UNIT-II

Points and Patches - An Introduction, Features detectors, Feature descriptors, Feature matching - Feature Tracking - Edge detectors - Edge linking - Successive approximation - Hough transform - Vanishing point.

Feature detection & Matching

Point and Patches.

Feature detection and matching are an essential component of many computer vision applications. A methods can be med

a) Point like interest operators
b) region-like interest operators
3) edges
4) straight lines.

Eg: If we wish to align two images to be matched. What kinds of feature might they can be seamlessly stitched into a composite mosaic. In some other examples one use to establish the correspondance between 2 images. so that set of correspondance should be identified. 3D model so based on examples some kind of feature can be computed should be detected and then match in order to establish an alignment or set of correspondances.

Point features can be used to find a sparse set of corresponding locations in different images.

often as a pre cursor to compute camera pose, which is pre-requisite for computing a dense set of correspondences. This is used to align different images. (When stitching image mosaics) the part of original image is replaced with another panorama.
+ They permit matching even in the presence of clutter and large scale and orientation changes. (occlusing → hitting together)

2 main approach for feature points & correspondences

1) Find feature in one image that can be accurately tracked using a local search tech. such as correlation or least squares. (used in images are taken from nearby viewpoints)
2) Independently detect features in all the images under consideration and then match features based on their local appearances. this is suitable for when a large amount of motion or appearance change is expected.

Keypoint detection and matching pipeline into 4 stages.

- Feature detection: Each image is searched for locations that are likely to match well in other images.
- Feature description → Each region around detected keypoint is converted into a more compact & stable descriptors that can be matched against other descriptors.

* Feature matching: Efficiently searches for likely matching candidates in other images.
* Feature tracking: An alternative to the third stage that only searches a small neighborhood around each detected feature and is therefore suitable for video processing.

1) Feature detectors:

How to find image locations where we can reliably find correspondences with other images (ie) what are good features to track. Featureless patches are nearly impossible or to localize. Although straight line segments at a single orientation suffer from the aperture pbm. It is only possible to align the patches along the direction normal to the edge direction.

(1) Patches with large contrast changes (gradients) are easy to localize

(2) Patches with gradients in atleast 2 different orientations are the easiest to localize. These orientations can be formalized by looking at the simplest possible matching criterion for comparing 2 image patches. (ie) their (weighted) summed square difference

$$E_{WSSD}(u) = \sum_i w(x_i) \left[I_1(x_i + u) - I_0(x_i) \right]^2$$

where I_0 & I_1 are the 2 images being (their intensity) compared $u = (u, v)$ is the displacement vector (shift component along x direction & y direction) $w(x)$ is a spatially varying weighting (or window) function, and the summation $\sum_i$ is over all the pixels in the patch.

When performing feature detection, we do not know which other image locations the feature will end up being matched against. ∴ we can only compute how stable this metric is w.r.t small variations in position Δu by comparing an image patch against itself. which is known as auto-correlation function or surface

$$E_{AC}(\Delta u) = \sum_i w(x_i) \left[I_0(x_i + \Delta u) - I_0(x_i) \right]^2$$

using a Taylor series expansion of the image function

$$I_0(x_i + \Delta u) \approx I_0(x_i) + \nabla I_0(x_i) \cdot \Delta u$$

we can approximate the auto-correlation surface as

$$E_{AC}(\Delta u) = \sum_i w(x_i) \left[I_0(x_i + \Delta u) - I_0(x_i) \right]^2$$

$$\approx \sum_i w(x_i) \left[I_0(x_i) + \nabla I_0(x_i) \cdot \Delta u - I_0(x_i) \right]^2$$

Auto correlation → measures the relationship b/w a Variables Current Value & its Past Value.

$$= \sum_i w(x_i) [\nabla I_0(x_i) \cdot \Delta u]^2$$

$$= \Delta u^T A \Delta u$$

where $\nabla I_0(x_i) = \left(\frac{\partial I_0}{\partial x}, \frac{\partial I_0}{\partial y}\right)(x_i)$ is the image gradient at x_i. This gradient can be computed using a variety of techniques. The classic 'Harris' detector uses a $[-2 -1 0 1 2]$ filter, but more modern variants convolve the image with horizontal and vertical derivatives of a Gaussian (typically with $\sigma = 1$). Auto correlation matrix A can be written as

$$A = w * \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

where we have replaced the weighted summations with discrete ~~correlations~~ convolutions with the weighting kernel w. This matrix can be interpreted as a tensor image, where the outer products of the gradients ∇I are convolved with a weighting function w to provide a per-pixel estimate of the local shape of the auto-correlation function.

Feature Description:

After detecting features we must match them. We must determine which feature come

from corresponding location in different images. In some situations like for video sequences or for stereo pairs that have been rectified, the local motion around each feature point may be mostly translational. In this case simple error metrics, such as the sum of square differences or normalized cross-correlation can be used to directly compare the intensities in small patches around each feature point, because feature points may not be exactly located a more accurate matching score can be computed by performing incremental motion refinement, as described even decrease performance

The local appearance of features will change in orientation and scale and sometimes even undergo affine deformation. Extracting a local scale, orientation or affine frame estimate and then using this to resample the patch before forming the feature descriptor is thus usually preferable.

Bias and gain normalization (MOPS)

For tasks that do not exhibit large amounts of foreshortening, such an image stitching ↳ reduce in time scale.

Feature

Feature may be specific structures in the image such as points, edges or object.

Two categories

1) Features that are in specific locations of the images, such as mountain peaks, building corners, doorways, or shaped patch of snow. These kind of localized features are often called keypoint features or interest points or corners.

2) Features that can be matched based on their orientation and local appearance (edge profiles) are called edges.

Interest point or feature point.

* It is the point which is expressive in texture.

Edges → can be matched based on their orientation & local appearance.

Point at which the direction of the boundary of the object changes abruptly or intersection point b/w 2 or more edge segments.

* described by the appearance of patches of pixels surrounding the point location.

Correlation based Method for Feature Sele

* Used for finding the association b/w the Continuous features & the class feature.

* It is a similarity measures b/w 2 features.

* If 2 features are linearly dependent their correlation coefficient is ± 1.

* If the features are uncorrelated the correlation coefficient is 0.

Linear Correlation Coefficient

For a pair of variables (x, y) the linear correlation coefficient r is

$$r = \frac{\sum (x_i - \bar{x}_i)(y_i - \bar{y}_i)}{\sqrt{\sum (x_i - \bar{x}_i)^2 \sum (y_i - \bar{y}_i)^2}}$$

If the value is higher than the threshold value (say 0.5) then the feature will be selected.

* The selection stops when the no. of features equals n logn.

$$E(u,v) = \sum_{x,y} w(x,y)\,[I(x+u, y+v) - I(x,y)]^2$$

$E(u,v)$ – Sum of all the sum squared difference (SSD)

u,v – are the x,y co-ordinate of every pixel in our 3×3 window

$I \rightarrow$ intensity Value of the pixel.

The features in the image are all pixels that have large value of $E(u,v)$ as defined by some threshold.

We have to maximize this for $E(u,v)$ for "corner detection"

We have to maximize the second term.

This can be written as

$$M = \sum w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

so the equation now become

$$E(u,v) \approx [u,v]\, M \begin{bmatrix} u \\ v \end{bmatrix}$$

By solving for the eigen vectors of M we obtain the direction for both largest & smallest increase in SSD

A score R is calculated for each window

$$R = \det M - K(\text{trace } M)^2$$

Approaches to find Interest Point

1) Based on the brightness of an image
2) Based on boundary extraction.

Introduction to Harris Corner Detector

Corner

A corner is a point whose local neighborhood stands in 2 dominant and different edge directions. ie junction of 2 edges.

Corner Detection → detects the locations of corner.

The idea is to consider a small window around each pixel P in an image. we want to identify all such pixel windows that are unique.

Uniqueness can be measured by shifting each window by a small amount in a given direction and measuring the amount of change that occurs in the pixel values.

- We take the sum squared difference (SSD) of the pixel values before and after the shift and identifying pixel windows where the SSD is large for shifts in all 8 directions.

Global descriptor:

- Describes the whole image
- If any change in part of the image, it will affect the resulting descriptor.

Image Patch → best descriptor for an image feature

Patches with similar content should have similar descriptors.

MOPS → Multi scale oriented patches. descriptors are formed using an 8×8 sampling of bias and gain normalized intensity values, with a sample spacing of five pixels relative to the detection scale.

SIFT Descriptor

The key idea is that local object appearance and shape can be described by the distribution of intensity gradients or edge directions.

- If we have images of different scales and rotations, we need to use the scale Invariant Feature transform (SIFT)

SIFT → we can change not only scale but

1) Scale
2) Rotation
3) Illumination
4) Viewpoint.

$\det M = \lambda_1 \lambda_2$

$\text{trace } M = \lambda_1 + \lambda_2$.

λ_1 & λ_2 are the eigen values of M.

When R is small λ_1 & λ_2 are small, the region is flat

R < 0, $\lambda_1 >> \lambda_2$ the region is edge.

R is large λ_1 & λ_2 are large, the region is corner.

Path → 9P of pixels in an image. 3+3 window.

Feature Descriptors:

- used to highlight specific features of the image. After detecting interest point we have to compute a descriptor for every one of them. 2D special pattern or 1D vector describing the appearance of the image path centred at the key point.

Local descriptor:

It is a compact representation of a Point's local neighborhood. Local descriptors try to resemble shape and appearance only in a local neighborhood around a point and thus very suitable for representing it in terms of matching.

simple normalized intensity patches perform reasonably well and are simple to implement
- In order to compensate for slight inaccuracies in the feature point detector, these multi-scale oriented patches (MOPS) are sampled at a (location, orientation, & scale) spacing of five pixels relative to the detection scale, using a coarser level of the image pyramid to avoid aliasing. To compensate for affine photometric variation, patch intensities are re-scaled so that their mean is zero and their variance is one.

Scale invariant Feature transform (SIFT)

SIFT features are formed by computing the gradient at each pixel in a 16x16 window around the detected keypoint, using the appropriate level of the Gaussian pyramid at which the key point was detected. The gradient magnitudes are downweighted by a Gaussian fall-off function.

In each 4x4 quadrant, a gradient orientation histogram is formed by adding the weighted gradient value to one of eight

orientation histogram bins. To reduce the effects of location and dominant orientation misestimation, each of the original 256 weighted Gradient magnitude is added to 2x2x2 histogram bins using trilinear interpolation. Softly distributing values to adjacent histogram bins is generally a good idea in any application where histogram are being computed, ex. for Hough transform.

PCA – SIFT

A simpler way to compute Principal component analysis (PCA) SIFT-like descriptors is to compute the x and y derivatives over a 39x39 patch and then reduce the resulting 3042-dimensional vector to 36 using principal component Analysis. Another popular variant of SIFT is SURF which uses box filters to approximate the derivatives and integrals used in SIFT.

Steerable Filters

These are combinations of derivative of Gaussian filters that permits the rapid computation of even and odd edge-like and corner-like features at all possible orientations. Because they are reasonably broad Gaussians, they too are somewhat insensitive to localization and orientation errors.

3) Feature matching

Once we have extracted features and their descriptors from 2 or more images, the next step is to establish some preliminary feature matches between those images. Two component is this operation.

① matching strategy → which determines which correspondance are passed on to the next stage for further processing.

② Data structure & algorithms → to perform this matching as quickly as possible.

Matching Strategy and error rates.

Determining which feature matches are reasonable to process further depends on the context in which the matching is being performed. say we are given 2 images that overlap to a fair amount. some feature may match with another image and some will not match because they are included or their appearance has changed too much. On otherhand if we are trying to recognize how many known objects appear in a cluttered scene (→ scattered). most of the feature may not match, a large no. of potentially matching objects must be searched, which requires more efficient strategies. So that Euclidean descriptors have been designed at feature can be directly used for rating potential matches. Euclidean distance metric, the simplest matching strategy is to set a threshold and to return all matches from other images within this threshold. setting the threshold too high result in (ie) too many false positives. incorrect matches being returned

Setting the threshold too low results in too many false negative, ie too many correct matches being missed.

We can quantity the performance of a matching algorithm at a particular threshold by first counting the no. of true and false matches and match failures, using the following definition.

TP - True positives, (ie) no. of correct matches

FN - false negative, matches that were not correctly detected

FP - false positives, proposed matches that are incorrect

TN - true negatives, non matches that were correctly rejected.

We can convert these numbers into unit rates by defining the following quantities.

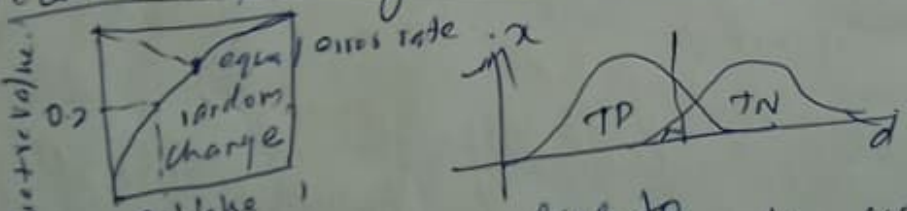
- True positive rate (TPR) $= \frac{TP}{TP+FN} = \frac{TP}{P}$
- False positive rate (FPR) $= \frac{FP}{FP+TN} = \frac{FP}{N}$
- Positive predictive value (PPV) $= \frac{TP}{TP+TP} = \frac{TP}{P}$

Accuracy (Acc)

$$Acc = \frac{TP+TN}{P+N}$$

In Information Retrieval, the term "Precision" (how many returned documents are relevant) is used instead of PPV and 'Recall' is used instead of TPR. (What fraction of relevant documents was found)

Any particular matching strategy can be rated by the TPR and FPR numbers. ideally, the TP rate will be close to 1 and the FP rate close to 0. As we vary the matching threshold, we obtain a family of such points, which are collectively known as the "receiver operating characteristic" (ROC curve).



The closer this curve lies to the upper left corner, (i.e) the large the area under the curve (AUC), the better its performance.

The ROC curve can be used to calculate the mean average precision, which is the average precision (PPV) as you vary the

threshold to select the best results, then the 2 top results etc.

(2) A better strategy is to simply match the nearest neighbor in feature space. Since some features may have no matches, a threshold is still used to reduce the no. of false +ves.

Ideally, this threshold itself will adapt to different regions of the feature space. If sufficient training data is available, it is sometimes possible to learn different thresholds for different features. We are simply given a collection of images to match, when stitching images or constructing 3D models from unordered photo collections. In this case, a useful heuristic can be to compare the nearest neighbour distance to that of the second nearest neighbor, preferably taken from an image that is known not to match the target. We can define this "Nearest neighbor distance Ratio"

$$NNDR = \frac{d_1}{d_2} = \frac{\|D_A - D_B\|}{\|D_A - D_C\|}$$

where d_1 and d_2 are the nearest and second nearest neighbor distances, D_A is the target descriptor, and D_B and D_C are its closest two neighbors.

The effectiveness of using these 3 different matching strategies for feature descriptors evaluated by NNDR produce ROC curves.

Efficient Matching.

Once decided on a matching strategy we will need to search efficiently for potential candidates. The simplest way to find all corresponding feature points is to compare all feature against all other features in each pair of potentially matching images. Unfortunately this is quadratic in the no. of extracted features, which makes it impractical for most applications.

A better approach is to devise an indexing structure, such as multi-dimensional feature tree or a hash table, to rapidly search

for features near a given feature. Such indexing structures can either be built for each image independently or globally for all the images in a given database which can potentially be faster, since it removes the need to iterate over the each image.

One of the simple technique to implement is multi-dimensional hashing, which maps descriptors in to fixed size buckets based on some function applied to each descriptor vectors. At matching time each feature is hashed into bucket and a search of nearby buckets is used to return potential candidates, which can then be sorted or graded to determine which are valid matches.

During the matching structure construction each 8×8 scaled oriented, and normalized mops path is converted into a 3element index by performing sums over different quadrats of the patch.

The resulting 3 values are normalized by their expected standard deviations and then mapped to the 2 nearest 1D bins. The 3D indices formed by concatenating the 3 quantized values are used to index the $2^3 = 8$ bins where the feature is stored.

At query time, only the primary (closest) indices are used to only a single 3D bins needs to be examined. The coefficients in the bin then be used to select K approximate nearest neighbors for further processing. A more complex, but more widely applicable, version of hashing is called locality sensitive hashing, which uses unions of independently computed hashing functions to index the features. Extend this technique to be more sensitive to the distribution of positive parameters space, which they call parameter-sensitive hashing.

Another widely used class of indexing structure are multi-dimensional search trees. The best known of these are kd-trees, also written as kd trees, which divide the multidimensional feature space along axis aligned hyperplanes, choosing the threshold along each axis so as to maximize some criterion, such as the search tree balance.

4) Feature Tracking -

An alternative to independently finding features in all candidate images and then matching them is to find a set of likely feature location in a first image and to then search for their corresponding location in subsequent images. This kind of detect then track approach is more widely used for video tracking applications, where the expected amount of motion and appearance deformation bw

adjacent frames is expected to be small.

The process of selecting good features to track is closely related to selecting good features.

For more general recognition application. In practice regions containing high gradient in both directions (i.e.) which have high eigen values in the auto-correlation matrix provide stable solution locations at which to find correspondences.

In subsequent frames, searching for locations where the corresponding patch has low squared difference often works well enough. The images are undergoing brightness change, explicitly compensating for such variations or using normalized cross-correlation may be preferable. If the search range is large, it is also often more efficient to use a hierarchical search strategy, which uses matches in lower resolution images to provide better initial guesses and hence speed up the search.

If features are being tracked over longer image sequences, their appearances can undergo larger changes. You then have to decide (1) whether to continue matching against the originally detected patch or (2) to resample each subsequent frame at the matching location. The former strategy is prone to failure. The later runs the risks of the feature drifting from its original location to some other location in the image.

The preferable soln is to compare the original patch to later image locations using an affine motion model. (1) First compare patches in neighboring frames using a translational model and then use the location estimates produced by this step to initialize an affine registration between the patch in the current frame and the base frame where a feature was first detected.

Features are only detected infrequently, (ie only in regions where tracking has failed. In the usual case, an area around the current predicted location of the feature is searched with an incremental registration algorithms. The resulting tracker is often called the Kanade-Lucas Tomasi (KLT) tracker.

One of the newest developments in feature tracking is the use of learning algorithms to build special-purpose recognizers to rapidly search for matching features anywhere in an image.

By taking the time to train classifiers on sample patches and their affine deformations, extremely fast and reliable feature detectors can be constructed, which enables much faster motions to be supported

Edges

Interest points are useful for finding image locations that can be accurately matched in 2D, edge points are more plentiful and often carry important semantic associations.

For eg, the boundaries of objects, which also correspond to occlusion (merge or combine with each other) events in 3D, are usually delineated by visible contours (can describe or bounding the shape or form of something) Other kinds of edges correspond to shadow boundaries or crease edges, where surface orientation changes rapidly. Isolated edge points can also be grouped into longer curves or contours as well as straight line segments. To draw outline of image edge detection is used.

Edge detection

Edges occur at boundaries b/w regions of different color, intensity or texture.

we can define an edge as a location of rapid intensity variation. Think of an image as a height field. On such a surface, edges occur at locations of steep slopes, or equivalently, in regions of closely packed contour lines (on a topographic map).

A mathematical way to define the slope and direction of a surface is through its gradient → Directional change in the intensity or color in an image.

$$J(x) = \nabla I(x) = \left(\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}\right)(x)$$

The local gradient vector J points in the direction of steepest ascent in the intensity function. Its magnitude is an indication of the slope or strength of the variation, while its orientation points in a direction perpendicular to the local contour.

unfortunately, taking image derivatives accentuates high frequencies and hence amplifies noise, since the proportion of noise to signal is larger at high frequecies. It is therefore prudent to smooth the image with a low-pass filter prior to computing the gradient.

Because we would like the respome of our edge detector to be independent of orientation, a circularly symmetric smoothing filter is desirable

The Gaussian is the only separable circularly symmetric filter and it is used in most edge detection algorithms.

Because differentiation is a linear operation, it commutes with other linear filtering operations. The gradient of the smoothed image can therefore be written as

$$\nabla I_\sigma(x) = \nabla [G_\sigma(x) * I(x)] = [\nabla G_\sigma](x) * I(x)$$

we can convolve the image with the horizontal and vertical derivatives of the Gaussian kernel function.

$$\nabla G_\sigma(x) = \left(\frac{\partial G_\sigma}{\partial x}, \frac{\partial G_\sigma}{\partial y}\right)(x) =$$

$$= [-x \; -y] \frac{1}{\sigma^3} \exp\left(\frac{-x^2+y^2}{2\sigma^2}\right)$$

The parameter σ indicates the width of the Gaussian. (standard deviation of the distribution).

For many applications, we wish to thin such a continuous gradient image to only return isolated edges. i.e as single pixels at discrete locations along the edge contours. This can be achieved by looking the maxima in the edge strength (gradient magnitude) in a direction perpendicular to the edge orientation. (i.e) along the gradient direction.

Finding this maximum corresponds to taking a directional derivative of the strength field in the direction of the gradient and then looking for zero crossings. The desired directional derivatives is equivalent to the dot product b/w a second gradient operator and the results of the first.

$$S_\sigma(x) = \nabla \cdot J_\sigma(x) = [\nabla^2 G_\sigma](x) * I(x)$$

The gradient operator dot product with the gradient is called the laplacian. The convolution kernel

$$\nabla^2 G_\sigma(x) = \frac{1}{\sigma^3}\left(2 - \frac{x^2+y^2}{2\sigma^2}\right) \exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)$$

is therefore called the Laplacian of Gaussian (LOG) kernal.

This kernal can be split into 2 separable parts.

$$\nabla^2 G_\sigma(x) = \frac{1}{\sigma^3}\left(1-\frac{x^2}{2\sigma^2}\right) G_\sigma(x)\, G_\sigma(y) + \frac{1}{\sigma^3}\left(1-\frac{y^2}{2\sigma^2}\right) G_\sigma(y)\, G_\sigma(x)$$

which allows for a much more efficient implementation using separable filtering. It is quite common to replace the Laplacian of Gaussian convolution with a difference of Gaussian (DoG) computation, since the kernel shapes are qualitatively small. It is not strictly necessary to take differences blw adjacent levels when computing the edge field. The finer Gaussian is a noise reduced version of the original image. The coarser Gaussian is an estimate of the average intensity over a larger region.

Scale selection and Blur estimation

All the filters require ~~Selection~~ Selection of a spatial scale parameter σ. If we want to detect sharp edges, the width of the filter can be determine from image noise characteristics. If we want to detect edges that occur at different resolutions; different scales may be necessary. ~~a scale-space approach~~ selects edges at that detects and then

Given a known image noise level, their technique computes, for every pixel, the minimum scale at which an edge ~~edge~~ can be reliably detected. First computes gradients densely over an image by selecting among gradient estimates computed at different scales, based on their gradient magnitudes. It then performs a similar estimate of minimum scale for directed second derivatives and uses zero crossings of this

Latter quantity to robustly select edges. Final step is, the blur width of each edge can be computed from the distance between extrema in the second derivative response minus the width of the Gaussian filter.

Color edge detection.

Noticeable edges between iso-luminant colors (colors that have the same luminance) are useful cues but fail to be detected by grayscale edge operators.

One simple approach is to combine the outputs of grayscale detectors run on each color band separately. It we simply sum up the gradients in each of the color bands, the signed gradients may actually cancel each other (eg a pure red-to-green edge). We could also detect edges independently

in each band and then take the union of these, but this might lead to thickened or doubled edges that are hard to link.

A better approach is to compute the oriented energy in each band (e.g. using a second-order steerable filter) and then sum up the orientation weighted energies and find their joint best orientation.

An alternative approach is to estimate local color statistics in regions around each pixel. This has the advantage that more sophisticated techniques (eg 3D color histograms) can be used to compare regional statistics and that additional measures such as texture, can also be considered.

Edge Linking

If the ~~Isolated~~ edges have been detected using zero crossings of some function, linking them up is straightforward since adjacent edgels share common endpoints. Linking the edgels into chains involves picking up an unlinked edgel and following its neighbor in both directions.

→ A pixel in an image that is recognised as edge

Either a sorted list of edgels (sorted first by x co-ordinates and then y co-ordinates) or a 2D array can be used to accelerate the neighbor finding. If edges were not detected using zero crossings, finding the continuation of an edgel can be tricky. In this case comparing the orientation and phase of adjacent edgels can be used for disambiguation.

Ideas from connected component computation can also be used to make the edge linking process even faster.

Once the edgels have been linked into chains, we can apply an optional thresholding with hysteresis to remove low-strength contour segments. The basic idea of hysteresis is to set 2 different thresholds and allow a curve being tracked above the higher threshold to dip in strength down to the lower threshold..

Linked edgel lists can be encoded more compactly using a variety of alternative representations.

A chain code encodes a list of connected points lying on an N_8 grid

using a three bit code corresponding to the eight cardinal directions (N, NE, E, SE, S, SW, W, NW) between a point and its successor.

A more useful representation is the arc length parameterization of a contour $x(s)$, where s denotes the arc length along a curve.

The advantage of the arc-length parameterization is that it makes matching and processing (eg smoothing) operations much easier. Consider the 2 curves describing similar shapes

To compare the curves, first subtract the average values $x_0 = \int x(s)$ from each descriptor. Next we rescale each descriptor so that s goes from 0 to 1 instead of 0 to S (ie) we divide $x(s)$ by S.

Finally we take the Fourier transform of each normalized descriptor, treating each $x = (x, y)$ value as a complex number. If the original curves are the same, the resulting Fourier transforms should differ only by a scale change in magnitude plus a constant complex phase shift due to rotation, and a linear phase shift in the domain, due to different starting points for S.

It can also be used to smooth curves in order to remove digitization noise. However, if we just apply a regular smoothing filter, the curve tends to shrink on itself. Controlling shrinkage without affecting its "sweep" wavelets allow the "character" of a curve to be interactively modified

Successive approximation

Describing curve as a series of 2D locations $x_i = x(s_i)$ provides a general representation suitable for matching and further processing. In many applications, however, it is preferable to approximate such a curve with a simpler representation eg as a piecewise-linear polyline or as a B-spline curve.

Line simplification

One method is recursively subdivide the curve at the point furthest away from the line joining the 2 endpoints (or the current coarse polyline approximation

Once the line simplication has been computed, it can be used to approximate the original curve. If a smoother representation or visualization is desired either approximating or interpolating splines or Curves can be used.

Hough Transforms

It is a feature extraction technique. It is used to find imperfect instances of objects within a certain class of shapes by a voting procedure.

This voting procedure is carried out in a parameter space, from which object candidates are obtained as local maxima in a so-called accumulator space that is explicitly constructed by the algorithm for computing the Hough transform.

First it was used for the identification of lines in the image, but later it is extended to identify positions of arbitrary shapes, most commonly circles or ellipses.

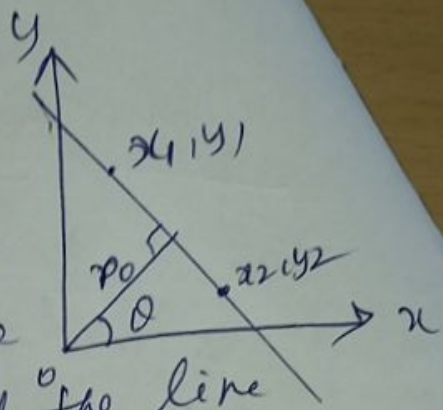
Detecting Lines

The simplest case of Hough transform is detecting straight lines. In general, the straight line $y = mx + b$ can be represented as a point (b, m) in the parameter space. We use

Hesse normal form:

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

where r is the distance from the origin to the closest point on the straight line, and θ is the angle between the x axis and the line connecting the origin with the closest point.



Algorithm:

Hough space

$$p = x_1\cos(\theta) + y_1\sin(\theta)$$

$$p = x_2\cos(\theta) + y_2\sin(\theta)$$

Hough space is represented with $p \propto \theta$ where the horizontal axis is for the θ values and the vertical axis are for the p values. The mapping of edge points onto the Hough space works in a similar manner except that an edge point (x_i, y_i) now generate a cosine curve in the hough space instead of a straight line. This representation of a line estimates the issue of the unbounded value of α that arises when dealing with Vertical line.

Vanishing Points

It is a point on the image plane of a perspective drawing where 2D perspective projections of mutually parallel lines in 3D space appear to converge. when the set of parallel lines is perpendicular to a picture plane, the construction is known as one-point perspective and their vanishing point corresponds to the "Oculus" or Eye Point, from which the image should be viewed for correct perspective Geometry.

* a point at which receding parallel lines seem to meet when represented in linear perspective.
* a point at which something disappears or ceases to exist.

* Finding Vanishing Points help 1) refine their position in the image
* to determine the intrinsic & extrinsic orientation of the camera.