**4. Image Analysis and Computer Vision**

The ultimate aim in a large number of image processing applications is to extract important features from image data , from which a description, interpretation, or understanding of a scene can be provides by the machine.

The possible examples of computer vision applications and the related problems of the image understanding are shown in the following table.

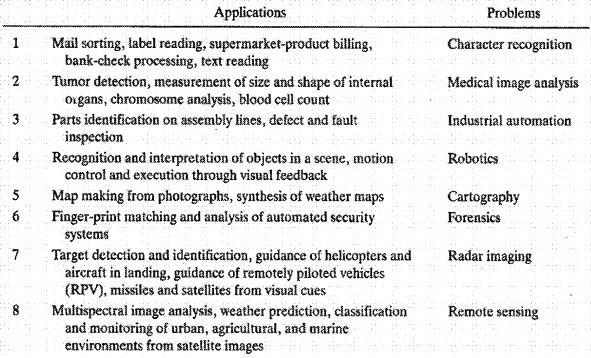
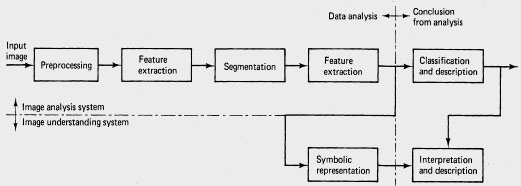
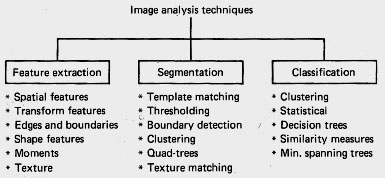
****

Image analysis basically involves the study of features extractions, segmentation, and classification techniques. The basic computer system and the image analysis techniques may be presented as:



A basic computer vision system



**(vii) Resume of the edge enhancement techniques**

The enhancement methods presented in previous sections are very basic and are found in the standard processing package of software routines supplied with most commercial vision systems. If the exact nature of how the image degrading function is known, it is possible to predict in advance how different operators will affect the image.

In practice, it may be necessary to experiment with various filters to determine which produce the best results.

Convolution masks

*New pixel value (3x3 mask)= f1,1 x p1.1+f2,1 x p2.1 +...+ f3,3 x p3.3*

fi,,j - coefficient of the mask in *i-th*row and *j-th* column,

*pi.j*- gray scale of the pixel in original image in *i-th* row and *j-th* column

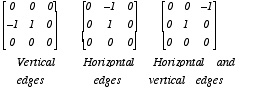
*1. Low-pass filters*

* All the coefficients must be positive
* The sum of all the coefficients must equal one. (if the sum will be more than one , amplification of magnitude of the pixels will result : light image, less than one - attenuation : dark image)

http://ict.udlap.mx/people/oleg/docencia/IMAGENES/chapter4/Image412_1.gif

*2. Shift and difference filters*

* Coefficients added to zero



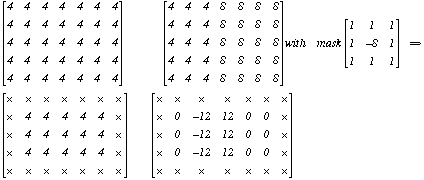
*3. High-pass filters*

* The coefficients can be positive or negative
* The sum of the coefficients is zero (DC component will be suppressed. It produces the lightening effect on background).

http://ict.udlap.mx/people/oleg/docencia/IMAGENES/chapter4/Image412_3.gif

An edge in an output image matrix will appear as a sharp change in the values of adjacent pixels. Thus the data in the edge region is modified and the edge effect is amplified as it shown in following image:



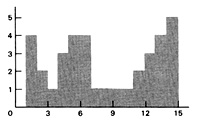
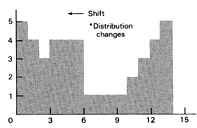


*4. High-pass filters with DC bias*

* The sum of all the coefficients equal one
* Elimination of the constant value bias ( the resultant matrix will have a line of low and high pixels values.

http://ict.udlap.mx/people/oleg/docencia/IMAGENES/chapter4/Image412_5.gif

Effect of the high-pass filter can be detected on the histogram.

Pixels values for image with Pixels values of image after

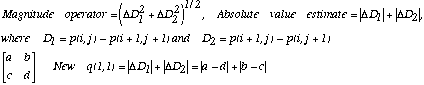
Background bias high-pass filter

*5. Gradient-directional operators*

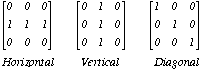
* Coefficients added to zero

http://ict.udlap.mx/people/oleg/docencia/IMAGENES/chapter4/Image412_7.gif

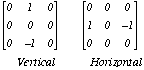
* Robert's operators use the diagonal derivatives to estimate gradient in the point
* The magnitude operator is equal to the square root of the sum of the squares of the two diagonal differences *D1* and *D2:*



*6. Blurs*



*7. Difference filters*



*8. Vertical differentiation filters*

* Take absolute value of difference filter (7)

*9. Horizontal differencing and vertical smoothing filters*

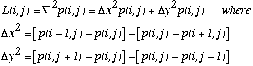
*http://ict.udlap.mx/people/oleg/docencia/IMAGENES/chapter4/Image412_11.gif*

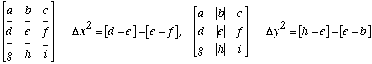
*10. Laplacian filters*

* The coefficients added to zero

http://ict.udlap.mx/people/oleg/docencia/IMAGENES/chapter4/Image412_12.gif

* Approximation by the difference operator. Pixel intensity is calculated simply as

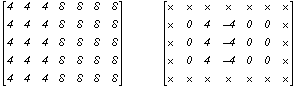
**

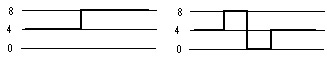
**

The collection of terms will result in the following for Laplacian operator

http://ict.udlap.mx/people/oleg/docencia/IMAGENES/chapter4/Image412_15.gif

* The Laplacian operator computes the difference between the gray level of the center pixel and the average of the gray levels of the four adjacent pixels in the horizontal and vertical directions.
* The presence of the positive (negative) edge is represented after application of the Laplacian operator by positive-negative (negative-positive) pulses

------



*11. Bright region expansion*

http://ict.udlap.mx/people/oleg/docencia/IMAGENES/chapter4/Image412_17.gif

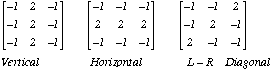
*12. Medium filter*

* Reduce camera noise

http://ict.udlap.mx/people/oleg/docencia/IMAGENES/chapter4/Image412_18.gif

*13. Enhance line segment*

* Coefficients added to zero



*14. Edge detector*

http://ict.udlap.mx/people/oleg/docencia/IMAGENES/chapter4/Image412_20.gif

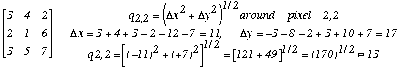
* Sobel edge detector compute magnitude of the gradient in point but does not use the value at the point itself in the calculation.
* The value of the pixel is given by:

http://ict.udlap.mx/people/oleg/docencia/IMAGENES/chapter4/Image412_21.gif

* Convolution masks for Sobel operator:

http://ict.udlap.mx/people/oleg/docencia/IMAGENES/chapter4/Image412_22.gif

* Example for Sobel filter computing.

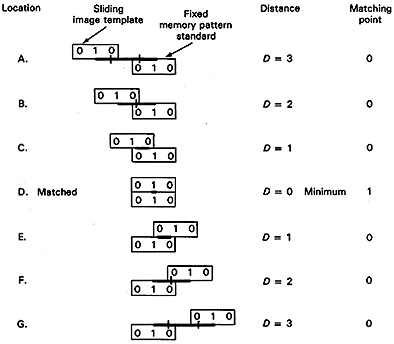


*15. Line detector*

*http://ict.udlap.mx/people/oleg/docencia/IMAGENES/chapter4/Image412_24.gif*

16. Template matching for edge detection

Cross correlation is determined by calculating the sum of the distance between corresponding pattern points in the two images (one is the original image and another is predefined template. The following figure shows the pattern matching for *3x1*image: the distance D and the number of matching corresponding reference elements is determined for all locations. The matching of *3x1*pattern in the *x* direction can be achieved by shifting the template from right to left one pixel at a time and calculating the sum of the distances between key registration points. In the following example the pixel containing *1* is selected as the registration point.



Summary of results of the template shifting routine for the seven locations:

Matching point\_\_\_\_\_\_ 0 0 0 1 0 0 0

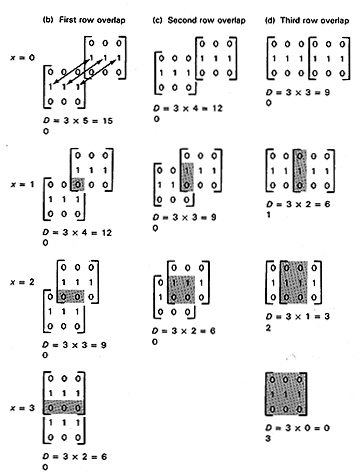
Location\_\_\_\_\_\_\_\_\_\_ A B C D E F G

The best fit is indicated to be location D where there is the only matching point condition of all seven locations.

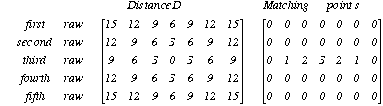
The presence of the noise or distortion will result in the cross-correlation factor not going to zero, but the minimum value is the best estimate of optimum location of a match.

The method may be extended to a *3x3* image pattern as it shown in the following figure. The template is first moved in the *x* direction and then the process is repeated after stepping one pixel in the*y* direction. The output is the expanded matrix containing the number of matching points.

pattern:http://ict.udlap.mx/people/oleg/docencia/IMAGENES/chapter4/Image412_42.gif template: http://ict.udlap.mx/people/oleg/docencia/IMAGENES/chapter4/Image412_42.gifCriteria: distance=*D* units, matching elements

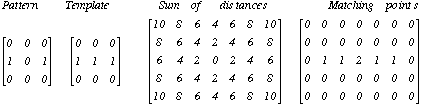


Resulting pattern for distance between the three reference points as *x=0* to *6*



Thus, the pattern location is fixed and template is moved horizontally one unit at a time, and the process repeated for each of the possible rows analyzing the best matching position indicated by the lowest *D* and highest mating point location is the third row, forth position.

The presence of the noise (element in pixel (2,2)) in the next example will not affect significantly the edge detection, only the number of the matching points is reduced



**4.1. Spatial feature extraction**

Spatial features of an object may be characterized by its gray levels, their joint probability distributions, spatial distribution and so on.

***(i) Amplitude features***

The simplest and perhaps the most useful features of an object are the amplitudes of its physical properties, such as reflectivity, transmissivity, tristimulus values (color), or multispectral response.

Amplitude features can be extracted easily by intensity window slicing or by the more general point transformations discussed in Features Extraction Chapter.

***(ii) Histogram Features***

Histogram features are based on the histogram of a region of the image. Let *u* be a random variable representing a gray level in a given region of the image. Define

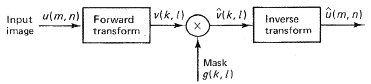
**http://ict.udlap.mx/people/oleg/docencia/IMAGENES/chapter4/Image41_8.gif**

Common features of *pu(x)* are its moments, entropy, dispersion, mean square value or average energy, skewness and so on.

All operations mentioned in the histogram modification chapter are useful for histogram features extraction.

***(iii) Transform features***

Image transforms provide the frequency domain information in the data. Transform features are extracted by zonal-filtering the image in the selected transform space.

******

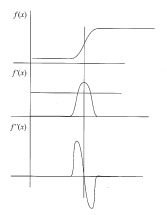
Transform feature extraction

The zonal filter, also called the feature mask, is simply a slit or an aperture. The Fourier transform of the different masks produce the well predefined shapes which are used for edge and boundary detection. Angular slits can be used for detection of orientation. A combination of an angular slit with a bandlimited low-pass, band-pass, high-pass filters can be used for discriminating periodic or quasiperiodic textures. Other transforms, such as Haar and Hadamard, are also potentially useful for feature extraction

***(iv) Edge Detection***

A problem of fundamental importance in image analysis is edge detection. Edges characterize object boundaries and therefore useful for segmentation, registration, and identification of objects in scenes. Edge points can be thought of as pixel location, of abrupt gray level change.

Basically, the edge detection techniques are based on derivative operators. There are two possibilities: to apply the first derivative (gradient) or second derivative (Laplacian). In the first case the high picks of the derivative of function are found and in the second one - the zero crossing (passing from high to low level) is found.

******

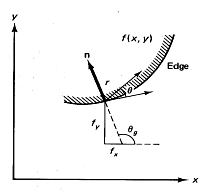
Edge detection by applying the first and second derivatives

It is reasonable to define edge points in binary images as black pixels with at least one white nearest neighbor, that is pixel locations *(m,n)* such that *u(m,n)=0* and *g(m,n)=1*, where

***http://ict.udlap.mx/people/oleg/docencia/IMAGENES/chapter4/Image41_9.gif***

here Å denotes the logical exclusive-OR operation. For continuous image *f(x,y)* its derivative assumes a local maximum in the direction of the edge. Therefore, one edge detection technique is to measure the gradient of *f* along*r*in a direction q , that is

***http://ict.udlap.mx/people/oleg/docencia/IMAGENES/chapter4/Image41_10.gif***

******

Gradient of *f(x,y)* along *r* direction

The maximum value of http://ict.udlap.mx/people/oleg/docencia/IMAGENES/chapter4/image41_11.jpg. This gives

http://ict.udlap.mx/people/oleg/docencia/IMAGENES/chapter4/Image41_12.gif

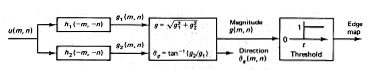
where*g*is the direction of the edge. Based on these concepts, two types of edge detection operators have been introduced: gradient operator and compass operator. For digital images these operators, also called masks, represent finite-difference approximations of either the orthogonal gradients*fx, fy*or the directional gradient d*f/dr* and can be calculated by correlation with specific masks.

Let ***H*** denote a *pxp* mask and define, for an arbitrary image ***U***, their inner product at location *(m,n)* as the correlation

http://ict.udlap.mx/people/oleg/docencia/IMAGENES/chapter4/Image41_13.gif

*1. Gradient Operators*

Gradient operators are represented by a pair of masks *H1, H2*, which measure the gradient of the image *u(m,n)* in two orthogonal directions as it shown in figure



Edge detection via gradient operators

Defining the bi-directional gradients

http://ict.udlap.mx/people/oleg/docencia/IMAGENES/chapter4/Image41_14.gif

the gradient vector magnitude and direction are given by

http://ict.udlap.mx/people/oleg/docencia/IMAGENES/chapter4/Image41_15.gif

Often for computation cost reduction, the magnitude gradient is calculated as

http://ict.udlap.mx/people/oleg/docencia/IMAGENES/chapter4/Image41_16.gif

rather than as in equation with square root. This calculation is easier to perform and is preferred especially when implemented in digital hardware.

As the images are not continuous signals the gradient is done by

http://ict.udlap.mx/people/oleg/docencia/IMAGENES/chapter4/Image41_17.gif

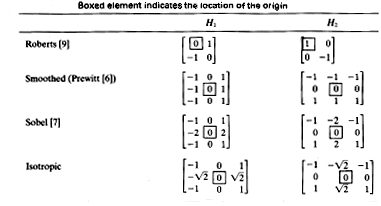
In the digital form it may be represented by masks:

http://ict.udlap.mx/people/oleg/docencia/IMAGENES/chapter4/Image41_18.gif

But in practice, these masks are not used because they are too sensible to noise due to operations on level of the one pixel. There are some developed well-known operators which are not so sensitive to noise.

Some common gradient operators.

Boxed element indicates the location of the origin



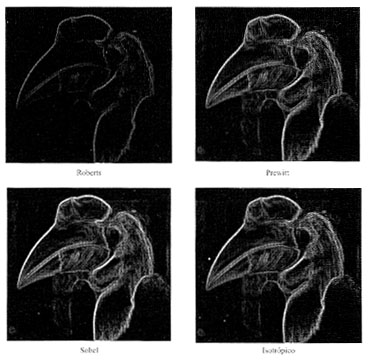
The Prewitt (Smoothed), Sobel, and Isotropic (Frei-Chen) operators compute horizontal and vertical differences of local sums. This reduces the effect of noise in the data. The pixel location *(m,n)*is declared an edge location if *g(m,n)* exceeds some threshold *t*. The locations of edge points constitute an edge map  *(m,n)* which is defined as

http://ict.udlap.mx/people/oleg/docencia/IMAGENES/chapter4/Image41_19.gif

The edge map gives the necessary data for tracing the object boundaries in an image. Typically, *t*may be selected using the accumulative histogram of *g(m,n)* so that 5 to 10 % of pixels with largest gradients are declared as edges.

The Prewitt operator detects better the vertical borders, Sobel does it good in diagonals, Isotropic operator is the is an equilibration between two previous ones.





Comparison between distinct border detection operators

**4.2 Boundary Extraction**

Boundaries are linked edges that characterize the shape of an object. They are useful in computation of geometry features such as size or orientation. Usually the following methods are used for boundary extraction:

1. Contour following. This hill-climbing technique works best with good image data.

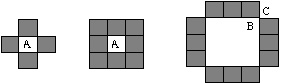
2. Edge linking as heuristic graph searching. This method represents the image of edge elements as a graph. Thus the boundary is a path through a graph and is generally applicable.

3. Dynamic programming. This method is also very general. It uses a mathematical formulation of the globally best boundary and can find boundaries in noisy image.

4. The Hough transform. This is elegant technique is used for boundary detection which shape can be described in an analytical or tabular form.

*Connectivity*

Conceptually, boundaries can be found by tracing the connected edges. On rectangular grid a pixel is four- or eight-connected when it has the same properties as one of its nearest four or eight neighbors.

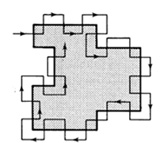


Connectivity on a rectangular grid. Pixel A has 4 and 8 connected neighbors.

Under the four-connectivity all the segments (of three pixels in line) would be classified as disjoint, although they are perceived to form a connected ring. Under the eight-connectivity these segments are connected, but inside hole (for example pixel B) is also connected to outside - pixel C (connectivity paradox). To avoid it the eight-connectivity is considered for object and four-connectivity - for background. An alternative is to use triangular or hexagonal grids where three- or six-connectedness can be defined.

**(i) Contour following**

The contour-following algorithms trace boundaries by ordering successive edge points. A simple algorithm for tracing closed boundaries in binary images is shown in figure.



Finding the boundary in a binary image.

This can be done simply by a procedure that functions like Papert's turtle:

* Scan the image until a region pixel is encountered
* If it is a region pixel, turn left and step, else turn right and step
* Terminate upon return to the starting pixel.

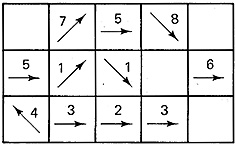
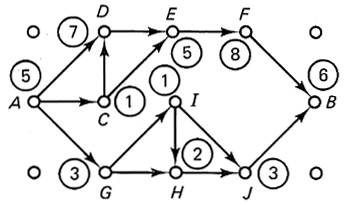
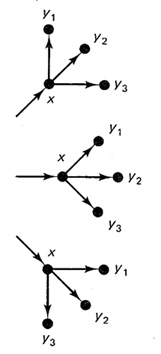
This procedure requires the region to be four-connected for a consistent boundary. Parts of the eight-connected regions can be missed. Also, some bookkeeping is necessary to generate an exact sequence of boundary pixels without duplication.

This method may be extended for gray scale images by searching for edges in the 45° to 135° direction from the direction of the gradient to move from the inside to the outside of the boundary and vise-versa.

**(ii) Edge linking as heuristic graph searching**

A boundary can also be viewed as a pass through a graph formed by linking the edge segments together. Linkage rules give the procedure for connecting the edge elements. If each pass has its weight or cost, the lowest-cost path between two nodes of weighted graph is found.

Assume *3x5* array of edges in image where a gradient operator is applied to this gray level image, creating the magnitude image *g(x)* and direction image (tangential contour direction) *(x)* (! the contour direction is *90°* to the gradient direction) as nodes in a graph, each with the weighting factor *g(x)* as it shown in following figure.

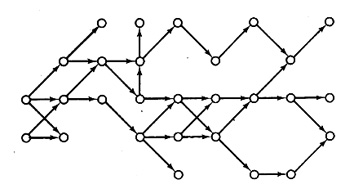
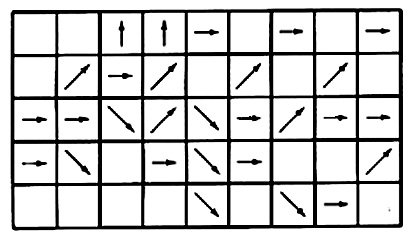
  

gradient magnitudes *g(x)* --------linkage rules ----------------graph interpretation

with contour directions *(x)*

Heuristic graph search method for boundary extraction

Another example of the linkage rules application for gradient of image is shown as:



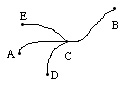
Interpreting a gradient image as a graph.

The pixel *x* is considered to be linked to *y* if the latter is one of the three eight-connected neighbors (*y1, y2, y3* in figure) in front of the contour direction and if |*(x)-(y)|<90°* . According the these linkage rules the graph is obtained [heuristic search by Nilsson, Martelli].

As an example, *S(xk)* is the sum of edge gradient magnitudes along the path from A to *xk*. At A, the successor nodes a D,C, and G with *S(D)=12, S(C)=6, and S(G)=8*. Therefore the node D is selected , and C and G are discarded. From here on nodes E,F, and B provide the remaining path (boundary path is ADEFB).

**(iii) Edge following as dynamic programming**

Dynamic programming is a method of finding the global optimum of multistage processes. It is based on Bellman and Dreyfus' principal of optimality, which states that *the optimum path between two given points is also optimum between any two points lying on the path*. Thus if C is a point on the optimum path between A and B Then the segment CB is the optimum path from C to B, *no matter how one arrives at C.*

Bellman's principle of optimality: 

To apply this idea to the boundary extraction, suppose the edge map has been converted into a forward-connected graph of N stages and the evaluation function may be defined as:

C:\Documents and Settings\khec\Desktop\chap3\image_42_IS548_files\image42_9.jpg

Here *xk, k=1,..., N* represents the nodes (that is, the vector of edge pixel location) in the *k*-th stage of the graph, *d(x,y)* is the distance between two nodes *x* and *y, |g(xk)|, (xk)* are the gradient magnitude and angle, respectively, at the node *xk*, and ** and ** are nonnegative parameters. The optimum boundary *(xN, N)* is given by connecting the nodes *x'k,* *k=1,..., N* so that *S(x'1, x'2, x'N, N)* is maximum.

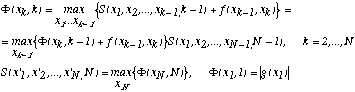
*(xN,N)=max {S(x'1, x'2, x'N, N)}*

Using the definition of *S(x'1, x'2, x'N, N)* the following equation is obtained:

C:\Documents and Settings\khec\Desktop\chap3\image_42_IS548_files\Image42_1.gif

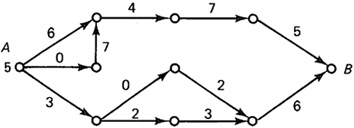
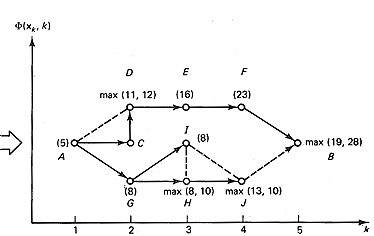
where *f(xN-1, xN)* represents the term in the brackets.

If *N=k* than the optimum boundary is computed as :



This procedure is remarkable in that the global optimization of *S(x1, x2, xN, N)* has been reduced to N stages of two variable optimizations and the total number of search operations is (N-1)(L(power2)-1)+(L-1) instead of *L(powerN)-1* exhaustive searches required for direct maximization of *S(x1, x2, xN, N)* when L and N are large.

Ex. Consider the gradient image in heuristic graph searching chapter and the same linkage rules, taking that * = 4/, =0* may be obtained the following graph with the values of various segments connecting different nodes.

**  **

Path with values *(xk,k)* at the various stages with optimal path (solid line)

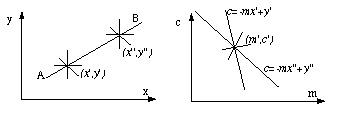
Dynamic programming for optimal boundary extraction

Specially, the *N=5* and *(A,1)=5.* For k=2, *(D,2)=max(11,12)=12,* which means in arriving at D, the path ACD is chosen. Proceeding in this manner some of candidate paths (shown by dotted lines) are eliminated. At *k=4*, only two paths are acceptable, namely, ACDEF and AGHJ. At *k=5*, the path JB is eliminated, giving the optimal boundary as ACDEFB.

**(iv) The Hough transform for curve detection**

The classical Hough technique for curve detection is applicable if little is known about the location of a boundary, but its shape can be described as a parametric curve (e.g. a straight line or conic) Its main advantages ate that it is relatively unaffected by graphs in curve and by noise.

To introduce the method, suppose that after some processes the image point have been selected that have a high likelihood of being on linear boundaries. The Hough technique organizes these points into straight lines. If the line *y=mx+c* paths through the point *(x' , y')* there are more lines satisfying *y'=mx'+ c* which paths over this point. If *(x',y')* are fixed the m-c space (parameter space) defines all this lines:



A line in image space and in parameter space

Repeating this reasoning, a second point *(x'',y'')* will also have an associated line in parameter space and, furthermore, these lines will intersect at the point *(m',c')* which corresponds to the line AB connecting these points. In fact, all points on the line AB will yield lines in parameter space which intersect at all the points *(m',c')* as shown above. This relation between image space *(x,y)* and parameter space *(m,c)* suggests the following Line Detection with the Hough algorithm:

1. Quantize parameter space between appropriate maximum and minimum values for *c* and *m*.

2. Form an accumulator array A *(c,m)* whose elements are initially zero

3. For each point *(x,y)* in a gradient image such that the strength of the gradient exceeds some thresholds, increment all points in the accumulator array along the appropriate line, i.e., *A (c, m):= A (c, m)+1*

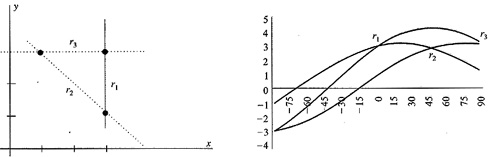
for *m* and *c* satisfying *c= - mx+y* within the limits of the digitalization.

4. Local maximum in the accumulator array now correspond to collinear points in the image array. The values of the accumulator array provide a measure of the number of points on the line.

This approach is known as the Hough technique. Since m may be infinite in the slope-intercept equation, a better parameterization of the line is

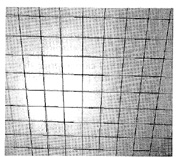
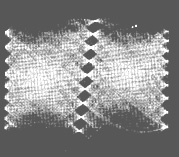
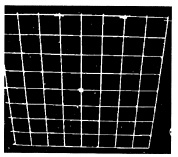
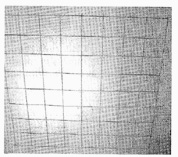
*x cos q + y sin q = r*

For any two points in *x* - space, the intersection of its sine curves will be in the set of values *(ri,* q*i)* which defines the line.



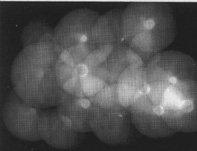
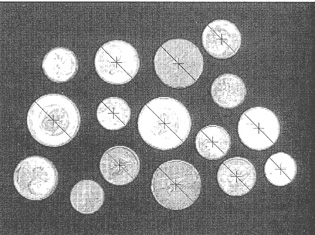
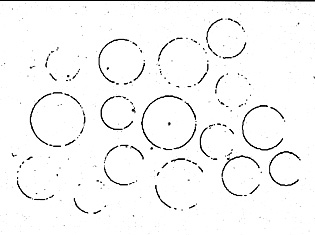
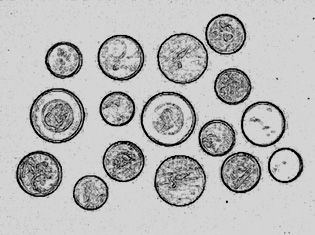
The Hough transform

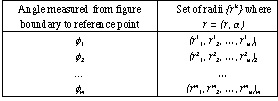
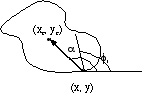
Finally, these values which are repeated more frequently are coincided with lines which have major number of pixels.

******

The Hough transform . a) original image, b) gradient (borders), c) lines in *r-*q *space, d) detected lines*

This technique works for either straightforward or any curve. The equation for circumference is:  
**C:\Documents and Settings\khec\Desktop\chap3\image_42_IS548_files\image42_20.jpg**  
and now has three parameters *(x0, y0, R)* instead of two for lines. In Hough transform the set of all possible circumferences are computed and the most frequently occurred are selected.

  
Hough transform: gradient, points of the border, x,y,R projection on X,Y plane, and  
detection the circumferences with the greater number of points.  
  
Suppose that the object has no simple analytic form, but has a particular silhouette. The basic strategy of the Hough technique is to compute the possible loci of reference points in parameter space from edge point data in image space and increment the parameter points in an accumulator array. The following figure shows the relevant geometry and table shows the form of the R-table(the set of locations of the reference point from boundary points indexed by the gradient angle).



Geometry used to form the R-table Table with locations indexed by gradient

The reference point coordinates *(xc, yc)* are the only parameters (assuming that rotation and scaling have been fixed). Thus an edge point *(x,y)* with gradient orientation ** constrains the possible reference points to be at

*{x+r1()cos[1()], y+r1()sin[1()]}* , and so on.

The generalized Hough algorithm now is described as:

1. Make a table for the shape to be located.

2. Form an accumulator array of possible reference points A(xc min: xc max, yc min: yc max) initialized to zero

3. For each edge point do the following :

3.1 Compute *f(x)*

3.2 Calculate the positive centers; that is, for each table entry for **, compute

*xc=x+r1()cos[1()]*

*yc=y+r1()sin[1()]*

3.3 Increment the accumulator array

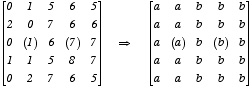
*A(xc, yc):=A(xc, yc)+1*

3.4 Possible locations for the shape are given by maximum in array A.

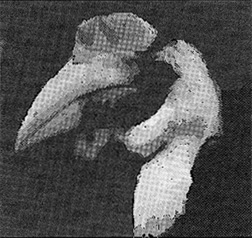
Thus, if the shape we want to detect in image is predefined and described by R-table, therefore this shape will be looked for in the image and presented by set of the points with the maximum of A*(xc, yc)* (as the principal corners of the object). The method is useful for aerial image analysis.

**4.3 Region growing**

An image region may correspond to a world object or a meaningful part of one. The region growing is based on the selection of a set of the initial pixels (may be called seed) and aggregation to these pixels other ones with the same properties (i.e., if they satisfy *|f(x,y)-f(sxi, syi| < Threshold*). For example, if the two pixels as seeds (*1* and *7*) are selected and the difference between these pixels and image pixels is taken less or equal threshold=3, two regions are obtained.



Applying this approach for the region growing the following result may be obtained:

****

Result of the region growing approach application with distinct seeds and thresholds.

The goal of the region growing is to use image characteristics to map individual pixels in an input image to set of pixels called region. Now the problem is how to find these seed pixels. The simplest way is to use the histogram and select the pixels which have the highest gray level, or to use analysis of the boundary for region describing and the region-finding and line-finding techniques are interconvertible and can cooperate to produce a more reliable segmentation.

The early techniques for region estimation may be classified as:

(i) Local techniques . Pixels are placed in a region on the basis of their properties or the properties of their close neighbors.

(ii) Global techniques. Pixels are grouped into regions on the basis of properties of large number of pixels distributed through the image.

(iii) Splitting and merging techniques.The foregoing techniques are related to individual pixels or sets of pixels. State place techniques merge or split regions using graph structures to represent the regions and boundaries. Both the local and global merging and splitting criteria can be used.

Used definitions:

Region *Rk* are considered to be sets of points with the following properties:

*\* xi* in a region *R* is connected to *xj* if there is a sequence *{xi, , xj}* such that *xk* and *xk+1* are connected and all the points are in *R* .

*\* R* is connected region if the set of points *x* in *R* has the property that every pair of points is connected.

*\* C:\Documents and Settings\khec\Desktop\chap3\43_files\image43_33.jpg*

\* C:\Documents and Settings\khec\Desktop\chap3\43_files\image43_1.gif

A set of regions satisfying these properties is known as a partition. In segmentation algorithms, each region often is a unique, homogeneous area. That is, for Boolean function H(R) that measures region homogeneity,

C:\Documents and Settings\khec\Desktop\chap3\43_files\image43_2.gif

**(i) Local techniques.**

The counterpart to the edge tracker for binary images is the blob-coloring (region to be grown) algorithm.

Given a binary image containing four-connected blobs of 1's on a background of 0's, the objective is to "color each blob"; that is, assign each blob a different label. To do this, scan the image from left to right and top to bottom with a special L-shaped template

**C:\Documents and Settings\khec\Desktop\chap3\43_files\image43_0.jpg**

L-shaped template for blob coloring

*The blob coloring algorithm*

Let the initial color*, k=1*. Scan the image from left to right and top to bottom.

If *f(xC) =0* then continue

else

begin

if *(f(xU) =1 and f(xL) =0)*

then color *(xC): =* color *(xU)*

if *(f(xL) =1 and f(xU) =0)*

then color *(xC): =* color *(xL)*

if *(f(xL) =1 and f(xU) =1)*

then begin

color *(xC): =* color *(xL)*

color *(xL)* is equivalent tocolor *(xU)*

*end*

comment: two colors are equivalent.

if *(f(xL) =0 and f(xU) =0)*

then color *(xC): = k; k:=k+1*

comment: new color

end

After one complete scan of the image the color equivalences can be used to assure that each object has only one color.

This binary image algorithm can be used as a simple region-grower for gray-level images with the following modifications. If in a gray-level image *f(xC)* is approximately equal to *f(xU)*, assign *xC* to the same region (blob) as *xU* . This is equivalent to the condition *f(xC)* = *f(xU) = 1* in previous algorithm.

**(ii) Global techniques**

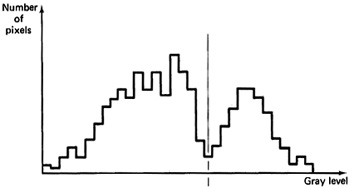
*1. Region growing via thresholding*

This approach assumes an object-background image and picks a threshold that divides the image pixels into either object or background:

*x* is a part of the object if *f(x) > T*

otherwise it is part of the background

The best way to pick the threshold *T* is search the histogram of gray levels, assuming it is bimodal



Threshold determination from gray-level histogram

But this way is not simple because the histogram is not smooth function. The modifications of the threshold approach to reduse the difficulty are:

high-pass filter the image to deemphasize the low-frequency background variations and then try the original technique.

use a spatially varying threshold method (divide the image into rectangular subimages and compute the threshold for each subimage).

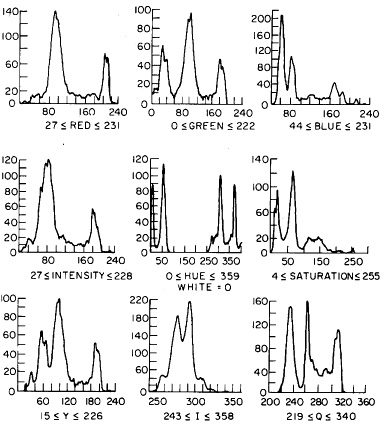
*2. Thresholding in multidimensional space*

This is the variation of the thresholding paradigm for color images where such color parameters are found for which thresholding does the desired segmentation (for example using RGB color components, or the intensity, saturation, and hue components, or the NTSC Y,I,Q components).

Region growing algorithm via recursive splitting

a) Consider the entire image as a region and compute histograms for each of the picture vector components.

b) Apply a peak-finding test to each histogram. If at least one component passes the test, pick the component with the most significant peak and determine two thresholds, one either side of the pick . Use this thresholds to divide the region into subregions.



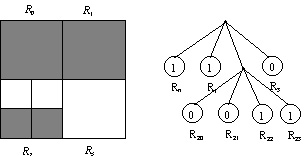
Pick detection and threshold determination

c) Each subregion may have a "noisy" boundary, so the binary representation of the image achieved by thresholding is smoothed so that only a single connected subregion remains.

d) Repeat steps 1 through 3 for each subregion until no new subregions are created (no histograms have significant picks)

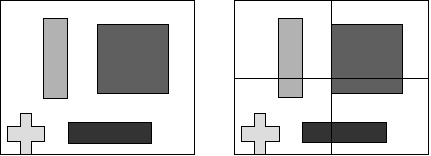
**(iii) Splitting and merging techniques**

Split and merge (division and union) of the regions typically, is based on the quadtree structure which permit to represent the objects with distinct resolutions.



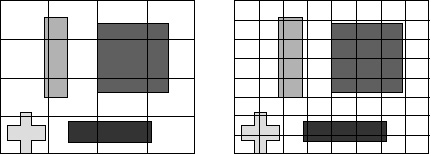
Quadtree structure

Each node has four outgoing arcs; they are clear define the structure on every level. Using the mentioned approach the original image of the next example may be divided (split into four subimages if original one does not satisfy the set of the common for all image).

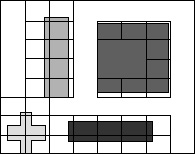
****

The average or deviation of the image may be used as criteria of the satisfaction of region characteristics. C:\Documents and Settings\khec\Desktop\chap3\43_files\image43_3.gif

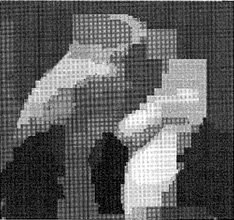
If the deviation is more than predefined value, it means that there is great variation in gray levels of the image, and last one is divided. Another criteria is analysis of the picks in histogram as it has been mentioned early. Then the same procedure is applied to each subimage.

****

In the certain step a subimage is not be divided due to criteria satisfaction. The merge procedure is applied now trying to connect the subimage with neighboring one which has the same properties and which cannot be divided too (the averages of two images are compared with deviation:C:\Documents and Settings\khec\Desktop\chap3\43_files\image43_4.gif. If this test is not passed, these regions are independent.

****

Disadvantage of the approach is that, the number of the iterations is limited (for image *512x512* the *log2 512=9* iterations are possible), and the regions have no the real boundaries as it shown in figure.



Formal presentation of the split and merge technique is shown as following.

Given a set of regions *Rk, k=1,...,m*, a low-level segmentation might require the basic region properties to hold. The equations *( )* are the basis of this segmentation. H is homogeneity property

If the first part of *( )* is not satisfied for some *k*, it means that the region is geterogeneous and should be split into subregions. If the second part of the *( )* is not satisfied for some *i* and *j*, then the regions *i* and j are collectively homogeneous and should be merged into a single region.

The following statements are important:

C:\Documents and Settings\khec\Desktop\chap3\43_files\image43_5.gif

and

C:\Documents and Settings\khec\Desktop\chap3\43_files\image43_6.gif

*1. Region growing algorithm via split and merge*

A way of working toward the satisfaction of these homogeneity criteria is the split-and-merge algorithm (Horowitz and Paulidis). To use the algorithm it is necessary to organize the image pixels into a grid structure, regions are organized into groups of four (not ever in rectangular subregions). Any region can be split into four subregions (except a region consisting of only one pixel), and the appropriate groups of four can be merged into a single larger region. This structure is incorporated into the following region-growing algorithm.

Algorithm for Region growing algorithm via split and merge

a) Pick any grid structure and homogeneity property H. If for any region R in that structure, H(R) = false, split that region into four subregions. If for any four appropriate regions *Rk1,..., Rk4, H(Rk1 U Rk2 U Rk3 U Rk4 )= true,* merge them into a single region. When no regions can be further split or merged, stop.

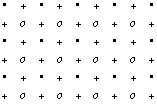
b) If there are any neighboring regions *Ri* and *Rj* (perhaps of different sizes) such that *H(Ri U Rj) =true*, merge these regions.

*2. State-space approach to region growing*

This approach regards the initial two-dimensional image as a discrete state, where every sample point is a separate region. Changes of state occur when a boundary between regions is either removed or inserted. The problem then becomes one of searching allowable changes in state to find the best partition. An important part of this approach is the use of data structures to allow regions and boundaries to be manipulated as units and to apply the same method as for pixels.

*a) Low-level boundary data structure*

Useful representation for boundaries allows the splitting and merging of regions to proceed in a simple manner using the notion of a supergrig *S* to the image grid *G*.

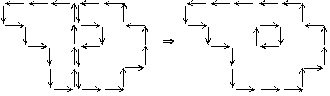


Grid structure for region representation, where · and + correspond to supergrid and o to the subgrid.

The representation is assumed to be four-connected (i.e. *x1* is a neighbor of *x2* if *||x1-x2|| 1*).

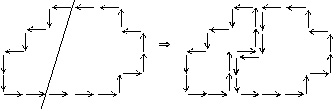
With this notation boundaries of region are directed crack edges at the point + . That is , if point *xk* is a neighbor of *xj* and *xk* is in a different region than *xj* , insert two edges for the boundaries of the regions containing *xj* and *xk* at the point + separating them, such that each edge traverses its associated region in counterclockwise sense . This makes the merge very simple:

To merge regions *Rk* and *RL* , remove edges of the opposite sense from the boundary as shown in figure



Split region operations on the grid structure of the previous figure

Similarly, to split a region along a line, insert edges of the opposite sense in nearby points as shown:



Merge region operations on the grid structure of the previous figure

The method uses three criteria for merging regions (three way to mergethe regions), reflecting a transition from local measurements to global measurements. These criteria use measures of boundary strength *sij* and *wij* defined as:

C:\Documents and Settings\khec\Desktop\chap3\43_files\image43_8.gif

where *xi* and *xj* are assumed to be on either side of a crack edge. The three criteria are applied sequentially in the following algorithm:

Algorithm for Region growing via boundary melting (*Tk, k=1,2,3* are present thresholds)

1). For all neighboring pairs of points, remove the boundary between *xi* and *xj* if *i=1* and *wij* =1. When no more boundaries can be removed, go to step 2)

2) remove the boundary between *Ri* and *Rj* if

C:\Documents and Settings\khec\Desktop\chap3\43_files\image43_9.gif

where W is the sum of the *wij* on the common boundary between *Ri* and *Rj* , that have perimeters *pi* and *pj* respectively. When no more boundaries can be removed, go to step 3).

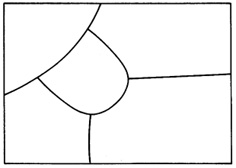
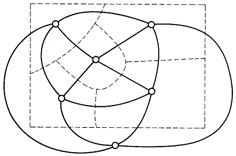
3) Remove the boundary between *Ri* and *Rj* if *W >= T3.*

*b). Graph-oriented region structures*

The previous approach stores boundaries but does not provide explicit representation of regions as units.

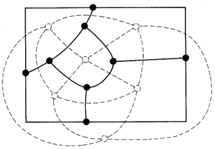
Using the graph theory both region and their boundaries may be represented and used scheme facilitates the storing and indexing of their semantic properties. The proposed scheme is based on a special graph called the region adjacency graph, and its dual graph. In the region adjacency graph, nodes are regions and arcs exist between neighboring regions.

Consider the figure with regions and then its representation as adjacency graph where arcs are crossing each separate boundary segment and node has junction between three or more boundary segments:

a) b) 

a) an image partition , b) the region adjacency graph (solid lines).

The dual graph of adjacency graph is constructed by placing the node in each separate region of the adjacency graph and connect them with arcs as shown:



The dual of the adjacency graph (solid lines)

The dual graph is like the original region boundary map. By maintaining both the region adjacency graph and its dual, one can merge regions using the following algorithm:

Algorithm for Merging using the region-adjacency graph and its dual

Task: Merge neighboring regions *Ri* and *Rj .*

Phase 1. Update the region-adjacency graph.

1) Place edges between *Ri* and all neighboring regions of *Rj* (excluding *Ri* ) that do not already have edges between themselves and *Ri* .

2) Delete *Rj* and all its associated edges.

Phase 2 Take care of the dual

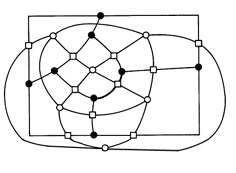
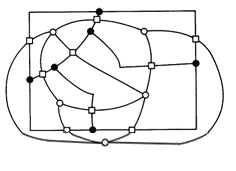
1) Delete the edges in the dual corresponding to the borders between *Ri* and *Rj* .

2) For each of the nodes associated with these edges:

a) if the resultant degree of the node is less than or equal to 2, delete the node and join the two dangling edges into a single edge.

b) otherwise, update the labels of the edges that were associated with *j* to reflect the new region label *i*.

The following figure shows these operations.

a) b) 

Merging operations using the region adjacency graph and its dual.

a) before merging regions separated by dark boundary line, b) after merging

**(iv) Incorporation of semantics.**

In previous section the region-merging decisions were based on raw image data and week heuristics of general applicability about the likely shape of boundaries. But possible interpretation of regions can use the domain-dependent knowledge, for example, sky, grass, car using corresponding properties such as intensity and hue. The region growing may be done by using the semantic labels for merging process [Feldman-Yakimovsky]. Early steps are same as it was shown in the previous sections (non-semantic criteria of region detection); once regions attain significant size, semantic criteria are used.

Algorithm for semantic region growing

Non-semantic criteria (*T1* and *T2* are preset thresholds)

1. Merge regions *i, j* as long as they have one weak separating edges until no two regions pass this test.

2. Merge regions *i, j* where *S(i, j )T2* where

C:\Documents and Settings\khec\Desktop\chap3\43_files\image43_10.gifwhere *c1* and *c2* are constants

C:\Documents and Settings\khec\Desktop\chap3\43_files\image43_11.gifuntil no two regions pass this test.

Semantic criteria:

3. Let *Bij* be the boundary between *Ri* and *Rj*. Evaluate each *Bij* with a Bayesian decision function that measures the (conditional) probability that *Bij* separates two regions *Ri* and *Rj* of the same interpretation. Merge *Ri* and *Rj* if this conditional probability is less than some threshold. Repeat step 3 until no regions pass the threshold test.

4. Evaluate the interpretation of each region *Ri* with a Bayesian decision function to confirm that an interpretation is correct one for that region. Assign the interpretation to the region with the highest confidence of correct interpretation. Update the conditional probabilities for different interpretations of neighbors. Repeat the entire process until all regions have interpretation assignments.

(P:S: Bayesian classificator. It is an statistical approach for pattern recognition. For example, the values of the specific characteristics have the normal distribution and may be described by its nominal value (average) and a tolerance (disviacion) with respect to this value. The certain quantity of the measurements need to obtain the numerical data. In this way the probability of found characteristic *X* is computed for each class iP(X|i). And now for classification, the probability values are compared; for example, the probability that diameter of the screw is 3mm. These probabilities are called *a priory* because the measurements have been done for known objects. The classification may be done by maximum probability value. Now it possible to say that P(i|X)=P(X|i), or the probability *a posteriori* P(i|X) is equal to probability a priory. It is denominated a posteriori because it wants to know what is a probability that the object with diameter 3 mm is the screw. Now it is it is possible to introduce the relationship between the number of the objects in the class p(i) - probability a priori that the selected element is presented in class i . Now the multiplication of the probability a priori that determined element is appeared and that this type has defined value is obtained as:

P(i|X)=P(X|i)p(i). Therefore the Bayes' follows as :

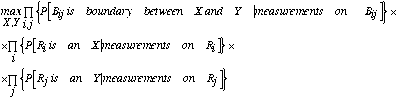
C:\Documents and Settings\khec\Desktop\chap3\43_files\image43_12.gif

where the new term *p(X)* is probability a priory that the element with vector of characteristics equal to *X* which value is the sum of the probabilities a priori of the all classes is appears:

C:\Documents and Settings\khec\Desktop\chap3\43_files\image43_13.gif

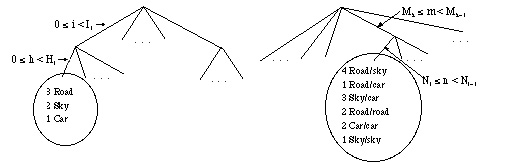
Finally, the element which characteristics *X* will belong to the class which probability is maximum.)

The semantic portion of previous algorithm had a goal of maximizing an evaluation function measuring the probability of correct interpretation (labeled partition), given the measurement on the boundaries and regions of the partition. An expression for the evaluation function is (for given partition and interpretation *X* and *Y*):



where *P* stands for probability and  is the product operator.

How are these terms to be computed? An approximation of [Feldman-Yakimovsky] is to quantize the measurements and represent them in terms of classification tree. The conditional probabilities then may be computed from data at the leaves of the tree. The hypothetical tree for interpretation of road, sky, and car using region measurement of intensity and hue may be constructed. Similarly, the equivalent tree for two boundary measurements m and n is obtained for the same interpretation.



Hypothetical classification tree for region Hypothetical classification tree for boundary  
measurements showing a particular branch measurements showing specific branch for  
for specific ranges of intensity and hue. specific ranges of two measurements *m* and *n*.

These two figures indicate that   
*P[Ri is a CAR | 0  i < I, 0  h < Hi]=1/6,*  
and  
*P[Bij* divides two car regions | Mk  m < Mk+1, Nk < n  Nk+1]=2/13.  
These trees were created by laborious trials with correct segmentation of test images. Now, finally, consider again step 3 of previous algorithm. Probability that a boundary *Bij* between regions *R*i and *Ri* is false is given by

C:\Documents and Settings\khec\Desktop\chap3\43_files\image43_18.jpg

where *Pf =**{P[Bij* is between two subregions *X* | *Bij* 's measurements]} x  
x *{P[Ri* is *X* | measurements*]}* x *{P[Rj* is *X* | measurements]}  
*Pt =*x,y *{P[Bij* is between *X* and *Y* | measurements]} x  
x *{P[Ri* is *X* | measurements*]}* x *{P[Rj* is *Y* | measurements]}

And for step 4 of the algorithm:

C:\Documents and Settings\khec\Desktop\chap3\43_files\image43_19.jpg

where *X1, X2* are the first and second most likely interpretations.  
After the region is assigned interpretation X1, the neighbors are updated using   
*P[Ri* is *X* | measurements*]:=*Prob *[Rj* is *X* | measurements] x *P[Bij* is between *X* and *X1* | measurements]

*2. Compass operators*

Compass operators (in Spanish literature: operador de ajuste al modelo) measure gradients in a selected number of directions as it shown in the following algorithm:

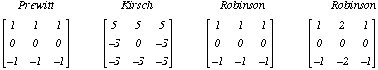
*C:\Documents and Settings\khec\Desktop\chap3\12_files\Image411_1.gif*

Edge detection via compass operators

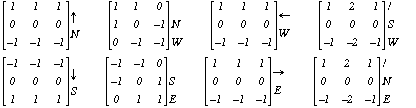
Taking as a base the five masks used for gradient operator the circular shift of the eight boundary elements of these masks gives a *45°* rotation of the gradient direction.

Compass gradients (North). Each Clockwise Circular Shift of Elements

about the Center Rotates the Gradient Direction by 45°

**

For example, the eight compass gradients corresponding to the third (Robinson) operator are:

**

Let *gk(m,n)* denote the compass gradient in the direction q *k=p /2+kp /4, k=0,...,7.* The gradient at location *(m,n)* is denoted as

C:\Documents and Settings\khec\Desktop\chap3\12_files\Image411_4.gif

which can be thresholded to obtain the edge map as before.



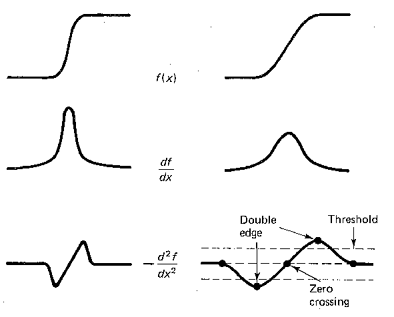
Edge detection by Kirsch operator. The gradient image and edge map.

Note that only four of the preceding eight compass gradients are linearly independent. Therefore, it is possible to define four *3x3* array that are mutually orthogonal. These arrays are called orthogonal gradients. Compass gradients with higher angular resolution can be designed by increasing the size of the mask.

*3. Laplace Operators and Zero Crossing*

The gradient operators are used for enhancement of abrupt gray-level transitions. As the transition region gets wider, it is more advantageous to apply the second-order derivatives. One frequently encountered operator is the Laplacian operator, defined as

C:\Documents and Settings\khec\Desktop\chap3\12_files\Image411_6.gif



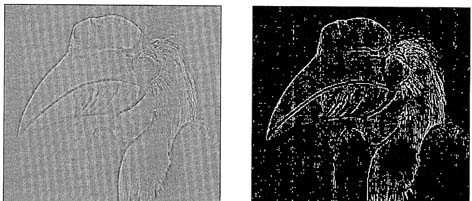
First and second derivatives for edge detection

Usually used three different discrete approximations of the Laplacian operator are:

C:\Documents and Settings\khec\Desktop\chap3\12_files\image411_20.jpg

where the central pixel takes the negative value of the sum of all neighbored pixels.

The example of second Laplace mask application is shown in the following figure



Laplacian gradient image and the edge map

Because the second-order derivatives, this gradient operator is more sensitive to noise than first-order gradient operators. Also the thresholded magnitude of Laplacian operator produces double edges. For these reasons, together with its inability to detect the edge direction, the Laplacian as such is not a good edge detection operator. A better utilization of it is to use its zero-crossing to detect the edge locations. A generalized Laplacian operator, which approximates the Laplacian of Gaussian functions, is a powerful zero-crossing detector defined as :

C:\Documents and Settings\khec\Desktop\chap3\12_files\Image411_9.gif

where ** controls the amplitude response of the filter output but does not affect the location of the zero-crossing; and *c* normalizes the sum of the elements of a given size mask. Operators based on derivatives of Gaussian function is known as the Marr-Hildreth operator where zero-crossing is applied.

This operator is based on the computing of the Laplacian of the Gaussian function. If an image is convoluted with Gaussian such as

C:\Documents and Settings\khec\Desktop\chap3\12_files\Image411_10.gif

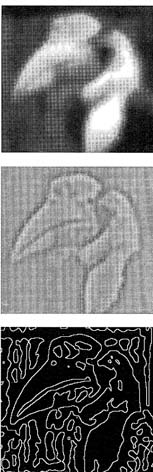
then the noise influence is reduced. If then the second derivatives is computed the Marr-Hildreth operator is obtained (LoG - Laplassian of the Gaussian):

C:\Documents and Settings\khec\Desktop\chap3\12_files\image411_21.jpg

In the resented expression first the convolution of image *I(x,y)* is done with Gaussian and then the Laplacian is applied. According to the Gaussian properties the previous expression may be computed more quickly by LoG and then image convolution:

C:\Documents and Settings\khec\Desktop\chap3\12_files\image411_22.jpg

In the following figure the convolution with the Gaussian and zero-crossing edge detection by application of Laplacian are shown.

Edge detection according zero-crossing Marr-Hildreth operator

Note that the more strict Gaussian deviations provide greater number of the detected borders (borders of the smallest objects may be detected).

*4. Canny edge detector*

Canny edge detector is the another kind of the derivative operators. It based on the concept that : the operator must to detect only edges; the distance between the pixel considered as a part of edge and the real edge must be minimum; and do not identify several pixels as border if there is only one. Canny finds the optimization of the signal-noise relationship by applying the derivative of the Gaussian.

The Canny's edge detection algorithms consists in the following sequence of steps:

\* The input data are image *I* and one-dimensional Gaussian *G*

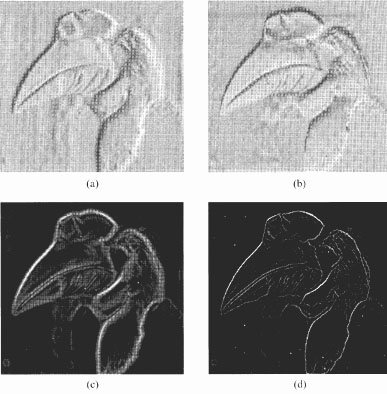
\* The one dimensional derivatives of Gausssian are obtained (*Gx* and *Gy*)

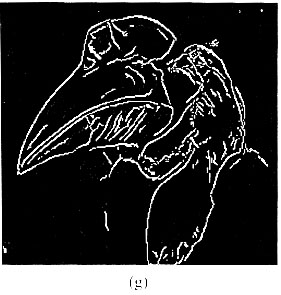
\* The convolutions of the obtained Gaussian derivatives and image are computed, that gives the images *Ix, Iy*. (in the following figure a) and b))

\* The image as magnitude *M* as square root of the sum of *Ix, Iy*'s squares is found (c).

\* The next step is the removal of the pixels which are not in maximum. The magnitude of each pixel is compared with its neighboring pixels, those ones which have the maximum magnitude in their neighborhood are maintained as they are (d).

\* For edge detection two thresholds *T1, T2* are used. If the less threshold *T1,* is used the small importance edges of the object may be detected (e). If only *T2* is used the part of the pixels belonging to the edges is detected (f). That is why the condition for belonging of pixel to the edge is to be more than *T2* or more than *T1,* always if there is at least one another in the pixel neighborhood which is more than *T2* (g).





Canny Edge Detector

*5. Performance of Edge Detection Operators*

Edge detection operators can be compared in a quantitative way. Let *n0* be the number of edge pixels declared as *n1* be number of missed or new edge pixels after adding the noise. The edge detection error rate is

C:\Documents and Settings\khec\Desktop\chap3\12_files\Image411_16.gif

Another possible analysis of the edge detector performance is the quantity

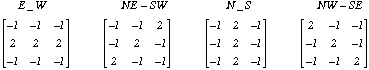
C:\Documents and Settings\khec\Desktop\chap3\12_files\image411_23.jpg

where di is the distance between a pixel declared as edge and the nearest ideal edge pixel, ** is a calibration constant, and *NId* and *NDe*t are the numbers of ideal and detected edge pixels respectively.

Among the gradient and compass operators the Sobel and Prewitt operators have been found to yield the highest performance (performance is proportional to *P*).

***(v) Line Detection***

Lines are extended edges. Usually, the set of four filters are used for line detection in the different directions.



The first mask is used for lines of the horizontal orientation (the maximum value of the output is obtained if a line passes through the central row pf the mask. The second mask is used for detection of the lines with inclination of *+45°*, third : *+90°,* fourth : *-45°*

***(vi) Spot and Corner detection***

Practically, there are two reasons of the corner existence and their analysis:

\* Corners correspond to intersection of two borders of the one object;

\* They represent discontinuity in gray scale levels due to textures of the objects.

*a) One technique for corner detection* is based on applying the border detection operators and then to find the points where the curvature has a significant variations.

*b) Another technique is Tomasi and Kanade method* which consists in determination of zones within the image where the horizontal and vertical gradients gives high response in the same time. For this the neighborhood of the pixel is analyzed by applying the following expression:

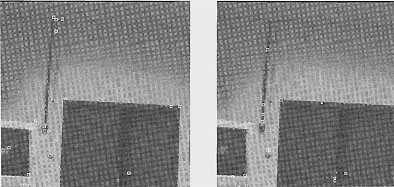
C:\Documents and Settings\khec\Desktop\chap3\12_files\Image411_18.gif

Using this equation it is possible to decide what distribution of the gray scale is in pixels neighborhood.

*c) The Kitchen and Rosenfeld method* like the previous method is based on the variation of the gradient directions computing :

C:\Documents and Settings\khec\Desktop\chap3\12_files\image411_24.jpg

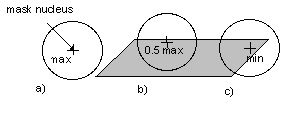
The following figure shows the result of applying the mentioned methods



Detected corners on the input image according

a) Tomasi and Kanade method b) Kitchen and Rosenfeld method

*d) SUSAN method* proposed by Steve Smith provides simple feature detection., The method based on manipulation with USAN (Univalue Segment Assimilating Nucleus) circular mask with center denominated as a nucleus. The brightness of each pixel within a mask is compared with the brightness of that mask nucleus and USAN area of the mask with the same or similar brightness of pixels as the nucleus can be defined. In the figure the circular masks have different USAN areas due to their position within image.



Circular masks with corresponding USAN areas

The USAN area is at a maximum when the nucleus has the same brightness as all pixels - a), it falls to half of this maximum very near a straight edge - b), and falls even further when inside a corner - c). This is a property of USAN area is used for detection of the presence of edges and corners in 2D frames. This approach uses no image derivatives and is good in the presence of noise due to the integrating effect of the principle. The noise is small enough to be interpreted as a part of USAN area. For this approach there is no other condition about structure of analyzed region and operations with controlling parameters are much simpler and less arbitrary than with most other edges detection algorithms.

**USAN-Based Algorithm for Corner Detection**

For estimation of motion the analysis of principal corners of objects within image sometimes is enough. In this case the proposed algorithm based on the property of minimum USAN area near corners of object is quite acceptable. The algorithm consists in the following steps:  
(1) Each point in the input image is used as the nucleus of a circular mask. The best digital approximation for calculation of the mask value is a Gaussian weighting because it is more smoother and stable than square similarity function (versus pixel brightness difference). The usual radius for each mask is 3.4 pixels that gives 37 pixels per mask.

(2) Using equation

C:\Documents and Settings\khec\Desktop\chap3\12_files\image411_245.jpg

the brightness function is calculated for each pixel under the mask. In the equation *I(r0), I(r)* present the brightness of nucleus position and the brightness of other point within the mask respectively*; t* is the brightness difference threshold which defines minimum contrast of edges and image background. The number of estimated corners directly depends on *t* value. The *6th* power of equation is a theoretical optimum, which permits good balance between stability and the sensitivity of method.

(3) Then value of USAN area *n* is calculated as sum of computed *C (r, r0)* functions.

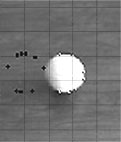
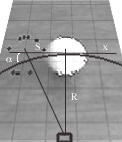
(4) For the detection of corners the USAN area property is used comparing the estimated areas with the threshold *g < nmax/2,* wher*e nmax* is maximum area of the circular mask. The threshold *g* defines quality and quantity of the detected corners.

(5) Finally, the wrongly detected corners must be rejected as a noise if the mask gravity has a great value (medium of the distance between point and nucleus). The same steps of the algorithm are applied for the second image in order to detect its principal corners. The composite image as result of corner detection procedure is presented in the second frame. The corners of the static objects cover each another and can be removed for the following analysis of motion characteristics. Therefore the advantage of this approach is the less amount of processed data for motion field construction.



Frame with objects' corners and composite image with superimposed corners

In case of estimating the principal corners of geometrical objects the complexity and the processing time are reduced significantly. For example, for rectangular objects the four points corresponding to corners will be processed. It is interesting to see as the algorithm detects the corners of the circular object shown in the following figure. Regularity of the corner appearance depends on the quality of the image and clarity of the objects' borders. That is why in the left part of the ball where the border is washed out the number of reported corners is less. This problem can be solved by appropriate selection of the threshold *t* value in the equation.

****

Objects' corners detection and linear displacement error of circular route interpretation

Disadvantage of this proposed approach is that the algorithm detects only the linear displacement due to analysis of the limited number of frames.

**4.4 Texture exposition**

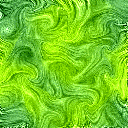
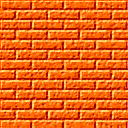
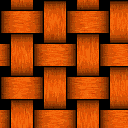
The dictionary description is "something composed of closely interwoven elements". Practically, the human attempts the individual elements if the number of them may be counted. If the number is large it is difficult to characterize these elements individually and they merge into less distinct homogeneous spatial repeated patterns which can be called as texture.

The exposition of texture takes place under four main headings:

**(i) Texture primitives.** (Texture may be described as being composed of elements of texture primitives)

For highlight the importance of the texture primitives the term texel (for texture element) is used. A texel is a visual primitive with certain invariant properties which occurs repeatedly in different positions, deformations, and orientations inside a given area (for example, property of such a unit might be that its pixels have a constant gray level)

Texture primitives may be pixels or aggregates of pixels such as curve segments or regions. The idea of appropriate resolution, or number of texels in a sub-image is implicit part of quantitative definition of texture.



Examples of textures

**(ii) Structural models.** (Structural models regards the primitives as forming a repeating pattern and describes such patterns in terms of rules or grammar). The grammar describes how to generate patterns by applying rewriting rules to a small number of symbols. There is no unique grammar for a given texture - infinitely many choices for rules and symbols.

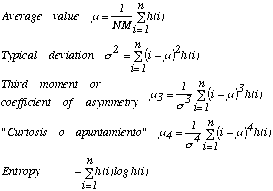
There are many variants of the basic idea of formal grammars: shape grammar (8 rules to describe the texture), tree grammar (85 rules for the same texture), array grammar, etc. In the shape grammar the high-level primitives closely correspond to the shapes in the texture. In the tree and arrays grammars the texels are defined as pixels and this makes the grammar more complicated.

**(iii) Statistical models** (Statistical models describes texture by statistical rules governing the distribution and relation of gray levels).

The statistical analysis of texture implies computing the distribution of certain property (gray level, average value, deviation, dispersion, entropy...) for each pixel of the image. According the number of points defined a texture the statistics of the first, second, and upper order are distinguished.

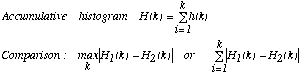
*1. First order statistics*

If the histogram of image is divided by total number of pixels the result represents the probability that the certain gray level appears in the image. Thus the following related properties may be estimated.



The average represents the gray level estimation, typical deviation shows the dispersion with respect this value, third moment is measurement of asymmetry of the histogram. The \_\_\_ (la curtosis o apuntamiento) indicates how the histogram is distributed in the central part and the extreme sides. The entropy measures the uniformity of the histogram.

Another way for analysis of texture is comparison of two histograms according the Kolmogorov-Smirnov test. For this the accumulative distribution of histogram *H1(k)* is used which then is compared with the accumulative distribution of histogram *H2(k)* of image with ideal texture:

**

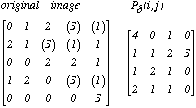
Disadvantage the first order statistics is analysis based on the histogram which is not operate with all spatial data (only probability of gray levels). That is why the second order statistics is used.

*2. Second order statistics*

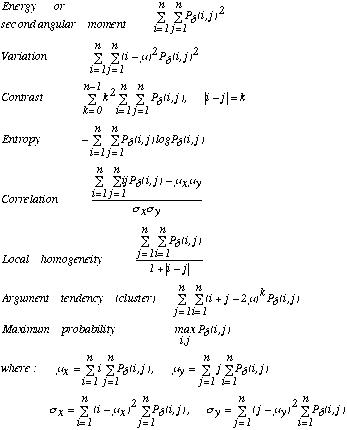
a) Co-occurrence matrixes

For the vector with polar coordinates  *=(r, )*, the conditional probability *P* that two properties appear separated by distance  may be calculated. The co-occurrence matrixes are computed taking into account certain limitations, for example, the *r* value is *1* (one pixel), angles are *0°, 45°,90°,* and *135°*).

Example. For following image (if *r=1* and  *=0°)* the *P (3,1)=3* because three pairs *3-1* are found in the image. Then the matrix *P (i,j)* is computed (*i* and *j* are properties, in the example, they are the gray levels: 0,1,2, and 3). First row is calculated as probability that two properties (gray levels from 0 to 3) have the distance *r=1* at the angle  *=0°: P (0,0)=4; P (1,0)=0; P (2,0)=1; P (3,0)=0.* The first column is computed as : *P (0,0)=4; P (0,1)=1; P (0,2)=1; P (0,3)=2.*



Some other properties may be extracted from the obtained matrix:



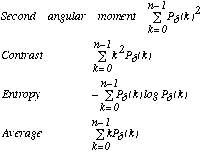
The second angular moment indicates the homogeneity of the image, the contrast - variation within the image, correlation - linearity of the image, entropy - uniformity of the matrix.

b) Difference statistics

This analysis is determined by the distribution of probability *P (k)* of values of intermediate pixels which lie between ones separated by distance  *.* It is given as:

C:\Documents and Settings\khec\Desktop\chap3\433_files\Image44_8.gif

The properties of this distribution are



**(iv) Frequency analysis of the texture**

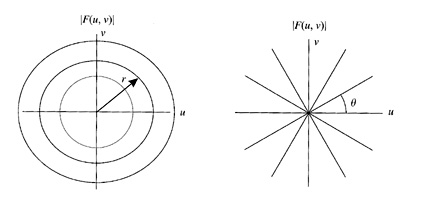
The Fourier transform may be used for texture analysis. If the Fourier transform of an image f(x,y) is F(u,v) its module is:

C:\Documents and Settings\khec\Desktop\chap3\433_files\Image44_10.gif

If the polar coordinates are used two distribution may be obtained:

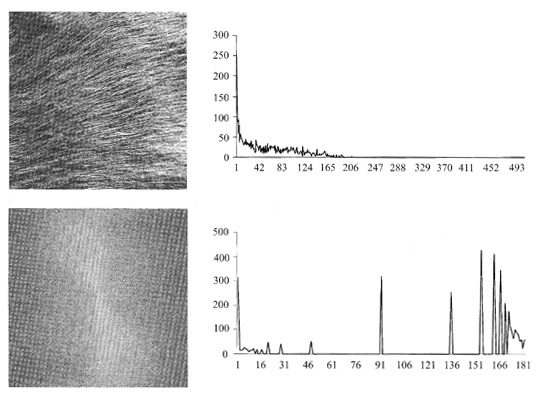
C:\Documents and Settings\khec\Desktop\chap3\433_files\Image44_11.gif

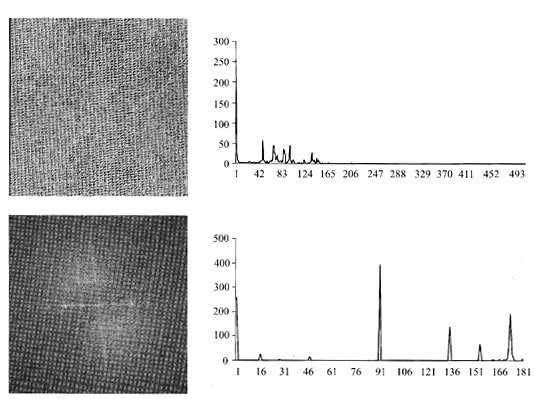
The first distribution indicates the dominated sizes of texture, the second - its direction.

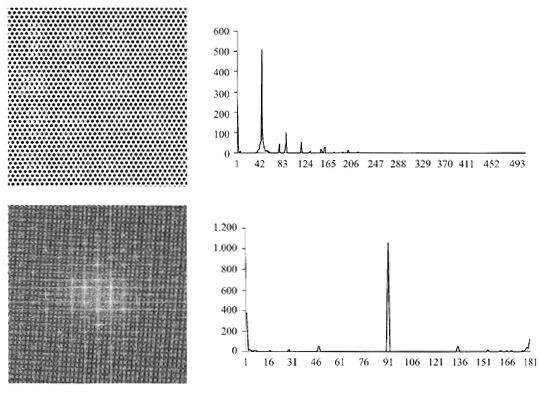


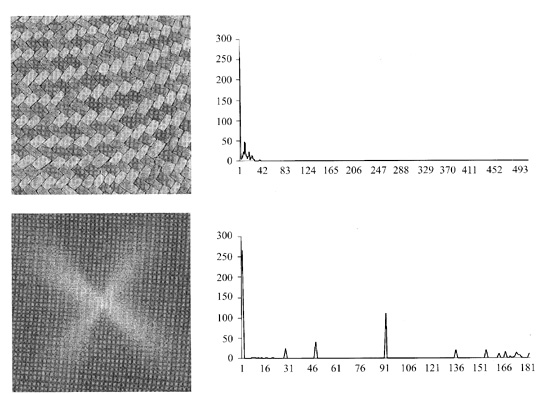
Determination of texture by Fourier *|F(u,v)|*

Some examples of the texture determination by Fourier are shown :

**

**

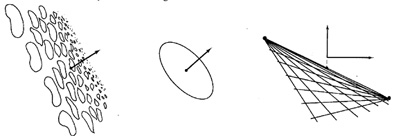
**

**

Texture determination by Fourier transform

**(v) Texture gradients** (Texture gradients are used for determining the direction of greatest change in size of primitives and spatial placement of primitives).

The important point of the texture analysis is determining the surface orientation. There are three ways assumed that texture is embedded on a planar surface.



Methods for calculation surface orientation from texture.

The first, the texture is segmented into primitives and then the direction of maximum rate of change of projected primitive size is computed as the direction of the texture gradient. The magnitude of gradient determines how much the plane is tilted with respect to camera.

The second way to measure surface orientation is by knowing the shape of the textel itself. For example, a circular textel appears as ellipse on the tilted surface. The orientation of the principal axes defines rotation with respect to the camera, and the ratio of minor to major axes defines tilt.

The third, if the texture is composed of a regular grid of texels, the vanishing points are computed. For a perspective image, vanishing points on a plane are the projection onto the image plane of the points at infinity in given direction. In the figure the texels themselves are small line segments on a plane that are oriented in two orthogonal directions in the physical world.

The general method applies whenever the placement tessellation defines lines of textels. Two vanishing points that arise from texels on the same surface can be used to determine orientation as follows. The line joining the vanishing points provides the orientation of the surface and the vertical position of the plane with respect to the z axis (i.e. the intersection of the joining the vanishing points with x=0) determines the tilt of the plane.