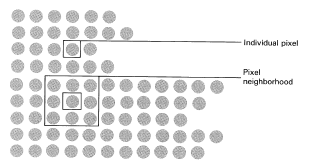
**2. Basic Machine Vision Processing**

The processing of data in the vision system can be categorized into :

(i) Point to point (monadic) alteration of data on a global scale

(ii) Multiple point (dyadic) determination of elements of a new array of the image.

The generation of the new image matrix will be a function of either individual pixel location values or values of pixels in the neighborhood of the individual cell.

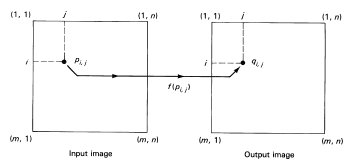


Point and neighborhood functions

**Monadic operations** involve the generation of a new array by modifying the pixel value at a single location based on a global rule applied to every location in the original array)

The sequence of one-to-one transformation is:

* Obtaining the pixel value of a given location in the array
* Modifying it by a simple linear or nonlinear operation
* Placing the new pixel value in the specific corresponding location of the new array

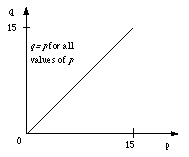
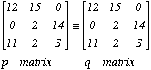
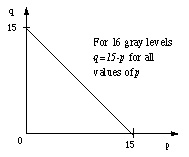
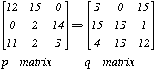
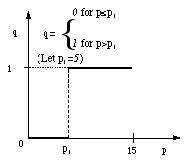
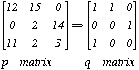
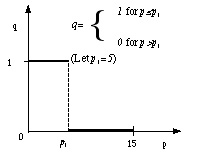
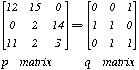
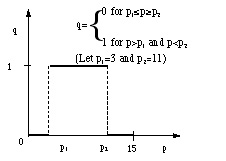
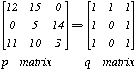
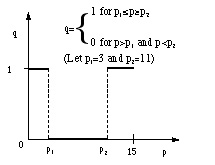
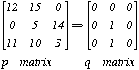
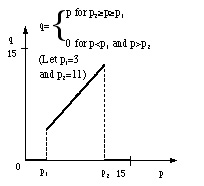
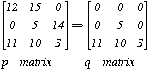
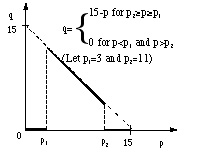
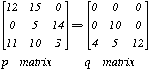
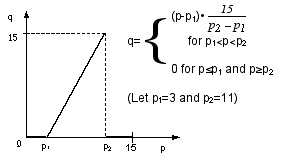
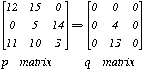
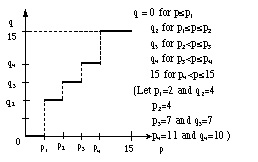
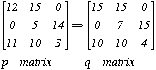


Monadic operator

Operator *f* transforms the gray level value of every pixel independently of others and the new value is defined as : *g(I,j) = f[p(i,,j)].*

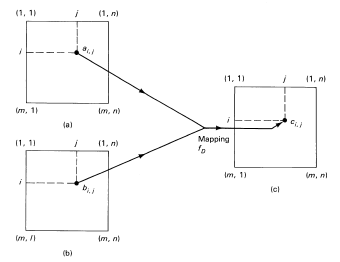
The resultant image matrix will be the same size as original image.

**Dyadic operations** use essentially the same procedure except that the output image is generated in one of two ways :

* The value of each pixel in the array is dependent on a combination of the values of the corresponding pixel locations in two or more similar image frames.
* The pixel values in the region adjacent to the location of the pixel value being generated are combined.
* **2.1 Monadic One Point Transformation.**
* ***(i) Identity Operator*** ( creation of the identical output image)
* The function *f* is a straight line starting at the origin and extending to the maximum pixel value of the image system.
* 
* Identity operator: input, output, and function
* 
* ***(ii) Inverse Operator*** (creation of the inverse to input image output)
* The function is the straight line have a value of maximum gray value at the minimum gray input value and equal zero at the maximum gray input value.
* 
* Inverse operator: input, output, and function
* 
* ***(iii) Threshold Operator*** ( result in a binary output image where the level of transformation is given by the input parameter *p1* known as threshold)
* All pixel values below the *p1* are converted to a zero and all pixel values equal or greater than *p1* are converted to a one.
* 
* Threshold operator: input, output, and function
* 
* ***(iv) Inverted Threshold Operator*** (the output image can be inverted by applying the threshold function). Every pixel in original image which was a light will be dark and pixels which were dark will be light.
* 
* Inverted threshold operator: input, output, and function
* 
* ***(v) Binary Threshold Interval*** (All pixels with value between *p1* and *p2* are converted to one and outside the interval are converted to zero.)
* 
* Binary Threshold Interval operator: input, output, and function
* 
* The pixel value is maintained if belongs to interval p1=100, p2=150 otherwise it equal zero
* 
* The pixel value is maintained if does not belongs to interval p1=100, p2=150 otherwise it equal zero
* ***(vi) Inverted Binary Threshold Operator*** ( is used to convert a multilevel gray scale image to a binary image or it can be applied to a binary image for inversion purposes)
* 
* Inverted Binary Threshold operator: input, output, and function
* ***(vii) Gray Scale Threshold Operator*** ( it results in a gray scale output image for gray scale image values between *p1* and *p2* and makes all input values outside the *p1* and *p2* zero) It used for identification of image features having a specific value for pseudocolor and application processing techniques.
* 
* Gray Scale Threshold Operator: input, output, and function
* ***(viii) Inverted Gray Scale Threshold Operator*** (This operator can highlight specific features such as roads or all the similar areas in the image)
* 
* Inverted Gray Scale Threshold Operator: input, output, and function
* ***(ix) Stretch Operator*** (it results in a full gray scale output image corresponding to the input interval *p1* and *p2* and suppresses all values outside this range)
* 
* Stretch Operator: input, output, and function
* ***(x) Gray Level Reduction Operator*** (the output image has a smaller number of the gray levels than the number of gray levels of the input image)
* 
* Gray Level Reduction Operator: function a), input b), and
* output with *q1*, *q2, q3, q4, 15 gray levels* c)

**2.2. Dyadic Two Point Transformations**

The dyadic point-by-point operator uses the information contained at the same location in two images. The size of the matrix does not change and the dyadic operator *fD* be linear or nonlinear.



Dyadic multipoint-to-point operator: input image a) and input image b)

are mapped into a new output image c).

The characteristic function is given by equation:

*ci,j=fD (ai,j ,bi,j)*

where *fD* is a function of two variables and *i* and *j* range from *0* to *m* and *n*, respectively. The function *fD* contains an appropriate SCALING FACTOR *k* to keep the magnitude of the output values within the scale range to avoid an overflow or a negative condition.

***(i) Image Addition*** (is used to reduce the effect of noise in the data)

The value of the output *ci,j*is given by

*ci,j=(ai,j ,bi,j) / k*

over the *i,j* range of values where *k* equals the scaling factor.

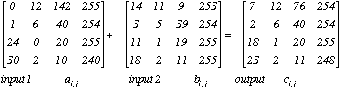


Image addition for *i=j=4 (ci,j=(ai,j +bi,j) / 2 values are rounded upward)*

The image addition dyadic process averages the data in the two input image matrices. There is an accuracy improvement and noise reduction when the procedure is used with a large number of samples.



Gray scale images addition



Color images addition

***(ii) Image Subtraction*** (is used to detect changes that have occurred during the time interval between when two images of the same scene were taken)

In order to obtain positive numbers in output image:

* Rescaling where the largest negative number is set equal to zero and the largest positive number is set to the maximum gray scale (for example 255). The relationship is defined as *ci,j = (ai,j - bi,j)*
* Define the difference as an unsigned difference or the absolute value of the difference (used in limited number of conditions)

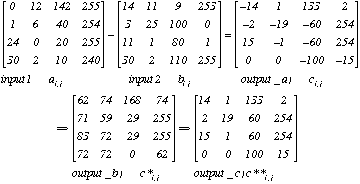


Image subtraction: a) resultant output with negative numbers, b) in the same range from 0 to 255, c) absolute values

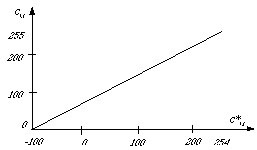


Image subtraction with function for output\_b) *c\*i,j*

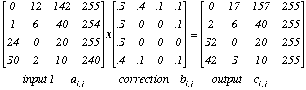
The values *Ri,j* will be 0 when *ci,j* is -100 and 255 when *ci,j* is 254 (*Ri,j* *=(ci,j* +100)x(255/354))



***(iii) Image Multiplication*** (is used to correct for nonlinearity of the sensor where there is a special nonuniform sensitivity over the viewing area (multiplication by correction matrix))

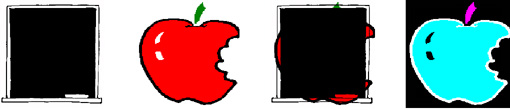
The relationship is given by the equation: *ci,j = K [( ai,j x bi,j ) + ai,j ]*

where all values are rounded up to next integer, the maximum value is 255, and *bi,j* is the correction factor.



Another use of the multiplication operator would be create a window in order to reduce the computation to a specific area of interest. The correction matrix *bi,j* would be given by the equation *ci,j = ( ai,j x bi,j )*, where *bi,j* is 1 for all pairs inside the window area and 0 for all pairs outside the window area. The size of he output image matrix is identical to the size of the input matrix.

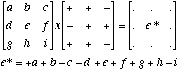
Example of multiplication/division is shown in following figure.



---------------Two original images ---------------Multiplication---------------Division

***(iv) Image Convolution (Spatial transformation)***

A new image matrix can be generated in the way where the pixel value assigned to each location is a function of the pixel values of the adjacent locations by the *3x3* convolution. In the 3x3 convolution the pixel value of the center location is computed on based of values of the nine locations. The size of the resultant matrix is reduced by 2 in each direction due to the edge effec



Convolution (neighborhood) transformation





Example of transform operations: original image, interpolation, subtraction of two previous images,

convolution of subtracted image and inverse of convoluted image.

**2. 3 Mathematic Transformation. Spatial - Frequency Approaches**

Mathematic transformations are there where the output image *g(x,y)* is obtained from input image *f(x,y)* applying the set of the linear or nonlinear functions. There are two types of the transformations:

a) Space domain transformations: convolution, correlation, geometrical transformations.

b) Frequency domain transformations: Fourier transform

**2.3.1. Spatial processing**

This type of processing is based on the techniques, which operate directly over the pixel values of the input image according the equation:

*S(x,y) = F [ N(x,y) ]*

***(i) Convolutions***

For one-dimensional case of the continuous functions it is defined for the function *f(x)* with respect to the function *h(x)* such that it produces the new function :

C:\Documents and Settings\khec\Desktop\chap3\24_files\Image23_2.gif

In the case of discrete functions for two sequences *f(x)* and *h(x)* the output function is obtained as:

[C:\Documents and Settings\khec\Desktop\chap3\24_files\Image23_3.gif](file:///C:\Documents%20and%20Settings\khec\Desktop\chap3\24_files\Image23_3.gif)

In case of the digital images the discrete bi-dimensional convolutions are used as

C:\Documents and Settings\khec\Desktop\chap3\24_files\Image23_4.gif

Usually, the *3x3* matrix convolution is applied for digital images. Therefore, the previous expression can be rewritten as:

C:\Documents and Settings\khec\Desktop\chap3\24_files\Image23_5.gif

For example, the output matrix *g(2,2)* will be computed as:

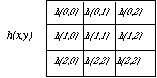
C:\Documents and Settings\khec\Desktop\chap3\24_files\Image23_6.gif

*= f(0,0)h(2,2)+f(0,1)h(2.1)+f(0,2)h(2,0)+*

*f(1,0)h(1,2)+f(1,1)h(1,1)+f(1,2)h(1,0)+*

*f(2,0)h(0,2)+f(2,1)h(0,1)+f(2,2)h(0,0)*

As result, in image processing instead of the convolution of the *f(x,y)* with *h(x,y)* the convolution of the *f(x,y)* with *3x3* mask is used. This mask can be presented as

**

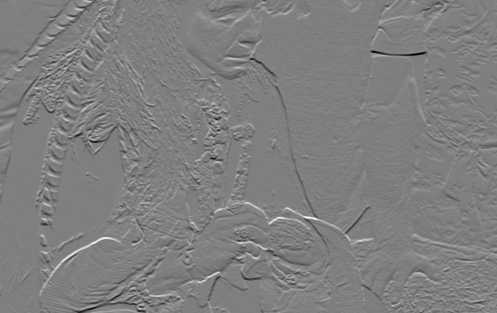
Graphically, in order to obtain new value of a pixel the 3x3 matrix of *f(x,y)* is imposed over another matrix (mask) *h(-i,-j)* multiplying corresponding elements and computing the sum of the products. In order to obtain the value of a next pixel the mask is displaced one pixel. For pixels on the borders it is not possible to apply the convolution, therefore, the output image will have the smaller size than the original with new dimensions as :

*(N-(n-1))x(M-(m-1))*

where *N* and *M* are dimensions of the original image and *n* and *m* the dimensions of the mask.

Usually, the convolution implies the normalization of the result for avoiding the negative numbers. One example of the convolution is shown in figure





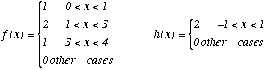
Convolution of the digital image

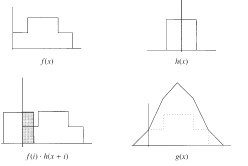
***(ii) Correlation***

The correlation between two functions *f(x)* and *h(x)* is defined as :

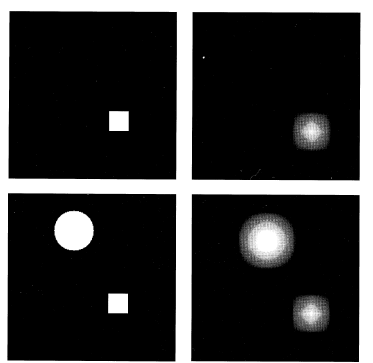
C:\Documents and Settings\khec\Desktop\chap3\24_files\Image23_8.gif

where the *f\** is a complex conjugates (for digital image processing where only the real number are used, the *f\** is equal to *f*). Example of the correlation between two function is presented in figure where





Analyzing the previous example, it is possible to note that correlation is useful for localization of the particular pattern within the image, because the coordinates with maximum correlation correspond to the coordinates where pattern lies.



Detection of the quadrate by applying the correlation

The correlation of the quadratic pattern with the first image gives the coordinates where the same patter lies. In the second image there is another object which covers the quadratic pattern. The function detects two objects but this result is not good for detection of the quadratic pattern.

In this case the normalized correlation is used which can be defined as:

C:\Documents and Settings\khec\Desktop\chap3\24_files\Image23_13.gif

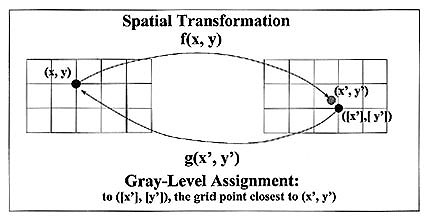
***(iii) Geometrical transformations***

Geometrical transformations modify the spatial relationship between pixels. Usually this type of transformation is based on the transformation algorithms for determination of the relationship between the coordinates of the original and output image and on the algorithms for interpolation which define the pixel gray level of output image on base of one or some pixel values of the input image.

**a) GEOMETRIC INTERPOLATION**

Interpolation may be visualized as a process of generation a continuous function ***g(t)*** from a given discrete sample ***g[m***] by application of a suitable fitting function to a number n of nearest neighbors at each of the sample point.

An image transformation of the form ***H (hx,hy)***, where ***x' hx(x,y***) and ***y' hy(x,y)*** denote pixel coordinates in the target image, prescribes the mapping of each original pixel value, ***I(x,y),*** into a new pixel value ***I(x',y').***



However, the transformation ***H*** will generally yield non-integer target coordinates ***x'*** and ***y'***.

The interpolation is necessary because in general case for one pixel it is not possible to obtain the integer value in output image (for example, after translation). Sometimes the set of the pixels of the input image will correspond to one pixel of the output image. Therefore, the speed and quality of transformation depends on the used interpolation algorithms.

In general case for interpolation the output image does not receive from the input image. The inverse operation is applied: for each pixel of the output image the input image's pixels are found which are used for computing the gray level of pixel in output image.

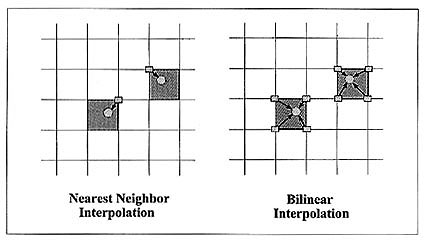
The equation for interpolation is:

C:\Documents and Settings\khec\Desktop\chap3\24_files\Image23_19.gif

where *g(x,y)* is the output image, *f(x,y)* input image, *h(x,y)* represents the interpolation.

**There are some forms of interpolation:**

* based on inspection of nearest-neighbor location (low order interpolation, ***n=1***)
* considering the shell of the four nearest ( high order interpolation, ***n=2***) or twelve next nearest (***n=3***) neighbors locations and computing a new gray value on the basis of a weights of these four (twelve) input values (bilinear interpolation).



Nearest-neighbor and bilinear interpolation

**Common Geometric Image Interpolation Procedures**

1. Nearest-neighbor Interpolation (simple rounding***, n=1***):

Strategy: assign gay value of closest pixel

Comments: image shifted by up to ***1/2*** pixel; gray value remain uncharged

Expression: ***Output[x',y'] = Input[(int)(y+0.5), (int)(x+0.5)]***

2. Bilinear Interpolation : ***n=2***

Strategy: select weighted sum of 4 nearest pixels in Input image

a) weights proportional to distance

b) weights proportional to square of distance

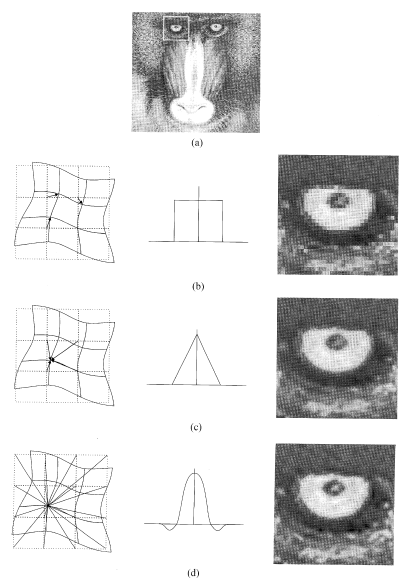
Comments: new gray values appear

Expression:

a) C:\Documents and Settings\khec\Desktop\chap3\24_files\Image1.gif

b) C:\Documents and Settings\khec\Desktop\chap3\24_files\Image2.gif

There are different techniques for interpolation: the nearest neighbor interpolation, bilinear interpolation, bicubic interpolation, display/aperture interpolation, Lagrange interpolation, etc.



Interpolation a) original image, b) the nearest neighbor, c) bilinear, d) bicubic

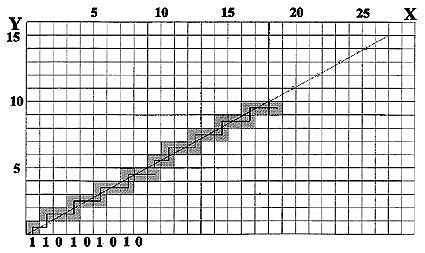
***b)* IMAGE SCALING**

Adjustment of the scale in an image is often desirable either to zoom in on details by magnification or to zoom out to gain an overall overview by reduction.

Simple integer scaling is accomplished by expansion or reduction of the horizontal and vertical dimensions separately with respect scale factor ***kx*** and ***ky*** preserving ***kx=ky=k.***

Magnification requires an replication each existing pixel ***kx*** times and ***ky*** times to fill in additional columns and rows of the magnified image. Reduction implies deletion of pixel from original: ***k-1*** of every ***k*** pixels are removed to achieve a ***k***-fold image reduction (additional information is in the chapter 3.2.vi)

The scaling if images by nonlinear factors is achieved by interpolation. The method is : the factor ***5/9*** may be achieved by plotting the ***5/9*** slope on a pixel grid. Discrete approximation is "***1"*** if there are changes in the interval and "***0"*** if the approximating staircase line is continuos.



Binary code for interpolated image reduction.

For horizontal reduction the columns marked by ***0*** are eliminated

**c) IMAGE ROTATION**

Image rotation is a transformation to change the orientation of the original image on screen with alteration of image size (except the rotation by ***180º*** sometimes ***90º***)

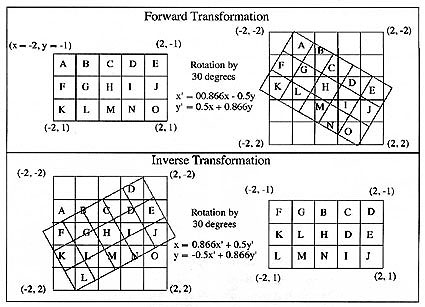
Rotation by arbitrary angle is more complex and explicitly specified in the form of an orthogonal transformation

C:\Documents and Settings\khec\Desktop\chap3\24_files\Image3.gif

There are two problems of computing the target image :

1. target coordinates may fall on non-integer locations (interpolation using the neighbor pixels)

2. Rotation of the input image will generate a shape that is no longer congruent with the display (strategy - use backwards - change image size, or consider that the outside locations of target image are lost)



Difficulties during the rotation of a rectangular input image (***3x5*** image by ***30º***), so that the target image is computed by proposed equations

***d) Resumen of transformation operations***

For displacement of the origin of input image the following equation is applied:



In the homogeneous coordinates is presented as:

C:\Documents and Settings\khec\Desktop\chap3\24_files\Image23_15.gif

*b) Magnification*

C:\Documents and Settings\khec\Desktop\chap3\24_files\Image23_16.gif

*c) Rotation with respect to origin*

C:\Documents and Settings\khec\Desktop\chap3\24_files\Image23_17.gif

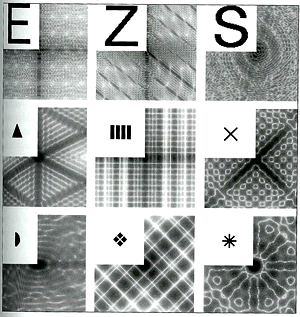
*d) Rotation with respect to any point*

C:\Documents and Settings\khec\Desktop\chap3\24_files\Image23_18.gif

**2.3.2. Frequency domain transformations**

***(i) Frequency Space***

The Fourier transform and other frequency space transforms are applied to two-dimensional images for many different reasons. Some of these *have a little to do with the purpose of enhancing visibility or selection of features or structures of interest for measurement*, for example, image compression, motion characteristics measurement, data loss analysis, etc. Some examples of two-dimensional and their frequency domain representations are shown in the next figure.



Let the function***f(x)*** is a continuous and well-behaved function where ***x*** is a real variable representing either time or distance in one direction across and image. Do not worry that in a digitized image the values of x are not continuous. They are discrete based on pixel spacing and the possible brightness values are quantized as well. These values are considered to sample the real or analog image that exists outside the computer. In this case the discrete Fourier transform will be used.

Fourier's theorem states that it is possible to form any one dimensional function ***f(x)*** as a summation of a series of sine and cosine terms of increasing frequency as it shown in next figure.

|  |  |
| --- | --- |
| C:\Documents and Settings\khec\Desktop\chap3\23_files\image_fourier.gif | ***f(x)=sin(2f)t*** |
| ***f(x)=sin(2f)t + 1/3sin3(2f)t*** |
| ***f(x)=sin(2f)t + 1/3sin3(2f)t + 1/5sin5(2f)t*** |
| ***f(x)=sin(2f)t + 1/3sin3(2f)t + 1/5sin5(2f)t + 1/7sin7(2f)t + 1/9sin9(2f)t +***  ***+ 1/11sin11(2f)t + 1/13sin13(2f)t*** |

The figure shows the familiar case of a step function, illustrated the ability to add up series of sine waves to produce the desired step with 1, 2, 3, and 7 terms.

***(ii) Fourier Series***

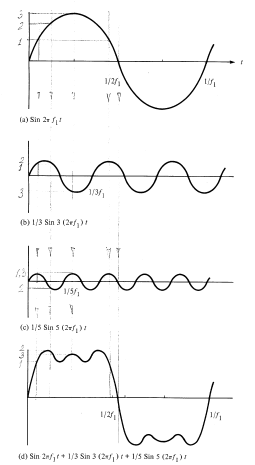
In general any periodic function *f(t)* with period *T0* may be represented as :

C:\Documents and Settings\khec\Desktop\chap3\23_files\Image232_1.gif

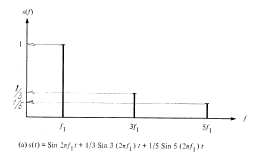
where

C:\Documents and Settings\khec\Desktop\chap3\23_files\Image232_2.gif

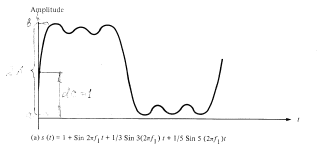
Another representation of the previous example with the step function tis shown as:



Decomposition of the periodic function by its Fourier series



Frequency domain representation



Signal with direct current component

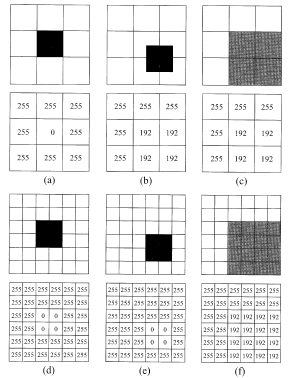
Another way of functions representation is the application of the Euler equation:

C:\Documents and Settings\khec\Desktop\chap3\23_files\Image232_6.gif

Therefore, the periodic function is obtained computing the module and argument as:

C:\Documents and Settings\khec\Desktop\chap3\23_files\Image232_7.gif

The reason of the Fourier transform is the simplification of the operation, manipulation with the principal characteristics of the signals in frequency domain, and for solution of the specific problems, which are difficult in time domain representation. One example is following: the digital camera captures the object, which occupies the area of exactly one pixel (or the set of the pixels). In this case representation of this object is simple and accurate. But usually the object takes only one part of the area defined by pixel. That is why the representation will be not correct (extension of the object size with the lost of contrast and generation of the weak boundaries)

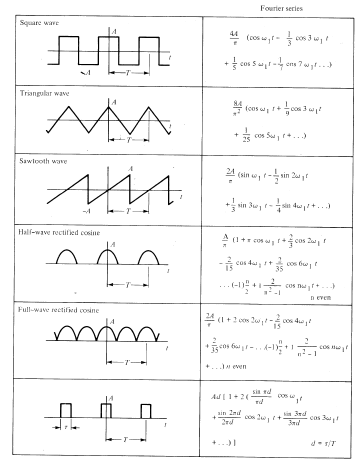


Digitizing the objects with respect of the their position

b), c) the object covers 1/3 part of the pixel's area and is represented

as 192 (3/4 of 255 of white)

Examples of periodic signals and their Fourier transform is shown in figure:



Common periodic signals and their Fourier transform

***(iii) Basic concepts of Fourier Series***

The Fourier series of the periodic function ***f(x)*** is written ***F(u)*** and describes the amount of each frequency term that be added together to make ***f(x)***.

C:\Documents and Settings\khec\Desktop\chap3\23_files\fo_Image1.gif

where ***i*** is root square of ***-1***. The use of the exponential notation relies on the mathematical identity (Euler's formula)

C:\Documents and Settings\khec\Desktop\chap3\23_files\fo_Image2.gif

One of the very important characteristics of this transform is that given ***F(u)***, it is possible to recover the spatial domain function ***f(x)*** in the same way, that comprise the forward and reverse Fourier transform.

C:\Documents and Settings\khec\Desktop\chap3\23_files\fo_Image3.gif

The ***f(x)*** is generally is a real function, such as a time-varying voltage or a spatially-varying image brightness. However, the transform function ***F(u)*** is generally complex, the sum of a real part ***Re*** and an imaginary part ***Im***.

***F(u) = Re(u) + iIm(u)***

It is usually more convenient to express this in polar rather than Cartesian form

C:\Documents and Settings\khec\Desktop\chap3\23_files\fo_Image4.gif

where ***|F(u)|*** is called the magnitude an ******is called the phase. The square of magnitude ***P(u) = | F(u)|2*** is commonly referred to as the power spectrum, or spectral density of ***f(x)***.

Integral from minus to plus infinity will in practice be reduced to a summation of terms of increasing frequency, limited by the finite spacing of the sampled points in the image. Therefore, the discrete Fourier transform is written

C:\Documents and Settings\khec\Desktop\chap3\23_files\fo_Image5.gif

where ***N*** is the number of sampled points along the function f(x), which are assumed to be uniformly spaced. Again, the reverse discrete transform is similar (but not identical; note the absence of the ***1/N*** term and the change is sign for the exponent).

C:\Documents and Settings\khec\Desktop\chap3\23_files\fo_Image6.gif

The values ***u*** from ***0*** to ***N-1*** represent the discrete frequency components added together to construct the function ***f(x)***

The summation is normally performed over terms up to one-half the dimension of the image (in pixels), since it requires a minimum of two pixels brightness values to define the highest frequency present. This limitation is described as Nyquist frequency. Because the summation has half as many terms as the width of the original image, but each term has a real and imaginary part, the total number of values produced by Fourier transform is the same as the number of pixels in the original image width, or the number of samples of a time-varying function.

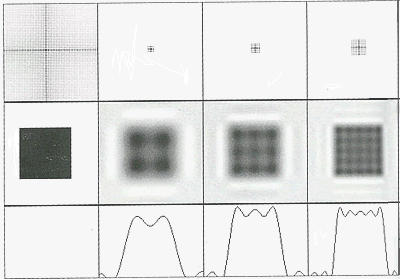
For extension to two- (three- and more dimensional functions transform) the ***f(x)*** is substitutes by ***f(x,y)*** and ***F(u)*** by ***F(u,v)***. Since the dimensions x,y,z are orthogonal, so are the ***u,v,w*** dimensions and transformation would be possible performed separately in each direction.  
For a two-dimensional image a one-dimensional transform on each horizontal line of the image is performed, producing an intermediate result with complex values for each point Then a second series of one-dimensional transforms can be performed on each vertical line, finally producing the desired two-dimensional transform.  
The one example of program fragment listed below shows how to compute the FFT of a function. On the input to the subroutine, ***F*** is the array of values to be transformed (usually the imaginary part of these complex numbers will be ***0***) and ***LN*** is the power of ***2*** (up to ***10*** for the maximum ***1024*** in this implementation). The transform is returned the same array ***F***. The first loop reorders the input data, the second performs the successive doubling that is a heart of the FFT method, and final loop normalizes the results.

Subroutine FFT(F,LN)

--------COMPLEX F(1024),U, W, T, CMPLX  
--------PI=3.14159265  
--------N=2\*\*LN  
--------NV2=N/2  
--------NM1=N-1  
--------J=1  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
--------DO 3 I=1, NM1  
-------- -------IF (I, GE,J) GOTO 1  
-------- -------T=F(J)  
-------- -------F(J)=F(I)  
-------- -------F(I)=T  
1-------- -----K=NV2  
2-------- -----IF (K, GE, J) GOTO 3  
-------- -------J=J-K  
-------- -------K=K/2  
-------- -------GOTO 2  
3-------- -----J=J+K  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
--------DO 5 L=1, LN  
-------- -------LE=2\*\*L  
-------- --------LE1=LE/2  
-------- -------U=(1.0, 0.0)  
-------- -------W=CMPLX (COS(PI/LE1), -SIN(PI/LE1))  
-------- -------DO 5 J=1, LE1  
-------- --------------DO 4 I=J, n, LE  
-------- ---------------------IP=I+LE1  
-------- ---------------------T=F(IP)\*U  
-------- ---------------------F(IP)=F(I)-T  
4-------- -------------------F(I)=F(I)+T  
5-------- ------------U=U\*W  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
--------DO 6 I=1MN  
6-------- -----F(I)=F(I)/FLOAT(N)  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
--------RETURN  
--------END

Applying this one-dimensional transform to each row and then each column of a two-dimensional image is not the fastest way to perform the calculation, but is the simplest one. The resulting transform of the original image into frequency space has complex values ate each pixel. This is difficult to display in any meaningful way. Therefore the display is based on only the magnitude of the value, ignoring the phase. If the square of the magnitude is used, this may be referred to as the image power spectrum, since different frequencies are represented at different distances from the origin, different directions represent different orientations in the original image, and the power at each location shows how much that frequency and orientation is present in the image. However, the power spectrum by itself cannot be used to restore the original image. The phase information is also needed, although it is rarely displayed and is usually difficult or impossible to interpret visually.

The example of Fourier transform applyed to the set of terms which construct the spep function is shown in the figure.



***(iv) Extension of Fourier Series, Fourier Transform***

Fourier Transform is the extension of the Fourier series to non periodic signals (functions).

In this case the coefficient is calculated by:

C:\Documents and Settings\khec\Desktop\chap3\23_files\Image232_10.gif

and function without period will be written taking into account that *0T0=2* :

C:\Documents and Settings\khec\Desktop\chap3\23_files\Image232_11.gif

Now the signals are not periodical, that means the period *T0* is from - infinity to + infinity and the coefficients are found:

C:\Documents and Settings\khec\Desktop\chap3\23_files\Image232_12.gif

***The one-dimensional discrete Fourier transform (DFT)*** and Inverse transform of the sequence *{f(x)=0,...N-1}* are defined as:

C:\Documents and Settings\khec\Desktop\chap3\23_files\Image232_13.gif

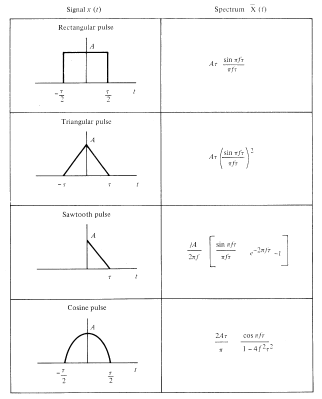
The previous equations are not scaled properly to be unitary transformations. In image processing it is more convenient to consider the *unitary DFT*, which is defined as:

C:\Documents and Settings\khec\Desktop\chap3\23_files\Image232_14.gif

***The two dimensional DFT*** and inverse transform of an *NxN* image are separable transforms defined as

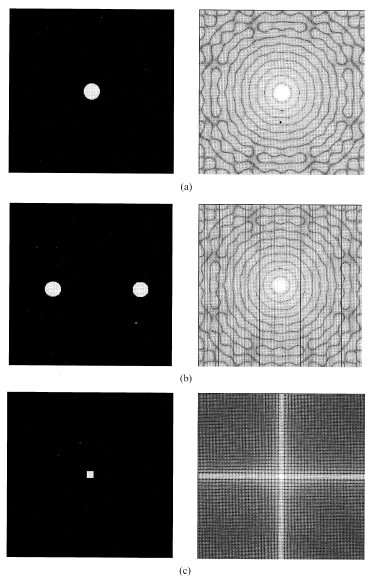
C:\Documents and Settings\khec\Desktop\chap3\23_files\Image232_15.gif

Examples of the non-periodic signals and their Fourier transform are shown in the following figure.



Some common non-periodic signals and their Fourier transform

Examples of the two dimensional Fourier transform are shown in following figure.



Fourier transform of a) a circle, b) two circles, c) quadrate

***(v) Properties of the Fourier transform***

**Translation**

If the image *f(x,y)* has a displacement, the new Fourier transform is defined as :

C:\Documents and Settings\khec\Desktop\chap3\23_files\Image232_18.gif

This property may be used for detecting the movement within image.

*Rotation*

It is possible to operate with function *f(x,y)* in Cartesian and its polar coordinates *f(r, )* :

*x=r cos y=r sin*

*u= cos v= sin*

Therefore, the Fourier transform *F( , )* is obtained from *f(r, ).* If there is a rotation by certain angle, the changes in Fourier transform are the similar:

*f(r, + ) <=> F( , + )*

*Separation ability*

Discrete Fourier transform may be expressed in another way comparing with standard definition, for example of two dimensional transform as:

C:\Documents and Settings\khec\Desktop\chap3\23_files\Image232_19.gif

It means that first, the one dimensional transform is applied for column and second, for rows of *NxN* image matrix

*Distributive property*

The Fourier transform of the two functions sum is equal to sum of Fourier transforms of these two functions:

*F(f(x,y)+g(x,y))=F(f(x,y))+F(g(x,y))*

*Magnification*

The following expressions are correct using the constants with Fourier transform:

C:\Documents and Settings\khec\Desktop\chap3\23_files\Image232_20.gif

*Average value*

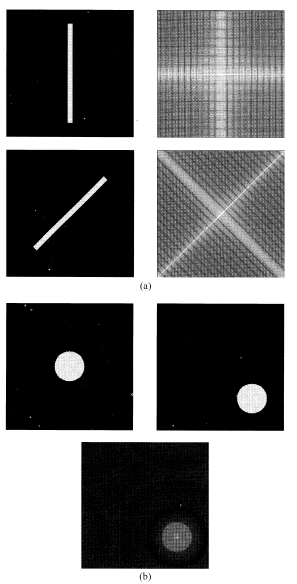
Important value, frequently used in image processing, is function average value of the neighborhood of the one particular point applying the equation:

C:\Documents and Settings\khec\Desktop\chap3\23_files\Image232_21.gif

If the selected point is the origin of the coordinates with *x* and *y* at *(0,0)* , therefore the average value is computed as :

C:\Documents and Settings\khec\Desktop\chap3\23_files\Image232_22.gif

Examples of the Fourier transform properties are shown in the following figure.



Some properties of the Fourier transform. a) rotation, b) translation

***(vi) Fast Fourier Transform***

The principal objective of the FFT is the reduction of number of operations necessary for transform from time to frequency domain. The restriction for FFT is that the dimension of the transformed discrete set must be power of 2. That is why

C:\Documents and Settings\khec\Desktop\chap3\23_files\Image232_24.gifwhere C:\Documents and Settings\khec\Desktop\chap3\23_files\Image232_25.gif

Now it is possible to rewrite

C:\Documents and Settings\khec\Desktop\chap3\23_files\Image232_26.gif

and according the definition

C:\Documents and Settings\khec\Desktop\chap3\23_files\Image232_27.gif

the FFT is given as

C:\Documents and Settings\khec\Desktop\chap3\23_files\Image232_28.gif

Using the definitions as

C:\Documents and Settings\khec\Desktop\chap3\23_files\Image232_29.gif

The FFT finally is defined as

C:\Documents and Settings\khec\Desktop\chap3\23_files\Image232_30.gif

Taking into account that the

C:\Documents and Settings\khec\Desktop\chap3\23_files\Image232_31.gif

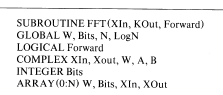
the following equation is obtained

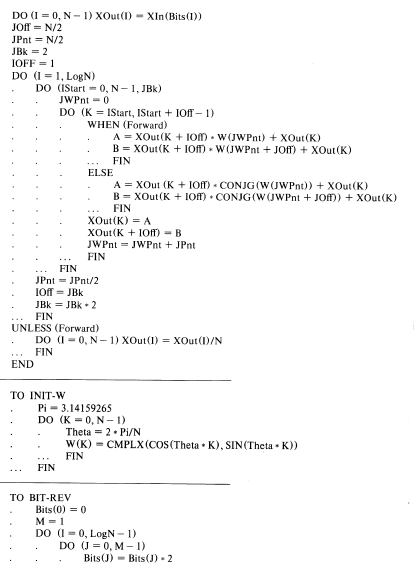
C:\Documents and Settings\khec\Desktop\chap3\23_files\Image232_32.gif

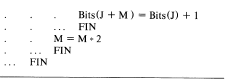
That means the computing of the first half of the points of array *N* for *F(u)* is obtained with the first expression for *Feven* and the second half - with the expression for *Fodd* . The FFT is applied first for columns and than for rows.

*Another example of the program for FFT*

The following routine computes the discrete Fourier transform of a one-dimensional complex array *XIn* of the length *N=2 Power (logN*) and produces the one-dimensional complex array *Xout*. It uses an array W of the *N* complex *Nth* roots of unity, computed as shown, and an array *Bits* containing a bit-reversal table of length *N. N, LogN, W*, and *Bits* are all global to the subroutine as written. If the logical variable *Forward* is TRUE, the FFT is performed, if *Forward* is FALSE, the inverse FFT is performed.







***(vii) Relationship between the transformations***

The relationship between convolution and Fourier transform is

*f(x,y)\*g(x,y) <=> F(u,v)· G(u,v)*

*f(x,y)· g(x,y) <=>F(u,v)\*G(u,v)*

The relationship between correlation and Fourier transform is

*f(x,y)° g(x,y) <=>F\*(u,v)· G(u,v)*

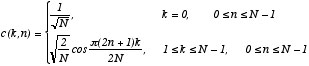
*f\*(x,y)· g(x,y) <=>F(u,v)° G(u,v)*

**2. 4. Other Mathematical Transformations**

There is a lot of different transforms which may be used for specific operations and the principal objectives of which is the reduction of the complexity during the operations in the new domain. For example the cosine, sine, Hadamard, Haar, Slant Karhunen-Loeve, Fast KL, sinusoidal, singular value decomposition transforms and others.

**2.4.1. The cosine transform**

The *NxN* cosine transform matrix ***C*={c(k,n)}**, also called discrete cosine transform (DCT), is defined as

[](http://ict.udlap.mx/people/oleg/docencia/IMAGENES/image24_1.gif)

The one-dimensional DCT of a sequence *{u(n), 0  n  N-1}* is defined as

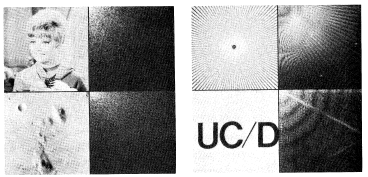
C:\Documents and Settings\khec\Desktop\chap3\25_files\Image24_3.gif

where C:\Documents and Settings\khec\Desktop\chap3\25_files\Image24_4.giffor *1  k  N-1*

The inverse transform is given by

C:\Documents and Settings\khec\Desktop\chap3\25_files\Image24_5.gif

The two-dimensional cosine pair is obtained by using the separable property of the transforms (for columns and rows). The cosine transform is the fast transform where the vector of *N* elements can be processed in *N log2N* operations via an *N*-point FFT. The following figure shows the results of the DCT.



a) Cosine transform examples of monochrome images

b) CDT examples of binary images

**2.4.2. The sine transform**

The *NxN* sine transform matrix  *={ (k,n)},* also called the discrete sine transform DST, is defined as

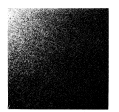
C:\Documents and Settings\khec\Desktop\chap3\25_files\Image24_7.gif

The sine transform pair of one dimensional sequences is defined as

C:\Documents and Settings\khec\Desktop\chap3\25_files\Image24_8.gif

C:\Documents and Settings\khec\Desktop\chap3\25_files\Image24_9.gif

Like in case of DCT the two-dimensional transforms can be obtained. In figure the DST is shown for the monochrome input image with portrait (input image of the CDT.



Sine transform for standard input image used in DCT

**2.4.3.Summary of Image Transforms**

