Image Analysis (LEC 5)

5-1 Image Analysis :

Image analysis involves manipulating the image data to determine exactly the information necessary to help solve a computer imaging problem. Image analysis is primarily data reduction process. The primary part of the image analysis task is to determine exactly what information is necessary. Image analysis is used both computer vision and image processing.

For computer vision: the extraction of high-level information for computer analysis or manipulation. This high-level information may include shape parameter to control a robotics manipulator or color and texture features to help in diagnosis of a skin tumor.

In image processing application: determining the type of processing required and the specific parameters needed for that processing. For example, determine the degradation function or an image restoration procedure, developing an enhancement algorithm and determining exactly what information is visually important for image compression methods.

5-2 System Model :

The image analysis process can be broken down into three primary stages:

1. Preprocessing.

2. Data Reduction.

3. Features Analysis.

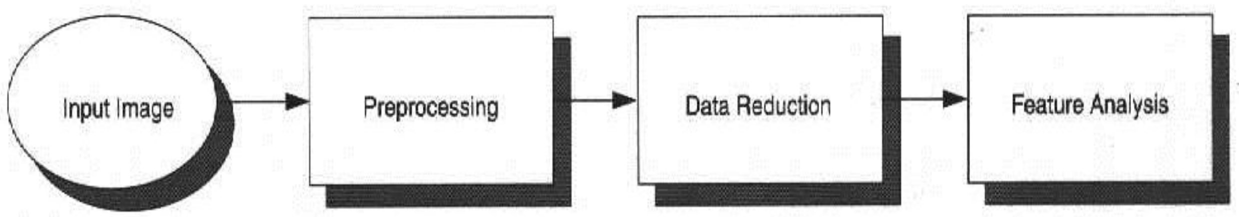


Figure (5-1) Image Analysis Model

1. Preprocessing

Is used to remove noise and eliminate irrelevant information, visually unnecessary information. Noise is unwanted information that can result from the image acquisition process.

The preprocessing algorithm, techniques and operators are used to perform initial processing that makes the primary data reduction and analysis task easier. They include operations related to:

• Extracting regions of interest.

• Performing basic algebraic operation on image.

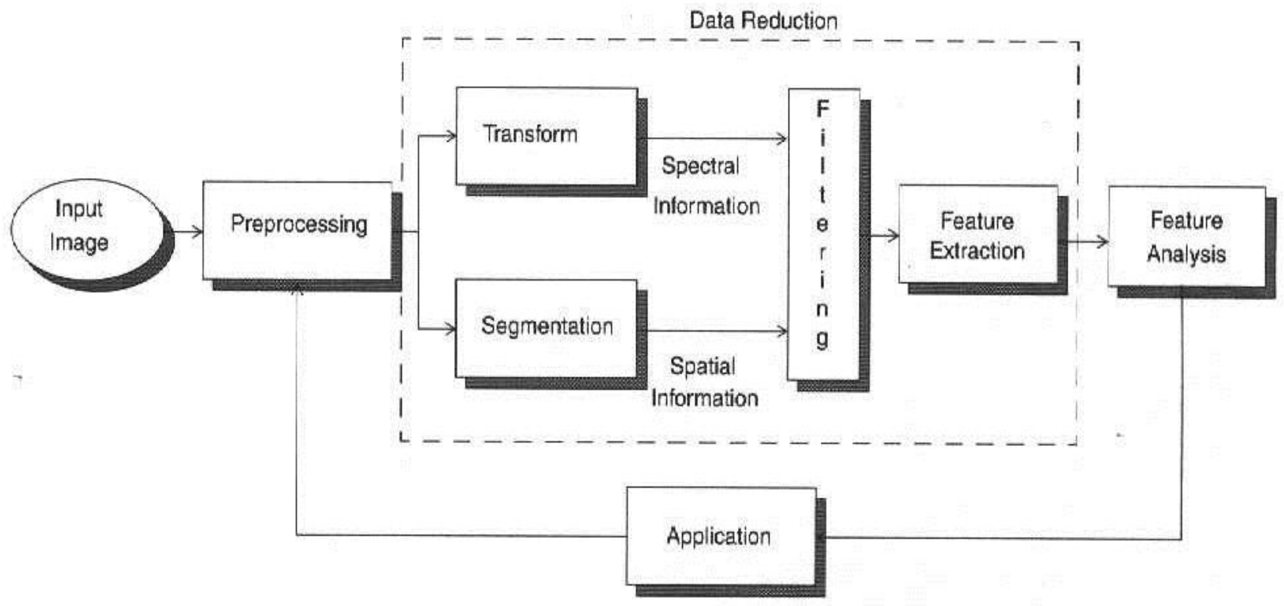
• Enhancing specific image features.

• Gray –level or spatial quantization (reducing the number of bits per pixel or the image size).

For example of preprocessing step involves a robotics gripper that needs to pick and place an object ; for this we reduce a gray-level image to binary (two-valued) image that contains all the information necessary to discern the object's outlines.

2. Data Reduction

Involves either reducing the data in the spatial domain or transforming it into another domain called the frequency domain, and then extraction features for the analysis process.



3. Features Analysis:

The features extracted by the data reduction process are examined and evaluated for their use in the application.

After preprocessing we can perform segmentation on the image in the spatial domain or convert it into the frequency domain via a mathematical transform. After these processes we may choose to filter the image. This filtering process further reduces the data and allows us to extract the feature that we may require for analysis.

5-3 Region of Interest Image Geometry (ZOOMING):

Often, for image analysis we want to investigate more closely a specific area within the image, called region of interest (ROI). To do this we need operation that *modifies the spatial coordinates* of the image, and these are categorized as image geometry operations. The image geometry operations discussed here include: crop, zoom, enlarge, shrink, translate and rotate.

The image crop process is the process of selecting a small portion of the image, a sub image and cutting it away from the rest of the image. After we have cropped a sub image from the original image we can zoom in on it by enlarge it that show in figure (5-3). The zoom process can be done in numerous ways:

1. Zero-Order Hold.
2. First-Order Hold.
3. Convolution.

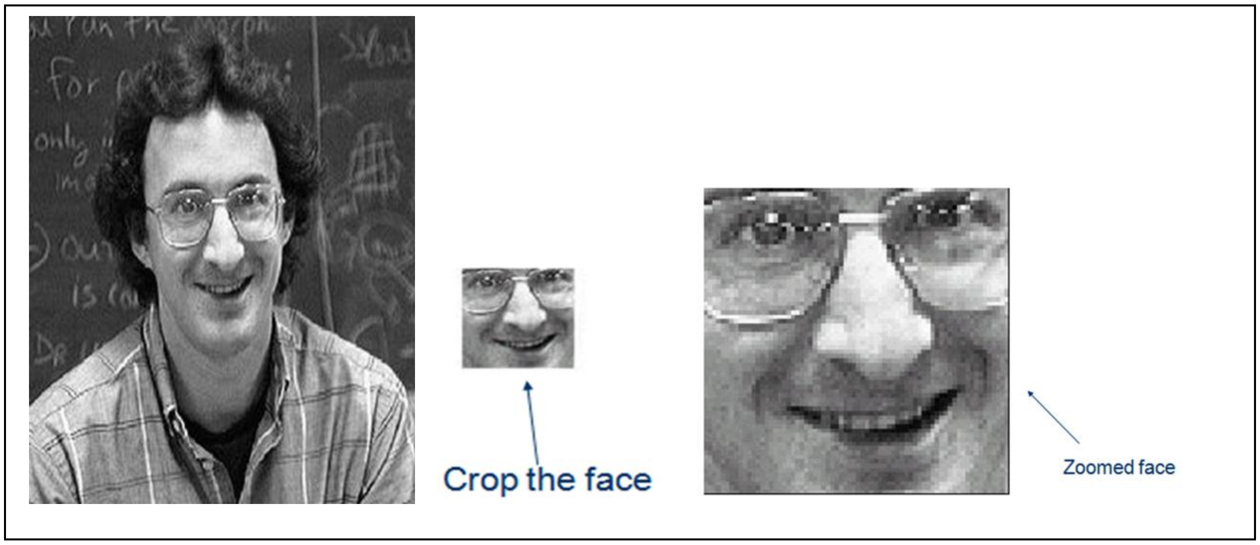


Figure (5-3) zoomed face

1. **Zero-Order hold:** is performed by repeating previous pixel values, thus creating a blocky effect as in the following figure(5-4):

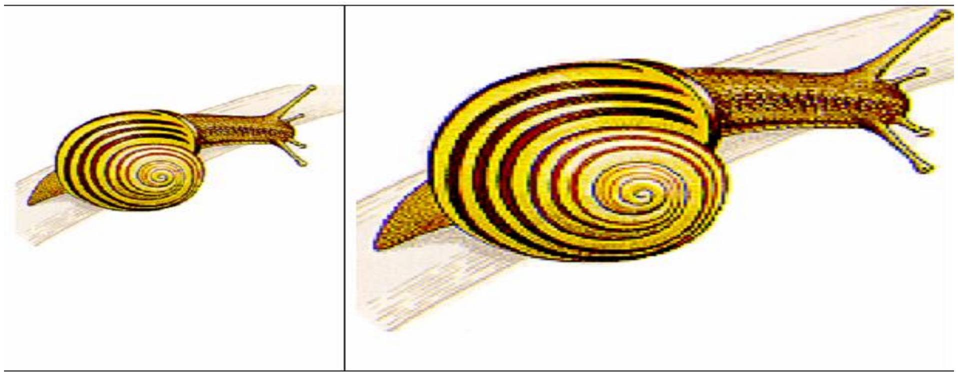


Figure (5.4): Zero _Order Hold Method

Example:

8	4	8
4	8	4
8	4	8

Original Image

8	4	8
8	4	8
4	8	4
4	8	4
8	2	8
8	2	8

Intermediate Stage

8	8	4	4	8	8
8	8	4	4	8	8
4	4	8	8	4	4
4	4	8	8	4	4
8	8	2	2	8	8
8	8	2	2	8	8

Result Image

2. **First-Order Hold:** is performed by finding linear interpolation between adjacent pixels, i.e., finding the average value between two pixels and use that as the pixel value between those two.

Figure (4.5): First _Order Hold Method

by using Interpolation:

RULE 1

Result= (End point - Start Point) / (No. of added pixel +1)

RULE 2

Intermediate Pixel Value = Result + Old Pixel

Example (1): Given two pixels and we want to add 1 pixel inside?

10	20

Original Image Array

Result=(20-10)/(1+1)=5
Intermediate Pixel Value=5+10=15

10	15	20

Result Image

Example (2): Given the image array?

8	4	8
4	8	4
8	4	8

Original Image Array

Stage -1- Row Expanded:

1- 4-8/2=-2
2- 8-4/2=2
3- 2-8/2=-3
4- 8-4/2=2
5- 4-8/2=-2

6- 8-2/2=2

Image with Rows Expanded

8	6	4	6	8
4	6	8	6	4
8	5	2	5	8

Stage -2- Row Expanded:

1- 4-8/2=-2	**6-** 8-4/2=2
2- 6-6/2=0	**7-** 5-6/2=-0.5=-1
3- 8-4/2=2	**8-** 2-8/2=-3
4- 6-6/2=0	**9-** 5-6/2=-0.5=-1
5- 4-8/2=-2	**10-** 8-4/2=2

The first two pixels in the first row are averaged (8+4)/2=6, and this number is inserted between those two pixels. This is done for every pixel pair in each row. Next, take result and expanded the columns in the same way as follows:

Image with rows and columns expanded

8	6	4	6	8
6	6	6	6	6
4	6	8	6	4
6	5.5	5	5.5	6
8	5	2	5	8

3- Convolution: This process requires a mathematical process to enlarge an image. This method required two steps:

1. Extend the image by adding rows and columns of zeros between the existing rows and columns.

Original Image Array

$$\begin{pmatrix} 3 & 5 & 7 \\ 2 & 7 & 6 \\ 3 & 4 & 9 \end{pmatrix}$$

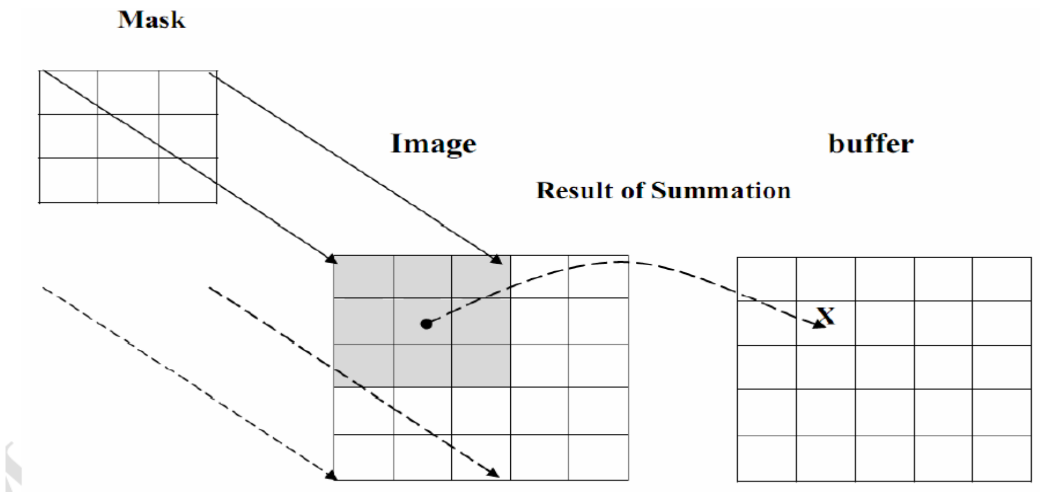
Image extended with zeros

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 5 & 0 & 7 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 7 & 0 & 6 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 4 & 0 & 9 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

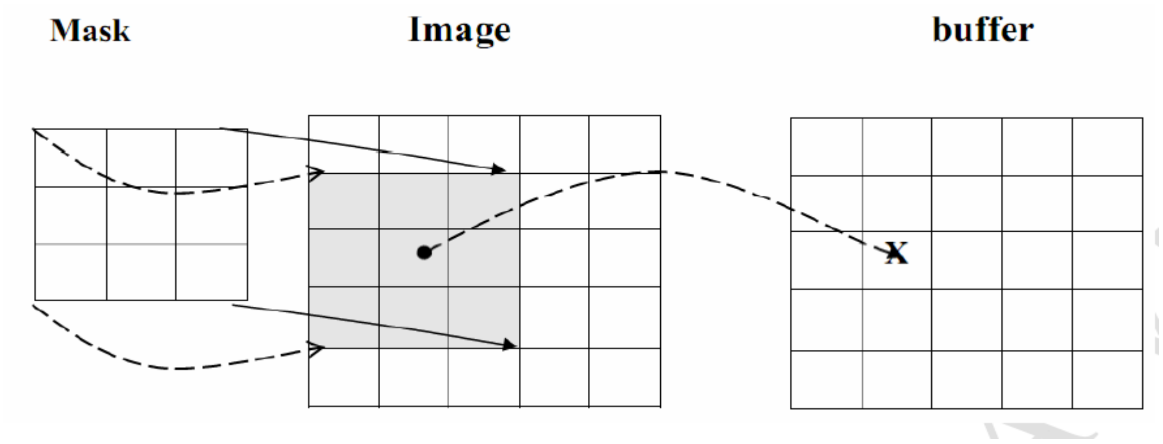
2. Perform the convolution.

The convolution process is shown in the next stages:

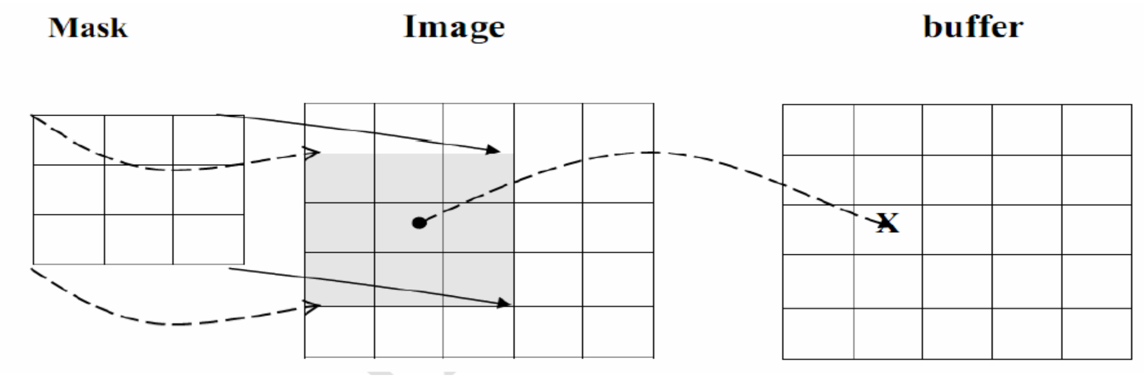
a. Overlay the convolution mask in the upper-left corner of the image. Multiply coincident terms, sum, and put the result into the image buffer at the location that corresponds to the masks current center, which is (r, c)=(1,1).



b. Move the mask one pixel to the right , multiply coincident terms sum , and place the new results into the buffer at the location that corresponds to the new center location of the convolution mask which is now at (r,c)=(1,2), continue to the end of the row.



c. Move the mask down on row and repeat the process until the mask is convolved with the entire image. Note that we lose the outer row(s) and columns(s).



Why we use this convolution method when it require, so many more calculation than the basic averaging of the neighbors method?

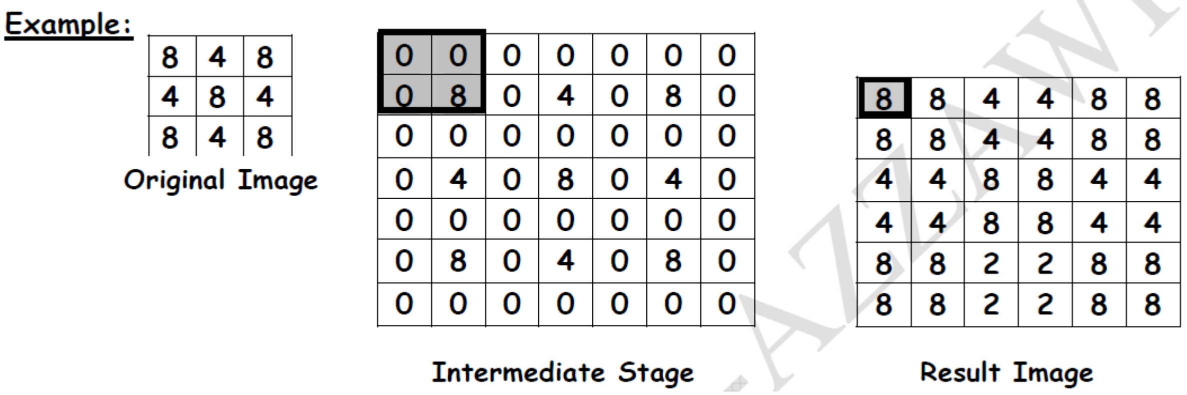
The answer is that many computer boards can perform convolution in hardware, which is generally very fast, typically much faster than applying a faster algorithm in software. Note, only first-order hold be performed via convolution, but zero-order hold can also achieved by extending the image with zeros and using the following convolution mask.

Zero-order hold convolution mask

$$\begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$$

Note that for this mask we will need to put the result in the pixel location corresponding to the lower-right corner because there is no center pixel.

Example:



8	4	8
4	8	4
8	4	8

Original Image

0	0	0	0	0	0	0
0	8	0	4	0	8	0
0	0	0	0	0	0	0
0	4	0	8	0	4	0
0	0	0	0	0	0	0
0	8	0	4	0	8	0
0	0	0	0	0	0	0

Intermediate Stage

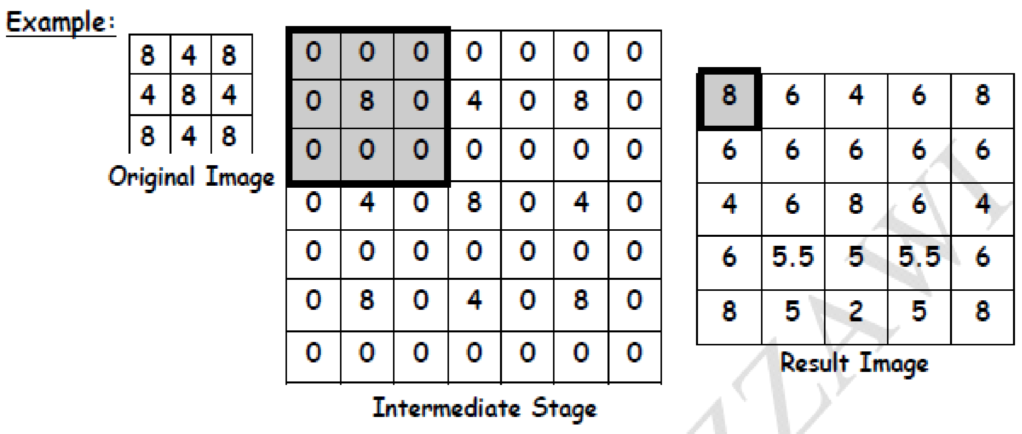
8	8	4	4	8	8
8	8	4	4	8	8
4	4	8	8	4	4
4	4	8	8	4	4
8	8	2	2	8	8
8	8	2	2	8	8

Result Image

Convolution Mask for first –order hold:

Next, we use convolution mask, which is slide a cross the extended image, and perform simple arithmetic operation at each pixel location

$\frac{1}{4}$	$\frac{1}{2}$	$\frac{1}{4}$
$\frac{1}{2}$	1	$\frac{1}{2}$
$\frac{1}{4}$	$\frac{1}{2}$	$\frac{1}{4}$



Example:

8	4	8
4	8	4
8	4	8

Original Image

0	0	0	0	0	0	0
0	8	0	4	0	8	0
0	0	0	0	0	0	0
0	4	0	8	0	4	0
0	0	0	0	0	0	0
0	8	0	4	0	8	0
0	0	0	0	0	0	0

Intermediate Stage

8	6	4	6	8
6	6	6	6	6
4	6	8	6	4
6	5.5	5	5.5	6
8	5	2	5	8

Result Image

These methods will only allow us to enlarge an image by a factor of (2N-1). If we want to enlarge an image by something rather than a factor of (2N-1), then we need to apply a more general method. We take two adjacent values and linearly interpolate more than one value between them. This is done by define an enlargement number k and then following this process:

1- Subtract the two adjacent values.

2- Divide the result by k.

3- Add the result to the smaller value, and keep adding the result from the second step in a running total until all (k-1) intermediate pixel locations are filled.

4- This is done for every pair of adjacent pixels.

Example: we want to enlarge an image to three times its original size, and we have two adjacent pixel values 125 and 140.

1. Find the difference between the two values, 140-125 =15.

2. The desired enlargement is k=3, so we get 15/3=5.

3. Next determine how many intermediate pixel values .we need:

K-1=3-1=2. The two pixel values between the 125 and 140 are 125+5=130 and 125+2*5 = 135.

• We do this for every pair of adjacent pixels .first along the rows and then along the columns. This will allows us to enlarge the image by any factor of K (N-1) +1 where K is an integer and N×N is the image size.

<u>**Note**:</u> the output image (from interpolation or convolution) must be put in a separate image array called a buffer, so that the existing values are not overwritten during the convolution process done