Corner detection

From Wikipedia, the free encyclopedia

|  |
| --- |
| [**Feature detection**](http://en.wikipedia.org/wiki/Feature_detection_(computer_vision)) |
| [Corner.png](http://en.wikipedia.org/wiki/File:Corner.png)  Output of a typical corner detection algorithm |
| [**Edge detection**](http://en.wikipedia.org/wiki/Edge_detection) |
| * [Canny](http://en.wikipedia.org/wiki/Canny_edge_detector)      * [Canny–Deriche](http://en.wikipedia.org/wiki/Canny_edge_detector#Conclusion)      * [Differential](http://en.wikipedia.org/wiki/Edge_detection#Differential_edge_detection)      * [Sobel](http://en.wikipedia.org/wiki/Sobel_operator) * [Prewitt](http://en.wikipedia.org/wiki/Prewitt)      * [Roberts cross](http://en.wikipedia.org/wiki/Roberts_cross) |
| **Corner detection** |
| * [Harris operator](http://en.wikipedia.org/wiki/Corner_detection#The_Harris_.26_Stephens_.2F_Plessey_.2F_Shi.E2.80.93Tomasi_corner_detection_algorithm)      * [Shi and Tomasi](http://en.wikipedia.org/wiki/Corner_detection#The_Harris_.26_Stephens_.2F_Plessey_.2F_Shi.E2.80.93Tomasi_corner_detection_algorithm) * [Level curve curvature](http://en.wikipedia.org/wiki/Corner_detection#The_level_curve_curvature_approach)      * [SUSAN](http://en.wikipedia.org/wiki/Corner_detection#The_SUSAN_corner_detector)      * [FAST](http://en.wikipedia.org/wiki/Corner_detection#AST_based_feature_detectors) |
| [**Blob detection**](http://en.wikipedia.org/wiki/Blob_detection) |
| * [Laplacian of Gaussian (LoG)](http://en.wikipedia.org/wiki/Blob_detection#The_Laplacian_of_Gaussian) * [Difference of Gaussians (DoG)](http://en.wikipedia.org/wiki/Difference_of_Gaussians) * [Determinant of Hessian (DoH)](http://en.wikipedia.org/wiki/Blob_detection#The_determinant_of_the_Hessian) * [Maximally stable extremal regions](http://en.wikipedia.org/wiki/Maximally_stable_extremal_regions)      * [PCBR](http://en.wikipedia.org/wiki/Principal_Curvature-Based_Region_Detector) |
| [**Ridge detection**](http://en.wikipedia.org/wiki/Ridge_detection) |
| [**Hough transform**](http://en.wikipedia.org/wiki/Hough_transform) |
| [**Structure tensor**](http://en.wikipedia.org/wiki/Structure_tensor) |
| **Affine invariant feature detection** |
| * [Affine shape adaptation](http://en.wikipedia.org/wiki/Affine_shape_adaptation)      * [Harris affine](http://en.wikipedia.org/wiki/Harris_affine_region_detector) * [Hessian affine](http://en.wikipedia.org/wiki/Hessian_Affine_region_detector) |
| **Feature description** |
| * [SIFT](http://en.wikipedia.org/wiki/Scale-invariant_feature_transform)      * [SURF](http://en.wikipedia.org/wiki/SURF)      * [GLOH](http://en.wikipedia.org/wiki/GLOH)      * [HOG](http://en.wikipedia.org/wiki/Histogram_of_oriented_gradients) |
| [**Scale space**](http://en.wikipedia.org/wiki/Scale_space) |
| * [Scale-space axioms](http://en.wikipedia.org/wiki/Scale-space_axioms)      * [Implementation details](http://en.wikipedia.org/wiki/Scale_space_implementation) * [Pyramids](http://en.wikipedia.org/wiki/Pyramid_(image_processing)) |
| * [v](http://en.wikipedia.org/wiki/Template:Feature_detection_(computer_vision)_navbox)      * [t](http://en.wikipedia.org/wiki/Template_talk:Feature_detection_(computer_vision)_navbox)      * [e](http://en.wikipedia.org/w/index.php?title=Template:Feature_detection_(computer_vision)_navbox&action=edit) |

**Corner detection** is an approach used within [computer vision](http://en.wikipedia.org/wiki/Computer_vision) systems to extract certain kinds of [features](http://en.wikipedia.org/wiki/Feature_detection) and infer the contents of an image. Corner detection is frequently used in [motion detection](http://en.wikipedia.org/wiki/Motion_detection), [image registration](http://en.wikipedia.org/wiki/Image_registration), [video tracking](http://en.wikipedia.org/wiki/Video_tracking), [image mosaicing](http://en.wikipedia.org/wiki/Photographic_mosaic), [panorama stitching](http://en.wikipedia.org/wiki/Panorama_stitching), [3D modelling](http://en.wikipedia.org/wiki/3D_modelling) and [object recognition](http://en.wikipedia.org/wiki/Object_recognition). Corner detection overlaps with the topic of [**interest point detection**](http://en.wikipedia.org/wiki/Interest_point_detection).

**Contents**

  [[hide](http://en.wikipedia.org/wiki/Corner_detection)]

* [1 Formalization](http://en.wikipedia.org/wiki/Corner_detection#Formalization)
* [2 The Moravec corner detection algorithm](http://en.wikipedia.org/wiki/Corner_detection#The_Moravec_corner_detection_algorithm)
* [3 The Harris & Stephens / Plessey / Shi–Tomasi corner detection algorithm](http://en.wikipedia.org/wiki/Corner_detection#The_Harris_.26_Stephens_.2F_Plessey_.2F_Shi.E2.80.93Tomasi_corner_detection_algorithm)
* [4 The Förstner corner detector](http://en.wikipedia.org/wiki/Corner_detection#The_F.C3.B6rstner_corner_detector)
* [5 The multi-scale Harris operator](http://en.wikipedia.org/wiki/Corner_detection#The_multi-scale_Harris_operator)
* [6 The level curve curvature approach](http://en.wikipedia.org/wiki/Corner_detection#The_level_curve_curvature_approach)
* [7 LoG, DoG, and DoH feature detection](http://en.wikipedia.org/wiki/Corner_detection#LoG.2C_DoG.2C_and_DoH_feature_detection)
* [8 Affine-adapted interest point operators](http://en.wikipedia.org/wiki/Corner_detection#Affine-adapted_interest_point_operators)
* [9 The Wang and Brady corner detection algorithm](http://en.wikipedia.org/wiki/Corner_detection#The_Wang_and_Brady_corner_detection_algorithm)
* [10 The SUSAN corner detector](http://en.wikipedia.org/wiki/Corner_detection#The_SUSAN_corner_detector)
* [11 The Trajkovic and Hedley corner detector](http://en.wikipedia.org/wiki/Corner_detection#The_Trajkovic_and_Hedley_corner_detector)
* [12 AST-based feature detectors](http://en.wikipedia.org/wiki/Corner_detection#AST-based_feature_detectors)
* [13 Automatic synthesis of detectors](http://en.wikipedia.org/wiki/Corner_detection#Automatic_synthesis_of_detectors)
* [14 Bibliography](http://en.wikipedia.org/wiki/Corner_detection#Bibliography)
* [15 Reference implementations](http://en.wikipedia.org/wiki/Corner_detection#Reference_implementations)
* [16 See also](http://en.wikipedia.org/wiki/Corner_detection#See_also)
* [17 External links](http://en.wikipedia.org/wiki/Corner_detection#External_links)

Formalization[[edit](http://en.wikipedia.org/w/index.php?title=Corner_detection&action=edit&section=1" \o "Edit section: Formalization)]

A corner can be defined as the intersection of two edges. A corner can also be defined as a point for which there are two dominant and different edge directions in a local neighborhood of the point.

An interest point is a point in an image which has a well-defined position and can be robustly detected. This means that an interest point can be a corner but it can also be, for example, an isolated point of local intensity maximum or minimum, line endings, or a point on a curve where the curvature is locally maximal.

In practice, most so-called corner detection methods detect interest points in general, rather than corners in particular.[[*citation needed*](http://en.wikipedia.org/wiki/Wikipedia:Citation_needed)] As a consequence, if only corners are to be detected it is necessary to do a local analysis of detected interest points to determine which of these are real corners. Examples of edge detection that can be used with post-processing to detect corners are the [Kirsch operator](http://en.wikipedia.org/wiki/Kirsch_operator) and the Frei-Chen masking set.[[1]](http://en.wikipedia.org/wiki/Corner_detection#cite_note-1)

"Corner", "interest point" and "feature" are used interchangeably in literature, confusing the issue. Specifically, there are several [blob detectors](http://en.wikipedia.org/wiki/Blob_detection) that can be referred to as "interest point operators", but which are sometimes erroneously referred to as "corner detectors". Moreover, there exists a notion of [ridge detection](http://en.wikipedia.org/wiki/Ridge_detection) to capture the presence of elongated objects.

Corner detectors are not usually very robust and often require expert supervision or large redundancies introduced to prevent the effect of individual errors from dominating the recognition task.

One determination of the quality of a corner detector is its ability to detect the same corner in multiple similar images, under conditions of different lighting, translation, rotation and other transforms.

A simple approach to corner detection in images is using [correlation](http://en.wikipedia.org/wiki/Correlation), but this gets very computationally expensive and suboptimal. An alternative approach used frequently is based on a method proposed by Harris and Stephens (below), which in turn is an improvement of a method by Moravec.

The Moravec corner detection algorithm[[edit](http://en.wikipedia.org/w/index.php?title=Corner_detection&action=edit&section=2" \o "Edit section: The Moravec corner detection algorithm)]

This is one of the earliest corner detection algorithms and defines a corner to be a point with low self-similarity.[[2]](http://en.wikipedia.org/wiki/Corner_detection#cite_note-moravec-2) The algorithm tests each pixel in the image to see if a corner is present, by considering how similar a patch centered on the pixel is to nearby, largely overlapping patches. The similarity is measured by taking the sum of squared differences (SSD) between the two patches. A lower number indicates more similarity.

If the pixel is in a region of uniform intensity, then the nearby patches will look similar. If the pixel is on an edge, then nearby patches in a direction perpendicular to the edge will look quite different, but nearby patches in a direction parallel to the edge will result only in a small change. If the pixel is on a feature with variation in all directions, then none of the nearby patches will look similar.

The corner strength is defined as the smallest SSD between the patch and its neighbors (horizontal, vertical and on the two diagonals). If this number is locally maximal, then a feature of interest is present.

As pointed out by Moravec, one of the main problems with this operator is that it is not [isotropic](http://en.wikipedia.org/wiki/Isotropic): if an edge is present that is not in the direction of the neighbours, then the smallest SSD will be large and the edge will be incorrectly chosen as an interest point.

The Harris & Stephens / Plessey / Shi–Tomasi corner detection algorithm[[edit](http://en.wikipedia.org/w/index.php?title=Corner_detection&action=edit&section=3" \o "Edit section: The Harris & Stephens / Plessey / Shi–Tomasi corner detection algorithm)]

Harris and Stephens[[3]](http://en.wikipedia.org/wiki/Corner_detection" \l "cite_note-harris-3) improved upon Moravec's corner detector by considering the differential of the corner score with respect to direction directly, instead of using shifted patches. (This corner score is often referred to as [autocorrelation](http://en.wikipedia.org/wiki/Autocorrelation), since the term is used in the paper in which this detector is described. However, the mathematics in the paper clearly indicate that the sum of squared differences is used.)

Without loss of generality, we will assume a grayscale 2-dimensional image is used. Let this image be given by I. Consider taking an image patch over the area (u, v) and shifting it by (x, y). The weighted *sum of squared differences* (SSD) between these two patches, denoted S, is given by:


S(x,y) = \sum_u \sum_v w(u,v) \, \left( I(u+x,v+y) - I(u,v)\right)^2


I(u+x,v+y) can be approximated by a [Taylor expansion](http://en.wikipedia.org/wiki/Taylor_series). Let  I_x and  I_y be the [partial derivatives](http://en.wikipedia.org/wiki/Partial_derivatives) of  I, such that


I(u+x,v+y) \approx I(u,v) + I_x(u,v)x+I_y(u,v)y


This produces the approximation


S(x,y) \approx \sum_u \sum_v w(u,v) \, \left( I_x(u,v)x + I_y(u,v)y \right)^2,


which can be written in matrix form:


S(x,y) \approx \begin{pmatrix} x & y \end{pmatrix} A \begin{pmatrix} x \\ y \end{pmatrix},


where *A* is the [structure tensor](http://en.wikipedia.org/wiki/Structure_tensor),


A = \sum_u \sum_v w(u,v) 
\begin{bmatrix}
I_x^2 & I_x I_y \\
I_x I_y & I_y^2 
\end{bmatrix}
=
\begin{bmatrix}
\langle I_x^2 \rangle & \langle I_x I_y \rangle\\
\langle I_x I_y \rangle & \langle I_y^2 \rangle
\end{bmatrix}


This matrix is a Harris matrix, and angle brackets denote averaging (i.e. summation over (u,v)). If a circular window (or circularly weighted window, such as a [Gaussian](http://en.wikipedia.org/wiki/Gaussian_function)) is used, then the response will be isotropic.

A corner (or in general an interest point) is characterized by a large variation of  S  in all directions of the vector  \begin{pmatrix} x & y \end{pmatrix} . By analyzing the eigenvalues of  A , this characterization can be expressed in the following way:  A  should have two "large" eigenvalues for an interest point. Based on the magnitudes of the eigenvalues, the following inferences can be made based on this argument:

1. If \lambda_1 \approx 0 and \lambda_2 \approx 0 then this pixel (x,y) has no features of interest.
2. If \lambda_1 \approx 0 and \lambda_2 has some large positive value, then an edge is found.
3. If  \lambda_1 and  \lambda_2 have large positive values, then a corner is found.

Harris and Stephens note that exact computation of the eigenvalues is computationally expensive, since it requires the computation of a [square root](http://en.wikipedia.org/wiki/Square_root), and instead suggest the following function M_c, where \kappa is a tunable sensitivity parameter:


M_c = \lambda_1 \lambda_2 - \kappa \, (\lambda_1 + \lambda_2)^2
= \operatorname{det}(A) - \kappa \, \operatorname{trace}^2(A)


Therefore, the algorithm does not have to actually compute the [eigenvalue decomposition](http://en.wikipedia.org/wiki/Eigenvalue_decomposition) of the matrix A and instead it is sufficient to evaluate the [determinant](http://en.wikipedia.org/wiki/Determinant) and [trace](http://en.wikipedia.org/wiki/Trace_(linear_algebra)) of A to find corners, or rather interest points in general.

The Shi–Tomasi[[4]](http://en.wikipedia.org/wiki/Corner_detection" \l "cite_note-shitomasi-4) corner detector directly computes min(\lambda_1, \lambda_2) because under certain assumptions, the corners are more stable for tracking. Note that this method is also sometimes referred to as the Kanade-Tomasi corner detector.

The value of \kappa has to be determined empirically, and in the literature values in the range 0.04–0.15 have been reported as feasible.

One can avoid setting the parameter \kappa by using Noble's[[5]](http://en.wikipedia.org/wiki/Corner_detection" \l "cite_note-noble-5) corner measure M_c' which amounts to the [harmonic mean](http://en.wikipedia.org/wiki/Harmonic_mean) of the eigenvalues:

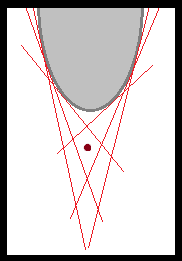

M_c' = 2 \frac{\operatorname{det}(A)}{\operatorname{trace}(A) + \epsilon},


\epsilon being a small positive constant.

The [covariance matrix](http://en.wikipedia.org/wiki/Covariance_matrix) for the corner position is  A^{-1} , i.e.


\frac{1}{\langle I_x^2 \rangle \langle I_y^2 \rangle - \langle I_x I_y \rangle^2}
\begin{bmatrix}
\langle I_y^2 \rangle & -\langle I_x I_y \rangle\\
-\langle I_x I_y \rangle & \langle I_x^2 \rangle
\end{bmatrix}.


The Förstner corner detector[[edit](http://en.wikipedia.org/w/index.php?title=Corner_detection&action=edit&section=4" \o "Edit section: The Förstner corner detector)]

[](http://en.wikipedia.org/wiki/File:Corner_detection_using_Foerstner_Algorithm.png)

[http://bits.wikimedia.org/static-1.23wmf2/skins/common/images/magnify-clip.png](http://en.wikipedia.org/wiki/File:Corner_detection_using_Foerstner_Algorithm.png)

Corner detection using the Förstner Algorithm

In some cases, one may wish to compute the location of a corner with subpixel accuracy. To achieve an approximate solution, the Förstner[[6]](http://en.wikipedia.org/wiki/Corner_detection" \l "cite_note-6) algorithm solves for the point closest to all the tangent lines of the corner in a given window and is a least-square solution. The algorithm relies on the fact that for an ideal corner, tangent lines cross at a single point.

The equation of a tangent line T_{\mathbf{x'}}(\mathbf{x}) at pixel \mathbf{x'} is given by:


T_\mathbf{x'}(\mathbf x)=\nabla I(\mathbf{x'})^{\top}(\mathbf{x}-\mathbf{x'})=0


where \nabla I(\mathbf{x'})=[I_{\mathbf{x}},I_{\mathbf{y}}]^{\top} is the gradient vector of the image I at \mathbf{x'}.

The point \mathbf{x}_{0} closest to all the tangent lines in the window N is:

\mathbf{x}_{0}=\underset{\mathbf{x}\in \mathbb{R}^{2\times 2}}{\operatorname{argmin}}\int_{\mathbf{x'}\in N}T_{\mathbf{x'}}(\mathbf{x})^{2}d\mathbf{x'}

The distance from \mathbf{x}_{0} to the tangent lines T_{\mathbf{x'}} is weighted by the gradient magnitude, thus giving more importance to tangents passing through pixels with strong gradients.

Solving for \mathbf{x}_{0}:


\begin{align}
\mathbf{x}_{0}&=\underset{\mathbf{x}\in \mathbb{R}^{2\times 2}}{\operatorname{argmin}} \int_{\mathbf{x'}\in N}(\nabla I(\mathbf{x'})^{\top}(\mathbf{x}-\mathbf{x'}))^{2}d\mathbf{x'}\\
&=\underset{\mathbf{x}\in \mathbb{R}^{2\times 2}}{\operatorname{argmin}}\int_{\mathbf{x'}\in N}(\mathbf{x}-\mathbf{x'})^{\top}\nabla I(\mathbf{x'})\nabla I(\mathbf{x'})^{\top}(\mathbf{x}-\mathbf{x'})d\mathbf{x'}\\
&=\underset{\mathbf{x}\in \mathbb{R}^{2\times 2}}{\operatorname{argmin}}\, (\mathbf{x}^{\top}A\mathbf{x}-2\mathbf{x}^{\top}\mathbf{b}+c)
\end{align}


A\in\mathbb{R}^{2\times 2},\textbf{b}\in\mathbb{R}^{2\times 1},c\in\mathbb{R} are defined as:


\begin{align}
A&=\int \nabla I(\mathbf{x'})\nabla I(\mathbf{x'})^{\top}d\mathbf{x'}\\
\mathbf{b}&=\int \nabla I(\mathbf{x'})\nabla I(\mathbf{x'})^{\top}\mathbf{x'}d\mathbf{x'}\\
c&=\int \mathbf{x'}^{\top}\nabla I(\mathbf{x'})\nabla I(\mathbf{x'})^{\top}\mathbf{x'}d\mathbf{x'}\\
\end{align}


Minimizing this equation can be done by differentiating with respect to x and setting it equal to 0:


2A\mathbf{x}-2\mathbf{b}=0 \Rightarrow A\mathbf{x}=\mathbf{b}


Note that A\in\mathbb{R}^{2\times 2} is the [structure tensor](http://en.wikipedia.org/wiki/Structure_tensor). For the equation to have a solution, A must be invertible, which implies that A must be full rank (rank 2). Thus, the solution

x_{0}=A^{-1}\mathbf{b}

only exists where an actual corner exists in the window N.

A methodology for performing *automatic scale selection* for this corner localization method has been presented by Lindeberg[[7]](http://en.wikipedia.org/wiki/Corner_detection" \l "cite_note-lindeberg94icip-7)[[8]](http://en.wikipedia.org/wiki/Corner_detection#cite_note-lindeberg98-8) by minimizing the normalized residual

\tilde{d}_{min} = \frac{c - b^T A^{-1} b}{\mbox{trace} A}

over scales. Thereby, the method has the ability to automatically adapt the scale levels for computing the image gradients to the noise level in the image data, by choosing coarser scale levels for noisy image data and finer scale levels for near ideal corner-like structures.

Notes:

* c can be viewed as a residual in the least-square solution computation: if c=0, then there was no error.
* this algorithm can be modified to compute centers of circular features by changing tangent lines to normal lines.

The multi-scale Harris operator[[edit](http://en.wikipedia.org/w/index.php?title=Corner_detection&action=edit&section=5" \o "Edit section: The multi-scale Harris operator)]

The computation of the second moment matrix (sometimes also referred to as the [structure tensor](http://en.wikipedia.org/wiki/Structure_tensor)) A in the Harris operator, requires the computation of image derivatives I_x, I_y in the image domain as well as the summation of non-linear combinations of these derivatives over local neighbourhoods. Since the computation of derivatives usually involves a stage of scale-space smoothing, an operational definition of the Harris operator requires two scale parameters: (i) a *local scale* for smoothing prior to the computation of image derivatives, and (ii) an *integration scale* for accumulating the non-linear operations on derivative operators into an integrated image descriptor.

With I denoting the original image intensity, let L denote the [scale space representation](http://en.wikipedia.org/wiki/Scale_space_representation) of I obtained by convolution with a Gaussian kernel

g(x, y, t) = \frac {1}{2{\pi} t}e^{-(x^2+y^2)/2t}

with local scale parameter t:

L(x, y, t)\ = g(x, y, t) * I(x, y)

and let L_x = \partial_x L and L_y = \partial_y L denote the partial derivatives of L. Moreover, introduce a Gaussian window function g(x, y, s) with integration scale parameter s. Then, the [*multi-scale second-moment matrix*](http://en.wikipedia.org/wiki/Structure_tensor#The_multi-scale_structure_tensor) [[9]](http://en.wikipedia.org/wiki/Corner_detection#cite_note-lindeberg94book-9)[[10]](http://en.wikipedia.org/wiki/Corner_detection#cite_note-lindeberg08enc-10) can be defined as


\mu(x, y; t, s) =
\int_{\xi = -\infty}^{\infty} \int_{\eta = -\infty}^{\infty}
\begin{bmatrix}
 L_x^2(x-\xi, y-\eta; t)                               & L_x(x-\xi, y-\eta; t) \, L_y(x-\xi, y-\eta; t) \\
L_x(x-\xi, y-\eta; t) \, L_y(x-\xi, y-\eta; t) & L_y^2(x-\xi, y-\eta; t) 
\end{bmatrix}
g(\xi, \eta; s) \, d\xi \, d\eta.


Then, we can compute eigenvalues of \mu in a similar way as the eigenvalues of A and define the *multi-scale Harris corner measure* as

M_c(x, y; t, s) = \operatorname{det}(\mu(x, y; t, s)) - \kappa \, \operatorname{trace}^2(\mu(x, y; t, s)).

Concerning the choice of the local scale parameter t and the integration scale parameter s, these scale parameters are usually coupled by a relative integration scale parameter \gamma such that s = \gamma^2 t, where \gamma is usually chosen in the interval [1, 2]. Thus, we can compute the multi-scale Harris corner measure M_c(x, y; t, \gamma^2 t) at any scale t in scale-space to obtain a multi-scale corner detector, which responds to corner structures of varying sizes in the image domain.

In practice, this multi-scale corner detector is often complemented by a *scale selection step*, where the scale-normalized Laplacian operator[[8]](http://en.wikipedia.org/wiki/Corner_detection" \l "cite_note-lindeberg98-8)[[9]](http://en.wikipedia.org/wiki/Corner_detection#cite_note-lindeberg94book-9)

\nabla^2_{norm} L(x, y; t)\ = t \nabla^2 L(x, y, t) = t (L_{xx}(x, y, t) + L_{yy}(x, y, t))

is computed at every scale in scale-space and *scale adapted corner points with automatic scale selection* (the "Harris-Laplace operator") are computed from the points that are simultaneously:[[11]](http://en.wikipedia.org/wiki/Corner_detection#cite_note-schmid-11)

* spatial maxima of the multi-scale corner measure M_c(x, y; t, \gamma^2 t)

(\hat{x}, \hat{y}; t) = \operatorname{argmaxlocal}_{(x, y)} M_c(x, y; t, \gamma^2 t)

* local maxima or minima over scales of the scale-normalized Laplacian operator[[8]](http://en.wikipedia.org/wiki/Corner_detection#cite_note-lindeberg98-8) \nabla^2_{norm}(x, y, t):

\hat{t} = \operatorname{argmaxminlocal}_{t} \nabla^2_{norm}L(\hat{x}, \hat{y}; t)

The level curve curvature approach[[edit](http://en.wikipedia.org/w/index.php?title=Corner_detection&action=edit&section=6" \o "Edit section: The level curve curvature approach)]

An earlier approach to corner detection is to detect points where the [curvature](http://en.wikipedia.org/wiki/Curvature) of level curves and the gradient magnitude are *simultaneously* high.[[12]](http://en.wikipedia.org/wiki/Corner_detection#cite_note-kitchen82-12)[[13]](http://en.wikipedia.org/wiki/Corner_detection#cite_note-richards88-13) A differential way to detect such points is by computing *the rescaled level curve curvature* (the product of the level curve curvature and the gradient magnitude raised to the power of three)

\tilde{\kappa}(x, y;t) = L_x^2 L_{yy} + L_y^2 L_{xx} - 2 L_x L_y L_{xy}

and to detect positive maxima and negative minima of this differential expression at some scale t in the [scale space representation](http://en.wikipedia.org/wiki/Scale_space_representation) L of the original image.[[7]](http://en.wikipedia.org/wiki/Corner_detection#cite_note-lindeberg94icip-7)[[8]](http://en.wikipedia.org/wiki/Corner_detection#cite_note-lindeberg98-8) A main problem when computing the rescaled level curve curvature entity at a single scale however, is that it may be sensitive to noise and to the choice of the scale level. A better method is to compute the *\gamma-normalized rescaled level curve curvature*

\tilde{\kappa_{norm}}(x, y;t) = t^{2 \gamma} (L_x^2 L_{yy} + L_y^2 L_{xx} - 2 L_x L_y L_{xy})

with \gamma = 7/8 and to detect *signed scale-space extrema* of this expression, that are points and scales that are positive maxima and negative minima with respect to both space and scale

(\hat{x}, \hat{y}; \hat{t}) = \operatorname{argminmaxlocal}_{(x, y; t)} \tilde{\kappa}_{norm}(x, y; t)

in combination with a complementary localization step to handle the increase in localization error at coarser scales.[[7]](http://en.wikipedia.org/wiki/Corner_detection#cite_note-lindeberg94icip-7)[[8]](http://en.wikipedia.org/wiki/Corner_detection#cite_note-lindeberg98-8)[[9]](http://en.wikipedia.org/wiki/Corner_detection#cite_note-lindeberg94book-9) In this way, larger scale values will be associated with rounded corners of large spatial extent while smaller scale values will be associated with sharp corners with small spatial extent. This approach is the first corner detector with automatic scale selection (prior to the "Harris-Laplace operator" above) and has been used for tracking corners under large scale variations in the image domain[[14]](http://en.wikipedia.org/wiki/Corner_detection#cite_note-brelin98feattrack-14) and for matching corner responses to edges to compute structural image features for geon-based object recognition.[[15]](http://en.wikipedia.org/wiki/Corner_detection#cite_note-lindebergli97-15)

LoG, DoG, and DoH feature detection[[edit](http://en.wikipedia.org/w/index.php?title=Corner_detection&action=edit&section=7" \o "Edit section: LoG, DoG, and DoH feature detection)]

LoG[[8]](http://en.wikipedia.org/wiki/Corner_detection" \l "cite_note-lindeberg98-8)[[11]](http://en.wikipedia.org/wiki/Corner_detection#cite_note-schmid-11) is an acronym standing for *Laplacian of Gaussian*, DoG[[16]](http://en.wikipedia.org/wiki/Corner_detection" \l "cite_note-sift-16) is an acronym standing for *difference of Gaussians* (DoG is an approximation of LoG), and DoH is an acronym standing for *determinant of the Hessian.*[[8]](http://en.wikipedia.org/wiki/Corner_detection#cite_note-lindeberg98-8)

These detectors are more completely described in [blob detection](http://en.wikipedia.org/wiki/Blob_detection), however the LoG and DoG blobs do not necessarily make highly selective features, since these operators may also respond to edges. To improve the corner detection ability of the DoG detector, the feature detector used in the [SIFT](http://en.wikipedia.org/wiki/Scale-invariant_feature_transform)[[16]](http://en.wikipedia.org/wiki/Corner_detection#cite_note-sift-16) system uses an additional post-processing stage, where the [eigenvalues](http://en.wikipedia.org/wiki/Eigenvalue) of the [Hessian](http://en.wikipedia.org/wiki/Hessian_matrix) of the image at the detection scale are examined in a similar way as in the Harris operator. If the ratio of the eigenvalues is too high, then the local image is regarded as too edge-like, so the feature is rejected. The DoH operator on the other hand only responds when there are significant grey-level variations in two directions.[[8]](http://en.wikipedia.org/wiki/Corner_detection#cite_note-lindeberg98-8)[[10]](http://en.wikipedia.org/wiki/Corner_detection#cite_note-lindeberg08enc-10)

Affine-adapted interest point operators[[edit](http://en.wikipedia.org/w/index.php?title=Corner_detection&action=edit&section=8" \o "Edit section: Affine-adapted interest point operators)]

The interest points obtained from the multi-scale Harris operator with automatic scale selection are invariant to translations, rotations and uniform rescalings in the spatial domain. The images that constitute the input to a computer vision system are, however, also subject to perspective distortions. To obtain an interest point operator that is more robust to perspective transformations, a natural approach is to devise a feature detector that is *invariant to affine transformations*. In practice, affine invariant interest points can be obtained by applying [affine shape adaptation](http://en.wikipedia.org/wiki/Affine_shape_adaptation) where the shape of the smoothing kernel is iteratively warped to match the local image structure around the interest point or equivalently a local image patch is iteratively warped while the shape of the smoothing kernel remains rotationally symmetric.[[9]](http://en.wikipedia.org/wiki/Corner_detection#cite_note-lindeberg94book-9)[[10]](http://en.wikipedia.org/wiki/Corner_detection#cite_note-lindeberg08enc-10)[[11]](http://en.wikipedia.org/wiki/Corner_detection#cite_note-schmid-11) Hence, besides the commonly used multi-scale Harris operator, affine shape adaptation can be applied to other corner detectors as listed in this article as well as to [differential blob detectors](http://en.wikipedia.org/wiki/Blob_detection) such as the Laplacian/difference of Gaussian operator, the determinant of the Hessian[[10]](http://en.wikipedia.org/wiki/Corner_detection#cite_note-lindeberg08enc-10) and the Hessian–Laplace operator.

The Wang and Brady corner detection algorithm[[edit](http://en.wikipedia.org/w/index.php?title=Corner_detection&action=edit&section=9" \o "Edit section: The Wang and Brady corner detection algorithm)]

The Wang and Brady[[17]](http://en.wikipedia.org/wiki/Corner_detection" \l "cite_note-wangbrady-17) detector considers the image to be a surface, and looks for places where there is large [curvature](http://en.wikipedia.org/wiki/Curvature) along an image edge. In other words, the algorithm looks for places where the edge changes direction rapidly. The corner score, C, is given by:


C = \nabla^2I - c|\nabla I|^2,


where c determines how edge-phobic the detector is. The authors also note that smoothing (Gaussian is suggested) is required to reduce noise. In this case, the first term of C becomes the Laplacian (single-scale) [blob detector](http://en.wikipedia.org/wiki/Blob_detection).

Smoothing also causes displacement of corners, so the authors derive an expression for the displacement of a 90 degree corner, and apply this as a correction factor to the detected corners.

The SUSAN corner detector[[edit](http://en.wikipedia.org/w/index.php?title=Corner_detection&action=edit&section=10" \o "Edit section: The SUSAN corner detector)]

SUSAN[[18]](http://en.wikipedia.org/wiki/Corner_detection" \l "cite_note-susan-18) is an acronym standing for *smallest univalue segment assimilating nucleus.* This method is the subject of a 1994 UK patent which is no longer in force.[[19]](http://en.wikipedia.org/wiki/Corner_detection#cite_note-19)

For feature detection, SUSAN places a circular mask over the pixel to be tested (the nucleus). The region of the mask is M, and a pixel in this mask is represented by \vec{m} \in M. The nucleus is at \vec{m}_0. Every pixel is compared to the nucleus using the comparison function:


c(\vec{m}) = e^{-\left(\frac{I(\vec{m}) - I(\vec{m}_0)}{t}\right)^6}


where t determines the radius, and the power of the exponent has been determined empirically. This function has the appearance of a smoothed [top-hat or rectangular function](http://en.wikipedia.org/wiki/Rectangular_function). The area of the SUSAN is given by:


n(M) = \sum_{\vec{m}\in M} c(\vec{m})


If c is the rectangular function, then n is the number of pixels in the mask which are within t of the nucleus. The response of the SUSAN operator is given by:


R(M) =     \begin{cases}
               g - n(M) & \mbox{if}\ n(M) < g\\
               0        & \mbox{otherwise,}
           \end{cases}


where g is named the `geometric threshold'. In other words the SUSAN operator only has a positive score if the area is small enough. The smallest SUSAN locally can be found using non-maximal suppression, and this is the complete SUSAN operator.

The value t determines how similar points have to be to the nucleus before they are considered to be part of the univalue segment. The value of g determines the minimum size of the univalue segment. If g is large enough, then this becomes an [edge detector](http://en.wikipedia.org/wiki/Edge_detection).

For corner detection, two further steps are used. Firstly, the [centroid](http://en.wikipedia.org/wiki/Centroid) of the SUSAN is found. A proper corner will have the centroid far from the nucleus. The second step insists that all points on the line from the nucleus through the centroid out to the edge of the mask are in the SUSAN.

The Trajkovic and Hedley corner detector[[edit](http://en.wikipedia.org/w/index.php?title=Corner_detection&action=edit&section=11" \o "Edit section: The Trajkovic and Hedley corner detector)]

In a manner similar to SUSAN, this detector[[20]](http://en.wikipedia.org/wiki/Corner_detection" \l "cite_note-hedley-20) directly tests whether a patch under a pixel is self-similar by examining nearby pixels. \vec{c} is the pixel to be considered, and \vec{p} \in P is point on a circle P centered around \vec{c}. The point \vec{p'} is the point opposite to \vec{p} along the diameter.

The response function is defined as:


  r(\vec{c}) = \min_{\vec{p} \in P}\quad (I(\vec{p}) - I(\vec{c}))^2 + (I(\vec{p'}) - I(\vec{c})) ^2


This will be large when there is no direction in which the centre pixel is similar to two nearby pixels along a diameter. P is a discretised circle (a [Bresenham circle](http://en.wikipedia.org/wiki/Midpoint_circle_algorithm" \o "Midpoint circle algorithm)), so [interpolation](http://en.wikipedia.org/wiki/Interpolation) is used for intermediate diameters to give a more isotropic response. Since any computation gives an upper bound on the \min, the horizontal and vertical directions are checked first to see if it is worth proceeding with the complete computation of c.

AST-based feature detectors[[edit](http://en.wikipedia.org/w/index.php?title=Corner_detection&action=edit&section=12" \o "Edit section: AST-based feature detectors)]

AST is an acronym standing for *accelerated segment test.* This test is a relaxed version of the SUSAN corner criterion. Instead of evaluating the circular disc only the pixels in a [Bresenham circle](http://en.wikipedia.org/wiki/Midpoint_circle_algorithm" \o "Midpoint circle algorithm) of radius r around the candidate point are considered. If n contiguous pixels are all brighter than the nucleus by at least t or all darker than the nucleus by t, then the pixel under the nucleus is considered to be a feature. This test is reported to produce very stable features.[[21]](http://en.wikipedia.org/wiki/Corner_detection#cite_note-fast-21) The choice of the order in which the pixels are tested is a so-called [Twenty Questions problem](http://en.wikipedia.org/wiki/Twenty_Questions). Building short decision trees for this problem results in the most computationally efficient feature detectors available.

The first corner detection algorithm based on the AST is FAST ([features from accelerated segment test](http://en.wikipedia.org/wiki/Features_from_accelerated_segment_test)).[[21]](http://en.wikipedia.org/wiki/Corner_detection#cite_note-fast-21) Although r can in principle take any value, FAST uses only a value of 3 (corresponding to a circle of 16 pixels circumference), and tests show that the best results are achieved with n being 9. This value of n is the lowest one at which edges are not detected. The order in which pixels are tested is determined by the [ID3 algorithm](http://en.wikipedia.org/wiki/ID3_algorithm) from a training set of images. Confusingly, the name of the detector is somewhat similar to the name of the paper describing Trajkovic and Hedley's detector.

Automatic synthesis of detectors[[edit](http://en.wikipedia.org/w/index.php?title=Corner_detection&action=edit&section=13" \o "Edit section: Automatic synthesis of detectors)]

Trujillo and Olague[[22]](http://en.wikipedia.org/wiki/Corner_detection" \l "cite_note-geneticprogramming-22) introduced a method by which [genetic programming](http://en.wikipedia.org/wiki/Genetic_programming) is used to automatically synthesize image operators that can detect interest points. The terminal and function sets contain primitive operations that are common in many previously proposed man-made designs.[Fitness](http://en.wikipedia.org/wiki/Fitness_function) measures the stability of each operator through the repeatability rate, and promotes a uniform dispersion of detected points across the image plane. The performance of the evolved operators has been confirmed experimentally using training and testing sequences of progressively transformed images. Hence, the proposed GP algorithm is considered to be human-competitive for the problem of interest point detection.

Bibliography[[edit](http://en.wikipedia.org/w/index.php?title=Corner_detection&action=edit&section=14" \o "Edit section: Bibliography)]

* 1. [**^**](http://en.wikipedia.org/wiki/Corner_detection#cite_ref-1) Shapiro, Linda and George C. Stockman (2001). *Computer Vision*, p. 257. Prentice Books, Upper Saddle River. [ISBN 0-13-030796-3](http://en.wikipedia.org/wiki/Special:BookSources/0130307963).
  2. [**^**](http://en.wikipedia.org/wiki/Corner_detection#cite_ref-moravec_2-0) H. Moravec (1980). ["Obstacle Avoidance and Navigation in the Real World by a Seeing Robot Rover"](http://www.ri.cmu.edu/pubs/pub_22.html). *Tech Report CMU-RI-TR-3 Carnegie-Mellon University, Robotics Institute*.
  3. [**^**](http://en.wikipedia.org/wiki/Corner_detection#cite_ref-harris_3-0) C. Harris and M. Stephens (1988). ["A combined corner and edge detector"](http://www.bmva.org/bmvc/1988/avc-88-023.pdf). *Proceedings of the 4th Alvey Vision Conference*. pp. 147–151.
  4. [**^**](http://en.wikipedia.org/wiki/Corner_detection#cite_ref-shitomasi_4-0) J. Shi and C. Tomasi (June 1994). ["Good Features to Track,"](http://citeseer.ist.psu.edu/shi94good.html). *9th IEEE Conference on Computer Vision and Pattern Recognition*. Springer.  
     C. Tomasi and T. Kanade (2004). ["Detection and Tracking of Point Features"](http://www.sciencedirect.com/science?_ob=ArticleURL&_udi=B6V14-49D6WH0-1&_user=10&_rdoc=1&_fmt=&_orig=search&_sort=d&view=c&_version=1&_urlVersion=0&_userid=10&md5=cecb6ab80d45107f1cedb17aa4b211fb). *Pattern Recognition* **37**: 165–168. [doi](http://en.wikipedia.org/wiki/Digital_object_identifier):[10.1016/S0031-3203(03)00234-6](http://dx.doi.org/10.1016%2FS0031-3203%2803%2900234-6).
  5. [**^**](http://en.wikipedia.org/wiki/Corner_detection#cite_ref-noble_5-0) A. Noble (1989). *Descriptions of Image Surfaces* (Ph.D.). Department of Engineering Science, Oxford University. p. 45.
  6. [**^**](http://en.wikipedia.org/wiki/Corner_detection#cite_ref-6) Förstner, W; Gülch (1987 1987). ["A Fast Operator for Detection and Precise Location of Distinct Points, Corners and Centres of Circular Features"](http://www.ipb.uni-bonn.de/fileadmin/publication/papers/1987/foerstner87.fast.pdf). *ISPRS*.
  7. ^ [***a***](http://en.wikipedia.org/wiki/Corner_detection#cite_ref-lindeberg94icip_7-0) [***b***](http://en.wikipedia.org/wiki/Corner_detection#cite_ref-lindeberg94icip_7-1) [***c***](http://en.wikipedia.org/wiki/Corner_detection#cite_ref-lindeberg94icip_7-2) T. Lindeberg (1994). ["Junction detection with automatic selection of detection scales and localization scales"](http://www.csc.kth.se/~tony/abstracts/Lin94-ICIP.html). *Proc. 1st International Conference on Image Processing* **I**. pp. 924–928. Unknown parameter |address= ignored ([help](http://en.wikipedia.org/wiki/Help:CS1_errors#parameter_ignored))
  8. ^ [***a***](http://en.wikipedia.org/wiki/Corner_detection#cite_ref-lindeberg98_8-0) [***b***](http://en.wikipedia.org/wiki/Corner_detection#cite_ref-lindeberg98_8-1) [***c***](http://en.wikipedia.org/wiki/Corner_detection#cite_ref-lindeberg98_8-2) [***d***](http://en.wikipedia.org/wiki/Corner_detection#cite_ref-lindeberg98_8-3) [***e***](http://en.wikipedia.org/wiki/Corner_detection#cite_ref-lindeberg98_8-4) [***f***](http://en.wikipedia.org/wiki/Corner_detection#cite_ref-lindeberg98_8-5) [***g***](http://en.wikipedia.org/wiki/Corner_detection#cite_ref-lindeberg98_8-6) [***h***](http://en.wikipedia.org/wiki/Corner_detection#cite_ref-lindeberg98_8-7) Tony Lindeberg (1998). ["Feature detection with automatic scale selection"](http://www.csc.kth.se/cvap/abstracts/cvap198.html). *International Journal of Computer Vision* **30** (2). pp. 77–116.
  9. ^ [***a***](http://en.wikipedia.org/wiki/Corner_detection#cite_ref-lindeberg94book_9-0) [***b***](http://en.wikipedia.org/wiki/Corner_detection#cite_ref-lindeberg94book_9-1) [***c***](http://en.wikipedia.org/wiki/Corner_detection#cite_ref-lindeberg94book_9-2) [***d***](http://en.wikipedia.org/wiki/Corner_detection#cite_ref-lindeberg94book_9-3) T. Lindeberg (1994). [*Scale-Space Theory in Computer Vision*](http://www.nada.kth.se/~tony/book.html). Springer. [ISBN](http://en.wikipedia.org/wiki/International_Standard_Book_Number) [0-7923-9418-6](http://en.wikipedia.org/wiki/Special:BookSources/0-7923-9418-6).
  10. ^ [***a***](http://en.wikipedia.org/wiki/Corner_detection#cite_ref-lindeberg08enc_10-0) [***b***](http://en.wikipedia.org/wiki/Corner_detection#cite_ref-lindeberg08enc_10-1) [***c***](http://en.wikipedia.org/wiki/Corner_detection#cite_ref-lindeberg08enc_10-2) [***d***](http://en.wikipedia.org/wiki/Corner_detection#cite_ref-lindeberg08enc_10-3) T. Lindeberg (2008/2009). [*Scale-Space*](http://www.nada.kth.se/~tony/abstracts/Lin08-EncCompSci.html). "Wiley Encyclopedia of Computer Science and Engineering". *Encyclopedia of Computer Science and Engineering (Benjamin Wah, ed), John Wiley and Sons* **IV**: 2495–2504. [doi](http://en.wikipedia.org/wiki/Digital_object_identifier):[10.1002/9780470050118.ecse609](http://dx.doi.org/10.1002%2F9780470050118.ecse609). [ISBN](http://en.wikipedia.org/wiki/International_Standard_Book_Number) [0-470-05011-X](http://en.wikipedia.org/wiki/Special:BookSources/0-470-05011-X).
  11. ^ [***a***](http://en.wikipedia.org/wiki/Corner_detection#cite_ref-schmid_11-0) [***b***](http://en.wikipedia.org/wiki/Corner_detection#cite_ref-schmid_11-1) [***c***](http://en.wikipedia.org/wiki/Corner_detection#cite_ref-schmid_11-2) K. Mikolajczyk, K. and C. Schmid (2004). ["Scale and affine invariant interest point detectors"](http://www.robots.ox.ac.uk/~vgg/research/affine/det_eval_files/mikolajczyk_ijcv2004.pdf) (PDF). *International Journal of Computer Vision* **60** (1): pp 63–86. [doi](http://en.wikipedia.org/wiki/Digital_object_identifier):[10.1023/B:VISI.0000027790.02288.f2](http://dx.doi.org/10.1023%2FB%3AVISI.0000027790.02288.f2). Unknown parameter |note= ignored ([help](http://en.wikipedia.org/wiki/Help:CS1_errors#parameter_ignored))
  12. [**^**](http://en.wikipedia.org/wiki/Corner_detection#cite_ref-kitchen82_12-0) L. Kitchen and A. Rosenfeld (1982). "Gray-level corner detection". *Pattern Recognition Letters* **1** (2). pp. 95–102.
  13. [**^**](http://en.wikipedia.org/wiki/Corner_detection#cite_ref-richards88_13-0) J. J. Koenderink and W. Richards (1988). "Two-dimensional curvature operators". *Journal of the Optical Society of America: Series A* **5** (number 7). pp. 1136–1141.
  14. [**^**](http://en.wikipedia.org/wiki/Corner_detection#cite_ref-brelin98feattrack_14-0) L. Bretzner and T. Lindeberg (1998). ["Feature tracking with automatic selection of spatial scales"](http://www.csc.kth.se/cvap/abstracts/cvap201.html). *Computer Vision and Image Understanding* **71,**. pp. 385–392.
  15. [**^**](http://en.wikipedia.org/wiki/Corner_detection#cite_ref-lindebergli97_15-0) T. Lindeberg and M.-X. Li (1997). ["Segmentation and classification of edges using minimum description length approximation and complementary junction cues"](http://www.csc.kth.se/cvap/abstracts/cvap186.html). *Computer Vision and Image Understanding* **67** (1). pp. 88–98.
  16. ^ [***a***](http://en.wikipedia.org/wiki/Corner_detection#cite_ref-sift_16-0) [***b***](http://en.wikipedia.org/wiki/Corner_detection#cite_ref-sift_16-1) D. Lowe (2004). ["Distinctive Image Features from Scale-Invariant Keypoints"](http://citeseer.ist.psu.edu/654168.html). *International Journal of Computer Vision* **60** (2): 91. [doi](http://en.wikipedia.org/wiki/Digital_object_identifier):[10.1023/B:VISI.0000029664.99615.94](http://dx.doi.org/10.1023%2FB%3AVISI.0000029664.99615.94).
  17. [**^**](http://en.wikipedia.org/wiki/Corner_detection#cite_ref-wangbrady_17-0) H. Wang and M. Brady (1995). ["Real-time corner detection algorithm for motion estimation"](http://www.sciencedirect.com/science?_ob=ArticleURL&_udi=B6V09-3Y450Y2-8&_coverDate=11%2F30%2F1995&_alid=442214633&_rdoc=1&_fmt=&_orig=search&_qd=1&_cdi=5641&_sort=d&view=c&_acct=C000057551&_version=1&_urlVersion=0&_userid=2493154&md5=b0979efc3df88572ec71d07e2e9f14da). *Image and Vision Computing* **13** (9): 695–703. [doi](http://en.wikipedia.org/wiki/Digital_object_identifier):[10.1016/0262-8856(95)98864-P](http://dx.doi.org/10.1016%2F0262-8856%2895%2998864-P).
  18. [**^**](http://en.wikipedia.org/wiki/Corner_detection#cite_ref-susan_18-0) S. M. Smith and J. M. Brady (May 1997). ["SUSAN – a new approach to low level image processing"](http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.24.2763). *International Journal of Computer Vision* **23** (1): 45–78. [doi](http://en.wikipedia.org/wiki/Digital_object_identifier):[10.1023/A:1007963824710](http://dx.doi.org/10.1023%2FA%3A1007963824710).  
      *S. M. Smith and J. M. Brady (January 1997), "Method for digitally processing images to determine the position of edges and/or corners therein for guidance of unmanned vehicle". UK Patent 2272285, Proprietor: Secretary of State for Defence, UK.*
  19. [**^**](http://en.wikipedia.org/wiki/Corner_detection#cite_ref-19) [GB patent 2272285](http://worldwide.espacenet.com/textdoc?DB=EPODOC&IDX=GB2272285), list of inventors (free format), "Determining the position of edges and corners in images", published 1994-05-11, issued 1994-05-11, assigned to Secr Defence
  20. [**^**](http://en.wikipedia.org/wiki/Corner_detection#cite_ref-hedley_20-0) M. Trajkovic and M. Hedley (1998). "Fast corner detection". *Image and Vision Computing* **16** (2): 75–87. [doi](http://en.wikipedia.org/wiki/Digital_object_identifier):[10.1016/S0262-8856(97)00056-5](http://dx.doi.org/10.1016%2FS0262-8856%2897%2900056-5).
  21. ^ [***a***](http://en.wikipedia.org/wiki/Corner_detection#cite_ref-fast_21-0) [***b***](http://en.wikipedia.org/wiki/Corner_detection#cite_ref-fast_21-1) E. Rosten and T. Drummond (May 2006). ["Machine learning for high-speed corner detection,"](http://citeseer.ist.psu.edu/741064.html). *European Conference on Computer Vision*.
  22. [**^**](http://en.wikipedia.org/wiki/Corner_detection#cite_ref-geneticprogramming_22-0) Leonardo Trujillo and Gustavo Olague (2008). ["Automated design of image operators that detect interest points"](http://cienciascomp.cicese.mx/evovision/olague_EC_MIT.pdf). *Evolutionary Computation* **16** (4): 483–507. [doi](http://en.wikipedia.org/wiki/Digital_object_identifier):[10.1162/evco.2008.16.4.483](http://dx.doi.org/10.1162%2Fevco.2008.16.4.483). [PMID](http://en.wikipedia.org/wiki/PubMed_Identifier) [19053496](http://www.ncbi.nlm.nih.gov/pubmed/19053496).