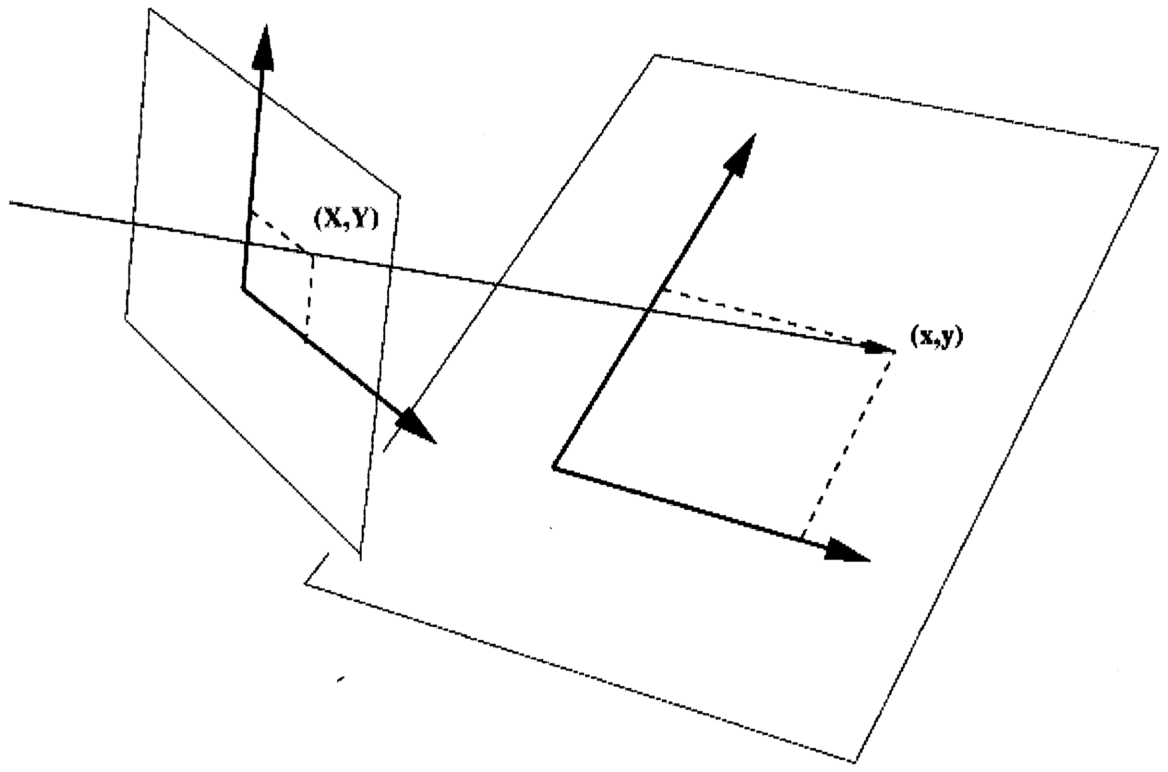
Perspective Transform Estimation

This page is intended as a companion to the online paper A Plane Measuring Device by A. Criminisi, I. Reid, and A. Zisserman at the University of Oxford.

The Task

is to find a transform that maps one arbitrary 2D quadrilateral into another. One approach is use the perspective transform. The following figure illustrates the situation:



In this illustration, the first quadrilateral is shown on the Image Plane and the second quadrilateral is shown on the World Plane.

Why use this transform over other mappings? It has the interesting property that it maps straight lines to straight lines. Other transforms of this nature introduce distortions that cause some straight lines in one space to map to curves in the other space.

So the task is, given the coordinates of the four corners of the first quadrilateral, and the coordinates of the four corners of the second quadrilateral, compute the perspective transform that maps a new point in the first quadrilateral onto the appropriate position on the second quadrilateral.

Deriving the Homographic Transform

The solution is stated in Section 3.1 of the paper, but the derivation is a bit lacking. I will briefly sketch the derivation here to make the solution more appetizing. We start with the claim that the camera model could be written as X=Hx, where X is vector of world plane coordinates, x is the vector of image plane coordinates, and H is a matrix transform. We can write the this form out in more detail as:

$$\begin{bmatrix} XW \\ YW \\ W \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Then if we realize that W is actually:

$$W = gx + hy + 1$$

we can rewrite the equation in a way that exposes its true non-linear form where the numerator supplies the parameters needed for affine transformation, and the denominator allows for the non-linear effect of perspective:

$$\begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} = \frac{\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}}{\begin{bmatrix} g & h & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}}$$

This is equivalent to the possibly more familiar, non-vector form of the perspective transform:

$$X = \frac{ax + by + c}{gx + hy + 1}$$

$$Y = \frac{dx + ey + f}{gx + hy + 1}$$

By multiplying each side of the equation by the denominator we get:

$$\begin{aligned} X(gx + hy + 1) &= ax + by + c \\ Y(gx + hy + 1) &= dx + ey + f \end{aligned}$$

and multiplying through by X and Y gives us:

$$gXx + hXy + X = ax + by + c$$
$$gYx + hYy + Y = dx + ey + f$$

Now we isolate the naked X and Y terms on the left:

$$X = ax + by + c - gXx - hXy$$
$$Y = dx + ey + f - gYx - hYy$$

If we add in some zero terms:

$$\begin{aligned} X &= ax + by + \ \ c - 0d + 0e + 0f - Xxg - Xyh \\ Y &= 0a + 0b + 0c + xd + yd + \ \ f - Yxg - Yyh \end{aligned}$$

then it becomes more clear that this is the product of a matrix and a vector:

$$\begin{bmatrix} x & y & 1 & 0 & 0 & 0 & -Xx & -Xy \\ 0 & 0 & 0 & x & y & 1 & -Yx & -Yy \\ & & & & \vdots & & & \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \\ g \\ h \end{bmatrix} = \begin{bmatrix} X \\ Y \\ \vdots \end{bmatrix}$$

and we have reached the previously mysterious point that Criminisi leaps to in a single step:

$$\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -X_1x_1 & -X_1y_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -Y_1x_1 & -Y_1y_1 \\ x_2 & y_2 & 1 & 0 & 0 & 0 & -X_2x_2 & -X_2y_2 \\ 0 & 0 & 0 & x_2 & y_2 & 1 & -Y_2x_2 & -Y_2y_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_n & y_n & 1 & 0 & 0 & 0 & -X_nx_n & -X_ny_n \\ 0 & 0 & 0 & x_n & y_n & 1 & -Y_nx_n & -Y_ny_n \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \\ g \\ h \end{bmatrix} = \begin{bmatrix} X_1 \\ Y_1 \\ X_2 \\ Y_2 \\ \vdots \\ X_n \\ Y_n \end{bmatrix}$$

The Solution

The form of this equation is Ax=b, and it can be solved by several methods that amount to a least squares estimation of the parameter vector x that satisfies the linear relationship between the matrix A and the vector of output coordinates B. The simplest, although not the most numerically stable, is to use the pseudo-inverse:

$$A\lambda = B$$

$$A^TA\lambda = A^TB$$

$$\lambda = (A^TA)^{-1}A^TB$$

The rest of the Criminisi paper discusses issues of measurement noise and estimation uncertainty that are important when choosing the right estimation method for a given application.

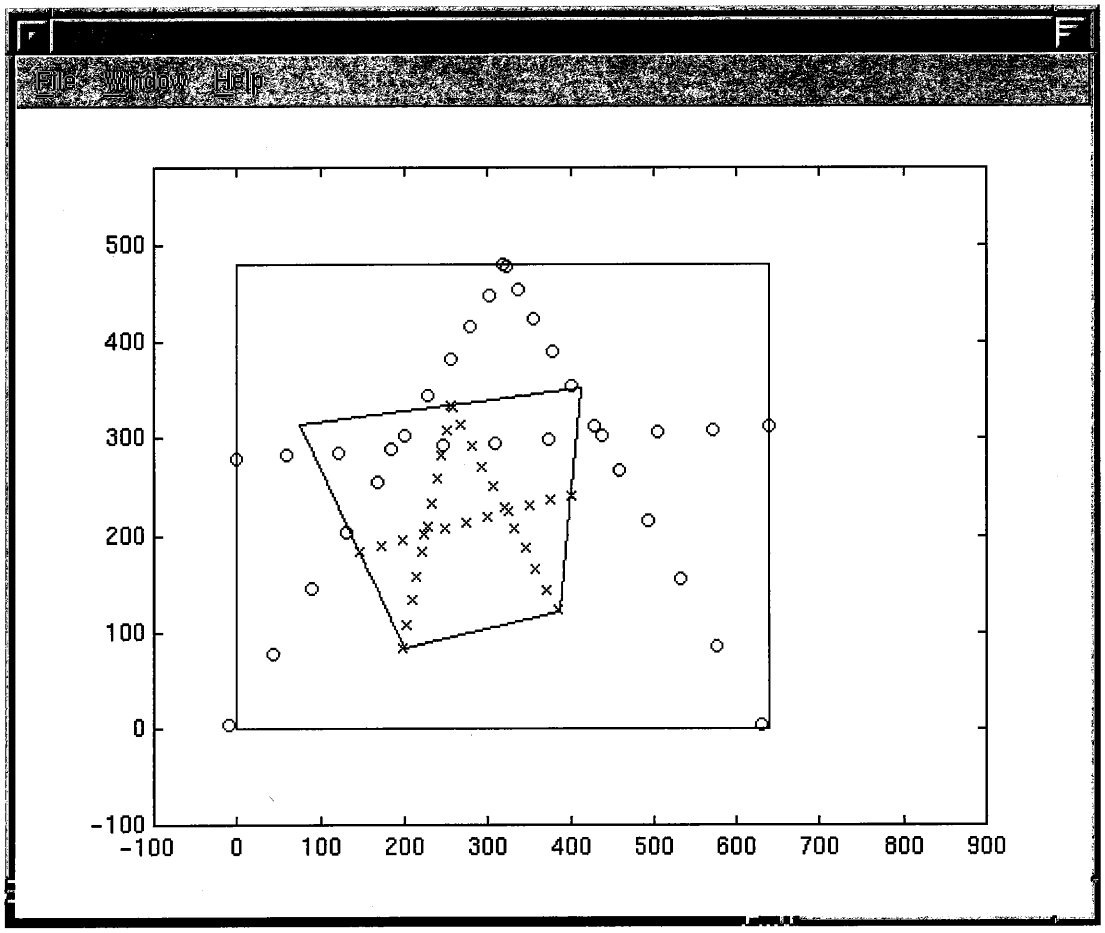
The Code

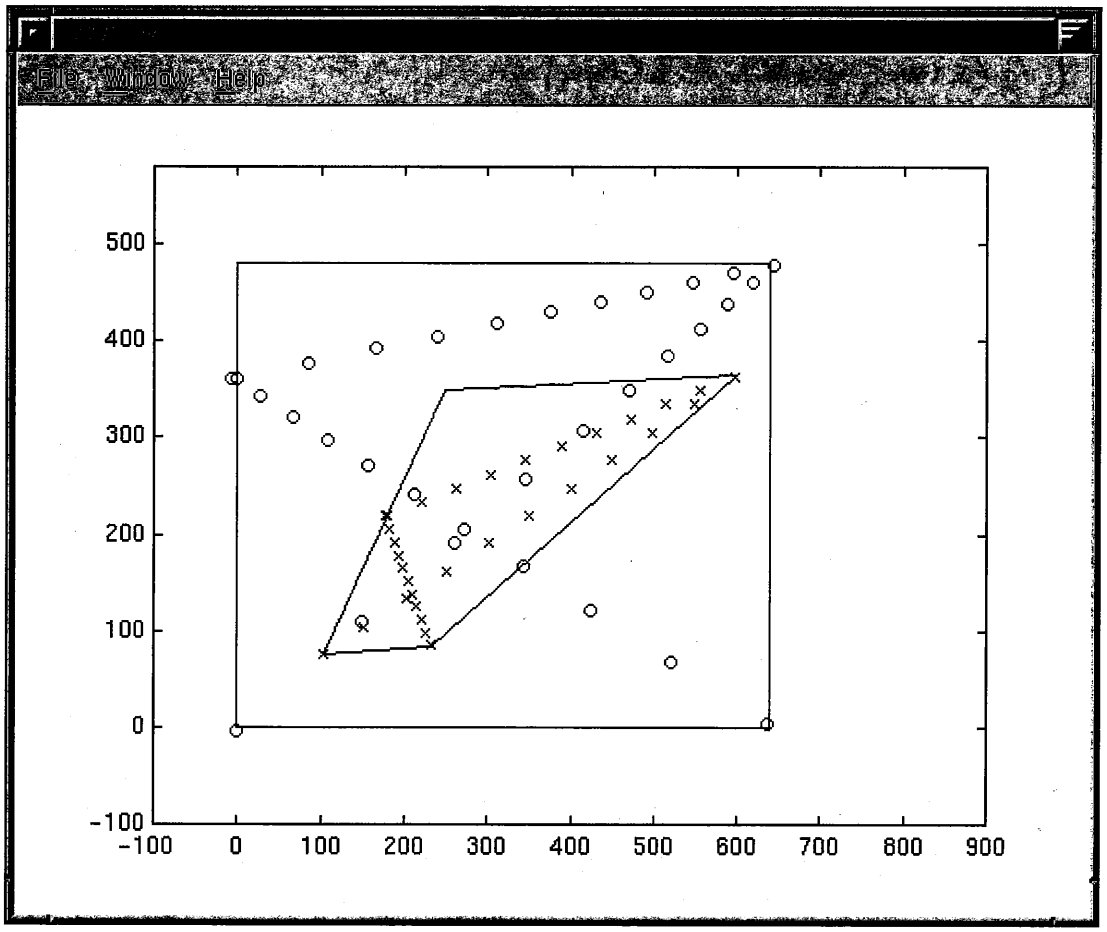
This code implements this solution in Matlab for the case of mapping an user selected quadrilateral onto a rectangle.

```
hold off
axis([0 640 0 480 ])
[X,Y] = ginput(4)
plot([X;X(1)],[Y;Y(1)],'r')
hold
plot([0 0 640 640 0], [0 480 480 0 0],'b')
axis([ -100 900 -100 580 ])
Xp=[0;   0; 640; 640];
Yp=[0; 480; 480;   0];
B = [ X Y ones(size(X)) zeros(4,3)       -X.*Xp -Y.*Xp ...
      zeros(4,3)        X Y ones(size(X)) -X.*Yp -Y.*Yp ];
```

```
B = reshape (B', 8 , 8 )';
D = [ Xp , Yp ];
D = reshape (D', 8 , 1 );
l = inv(B' * B) * B' * D;
A = reshape([l(1:6)' 0 0 1 ],3,3)';
C = [l(7:8)' 1];
while 1 ,
[x1,y1]=ginput(1);
[x2,y2]=ginput(1);
for u=0:.1:1,
  x = u*x1+(1-u)*x2;
  y = u*y1+(1-u)*y2;
  plot(x,y,'xr'); t=A*[x;y;1]/(C*[x;y;1]);plot(t(1),t(2),'ob')
  end
end
```

Screen Shots





Acknowledgments

Many thanks to Stephen Intille , Lee Campbell, Dave Berger, Sumit Basu, and Tony Jebara for their advise on this topic.

Go Up a Level

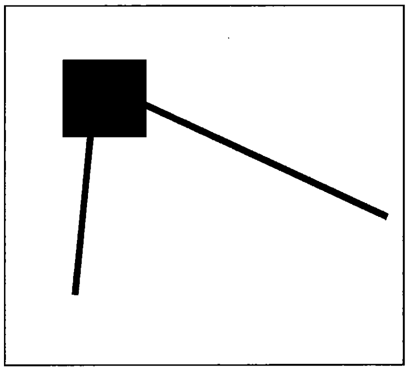
"Christopher R. Wren" <wren@media.mit.edu>

$Id: interpolator.html,v 1.2 1998/12/10 16:15:28 cwren Exp cwren $

Harris Corners: The Basic Idea

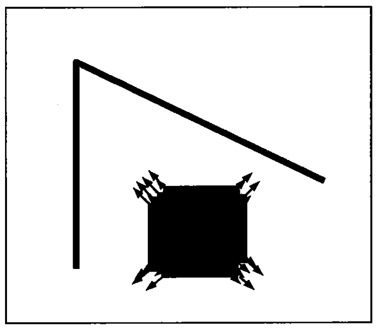
We should easily recognize the point by looking through a small window

Shifting a window in any direction should give a large change in intensity

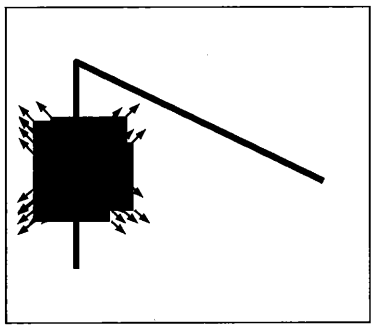


NOKIA

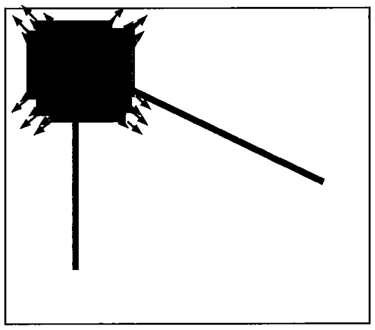
Harris Detector: Basic Idea



"flat" region:
no change in all directions



"edge":
no change along the edge direction



"corner":
significant change in all directions

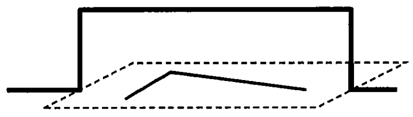
NOKIA

Harris Detector: Mathematics

Window-averaged change of intensity for the shift [*u,v*]:

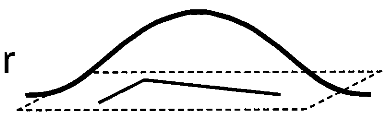
$$E(u,v) = \sum_{x,y} \underbrace{w(x,y)}_{\text{Window function}} \Big[\underbrace{I(x+u, y+v)}_{\text{Shifted intensity}} - \underbrace{I(x,y)}_{\text{Intensity}} \Big]^2$$

Window function $w(x,y)$ =



1 in window, 0 outside

or



Gaussian

Harris Detector: Mathematics

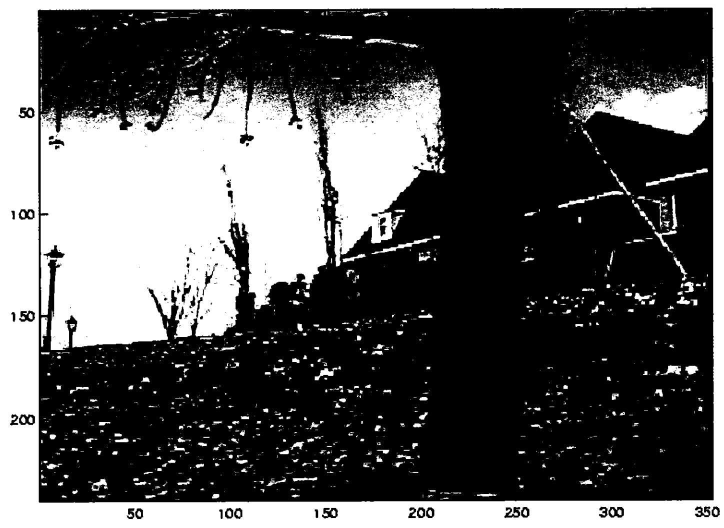
Expanding *E(u,v)* in a 2nd order Taylor series expansion, we have, for small shifts [*u,v*], a *bilinear* approximation:

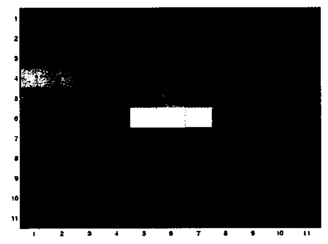
$$E(u,v) \cong [u, v] \; M \begin{bmatrix} u \\ v \end{bmatrix}$$

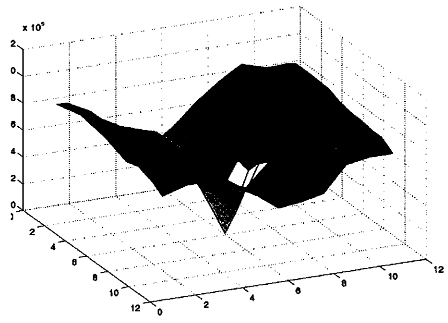
where *M* is a 2×2 matrix computed from image derivatives:

$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

Eigenvalues λ_1, λ_2 of M at different locations



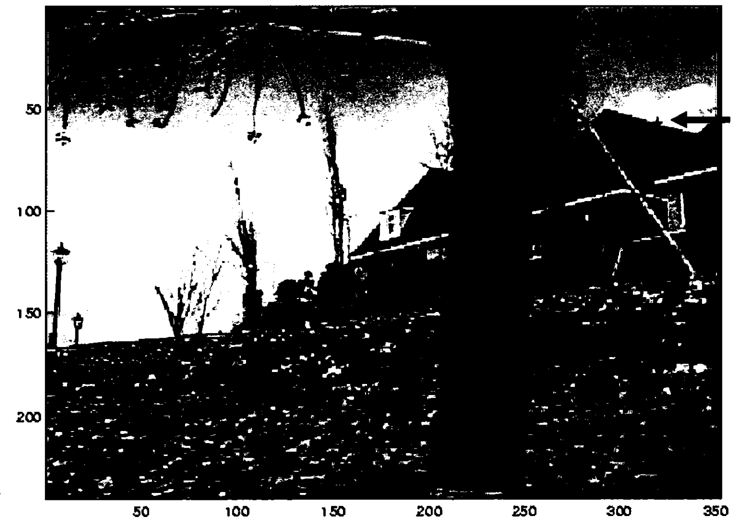


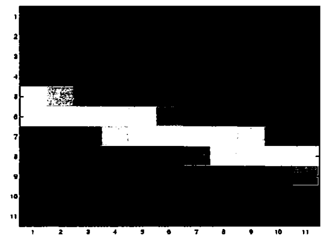


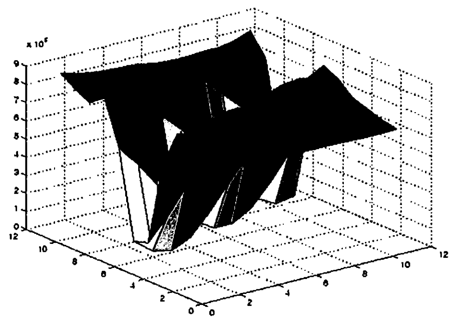
λ_1 and λ_2 are large

NOKIA

Eigenvalues λ_1, λ_2 of M at different locations



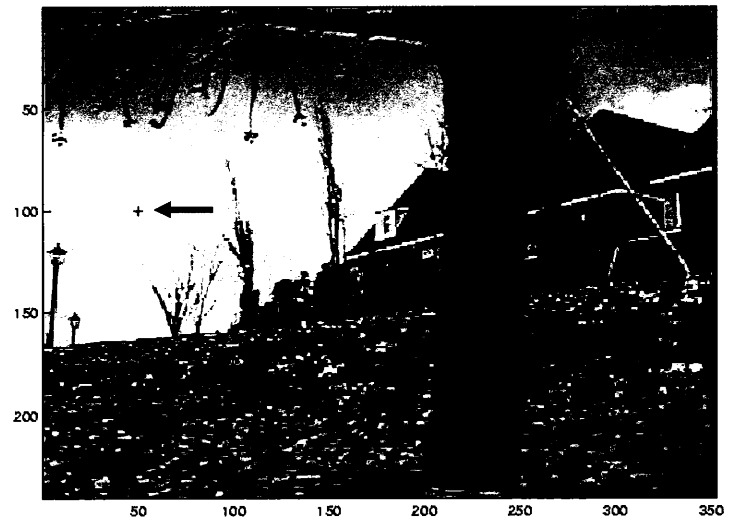


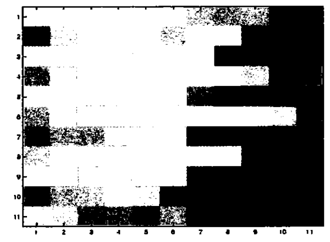


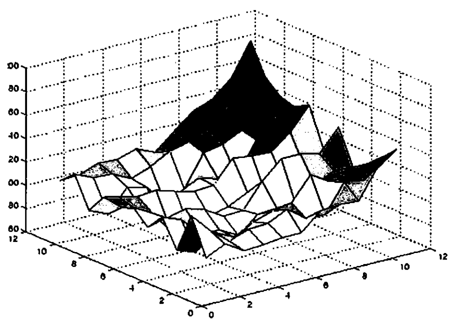
large λ_1, small λ_2

NOKIA

Eigenvalues λ_1, λ_2 of M at different locations





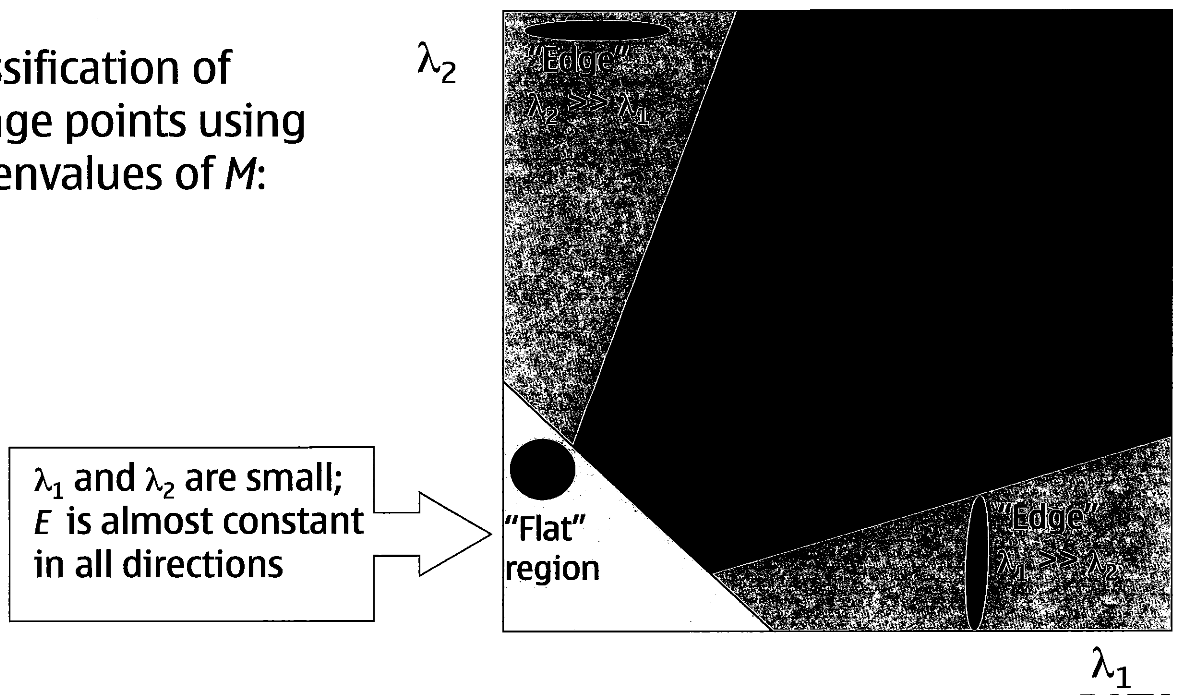


small λ_1, small λ_2

NOKIA

Harris Detector: Mathematics

Classification of image points using eigenvalues of M:



NOKIA

Harris Detector: Mathematics

Measure of corner response:

$$R = \det M - k\left(\operatorname{trace} M\right)^2$$

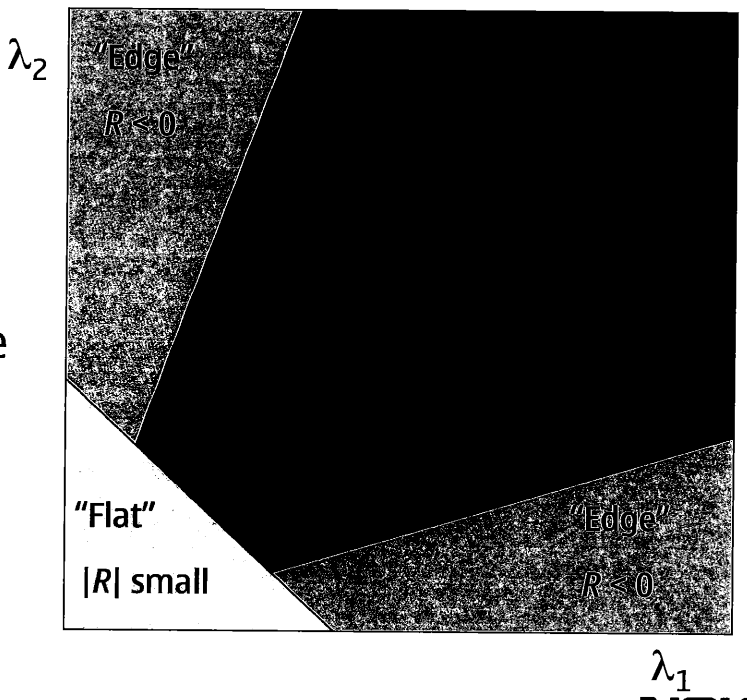
$$\det M = \lambda_1 \lambda_2$$
$$\operatorname{trace} M = \lambda_1 + \lambda_2$$

(*k* – empirical constant, *k* = 0.04-0.06)

NOKIA

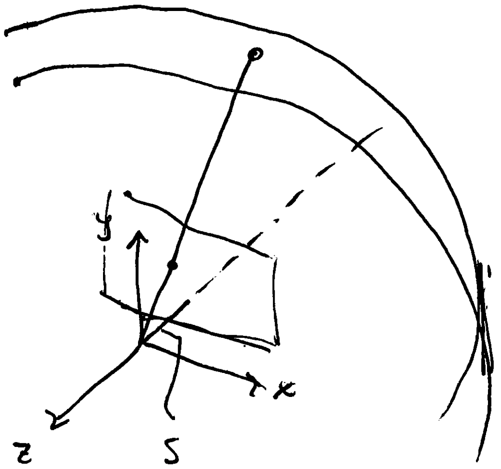
Harris Detector: Mathematics

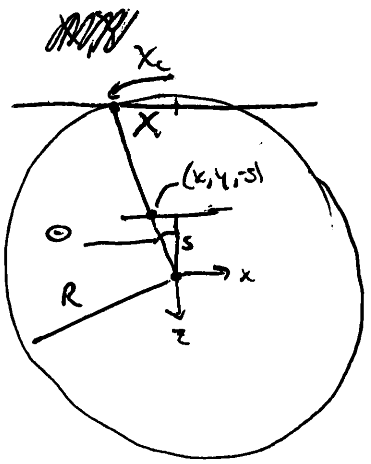
- R depends only on eigenvalues of M
- R is large for a corner
- R is negative with large magnitude for an edge
- $|R|$ is small for a flat region

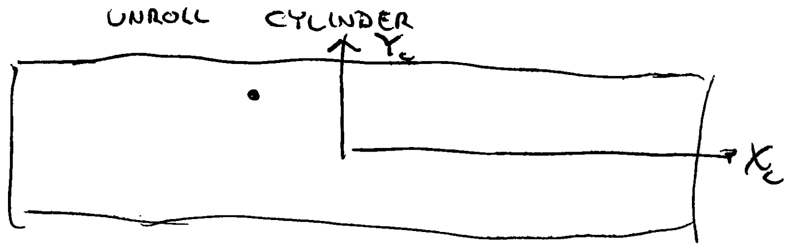


NOKIA

Cylindrical Projection







$$\frac{X}{R} = \frac{x}{\sqrt{x^2+s^2}} \qquad \frac{Y}{R} = \frac{y}{\sqrt{x^2+s^2}}$$

$$X_c = R\sin^{-1}\left(\frac{X}{R}\right) \qquad Y_c = Y$$

ALGORITHM

① START WITH LARGE ARRAY (X_c, Y_c)

② CONVERT TO (X, Y) with $X = R\sin\left(\frac{X_c}{R}\right)$ $Y = Y_c$

③ CONVERT TO (x, y) with $x = \dfrac{\frac{s}{R}X}{\sqrt{1-\frac{X^2}{R^2}}}$ $y = \dfrac{\frac{s}{R}Y}{\sqrt{1-\frac{Y^2}{R^2}}}$

④ IF (x, y) with boundaries of image, assign pixel value at (x, y) to large array pixel (X_c, Y_c).

⑤ wash, rinse, repeat

PURE TRANSLATION OF THE IMAGES CAN NOW BE USED TO ALIGN.