DIGITAL IMAGE PROCESSING

UNIT – III

Image Segmentation: Point and Line Detection – Basic Edge Detection Techniques Hough Transform – Thresholding - Global Thresholding - Optimal Thresholding using Otsu's Method - Multispectral Thresholding – Region Based Segmentation – Region Growing - Region Splitting and Merging.

➢ SEGMENTATION

- Image segmentation is a commonly used technique in digital image processing and analysis to partition an image into multiple parts or regions, often based on the characteristics of the pixels in the image.

- Image segmentation could involve separating foreground from background, or clustering regions of pixels based on similarities in color or shape.

- For example, a common application of image segmentation in medical imaging is to detect and label pixels in an image or voxels of a 3D volume that represent a tumor in a patient's brain or other organs.

➢ ***Region Based Segmentation***

The region based segmentation methods involve the algorithm creating segments by dividing the image into various components having similar characteristics. These components, simply put, are nothing but a set of pixels. Region-based image segmentation techniques initially search for some seed points – either smaller parts or considerably bigger chunks in the input image.

Next, certain approaches are employed, either to add more pixels to the seed points or further diminish or shrink the seed point to smaller segments and merge with other smaller seed points. Hence, there are two basic techniques based on this method.

3.1 Region Growing

It's a bottom to up method where we begin with a smaller set of pixel and start accumulating or iteratively merging it based on certain pre-determined similarity constraints. Region growth algorithm starts with choosing an arbitrary seed pixel in the image and compare it with its neighboring pixels.

If there is a match or similarity in neighboring pixels, then they are added to the initial seed pixel, thus increasing the size of the region. When we reach the saturation and hereby, the growth of that region cannot proceed further, the algorithm now chooses another seed pixel, which necessarily does not belong to any region(s) that currently exists and start the process again.

Region growing methods often achieve effective Segmentation that corresponds well to the observed edges. But sometimes, when the algorithm lets a region grow completely before trying other seeds, that usually biases the segmentation in favour of the regions which are segmented first. To counter this effect, most of the algorithms begin

with the user inputs of similarities first, no single region is allowed to dominate and grow completely and multiple regions are allowed to grow simultaneously.

Region growth, also a pixel based algorithm like thresholding but the major difference is thresholding extracts a large region based out of similar pixels, from anywhere in the image whereas region-growth extracts only the adjacent pixels. Region growing techniques are preferable for noisy images, where it is highly difficult to detect the edges.

the neighboring pixel is added to the current region. In other words, for every pixel p(i) in a region, there exists another pixel p(j) in the region such that

abs (I[p(i)]) - I[p(j)] < Threshold

This algorithm works well with multiband data such as for color images. If there are multiple input images (I1...In), the threshold criteria must hold for all input images

(abs (I1[p(i)] - I1[p(j)]) < Threshold)

&...&

(abs (In[p(i)] - In[p(j)]) < Threshold)

The concept can be defined as a four-connected pixel neighborhood, whereby only the north, south, east, and west neighbors are considered for growing the region.

It can also be defined as an eight-connected pixel neighborhood, in which the NW, NE, SW, and SE neighbors are additionally incorporated for growing the region.

The region-growing algorithm can also be made more adaptive to the data by modifying the threshold dynamically according to the mean and

standard deviations of the region as it is being grown. One method of making the threshold adapt to the local data is

(1 - minimum(limit, standard_deviation/mean)) *Threshold

3.2 Region Splitting and Merging

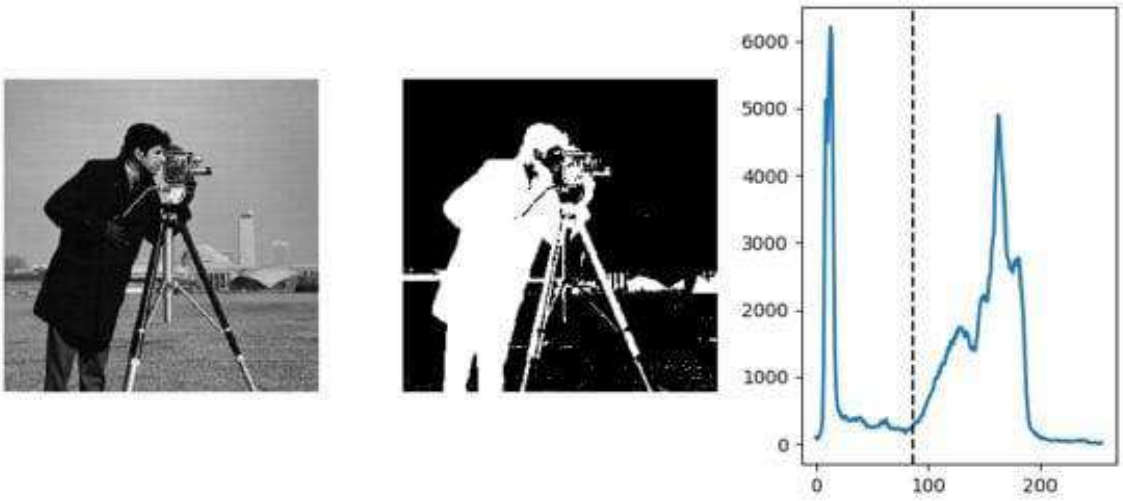
The splitting and merging based segmentation methods use two basic techniques done together in conjunction – region splitting and region merging – for segmenting an image. Splitting involves iteratively dividing an image into regions having similar characteristics and merging employs combining the adjacent regions that are somewhat similar to each other.

A region split, unlike the region growth, considers the entire input image as the area of business interest. Then, it would try matching a known set of parameters or pre-defined similarity constraints and picks up all the pixel areas matching the criteria. This is a divide and conquers method as opposed to the region growth algorithm.

Now, the above process is just one half of the process, after performing the split process, we will have many similarly marked regions scattered all across the image pixels, meaning, the final segmentation will contain scattered clusters of neighbouring regions that have identical or similar properties. To complete the process, we need to perform merging, which after each split which compares adjacent regions, and if required, based on similarity degrees, it merges them. Such algorithms are called split-merge algorithms.

Threshold Method

This is perhaps the most basic and yet powerful technique to identify the required objects in an image. Based on the intensity, the pixels in an image get divided by comparing the pixel's intensity with a threshold value. The threshold method proves to be advantageous when the objects in the image in question are assumed to be having more intensity than the background (and unwanted components) of the image.



At its simpler level, the threshold value T is considered to be a constant. But that approach may be futile considering the amount of noise (unwanted information) that the image contains. So, we can either keep it constant or change it dynamically based on the image properties and thus obtain better results. Based on that, thresholding is of the following types:

Simple Thresholding

This technique replaces the pixels in an image with either black or white. If the intensity of a pixel ($I_{i,j}$) at position (i,j) is less than the threshold (T), then we replace that with black and if it is more, then we replace that pixel with white. This is a binary approach to thresholding.

Otsu's Binarization

In global thresholding, we had used an arbitrary value for threshold value and it remains a constant. The major question here is, how can we define and determine the correctness of the selected threshold? A simpler but rather inept method is to trial and see the error.

But, on the contrary, let us take an image whose histogram has two peaks (bimodal image), one for the background and one for the foreground. According to Otsu binarization, for that image, we can approximately take a value in the middle of those peaks as the threshold value. So in simply put, it automatically calculates a threshold value from image histogram for a bimodal image.

The disadvantage here, however, is for images that are not bimodal, the image histogram has multiple peaks, or one of the classes (peaks) present has high variance.

However, Otsu's Binarization is widely used in document scans, removing unwanted colors from a document, pattern recognition etc.

Optimum Global Thresholding using Otsu's Method

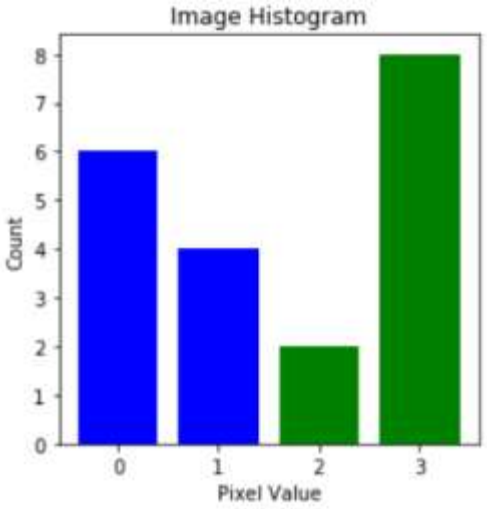
we will discuss **Otsu's method**, named after Nobuyuki Otsu, that automatically finds the global threshold.

Note: This method assumes that the image histogram is bimodal and a reasonable contrast ratio exists between the background and the region of interest.

In simple terms, Otsu's method tries to find a threshold value which minimizes the **weighted within-class variance**. Since Variance is the spread of the distribution about the mean. Thus, minimizing the within-class variance will tend to make the classes compact.

Let's say we threshold a histogram at a value "t". This produces two regions – left and right of "t" whose variance is given by σ^2_0 and σ^2_1. Then the weighted within-class variance is given by

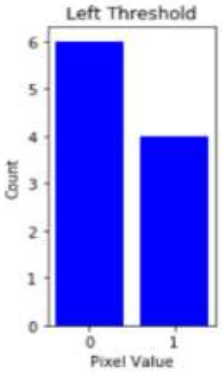
$$\sigma_w^2(t) = \omega_0(t)\sigma_0^2(t) + \omega_1(t)\sigma_1^2(t)$$



where $w_0(t)$ and $w_1(t)$ are the weights given to each class. Weights are total pixels in a thresholded region (left or right) divided by the total image pixels. Let's take a simple example to understand how to calculate these.

Suppose we have the following histogram and we want to find the weighted within-class variance corresponding to threshold value 1.

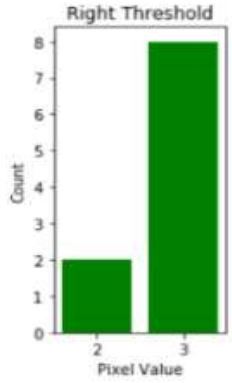
Below are the weights and the variances calculated for left and the right regions obtained after thresholding at value 1.



$$w_0 = \frac{\textit{Total pixels in left region}}{\textit{Total pixels in the image}} = \frac{6+4}{20} = 0.5$$

$$\mu_0 = \frac{\textit{weighted sum of intensities}}{\textit{Total pixels in left region}} = \frac{(0\times 6) + (1\times 4)}{10} = 0.4$$

$$\sigma_0^2 = \frac{((0-0.4)^2\times 6) + ((1-0.4)^2\times 4)}{10} = 0.24$$



$$w_1 = \frac{\textit{Total pixels in right region}}{\textit{Total pixels in the image}} = \frac{2+8}{20} = 0.5$$

$$\mu_1 = \frac{\textit{weighted sum of intensities}}{\textit{Total pixels in right region}} = \frac{(2\times 2) + (3\times 8)}{10} = 2.8$$

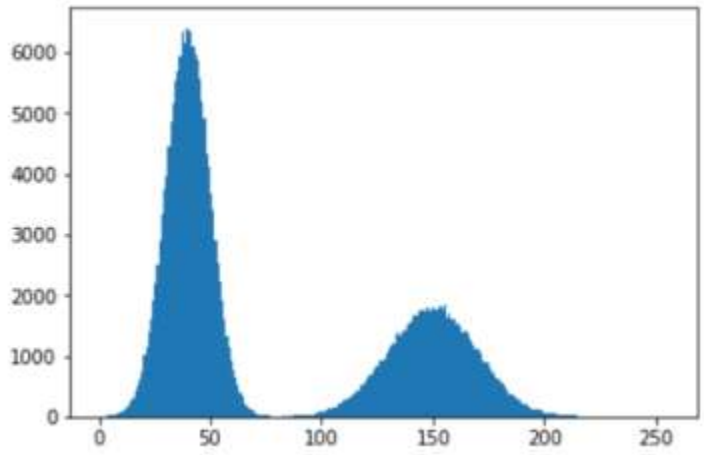
$$\sigma_1^2 = \frac{((2-2.8)^2\times 2) + ((3-2.8)^2\times 8)}{10} = 0.16$$

$$\textit{Weighted within} - \textit{class variance is}:$$

$$\sigma_w^2 = w_0\sigma_0^2 + w_1\sigma_1^2 = 0.5\times 0.24 + 0.5\times 0.16 = 0.2$$

Similarly, we will iterate over all the possible threshold values, calculate the weighted within-class variance for each of the thresholds. The optimum threshold will be the one with the minimum within-class variance.

The image histogram is shown below



A Faster Approach

We all know that minimizing within-class variance is equivalent to maximizing between-class variance. This maximization operation can be implemented recursively and is faster than the earlier method. The expression for between-class variance is given by

$$\sigma_B^2 = w_0 w_1 (\mu_0 - \mu_1)^2$$

Below are the steps to calculate recursively between-class variance.

1. Calculate the histogram of the image.
2. Set up weights and means corresponding to the “0” threshold value.
3. Loop through all the threshold values
    1. Update the weights and the mean
    2. Calculate the between-class variance
4. The optimum threshold will be the one with the max variance..

Limitations

Otsu’s method is only guaranteed to work when

- The histogram should be bimodal.
- Reasonable contrast ratio exists between the background and the roi.
- Uniform lighting conditions are there.
- Image is not affected by noise.
- Size of the background and the roi should be comparable.

There are many modifications done to the original Otsu's algorithm to address these limitations such as **two-dimensional Otsu's method** etc. We will discuss some of these modifications in the following blogs.

➢ Adaptive Thresholding

A global value as threshold value may not be good in all the conditions where an image has different background and foreground lighting conditions in different actionable areas. We need an adaptive approach that can change the threshold for various components of the image. In this, the algorithm divides the image into various smaller portions and calculates the threshold for those portions of the image.

Hence, we obtain different thresholds for different regions of the same image. This in turn gives us better results for images with varying illumination. The algorithm can automatically calculate the threshold value. The threshold value can be the mean of neighborhood area or it can be the weighted sum of neighborhood values where weights are a Gaussian window (a window function to define regions).

2. Edge Based Segmentation

Edge detection is the process of locating edges in an image which is a very important step towards understanding image features. It is believed that edges consist of meaningful features and contains significant information. It significantly reduces the size of the image that will be processed and filters out information that may be regarded as less relevant, preserving and focusing solely on the important structural properties of an image for a business problem.

Edge-based segmentation algorithms work to detect edges in an image, based on various discontinuities in grey level, colour, texture, brightness, saturation, contrast etc. To further enhance the results, supplementary processing steps must follow to concatenate all the edges into edge chains that correspond better with borders in the image.

Edge detection algorithms fall primarily into two categories – Gradient based methods and Gray Histograms. Basic edge detection operators like sobel operator, canny, Robert's variable etc are used in these algorithms. These operators aid in detecting the edge discontinuities and hence mark the edge boundaries. The end goal is to reach at least a partial segmentation using this process, where we group all the local edges into a new binary image where only edge chains that match the required existing objects or image parts are present.

➢ ***Multispectral Thresholding***

Section of your text describes a technique for segmenting images with multiple components (color images,Landsat images, or MRI images with T1, T2, and proton-density bands). It works by estimating the optimal threshold in one channel and then segmenting the overall image based on that threshold. We then subdivide each of these regions independently using properties of the second channel. We repeat it again for the third channel, and so on, running through all channels repeatedly until each region in the image exhibits a distribution indicative of a coherent region (a single mode).

➢ ***Shape-based segmentation***

Occasionally, objects to be segmented from an image are confined to some well-defined shape. Such is the case for fibers, spheres, and rectangles. It is useful in this case to again visualize the image surface as a topographical map, where the objects have a three-dimensional shape. Fibers can be ridges or valleys, spheres can be hills, and so forth. If an example of the shape, called a template, is created, it can be "slid under" the topographical surface. Where it "fits," it will then slide up into the shape.

The template can also be slid under the surface of the entire image to create an output image that contains gray-scale values corresponding to how high the shape can go. Where the shape does not fit, then large contiguous areas of high or low values will appear in the output image. Where the shape does fit, small local "peaks" will emerge. These peaks can be detected with local maximal operators and then used to mark the existence of desired shapes in the image, thereby "segmenting out" the desired objects. This technique can be implemented with morphological gray-scale erosion and dilation techniques (see Fig. 4).

➢ PETER EGGLESTON

Click here to enlarge image

FIGURE 1. A multispectral, seeded, region-growing algorithm created this masking plug-in for Adobe Photoshop. A user selects sample background (red) and foreground (green) seeds. Color information derived from the seeds is then used to grow a region corresponding to the object of interest--the sunflower. Once a mask is derived, the extracted image portion can then be color corrected, used in the creation of a new image, or have special effects applied.

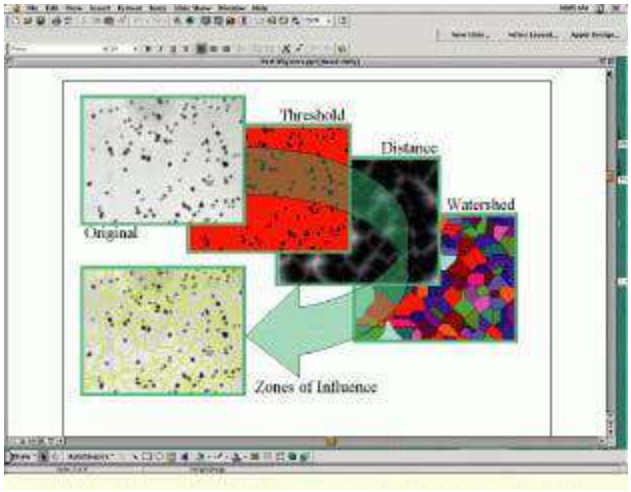


FIGURE 2. The watershed segmentation operator can measure object dispersion using the Skeletonization by Influence Zones (SKIZ) approach. In this example, an inorganic pigment in a polymer material is imaged as black dots (pigment) on a gray background (polymer). First, the background is extracted by thresholding the original image. Next, a distance image is created by setting every background pixel value to the shortest distance to the nearest pigment. This distance image is then thresholded using the watershed technique, creating equi-distant zones about each pigment particle.

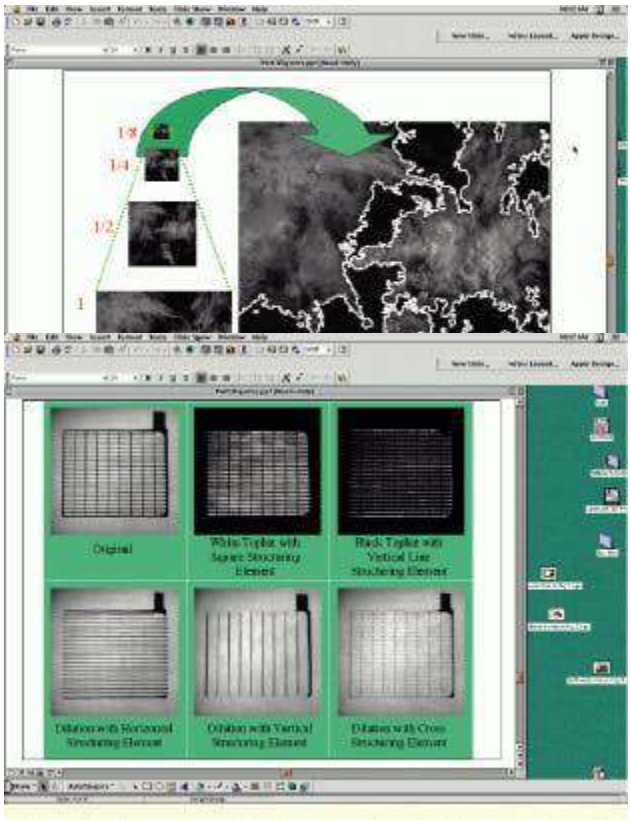


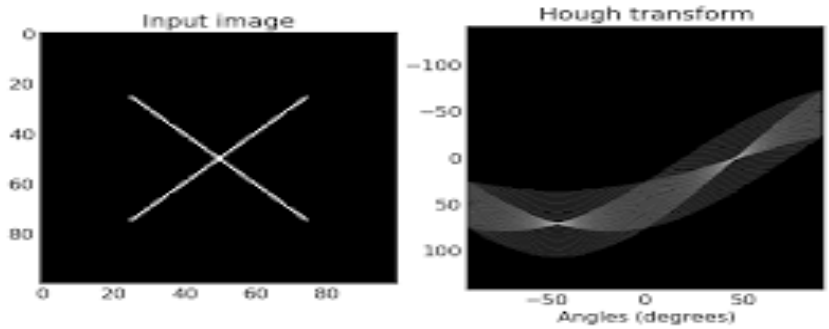
FIGURE 3. Image pyramids and multiresolution processing techniques are useful in the segmentation of wispy objects, such as the clouds in this satellite photograph. These techniques can be used to avoid the oversegmentation of images that have a good deal of variation in the areas of interest.

FIGURE 4. Morphological templates support the extraction of specific shapes. In a top-hat segmentation, templates are slid over or under the image to mark areas of interest. Dilation and erosion can remove dark or light areas respectively, which do not correspond to shapes of interest.

❖ ***Basic Edge Detection Techniques Hough Transform***

❖*Hough Transform*

A comprehensive guide to edge detection with Hough transform with code

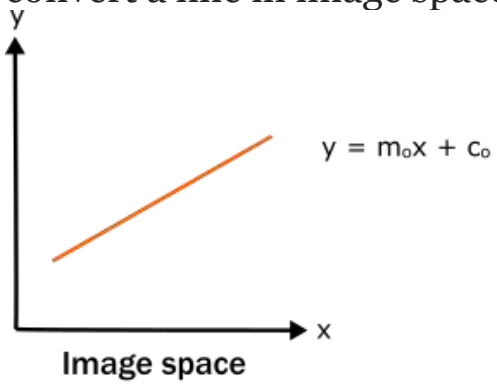


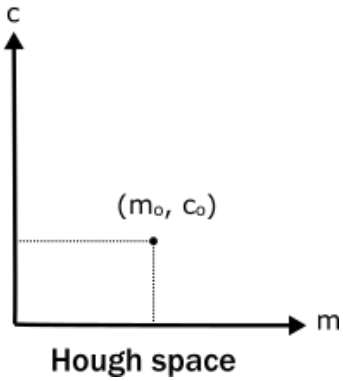
Hough transform is a feature extraction method used in image analysis. Hough transform can be used to isolate features of any regular curve like lines, circles, ellipses, etc. Hough transform in its simplest from can be used to detect straight lines in an image. A generalized Hough transform can be used in applications where simple analytic description of features is not possible. Due to the computational complexity of the algorithm, people generally refrain from using it. Moreover learning based methods can extract complex features from the image or a video that better suit the problem we are trying to solve.

➢ ***Algorithm***

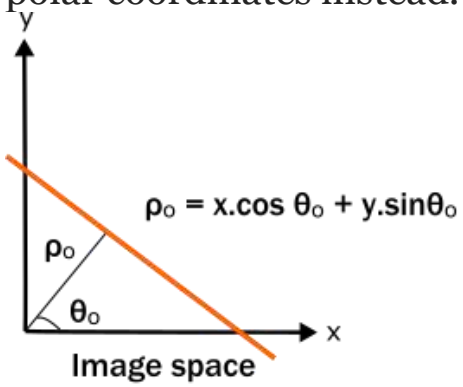
A straight line is the simplest boundary we can recognize in an image. Multiple straight lines can form a much complex boundary.

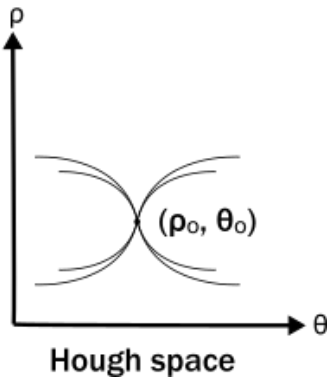
We transform the image space into hough space. By doing this we convert a line in image space to a point on hough space.





The equation of the line in the image space is of the form y = mx + c where m is the slope and c is the y-intercept of the line. This line will be transformed to a point of the form (m, c) in the hough space. But in this representation m goes to infinity for vertical lines. So let us use the polar coordinates instead.



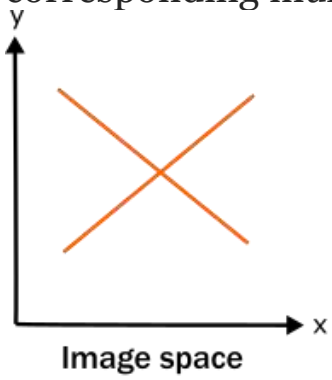


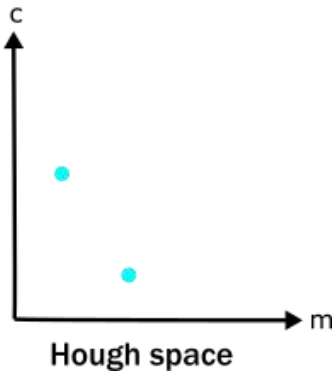
The line is represented by the length of that segment ρ , and the angle θ it makes with the x-axis. This line will be transformed to a point of the form (ρ,θ) in the hough space.

The Hough transform constructs a histogram array representing the parameter space (i.e., an M x N matrix, for M different values of the radius ρ and N different values of angle θ). For each parameter combination,ρ and θ we then find the number of non-zero pixels in the input image that would fall close to the corresponding line, and increment the array at position (ρ,θ) appropriately.

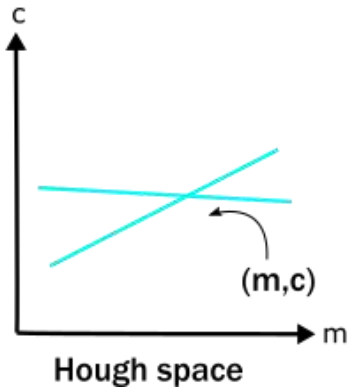
➢ Intuition for line detection

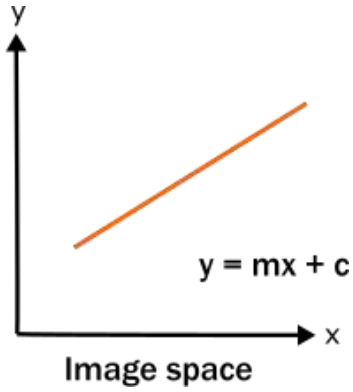
The intersection of multiple lines in image space represent corresponding multiple points in hough space.





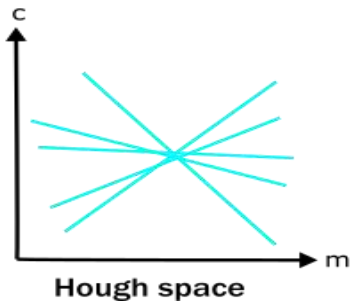
Similarly the reverse i.e lines intersecting at a point (m, c) in hough space can be transformed to a line y = mx + c in image space.

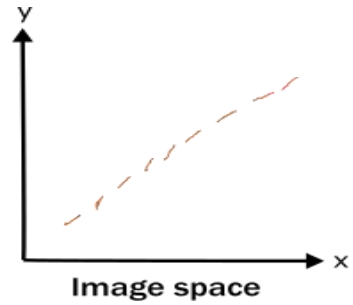




If we have a line made up of many segments or points close to the same line equation in the image space, that turns into many intersecting lines in hough space.

So, consider a line in the image space which is an edge detected and has small discontinuities. To find the continous line in an image we can transform this dicontinous line in image space to hough space and look for intersection points in hough space. This intersection point in hough space will represent the continous line in image space.





Point Detection

This is used to detect isolated spots in an image.

The graylevel of an isolated point will be very different from its neighbors.

It can be accomplished using the following 3 3 mask:

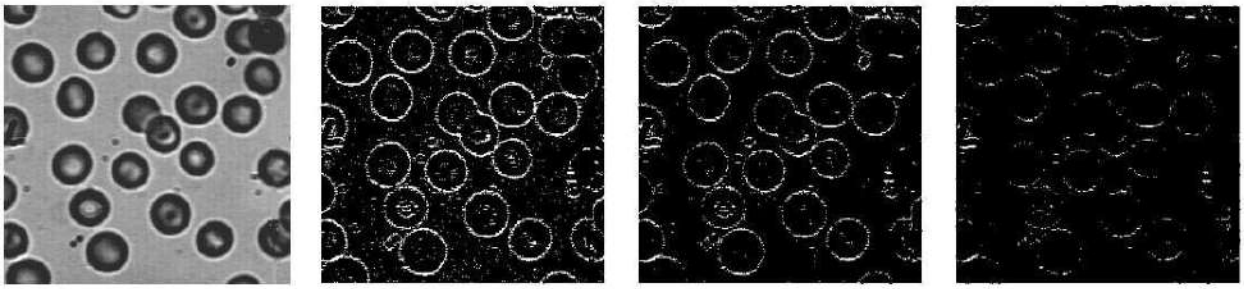
$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 8 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

The output of the mask operation is usually thresholded.

We say that an isolated point has been detected if

$$\left| 8f_5 \quad f_1 \quad f_2 \quad f_3 \quad f_4 \quad f_6 \quad f_7 \quad f_8 \quad f_9 \right| \quad T$$

for some pre-specified non-negative threshold T.



Orig. Image T 0.25 T 0.35 T 0.5

Detection of lines

This is used to detect lines in an image.

It can be done using the following four masks:

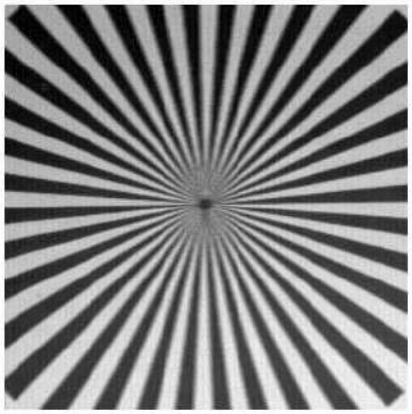
$$\mathcal{D}_{0^\circ} \begin{bmatrix} 1 & 1 & 1 \\ 2 & 2 & 2 \\ 1 & 1 & 1 \end{bmatrix}$$ Detects horizontal lines

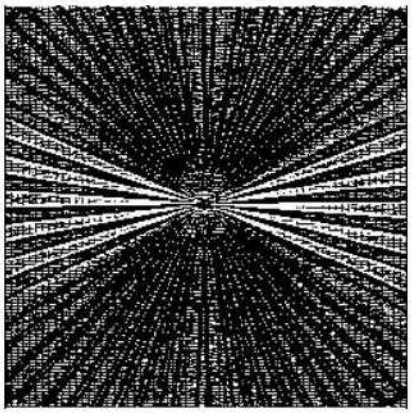
$$\mathcal{D}_{45^\circ} \begin{bmatrix} 1 & 1 & 2 \\ 1 & 2 & 1 \\ 2 & 1 & 1 \end{bmatrix}$$ Detects 45° lines

$$\mathcal{D}_{90^\circ} \begin{bmatrix} 1 & 2 & 1 \\ 1 & 2 & 1 \\ 1 & 2 & 1 \end{bmatrix}$$ Detects vertical lines

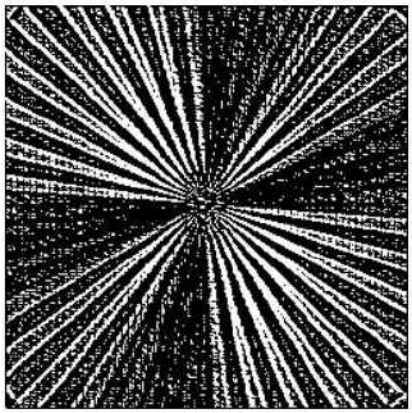
$$\mathcal{D}_{135^\circ} \begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix}$$ Detects 135° lines

Let R_{0°, R_{45°, R_{90°, and R_{135°, respectively be the response to masks $\mathcal{D}_{0^\circ}$, $\mathcal{D}_{45^\circ}$, $\mathcal{D}_{90^\circ}$, and $\mathcal{D}_{135^\circ}$, respectively. At a given pixel (m,n), if R_{135° is the maximum among $\{R_{0^\circ}, R_{45^\circ}, R_{90^\circ}, R_{135^\circ}\}$, we say that a 135° line is most likely passing through that pixel.

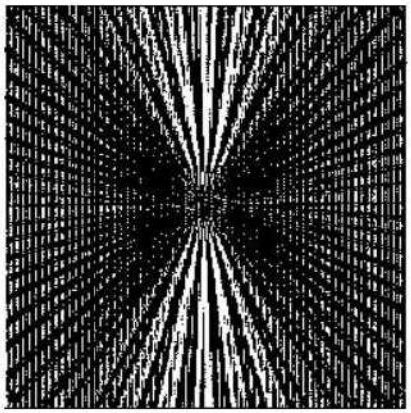




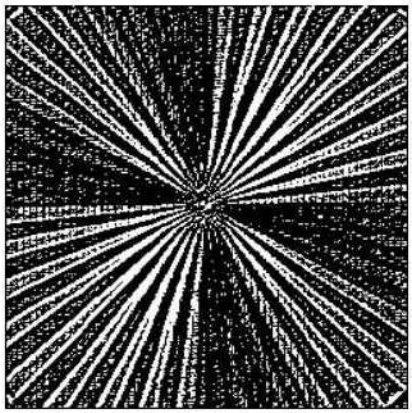
R_{0° max $R_{0^\circ}, R_{45^\circ}, R_{90^\circ}, R_{135^\circ}$



R_{45° max $R_{0^\circ}, R_{45^\circ}, R_{90^\circ}, R_{135^\circ}$



R_{90° max $R_{0^\circ}, R_{45^\circ}, R_{90^\circ}, R_{135^\circ}$



R_{135° max $R_{0^\circ}, R_{45^\circ}, R_{90^\circ}, R_{135^\circ}$

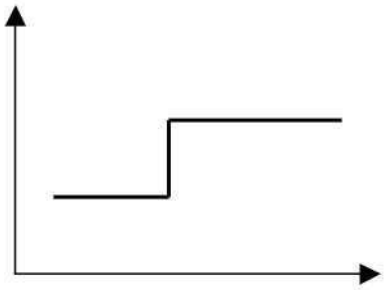
Edge Detection

Isolated points and thin lines do not occur frequently in most practical applications.

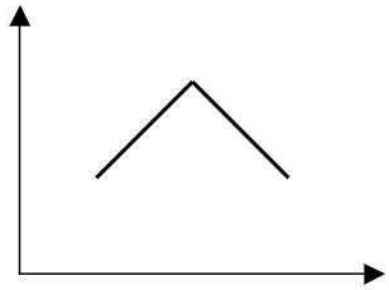
For image segmentation, we are mostly interested in detecting the boundary between two regions with relatively distinct gray-level properties.

We assume that the regions in question are sufficiently homogeneous so that the transition between two regions can be determined on the basis of gray-level discontinuities alone.

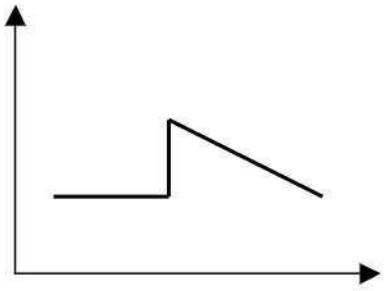
An edge in an image may be defined as a discontinuity or abrupt change in gray level.



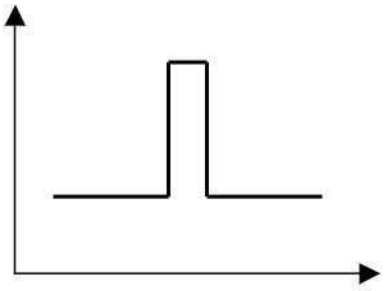
Ideal Step Edge



Ideal Roof Edge



Combination of Step and Roof Edges

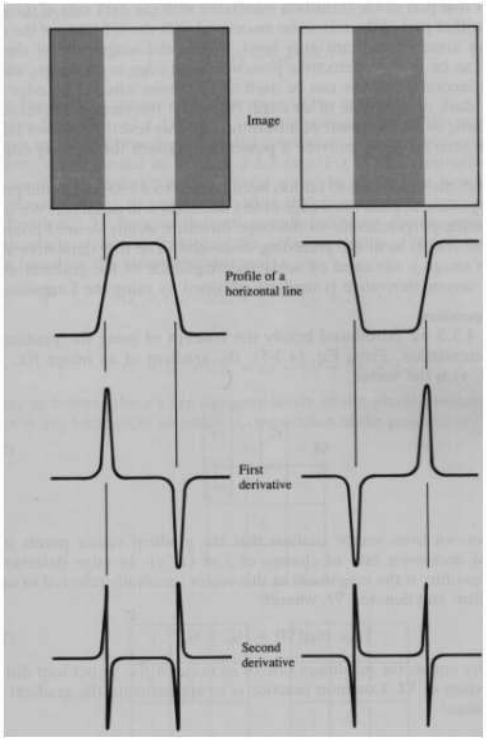


Ideal Spike Edge

These are ideal situations that do not frequently occur in practice. Also, in two dimensions edges may occur at any orientation.

Edges may not be represented by perfect discontinuities. Therefore, the task of edge detection is much more difficult than what it looks like.

A useful mathematical tool for developing edge detectors is the first and second **derivative operator**s.



From the example above it is clear that the **magnitude** of the **first derivative** can be used to detect the presence of an edge in an image.

The **sign** of the **second derivative** can be used to determine whether an edge pixel lies on the dark or light side of an edge.

The **zero crossings** of the **second derivative** provide a powerful way of locating edges in an image.

We would like to have small-sized masks in order to detect fine variation in graylevel distribution (i.e., micro-edges).

On the other hand, we would like to employ large-sized masks in order to detect coarse variation in graylevel distribution (i.e., macro-edges) and filter-out noise and other irregularities.

We therefore need to find a mask size, which is a compromise between these two opposing requirements, or determine edge content by using different mask sizes.

Most common differentiation operator is the gradient.

$$f(x,y) \quad \begin{bmatrix} \frac{f(x,y)}{x} \\ \frac{f(x,y)}{y} \end{bmatrix}$$

The magnitude of the gradient is:

$$| f(x,y)| \quad \left[\left(\frac{f(x,y)}{x} \right)^2 \quad \left(\frac{f(x,y)}{y} \right)^2 \right]^{1/2}$$

The direction of the gradient is given by :

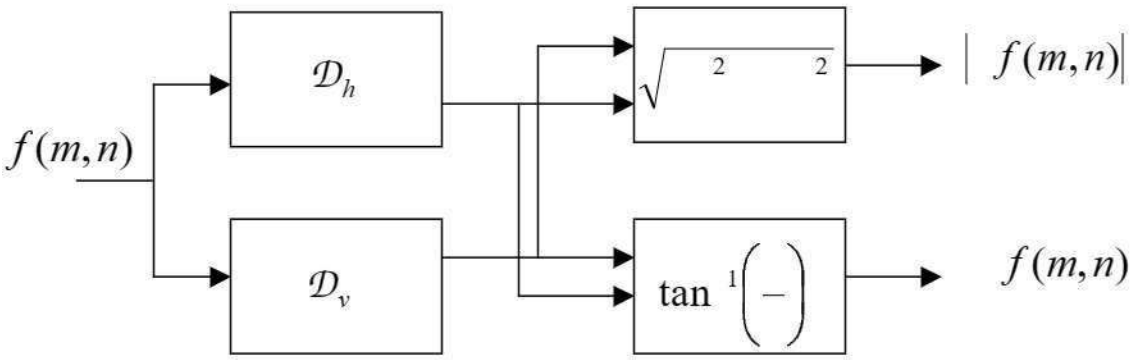
$$f(x,y) \quad \tan^{1} \left\{ \frac{\frac{f}{y}}{\frac{f}{x}} \right\}$$

In practice, we use discrete approximations of the partial derivatives $\frac{f}{x}$ and $\frac{f}{y}$, which are implemented using the masks:

$$\mathcal{D}_h \quad \frac{1}{2} \quad 1 \;\; 1 \;\; 0 \quad 0 \quad 1 \;\; 1 \quad \frac{1}{2} \quad 1 \;\; 0 \;\; 1$$

$$\mathcal{D}_v \quad \frac{1}{2}\left\{\begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} \quad \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}\right\} \quad \frac{1}{2}\begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$$

The gradient can then be computed as follows:



Other discrete approximations to the gradient (more precisely, the appropriate partial derivatives) have been proposed (Roberts, Prewitt).

Because derivatives enhance noise, the previous operators may not give good results if the input image is very noisy.

One way to combat the effect of noise is by applying a smoothing mask. The Sobel edge detector combines this smoothing operation along with the derivative operation give the following masks:

Since the gradient edge detection methodology depends only on the relative magnitudes within an image, scalar multiplication by

factors such as 1/2 or 1/8 play no essential role. The same is true for the signs of the mask entries. Therefore, masks like

$$\frac{1}{8}\begin{bmatrix} 1 & 0 & 1 \\ 2 & 0 & 2 \\ 1 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} 1 & 0 & 1 \\ 2 & 0 & 2 \\ 1 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} 1 & 0 & 1 \\ 2 & 0 & 2 \\ 1 & 0 & 1 \end{bmatrix}$$

correspond to the same detector, namely the Sobel edge detector.

However when the exact magnitude is important, the proper scalar multiplication factor should be used.

All masks considered so far have entries that add up to zero. This is typical of any derivative mask.

Example