3 Feature Detection

Syllabus

Edge detection, corner detection, line and curve detection, active contours, SIFT and HOG descriptors, shape context descriptors, Morphological operations

Contents

3.1 Feature Detection

A feature is a piece of information about the content of an image in computer vision and image processing, usually concerning whether a certain portion of the image has certain attributes. Specific structures in the image, such as points, edges, or objects, might be used as features. A generic neighborhood procedure or feature detection performed to the image could potentially result in features. Other elements include motion in image sequences, as well as shapes defined by curves or the boundaries between distinct image sections. A feature, in a broader sense, is any piece of data that is useful to completing the computing job associated with a certain application. The feature idea is fairly broad, and the features used in a computer vision system might vary greatly depending on the situation at hand.

When features are defined in terms of local neighborhood operations applied to an image, a procedure known as **feature extraction**, it is possible to distinguish between feature detection approaches that produce local decisions about whether or not a given type of feature exists at a given image point, and those that produce non-binary data as a result.

Feature detection is a low-level image processing operation. That is, it is usually performed as the first operation on an image, and examines every pixel to see if there is a feature present at that pixel. If this is part of a larger algorithm, then the algorithm will typically only examine the image in the region of the features. As a built-in pre-requisite to feature detection, the input image is usually smoothed by a Gaussian kernel in a scale-space representation and one or several feature images are computed, often expressed in terms of local image derivatives operations.

Methods for computing abstractions of image information and making local choices whether or not there is an image feature of a certain type at each picture point are included in feature detection. The generated features will frequently take the shape of single dots, continuous curves, or connected regions, and will represent subsets of the image domain.

3.1.1 Edge Detection

Edge detection refers to a set of mathematical techniques for recognizing points in a digital image where the image brightness abruptly changes or, more formally, where there are discontinuities. The sharp fluctuations in image brightness are usually grouped into a collection of curved line segments called **edges**. The goal of detecting sharp changes in image brightness is to record significant events and changes in the world's

attributes. It may be demonstrated that, assuming very broad assumptions for an image creation model, picture brightness discontinuities are likely to correlate to :

- Discontinuities in depth,
- Discontinuities in surface orientation,
- Changes in material properties and
- Variations in scene illumination.

For picture segmentation, there are numerous edge detection algorithms in the literature. This section examines the most widely used discontinuity-based edge detection approaches. Roberts edge detection, Sobel edge detection, Prewitt edge detection, Kirsh edge detection, Robinson edge detection, Marr-Hildreth edge detection, LoG edge detection, and Canny edge detection are some of the approaches available.

Roberts edge detection

Lawrence roberts is the inventor of the roberts edge detection (1965). It computes a simple 2-D spatial gradient measurement on an image in a short amount of time. This approach highlights high spatial frequency regions, which frequently correspond to edges. The most common application of this technology is to give the operator a grayscale image that matches the output. Every pixel value in the output represents the estimated entire magnitude of the input image's spatial gradient at that place.

– 1	0
0	+ 1

G_x

0	– 1
+ 1	0

G_y

Sobel edge detection

Sobel proposed the Sobel edge detection method in 1970 (Rafael C.Gonzalez, 2004). The Sobel method of image segmentation edge detection uses the Sobel approximation to the derivative to find edges. At the points where the gradient is the highest, it comes before the edges. The Sobel approach emphasizes regions of high spatial frequency that correlate to edges by performing a 2-D spatial gradient quantity on an image. It's typically used to calculate the estimated absolute gradient magnitude at each location in a grayscale image. It's typically used to calculate the estimated absolute gradient magnitude at each location in a grayscale image. At the very least, the operator is made

up of a pair of 3 x 3 complication kernels, as shown in the table below. One kernel is just 90 degree rotated from the other. The Roberts cross operator is extremely similar to this.

– 1	– 2	– 1
0	0	0
+ 1	+ 2	+ 1

G_x

– 1	0	– 1
– 2	0	+ 2
– 1	0	+ 1

G_y

Marr-Hildreth edge detection

The Marr-Hildreth (1980) methodology is a continuous curves methodology for detecting edges in digital images where there are well-built and quick fluctuations in image brightness. It's simple and works by convolving the image with the LoG function or using DoGs as a rapid approximation. The edges are then detected by looking for zero-crossings in the filtered output. Because of its image structure when turned upside-down, the LoG technique is also known as the Mexican hat wavelet. The Marr-Hildreth edge detector has the following algorithm :

- Apply a Gaussian smoothing filter to the image.
- Smooth the image with a two-dimensional Laplacian (often the first two steps are combined into a single operation)
- Look for sign changes as you loop over the results. Mark as an edge if there is a sign change and the slope across the sign change is greater than a certain threshold.
- Although this is not how the edge detector was originally constructed, it is feasible to improve results by running the Laplacian result through a hysteresis similar to Canny's edge detection.

LoG edge detection

The Laplacian of Gaussian (LoG) was proposed by Marr(1982). The LoG of an image f(x, y) is a second order derivative defined as,

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

It has two effects, it smoothes the image and it computes the Laplacian, whch yields a double - edge image. Locating edges then consists of finding the zero crossings between the double edges. The digital implementation of the Laplacian function is commonly produced through the mask below,

0	– 1	0
– 1	4	– 1
0	– 1	0

G_x

– 1	– 1	– 1
– 1	8	– 1
– 1	– 1	– 1

G_y

The Laplacian is generally used to found whether a pixel is on the dark or light side of an edge.

Canny edge detection

The Canny edge detection technique is a standard edge detection technique in industry. It was developed by John Canny for his Master's thesis at MIT in 1983, and it still outperforms many of the more recent algorithms. Canny is a highly essential method for finding edges by isolating noise from the image before finding picture edges. Canny is a highly essential method for finding edges by isolating noise from the image before finding picture edges. The Canny approach is a better method since it preserves the features of the image's edges after applying the tendency to detect the edges and a substantial threshold setting. The following are the algorithmic steps :

- Convolve image f(r, c) with a Gaussian function to get smooth image f^(r, c). f^(r, c) = f(r,c)*G(r,c,6)
- Compute edge strength using the difference gradient operator first, then compute edge magnitude and direction as before.
- Suppress the gradient magnitude with non-maximal or critical suppression.
- Apply a threshold to the image with non-maximal suppression.

The Canny operation, unlike Roberts and Sobel, is not very sensitive to noise. It would be superior if the Canny detector worked properly.

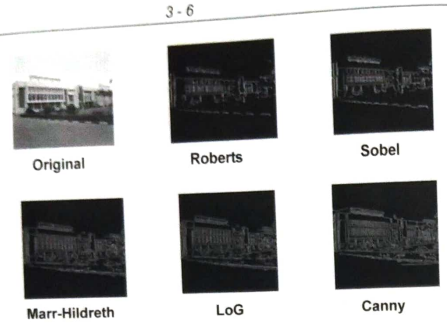


Fig. 3.1.1 Original,roberts, Sobel,Marr-Hildreth,LoG,Canny

3.1.2 Corner Detection

Corner detection is a method for extracting certain features and inferring the contents of an image in computer vision systems. Motion detection, picture registration, video tracking, picture mosaicing, panorama stitching, 3D reconstruction, and object recognition all require corner detection. The concept of corner detection is similar to that of interest point detection.

In the literature, the terms "corner," "interest point," and "feature" are used interchangeably, causing confusion. There are various blob detectors that can be referred to as "interest point operators," but are sometimes referred to as "**corner detectors**" incorrectly. Furthermore, there is a concept known as **ridge detection** that is used to identify the existence of elongated items.

Moravec corner detection algorithm

This is one of the first corner identification techniques, and a corner is defined as a point with a low self-similarity. The algorithm compares the similarity of a patch centered on the pixel to surrounding, mainly overlapping patches to see if a corner is present in the image. The Sum of Squared Differences (SSD) between the relevant pixels of two images is used to determine similarity. The smallest SSD between the patch and its neighbors is defined as corner strength (horizontal, vertical and on the two diagonals). The reason for this is that if this value is high, the variance along all shifts is either equal to or greater than it, resulting in all surrounding patches appearing to be different.

Susan corner detectors

This detector does not smooth the image or employ spatial derivatives. Rather, a circular mask is applied around each pixel, and the greyscale values of all the pixels

within the mask are compared to the nucleus pixel. Determine the number of pixels in the circular mask that are brighter than the nucleus. This circular mask was placed on several parts of a black rectangle with the USAN shown red colour. The USAN shrinks as it approaches an edge, and this reduction is more pronounced at corners, allowing SUSAN to be utilized for both line and edge identification. This corner detector is quick to compute and has a high rate of repeatability. This simple detector is insensitive to rotation and variations in illumination, but it is sensitive to noise.

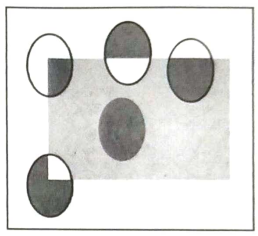


Fig. 3.1.2 USAN for different circular mask on uniform rectangle

The Moravec corner detection algorithm

This is one of the first corner identification methods, and it defines a corner as a location with low self-similarity. The system examines how similar a patch input on a pixel is to surrounding, largely overlapping patches to check if a corner is present. The sum of squared differences (SSD) between the two patches is used to determine similarity. A lower number implies greater resemblance. If the pixel is in an area of uniform intensity, the patches surrounding will appear similar. If a pixel is on an edge, nearby patches in a direction perpendicular to the edge will seem drastically different, whereas nearby patches in a direction parallel to the edge will only show a minor difference. If the pixel sits on a feature with a lot of variety in all directions, none of the patches around it will appear the same. The corner strength is defined as the smallest SSD between the patch and its neighbors (horizontal, vertical and on the two diagonals). If this value is locally maximum, then there is a feature of importance. One of the primary issues with this operator, as Moravec points out, is that it is not isotopic : If an edge exists that is not in the direction of the neighbors, the smallest SSD will be big, and the edge will be selected mistakenly as an interest point.

The Förstner corner detector

The Förstner algorithm is used to detect corners. In some circumstances, computing the location of a corner with sub-pixel accuracy is desirable. The Förstner algorithm is a least-square solution that finds the point nearest to all the tangent lines of the corner in a given window to arrive at an estimated answer. The concept is based on the notion that tangent lines cross at an ideal corner.

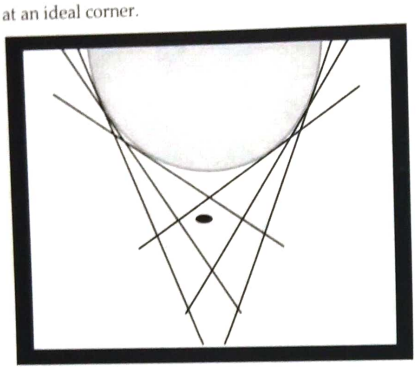


Fig. 3.1.3 Forstner corner detector

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

$$T_{x'}(x) = \nabla I(x')^T (x - x') = 0$$

where $\nabla I(x') = [I_x \; I_y]^T$ is the gradient vector of the image I at x'.

The point x_0 closest to all the tangent lines in the window N is :

$$x_0 = \underset{x \in \mathbb{R}^{2 \times 1}}{\text{argmin}} \int_{x' \in N} T_{x'}(x)^2 dx'$$

The distance from x_0 to the tangent lines Tx' is weighted by the gradient magnitude, thus giving more importance to tangents passing through pixels with strong gradients.

Solving for x_0 :

$$x_0 = \underset{x \in \mathbb{R}^{2 \times 1}}{\text{argmin}} \int_{x' \in N} (\nabla I(x')^T (x - x'))^2 \, dx'$$

$$= \underset{x \in \mathbb{R}^{2 \times 1}}{\text{argmin}} \int_{x' \in N} (x - x')^T \nabla I(x') \nabla I(x')^T (x - x') dx'$$

$$= \underset{x \in \mathbb{R}^{2 \times 1}}{\text{argmin}} (x^T A x - 2x^T b + c)$$

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

$$A = \int \nabla I(x') \nabla I(x')^T dx'$$

$$b = \int \nabla I(x') \nabla I(x')^T x' dx'$$

$$c = \int x'^T \nabla I(x') \nabla I(x')^T x' dx'$$

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

$$2Ax - 2b = 0 \Rightarrow Ax = b$$

Note that $A \in \mathbb{R}^{2 \times 2}$ is the 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 $x_0 = A^{-1} b$ 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 by minimizing the normalized residual, $\tilde{d}_{min} = \frac{c - b^T A^{-1} b}{\text{trace } A}$ 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.

3.1.3 Line and Curve Detection

The Hough transform and convolution-based algorithms are the most prominent line detectors in image processing. Line detection is an algorithm that takes a collection of n edge points and detects all the lines on which these edge points lie.

Hough transform

The Hough transform is a feature extraction approach used in image analysis, computer vision, and digital image processing. Its goal is to utilize a voting system to locate imperfect instances of objects within a given class of shapes. The Hough Transform (HT) is a well-known method for recognizing straight lines and curves in grayscale

photographs. It converts image data from image to parameter space, transforming curve detection into a peak detection problem.

This voting mechanism is carried out in a parameter space, from which object candidates are produced as local maxima in an accumulator space, which is generated explicitly by the Hough transform method. The traditional Hough transform was concerned with detecting lines in an image, but it was later expanded to recognizing positions of arbitrary shapes, most often circles or ellipses. Richard Duda and Peter Hart devised the Hough transform as we know it today in 1972, calling it a "**generalized Hough transform**" after the related 1962 patent. Dana H. Ballard popularized the transform in the computer vision community with his 1981 journal article "Generalizing the Hough transform to detect arbitrary shapes."

In automated analysis of digital images, a subproblem often arises of detecting simple shapes, such as straight lines, circles or ellipses. In many cases an edge detector can be used as a pre-processing stage to obtain image points or image pixels that are on the desired curve in the image space. There may be missing points or pixels on the required curves due to flaws in either the image data or the edge detector, as well as spatial variations between the ideal line/circle/ellipse and the noisy edge points as acquired by the edge detector. As a result, grouping the extracted edge features to an appropriate set of lines, circles, or equilateral triangles is frequently difficult. The Hough transform is designed to solve this problem by allowing users to organize edge points into object candidates using an explicit voting mechanism applied to a collection of specified image objects.

Detecting lines

Detecting straight lines is the most basic application of the Hough transform. In general, the straight line y = mx + b can be represented in the parameter space as a point (b, m). Vertical lines, on the other hand, are problematic. They'd cause the slope parameter m to have limitless values. As a result, Duda and Hart advocated using the Hesse normal form for computational reasons.

$$r = x \cos \theta + y \sin \theta$$

where r is the distance from the origin to the closest point on the straight line, and θ (theta) is the angle between the x axis and the line connecting the origin with that closest point.The intuition for this form, similarly to the plane equation, is that every vector on the line must be perpendicular (orthogonal) to the straight line of length r that comes from the origin. The intersection point between the function line and the perpendicular line that comes from the origin can be seen easily. So, for any point P on the line, the

vector must be orthogonal to the vector P_{0}-0 = P_{0}Therefore, we get that for any point P = (x, y) on the function line, the equation must be satisfied. Therefore, since and we get

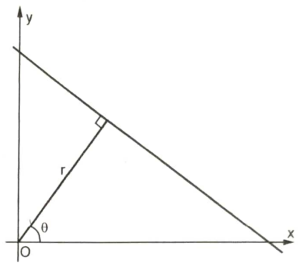


Fig. 3.1.4 Hough transform of line

As a result, each line of the image can be associated with a pair of numbers (r,θ). For the set of straight lines in two dimensions, the (r,θ) plane is frequently referred to as Hough space. The Hough transform is theoretically very similar to the two-dimensional Radon transform because of its representation. In truth, the Hough transform is theoretically equal to the Radon transform, although the two transformations are generally associated with different computational interpretations. The collection of all straight lines passing through a single point in the plane corresponds to a sinusoidal curve in the (r,θ) plane that is unique to that place. Sinusoids will develop when two or more points form a straight line and cross at the (r,θ) for that line. As a result, the problem of detecting collinear points can be transformed into a problem of locating concurrent curve

Algorithm Hough_Lines :

- The input image E is M ? N binary array with edge pixels marked with ones and other pixels marked as zeroes. Let P_d, Δ_d be the arrays containing the discretized intervals of the parameter space P ? [0, √N2 + M2] , Δ ? [0, π]. note : (Δ_{delta})
- Discretize the parameter spaces of P and Δ using sampling steps δp, δΔ, yielding acceptable and manageable resolution of R, T in the parameter space.

  Let A(R, T) be the counter array, initialized as zeroes,

  For each pixel E(i, j) = 1 and for h = 1…T
  - Let P = isinΔ_d(h) + jcosΔ_d(h)

- Find index k so that $P_d(k)$ is closer to P
- Increment A(k, h) by one

- Find all local maxima (k_p, h_p) such that $(k_p, h_p) > \tau$, where τ is a user defined threshold
- The output is a set of lines described by $(P_d(k_p), \Delta_d(h_p))$

Algorithm Hough_Curves :

- Let f(x, y, a) = 0 be the parametric form of the curves
- Discretize the parameters $a_1,...a_p$ with sampling steps yielding acceptable and manageable resolution of $s_1,...s_p$
- Let $A(s_1,...s_p)$ be the counter array, initialized as zeroes
- For each pixel E(i, j) = 1, increment all counters such that f(I, j, a) = 0
- Find all local maxima a_m such that $a_m > \tau$, where τ is a user defined threshold
- The output is a set of lines described by a_m

3.1.4 Active Contours

Michael Kass, Andrew Witkin, and Demetri Terzopoulos proposed the active contour model, commonly known as **snakes**, as a framework for outlining an object outline from a potentially noisy 2D image. Snakes are commonly utilized in applications such as object tracking, form identification, segmentation, edge detection, and stereo matching, and the snakes model is prominent in computer vision. A snake is a malleable, energy-saving spline that is impacted by constraint and image forces that pull it towards object outlines, as well as internal forces that resist deformation. Snakes can be thought of as a subset of the general concept of using energy minimization to fit a deformable model to an image. The active shape model is a discrete variant of this approach in two dimensions, using the point distribution model to limit the shape range to an explicit domain learned from a training set.

Snake model

Because the method requires prior knowledge of the intended contour shape, snakes do not address the complete challenge of identifying contours in photos. Instead, they rely on additional mechanisms including human involvement, interaction with a higher-level picture comprehension process, or information from picture data that is close in time

or location. A snake is an elastic curve c(s) = (x(s),y(s)),s ε [0,1], that moves through the spatial domain of an image to minimize the following energy functional,

$$E_{snake} = \int_0^1 \left[\frac{1}{2}(\alpha|c'(s)|^2 + \beta|c''(s)|^2) + \nabla E_{ext}(c)\right]$$

Where c'(s) and c''(s) are first and second derivatives of c(s) with respect to s, α and β are weighting parameters. The external energy E_{ext} is derived from the image data and takes smallest values at boundaries. The typical external force for gray-value image is $E_{ext} = -|\nabla G\sigma \otimes I|2$, where $G\sigma$ is the Gaussian kernel of standard deviation σ. A snake that minimizes E_{Snake} must satisfy the Euler equation,

$$\alpha c''(s) - \beta c''(s) - \nabla E_{ext} = 0$$

This can be viewed as a force balance equation of internal and external force

$$F_{int} + F_{ext} = 0$$

where Fint = $\alpha c''(s) - \beta c'' ''(s)$ and $F_{ext} = -\nabla E_{ext}$. The internal force F_{int} makes the curve to be continuous and smooth while the external force F_{ext} attracts the curve to the desired features of the image.

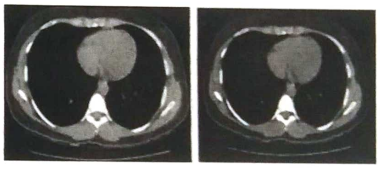


Fig. 3.1.5 Snake model

GVF : Gradient Vector Flow

X_u and Prince presented the Gradient Vector Flow (GVF) external force to counteract the 'myopia' character of the standard external force based on the picture edge map. The GVF is a vector field created by minimizing the following energy functional : v(x, y) = [u(x, y), v(x, y)].

$$E_{GVF} = \iint \mu(u_x^2 + u_y^2 + v_x^2 + v_y^2) + |\nabla f|^2 |v - \nabla f|^2 \, dxdy$$

Φ where f is the edge map of an image, usually, $f = -|\nabla G\sigma I|$; μ is $\otimes$ regularization parameter governing the tradeoff between the smoothness constraint and the fidelity term in. Using the calculus of variations, to minimize the EGVF is converted to solve the following Euler-Lagrange equation,

$$\begin{cases} \partial u/\partial t = \mu\nabla^2 u - |\nabla f|^2 (u - f_x) \\ \partial v/\partial t = \mu\nabla^2 v - |\nabla f|^2 (v - f_x) \end{cases}$$

where ∇^2 is the Laplacian operator

Fig. 3.1.6 GVF : Gradient Vector Flow

The balloon model

With the default active contour model, the balloon model resolves the following issues :

- The snake isn't attracted to the edges that are far away.
- If no significant visual forces are exerted on the snake, it will shrink inwards.
- A snake that is larger than the minima contour will eventually shrink into it, while a snake that is smaller will not discover the minima and will continue to shrink.

The balloon model introduces an inflation term into the forces acting on the snake

$$F_{inflation} = k_1 \vec{n}(s)$$

Where $\vec{n}(s)$ is the normal unitary vector of the curve at v(s) and k_1 denotes the force magnitude. To allow forces at image edges to overcome the inflation force, k_1 should have the same magnitude as the image normalization factor k and be smaller in value than k.

When employing the balloon model, three concerns arise :

- Instead of decreasing, the snake grows into the minima and will not locate any smaller minimum contours.
- The contour is significantly larger than the actual minima due to the outward thrust. After a stable solution has been discovered, this can be solved by lowering the balloon force.

- The inflation force can overcome weak edge forces, exacerbating the problem of snakes ignoring weaker features in an image.

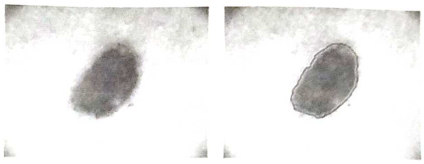


Fig. 3.1.7 The balloon model

Geometric active contours

Geometric active contours, also known as Geodesic Active Contours (GAC) or conformal active contours, are based on Euclidean curve shortening concepts. Object detection in the image causes contours to break and merge. These models are based on level sets and have been used extensively in medical image processing.

GAC's gradient descent curve evolution equation, for example.

$$\frac{\partial C}{\partial t} = g(I)(c + k)\vec{N} - (\nabla g, \vec{N})\vec{N}$$

Where g(I) is a halting function, c is a Lagrange multiplier, k is the curvature, and $\vec{N}$ is the unit inward normal. This particular form of curve evolution equation is only dependent on the velocity in the normal direction. It therefore can be rewritten equivalently in an Eulerian form by inserting the level set function ϕ into it as follows,

$$\frac{\delta\phi}{\delta t} = |\nabla\phi| \operatorname{div}\left(g(I)\frac{\nabla\phi}{|\nabla\phi|} + cg(I)\right)|\nabla\phi|$$

Active contours can handle topological changes during the gradient descent curve evolution using this basic yet strong level-set reformation. It has sparked enormous progress in adjacent domains, and the level-set technique, which employs numerical methods to solve the level-set reformulation, is now widely known.

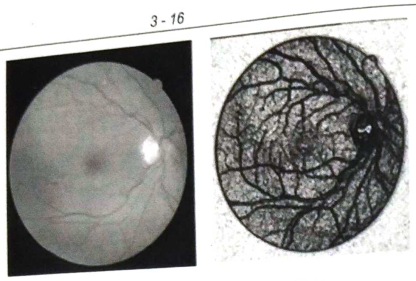


Fig. 3.1.8 Geometric active contours

3.2 SIFT Descriptors and HOG Descriptors

3.2.1 SIFT Descriptors

SIFT (scale-invariant feature transform) is a computer vision feature detection approach for detecting and describing local features in images. Object recognition, robotic mapping and navigation, picture stitching, 3D modeling, gesture recognition, video tracking, individual identification of wildlife, and match moving are some of the applications. SIFT (scale-invariant feature transform) is a computer vision feature detection approach for detecting and describing local features in images. Object recognition, robotic mapping and navigation, picture stitching, 3D modeling, gesture recognition, video tracking, individual identification of wildlife, and match moving are some of the applications.

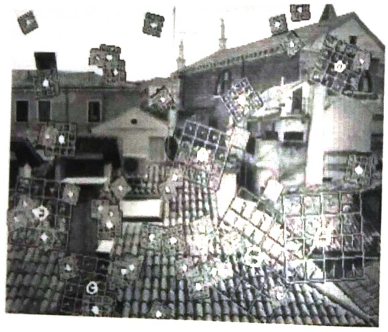


Fig. 3.2.1 A demonstration of interest points extracted by SIFT

Lowe's image feature generation approach divides a picture into a large number of feature vectors, each of which is invariant to image translation, scaling, and rotation, as well as partially invariant to light changes and robust to local geometric distortion. These characteristics are similar to neurons in the primary visual cortex that encode basic forms, color, and movement for objects. Another key property of these elements is that their relative placements in the original picture should remain constant from one image to the next. If only the four corners of a door were used as features, the recognition would succeed regardless of the door's position; but, if points in the frame were also used, the recognition would fail regardless of whether the door was open or closed. Similarly, features in articulated or flexible objects are unlikely to operate if the interior geometry of the object changes between two photos in the processing set. SIFT, on the other hand, recognizes and uses a significantly higher number of features from the images in practice, reducing the proportion of mistakes produced by these local changes in the average error of all feature matching mistakes.

3.2.2 HOG Descriptors

The histogram of oriented gradients (HOG) is a feature descriptor for object detection in computer vision and image processing. The technique counts the number of times a gradient orientation appears in a certain area of an image. The technique counts the number of times a gradient orientation appears in a certain area of an image. Edge orientation histograms, scale-invariant feature transform descriptors, and shape contexts are all comparable methods, but this one differs in that it is computed on a dense grid of uniformly spaced **cells** and uses overlapping local contrast normalization for enhanced accuracy.

The core idea behind the histogram of directed gradients descriptor is that the distribution of intensity gradients or edge directions can be used to characterize the appearance and shape of local objects within an image. The image is divided into small connected sections called **cells,** and a histogram of gradient directions is created for the pixels within each cell. The concatenation of these histograms is the descriptor. Local histograms can be contrast-normalized for better accuracy by generating an intensity measure across a larger portion of the image, known as a **block**, and then using this value to normalize all cells within the block. Because of this normalization, the invariance to changes in lighting and shadowing is improved. Compared to other descriptors, the HOG descriptor has a few major advantages. Except for object orientation, it is invariant to geometric and photometric alterations because it operates on local cells. Such changes would only be visible in wider geographic areas.

Algorithm implementation

Gradient computation

In many feature detectors used in image pre-processing, the first step in the calculation is to guarantee that the color and gamma values are normalized. However, as Dalal and Triggs point out, in HOG descriptor calculation, this step can be skipped because the subsequent descriptor normalization effectively achieves the same outcome. As a result, image pre-processing has little effect on performance. The computation of the gradient values, on the other hand, is the initial step in the calculation. Applying the 1-D centered, point discrete derivative mask in one or both horizontal and vertical axes is the most typical method. This method necessitates using the following filter to filter the image's color or intensity data.

$[-1, 0, 1]$ and $[-1, 0, 1]^T$

Orientation binning

The creation of cell histograms is the second stage in the process. Based on the values determined in the gradient computation, each pixel within the cell casts a weighted vote for an orientation-based histogram channel. The cells can be rectangular or radial in shape, and the histogram channels are evenly spaced from 0 to 180 degrees or 0 to 360 degrees, depending on whether the gradient is linear or nonlinear. In their human detection trials, Dalal and Triggs discovered that unsigned gradients combined with 9 histogram channels functioned best. When it comes to vote weight, pixel contribution can be either the gradient magnitude or a function of it. In testing, the magnitude of the gradient generates the best results.

Fig. 3.2.2 A demonstration of the histogram of gradients in each cell

3.2.3 Shape Context Descriptors

The shape context is meant to be a way of defining shapes that may be used to measure shape similarity and retrieve point correspondences. The main idea is to select n spots along a shape's outlines. Consider the n–1 vectors created by linking pi to all other points for each point pi on the form.

The collection of all these vectors provides a thorough description of the shape at that location, but it is far too comprehensive. The key idea is that the distribution of relative locations is a reliable, compact, and discriminative descriptor. So, given the point pi, the crude histogram of the remaining n–1 points' relative coordinates,

$$h_i(k) = \#\{q \neq p_i : (q - p_i) \in \text{bin}(k)\}$$

is defined to be the shape context of Pi.The bins are normally taken to be uniform in log-polar space. The shape context is a rich and discriminative descriptor, as evidenced by the Fig. 3.2.3 below, which depicts the shape contexts of two different renditions of the letter "A."

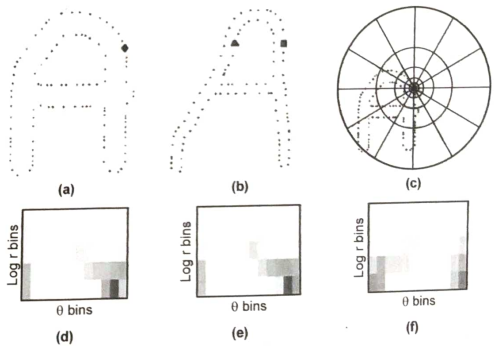


Fig. 3.2.3 (a) and (b) are the sampled edge points of the two shapes. (c) is the diagram of the log-polar bins used to compute the shape context. (d) is the shape context for the point marked with a circle in (a), (e) is that for the point marked as a diamond in (b), and (f) is that for the triangle. As can be seen, since (d) and (e) are the shape contexts for two closely related points, they are quite similar, while the shape context in (f) is very different.

Certain invariances are required for a feature descriptor to be helpful. It must be invariant to translation, scaling, minor perturbations, and rotation, depending on the application. Context naturally shapes translational invariance. Scale invariance is

obtained by normalizing all radial distances by the mean distance a between all the point pairs in the shape although the median distance can also be used. Shape contexts are empirically demonstrated to be robust to deformations, noise, and outliers using synthetic point set matching experiments. One can give perfect rotational invariance in form situations. One technique is to measure angles at each point relative to the direction of the tangent at that point (because the points are chosen on edges) (since the points are chosen on edges). As a result, the descriptor is totally rotationally invariant. As a result, the descriptor is totally rotationally invariant. However, because some local traits lose their discriminative value when not measured in the same frame, this isn't always desirable. Many applications, for example, distinguishing a "6" from a "9," forbid rotational invariance.

Step 1 : Finding a list of points on shape edges

The method presupposes that an object's shape is fundamentally captured by a finite selection of points on the object's internal or external outlines. These can be easily obtained by selecting a random collection of points from the edges using the Canny edge detector. Note that these locations do not have to, and in most cases do not, correspond to critical locations like curvature maxima or inflection points. It is preferable to sample the shape with roughly uniform spacing, though it is not critical.

Step 2 : Computing the shape context

Step 3 : Computing the cost matrix

Consider two points p and q with normalized K-bin histograms g(k) and h(k). Because shape contexts are histograms, it's only reasonable to apply the x^2 test statistic to calculate the "shape context cost" of matching the two points :

$$C_S = \frac{1}{2} \sum_{k=1}^{k} \frac{[g(k) - h(k)]^2}{g(k) + h(k)}$$

Between the unit vectors with angles θ_1 and θ_2, this is half the length of the chord in a unit circle. It has a value range of 0 to 1. The total cost of matching the two points might now be calculated as a weighted average of the two costs :

$$C_A = \frac{1}{2} \left\| \begin{pmatrix} \cos(\theta_1) \\ \sin(\theta_1) \end{pmatrix} - \begin{pmatrix} \cos(\theta_2) \\ \sin(\theta_2) \end{pmatrix} \right\|$$

$$C = (1 - \beta) C_S + \beta C_A$$

Now for each point pi on the first shape and a point qj on the second shape, compute the cost as indicated and label it Ci,j. This is the **cost matrix**.

Step 4 : Finding the matching that minimizes total cost

Now, a one-to-one matching p_i that matches each point p_i on shape 1 and q_j on shape 2 that minimizes the total cost of matching,

$$H(\pi) = \sum_i C(p_i, q_{\pi(i)})$$

is needed. This can be done in $O(N^3)$ time using the Hungarian method, although there are more efficient algorithms. To have robust handling of outliers, one can add "dummy" nodes that have a constant but reasonably large cost of matching to the cost matrix. This would cause the matching algorithm to match outliers to a "dummy" if there is no real match.

Step 5 : Modeling transformation

Given the set of correspondences between a finite set of points on the two shapes, a transformation can be estimated to map any point from one shape to the other. There are several choices for this transformation, described below. Affine, thin plate spline, regularized TPS.

Fig. 3.2.4 matching algorithm

Step 6 : Computing the shape distance

Now, a shape distance between two shapes P and Q . This distance is going to be a weighted sum of three potential terms :

Shape context distance : This is the symmetric sum of shape context matching costs over best matching points :

$$D_{SC}(P,Q) = \frac{1}{n} \sum_{p \in P} \arg \min_{q \in Q} C(p, T(q)) + \frac{1}{m} \sum_{q \in Q} \arg \min_{p \in P} C(p, T(q))$$

where $T(\bullet)$ is the estimated TPS transform that maps the points in Q to those in P.

Appearance cost : After establishing image correspondences and properly warping one image to match the other, one can define an appearance cost as the sum of squared brightness differences in Gaussian windows around corresponding image points :

$$D_{ac}(P,Q) = \frac{1}{n} \sum_{i=1}^{n} \sum_{\Delta \in Z^2} G(\Delta) [I_P(p_i + \Delta) - I_Q(T(q_{\pi(i)}) + \Delta)^2]$$

where I_p and I_Q are the gray-level images (is the image after warping) and G is a Gaussian windowing function.

Transformation cost : The final cost $D_{be}(P,Q)$ measures how much transformation is necessary to bring the two images into alignment. In the case of TPS, it is assigned to be the bending energy.

Now that we have a way of calculating the distance between two shapes, we can use a nearest neighbor classifier (k-NN) with distance defined as the shape distance calculated here. The results of applying this to different situations is given in the following section.

3.3 Morphological Operations

Morphological image processing is a set of non-linear processes that deal with the shape or morphology of picture features. Morphological procedures, according to Wikipedia, rely solely on the relative ordering of pixel values, rather than their numerical values, and are hence well adapted to the processing of binary images. Greyscale images can also be subjected to morphological treatments in which the light transfer functions are unknown and the absolute pixel values are of no or small importance. Morphological approaches use a small form or pattern called a **structuring element** to explore an image. The structuring element is placed in all available locations in the image and compared to the pixels in its immediate vicinity. Some operations determine if an element "fits" into the neighborhood, while others determine whether it "hits" or intersects it :

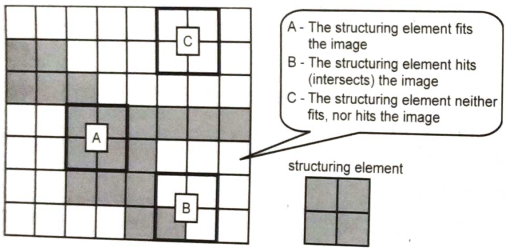


Fig. 3.3.1 Probing of an image with a structuring element

The structuring element is a small binary image, i.e. a small matrix of pixels, each with a value of zero or one :

A morphological operation on a binary image creates a new binary image in which the pixel has a non-zero value only if the test is successful at that place in the input picture.

- The size of the structuring element is determined by the matrix dimensions.
- The shape of the structuring element is determined by the arrangement of ones and zeros.
- The structuring element's origin is normally one of its pixels, though it can sometimes be outside the structuring element.

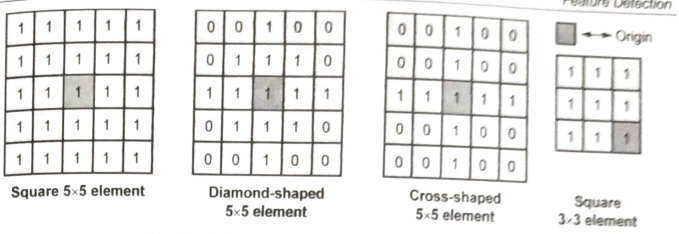
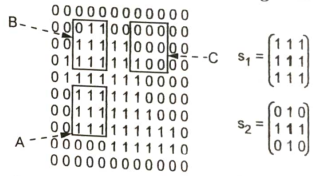


Fig. 3.3.2 Examples of simple structuring elements

The origin of the structuring matrix is usually defined as the **center of the matrix**, which is a common practice. Structuring elements, like convolution kernels in linear image filtering, have a role in moprphological image processing.

Each pixel of a structuring element is associated with the equivalent pixel of the neighborhood under the structuring element when it is inserted in a binary image. If each of the structuring element's pixels is set to one, the corresponding picture pixel is also one, the structuring element is said to fit the image. A structuring element is said to touch, or intersect, a picture if at least one of its pixels set to 1 corresponds to a pixel in the image.

Fig. 3.3.3 illustrates the dilation of a binary image. Note how the structuring element defines the neighborhood of the pixel of interest, which is circled. The dilation function applies the appropriate rule to the pixels in the neighborhood and assigns a value to the corresponding pixel in the output image. In the figure, the morphological dilation function sets the value of the output pixel to 1 because one of the elements in the neighborhood defined by the structuring element is on.



		A	B	C
fit	s_1	yes	no	no
	s_2	yes	yes	no
hit	s_1	yes	yes	yes
	s_2	yes	yes	no

Fig. 3.3.3 Fitting and hitting of a binary image with structuring elements s_1 and s_2

Fundamental operations

Operation	Rule
Dilation	The value of the output pixel is the *maximum* value of all the pixels in the input pixel's neighborhood. In a binary image, if any of the pixels is set to the value 1, the output pixel is set to 1.
Erosion	The value of the output pixel is the *minimum* value of all the pixels in the input pixel's neighborhood. In a binary image, if any of the pixels is set to 0, the output pixel is set to 0.

Erosion and dilation

The erosion of a binary image f by a structuring element s (denoted f θ s) produces a new binary image g = f θ s with ones in all locations (x, y) of a structuring element's origin at which that structuring element s fits the input image f, i.e. g(x, y) = 1 is s fits f and 0 otherwise, repeating for all pixel coordinates (x, y).

Greyscale image | Binary image by thresholding | Erosion : a 2×2 square structuring element

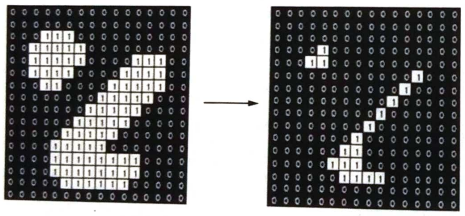
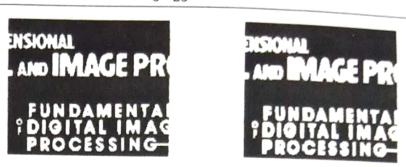


Fig. 3.3.4 Erosion : a 3*3 square structuring element

The **dilation** of an image f by a structuring element s (denoted f ⊗ s) produces a new binary image g = f ⊗ s with ones in all locations (x, y) of a structuring element's origin at which that structuring element s hits the the input image f, i.e. g(x, y) = 1 if s hits f and 0 otherwise, repeating for all pixel coordinates (x, y). Dilation has the opposite effect to erosion – it adds a layer of pixels to both the inner and outer boundaries of regions.



Binary image | Dilation : a 2×2 square structuring element

Fig. 3.3.5

Compound operations

Many morphological operations are represented as combinations of erosion, dilation, and simple set-theoretic operations such as the **complement** of a binary image :

$f^c(x, y) = 1$ if $f(x, y) = 0$ and $f^c(x, y) = 0$ if $f(x, y) = 1$, the **intersection** $h = f \cap g$ of two binary images f and g : $h(x,y) = 1$ if $f(x, y) = 1$ and $g(x, y) = 1$ and $h(x, y) = 0$ otherwise and the union $h = f \cup g$ of two binary images f and g : $h(x, y) = 1$ if $f(x, y) = 1$ or $g(x, y) = 1$ and $h(x, y) = 0$ otherwise :

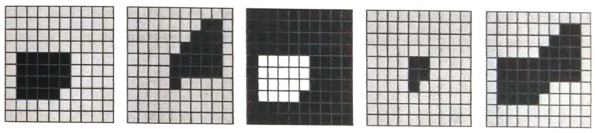


Fig. 3.3.6 Set operations on binary images:from left to right :a binary image f,a binary image g,the complement fc of f,the intersection f ∩ g and the union f ∪ g

The **opening** of an image f by a structuring element s (denoted by f • s) is an erosion followed by a dilation :

$$f \bullet s = (f \ominus s) \oplus s$$

Binary image | Opening : a 2 × 2 square structuring element

Fig. 3.3.7

The **closing** of an image f by a structuring element s (denoted by f • s) is a dilation followed by an erosion :

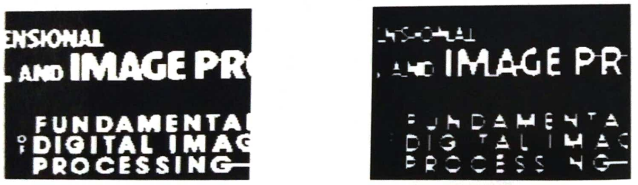
$$f \bullet s = (f \oplus s_{rot}) \ominus s_{rot}$$

Binary image — Closing : a 2 × 2 square structuring element

Fig. 3.3.8

The **hit and miss transform** allows to derive information on how objects in a binary image are related to their surroundings. The operation requires a matched pair of structuring elements, $\{s_1, s_2\}$, that probe the inside and outside, respectively, of objects in the image :

$$f \circledast \{s_1, s_2\} = (f \ominus s_1) \cap (f^c \ominus s_2)$$



Binary image — Hit and miss transform : an elongated 2×5 square structuring element

Fig. 3.3.9

If and only if s_1 translated to that pixel fits inside the object AND s_2 translated to that pixel fits outside the object, the hit and miss transform preserves that pixel as an object pixel. It is assumed that s_1 and s_2 do not intersect; otherwise, both fits would be impossible to occur at the same time. It's easier to understand if you think of s_1 and s_2 as a

single structuring element, with 1s for s_1 pixels and 0s for s_2 pixels; in this case, the hit-and-miss transform assigns 1 to an output pixel only if the object (with the value of 1) and background (with the value of 0) pixels in the structuring element exactly match the object (1) and background (0) pixels in the input image. If the two structural components provide the desired shape, the hit and miss transform can be used to detect certain shapes (spatial arrangements of object and background pixel values), as well as to thin or thicken object linear elements.

❑❑❑