Edge detection and patches

Francisco Gómez J
Computer Vision
MMS
U Central and UJTL

Edge

- Edge point in the image where intensities are changing rapidly
- Sobel operator does not provide an edge it provides the magnitude of the gradient in each pixel. <u>How we can extract the edge?</u>

-1	0	+1
-2	0	+2
-1	0	+1

x filter

+1	+2	+1
0	0	0
-1	-2	-1

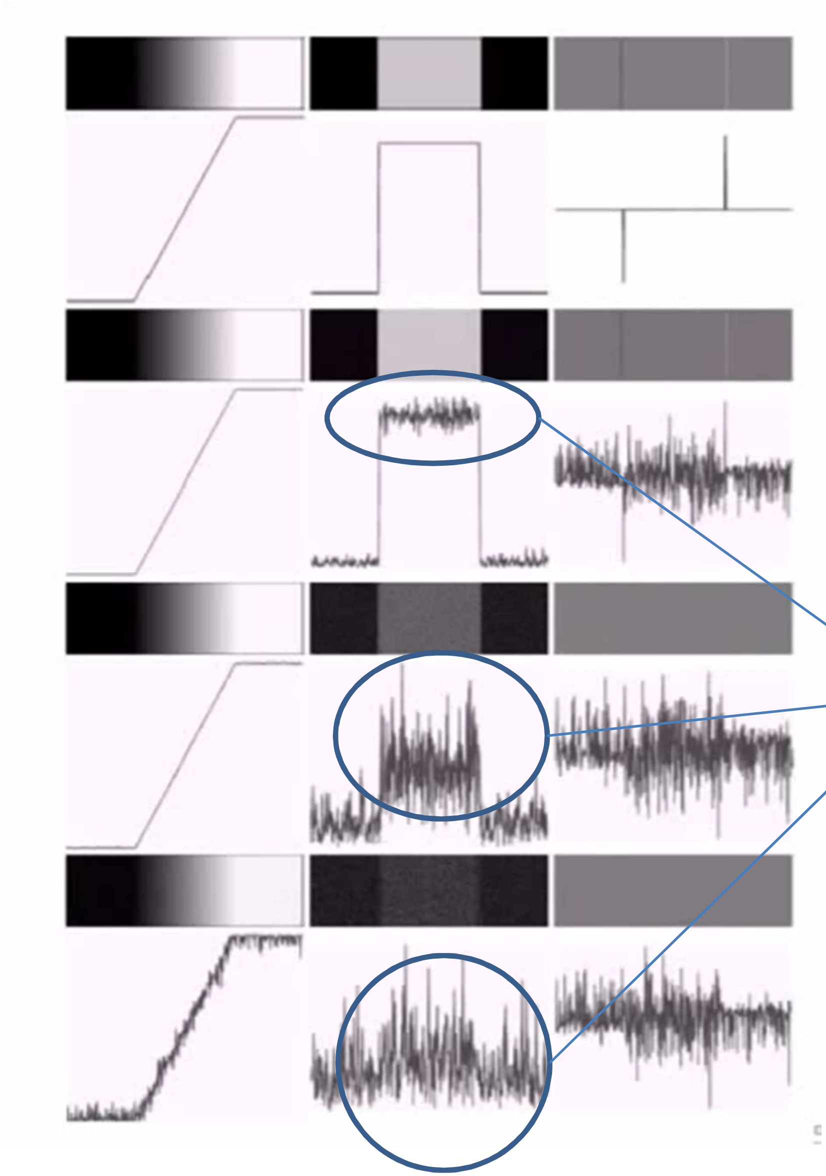
y filter

$$|\nabla f| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

Gradient magnitude is not binary

Thresholding is not enought

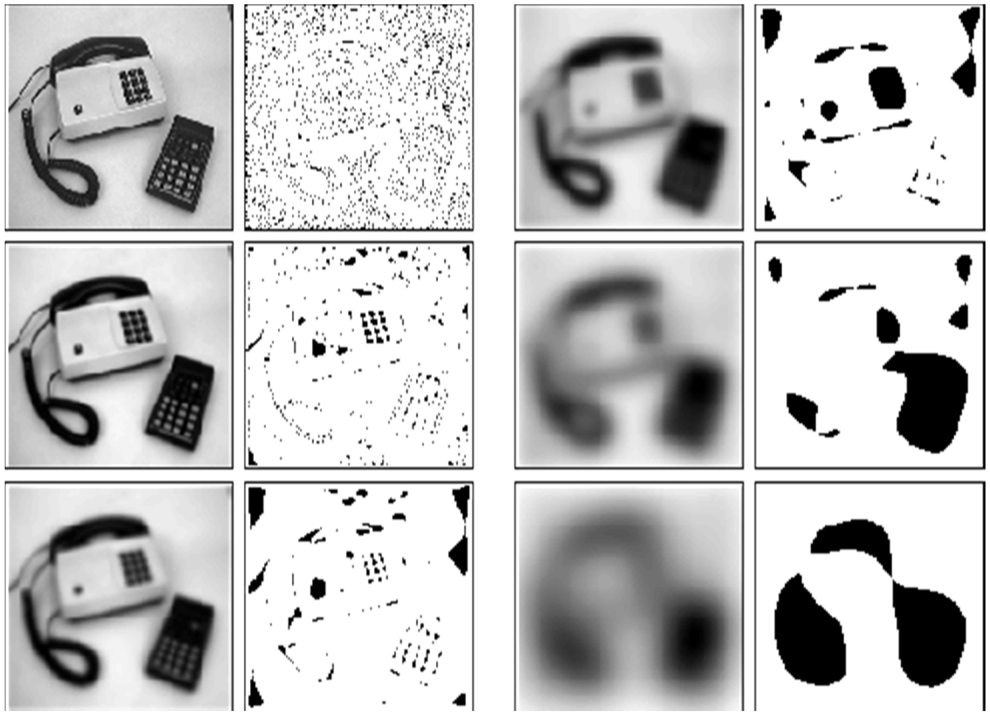
Derivatives amplifies noise



Noise is amplified by derivatives

Smooth before taking
Derivatives!!

Scale space operator

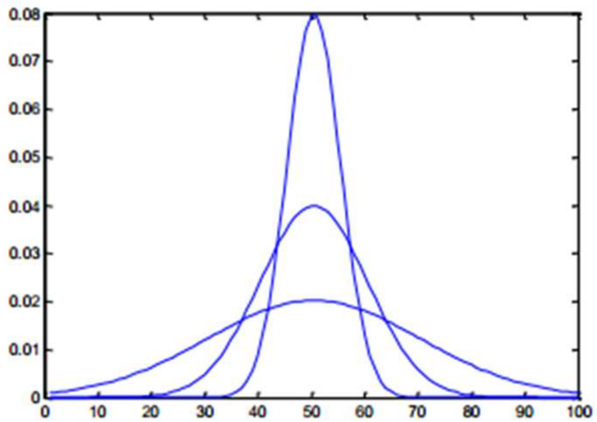


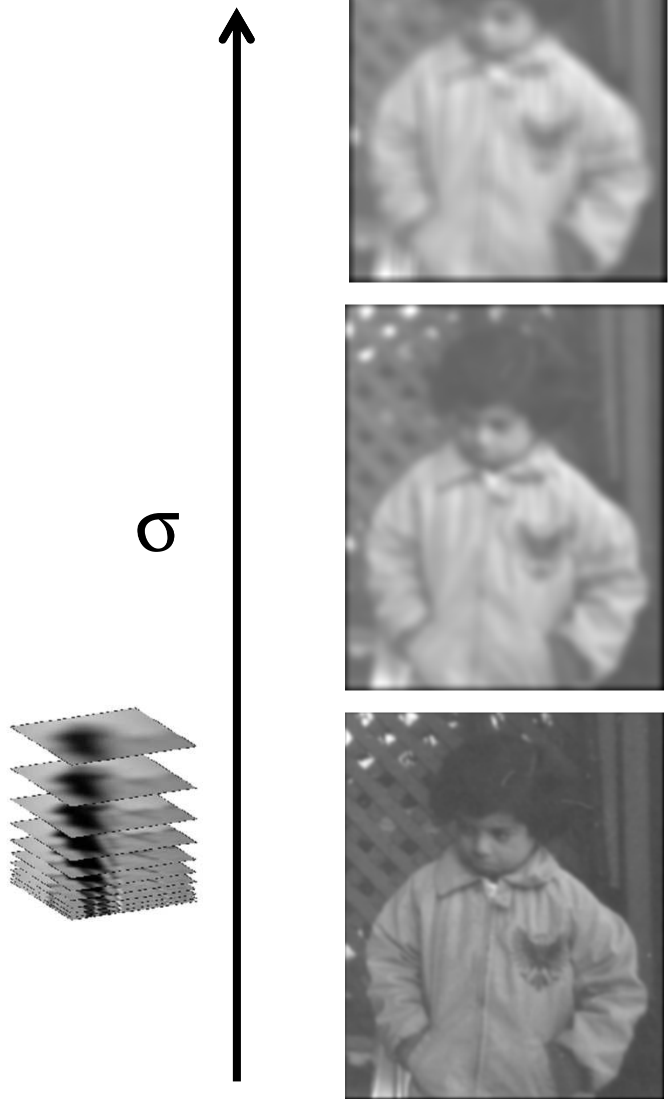
How much the image should be smoothed?

Scale space

Images created by applying a series of operators at different scales

$$G_\sigma(x,y) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/(2\sigma^2)}$$





Convolving with a Gaussian blur helps to remove structures smaller that

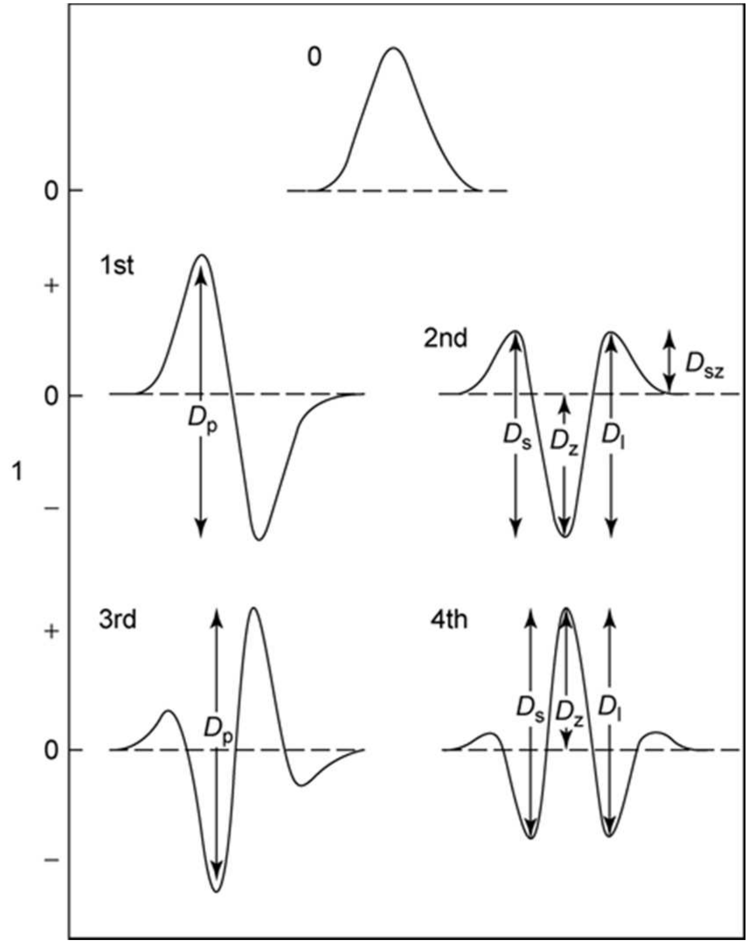
Derivatives of a Gaussian

Gradient of smoothed image

$$\nabla[G_\sigma * I] = [\nabla G_\sigma] * I$$

$$\nabla G_\sigma = \begin{pmatrix} \dfrac{\partial G_\sigma}{\partial x} & \dfrac{\partial G_\sigma}{\partial y} \end{pmatrix}^T$$

$$= \begin{pmatrix} -x & -y \end{pmatrix} \frac{1}{\sigma^3} \exp\left(-\frac{x^2 + y^2}{2\sigma^2} \right)$$



Edge points

- Properties
  - Good detection (minimize the probability of detecting false edges and missing real edges)
  - Good localization (edges should be detected close to real edges)
  - Single response (just one point for each true edge point)
- The derivative of the Gaussian is a good aproximation to this operator!!

Algorithm

1. Convolve image with derivative of Gaussian operators $\left(\frac{\partial G_\sigma}{\partial x} \quad \frac{\partial G_\sigma}{\partial y}\right)^T$
2. Find the gradient direction in each pixel (atan2(Gy,Gx))
3. Quantize into 0, 45, 90 and 135 degrees directions
4. If magnitude of gradient is larger than the two neighbors along this direction, it is a candidate edge point

Edge linking

- To recognize objects it would be desirable to have connected curves or lines
- But some point maybe weak and maybe missed, aka, broken curve
- Solution:
  - First use a high threshold to capture strong edge pixels
  - Links points into a contourn using a lower threshold («hystheresis»)

Algorithm

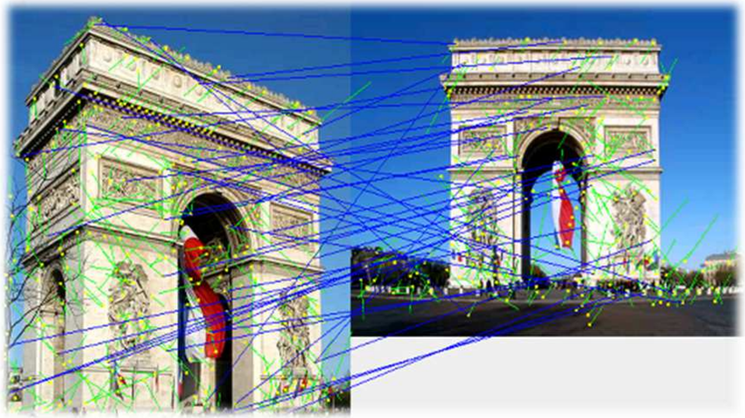
- Algorithm
  - Find all edge points greater than t_{high}
  - From each strong edge point, follow the chains of connected edge points in both directions perpendicular to the edge normal
  - Mark all points greater than t_{low}

Demo

```
[E,thresh]=edge(I, ‘canny’, thresh, sigma);
for s = 0.5:0.5:5
        E = edge(I, 'canny', [], s);
        imshow(E);
        pause;
end
for tHigh = 0.05:0.05:0.4
                E = edge(I, 'canny', [0.4*tHigh tHigh], 1.5);

                imshow(E);
                pause;
end
```

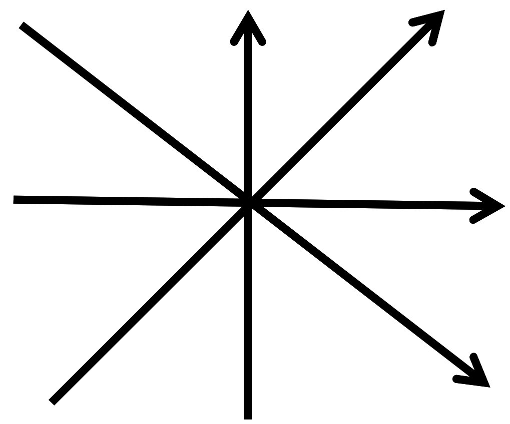
Point and patch features

- How to find interesting points in the image?
  - These features can be used for object recognition
  - Can be used to track objects in motion
- Point features are locally unique:
  - Good: Ej. Corners
  - Bad: Flat regions or long edges



Moravec

- Find points in which local variances in different directions (vertical, horizontal and diagonal) are high
- Algorithm
  - V1=Variance for pixels I(x-w,y):I(x+w,y)
  - V2=Variance for pixels I(x,y-w):I(x,y-w)
  - V3=Variance for pixels I(x-w,y-w):I(x-w,y-w)
  - V4=Variance for pixels I(x+w,y-w):I(x-w,y+w)

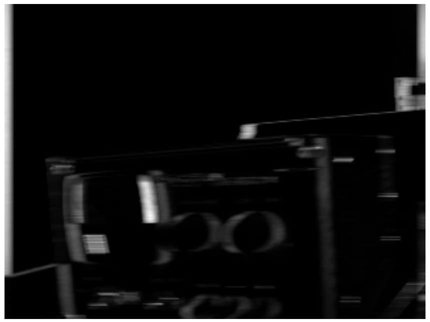


- Interest value min(v1,v2,v3,v4)

Implementing Moravec operator

The variance can be estimated by using the

$$\sigma^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2$$



$$= \frac{1}{N}\sum_{i=1}^{N}\left(x_i^{\,2} - 2\mu x_i + \mu^2\right)$$

$$= \frac{1}{N}\sum_{i=1}^{N}x_i^{\,2} - \frac{2\mu}{N}\sum_{i=1}^{N}x_i + \frac{\mu^2}{N}\sum_{i=1}^{N}1$$

$$= \frac{1}{N}\sum_{i=1}^{N}x_i^{\,2} - 2\mu^2 + \mu^2$$

$$= \frac{1}{N}\sum_{i=1}^{N}x_i^{\,2} - \mu^2$$

Implementing movarec

```
function Ir = moravev(I,N)

    aveh = ones(1,N)/N;
    var1 = applyFilter(I,aveh);
    avev = ones(N,1)/N;
    var2 = applyFilter(I,avev);
    aved1 = eye(N,N)/N
    var3 = applyFilter(I,aved1);
    aved2 = fliplr(aved1)
    var4 = applyFilter(I,aved1);
    Ir = min(min(var1,var2),min(var3,var3));

function varh = applyFilter(I,ave)

    % computes the means
    u = imfilter(I,ave);
    % computes the means squares
    u2 = u.*u;

    % computes the image square
    Isq = I.*I;
    % Sum of the squares
    u2ave = imfilter(Isq,ave);

    varh = u2ave - u2;
```

Movarec results

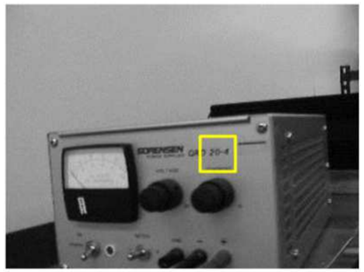
Feature detection

- We want to MATCH a patch from image I_0 to image I_1
- If we assume that intensities does not change between frames, and there is only translational motion

Translation

$$I_1(\mathbf{x}_i + \mathbf{u}) = I_0(\mathbf{x}_i)$$

Equal intensity



Solution

- We can find the displacement **u** that minimize the sum of the square differences

$$E_{\mathrm{WSSD}}(\boldsymbol{u}) = \sum_i w(\boldsymbol{x}_i)[I_1(\boldsymbol{x}_i + \boldsymbol{u}) - I_0(\boldsymbol{x}_i)]^2,$$

How to choose patches?

- For the correct value **u** we want that **E(u)** have a minimum
- The information in the patch determines the stability
  - Featureless patches cannot be determined uniquely
- Then how stable is a patch?

$$E_{\mathrm{AC}}(\Delta \boldsymbol{u}) = \sum_i w(\boldsymbol{x}_i)[I_0(\boldsymbol{x}_i + \Delta \boldsymbol{u}) - I_0(\boldsymbol{x}_i)]^2$$

- If this surface has a minimum then I0 is a good patch

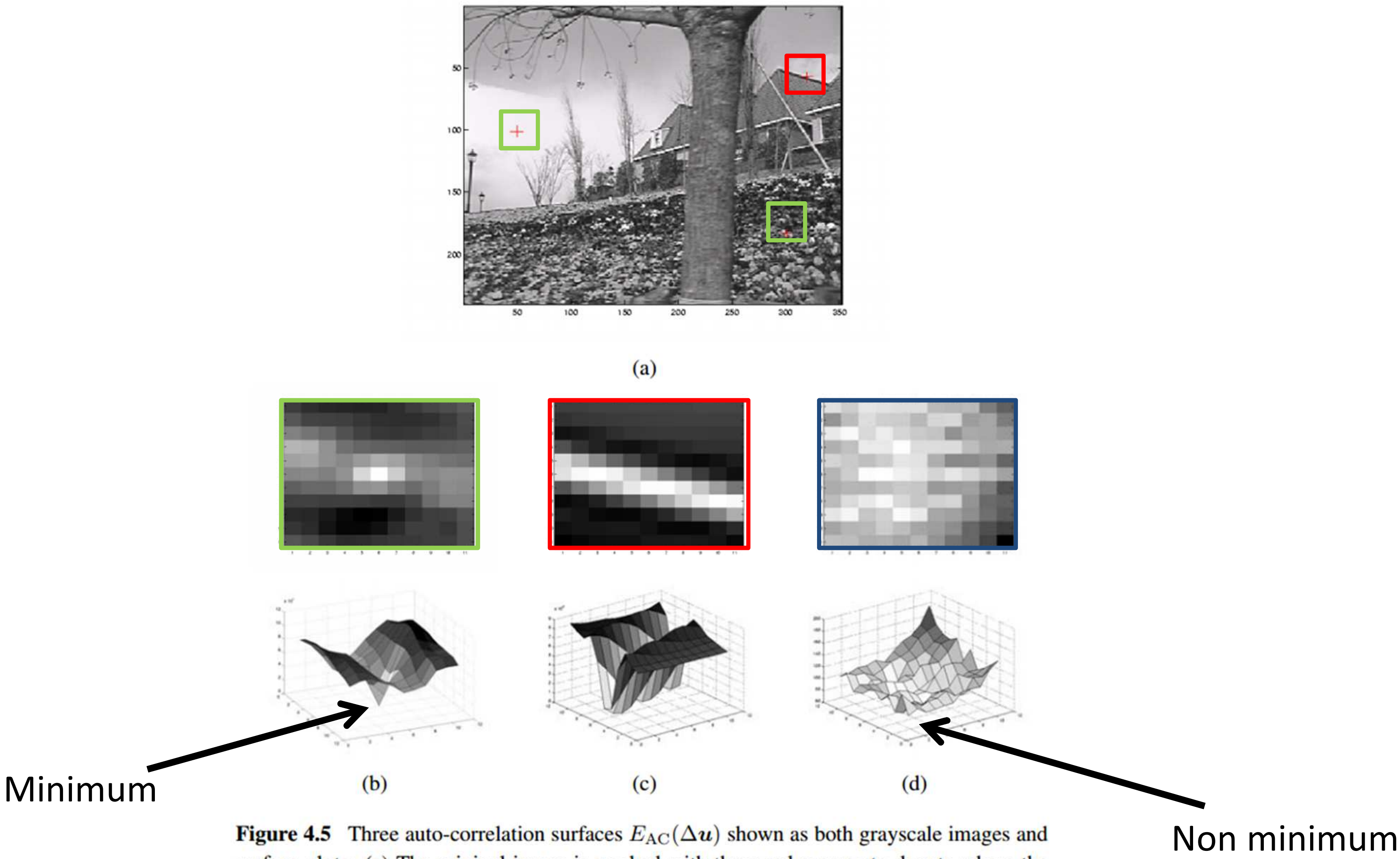


Figure 4.5 Three auto-correlation surfaces $E_{\mathrm{AC}}(\Delta \boldsymbol{u})$ shown as both grayscale images and surface plots: (a) The original image is marked with three red crosses to denote where the auto-correlation surfaces were computed; (b) this patch is from the flower bed (good unique minimum); (c) this patch is from the roof edge (one-dimensional aperture problem); and (d) this patch is from the cloud (no good peak). Each grid point in figures b–d is one value of $\Delta \boldsymbol{u}$.

- Using the Taylor series expansion

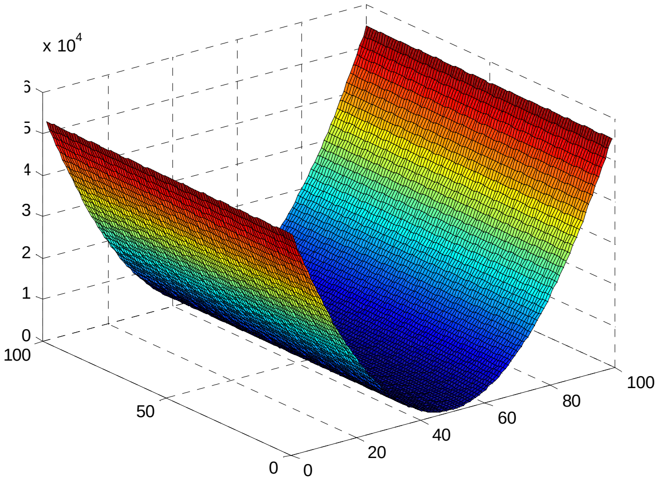
$$
\begin{aligned}
E_{\mathrm{AC}}(\Delta \boldsymbol{u}) &= \sum_i w(\boldsymbol{x}_i)[I_0(\boldsymbol{x}_i + \Delta \boldsymbol{u}) - I_0(\boldsymbol{x}_i)]^2 \\
&\approx \sum_i w(\boldsymbol{x}_i)[I_0(\boldsymbol{x}_i) + \nabla I_0(\boldsymbol{x}_i) \cdot \Delta \boldsymbol{u} - I_0(\boldsymbol{x}_i)]^2 \\
&= \sum_i w(\boldsymbol{x}_i)[\nabla I_0(\boldsymbol{x}_i) \cdot \Delta \boldsymbol{u}]^2 \\
&= \Delta \boldsymbol{u}^T \boldsymbol{A} \Delta \boldsymbol{u},
\end{aligned}
$$

Correlation

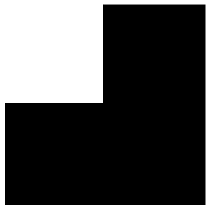
$$\nabla I_0(\boldsymbol{x}_i) = \left(\frac{\partial I_0}{\partial x}, \frac{\partial I_0}{\partial y}\right)(\boldsymbol{x}_i) \qquad \boldsymbol{A} = w * \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

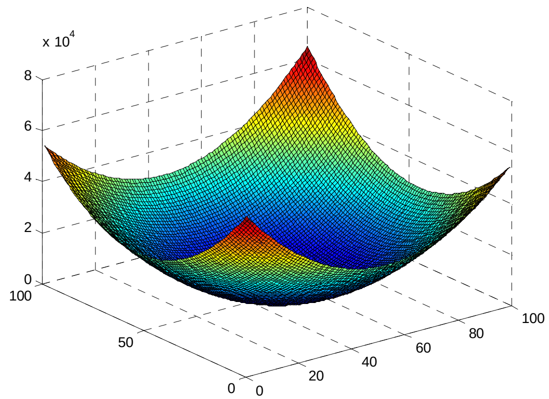
Autocorrelation matrix

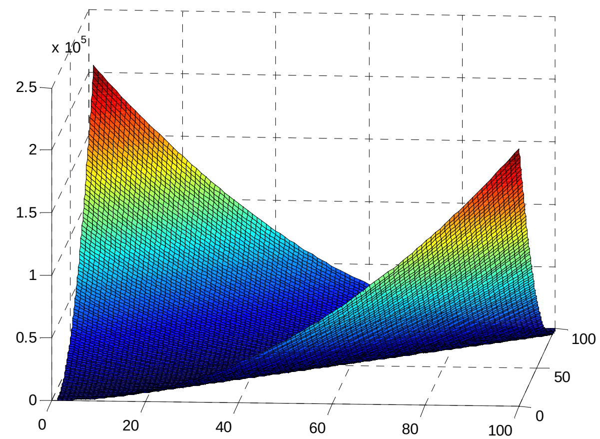
Demo



```
EAc = zeros(100,100);
for ux=1:100
   for uy=1:100
                  A = [12 1; 1 12];
                  %[0 0; 0 22];
                  %[21 -21;-21 21]
         EAc(ux,uy) = [ux-50 uy-50]*A*[ux-50 uy-50]';
   end
end

figure
surf(EAc);
```

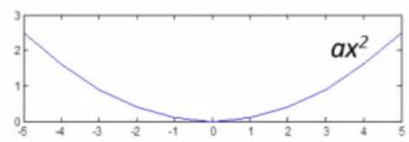


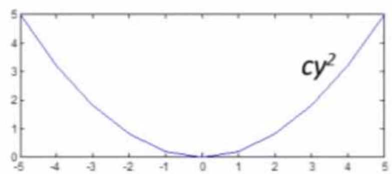




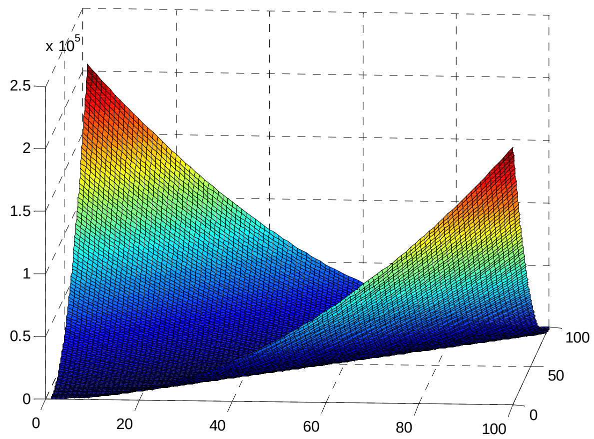
- $\Delta u^T A \Delta u,$ is a second order polynomial

$$\mathbf{A} = \begin{pmatrix} a & b \\ b & c \end{pmatrix}$$



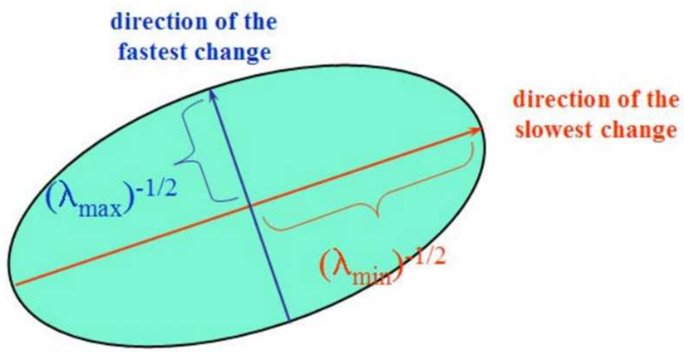


- <u>If a and c are big there is a clear minimum</u>
- If b is not zero it would be flat:



Using eigenvalues

- Using the eigen values of A (Demo…)



- λ_1 and λ_2 are big clear minimun, if one of them is small then flat in that direction…

We don't need to compute the eigen values

- Minimum eigen-value

- Compute this quantity (Shi and Tomasi)

$$\det(\boldsymbol{A}) - \alpha \operatorname{trace}(\boldsymbol{A})^2 = \lambda_0\lambda_1 - \alpha(\lambda_0 + \lambda_1)^2$$

$$\alpha = 0.06$$

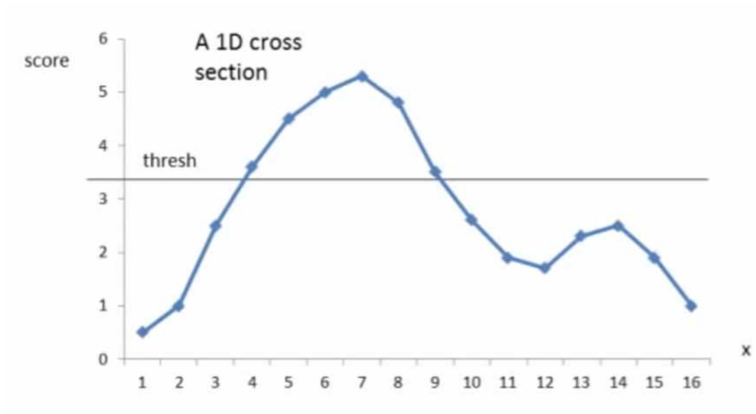
- Use the harmonic mean

$$\frac{\det \boldsymbol{A}}{\operatorname{tr} \boldsymbol{A}} = \frac{\lambda_0\lambda_1}{\lambda_0 + \lambda_1},$$

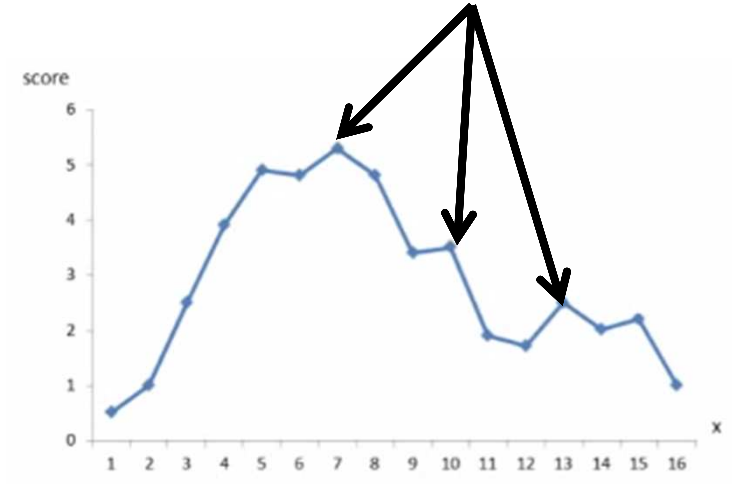
Demo

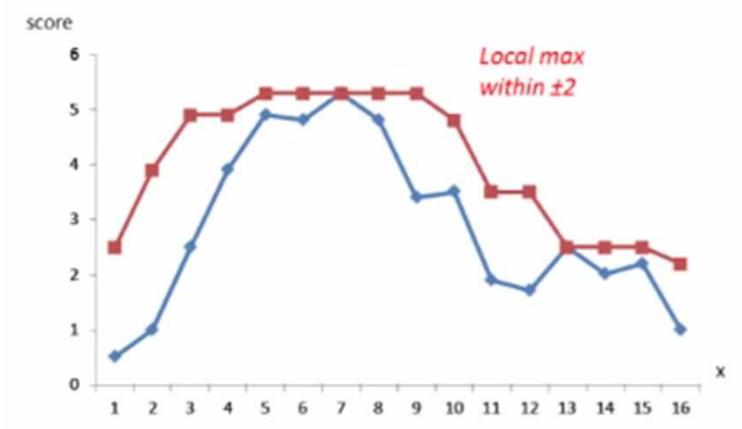
```
clear all
close all

I = double(imread('test000.jpg'));

% gaussian blur
s = 1.0;
I =imfilter(I,fspecial('gaussian',round(6*s),s));

figure
imshow(I,[]);

gx = imfilter(I,[1 -1]);
gy = imfilter(I,[1 -1]');

% compute the square derivatives in each pixel
gxx = gx.*gx;
gyy = gy.*gy;
gxy = gx.*gy;

% now the values are averaged in a neighboorhood
N = 13;
w = ones(N);

A11 = imfilter(gxx,w);
A12 = imfilter(gxy,w);
A22 = imfilter(gyy,w);

detA = A11.*A22 - A12.*A12;
traceA = A11 + A22;
s = detA./traceA;

s(isnan(s)) = 0;

figure
imshow(s,[])
```

Extracting local maxima



Multiple minimums



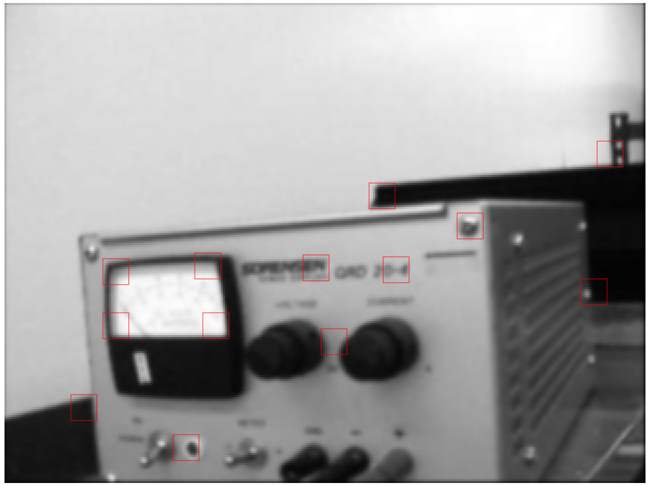


Smax = (S==imdilate(S,ones(N)))

Demo

```
r= N;
Lmax = (s==imdilate(s,strel('disk',2*r)));

% everything near to the border is zero

Lmax(1:N,:) = false;
Lmax(:,1:N) = false;
Lmax(end-N:end,:) = false;
Lmax(:,end-N:end) = false;

[rows cols] = find(Lmax);

vals = s(Lmax)

figure
imshow(I,[])
hold on
 for i=1:size(rows,1)
    if vals(i)>4000
       rectangle('position',[cols(i)-N/2,rows(i)-N/2,N,N],'EdgeColor','r');
    end
 end
```

•



Template matching

- After extracting the patches how to identify it in a new image?

$$E(\mathbf{u}) = \sum_i \left[I_1(\mathbf{x}_i + \mathbf{u}) - I_0(\mathbf{x}_i)\right]^2$$

$$= \sum_i I_1(\mathbf{x}_i + \mathbf{u})^2 - 2\underbrace{\sum_i I_1(\mathbf{x}_i + \mathbf{u}) I_0(\mathbf{x}_i)} + \sum_i I_0(\mathbf{x}_i)^2$$

This is the cross-correlation
This value is high when I0 matches I1

Cross correlation as a distance

- Correlation is a sum of products
- Correlation is a dot product
- It measure the similarity

$$c(x,y) = \sum_{s=-m/2}^{m/2} \sum_{t=-n/2}^{n/2} w(s,t)\, f(x+s, y+t)$$

$$= w(x,y) \otimes f(x,y)$$

$$c = w_1 f_1 + w_2 f_2 + \ldots + w_{mn} f_{mn} = \mathbf{w} \cdot \mathbf{f}$$

$$c = |\mathbf{w}||\mathbf{f}| \cos\theta$$

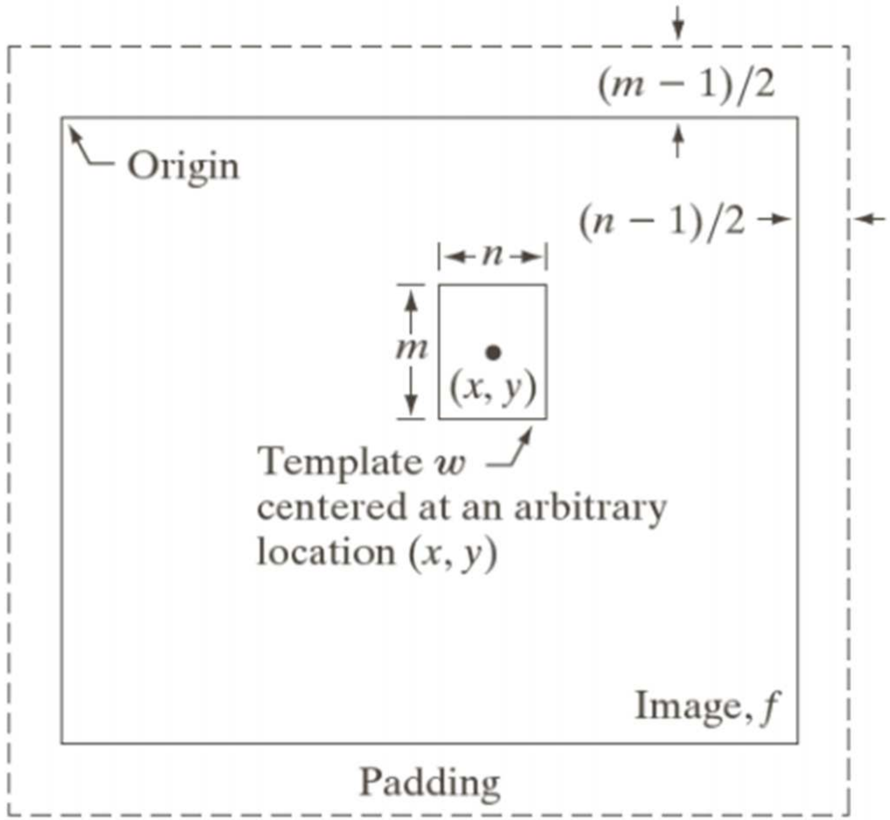
Correlation is matching

- Find w in the image f(x,y)

- The correlation

$$c(x,y)=\sum_{s=-m/2}^{m/2}\sum_{t=-n/2}^{n/2} w(s,t)\, f(x+s,y+t)$$
$$= w(x,y)\otimes f(x,y)$$

- Is high when w matches f



Template matching

- Precision can be improved by substracting the mean

$$c(x,y) = \frac{\sum_{s,t}[w(s,t)-\bar{w}][f(x+s,y+t)-\bar{f}]}{\left\{\sum_{s,t}[w(s,t)-\bar{w}]^2 \sum_{s,t}[f(x+s,y+t)-\bar{f}]^2\right\}^{1/2}}$$

- This is the normalized cross-coefficient (-1,1)

Demo

```
I = imread('test000.jpg');
W = imcrop(I)
c = normxcorr2(w,I);
imshow(c,[])
cmax = max(c(:))
[y2 x2] = find(C==cmax);
```