#), they needed a method to match corresponding points in consecutive image frames, but were interested in tracking both corners and edges between frames.

Harris and Stephens developed this combined corner and edge detector by addressing the limitations of the Moravec operator. The result is a far more desirable detector in terms of detection and repeatability rate at the cost of requiring significantly more computation time. Despite the high computational demand, this algorithm is widely used in practice.

The literature refers to this detector as both the Harris corner detector and the Plessey corner detector.

Discussion

The Plessey operator differs from the [Moravec](#) operator in how the measurement of local [autocorrelation](#) is estimated. This measurement allows the variation of the autocorrelation (i.e. intensity variation) over all different orientations to be obtained. The rationale for the Plessey operator follows from addressing the limitation of the Moravec operator.

Anisotropic Response of Operator

The [Moravec](#) operator has an [anisotropic](#) response as the intensity variation is only calculated at a discrete set of shifts (namely, in the eight principle directions). To remove this limitation, a function is needed that allows the intensity variation to be measured in any direction. Harris and Stephen performed an analytic expansion of the Moravec operator to derive such a function. This derivation is not complicated, but provides little insight into the derivation of the Plessey operator. Here a more intuitive, but less mathematically rigorous, approach is taken to arrive at the desired function.

The [Prewitt operator](#) is commonly used to approximate the gradient of an image. However, in many applications the first-order gradient is approximated using the simple formulas given in Figure 6.1. With these gradient formulas in mind, it is reasonable to suggest that taking the sum of differences between corresponding pixels in the two [Moravec](#) windows (see Figure 5.1) is a reasonable approximation of the gradient. As shown in Figure 6.2, the intensity variation calculation for the [Moravec](#) operator can now be approximated using the gradient of the image.

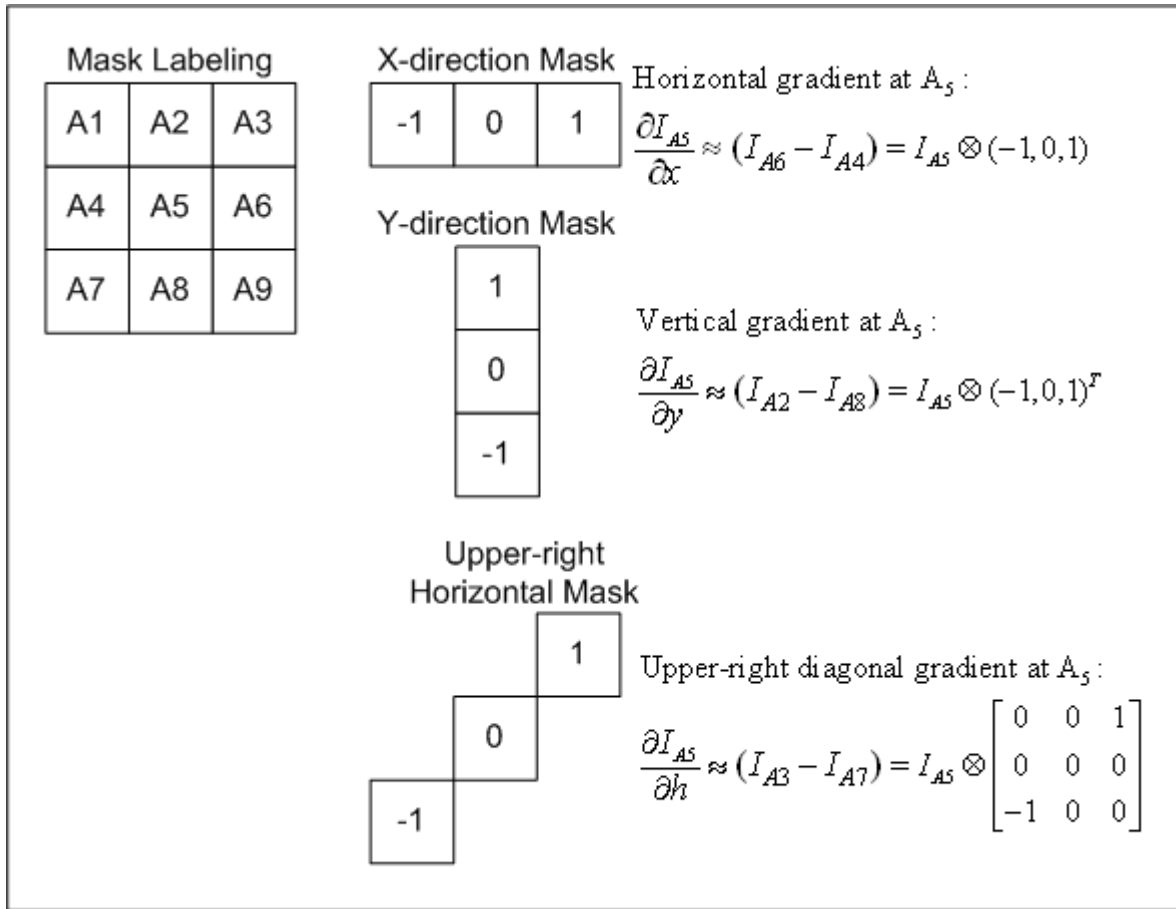


Figure 6.1: Simple discrete gradient approximations

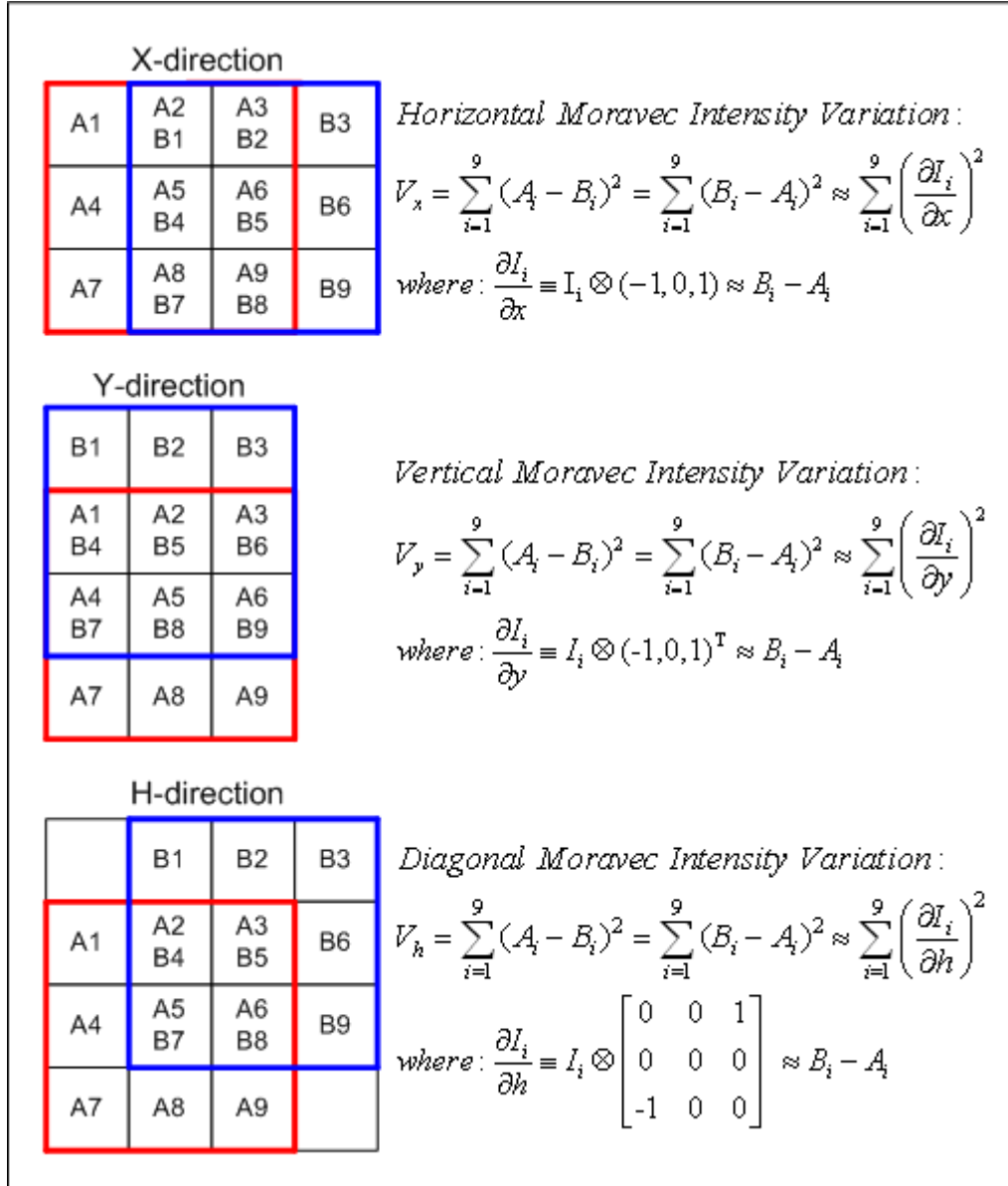


Figure 6.2: Moravec operator approximated using the image gradient

The above analysis indicates that the intensity variation can be written as a function of the gradient of the image. For an arbitrary shift (u,v) we can state the intensity variation as:

$$V_{u,v}(x,y) = \sum_{W \text{ in the window centered at } (x,y)} \left(u \frac{\partial I_i}{\partial x} + v \frac{\partial I_i}{\partial y} \right)^2 \quad (1)$$

where: $\frac{\partial I_i}{\partial x}$ and $\frac{\partial I_i}{\partial y}$ are calculated as in Figure 6.2

For a shift in the x-direction where $(u,v)=(1,0)$ or y-direction where $(u,v)=(0,1)$ we have precisely the approximation for the intensity variation derived in Figure 6.2. The results obtained by performing an analytic expansion of the Moravec operator are identical to equation (1) except that there are terms representing higher order properties which Harris and Stephen ignore.

Let us be clear that equation (1) is not equal to the intensity variation calculation performed by the Moravec operator. However, it is suggested by the Moravec operator and performs a similar measure of intensity variation. The benefit of this new measure for intensity variation is that it can measure intensity variation in any direction by appropriate choice of u and v . Critics of the Plessey operator (specifically [6]) have pointed out that although equation (1) does allow an *estimate* of the intensity variation to be calculated in any direction, it is still [anisotropic](#) as these estimates are based on the horizontal and vertical gradients (see the [Limitations](#) section for an example of the anisotropic response of the Plessey operator). Despite being anisotropic, equation (1) allows for a strong edge detector as indicated by the popularity of the Plessey corner detector.

How should u and v be selected for a given direction? Figure 6.3 shows that u and v are simply the x and y distances required to construct a line within the unit triangle that is in the desired direction (i.e the unit circle under the Manhattan or [city block](#) distance). Clearly measuring the intensity variation in all directions in order to estimate a measure of local [autocorrelation](#) is not possible (there are infinitely many directions). This apparent difficulty is addressed in the [Large Response to Edges](#) section below.

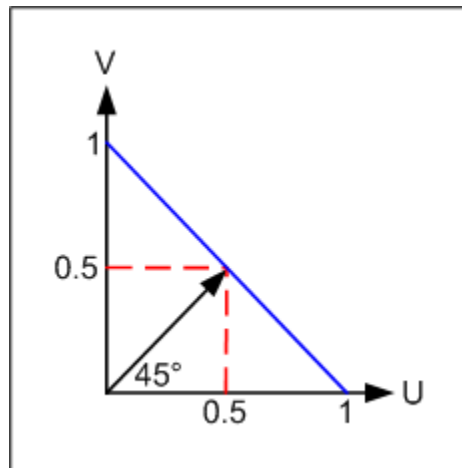


Figure 6.3: Selecting u and v values for an arbitrary direction

Noisy Response

Since the window used by the [Moravec](#) operator is square and binary, the estimate of the intensity variation is considered noisy. Having a square window results in the Euclidean distance from the center pixel of the window to the edge of the window varying for

different directions. This is easily resolved by using a circular window. Being binary puts equal emphasis on all intensity variation measures regardless of their distance from the center of the window. Intuitively, more weight should be put on measurements made closer to the center of the window. These two limitations of the Moravec operator are addressed by using a [Gaussian window](#). For a more detailed discussion on these issues please refer to the *Limitation* section of the Moravec operator.

The measurement of the intensity variation can now be done as illustrated in Figure 6.4 for a horizontal shift and a 3x3 Gaussian window.

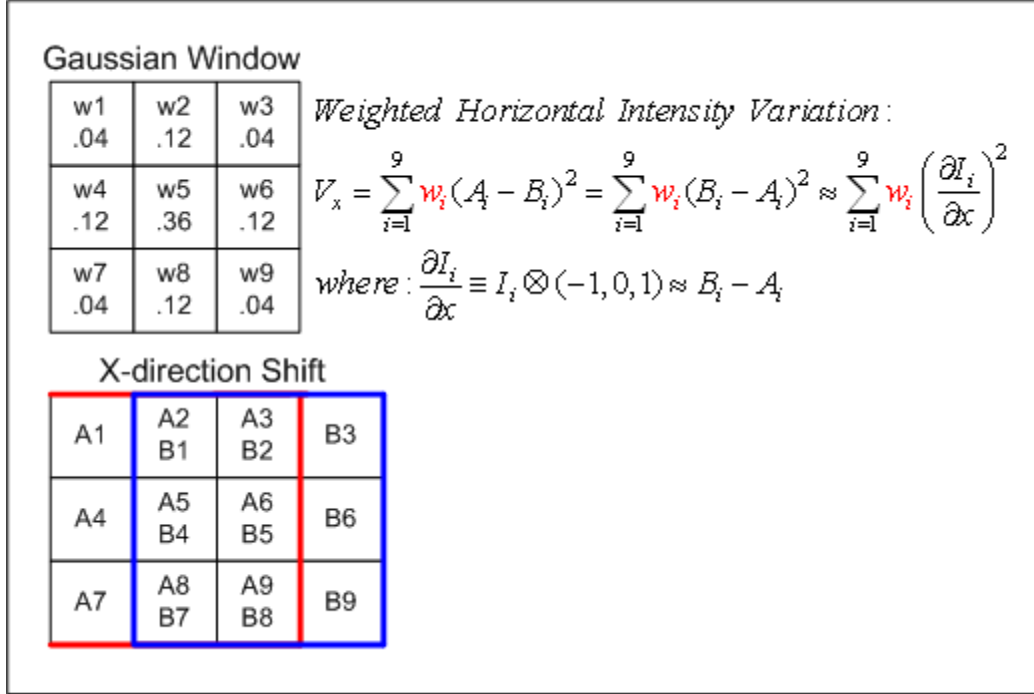


Figure 6.4: Calculation of a vertical shift using a Gaussian window

This intensity variation measure can be stated as:

$$V_{u,v}(x,y) = \sum_{\forall i \text{ in the window centered at } (x,y)} w_i \left(u \frac{\partial I_i}{\partial x} + v \frac{\partial I_i}{\partial y} \right)^2 \quad (2)$$

where: w_i is the weight of the Gaussian window at position i

Large Response to Edges

It is known the [Moravec](#) operator responds too readily to edges since any imperfections in an edge due to noise, pixilation, or intensity quantization may result in the local minimum intensity variation over all shift directions being relatively large. The Plessey operator addresses this limitation by reformulating the cornerness measure to consider the

variation between the intensity variation measures in different directions. Recall that the Moravec corner detector considers only the minimum intensity variation measure found over all considered shift directions.

Equation (2) can be re-write as:

$$\begin{aligned}
 V_{u,v}(x,y) &= \sum_{W \text{ is the window centered at } (x,y)} w_i \left(u \frac{\partial I}{\partial x} + v \frac{\partial I}{\partial y} \right)^2 \\
 &= \sum_{W \text{ is the window centered at } (x,y)} w_i \left(u^2 \frac{\partial^2 I}{\partial x^2} + 2uv \frac{\partial^2 I}{\partial x \partial y} + v^2 \frac{\partial^2 I}{\partial y^2} \right) \\
 &= Au^2 + 2Cuv + Bv^2 \\
 \text{where: } A &= \left(\frac{\partial^2 I}{\partial x^2} \right) \otimes w, B = \left(\frac{\partial^2 I}{\partial y^2} \right) \otimes w, C = \left(\frac{\partial^2 I}{\partial x \partial y} \right) \otimes w \\
 &\quad \otimes \text{ is the convolution operator}
 \end{aligned}$$

Harris and Stephen noticed that this could be written as a matrix equation:

$$\begin{aligned}
 V_{u,v}(x,y) &= Au^2 + 2Cuv + Bv^2 \\
 &= \begin{bmatrix} u & v \end{bmatrix} M \begin{bmatrix} u \\ v \end{bmatrix} \\
 \text{where: } M &= \begin{bmatrix} A & C \\ C & B \end{bmatrix}
 \end{aligned}$$

This form of equation (2) is interesting as the matrix M now contains all the differential operators describing the geometry of the [image surface](#) at a given point (x,y). The [eigenvalues](#) of M will be proportional to the principle curvatures of the image surface and form a rotationally invariant description of M. However, since the components of M are estimated using only the horizontal and vertical gradients, they are not rotationally invariant.

Let us consider the four different types of window positions that were considered by the [Moravec](#) operator. These different window positions are shown in Figure 6.5. Position A is interior to an object (or on the background) where it is assumed image intensity will be relatively constant within the window. Since there is little curvature in the surface within this window both the eigenvalues will be relatively small. For local windows straddling an edge, as in position B, there is significant curvature perpendicular to the edge and very little curvature along the edge so one of the eigenvalues will be large and the other small. Both positions C and D, which correspond to a corner and isolated pixel, will have significant curvature in both directions so both eigenvalues will be large.

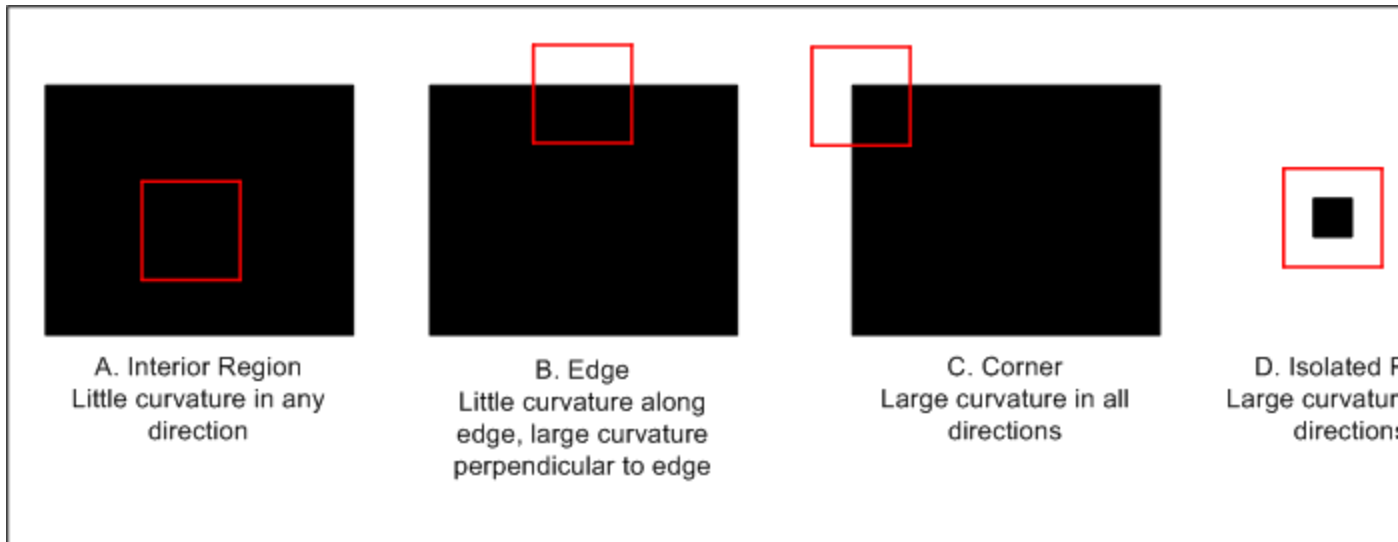


Figure 6.5: Different cases for the Plessey operator

Let the [eigenvalues](#) of M be denoted by λ_1 and λ_2 . The above analysis indicates the plane described by λ_1 and λ_2 can be divided into 3 distinct regions as shown in Figure 6.6.

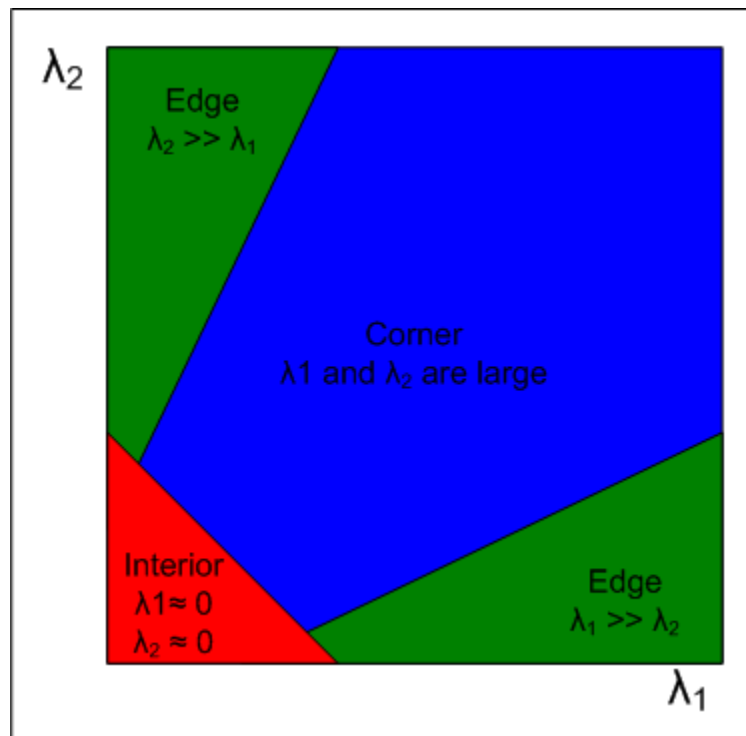


Figure 6.6: Division of eigenvalue space into distinct feature regions

To classify pixels, the three lines dividing the plane in Figure 6.6 must be selected. However, it is useful to have a measure of cornerness instead of just a discrete label for each pixel. A measure of cornerness allows edges to be thinned and improves

localization of corners by taking local maxima to be the true position of corners or edges. Harris and Stephens proposed the following cornerness measure:

$$C(x, y) = \det(M) - k(\text{trace}(M))^2$$

$$\det(M) = \lambda_1 \lambda_2 = AB - C^2$$

$$\text{trace}(M) = \lambda_1 + \lambda_2 = A + B$$

$$k = \text{constant}$$

Figures 6.7-6.9 show contours of $C(x, y)$ for $k=0.2$, $k=0.1$, and $k=0.05$, respectively. Empirical testing has indicated that values of $k = 0.04$ to 0.06 give the best results. Comparing these plots to Figure 6.6 indicates that edges have a negative cornerness measure while corners and interior points have a positive cornerness measure. A threshold value is required to distinguish between corners and interior points. From the figures it is clear that the interior points have a very small cornerness measure. In practice, the threshold must be set high enough to avoid the detection of false corners which may have a relatively large cornerness value due to noise. As for the [Moravec](#) operator, corners are the local maximum in the thresholded cornerness map.

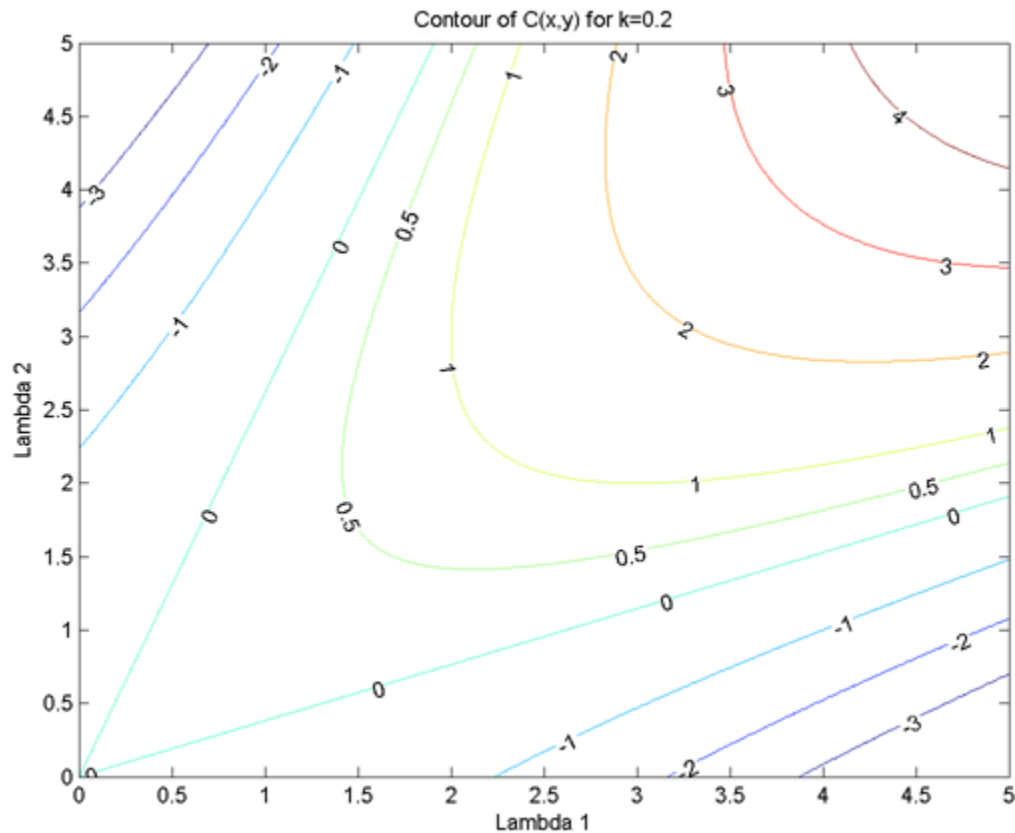


Figure 6.7: Contour plot of $C(x, y)$ for $k=0.2$

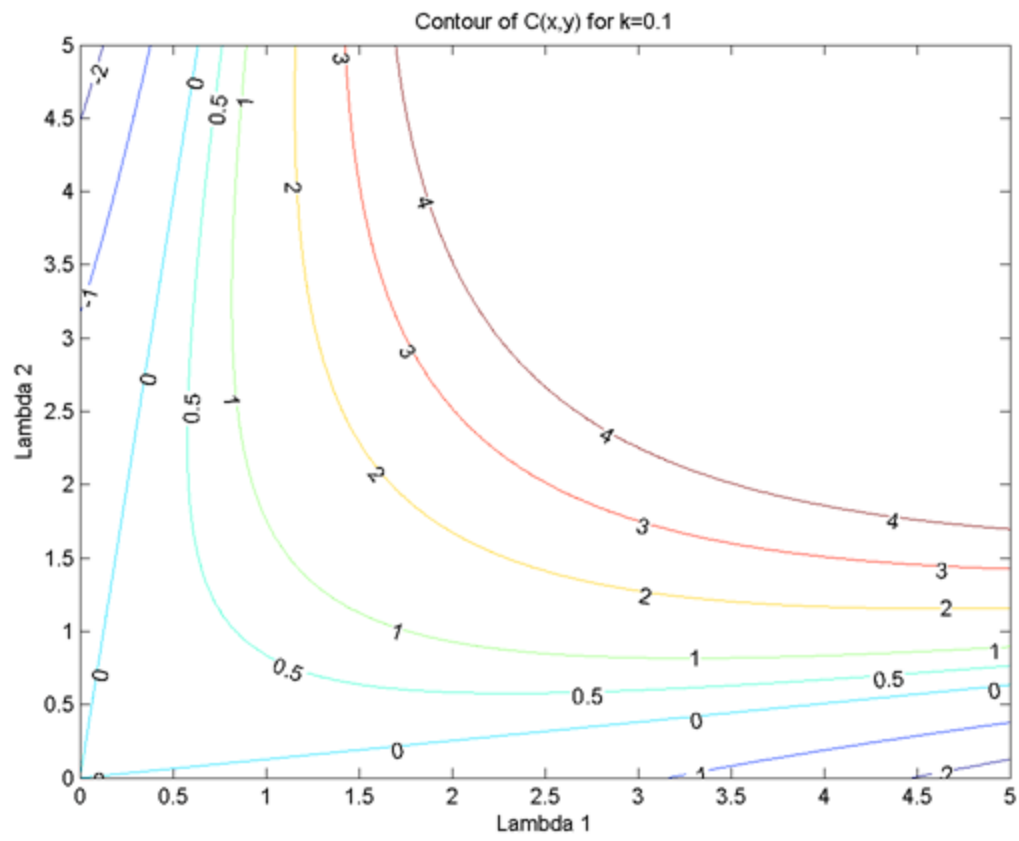


Figure 6.8: Contour plot of $C(x, y)$ for $k=0.1$

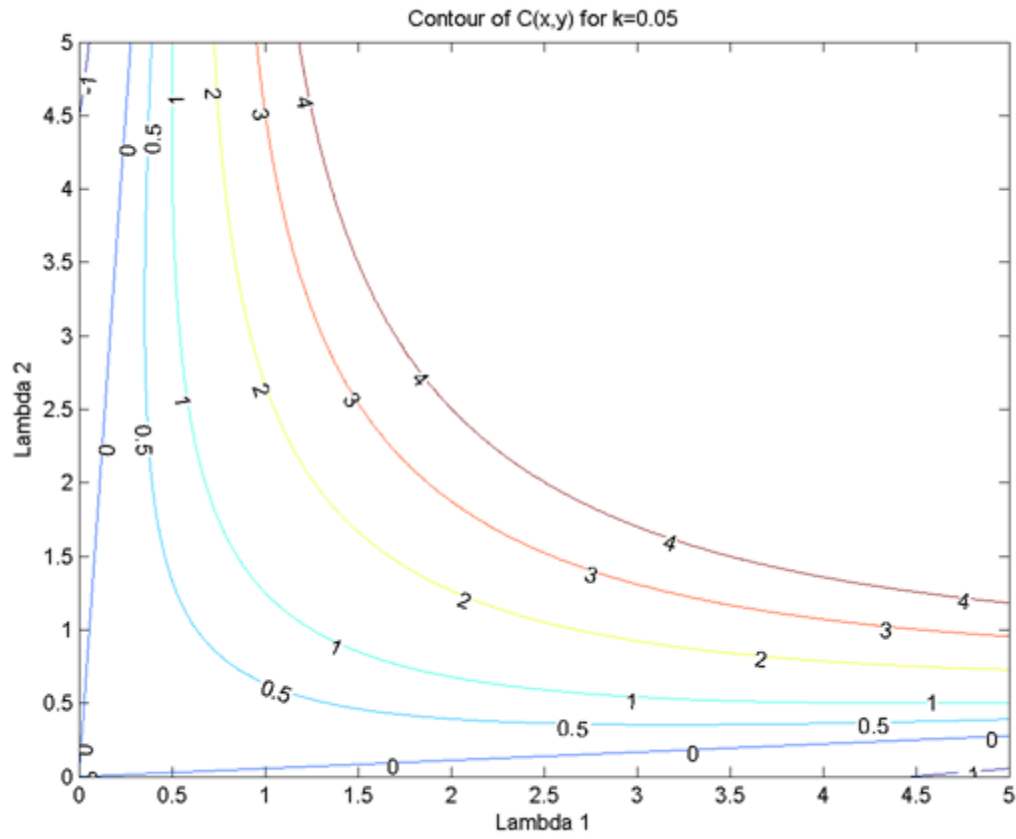


Figure 6.9: Contour plot of $C(x, y)$ for $k=0.05$

Algorithm

The Plessey corner detector is stated formally below:

Input: grayscale image, Gaussian variance (window typically has a radius of 3 times the standard deviation), k value, threshold T

Output: map indicating position of each detected corner

1. For each pixel (x, y) in the image calculate the autocorrelation matrix M :

$$M = \begin{bmatrix} A & C \\ C & B \end{bmatrix}$$

$$\text{where: } A = \left(\frac{\partial f}{\partial x} \right)^2 \otimes w, B = \left(\frac{\partial f}{\partial y} \right)^2 \otimes w, C = \left(\frac{\partial f}{\partial x} \frac{\partial f}{\partial y} \right) \otimes w$$

\otimes is the convolution operator

w is the Gaussian window

2. Construct the cornerness map by calculating the cornerness measure $C(x, y)$ for each pixel (x, y) :

$$C(x, y) = \det(M) - k(\text{trace}(M))^2$$

$$\det(M) = \lambda_1 \lambda_2 = AB - C^2$$

$$\text{trace}(M) = \lambda_1 + \lambda_2 = A + B$$

$$k = \text{constant}$$

3. Threshold the interest map by setting all $C(x, y)$ below a threshold T to zero.

4. Perform non-maximal suppression to find local maxima.

All non-zero points remaining in the cornerness map are corners.

Results on Test Images

The Plessey operator was applied to the three test images discussed in the [Introduction](#).

Figure 6.10 shows the Plessey operator applied to the artificial test image (variance = 0.7, $k=0.14$, threshold=10). The Plessey operator detects all of the corners except one, but has poor localization at many of the corners and does detect a few false corners. It has a higher detection rate than the [Moravec](#) operator as it finds far fewer false corners.

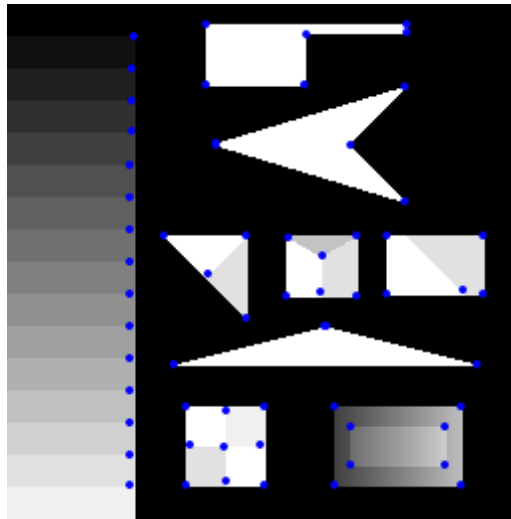


Figure 6.10: Plessey operator applied to Artificial Test Image

Figure 6.11 shows the Plessey operator applied to the blocks test image. Parameters were chosen in order to detect most of the corners while trying to minimize the number of false corners detected. The Plessey operator does a reasonable job of finding the majority of true corners. Comparing these results to the [Moravec](#) operator again indicates that the Plessey operator has a better detection rate, although there are still a number of false corners detected and some corners are missed.

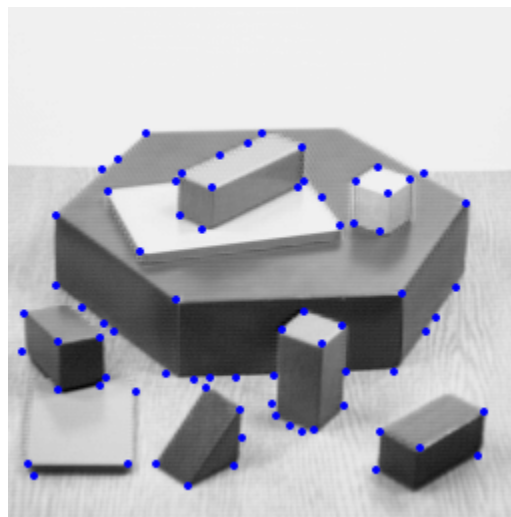


Figure 6.11: Plessey operator applied to Blocks Test Image

Figure 6.12 shows the Plessey operator applied to the house test image. Parameters were manually set high enough to avoid detecting corners due to the texture of the house. Clearly some corners have not been detected, but the results are reasonable and much improved over the [Moravec](#) operator.

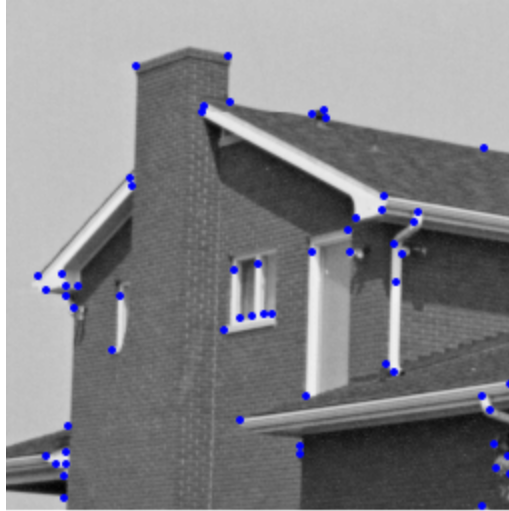


Figure 6.12: Plessey operator applied to House Test Image

The above results were obtained using a 5x5 window and manually picking a variance, k value, and threshold for each image that gave desirable results. To see the effect of different parameter values use the [Corner Detection Applet](#). Results obtained here can also be compared with the results given on the [Detection of Interest Points](#) website.

Limitations

Despite the widespread use of the Plessey corner detection algorithm, it suffers from a number of limitations.

Computationally Demanding

The Plessey corner detector is one of the most computationally demanding corner detectors in use. Comparing the [Moravec](#) and Plessey corner detection algorithms shows they differ only in how the cornerness measure is calculated and this is true for most interest point corner detectors. The Plessey operator is far more expensive to compute than the Moravec operator mainly because it requires convolution with a Gaussian window.

Sensitivity to Noise

Any algorithm using gradient or high-order derivative terms will likely be sensitive to noise. To understand why, consider the effect on the response of the Plessey operator to salt noise in the interior region of a black object. Any pixel affected by the salt noise will have a large gradient in all directions. That is, the salt noise is essentially an isolated pixel so has high intensity variation in all directions which is precisely how a corner is defined under both the [Moravec](#) and Plessey operators. The effect of noise is reduced by

increasing the size of the Gaussian window as this causes more pixels to be considered when calculating the gradient, thus reducing the relative weight of any pixels affected by noise. However, increasing the Gaussian window size increases the computational demand of the algorithm.

Good Localization only at L-Junctions

The Plessey operator has poor localization at all junctions, except L-Junctions. Figure 6.13 demonstrates why the localization is poor at T-junctions where the two "smaller" regions have similar intensity, and the "larger" region has a very different intensity from the two smaller regions. The gradient perpendicular to the edge between the smaller regions will be constant until the Gaussian window starts to enter the larger region. At this point, the gradient perpendicular to the edge between the smaller regions starts to decrease, but the gradient parallel to this edge starts to grow. The cornerness measure is a function of these two gradients and is maximal somewhere along this edge and not at the true T-junction position.

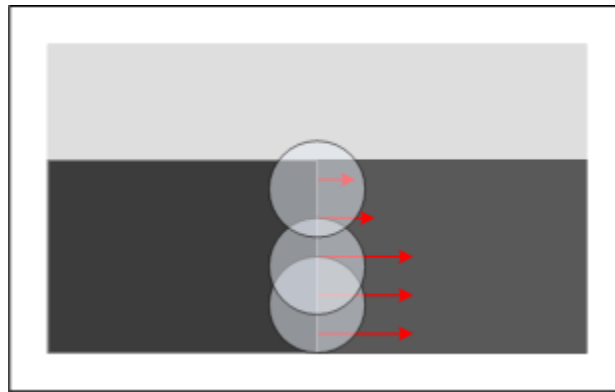


Figure 6.13: Poor localization of Plessey operator at T-junctions

Other factors leading to poor localization are beyond the scope of this website, but can be found in [9].

Anisotropic Response of Operator

Like the [Moravec](#) operator, the Plessey operator has an [anisotropic](#) response. The Plessey operator is commonly misinterpreted to be rotationally invariant (i.e. have an isotropic response) as the autocorrelation matrix M is indeed rotationally invariant. However, the Plessey operator is anisotropic since the elements of M are calculated using only the horizontal and vertical gradients. Figure 6.14 illustrated that the corners detected vary greatly with rotation (both of these images were generated with the same parameter values!). [6] empirically verifies that the response of the Plessey operator varies most with a 45° degree rotation and proposes using derivatives of the Gaussian function to construct the elements of M which results in an isotropic corner detector. As illustrated

in Figure 6.14, a side-effect of this anisotropic response is diagonal edges can be falsely detected as corners.

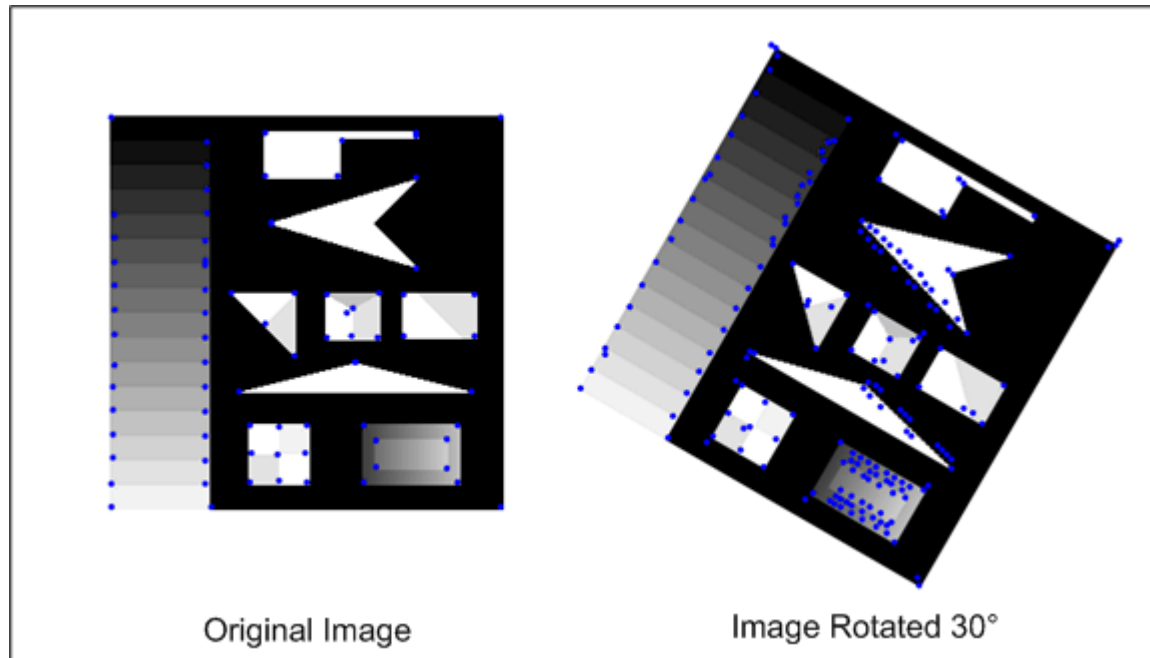


Figure 6.14: Rotational instability of Plessey operator due to anisotropic response

Conclusions

The Plessey corner detector is widely used thanks to its improved detection rate over the [Moravec](#) corner detector and high repeatability rate. However, it is computationally demanding, sensitive to noise since it relies on gradient information, has poor localization on many junction types, and is not rotationally invariant. Sensitivity to noise can be reduced by using a larger window size, but this further increases the computational demand of the algorithm and adversely affect localization. The significance of poor localization of certain junction types is application dependent and the widespread use of this algorithm suggests this limitation can be overcome. Even with its [anisotropic](#) response, it is still more robust in terms of repeatability than the Moravec corner detector and recent modifications to the Plessey algorithm have made the response isotropic which will help ensure the Plessey corner detector remains popular.

Trajkovic Operator (4-Neighbours)

Introduction

This operator was developed by Miroslav Trajkovic and Mark Hedley in 1998 [\[5\]](#) with the intent of obtaining comparable repeatability rates and localization performance as the most popular corner detectors, while requiring a minimum of computation. They

compared their operator to the [Plessey](#) operator (and others) and claim that the repeatability rate is slightly inferior, but that the localization is comparable on L-junctions and improved on other junction types. They empirically show that their operator is five times faster than the Plessey operator and at least three times faster than all of the operators considered, including the SUSAN operator that is already considered computationally efficient.

Trajkovic and Hedley adopt the same definition for a corner as the [Moravec](#) and [Plessey](#) operators: corners are points where the change of image intensity is high in all directions. Like Moravec, they define the cornerness measure as the minimum change of intensity over all possible directions. Performance is improved over the Moravec operator by performing an interpixel approximation in order to estimate the intensity change in all directions (unlike the finite number of directions considered by Moravec). Performance and computational demand are both improved by using a multigrid approach where likely locations of corners are first found in a low resolution version of the original image.

The minimal computational demands of this operator make it well-suited for real-time applications. However, our analysis shows that it is not rotationally invariant, is sensitive to noise, and responds too readily to diagonal edges. Applications requiring a high repeatability rate will likely still favour the Plessey operator, especially with the recent extensions to make it rotationally invariant.

Discussion

Cornerness Measurement

The Trajkovic operator considers a small circular window and all the lines which pass through the center of this circle. Let the center of the circle be denoted by C . Now consider an arbitrary line that passes through C and intersects the boundary of the circular window at P and P' . Denote the intensity at a point X by I_X . This notation is summarized in Figure 7.1.

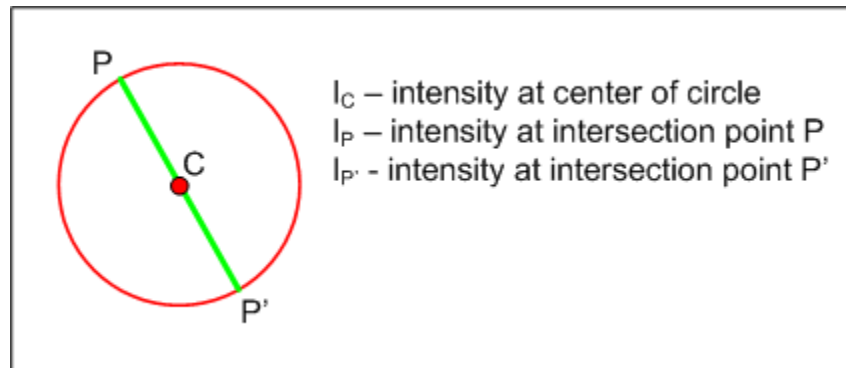


Figure 7.1: Notation for Trajkovic Operator

The cornerness measure for the Trajkovic operator is then given as:

$$C(x,y) = \min((I_P - I_C)^2 - (I_P - I_{P'})^2), \forall P, P'$$

This cornerness measurement can be understood by considering its response to the different cases shown in Figure 7.2:

- *Interior Region* - for cases where the majority of the circular window is within an interior region (i.e. region of near uniform colour) there will be at least one line where the intensity at the center of the circle I_C is approximately equal to I_P and $I_{P'}$. As illustrated by the green lines in Figure 7.1, there is in general several lines where I_C is approximately equal to both I_P and $I_{P'}$. Therefore, the cornerness measure will be low and robust to noise.
- *Edge* - for the case where the center of circle lies just on an edge there will be exactly one line, shown in green, where I_C is approximately equal to both I_P and $I_{P'}$. Since there is only one line where I_C is approximately equal to both I_P and $I_{P'}$ the cornerness measure along edges is susceptible to noise (see [Limitations](#)).
- *Corner* - for the case where the center of the circle is on a corner, for every line at least one of I_P or $I_{P'}$ will not be in the interior region so *should* be different than I_C . Therefore, the cornerness measure will be high. However, this measure is not particularly robust to noise as explored in the [Limitations](#) section.
- *Isolated Pixel* - for the case of an isolated pixel, for every line both I_P and $I_{P'}$ will be different than I_C so the cornerness measure will be high. An isolated pixel is likely the result of noise.

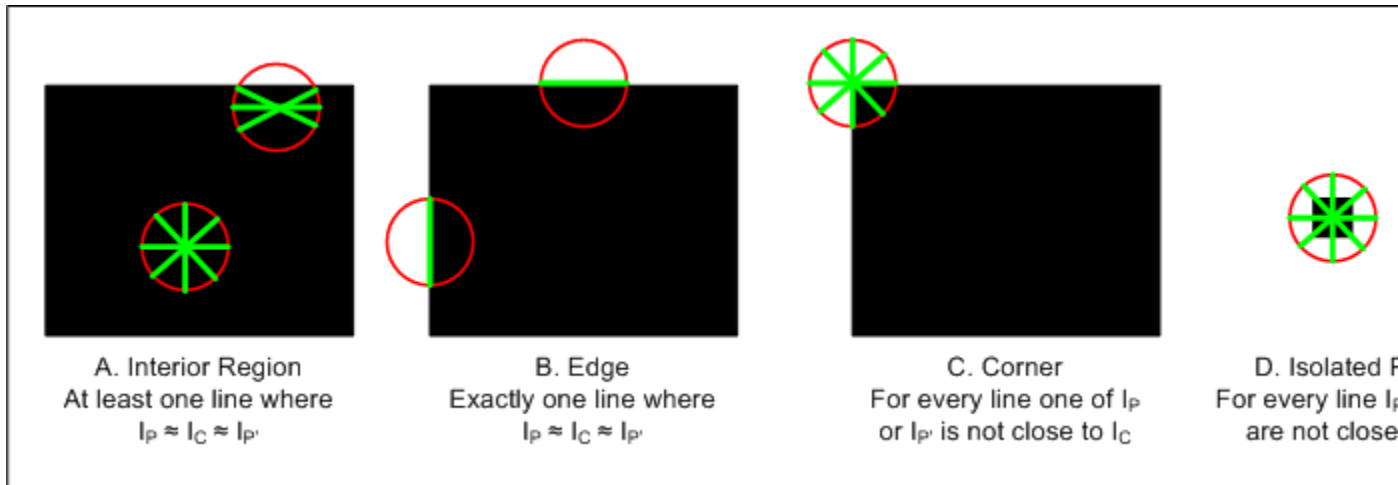


Figure 7.2: Different cases for the Trajkovic Operator

Based on the above analysis, the Trajkovic Operator will only perform well for relatively clean images. In practice, noise can often be handled through [Gaussian smoothing](#).

Calculation of Cornerness Measure (Interpixel Approximation)

How can the Trajkovic cornerness measure be computed? Referring to Figure 7.3, the horizontal (r_A) and vertical (r_B) intensity variations are easily computed and the simple Trajkovic cornerness measure is given by:

$$C_{\text{corn}}(x,y) = \min(r_A, r_B)$$
$$\text{where: } r_A = (I_A - I_C)^2 + (I_E - I_C)^2, r_B = (I_B - I_C)^2 + (I_F - I_C)^2$$

As will be seen later, this simple cornerness measure is used to quickly reject pixels as being potential corner positions.

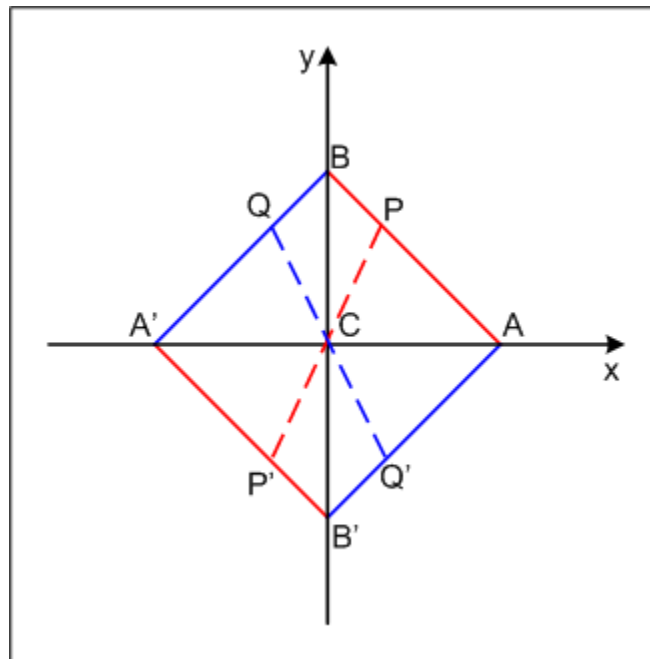


Figure 7.3: Interpixel positions

A method is needed to measure the intensity variation of arbitrary lines through C such as PP' and QQ' in Figure 7.3. This can be done using a linear interpixel approximation. All lines passing through C can be specified by introducing a parameter x that determines the position of points P and Q (thus, P' and Q') along the lines AB and BA', respectively. For example:

- when $x=0$, $P=Q'=A$ and $P'=Q'=A'$
- when $x=1$, $P=Q=B$ and $P'=Q'=B'$
- when $x=0.5$, $P=(\text{halfway along the line AB})$, $P'=(\text{halfway along the line A'B'})$, $Q=(\text{half way along the line BA'})$, and $Q'=(\text{halfway along the line B'A})$.

The cornerness measure can then be stated as:

$$C_{\text{INTERPIXEL}}(x, y) = \min_{z \in \{0,1\}} (r_1(x), r_2(x)) \quad (1)$$

$$\text{where: } r_1(x) = (I_P - I_C)^2 + (I_{P'} - I_C)^2, r_2(x) = (I_Q - I_C)^2 + (I_{Q'} - I_C)^2$$

Now, the question becomes what is the intensity value at the interpixel locations P, P', Q, and Q'. Since we are interested in finding corners in digital images we must consider the discrete nature of such pictures. Figure 7.4 shows the construction given in Figure 7.3 as it applies to a discretized 3x3 window. This shows that the intensity values at A, A', B, and B' are known. The intensity at an arbitrary point along the lines is not available, so must be approximated using the intensity values at A, B, A', and B'. For a linear approximation, the intensity values at P, P', Q, and Q' are:

$$I_P = (1-x)I_A + xI_B \quad (2)$$

$$I_{P'} = (1-x)I_{A'} + xI_{B'}$$

$$I_Q = (1-x)I_A + xI_B$$

$$I_{Q'} = (1-x)I_{A'} + xI_{B'}$$

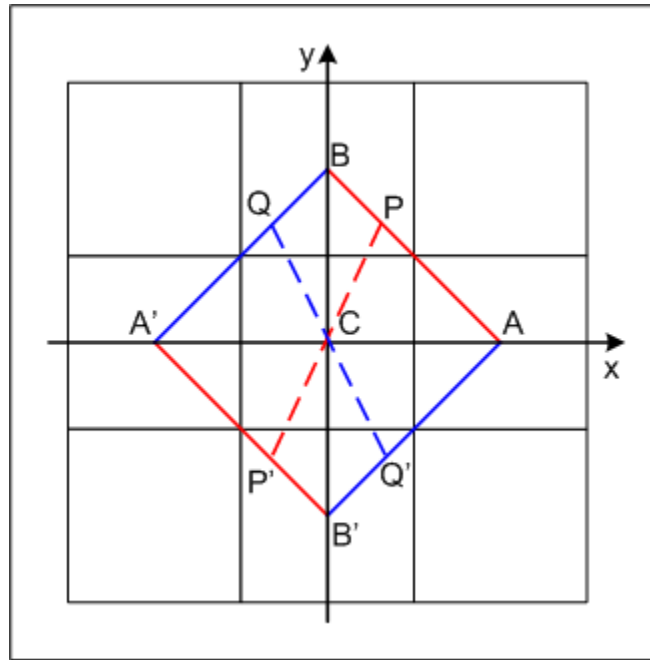


Figure 7.4: Interpixel approximation for a 3x3 window using 4-neighbours

Substituting (2) into (1) allows the corneriness measure to be expressed as:

$$C_{\text{cornerness}}(x, y) = \min_{\theta \in \{0\}} (r_1(x), r_2(x))$$

where:

$$r_1(x) = A_1 x^2 + 2B_1 x + C$$

$$r_2(x) = A_2 x^2 + 2B_2 x + C$$

$$B_1 = (I_B - I_A)(I_A - I_C) + (I_B - I_A)(I_A - I_C)$$

$$B_2 = (I_B - I_A)(I_A - I_C) + (I_B - I_A)(I_A - I_C)$$

$$C = r_A$$

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

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

The cornerness measure is the minimum intensity variation in any direction where the intensity variation is approximated as explained above. When is this minimum reached?

The minimum is given by:

$$C_{\text{cornerness}}(x, y) = \begin{cases} C - \frac{B^2}{A} & \text{if } B < 0 \text{ and } (A+B) > 0 \\ C_{\text{cornerness}}(x, y) & \text{otherwise} \end{cases}$$

where,

$$B_1 = (I_B - I_A)(I_A - I_C) + (I_B - I_A)(I_A - I_C)$$

$$B_2 = (I_B - I_A)(I_A - I_C) + (I_B - I_A)(I_A - I_C)$$

$$C = r_A$$

$$B = \min(B_1, B_2)$$

$$A = r_B - r_A - 2B$$

Some examples illustrating the use of this operator are given [here](#) and a proof of the above formula is given [here](#).

This analysis is for a circular window of diameter 3 that considers only the 4-connected neighbours of the center pixel. Use of a bigger window can improve sensitivity to noise, but at the cost of poorer localization and increased computation. It is also possible to perform a circular interpixel approximation as described in [5]. However, Trajkovic claims better results are obtained with a linear approximation and performs all evaluations of the operator when using a window of diameter 3. However, it will be shown that using only 4-neighbours (as opposed to all 8-neighbours) results in a rather poor corner detector.

Multigrid Algorithm

Corners can be considered as coming from two separate categories - geometric and texture. Geometric corners belong to the boundaries of objects in the image (e.g. the corners of a television set) whereas texture corners come from the textures of objects in the image (e.g. grass or patterned clothing). In a typical image there will be more texture corners than geometric corners as objects are generally of a reasonable size, so there are few objects in the image and each object will only contain a few corners.

The multigrid algorithm attempts to reduce the number of texture corners found while retaining the majority of geometric corners detected in the image. Trajkovic believes geometric corners are more stable than texture corners, so texture corners should be eliminated. It is not apparent that this is true and [5] gives no quantitative evidence or qualitative argument supporting this claim. However, does one expect to find corners in grass? No. Geometric corners match our intuitive notion of what a corner is far more closely than texture corners do and removing texture corners can be justified in this sense.

How can texture corners be eliminated? When taking a smaller resolution version of an image, nearby image pixels are averaged. For example, say an 8x8 input image is to be scaled to 4x4. A simple way to do this is to take the average of every group of four adjacent pixels as illustrated in Figure 7.5. Since texture corners are small regions of intensity variation they tend to disappear on the lower resolution image due to this averaging.

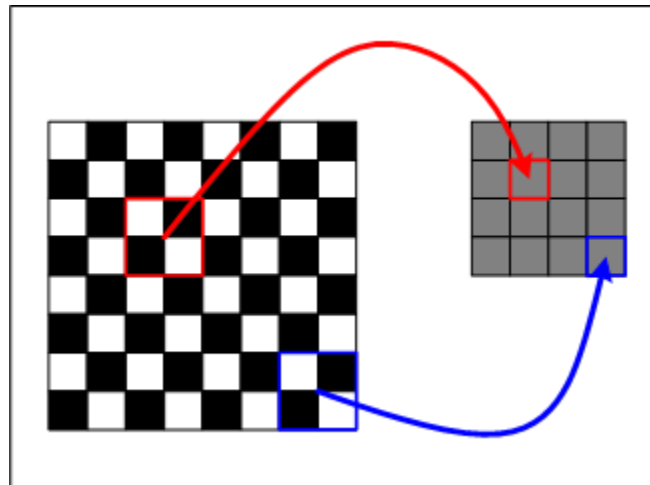


Figure 7.5: Constructing a lower resolution image by averaging pixels (texture is eliminated due to averaging)

The Trajkovic corner detection algorithm makes use of a low resolution version of the input image in order to reduce the number of texture corners reported and to increase the speed of the algorithm. This is done by applying the simple Trajkovic operator to the lower resolution image and marking potential corner points. These potential corner points are then examined in the original input image using the interpixel Trajkovic operator to verify the presence of a corner. Figure 7.6 shows the results of running the simple Trajkovic operator on different resolution versions of the same image. Notice that far fewer texture corners are identified at lower resolutions, but that some geometric

corners are also lost. The optimal scale for the low resolution version of the input image depends on the application.



Figure 7.6: Corner points detected at different image resolutions

Algorithm

The Trajkovic corner detector is stated formally below:

Input: grayscale image, scale for low resolution version of image, threshold T_1 , threshold T_2

Output: map indicating position of each detected corner

1. For each pixel (x, y) in the low resolution image calculate the simple cornerness measure:

$$C_{\text{simple}}(x, y) = \min\{r_A, r_B\}$$

$$\text{where: } r_A = (I_A - I_C)^2 + (I_E - I_C)^2, r_B = (I_B - I_C)^2 + (I_F - I_C)^2$$

Flag any pixel (x, y) with a cornerness measure $C_{\text{simple}}(x, y) \geq T_1$ as a potential corner.

2. Initialize a cornerness map M , with dimensions of the input image, to be all zeros.
3. For each potential corner pixel (x, y) found in step 1:
 - 3a. Compute the location of the pixel in the original input image, (x', y') .

3b. Compute the simple cornerness measure $C_{SIMPLE}(x', y')$.

3c. If $C_{SIMPLE}(x', y') \leq T_2$ the pixel is not a corner so leave its cornerness measure in M as zero and do not perform steps 3d and 3e.

3d. Compute the interpixel approximation cornerness measure:

$$C_{INTERPIXEL}(x, y) = \begin{cases} C - \frac{B^2}{A} & \text{if } B < 0 \text{ and } (A+B) > 0 \\ C_{SIMPLE}(x, y) & \text{otherwise} \end{cases}$$

where,

$$B_1 = (I_B - I_A)(I_A - I_C) + (I_B - I_X)(I_X - I_C)$$

$$B_2 = (I_B - I_X)(I_X - I_C) + (I_B - I_A)(I_A - I_C)$$

$$C = r_A$$

$$B = \min(B_1, B_2)$$

$$A = r_B - r_A - 2E$$

3e. If $C_{INTERPIXEL}(x', y') \leq T_2$ leave the cornerness measure in M as zero, else set $M(x', y')$ to $C_{INTERPIXEL}(x', y')$.

4. Perform non-maximal suppression on M to find local maxima.

All non-zero points remaining in the cornerness map M are corners.

Results on Test Images

The Trajkovic operator was applied to the three test images discussed in the [Introduction](#).

Figure 7.7 shows the Trajkovic operator applied to the artificial test image. The Trajkovic operator detects all of the corners and has good localization, but detects lots of false corners along diagonal edges. It has a lower detection rate than the [Moravec](#) operator due to the false corners detected along the diagonals.



Figure 7.7: Trajkovic operator applied to Artificial Test Image

Figure 7.8 shows the Trajkovic operator applied to the blocks test image. Parameters were chosen in order to detect most of the corners while trying to minimize the number of false corners detected. Comparing these results to the [Moravec](#) operator again indicates the Trajkovic operator has a worse detection rate due to the large number of false corners detected along diagonal edges.

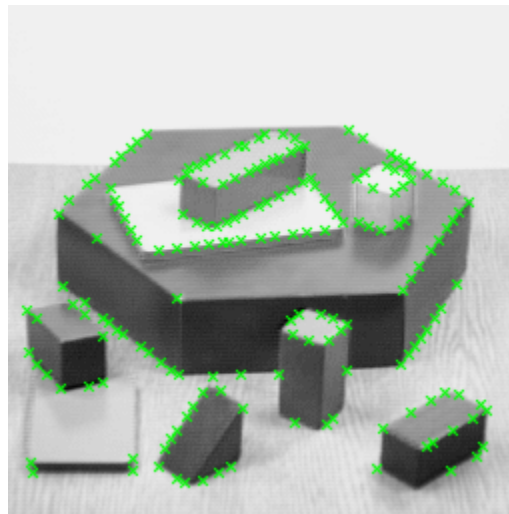


Figure 7.8: Trajkovic operator applied to Blocks Test Image

Figure 7.9 shows the Trajkovic operator applied to the house test image. Parameters were manually set high enough to avoid detecting corners due to the texture of the house. Again, the most notable aspect of this test image is the large number of false corners detected along diagonal edges.

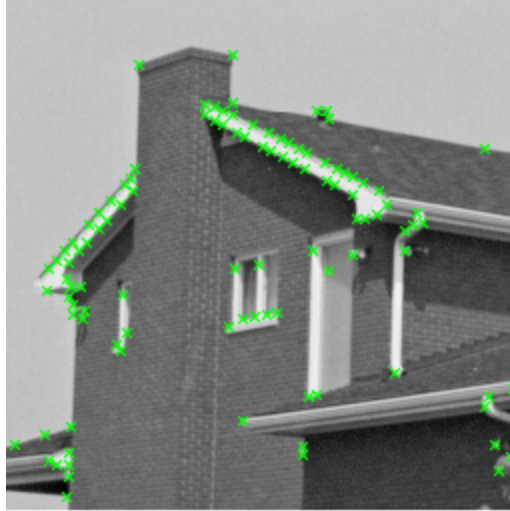


Figure 7.9: Trajkovic operator applied to House Test Image

These results clearly indicate the Trajkovic operator, when using only 4-neighbours, responds too readily to diagonal edges. This issue is explored below in the Limitations section.

To see the effect of different parameter values use the [Corner Detection Applet](#).

Limitations

The computational efficiency of the Trajkovic corner detection algorithm makes it desirable. However, it is not without its limitations as it is not rotationally invariant, is sensitive to noise along edges and at corners, and responds too readily to diagonal edges.

Anisotropic Response of Operator

Like both the [Moravec](#) and [Plessey](#) operators, the Trajkovic operator has an [anisotropic](#) response. The Trajkovic operator *estimates* the change of intensity variation in all direction by using an interpixel approximation, which is an improvement over the discrete set of directions considered by the Moravec operator. However, the Trajkovic operator is anisotropic as both the simple and interpixel cornerness measures use only pixels along the two primary axes. Figure 7.10 shows a 3 pixel by 3 pixel checkerboard pattern and the same pattern rotated by 45°. Both cornerness measures use only the intensity of the pixels labeled A, A', B, and B' to measure the intensity variation at pixel C. Therefore, the response will be different for these two images and the operator is not rotationally invariant.

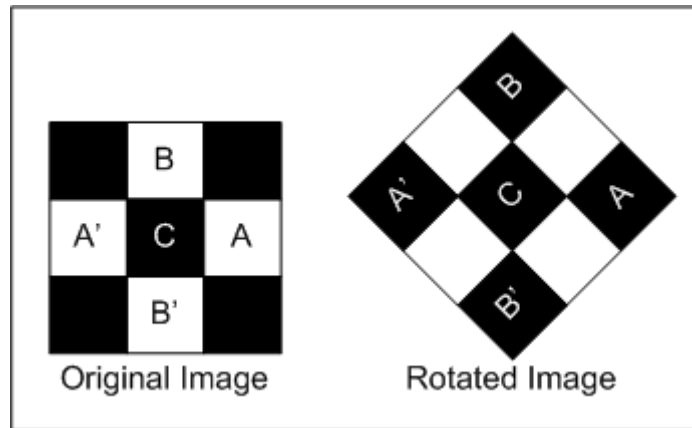


Figure 7.10: Anisotropic response of Trajkovic operator

For an actual image, rotating the original image in Figure 7.10 by 45° obviously does not rotate the image as shown. Rotations can not be done precisely in digital images due to the discretization of the image. In practice, a rotation of a digital image often results in averaging of nearby pixel values to approximate the appearance of the desired rotation. Many different methods can be used to approximate a rotation, but any reasonable method will have to change the values of A, A', B, and B' in order to approximate the appearance of a 45° rotation and consequently will change the cornerness measure.

Sensitivity to Noise and False Corners along Diagonal Edges

Both the simple and interpixel cornerness measures use only the intensity of the pixels labeled A, A', B, and B' to measure the intensity variation at pixel C. Figure 7.11 shows that the Trajkovic operator is sensitive to noise both along edges and at corners. A single corrupt pixel along an edge, marked in red in Figure 7.11, can change the cornerness measure along that edge to have the same measure as a corner. Alternatively, a corrupt pixel at a true corner, can change the cornerness measure at the corner to appear as if it is an edge.

Figure 7.11 also illustrates why the Trajkovic operator responds too readily to diagonal edges. The 45° edge has the exact same cornerness measure as the true corner. It is easy to imagine that for other angled edges, especially in the presence of anti-aliasing, that there will be places along the edge that have a large cornerness measure. Notice that if the 8-neighbours of pixel C were considered, then the true corner would be more easily distinguishable from a diagonal edge. This is explored in the [Trajkovic operator \(8-Neighbours\)](#) section of this website.

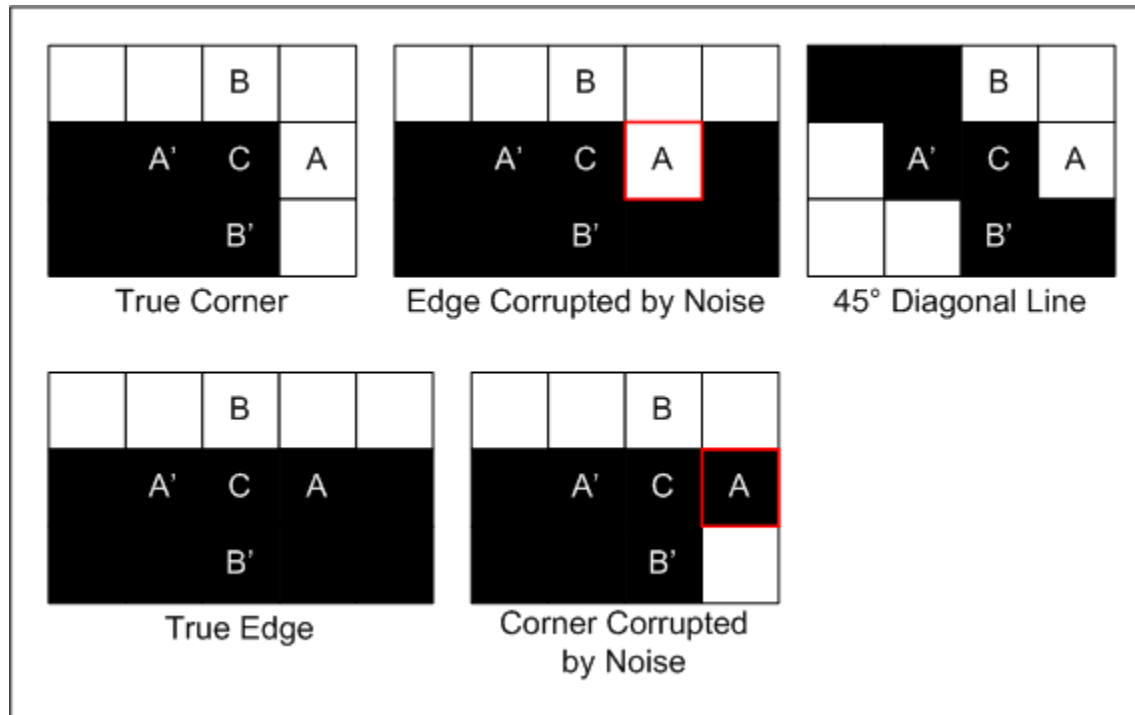


Figure 7.11: Trajkovic operators sensitivity to noise

Conclusions

The minimal computational demands of the Trajkovic operator make it well-suited for real-time applications. Computational demand is kept small through the use of a multigrid algorithm and by first evaluating potential corners using the simple Trajkovic operator. Note, that the interpixel Trajkovic operator requires 5 multiplications and a divide so is relatively expensive to compute. The multigrid algorithm and the notion of having a simple operator to quickly reject corners are important features of this corner detection algorithm that could be used by other corner detection algorithms. For example, some applications may find it desirable to use the Trajkovic algorithm, but replace the interpixel Trajkovic operator with the Plessey operator.

When only 4-neighbours are considered, the Trajkovic operator has a worse detection rate than the [Moravec](#) operator and is sensitive to noise. This low detection rate is due to the operator responding too readily to diagonal edges. Most applications will find this level of performance unacceptable, so an [8-neighbourhood version](#) of the Trajkovic operator must be used.

Trajkovic Operator (8-Neighbours)

Introduction

The [4-neighbourhood version](#) of the Trajkovic operator was shown to be sensitive to noise and to respond too readily to diagonal edges. These limitations can be partially overcome by considering all 8-neighbours around a given pixel. However, it will be seen that even when all 8-neighbours are considered, the operator still responds too readily to certain diagonal edges.

The notation Trajkovic8 and Trajkovic4 are used to distinguish between these two versions of the Trajkovic operator.

Discussion

The Trajkovic8 algorithm is identical to the Trajkovic4 algorithm except for how it calculates the simple and interpixel cornerness measures. Instead of considering only 4-neighbours, the Trajkovic8 algorithm considers all 8-neighbours as shown in Figure 8.1. The intensity values at the points A,B,C,D,E,A',B',D', and E' are known and the intensity between these pixels are interpolated as for the Trajkovic4 operator.

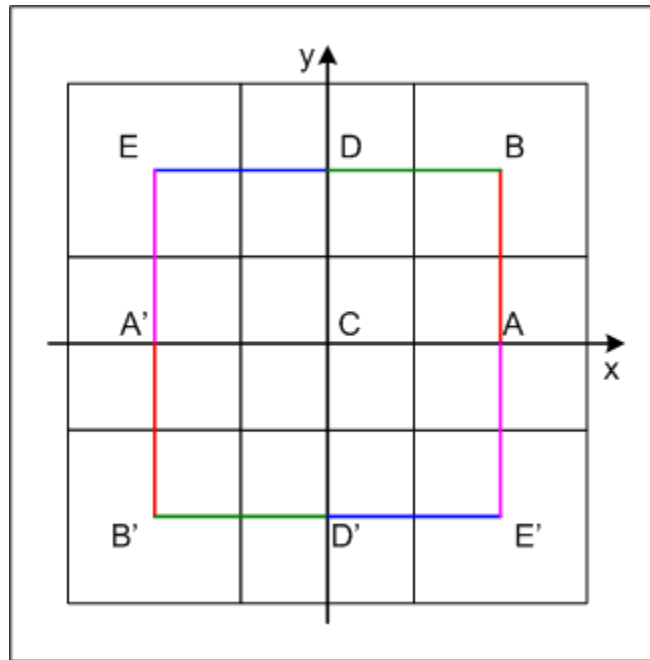


Figure 8.1: Interpixel approximation for a 3x3 window using 8-neighbours

For the Trajkovic8 operator, the simple cornerness measure is defined as:

$$C_{\text{cornerness}}(x, y) = \min(r_A, r_B, r_D, r_E)$$

where:

$$r_A = (I_A - I_C)^2 + (I_E - I_C)^2$$

$$r_B = (I_B - I_C)^2 + (I_D - I_C)^2$$

$$r_D = (I_D - I_C)^2 + (I_E - I_C)^2$$

$$r_E = (I_E - I_C)^2 + (I_B - I_C)^2$$

The interpixel cornerness measure is given as:

$$C_{\text{interpixel}}(x, y) = \min_{i \in \{1, 2, 3, 4\}} (r_i(x))$$

where:

$$r_1(x) = A_1 x^2 + 2B_1 x + r_A$$

$$r_2(x) = A_2 x^2 + 2B_2 x + r_B$$

$$r_3(x) = A_3 x^2 + 2B_3 x + r_D$$

$$r_4(x) = A_4 x^2 + 2B_4 x + r_E$$

$$B_1 = (I_B - I_D)(I_A - I_C) + (I_E - I_B)(I_E - I_C)$$

$$B_2 = (I_D - I_B)(I_B - I_C) + (I_D - I_E)(I_E - I_C)$$

$$B_3 = (I_B - I_D)(I_B - I_C) + (I_E - I_D)(I_E - I_C)$$

$$B_4 = (I_E - I_B)(I_B - I_C) + (I_A - I_E)(I_E - I_C)$$

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

$$A_2 = r_D - r_B - 2B_2$$

$$A_3 = r_E - r_D - 2B_3$$

$$A_4 = r_A - r_E - 2B_4$$

This formula can be derived in a manner similar to that used for the Trajkovic4 operator and is left as an exercise for the reader. The minimum of this formula is given by:

$$C_{\text{interpixel}}(x, y) = \begin{cases} C_{\text{cornerness}}(x, y) & \text{if for all } i = 1, 2, 3, 4 \text{ either } B_i \geq 0 \text{ or } A_i + B_i \leq 0 \\ \min \left(x_i - \frac{B_i^2}{A_i} \right) & \text{for all } i = 1, 2, 3, 4 \text{ that satisfy } B_i < 0 \text{ and } A_i + B_i > 0 \end{cases}$$

$$\text{where: } r_1 = r_B, r_2 = r_D, r_3 = r_E, r_4 = r_A$$

A proof of the above formula can be obtained in a similar manner to the proof given for the Trajkovic4 operator.

Algorithm

The Trajkovic8 corner detector is stated formally below:

Input: grayscale image, scale for low resolution version of image, threshold T_1 , threshold T_2

Output: map indicating position of each detected corner

1. For each pixel (x, y) in the low resolution image calculate the simple cornerness measure:

$$C_{\text{simple}}(x, y) = \min(r_A, r_B, r_C, r_D)$$

where:

$$r_A = (I_A - I_C)^2 + (I_B - I_C)^2$$

$$r_B = (I_B - I_C)^2 + (I_D - I_C)^2$$

$$r_D = (I_D - I_C)^2 + (I_A - I_C)^2$$

$$r_C = (I_A - I_C)^2 + (I_D - I_C)^2$$

Flag any pixel (x, y) with a cornerness measure $C_{\text{simple}}(x, y) \geq T_1$ as a potential corner.

2. Initialize a cornerness map M with dimensions of the input image to be all zeros.
3. For each potential corner pixel (x, y) found in step 1:
 - 3a. Compute the location of the pixel in the original input image, (x', y') .
 - 3b. Compute the simple cornerness measure $C_{\text{simple}}(x', y')$.
 - 3c. If $C_{\text{simple}}(x', y') \leq T_2$ the pixel is not a corner so leave its cornerness measure in M as zero and do not perform steps 3d and 3e.
 - 3d. Compute the interpixel approximation cornerness measure:

$$C_{\text{INTERPIXEL}}(x,y) = \begin{cases} C_{\text{INTERPIXEL}}(x,y) & \text{if for all } i = 1,2,3,4 \text{ either } E_i \geq 0 \text{ or } A_i + B_i \leq 0 \\ \min \begin{pmatrix} r_1 & B_1^1 \\ A_1 & \end{pmatrix} & \text{for all } i = 1,2,3,4 \text{ that satisfy } E_i < 0 \text{ and } A_i + B_i > 0 \end{cases}$$

where:

$$r_1 = r_A, r_2 = r_B, r_3 = r_D, r_4 = r_S$$

$$B_1 = (I_B - I_A)(I_A - I_C) + (I_S - I_X)(I_X - I_C)$$

$$E_2 = (I_D - I_B)(I_B - I_C) + (I_S - I_S)(I_S - I_C)$$

$$B_3 = (I_S - I_D)(I_D - I_C) + (I_S - I_S)(I_S - I_C)$$

$$E_4 = (I_X - I_S)(I_S - I_C) + (I_A - I_S)(I_S - I_C)$$

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

$$A_2 = r_D - r_B - 2B_2$$

$$A_3 = r_S - r_D - 2B_3$$

$$A_4 = r_A - r_S - 2B_4$$

3e. If $C_{\text{INTERPIXEL}}(x', y') \leq T_2$ leave the corneriness measure in M as zero, else set $M(x', y')$ to $C_{\text{INTERPIXEL}}(x', y')$.

4. Perform non-maximal suppression on M to find local maxima.

All non-zero points remaining in the corneriness map M are corners.

Results on Test Images

The Trajkovic8 operator was applied to the three test images discussed in the [Introduction](#).

Figure 8.2 shows the Trajkovic4 and Trajkovic8 operators applied to the artificial test image. The Trajkovic8 operator finds fewer false corners than the Trajkovic4 operator as it does not detect false corners along the 45° degree edges. However, the artificial test image clearly shows that the Trajkovic8 operator is still responding too readily to certain diagonal lines.

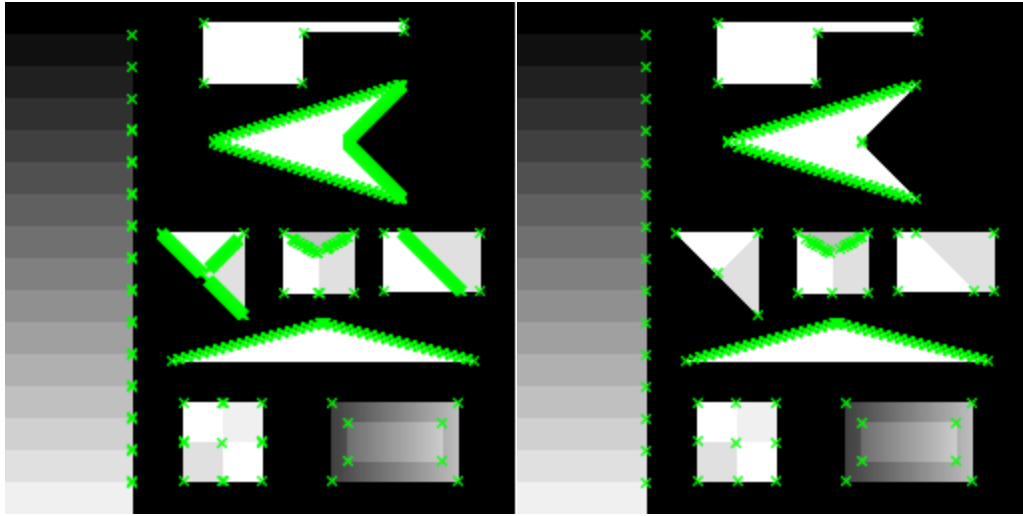


Figure 8.2: Trajkovic4 (left) and Trajkovic8 (right) operators applied to Artificial Test Image

Figure 8.3 shows the Trajkovic4 and Trajkovic8 operators applied to the blocks test image. Parameters were chosen in order to detect most of the corners while trying to minimize the number of false corners detected. The results of the Trajkovic8 operator are much improved over the Trajkovic4 operator, but there are still a number of false corners detected along the diagonal edges.

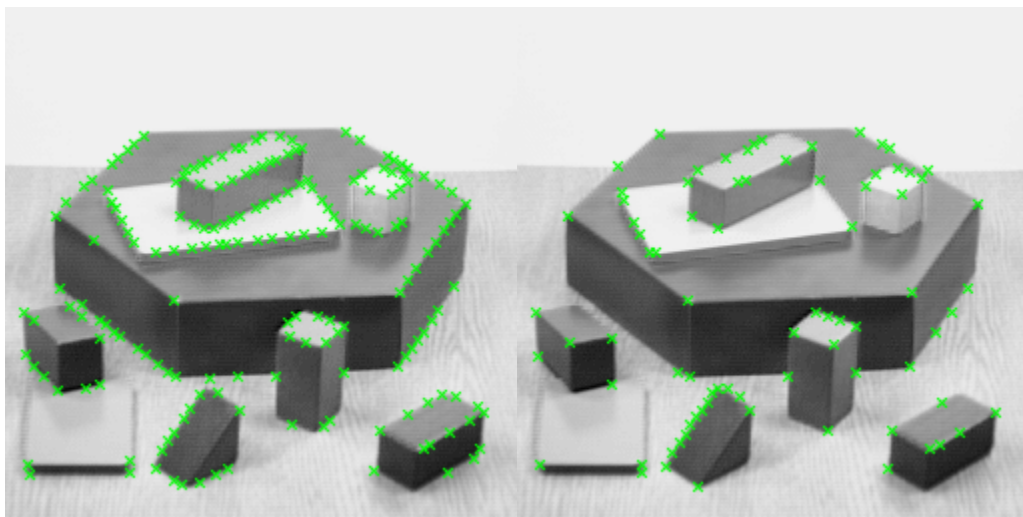


Figure 8.3: Trajkovic4 (left) and Trajkovic8 (right) operators applied to Blocks Test Image

Figure 8.4 shows the Trajkovic4 and Trajkovic8 operators applied to the house test image. Parameters were manually set high enough to avoid detecting corners due to the texture of the house. The results of the Trajkovic8 operator are vastly improved over the Trajkovic4 operator and are comparable to those achieved by the [Plessey](#) operator.



Figure 8.4: Trajkovic4 (left) and Trajkovic8 (right) operators applied to House Test Image

The Trajkovic8 operator has good localization and is computationally efficient. However, these images clearly show that it can have a poor detection rate on some images due to finding false corners along diagonal edges. The Limitations section below discusses why these false corners are detected.

To see the effect of different parameter values use the [Corner Detection Applet](#).

Limitations

The Trajkovic8 operator has improved performance over the Trajkovic4 operator, but still has a number of limitations.

Anisotropic Response of Operator

Both the Trajkovic4 and Trajkovic8 corner detection algorithms are [anisotropic](#). The anisotropic response of the Trajkovic8 algorithm is illustrated in Figure 8.5.

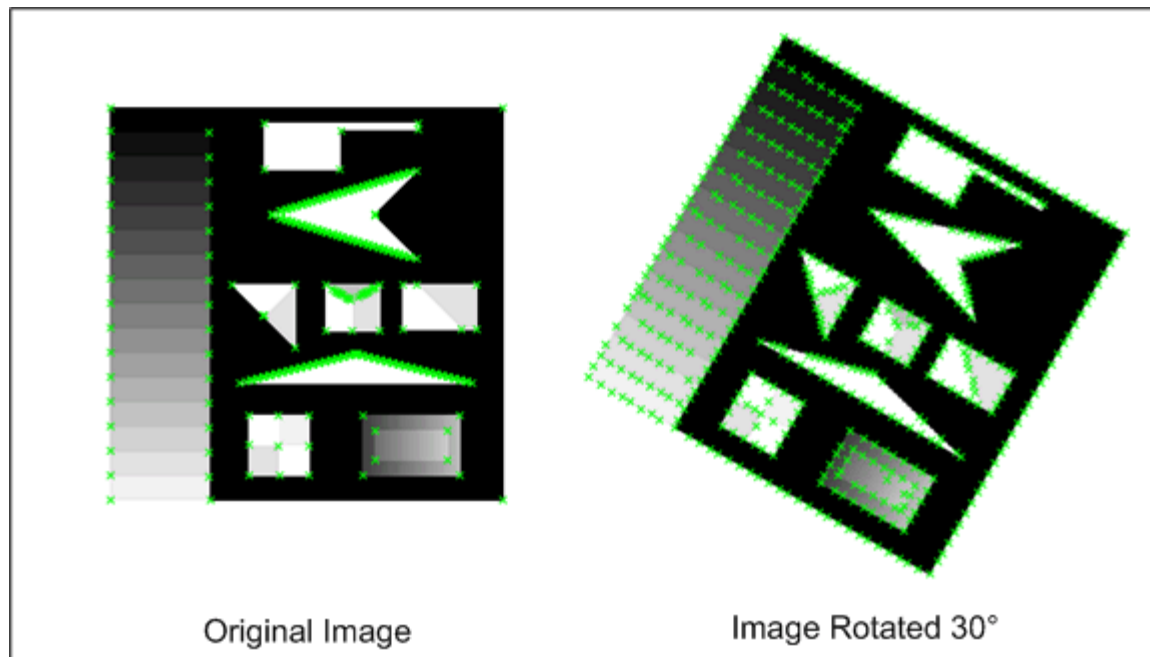


Figure 8.5: Anisotropic response of Trajkovic8 algorithm

False Corners along Diagonal Lines

The [Results on Test Images](#) section illustrated that the Trajkovic8 operator can respond too readily to diagonal edges. Figure 8.6 shows the structure of one of the diagonal edges in the artificial test image. Due to the discrete nature of digital images, the diagonal lines appear as a 'staircase'. The question is what effect this digitization of an ideal line has on the Trajkovic operator.

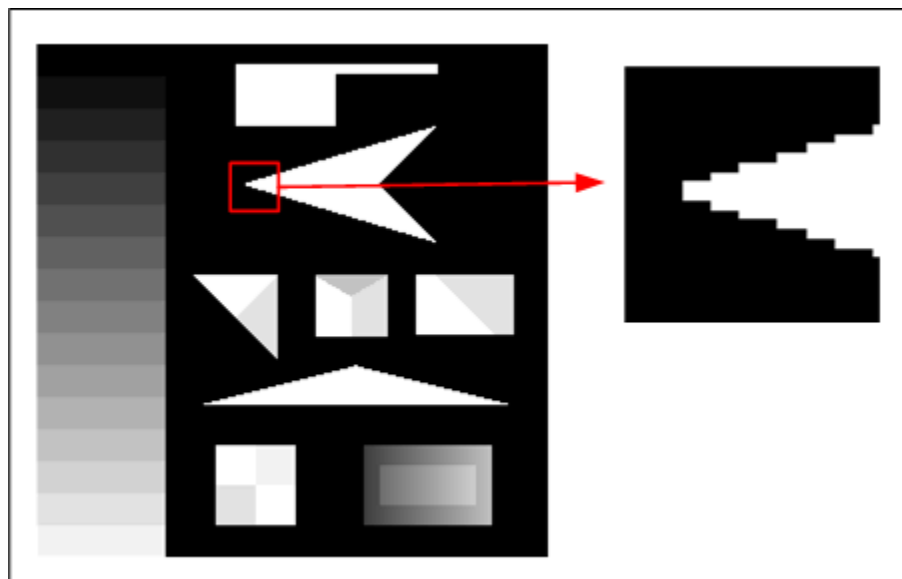


Figure 8.6: Close-up of diagonal lines in artificial test image

Similar to the analysis performed for the Trajkovic4 operator, it can be shown that along certain diagonal edges the response of the Trajkovic8 operator is the same as for a true corner. Figure 8.7 shows a true corner and a diagonal edge with a structure similar to that shown in Figure 8.6. Since corresponding neighbours in the true corner and diagonal edge have the same intensity value, the cornerness measure will be the same.

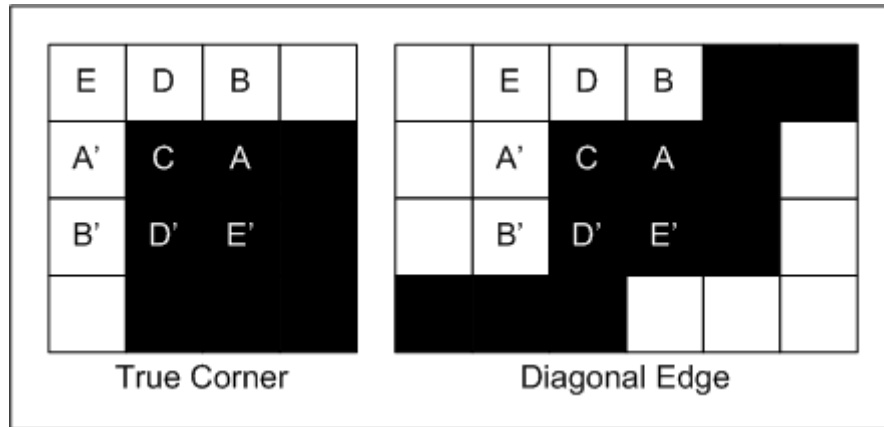


Figure 8.7: Identical response of Trajkovic8 operator to a true corner and diagonal edge

Conclusions

Real-time applications may consider using the Trajkovic corner detection algorithm, but most applications will still favour the [Plessey](#) operator thanks to its strong repeatability rate. Both the Trajkovic4 and Trajkovic8 operator suffers from having an anisotropic response and respond too readily to diagonal edges. Both of these factors will adversely affect the repeatability rate of the Trajkovic operators and this criteria is critical for many applications. However, the house test image suggests that for some sets of images the Trajkovic operator may have a detection rate comparable to the Plessey operator. For applications requiring a computationally efficient operator and working on a restricted set of image where it can be shown that the performance of the Trajkovic operator is acceptable, this operator may be useful.

Evaluation Summary

Table 9.1 gives the results of each of the corner detectors on the three test images.

- *Artificial Test Image:* Results on the artificial test image show that the Plessey operator has the best detection rate since both the Moravec and Trajkovic operators report many false corners along diagonal edges. However, this image also illustrates the poor localization of the Plessey operator on most junction types. Both the Moravec and Trajkovic operators show good localization on all junction types.
- *Block Test Image:* The Plessey and Trajkovic operators perform much better than the Moravec operator on this test image. Both the Plessey and Trajkovic

operators detect the majority of true corners, but only when a relatively large number of false corners are present. The Plessey operator has the better detection rate, but only due to the large number of false corners detected by the Trajkovic operator along the edge of the triangle block. Again the poor localization of the Plessey operator can clearly be seen at certain corners.

- *House Test Image*: Again, the Moravec operator finds a large number of false corners along diagonal edges. Although the Trajkovic operator does detect false corners on some diagonal edges, it is clearly more robust than the Moravec operator and has a reasonable performance on this image. For this image, the performance of the Plessey and Trajkovic operators are comparable.

The following conclusions can be made based on these results:

- The Moravec operator finds many false corners on diagonal edges. Given this, there are few applications that will decide to use this operator.
- The Trajkovic operator finds many false corners on *some* diagonal edges. This operator performs poorly on the artificial test image, but well on the house test image. For applications requiring a computationally efficient operator and working on a restricted set of images where it can be shown that the detection rate of the Trajkovic operator is acceptable, this operator may be useful.
- The Plessey operator suffers from poor localization and is computationally expensive. However, it has the best detection rate of the three operators and has been shown to have a good repeatability rate. Localization is not critical for many applications. For these reasons the Plessey operator is widely used in practice.

	Moravec Corner Detector	Plessey Corner Detector	Trajkovic Corner Detector (8-Neighbours)
Artificial Test Image			

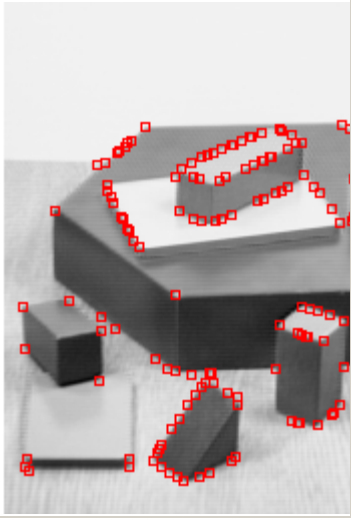
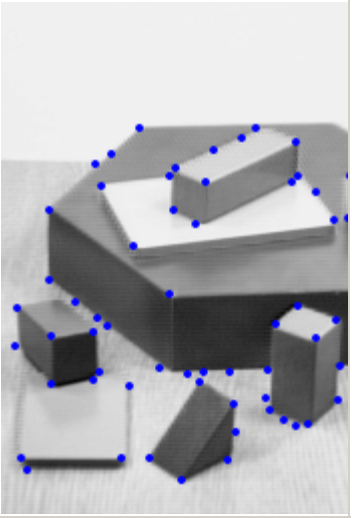
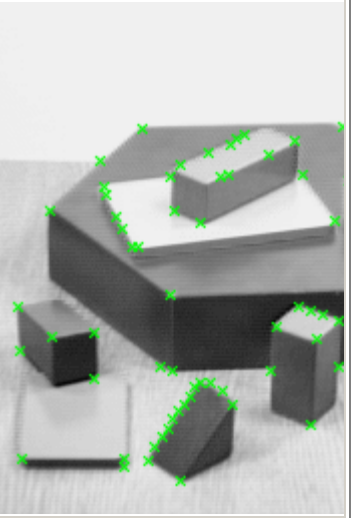
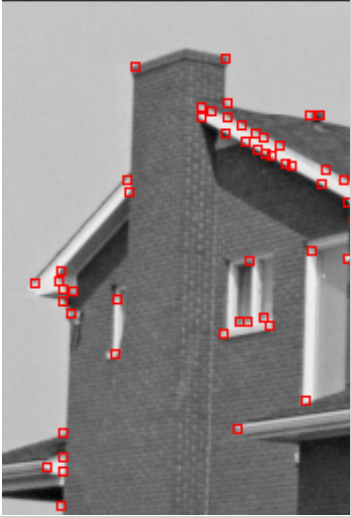
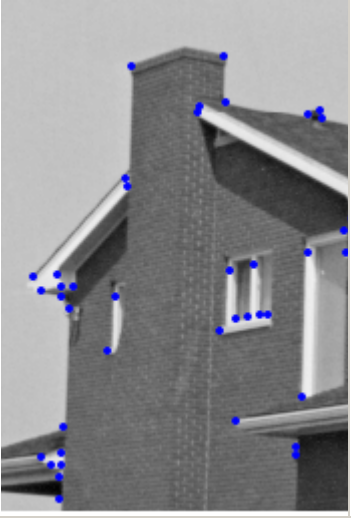
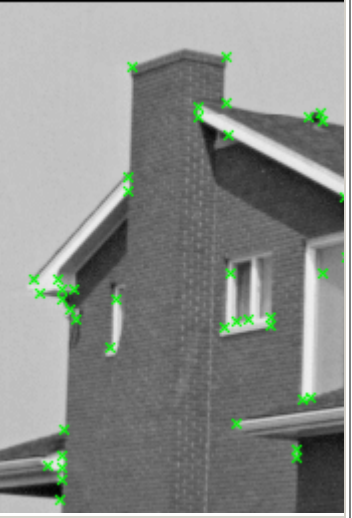
Block s Test Image			
Hous e Test Image			

Table 9.1: Comparison of Corner Detectors on the Three Test Images

Useful Links

- [Detection of Interest Points](#) - online demonstration of six different interest point algorithms (including the Plessey, SUSAN, and Zheng corner detectors).
- [Detection of High Curvature Points in Planar Curves](#) - discussion of corner detection using contour tracing with an online comparison of five popular algorithms.
- [Matching with Invariant Features](#) - excellent presentation discussing the Plessey corner detector and addressing issues regarding invariance.

- [Basic Algorithms for Digital Image Analysis: Corner Detection](#) - informative slides on the Plessey and KLT corner detectors.
- [Computer Vision Slides](#) - information on the Moravec and Plessey corner detectors along with line fitting, medial axis, and Hough transform.
- [Geometric Feature Extraction Methods](#) - rich resource from [CVonline](#) with information about corner and interest point detectors.
- [Morphological Corner Detector](#) - use of morphological operators for the detection of corners.
- [Differential Geometry](#) - a brief introduction to differential geometry (the mathematical basis of many corner detectors).

References

- [1] H. P. Moravec. [Towards Automatic Visual Obstacle Avoidance](#). *Proc. 5th International Joint Conference on Artificial Intelligence*, pp. 584, 1977.
- [2] H. P. Moravec. [Visual Mapping by a Robot Rover](#). *International Joint Conference on Artificial Intelligence*, pp. 598-600, 1979.
- [3] C. Harris and M. Stephens. [A Combined Corner and Edge Detector](#). *Proc. Alvey Vision Conf.*, Univ. Manchester, pp. 147-151, 1988.
- [4] S.M. Smith and M. Brady. [SUSAN - A New Approach to Low Level Image Processing](#). *International Journal of Computer Vision*, Vol. 23(1), pp. 45-78, 1997.
- [5] M. Trajkovic and M. Hedley. Fast Corner Detection. *Image and Vision Computing*, Vol. 16(2), pp. 75-87, 1998.
- [6] C. Schmid, R. Mohr, and C. Bauckhage. Comparing and Evaluating Interesting Points. *International Conference on Computer Vision*, pp. 230-235, 1998.
- [7] F. Mohanna and F. Mokhtarian. [Performance Evaluation of Corner Detection Algorithms under Similarity and Affine Transforms](#). *BMVC 2001*.
- [8] Z.Zheng, H.Wang and EKTeoh. Analysis of Gray Level Corner Detection. *Pattern Recognition Letters*, Vol. 20, pp. 149-162, 1999.
- [9] S.M. Smith. Extracting Information from Images. First year D.Phil. report, Robotics Research Group, Department of Engineering Science, Oxford University, June 1990.