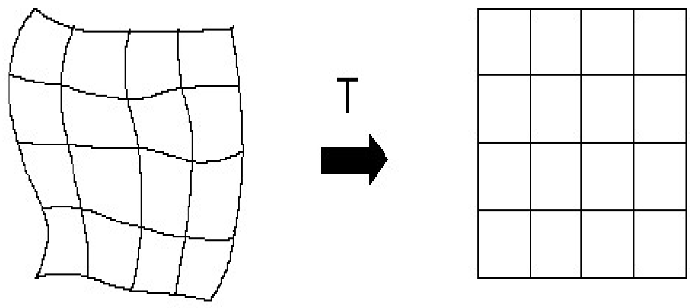
UNIT-5

IMAGE REGISTRATION & SEGMENTATION

5.1 REGISTRATION

5.1.1 Geometric Transformation

Geometric transforms permit the elimination of geometric distortion that occurs when an image is captured. An example is an attempt to match remotely sensed images of the same area taken after one year, when the more recent image was probably not taken from precisely the same position. To inspect changes over the year, it is necessary first to execute a geometric transformation, and then subtract one image from the other.



A geometric transform is a vector function T that maps the pixel (x,y) to a new position (x',y').

$$x' = T_x(x,y), \qquad y' = T_y(x,y)$$

The transformation equations are either known in advance or can be determined from known original and transformed images. Several pixels in both images with known correspondence are used to derive the unknown transformation.

5.1.1.1 Plane to Plane Transformation

General case of finding the co-ordinates of a point in the output image after a geometric transform. It is usually approximated by a polynomial equation

$$x' = \sum_{r=0}^{m} \sum_{k=0}^{m-r} a_{rk}\, x^r\, y^k \qquad y' = \sum_{r=0}^{m} \sum_{k=0}^{m-r} b_{rk}\, x^r\, y^k$$

This transform is linear with respect to the coefficients a_{rk}, b_{rk}. If pairs of corresponding points (x,y), (x',y') in both images are known, it is possible to determine a_{rk}, b_{rk} by solving a set of linear equations. More points than coefficients are usually used to get robustness. If the geometric transform does not change rapidly depending on position in the image, low order approximating polynomials, m=2 or m=3, are used, needing at least 6 or 10 pairs of corresponding points. The corresponding points should be distributed in the image in a way that can express the geometric transformation - usually they are spread uniformly. The higher the degree of the approximating polynomial, the more sensitive to the distribution of the pairs of corresponding points the geometric transform.

In practice, the geometric transform is often approximated by the bilinear transformation. 4 pairs of corresponding points are sufficient to find transformation coefficients

$$\begin{aligned} x' &= a_0 + a_1 x + a_2 y + a_3 xy \\ y' &= b_0 + b_1 x + b_2 y + b_3 xy \end{aligned}$$

Even simpler is the affine transformation for which three pairs of corresponding points are sufficient to find the coefficients

$$\begin{aligned} x' &= a_0 + a_1 x + a_2 y \\ y' &= b_0 + b_1 x + b_2 y \end{aligned}$$

The affine transformation includes typical geometric transformations such as rotation, translation, scaling and skewing.

Rotation - by the angle phi about the origin

$$\begin{aligned} x' &= x \cos\phi + y \sin\phi \\ y' &= -x \sin\phi + y \cos\phi \end{aligned}$$

Change of scale - a in the x axis and b in the y axis

$$\begin{aligned} x' &= ax \\ y' &= bx \end{aligned}$$

Skewing by the angle phi

$$\begin{aligned} x' &= x + y \tan\phi \\ y' &= y \end{aligned}$$

5.2 <u>Segmentation</u>

5.2.1 **<u>Introduction</u>**

If an image has been preprocessed appropriately to remove noise and artifacts, segmentation is often the key step in interpreting the image. Image segmentation is a process in which regions or features sharing similar characteristics are identified and grouped together. Image segmentation may use statistical classification, thresholding, edge detection, region detection, or any combination of these techniques. The output of the segmentation step is usually a set of classified elements, Most segmentation techniques are either region-based or edge based.

- Region-based techniques rely on common patterns in intensity values within a cluster of neighboring pixels. The cluster is referred to as the region, and the goal of the segmentation algorithm is to group regions according to their anatomical or functional roles.
- Edge-based techniques rely on discontinuities in image values between distinct regions, and the goal of the segmentation algorithm is to accurately demarcate the boundary separating these regions.

Segmentation is a process of extracting and representing information from an image is to group pixels together into regions of similarity.

Region-based segmentation methods attempt to partition or group regions according to common image properties. These image properties consist of :

i. Intensity values from original images, or computed values based on an image operator
ii. Textures or patterns that are unique to each type of region
iii. Spectral profiles that provide multidimensional image data

Elaborate systems may use a combination of these properties to segment images, while simpler systems may be restricted to a minimal set on properties depending of the type of data available.

Categories of Image Segmentation Methods

- Clustering Methods
- Histogram-Based Methods
- Edge Detection Methods
- Region Growing Methods

Level Set Methods
Graph portioning methods
Watershed Transformation
Neural Networks Segmentation

- Model based Segmentation/knowledge-based segmentation - involve active shape and appearance models, active contours and deformable templates.
- Semi-automatic Segmentation - Techniques like Livewire or Intelligent Scissors are used in this kind of segmentation.

5.2.2 **Pixel-Based Approach**

Gray level thresholding is the simplest segmentation process. Many objects or image regions are characterized by constant reflectivity or light absorption of their surface. Thresholding is computationally inexpensive and fast. Thresholding can easily be done in real time using specialized hardware. Complete segmentation can result from thresholding in simple scenes.

$$R = \bigcup_{i=1}^{S} R_i \qquad R_i \cap R_j = \emptyset \qquad i \neq j$$

Search all the pixels f(i,j) of the image f. An image element g(i,j) of the segmented image is an object pixel if f(i,j) >= T, and is a background pixel otherwise.

$$\begin{aligned} g(i,j) &= 1 \quad \text{for } f(i,j) \geq T \\ &= 0 \quad \text{for } f(i,j) < T \end{aligned}$$

Correct threshold selection is crucial for successful threshold segmentation. Threshold selection can be interactive or can be the result of some threshold detection method

5.2.2.1 Multi-Level Thresholding

The resulting image is no longer binary

$$\begin{aligned} g(i,j) &= 1 \quad \text{for } f(i,j) \in D_1 \\ &= 2 \quad \text{for } f(i,j) \in D_2 \\ &= 3 \quad \text{for } f(i,j) \in D_3 \\ &= 4 \quad \text{for } f(i,j) \in D_4 \\ &\dots \\ &= n \quad \text{for } f(i,j) \in D_n \\ &= 0 \quad \text{otherwise} \end{aligned}$$

5.2.2.2 Local Thresholding

It is successful only under very unusual circumstances. Gray level variations are likely due to non-uniform lighting, non-uniform input device parameters or a number of other factors.

$$T=T(f)$$

5.2.2.3 Threshold Detection Method

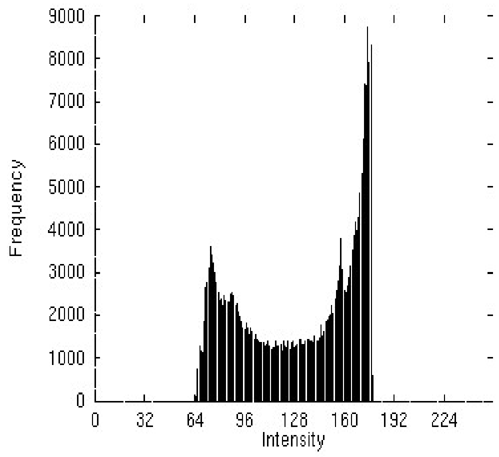
If some property of an image after segmentation is known a priori, the task of threshold selection is simplified, since the threshold is chosen to ensure this property is satisfied. A printed text sheet may be an example if we know that characters of the text cover 1/p of the sheet area.

- P-tile-thresholding

  - choose a threshold T (based on the image histogram) such that 1/p of the image area has gray values less than T and the rest has gray values larger than T
  - in text segmentation, prior information about the ratio between the sheet area and character area can be used
  - if such a priori information is not available - another property, for example the average width of lines in drawings, etc. can be used - the threshold can be determined to provide the required line width in the segmented image

More complex methods of threshold detection

- based on histogram shape analysis
- bimodal histogram - if objects have approximately the same gray level that differs from the gray level of the background



- **Mode method -** find the highest local maxima first and detect the threshold as a minimum between them. To avoid detection of two local maxima belonging to the same global maximum, a minimum distance in gray levels between these maxima is usually required or techniques to smooth histograms are applied.

5.2.3 Region-Based Approach

5.2.3.1 Region-Growing Based segmentation

Homogeneity of regions is used as the main segmentation criterion in region growing.
The criteria for homogeneity:

- gray level
- color
- texture
- shape
- model

The basic purpose of region growing is to segment an entire image R into smaller sub-images, Ri, i=1,2,….,N. which satisfy the following conditions:

$$R = \bigcup_{i=1}^{N} R_i; \quad R_1 \bigcap R_j = \Phi, \; i \neq j$$

$$H(R_i) = True; \; i = 1,2,\ldots, N;$$

$$H(R_i \bigcup R_j) = False, \; i \neq j;$$

5.2.3.2 Region Splitting

The basic idea of region splitting is to break the image into a set of disjoint regions, which are coherent within themselves:

- Initially take the image as a whole to be the area of interest.
- Look at the area of interest and decide if all pixels contained in the region satisfy some *similarity constraint.*
- If TRUE then the area of interest corresponds to an entire region in the image.
- If FALSE split the area of interest (usually into four equal subareas) and consider each of the sub-areas as the area of interest in turn.
- This process continues until no further splitting occurs. In the worst case this happens when the areas are just one pixel in size.

If only a splitting schedule is used then the final segmentation would probably contain many neighboring regions that have identical or similar properties. We need to merge these regions.

5.2.3.3 Region Merging

The result of region merging usually depends on the order in which regions are merged. The simplest methods begin merging by starting the segmentation using regions of 2x2, 4x4 or 8x8 pixels. Region descriptions are then based on their statistical gray level
properties. A region description is compared with the description of an adjacent region; if they match, they are merged into a larger region and a new region description is computed. Otherwise regions are marked as non-matching.
Merging of adjacent regions continues between all neighbors, including newly formed ones. If a region cannot be merged with any of its neighbors, it is marked `final' and the merging process stops when all image regions are so marked.

Merging Heuristics:

- Two adjacent regions are merged if a significant part of their common boundary consists of weak edges
- Two adjacent regions are also merged if a significant part of their common boundary consists of weak edges, but in this case not considering the total length of the region borders.

Of the two given heuristics, the first is more general and the second cannot be used alone because it does not consider the influence of different region sizes.
Region merging process could start by considering

- small segments (2*2,…,8*8) selected a priori from the image
- segments generated by thresholding
- regions generated by a region splitting module

The last case is called as “Split and Merge” method. Region merging methods generally use similar criteria of homogeneity as region splitting methods, and only differ in the direction of their application.

5.2.3.4 Split & Merge

To illustrate the basic principle of split and merge methods, let us consider an imaginary image.

• Let I denote the whole image shown in Fig. (a)

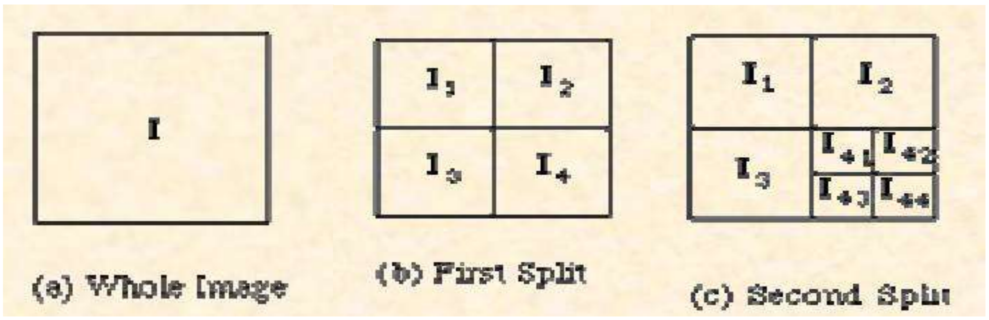
• Not all the pixels in Fig (a) are similar. So the region is split as in Fig. (b).

• Assume that all pixels within each of the regions I1, I2 and I3 are similar, but those in I4 are not.

• Therefore I4 is split next, as shown in Fig. (c).

• Now assume that all pixels within each region are similar with respect to that region, and that after comparing the split regions, regions I43 and I44 are found to be identical.

• These pair of regions is thus merged together, as in shown in Fig. (d).



A combination of splitting and merging may result in a method with the advantages of both the approaches. Split-and-merge approaches work using pyramid image representations. Regions are square-shaped and correspond to elements of the appropriate pyramid level.

If any region in any pyramid level is not homogeneous (excluding the lowest level), it is split into four sub-regions -- these are elements of higher resolution at the level below. If four regions exist at any pyramid level with approximately the same value of homogeneity measure, they are merged into a single region in an upper pyramid level.

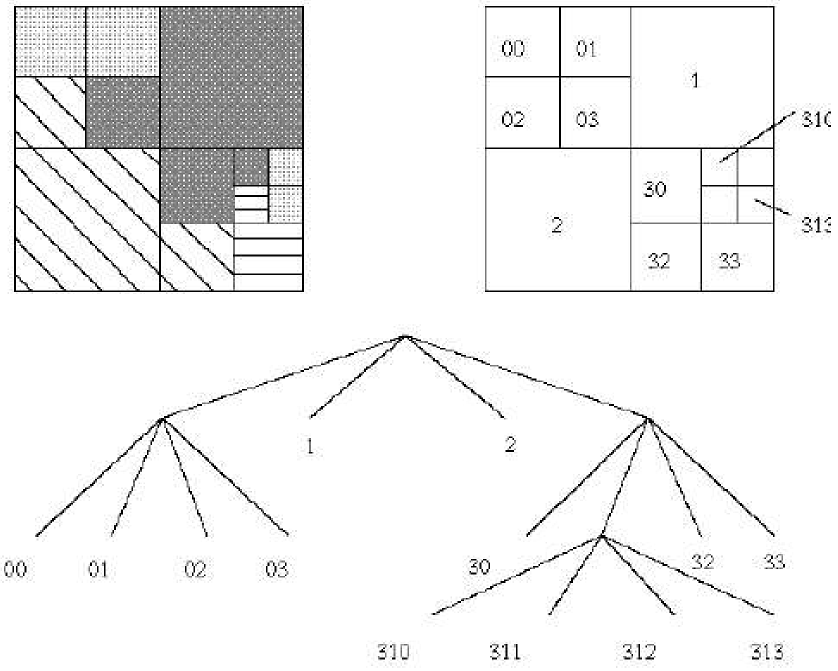
We can also describe the splitting of the image using a tree structure, called a modified *quadtree*. Each non-terminal node in the tree has at most four descendants, although it may have less due to merging.

Quadtree decomposition is an operation that subdivides an image into blocks that contain "similar" pixels. Usually the blocks are square, although sometimes they may be rectangular. For the purpose of this demo, pixels in a block are said to be "similar" if the range of pixel values in the block are not greater than some threshold.

Quadtree decomposition is used in variety of image analysis and compression applications.

An unpleasant drawback of segmentation quadtrees, is the square region shape assumption. It is not possible to merge regions which are not part of the same branch of the segmentation tree. Because both split-and-merge processing options are available, the starting segmentation does not have to satisfy any of the homogeneity conditions.

The segmentation process can be understood as the construction of a segmentation quadtree where each leaf node represents a homogeneous region. Splitting and merging corresponds to removing or building parts of the segmentation quadtree.

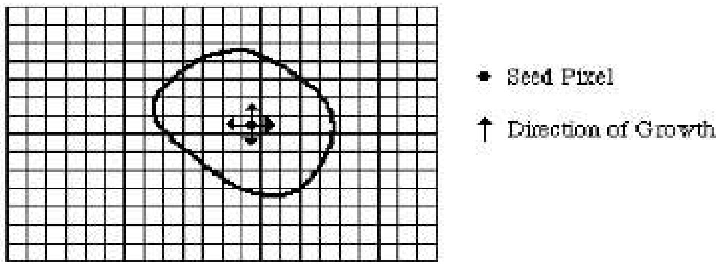


5.2.3.5 Region Growing

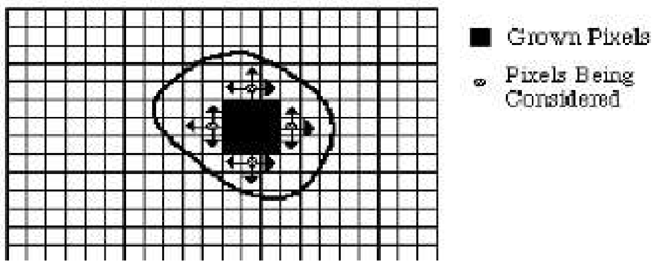
Region growing approach is the opposite of the split and merges approach:

- An initial set of small areas is iteratively merged according to similarity constraints.
- Start by choosing an arbitrary *seed pixel* and compare it with neighboring pixels
- Region is *grown* from the seed pixel by adding in neighboring pixels that are similar, increasing the size of the region.
- When the growth of one region stops we simply choose another seed pixel which does not yet belong to any region and start again.
- This whole process is continued until all pixels belong to some region.
- A *bottom up* method.

Region growing methods often give very good segmentations that correspond well to the observed edges.



(a) Start of Growing a Region



However starting with a particular seed pixel and letting this region grow completely before trying other seeds biases the segmentation in favour of the regions which are segmented first.

This can have several undesirable effects:

- Current region dominates the growth process -- ambiguities around edges of adjacent regions may not be resolved correctly.
- Different choices of seeds may give different segmentation results.
- Problems can occur if the (arbitrarily chosen) seed point lies on an edge.

To counter the above problems, *simultaneous region growing* techniques have been developed.

- Similarities of neighboring regions are taken into account in the growing process.
- No single region is allowed to completely dominate the proceedings.
- A number of regions are allowed to grow at the same time.
- Similar regions will gradually coalesce into expanding regions.
- Control of these methods may be quite complicated but efficient methods have been developed.
- Easy and efficient to implement on parallel computers.

5.2.4 **Edge and Line Detection**

5.2.4.1 Edge Detection

Edges are places in the image with strong intensity contrast. Since edges often occur at image locations representing object boundaries, edge detection is extensively used in image segmentation when we want to divide the image into areas corresponding to different objects. Representing an image by its edges has the further advantage that the amount of data is reduced significantly while retaining most of the image information.

Canny edge Detection

It is optimal for step edges corrupted by white noise. Optimality related to three criteria

- detection criterion ... important edges should not be missed, there should be no spurious responses
- localization criterion ... distance between the actual and located position of the edge should be minimal
- one response criterion ... minimizes multiple responses to a single edge (also partly covered by the first criterion since when there are two responses to a single edge one of them should be considered as false)

Canny's edge detector is based on several ideas:

1) The edge detector was expressed for a 1D signal and the first two optimality criteria. A closed form solution was found using the calculus of variations.

2) If the third criterion (multiple responses) is added, the best solution may be found by numerical optimization. The resulting filter can be approximated effectively with error less than 20% by the first derivative of a Gaussian smoothing filter with standard deviation σ; the reason for doing this is the existence of an effective implementation.

3) The detector is then generalized to two dimensions. A step edge is given by its position, orientation, and possibly magnitude (strength).

- Suppose G is a 2D Gaussian and assume we wish to convolute the image with an operator G_n which is a first derivative of G in the direction n.

$$G_n = \frac{\partial G}{\partial \mathbf{n}} = \mathbf{n} \cdot \nabla G$$

- The direction **n** should be oriented perpendicular to the edge
  - this direction is not known in advance
  - however, a robust estimate of it based on the smoothed gradient direction is available
  - if g is the image, the normal to the edge is estimated as

$$\mathbf{n} = \frac{\nabla(G * g)}{|\nabla(G * g)|}$$

- The edge location is then at the local maximum in the direction **n** of the operator G_n convoluted with the image g

$$\frac{\partial}{\partial \mathbf{n}} G_n * g = 0$$

- Substituting in equation for G_n from equation, we get

$$\frac{\partial^2}{\partial \mathbf{n}^2} G * g = 0$$

- This equation shows how to find local maxima in the direction perpendicular to the edge; this operation is often referred to as **non-maximum suppression**.

- As the convolution and derivative are associative operations in equation
  - first convolute an image g with a symmetric Gaussian G
  - then compute the directional second derivative using an estimate of the direction **n**
  - strength of the edge (magnitude of the gradient of the image intensity function g) is measured as

$$|G_n * g| = |\nabla(G * g)|$$

4) Spurious responses to the single edge caused by noise usually create a so called **'streaking'** problem that is very common in edge detection in general.

- Output of an edge detector is usually thresholded to decide which edges are significant.

- Streaking means breaking up of the edge contour caused by the operator fluctuating above and below the threshold.

- Streaking can be eliminated by **thresholding with hysteresis**.
    - If any edge response is above a **high threshold**, those pixels constitute definite output of the edge detector for a particular scale.
    - Individual weak responses usually correspond to noise, but if these points are connected to any of the pixels with strong responses they are more likely to be actual edges in the image.
    - Such connected pixels are treated as edge pixels if their response is above a **low threshold**.
    - The low and high thresholds are set according to an estimated signal to noise ratio.

5) The correct scale for the operator depends on the objects contained in the image.

- The solution to this unknown is to use multiple scales and aggregate information from them.

- Different scale for the Canny detector is represented by different standard deviations σ of the Gaussians.

- There may be several scales of operators that give significant responses to edges (i.e., signal to noise ratio above the threshold); in this case the operator with the smallest scale is chosen as it gives the best localization of the edge.

- Feature synthesis approach.
    - All significant edges from the operator with the smallest scale are marked first.
    - Edges of a hypothetical operator with larger σ are synthesized from them (i.e., a prediction is made of how the large σ should perform on the evidence gleaned from the smaller σ).
    - Then the synthesized edge response is compared with the actual edge response for larger σ.
    - Additional edges are marked only if they have significantly stronger response than that predicted from synthetic output.

- This procedure may be repeated for a sequence of scales, a cumulative edge map is built by adding those edges that were not identified at smaller scales.

Algorithm: Canny edge detector

1. Repeat steps (2) till (6) for ascending values of the standard deviation σ.
2. Convolve an image g with a Gaussian of scale σ.
3. Estimate local edge normal directions **n** for each pixel in the image.
4. Find the location of the edges (non-maximal suppression).
5. Compute the magnitude of the edge
6. Threshold edges in the image with hysteresis to eliminate spurious responses.

7. Aggregate the final information about edges at multiple scale using the `feature synthesis' approach.

5.2.4.2 **Edge Operators**

Since edges consist of mainly high frequencies, we can, in theory, detect edges by applying a high pass frequency filter in the Fourier domain or by convolving the image with an appropriate kernel in the spatial domain. In practice, edge detection is performed in the spatial domain, because it is computationally less expensive and often yields better results.

`Since edges correspond to strong illumination gradients, we can highlight them by calculating the derivatives of the image.We can see that the position of the edge can be estimated with the maximum of the 1st derivative or with the zero-crossing of the 2nd derivative. Therefore we want to find a technique to calculate the derivative of a two-dimensional image. For a discrete one-dimensional function *f(i)*, the first derivative can be approximated by

$$\frac{d\,f(i)}{d(i)} = f(i+1) - f(i)$$

Calculating this formula is equivalent to convolving the function with [-1 1]. Similarly the 2nd derivative can be estimated by convolving *f(i)* with [1 -2 1].

Different edge detection kernels which are based on the above formula enable us to calculate either the 1st or the 2nd derivative of a two-dimensional image. There are two common approaches to estimate the 1st derivative in a two-dimensional image, Prewitt compass edge detection and *gradient edge detection.*

Prewitt compass edge detection involves convolving the image with a set of (usually 8) kernels, each of which is sensitive to a different edge orientation. The kernel producing the maximum response at a pixel location determines the edge magnitude and orientation. Different sets of kernels might be used: examples include Prewitt, Sobel, Kirsch and Robinson kernels.

Gradient edge detection is the second and more widely used technique. Here, the image is convolved with only two kernels, one estimating the gradient in the *x*-direction, *Gx*, the other the gradient in the *y*-direction, *Gy*. The absolute gradient magnitude is then given by

$$|G| = \sqrt{Gx^2 + Gy^2}$$

and is often approximated with

$$|G| = |Gx| + |Gy|$$

In many implementations, the gradient magnitude is the only output of a gradient edge detector, however the edge orientation might be calculated with

The most common kernels used for the gradient edge detector are the Sobel, Roberts Cross and Prewitt operators.

After having calculated the magnitude of the 1st derivative, we now have to identify those pixels corresponding to an edge. The easiest way is to threshold the gradient image, assuming that all pixels having a local gradient above the threshold must represent an edge. An alternative technique is to look for local maxima in the gradient image, thus producing one pixel wide edges. A more sophisticated technique is used by the Canny edge detector. It first applies a gradient edge detector to the image and then finds the edge pixels using *non-maximal suppression* and *hysteresis tracking.*

An operator based on the 2nd derivative of an image is the Marr edge detector, also known as *zero crossing detector*. Here, the 2nd derivative is calculated using a Laplacian of Gaussian (LoG) filter. The Laplacian has the advantage that it is an isotropic measure of the 2nd derivative of an image, *i.e.* the edge magnitude is obtained independently from the edge orientation by convolving the image with only one kernel. The edge positions are then given by the zero-crossings in the LoG image. The scale of the edges which are to be detected can be controlled by changing the variance of the Gaussian.

A general problem for edge detection is its sensitivity to noise, the reason being that calculating the derivative in the spatial domain corresponds to accentuating high frequencies and hence magnifying noise. This problem is addressed in the Canny and Marr operators by convolving the image with a smoothing operator (Gaussian) before calculating the derivative.

5.2.5 **Line Detection**

While edges (I.E. boundaries between regions with relatively distinct graylevels) are by far the most common type of discontinuity in an image, instances of thin lines in an image occur frequently enough that it is useful to have a separate mechanism for detecting them. A convolution based technique can be used which produces an image description of the thin lines in an input image. Note that the Hough transform can be used to detect lines; however, in that case, the output is a PARAMETRIC description of the lines in an image.

The line detection operator consists of a convolution kernel tuned to detect the presence of lines of a particular width *n*, at a particular orientation ɵ. Figure below shows a collection of four such kernels, which each respond to lines of single pixel width at the particular orientation shown.

a)

-1	-1	-1
2	2	2
-1	-1	-1

b)

-1	2	-1
-1	2	-1
-1	2	-1

c)

-1	-1	2
-1	2	-1
2	-1	-1

d)

2	-1	-1
-1	2	-1
-1	-1	2

Four line detection kernels which respond maximally to horizontal, vertical and oblique (+45 and - 45 degree) single pixel wide lines.

These masks above are tuned for light lines against a dark background, and would give a big negative response to dark lines against a light background. If we are only interested in detecting dark lines against a light background, then we should negate the mask values. Alternatively, we might be interested in either kind of line, in which case, we could take the absolute value of the convolution output.

If R_i denotes the response of kernel I, we can apply each of these kernels across an image, and for any particular point, if $R_i > R_j$ for all $j \neq i$ that point is more likely to contain a line whose orientation (and width) corresponds to that of kernel I. One usually thresholds R_i to eliminate weak lines corresponding to edges and other features with intensity gradients which have a different scale than the desired line width. In order to find complete lines, one must join together line fragments, E.G., with an EDGE TRACKING operator.

5.2.6 **Corner Detection**

Input to the corner detector is the gray-level image. Output is an image in which values are proportional to the likelihood that the pixel is a corner. The Moravec detector is maximal in pixels with high contrast. These points are on corners and sharp edges.

$$MO(i,j) = \frac{1}{8} \sum_{k=i-1}^{i+1} \sum_{l=j-1}^{j+1} |g(k,l) - g(i,j)|$$

Using ZH Operator

The image function f is approximated in the neighborhood of the pixel (i,j) by a cubic polynomial with coefficients c_k. This is a cubic facet model. The ZH operator estimates the corner strength based on the coefficients of the cubic facet model.

$$g(i,j) = c_1 + c_2 x + c_3 y + c_4 x^2 + c_5 xy + c_6 y^2 + c_7 x^3 + c_8 x^2 y + c_9 x y^2 + c_{10} y^3$$

$$ZH(i,j) = \frac{-2\left(c_2^2 c_6 - c_2 c_3 c_5 - c_3^2 c_4\right)}{\left(c_2^2 + c_3^2\right)^{\frac{3}{2}}}$$