



جامعة المستقبل
AL MUSTAQBAL UNIVERSITY

قسم الانظمة الطبية الذكية كلية العلوم

المحاضرة التاسعة

Image Analysis

المادة : DSP
المرحلة : الثالثة
اسم الاستاذ: م.م. ريام ثائر احمد

- 1. Why Digital Image Analysis?**
- 2. What are Image Features?**
- 3. System Model**
- 4. Why Preprocessing?**
- 5. Region of Interest in Image Geometry**
 - 5.1. Zero Order Hold**
 - 5.2. First Order Hold**
 - 5.3. Convolution**
 - A- Convolution mask for the first-order hold**
 - B- Convolution mask for Zero-order hold**
 - 5.4. Zoom Using K-factor**
- 6. Translation and Rotation**

Prepared by Assist. Teacher .Nada Sabeeh Mohammed

Image Analysis

Image processing is the mathematical analysis of an image. Mathematical Signal and Image Processing is an innovative, rapidly developing branch of Applied Mathematics which involves various mathematical fields and addresses also topics in Computer Science and Engineering.

Image analysis methods extract information from an image. Image analysis differs from other types of image processing methods, such as enhancement or restoration in that the final result of image analysis procedures is a numerical output rather than a picture.

Image analysis involves manipulating the image data to determine exactly the information necessary to help solve a computer imaging problem. This analysis is typically part of a larger process, is iterative and allows us to answer application-specific equations:

Do we need color information?

Do we need to transform the image data into the frequency domain?

Do we need to segment the image to find object information?

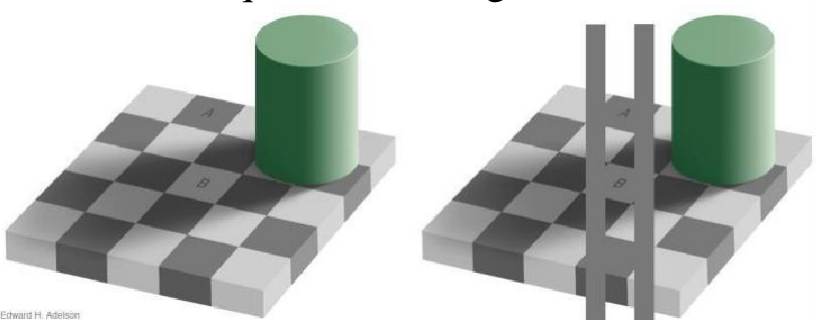
What are the important features of the image?

1. Why Digital Image Analysis?

Human	Computer
Identification	Measuring absolute values
Recognition	Perform calculations
See and describe relations	Never gets tired
Interpretation using experience	Cheap
	Fast
	Objective

For instant:

Which inner square is the brightest?



Edward H. Adelson

How much is dark and bright respectively?

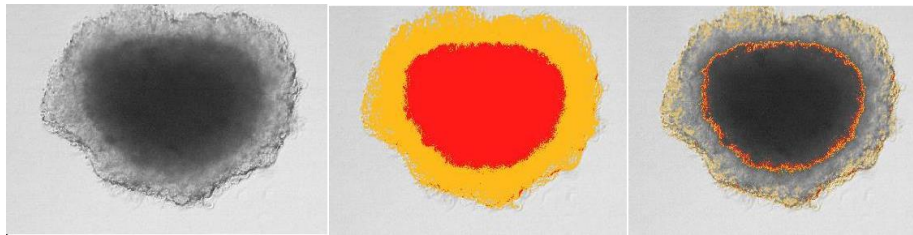





Image Processing vs Image Analysis: Difference in Terminology:

Image Processing		
	⇒	
Image Analysis		
	⇒	<ul style="list-style-type: none">• Blue eyes• 8cm ears• Greek flag

- **Image analysis** is primarily a data reduction process. As we have seen, images contain an enormous amount of data, typically on the order hundreds of kilobytes or even megabytes. Often much of this information is not necessary to solve a specific computer imaging problem, so 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 end product is typically the extraction of high-level information for computer analysis or manipulation. This high-level information

may include a shape parameter to control a robotic manipulator or color and texture features to help in the diagnosis of a skin tumor.

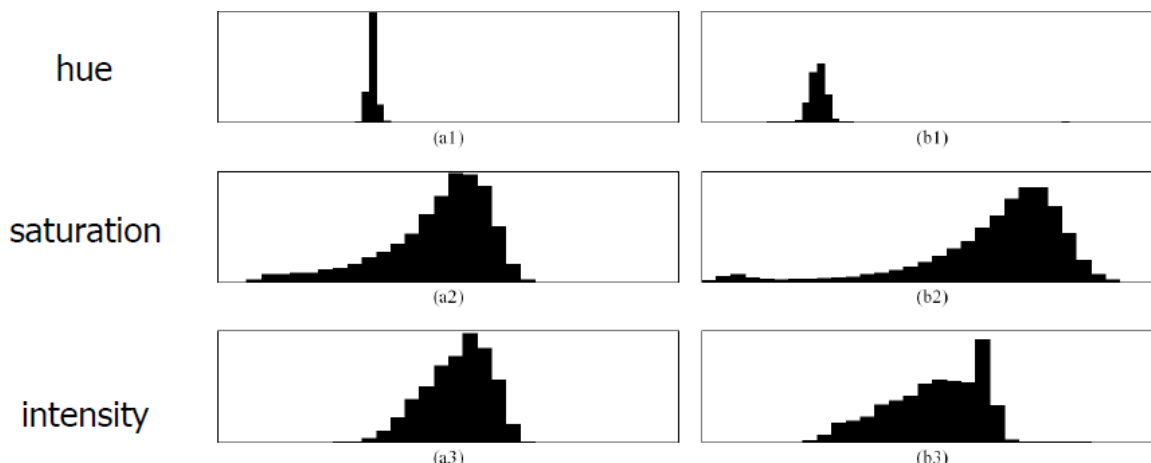
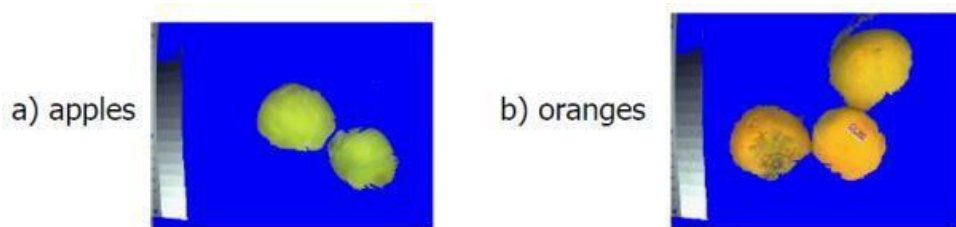
- *In image processing applications*, image analysis methods may be used to help determine the type of processing required and the specific parameters needed for that processing.

For example, determine the degradation function for an image restoration procedure, developing an enhancement algorithm and determining exactly what information is visually important for image compression methods.

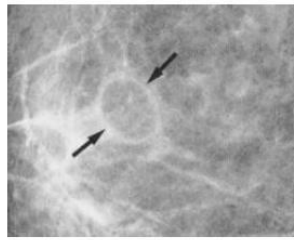
1. What are Image Features? Image features can refer to:

- **Global properties of an image:** i.e. average gray level, the shape of intensity histogram, etc.
- **Local properties of an image:** First we can refer to some local features as image primitives: circles, lines, texels (elements composing a textured region). Second, other local features: the shape of contours, etc.

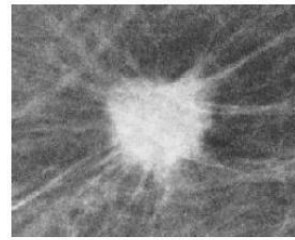
Example of global image features



Example of local image features



Circumscribed (benign) lesions in digital mammography



Spiculated lesions in (digital mammography)

The feature of interest: shape of contour; regularity of contour

-Can be described by Fourier coefficients

-We can build a feature vector for each contour containing its Fourier coefficients

2. System Model

Can be broken down into three primary stages:

2.1. Preprocessing

It is used to remove noise and eliminate irrelevant, visually unnecessary information.

- *Noise* is unwanted information that can result from the image acquisition process.
- Other preprocessing steps might include gray-level or spatial quantization (reducing the number of bits per pixel or the image size) or
- Finding regions of interest for further processing.
- Pre-processing does not increase the image information content
- It is useful on a variety of situations where it helps to suppress information that is not relevant to the specific image processing or analysis task (i.e. background subtraction)
- The aim of preprocessing is to improve image data so that it suppresses undesired distortions and/or it enhances image features that are relevant for further processing
- Early vision: pixel-wise operations; no high-level mechanisms of image analysis are involved
- Types of pre-processing
 - Enhancement (contrast enhancement for contour detection)

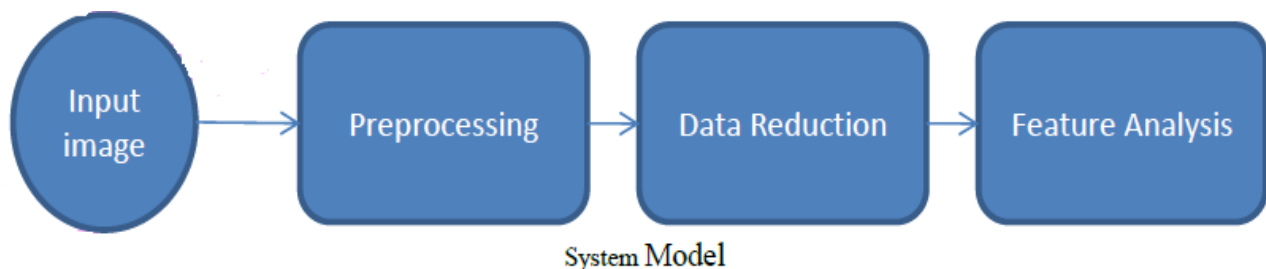
- Restoration (aim to suppress degradation using knowledge about its nature; i.e. relative motion of camera and object, wrong lens focus, etc.)
- Compression (searching for ways to eliminate redundant information from images)

2.2. Data Reduction:

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

2.3. Features Analysis:

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



We can summarize the above steps as the following:

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.

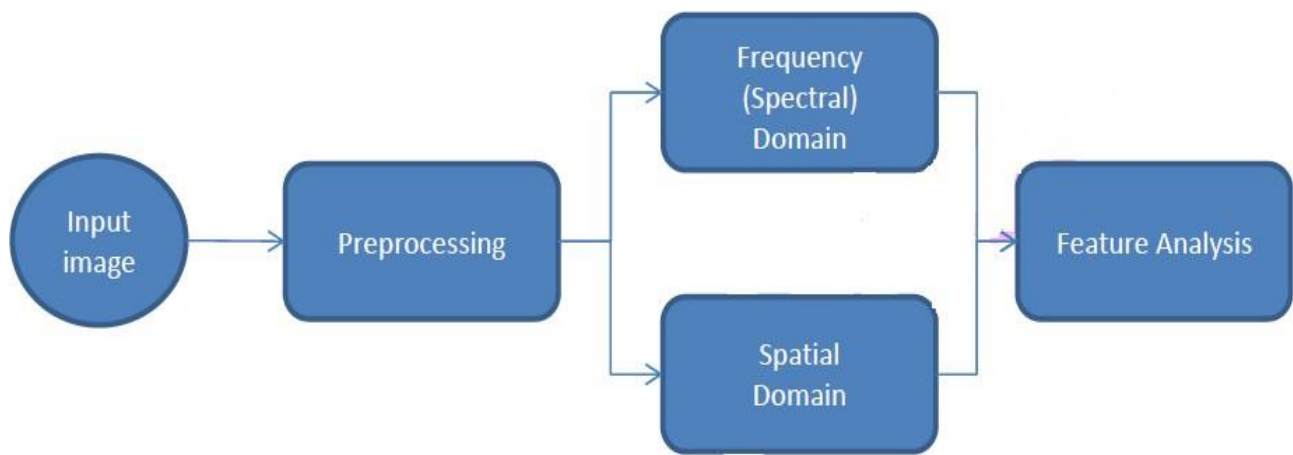


Image Analysis Structure

3. Why Preprocessing?

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

- The above preprocessing include operations related to:

- 1- Extracting regions of interest.
- 2- Performing a basic algebraic operation on the image.
- 3- Enhancing specific image features.
- 4- Reducing data in resolution and brightness.

- A stage where the requirements are typically obvious and simple, such as the removal of artifacts from images, or the elimination of image information that is not required for the application.
- For example, in one application we needed to eliminate borders from the images that had been digitized from the film (the film frames); in another, we had to mask out rulers that were present in skin tumor slides.
- Another example of a preprocessing step involves a robotic gripper that needs to pick and place an object; for this, we reduce a gray-level image to a binary (two-valued) image that contains all the information necessary to discern the object's outline.

4. Region of Interest in Image Geometry

Often, for image analysis, we want to investigate more closely a specific area within the image, called Regions of Interest (ROI). To do this we need operations that modify the spatial coordinates of the image, and these are categorized as image geometry operations.

The image geometry operations include crop, zoom, enlarge, shrink, translate, and rotate.

- Image crop process is the process of selecting a portion of the image, a sub-image, and cutting it away from the rest of the image that's how the border was removed in the following:



An image needing border removal



The image after the border is removed

- Then zoom in by enlarging it.
- Image enlargement is useful in a variety of applications since it can help visual analysis of the detailed object.
- Zoom process can be done in numerous ways, **1. Zero Order Hold. 2. First Order Hold. 3. Convolution.**
- A zero-order hold is performed by repeating previous pixel values, thus creating a blocky effect (give example).
- To extend the image size with a first-order hold we do a linear interpolation between adjacent pixels.
- A comparison of the images resulting from these two methods is shown in the following:



Original image. The ape's face will be zoomed



Image enlarged by zero-order hold, notice the blocky effect



Image enlarged by the first-order hold. Note the smoother effect

- Although the implementation of the zero-order hold is straightforward, the first-order hold is more complicated.
- The easiest way to do this is to find the average value between two pixels and use that as the pixel value between those two; we can do this for the row first (example)
- This method will allow enlarging an $N \times N$ sized image to a $(2N - 1) \times (2N - 1)$ & can be repeated as desired.

4.1. Zero Order Hold: is performed by repeating previous pixel values, thus creating a blocky effect, we can do the zero-order hold as follows:

$$\begin{bmatrix} 8 & 4 & 8 \\ 4 & 8 & 4 \\ 8 & 2 & 8 \end{bmatrix}$$

Original Image Array
 $N \times N = 3 \times 3$

$$\begin{bmatrix} 8 & 8 & 4 & 4 & 8 & 8 \\ 4 & 4 & 8 & 8 & 4 & 4 \\ 8 & 8 & 2 & 2 & 8 & 8 \end{bmatrix}$$

Image with Rows Expanded

Image with rows and columns expanded

$$\begin{bmatrix} 8 & 8 & 4 & 4 & 8 & 8 \\ 8 & 8 & 4 & 4 & 8 & 8 \\ 4 & 4 & 8 & 8 & 4 & 4 \\ 4 & 4 & 8 & 8 & 4 & 4 \\ 8 & 8 & 2 & 2 & 8 & 8 \\ 8 & 8 & 2 & 2 & 8 & 8 \end{bmatrix}$$

$$2N \times 2N = 6 \times 6$$

In zero-order hold the output image size is double original image size $2N \times 2N$, which $N \times N$ is the dimension of the image .

4.2. First Order Hold: is performed by finding linear interpolation between adjacent pixels, finding the average value between two pixels and use that as the pixel value between those two, we can do this for the rows first as follows:

Original Image Array

$$\begin{bmatrix} 8 & 4 & 8 \\ 4 & 8 & 4 \\ 8 & 2 & 8 \end{bmatrix}$$

Image with Rows Expanded

$$\begin{bmatrix} 8 & 6 & 4 & 6 & 8 \\ 4 & 6 & 8 & 6 & 4 \\ 8 & 5 & 2 & 5 & 8 \end{bmatrix}$$

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 the result and expanded the columns in the same way as follows:

$$\begin{bmatrix} 8 & 6 & 4 & 6 & 8 \\ 6 & 6 & 6 & 6 & 6 \\ 4 & 6 & 8 & 6 & 4 \\ 6 & 5.5 & 5 & 5.5 & 6 \\ 8 & 5 & 2 & 5 & 8 \end{bmatrix}$$

Image with rows and columns expanded $(2N-1) \times (2N-1) = 5 \times 5$

This method allows us to enlarge an $N \times N$ sized image to a size of $(2N-1) \times (2N-1)$ and be repeated as desired.

4.3. Convolution:

Another method that achieves a similar result requires a mathematical process called convolution. With this method of image enlargement a two-step process: 1- Extend the image by adding row and columns of zeros between the existing rows and columns.

2- Perform the convolution.

The image is extended as follows:

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

Original Image Array

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

Image extended with zeros

A- Convolution mask for the first-order hold:

The *convolution process* requires us to overlay the mask on the image, multiply the coincident values, and sum all the results.

This is equivalent to finding the *vector inner product* of the mask with the underlying sub-image.

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

Convolution mask for first –order hold

$$\begin{pmatrix} \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \\ \frac{1}{2} & 1 & \frac{1}{2} \\ \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \end{pmatrix}$$

For example, if we put the mask over the upper-left corner of the image, we obtain (from right to left, and top to bottom):

$$\frac{1}{4}(0) + \frac{1}{2}(0) + \frac{1}{4}(0) + \frac{1}{2}(0) + 1(3) + \frac{1}{2}(0) + \frac{1}{4}(0) + \frac{1}{2}(0) + \frac{1}{4}(0) = 3$$

Note that the existing image values do not change. The next step is to slide the mask over by one pixel and repeat the process, as follows:

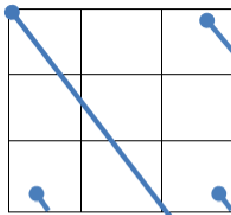
$$\frac{1}{4}(0) + \frac{1}{2}(0) + \frac{1}{4}(0) + \frac{1}{2}(3) + 1(0) + \frac{1}{2}(5) + \frac{1}{4}(0) + \frac{1}{2}(0) + \frac{1}{4}(0) = 4$$

Note this is the average of the two existing neighbors. This process continues until we get to the end of the row, each time placing the result of the operation in the location corresponding to the center of the mask.

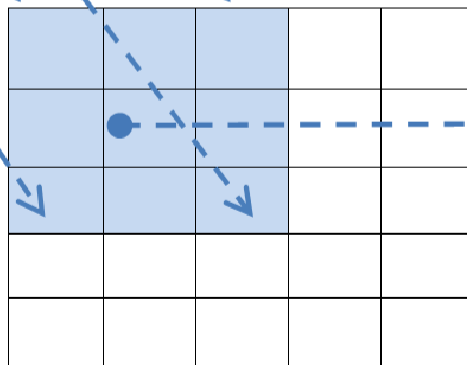
When the end of the row is reached, the mask is moved down one row, and the process is repeated row by row. This procedure has been performed on the entire image, the process of sliding, multiplying and summing is called convolution.

Note that the output image must be put in a separate image array called a buffer, so that the existing values are not overwritten during the convolution process.

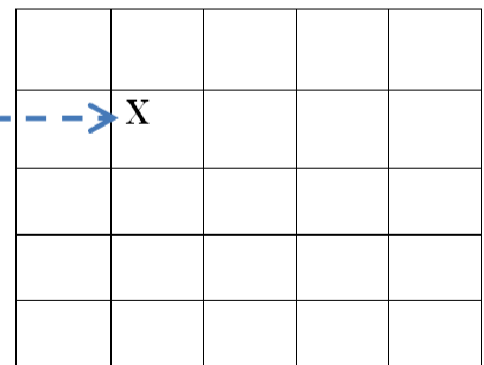
Mask



Image

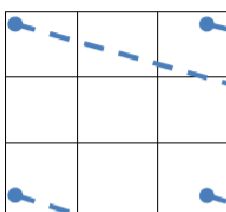


buffer

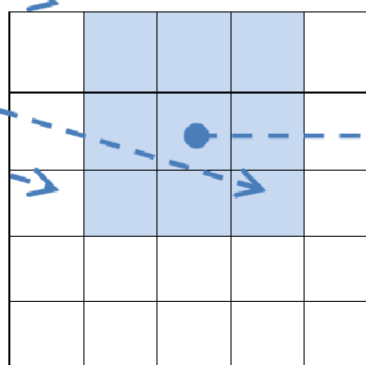


a- Overlay the convolution mask in the upper-left corner of the image. Multiply coincident terms, sum, and put the result into the image buffer at the location that corresponds to the mask's current center, which is $(r,c)=(1,1)$.

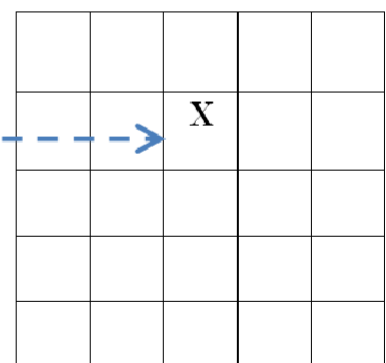
Mask



Image

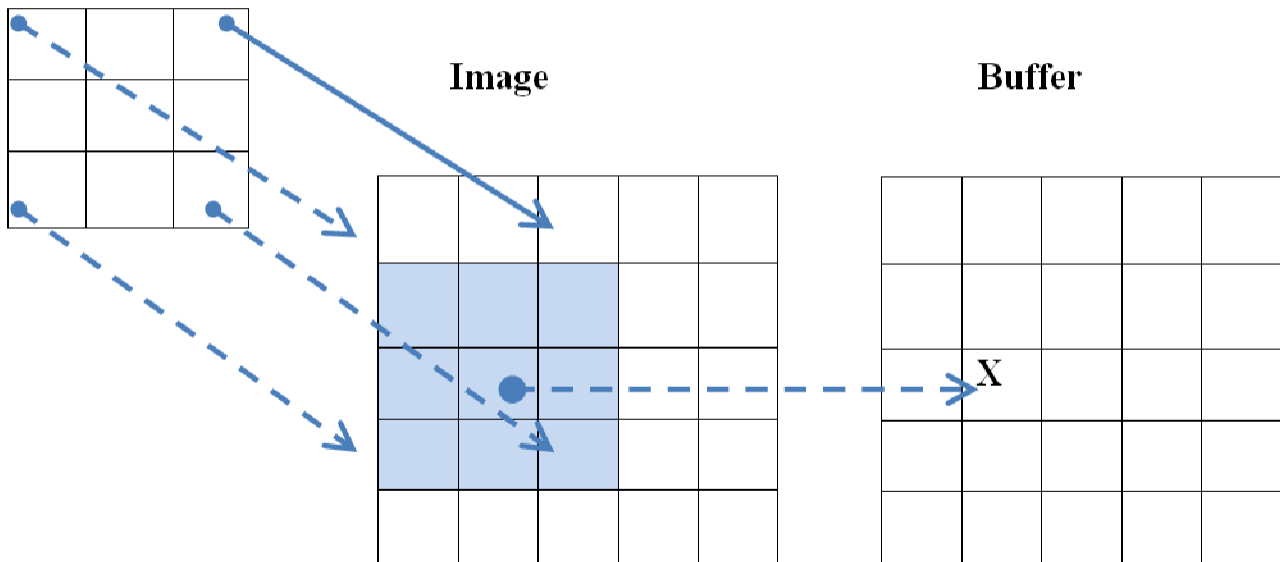


Buffer



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.

Mask



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

Example: Enlarge the following image using a convolution mask for the first-order hold.

$$\begin{bmatrix} 3 & 5 \\ 2 & 7 \end{bmatrix}$$

Solution:-

- Extend the image by adding rows and columns of zeros between the existing rows and columns and the image is extended as follows:

0	0	0	0	0
0	3	0	5	0
0	0	0	0	0
0	2	0	7	0
0	0	0	0	0

Image extended with zeros

b. Perform the convolution:

- 1) $[\frac{1}{4} * 0 + \frac{1}{2} * 0 + \frac{1}{4} * 0 + \frac{1}{2} * 0 + 1 * 3 + \frac{1}{2} * 0 + \frac{1}{4} * 0 + \frac{1}{2} * 0 + \frac{1}{4} * 0] = 3$
- 2) $[\frac{1}{4} * 0 + \frac{1}{2} * 0 + \frac{1}{4} * 0 + \frac{1}{2} * 3 + 1 * 0 + \frac{1}{2} * 5 + \frac{1}{4} * 0 + \frac{1}{2} * 0 + \frac{1}{4} * 0] = 1.5 + 2.5 = 4$
- 3) $[\frac{1}{4} * 0 + \frac{1}{2} * 0 + \frac{1}{4} * 0 + \frac{1}{2} * 0 + 1 * 5 + \frac{1}{2} * 0 + \frac{1}{4} * 0 + \frac{1}{2} * 0 + \frac{1}{4} * 0] = 5$
- 4) $[\frac{1}{4} * 0 + \frac{1}{2} * 3 + \frac{1}{4} * 0 + \frac{1}{2} * 0 + 1 * 0 + \frac{1}{2} * 0 + \frac{1}{4} * 0 + \frac{1}{2} * 2 + \frac{1}{4} * 0] = 1.5 + 1 = 2.5$
- 5) $[\frac{1}{4} * 3 + \frac{1}{2} * 0 + \frac{1}{4} * 5 + \frac{1}{2} * 0 + 1 * 0 + \frac{1}{2} * 0 + \frac{1}{4} * 2 + \frac{1}{2} * 0 + \frac{1}{4} * 7] = 0.75 + 1.25 + 0.5 + 1.75 = 4.25$
- 6) $[\frac{1}{4} * 0 + \frac{1}{2} * 5 + \frac{1}{4} * 0 + \frac{1}{2} * 0 + 1 * 0 + \frac{1}{2} * 0 + \frac{1}{4} * 0 + \frac{1}{2} * 7 + \frac{1}{4} * 0] = 2.5 + 3.5 = 6$
- 7) $[\frac{1}{4} * 0 + \frac{1}{2} * 0 + \frac{1}{4} * 0 + \frac{1}{2} * 0 + 1 * 2 + \frac{1}{2} * 0 + \frac{1}{4} * 0 + \frac{1}{2} * 0 + \frac{1}{4} * 0] = 2$
- 8) $[\frac{1}{4} * 0 + \frac{1}{2} * 0 + \frac{1}{4} * 0 + \frac{1}{2} * 2 + 1 * 0 + \frac{1}{2} * 7 + \frac{1}{4} * 0 + \frac{1}{2} * 0 + \frac{1}{4} * 0] = 1 + 3.5 = 4.5$
- 9) $[\frac{1}{4} * 0 + \frac{1}{2} * 0 + \frac{1}{4} * 0 + \frac{1}{2} * 0 + 1 * 7 + \frac{1}{2} * 0 + \frac{1}{4} * 0 + \frac{1}{2} * 0 + \frac{1}{4} * 0] = 7$

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 4 & 5 & 0 \\ 0 & 2.5 & 4.25 & 6 & 0 \\ 0 & 2 & 4.5 & 7 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Enlarge image by convolution mask for the first-order hold

Why we use this convolution method when it requires, so many more calculations 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. Not only the first-order hold be performed via convolution, but zero-order hold can also be achieved by extending the image with zeros and using the following convolution mask.

B- Convolution mask for Zero-order hold:

Not only the first-order hold is performed via convolution, but zero-order hold can also be achieved by extending the image with zeros and using the following convolution mask.

Convolution mask for Zero-order hold

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

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: Enlarge the following image using convolution mask for zero order hold.

$$\begin{bmatrix} 3 & 5 \\ 2 & 7 \end{bmatrix}$$

a. Extend the image by adding rows and columns of zeros between the existing rows and columns and the image is extended as follows:

Image extended with zeros

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 5 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 7 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

b. Perform the convolution:

1) $1*0 + 1*0 + 1*0 + 1*3 = 3$

2) $1*0 + 1*0 + 1*3 + 1*0 = 3$

3) $1*0 + 1*0 + 1*0 + 1*5 = 5$

4) $1*0 + 1*0 + 1*5 + 1*0 = 5$

5) $1*0 + 1*3 + 1*0 + 1*0 = 3$

6) $1*3 + 1*0 + 1*0 + 1*0 = 3$

7) $1*0 + 1*5 + 1*0 + 1*0 = 5$

8) $1*5 + 1*0 + 1*0 + 1*0 = 5$

9) $1*0 + 1*0 + 1*0 + 1*2 = 2$

10) $1*0 + 1*0 + 1*2 + 1*0 = 2$

$$11) 1*0 + 1*0 + 1*0 + 1*7 = 7$$

$$12) 1*0 + 1*0 + 1*7 + 1*0 = 7$$

$$13) 1*0 + 1*2 + 1*0 + 1*0 = 2$$

$$14) 1*2 + 1*0 + 1*0 + 1*0 = 2$$

$$15) 1*0 + 1*7 + 1*0 + 1*0 = 7$$

$$16) 1*7 + 1*0 + 1*0 + 1*0 = 7$$

0	0	0	0	0
0	3	3	5	5
0	3	3	5	5
0	2	2	7	7
0	2	2	7	7

Enlarge image by convolution mask for zero-order hold

4.4. Zoom Using K-factor:

The above methods will only enlarge an image by a factor of (2N - 1), but what if we want to enlarge an image by something other than a factor of (2N - 1)?

To do this we need to apply a more general method; we take two adjacent values and linearly interpolate more than one value between them.

This *linear interpolation technique* is equivalent to finding the line that connects the two values in the brightness space and sampling it faster to get more samples, thus artificially increasing the resolution.

This is done by defining an enlargement number k and then following this process:

1. Subtract the result by k.
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.

Example: we want to enlarge an image to five times its original size, and we have two adjacent pixel values 210 and 240.

1. Find the difference between the two values, $240 - 210 = 30$.
2. The desired enlargement is $k = 5$, so we get $30 / 5 = 6$.
3. Nextmask's determine how many intermediate pixel values, we need $K-1 = (5 - 1) = 4$. The four-pixel values between the 210 and 240 are:

$K = 1$:

$$210 + (6 * 1) = 216.$$

$K = 2$:

$$210 + (6 * 2) = 222.$$

$K = 3$:

$$210 + (6 * 3) = 228.$$

$K = 4$:

$$210 + (6 * 4) = 234.$$

- We do this for every pair of adjacent pixels first along the rows and then along the columns. This will allow us to enlarge the image by any factor of $K (N-1) + 1$ where K is an integer and $N \times N$ is the image size.
- To process opposite to enlarging an image is shrinking. This process is done by reducing the amount of data that needs to be processed.

6. Translation and Rotation:

Two other operations of interest image geometry are Translation and Rotation. These processes may be performed for many application-specific reasons, for example, to align an image with a known template in the pattern matching process or make certain image details easier to see.

The translation process can be done in the following equations:

$$R' = R + R_0$$

$$C' = C + C_0$$

Where R' and C' are new coordinates, R and C are the original coordinates and R_0 and C_0 are the distances to move or translate the image.

The rotation process requires the use of these equations:

$$R^{\wedge} = R (\cos \theta) + C (\sin \theta)$$

$$C^{\wedge} = -R (\sin \theta) + C (\cos \theta)$$

Where R^{\wedge} and C^{\wedge} are the new coordinates, R and C are the original coordinates, θ is the angle to rotate the image, θ is defined in a clockwise direction from the horizontal axis at the image origin in the upper left corner.

The rotation and translation process can be combined into one set of equations:

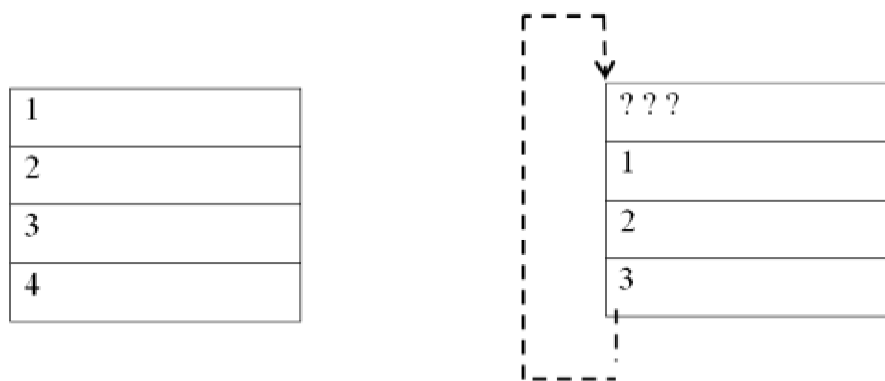
$$R^{\wedge} = (R + R0) (\cos \theta) + (C + C0) (\sin \theta)$$

$$C^{\wedge} = - (R + R0) (\sin \theta) + (C + C0) (\cos \theta)$$

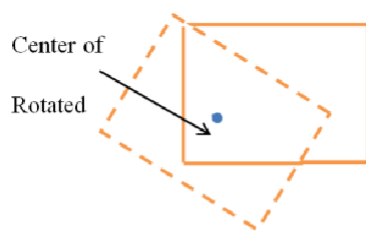
Where R^{\wedge} and C^{\wedge} are the new coordinates and R , C , $R0$, $C0$, and θ are previously defined.

There are some practical difficulties with the direct application of these equations when translating, if we move everything one row down, what do we put in the top row? There are two basic options.

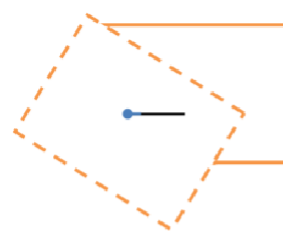
- 1- Fill the top row with a constant value typically black (0) or white (255).
- 2- Warp around by shifting the bottom row to the top, shown in the following:



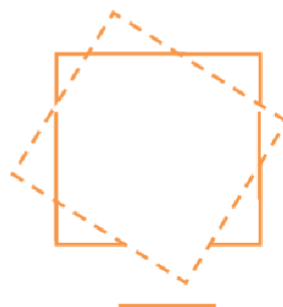
Translation



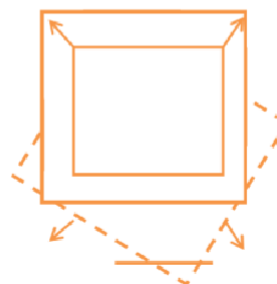
A. Image is rotated off the screen.



B. fix by translating toward center



C. Translation complete



D. Crop and enlarge if desired

Rotation

7. Image Quality Metrics

In this section, we review the most popular pixel-based distortion criteria that measure the differences between two images and measurements of image quality.

1.7 Definition: (Decibel)

Decibel (dB) is a unit of measurement that expresses the logarithmic ratio of two physical quantities of the same dimensions. The logarithm is to base 10. This logscale definition is useful when the quantities have a wide range and losses or gains are proportional.

1.7 Root Mean-Square Error (RMSE)

A very simple and very common distortion measure is RootMean-Square Error (RMSE). It is defined in the following equation.

$$\text{RMSE} = \left(\frac{\sum_i (p_i - q_i)^2}{n} \right)^{1/2}$$

The pixels of the two images are given by p_i and q_i . Hence, the images are compared pixel for pixel, and the resulting distortion measure is averaged over all pixels.

7.3 Mean Square Error (MSE)

The MSE represents the cumulative squared error between the can also be used. The mathematical formula for the mean square error is:

$$\text{MSE} = \frac{1}{m \times n} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} [I(i, j) - K(i, j)]^2$$

RMSE may not be comparable between different images if the signal in the images differs greatly. A better choice, in that case, is the Signal-to-NoiseRatio (SNR) or Peak Signal-to-noise Ratio (PSNR). These measures relate the error to the content of the image, thereby yielding a better impression of how important the noise is.

7.4 Signal-to-Noise Ratio (SNR)

SNR is expressed in dB and is commonly defined as in the following equation

$$SNR = 20 \cdot \log_{10} \left(\frac{\sum_i p_i^2}{\sum_i (p_i - q_i)^2} \right)$$

7.5 Peak Signal to Noise Ratio (PSNR)

The PSNR block computes the peak signal-to-noise ratio, in decibels, between two images. This ratio is used as a quality measurement between the original and a compressed image. The higher the PSNR, the better the quality of the compressed, or reconstructed image.

$$PSNR = 10 \log(\text{MAX}_i^2 / \text{MSE})$$

where MAX_i is the maximum possible pixel value of the image (255 for grayscale images and 8 bits per pixel). The high PSNR value indicates high security. PSNR is measured in decibels (dB).

The MSE represents the cumulative squared error between the compressed or reconstructed and the original image, whereas PSNR represents a measure of the peak error. The lower the value of MSE, the lower the error.

7.6 Bit Error Rate (BER)

Bit error rate (BER) can be calculated as the actual number of bit positions which are changed in the processed image compared with the original image.

BER measurements compare digital input and output images to assess what fraction of the bits are received incorrectly (percentage %). It is defined as

$$BER = \frac{E(t)}{N(t)}$$

Where $E(t)$ is the number of bits received in error over time t , and is the total number of bits transmitted in time t .

on the other hand, the BER is related to SNR through the following equation.

$$BER = \frac{1}{2} \left[1 - \text{erf} \left(\frac{\sqrt{SNR}}{2\sqrt{2}} \right) \right]$$



where erf represents the error function. For $x > 3$, a very good approximation to erf(x) is

$$\text{erf}(x) = 1 - \frac{1}{\sqrt{\pi} x} e^{-x^2}$$

Thus, for $\text{SNR} > 72$, the first equation can be approximated by

$$\text{BER} \approx \left(\frac{2}{\pi \text{SNR}} \right)^{\frac{1}{2}} e^{-\frac{\text{SNR}}{8}}$$

7.8 Normalized Correlation (NC)

Normalized Correlation (NC) is a measure of similarity of two images.

$$\text{NC}(X, \hat{X}) = \frac{\sum_i \sum_j X(i, j) \hat{X}(i, j)}{\sqrt{\sum_i \sum_j X(i, j)^2} \sqrt{\sum_i \sum_j \hat{X}(i, j)^2}}$$

X and \hat{X} stand for the original and the processed images respectively