function ellipse\_t = fit\_ellipse( x,y,axis\_handle )

%

% fit\_ellipse - finds the best fit to an ellipse for the given set of points.

%

% Format: ellipse\_t = fit\_ellipse( x,y,axis\_handle )

%

% Input: x,y - a set of points in 2 column vectors. AT LEAST 5 points are needed !

% axis\_handle - optional. a handle to an axis, at which the estimated ellipse

% will be drawn along with it's axes

%

% Output: ellipse\_t - structure that defines the best fit to an ellipse

% a - sub axis (radius) of the X axis of the non-tilt ellipse

% b - sub axis (radius) of the Y axis of the non-tilt ellipse

% phi - orientation in radians of the ellipse (tilt)

% X0 - center at the X axis of the non-tilt ellipse

% Y0 - center at the Y axis of the non-tilt ellipse

% X0\_in - center at the X axis of the tilted ellipse

% Y0\_in - center at the Y axis of the tilted ellipse

% long\_axis - size of the long axis of the ellipse

% short\_axis - size of the short axis of the ellipse

% status - status of detection of an ellipse

%

% Note: if an ellipse was not detected (but a parabola or hyperbola), then

% an empty structure is returned

% =====================================================================================

% Ellipse Fit using Least Squares criterion

% =====================================================================================

% We will try to fit the best ellipse to the given measurements. the mathematical

% representation of use will be the CONIC Equation of the Ellipse which is:

%

% Ellipse = a\*x^2 + b\*x\*y + c\*y^2 + d\*x + e\*y + f = 0

%

% The fit-estimation method of use is the Least Squares method (without any weights)

% The estimator is extracted from the following equations:

%

% g(x,y;A) := a\*x^2 + b\*x\*y + c\*y^2 + d\*x + e\*y = f

%

% where:

% A - is the vector of parameters to be estimated (a,b,c,d,e)

% x,y - is a single measurement

%

% We will define the cost function to be:

%

% Cost(A) := (g\_c(x\_c,y\_c;A)-f\_c)'\*(g\_c(x\_c,y\_c;A)-f\_c)

% = (X\*A+f\_c)'\*(X\*A+f\_c)

% = A'\*X'\*X\*A + 2\*f\_c'\*X\*A + N\*f^2

%

% where:

% g\_c(x\_c,y\_c;A) - vector function of ALL the measurements

% each element of g\_c() is g(x,y;A)

% X - a matrix of the form: [x\_c.^2, x\_c.\*y\_c, y\_c.^2, x\_c, y\_c ]

% f\_c - is actually defined as ones(length(f),1)\*f

%

% Derivation of the Cost function with respect to the vector of parameters "A" yields:

%

% A'\*X'\*X = -f\_c'\*X = -f\*ones(1,length(f\_c))\*X = -f\*sum(X)

%

% Which yields the estimator:

%

% ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

% | A\_least\_squares = -f\*sum(X)/(X'\*X) ->(normalize by -f) = sum(X)/(X'\*X) |

% ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

%

% (We will normalize the variables by (-f) since "f" is unknown and can be accounted for later on)

%

% NOW, all that is left to do is to extract the parameters from the Conic Equation.

% We will deal the vector A into the variables: (A,B,C,D,E) and assume F = -1;

%

% Recall the conic representation of an ellipse:

%

% A\*x^2 + B\*x\*y + C\*y^2 + D\*x + E\*y + F = 0

%

% We will check if the ellipse has a tilt (=orientation). The orientation is present

% if the coefficient of the term "x\*y" is not zero. If so, we first need to remove the

% tilt of the ellipse.

%

% If the parameter "B" is not equal to zero, then we have an orientation (tilt) to the ellipse.

% we will remove the tilt of the ellipse so as to remain with a conic representation of an

% ellipse without a tilt, for which the math is more simple:

%

% Non tilt conic rep.: A`\*x^2 + C`\*y^2 + D`\*x + E`\*y + F` = 0

%

% We will remove the orientation using the following substitution:

%

% Replace x with cx+sy and y with -sx+cy such that the conic representation is:

%

% A(cx+sy)^2 + B(cx+sy)(-sx+cy) + C(-sx+cy)^2 + D(cx+sy) + E(-sx+cy) + F = 0

%

% where: c = cos(phi) , s = sin(phi)

%

% and simplify...

%

% x^2(A\*c^2 - Bcs + Cs^2) + xy(2A\*cs +(c^2-s^2)B -2Ccs) + ...

% y^2(As^2 + Bcs + Cc^2) + x(Dc-Es) + y(Ds+Ec) + F = 0

%

% The orientation is easily found by the condition of (B\_new=0) which results in:

%

% 2A\*cs +(c^2-s^2)B -2Ccs = 0 ==> phi = 1/2 \* atan( b/(c-a) )

%

% Now the constants c=cos(phi) and s=sin(phi) can be found, and from them

% all the other constants A`,C`,D`,E` can be found.

%

% A` = A\*c^2 - B\*c\*s + C\*s^2 D` = D\*c-E\*s

% B` = 2\*A\*c\*s +(c^2-s^2)\*B -2\*C\*c\*s = 0 E` = D\*s+E\*c

% C` = A\*s^2 + B\*c\*s + C\*c^2

%

% Next, we want the representation of the non-tilted ellipse to be as:

%

% Ellipse = ( (X-X0)/a )^2 + ( (Y-Y0)/b )^2 = 1

%

% where: (X0,Y0) is the center of the ellipse

% a,b are the ellipse "radiuses" (or sub-axis)

%

% Using a square completion method we will define:

%

% F`` = -F` + (D`^2)/(4\*A`) + (E`^2)/(4\*C`)

%

% Such that: a`\*(X-X0)^2 = A`(X^2 + X\*D`/A` + (D`/(2\*A`))^2 )

% c`\*(Y-Y0)^2 = C`(Y^2 + Y\*E`/C` + (E`/(2\*C`))^2 )

%

% which yields the transformations:

%

% X0 = -D`/(2\*A`)

% Y0 = -E`/(2\*C`)

% a = sqrt( abs( F``/A` ) )

% b = sqrt( abs( F``/C` ) )

%

% And finally we can define the remaining parameters:

%

% long\_axis = 2 \* max( a,b )

% short\_axis = 2 \* min( a,b )

% Orientation = phi

%

%

% initialize

orientation\_tolerance = 1e-3;

% empty warning stack

warning( '' );

% prepare vectors, must be column vectors

x = x(:);

y = y(:);

% remove bias of the ellipse - to make matrix inversion more accurate. (will be added later on).

mean\_x = mean(x);

mean\_y = mean(y);

x = x-mean\_x;

y = y-mean\_y;

% the estimation for the conic equation of the ellipse

X = [x.^2, x.\*y, y.^2, x, y ];

a = sum(X)/(X'\*X);

% check for warnings

if ~isempty( lastwarn )

disp( 'stopped because of a warning regarding matrix inversion' );

ellipse\_t = [];

return

end

% extract parameters from the conic equation

[a,b,c,d,e] = deal( a(1),a(2),a(3),a(4),a(5) );

% remove the orientation from the ellipse

if ( min(abs(b/a),abs(b/c)) > orientation\_tolerance )

orientation\_rad = 1/2 \* atan( b/(c-a) );

cos\_phi = cos( orientation\_rad );

sin\_phi = sin( orientation\_rad );

[a,b,c,d,e] = deal(...

a\*cos\_phi^2 - b\*cos\_phi\*sin\_phi + c\*sin\_phi^2,...

0,...

a\*sin\_phi^2 + b\*cos\_phi\*sin\_phi + c\*cos\_phi^2,...

d\*cos\_phi - e\*sin\_phi,...

d\*sin\_phi + e\*cos\_phi );

[mean\_x,mean\_y] = deal( ...

cos\_phi\*mean\_x - sin\_phi\*mean\_y,...

sin\_phi\*mean\_x + cos\_phi\*mean\_y );

else

orientation\_rad = 0;

cos\_phi = cos( orientation\_rad );

sin\_phi = sin( orientation\_rad );

end

% check if conic equation represents an ellipse

test = a\*c;

switch (1)

case (test>0), status = '';

case (test==0), status = 'Parabola found'; warning( 'fit\_ellipse: Did not locate an ellipse' );

case (test<0), status = 'Hyperbola found'; warning( 'fit\_ellipse: Did not locate an ellipse' );

end

% if we found an ellipse return it's data

if (test>0)

% make sure coefficients are positive as required

if (a<0), [a,c,d,e] = deal( -a,-c,-d,-e ); end

% final ellipse parameters

X0 = mean\_x - d/2/a;

Y0 = mean\_y - e/2/c;

F = 1 + (d^2)/(4\*a) + (e^2)/(4\*c);

[a,b] = deal( sqrt( F/a ),sqrt( F/c ) );

long\_axis = 2\*max(a,b);

short\_axis = 2\*min(a,b);

% rotate the axes backwards to find the center point of the original TILTED ellipse

R = [ cos\_phi sin\_phi; -sin\_phi cos\_phi ];

P\_in = R \* [X0;Y0];

X0\_in = P\_in(1);

Y0\_in = P\_in(2);

% pack ellipse into a structure

ellipse\_t = struct( ...

'a',a,...

'b',b,...

'phi',orientation\_rad,...

'X0',X0,...

'Y0',Y0,...

'X0\_in',X0\_in,...

'Y0\_in',Y0\_in,...

'long\_axis',long\_axis,...

'short\_axis',short\_axis,...

'status','' );

else

% report an empty structure

ellipse\_t = struct( ...

'a',[],...

'b',[],...

'phi',[],...

'X0',[],...

'Y0',[],...

'X0\_in',[],...

'Y0\_in',[],...

'long\_axis',[],...

'short\_axis',[],...

'status',status );

end

% check if we need to plot an ellipse with it's axes.

if (nargin>2) & ~isempty( axis\_handle ) & (test>0)

% rotation matrix to rotate the axes with respect to an angle phi

R = [ cos\_phi sin\_phi; -sin\_phi cos\_phi ];

% the axes

ver\_line = [ [X0 X0]; Y0+b\*[-1 1] ];

horz\_line = [ X0+a\*[-1 1]; [Y0 Y0] ];

new\_ver\_line = R\*ver\_line;

new\_horz\_line = R\*horz\_line;

% the ellipse

theta\_r = linspace(0,2\*pi);

ellipse\_x\_r = X0 + a\*cos( theta\_r );

ellipse\_y\_r = Y0 + b\*sin( theta\_r );

rotated\_ellipse = R \* [ellipse\_x\_r;ellipse\_y\_r];

% draw

hold\_state = get( axis\_handle,'NextPlot' );

set( axis\_handle,'NextPlot','add' );

plot( new\_ver\_line(1,:),new\_ver\_line(2,:),'r' );

plot( new\_horz\_line(1,:),new\_horz\_line(2,:),'r' );

plot( rotated\_ellipse(1,:),rotated\_ellipse(2,:),'r' );

set( axis\_handle,'NextPlot',hold\_state );

end