Robotics

Computer Vision module

Marcello Calisti, PhD

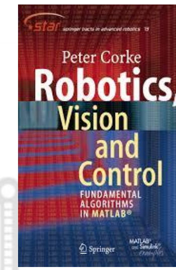
m.calisti@sssup.it

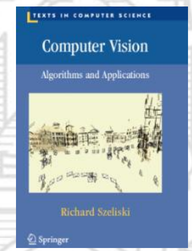
March, 14th, 2016

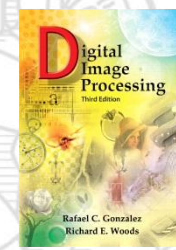
Reference materials and credits

Most of the material presented in these lessons can be find on the brilliant, seminal books on robotics and image analysis reported hereafter:

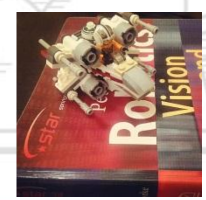
1. P.I. Corke, “Robotics, Vision & Control”, Springer 2011, ISBN 978-3-642-20143-1
2. R. Szeliski, “Computer Vision: Algorithms and Applications”, Springer-Verlag New York, 2010
3. R.C. Gonzalez & R.E. Woods, “Digital Image Processing (3rd edition)”, Prentice-Hall, 2006







Most of the images of these lessons are downloaded from RVC website http://www.petercorke.com/RVC/index.php and, despite they are free to use, they belong to the author of the book.



Outline

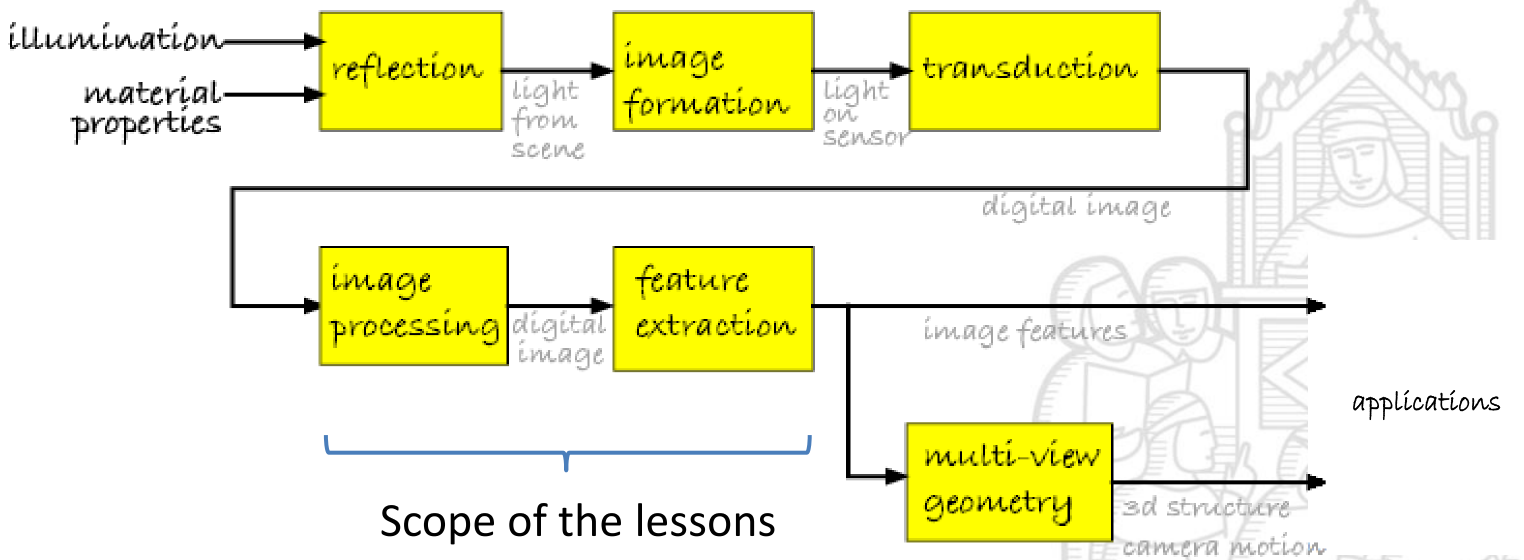
- Part1: Image processing
  - Digital images
  - Punctual, local, global operators
  - Morphological operations
  - Template matching

- Part 2: Features extraction
  - Thresholding
  - Region features
  - Parametric features
  - Point features

Detailed program

- Part1: Image processing
  - Digital image
  - Colour image
  - Monadic operations
  - Gray conversion
  - Lightening, darkening
  - Histogram
  - Sigmoid, power law, piece-wise transform
  - Histogram equalization
  - Thresholding, posterization
  - Diadic operations
  - Green screen and HDR
  - Background subtraction
  - Background estimation
  - Spatial operators
  - Convolution & boundary effect
  - Smoothing and kernel masks
  - Edge detection
  - Gradient (magnitude & direction)
  - DoG and noise
  - Canny Edge Detection
  - Laplacian mask & LoG
  - Template matching
  - Similarity measures
  - Census and rank
  - Morphological operators
  - Erosion & Dilation
  - Closing & Opening
  - Basic geometric manipulations
  - DLR example
- Part 2: Features extraction
  - Region features
  - Otsu
  - Local thresholding
  - Niblack
  - Colour classification
  - K-mean clustering
  - Labeling
  - 4- & 8-neighbourhood
  - <u>MSA (motivated student algorithm)</u>
  - Graph-based segmentation
  - Concise representation: bounding boxes
  - Moments
  - Equivalent ellipses
  - Features invariance
  - Boundary representation
  - Hough transformation
  - Accumulation matrix and threshold
  - Point features
  - Common interest metrics
  - Optic flow & aperture
  - Lucas-Kanade solution
  - Bee-inspired navigation

Overall computer vision process and scope of the lessons



Part 1

Image processing

Introduction

The first theory on visual processing was **emission (or extramission) theory** which suggested that vision occurs when **rays emanate from the eyes** and are intercepted by visual objects. These rays, interacting with visible objects, produce the perception of the objects.

Our eye is like a torch

Sounds ridiculous? Not for half of you.

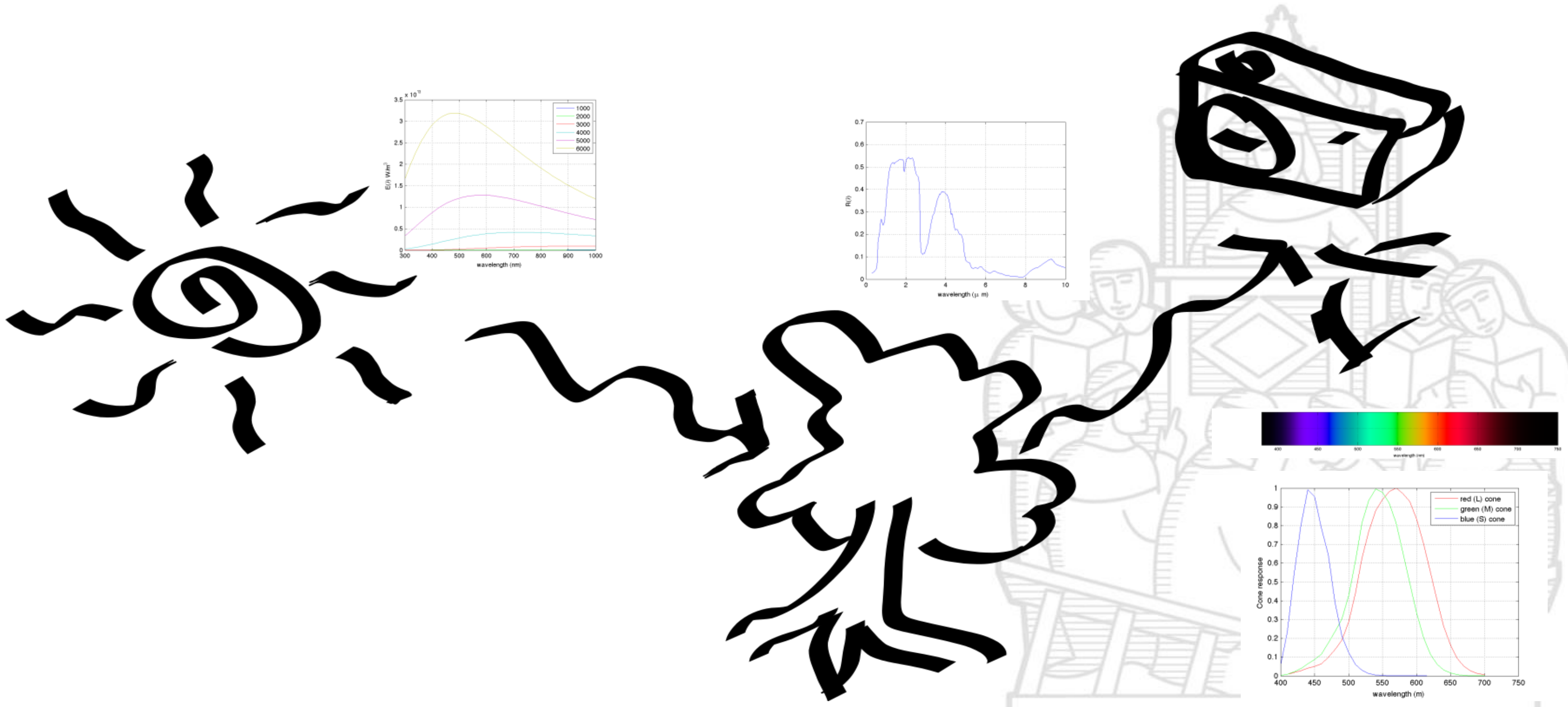
A survey conducted by Winer *et al.* stated that 50% of the adults considered in the survey believe in extramission theory

Winer, G. A., Cottrell, J. E., Gregg, V., Fournier, J. S., & Bica, L. A. (2002). Fundamentally misunderstanding visual perception: Adults' beliefs in visual emissions. *American Psychologist, 57,* 417-424

Gerald A. Winer

Introduction

Today we consider that light from an **illuminant** falls on the scene, some of which is **reflected** into the eye of the observer to create a **perception** about the scene. The amount of light that reaches the eye, or the camera, is a function of the illumination impinging on the scene and the material property known as reflectivity.



Introduction

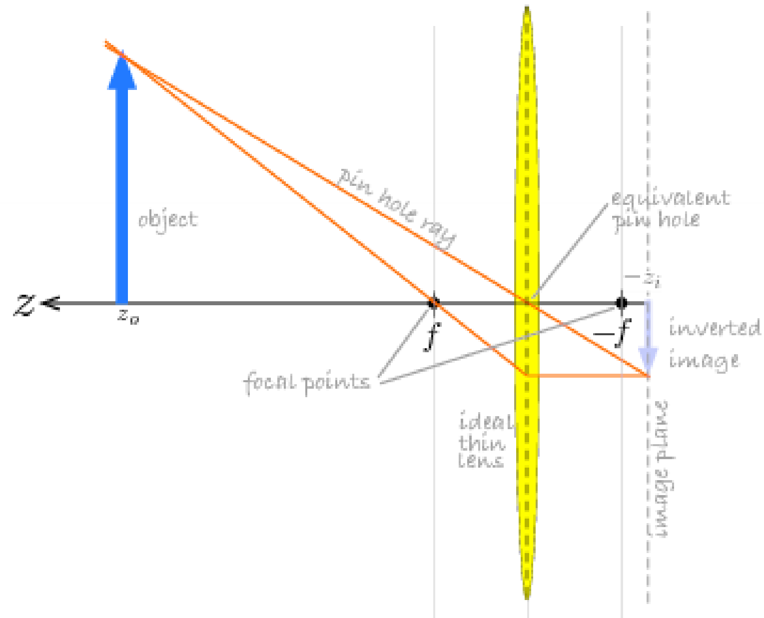
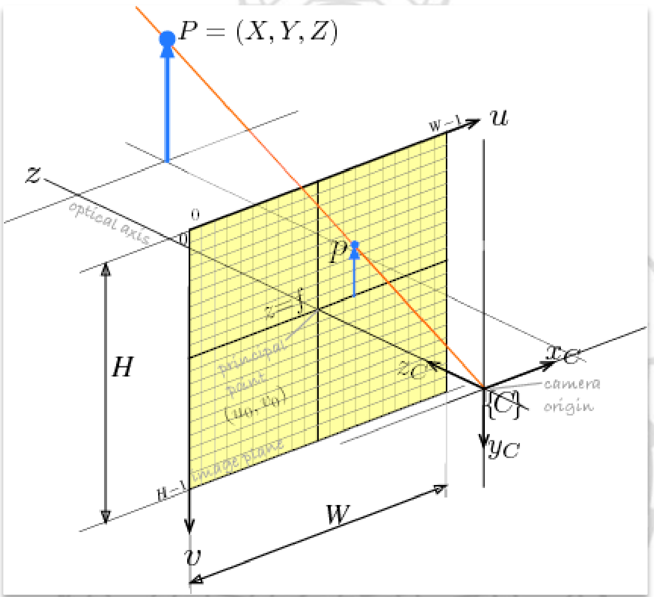


Image formation

$$\frac{1}{z_0} + \frac{1}{z_i} = \frac{1}{f}$$



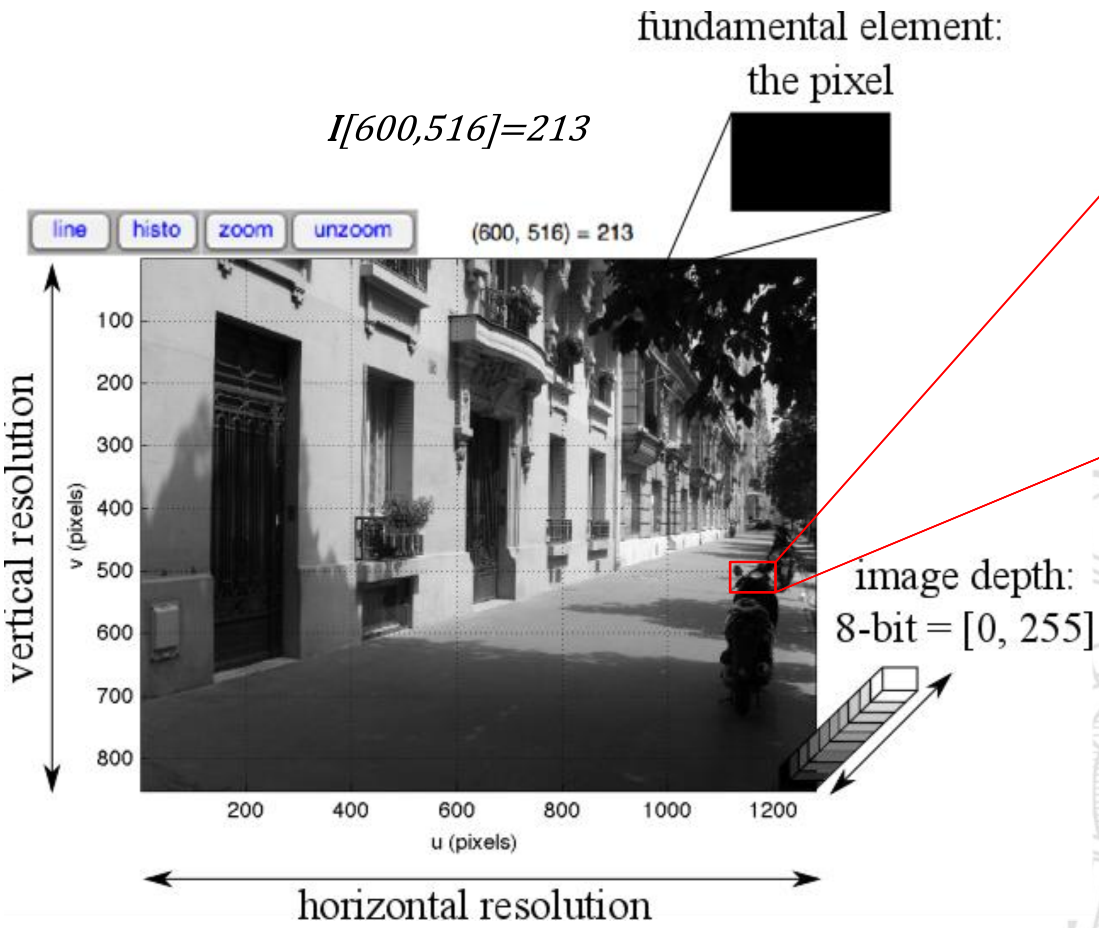
Perspective image model

Where point in world coordinate $P = (X, Y, Z)$ projects into a point in the image plane $p = (x, y)$

$$x = f\frac{X}{Z}, y = f\frac{Y}{Z}$$

The points on the image plane, in our case, is a **digital image**

What’s a digital image?



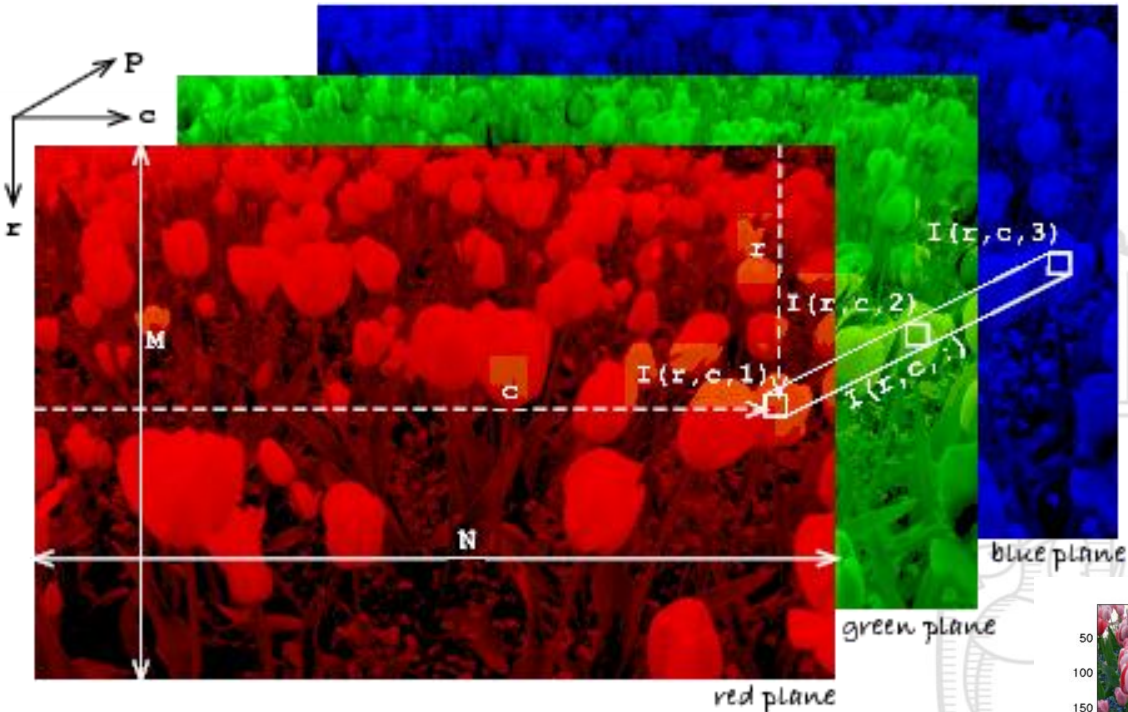
Digital images are **mosaics** made of **pixels**

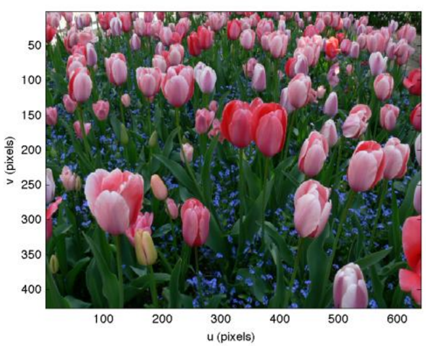
Image resolution is the number of pieces (pixel) used to build the mosaic (image)

Image depth is the number of colours (levels) of mosaic pieces

Colour images

Colour images have three channels: the most common triplet is the R-G-B



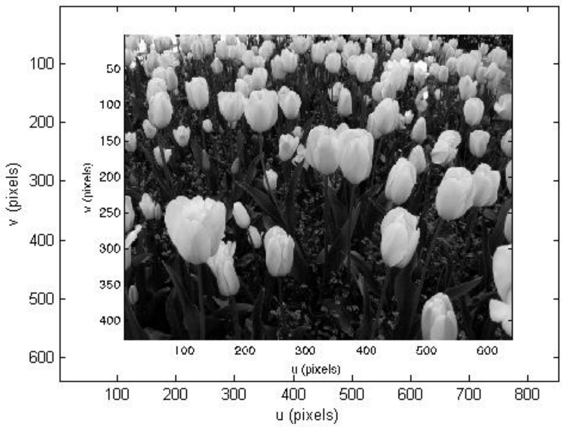


There are other very useful common space:
HSV, XYZ, CIE, YUY, …

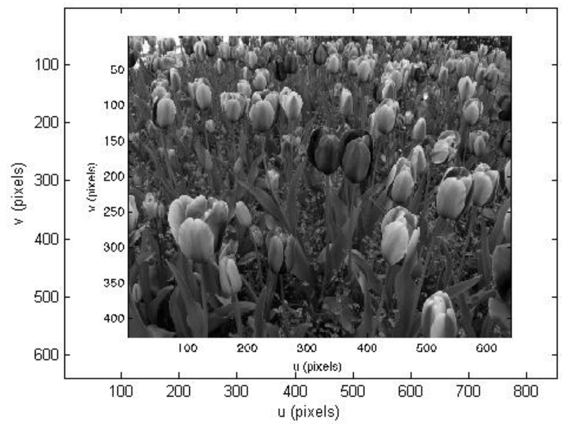
Colour images

640x854x3 uint8

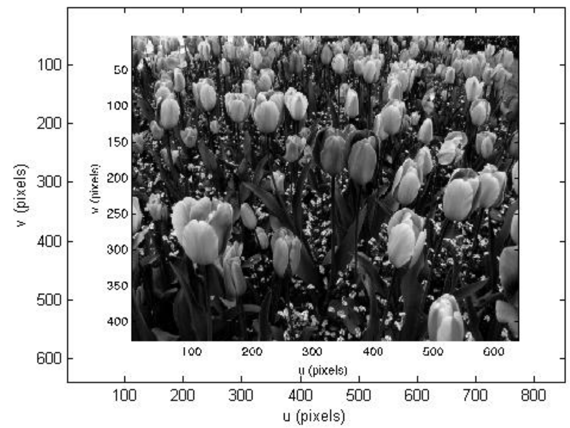
Red channel



Green channel



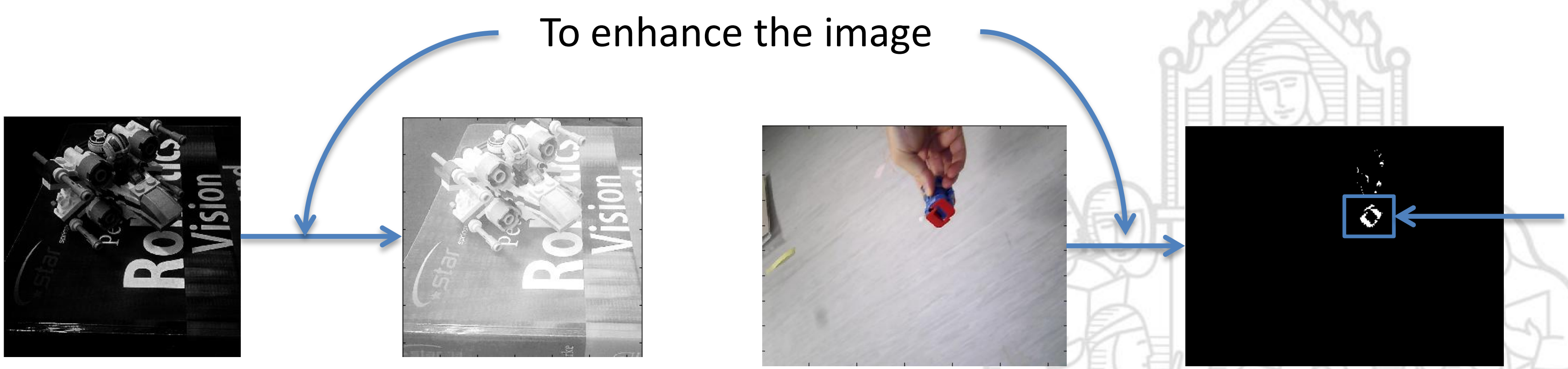
Blue channel



640x854x1 uint8

Image processing

Transform one or more input images into an output image.

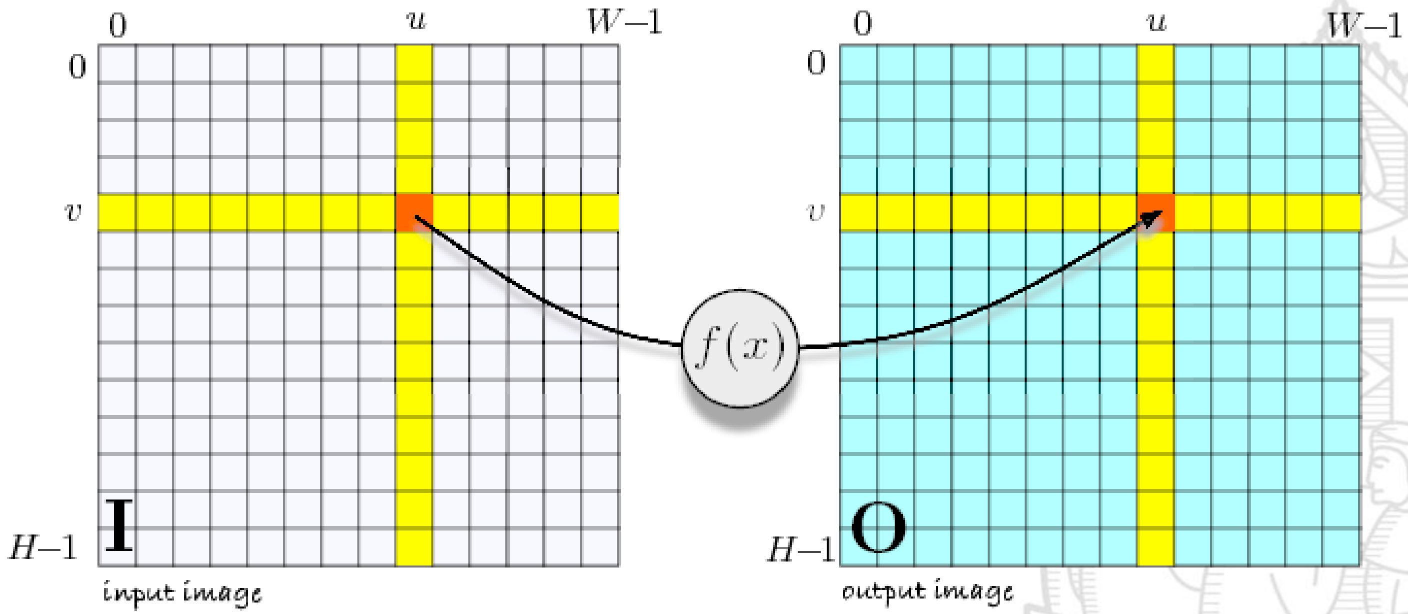


Human interpretation

Features extraction

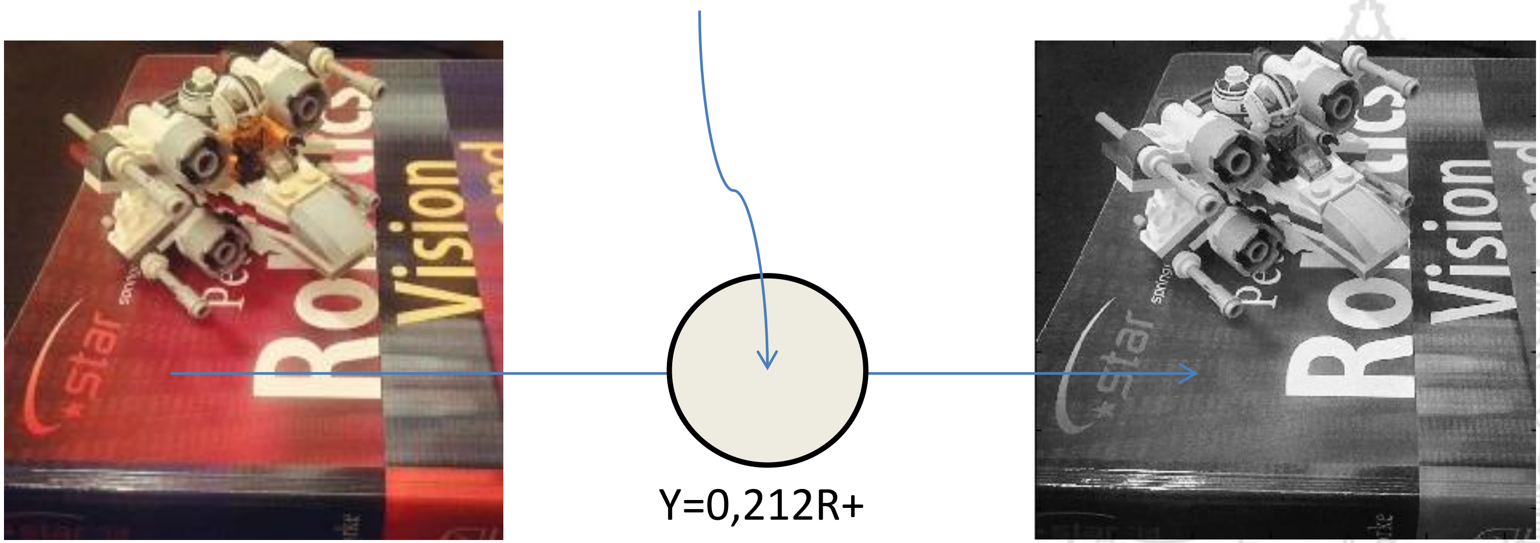
Monadic operations (pixel operators)

$$\boldsymbol{O}[u,v] = f(\boldsymbol{I}[u,v]), \qquad \forall (u,v) \in \boldsymbol{I}$$



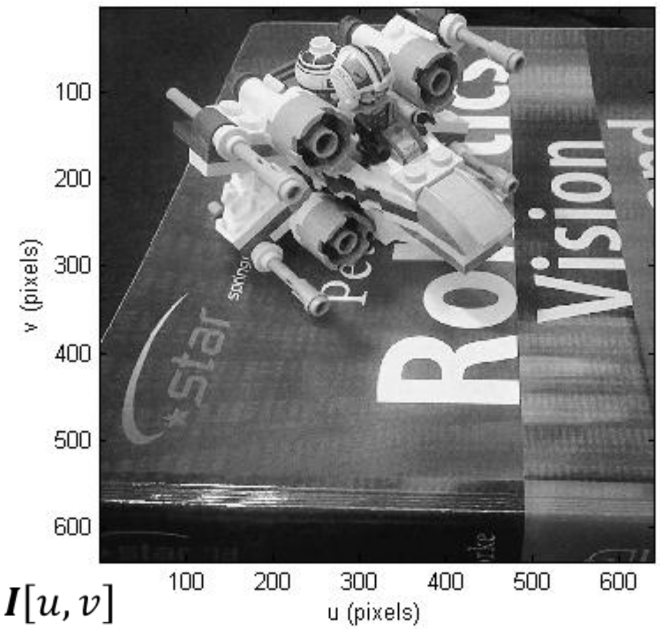
Simple monadic operation:

Gray-scale conversion with International Telecommunication Unit (ITU) recommendation 709

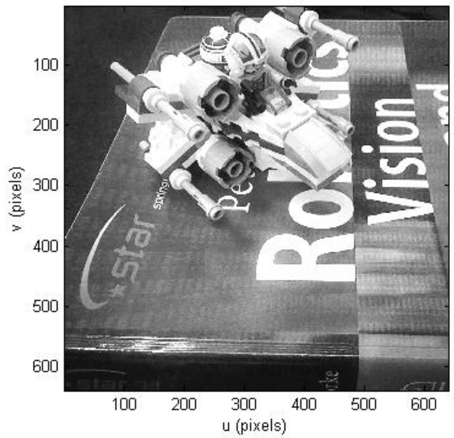


Y=0,212R+
0,7152G+0
,0722B

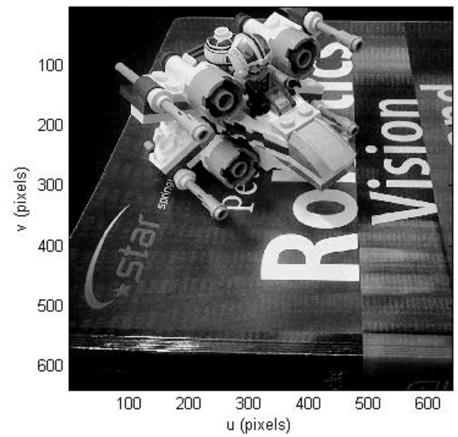
Lightening and darkening



$\boldsymbol{I}[u, v]$



$$\boldsymbol{O}[u, v] = \boldsymbol{I}[u, v] + 50$$

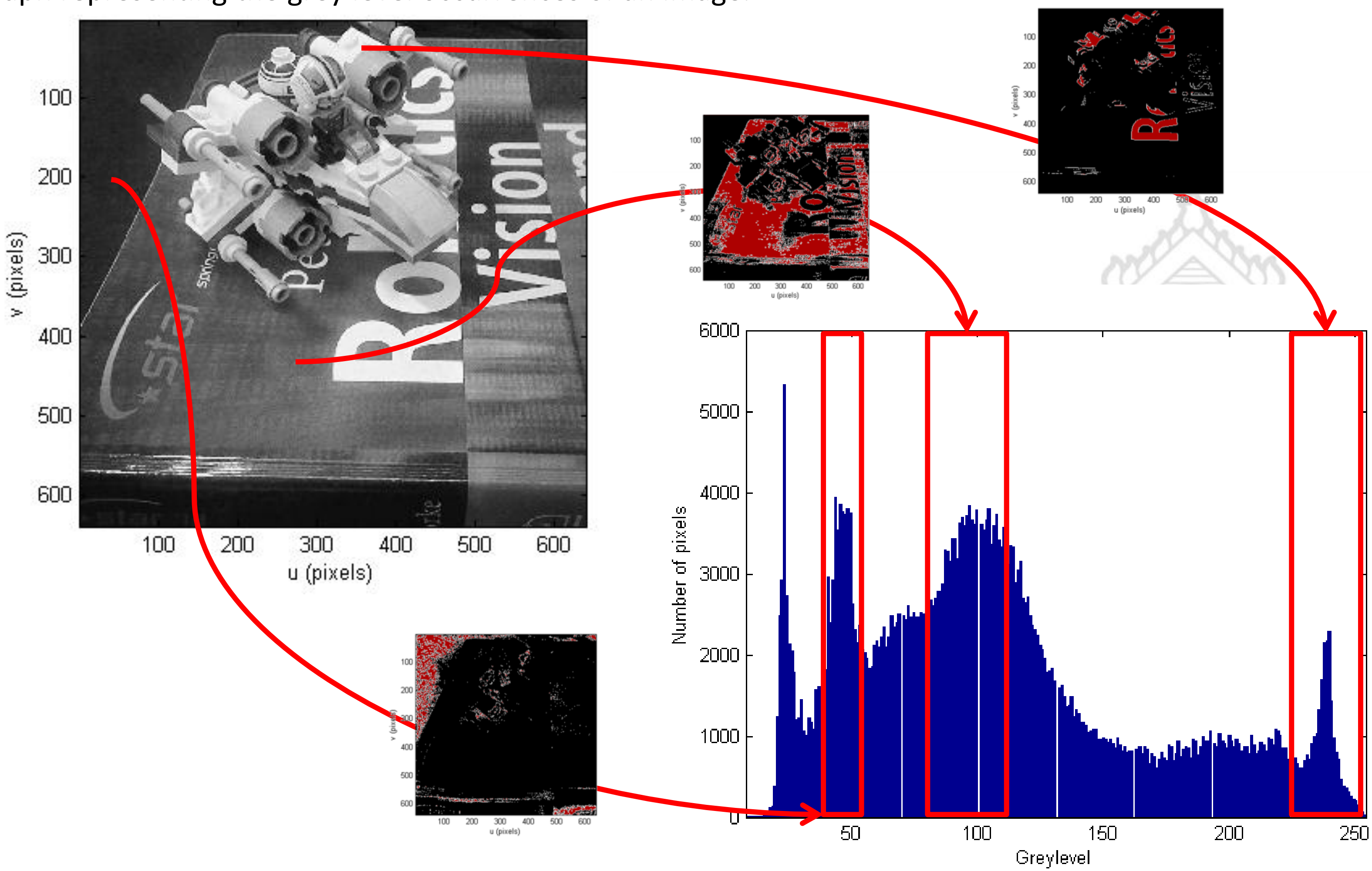


$$\boldsymbol{O}[u, v] = \boldsymbol{I}[u, v] - 50$$

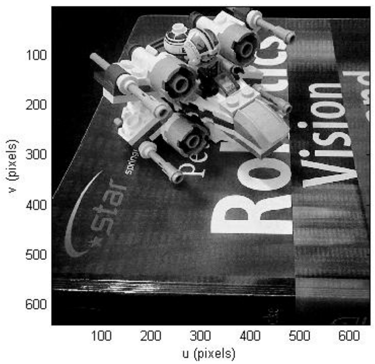
Monadic operations change the distribution of grey levels on images

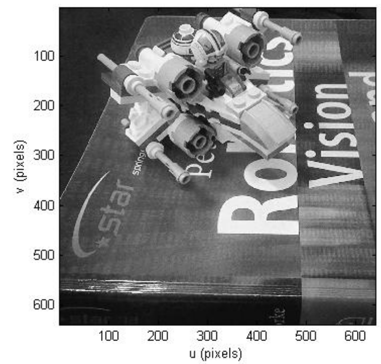
Histogram

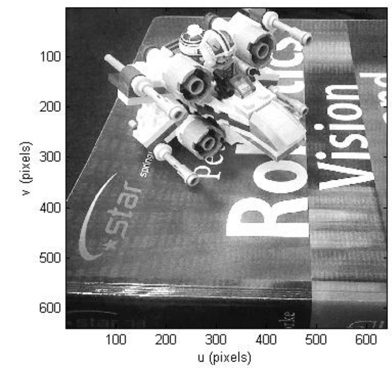
Is a graph representing the grey level occurrences of an image.



Histograms and monadic operations



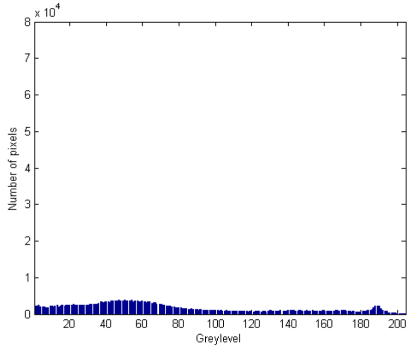


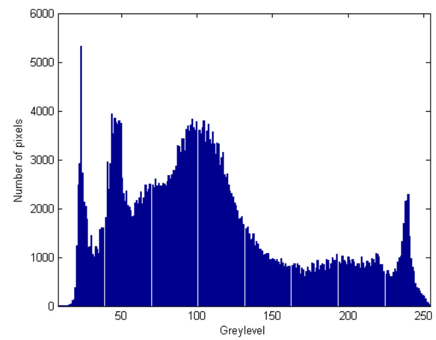


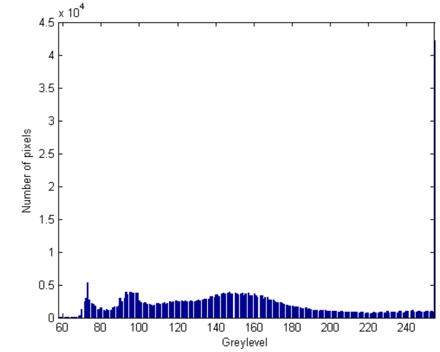
$\boldsymbol{O}[u,v] = \boldsymbol{I}[u,v]$ - 50

$\boldsymbol{I}[u,v]$

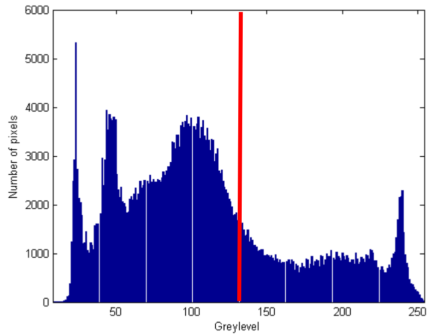
$\boldsymbol{O}[u,v] = \boldsymbol{I}[u,v] + 50$





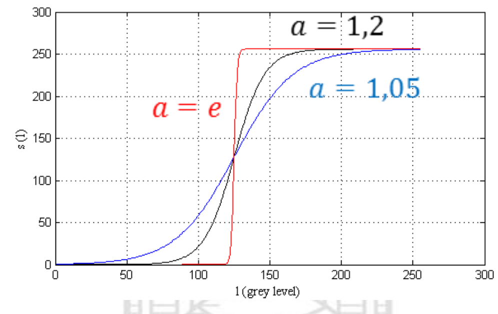


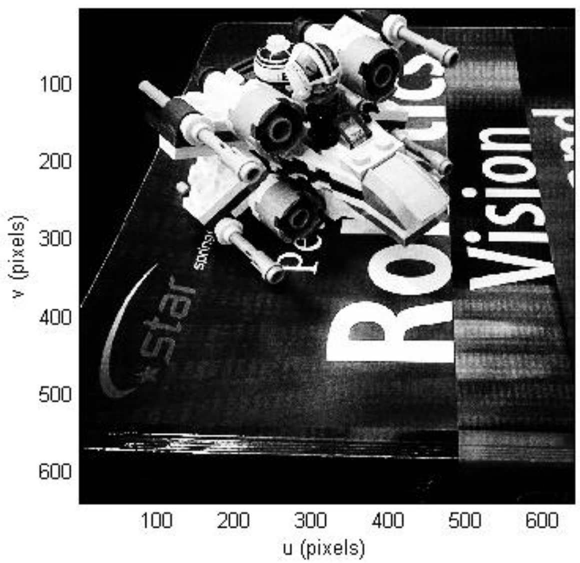
Contrast enhancement

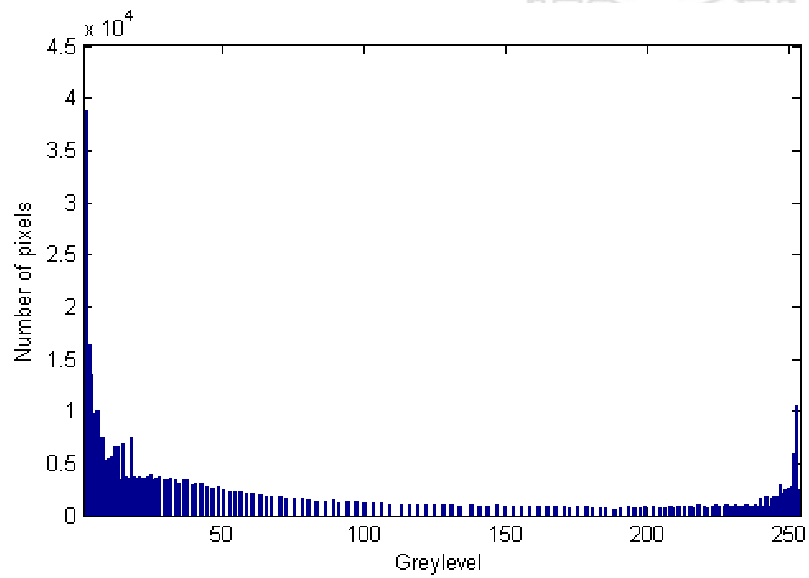


Sigmoid function

$$s(l) = \frac{256}{1 + a^{-(l-125)}}$$





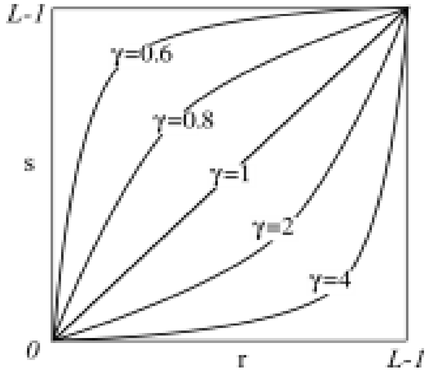


Monadic operations

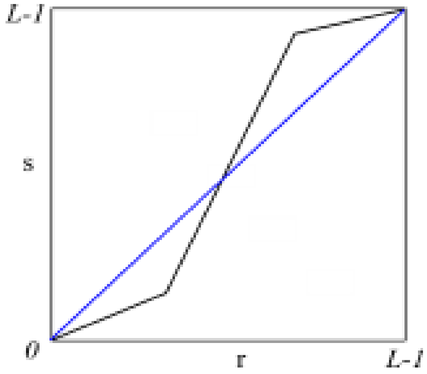
Code sample >

```
% lightening/darkening
xwing_light=xwing_grey+50;
idisp(xwing_light);
xwing_dark=xwing_grey-50;
idisp(xwing_dark);
% select areas by levels
level48 = (xwing_grey>=40) & (xwing_grey<=50) ;
idisp(level48);
level225 = (xwing_grey>=225) &
(xwing_grey<=255) ;
idisp(level225);
% contrast enanch
xwing_contrast=zeros(r,c);
    for i=1:r
       for j=1:c
           xwing_contrast(i,j)=256./(1+1.05.^-(
double(xwing_grey(i,j))-150));    % Sigmoid
       end
    end
 idisp(xwing_contrast)
```

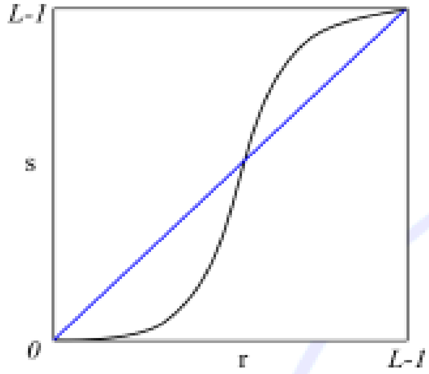
Common operation



(a) Power law



(b) Piecewise



(c) Sigmoid

Figure: Transformations

Power law	lighten or darken
Piecewise	flexible
Sigmoid	enhance the contrast

Common operation

$$s(r) = c \cdot r^{\gamma}$$

$$s(r) = \begin{cases} c_1 \cdot r & 0 \le r < r_{min} \\ c_2 \cdot r & r_{min} \le r < r_{max} \\ c_3 \cdot r & r_{max} \le r < L - 1 \end{cases}$$

$$s(r) = \frac{c_1}{1 + e^{-r}}$$

- power law
- piecewise
- sigmoid

Each function requires parameters definition

Pay attention

$$\begin{aligned} L &= 21 \\ c &= 1 \\ \gamma &= 2 \\ s(r) &= c \cdot r^{\gamma} \end{aligned}$$

$$\begin{aligned} s(1) &= 1 \\ s(2) &= 2^2 = 4 \\ s(3) &= 3^2 = 9 \\ s(4) &= 4^2 = 16 \\ s(5) &= 5^2 = 25 \\ \ldots &= \ldots \\ s(20) &= 20^2 = 400 \end{aligned}$$

???

We have only 21 levels, but:

$$s(5) = 5^2 = 25$$

Pay attention

We need to remap the output between $[0, L-1]$:

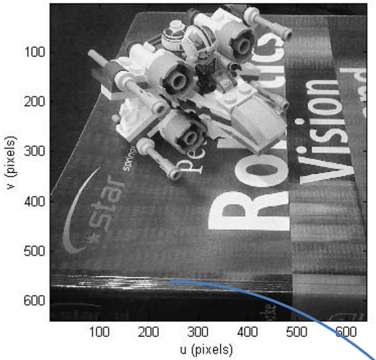
$$\frac{s'}{s} = \frac{20}{400}$$

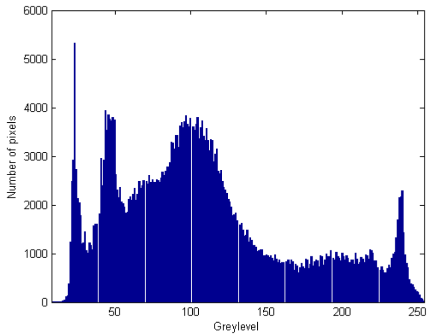
$$\frac{20}{400} = \frac{L-1}{(L-1)^{\gamma}} = (L-1)^{1-\gamma}$$

$$s' = (L-1)^{1-\gamma} s = c \cdot s = c \cdot r^{\gamma}$$

Thus c is related to L and γ.

Histogram equalization





$$c(l) = \frac{1}{N}\sum_{i=0}^{l} h(l) = c(l-1) + \frac{h(l)}{N}$$

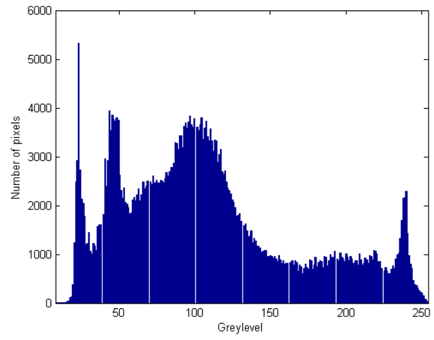
c(l)	Cumulative distribution
h(l)	histogram
l	Grey level

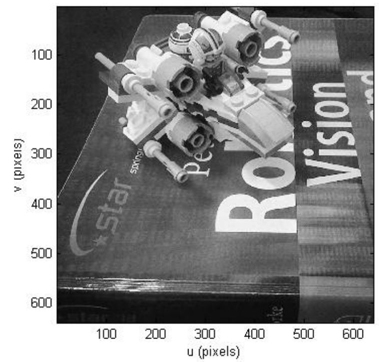
Monadic operation

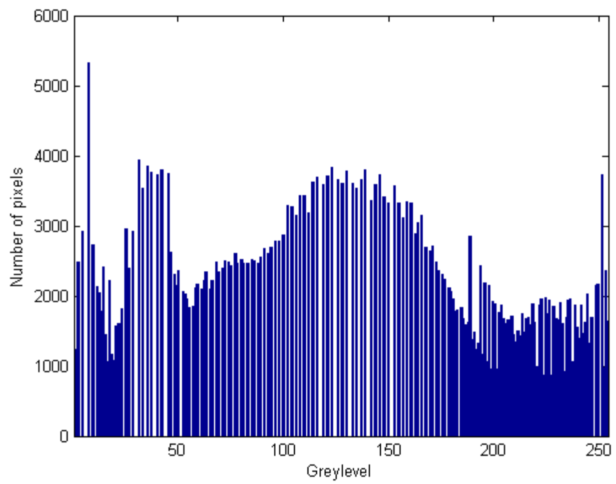
$f(l)=c(l)$

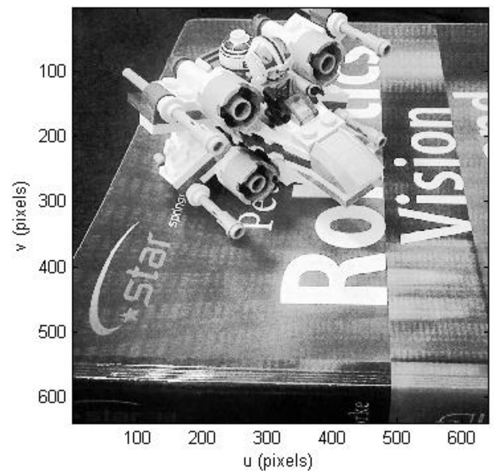
$$\boldsymbol{O}[u,v] = c(\boldsymbol{I}[u,v]), \forall (u,v) \in \boldsymbol{I}$$

Histogram equalization



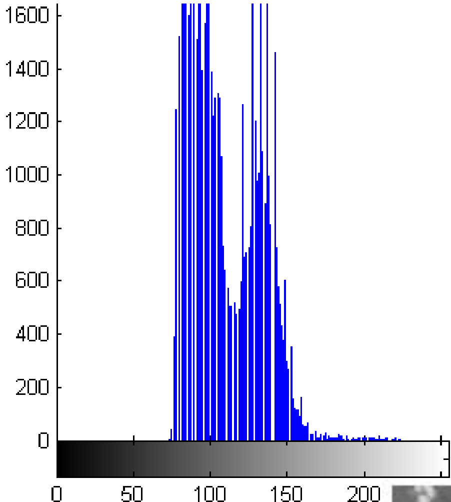






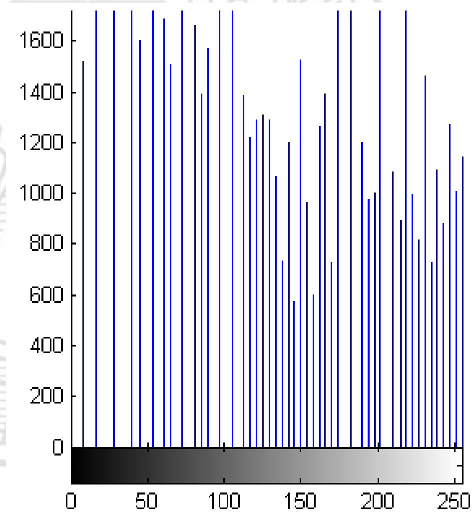
After equalization

Histogram equalization

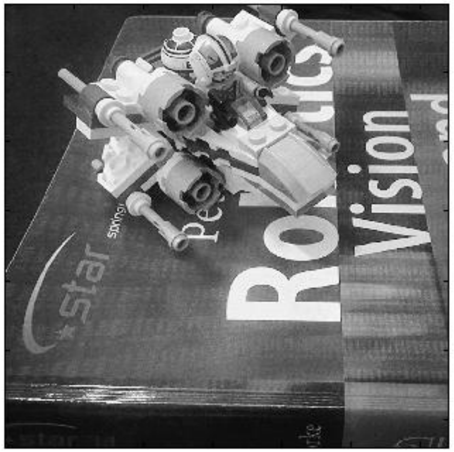


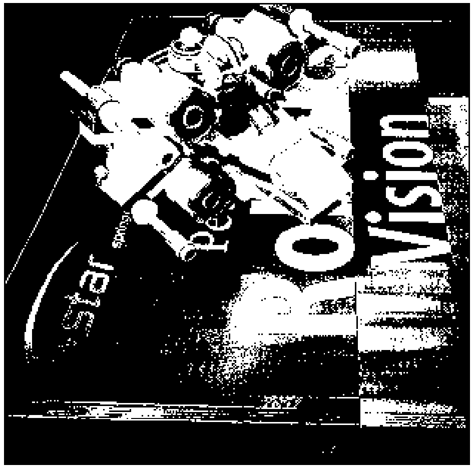
after

before



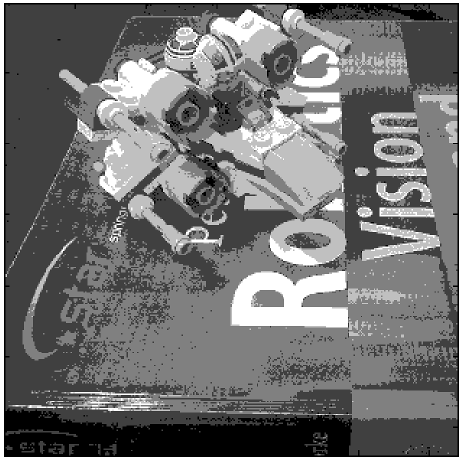
Thresholding and posterization





$$\boldsymbol{O}[u,v] = \begin{cases} 1, & if\ \boldsymbol{I}[u,v] > t \\ 0, & if\ \boldsymbol{I}[u,v] \leq t \end{cases}$$

Where t is the threshold, and 1 represents the maximum grey level value of the pixel



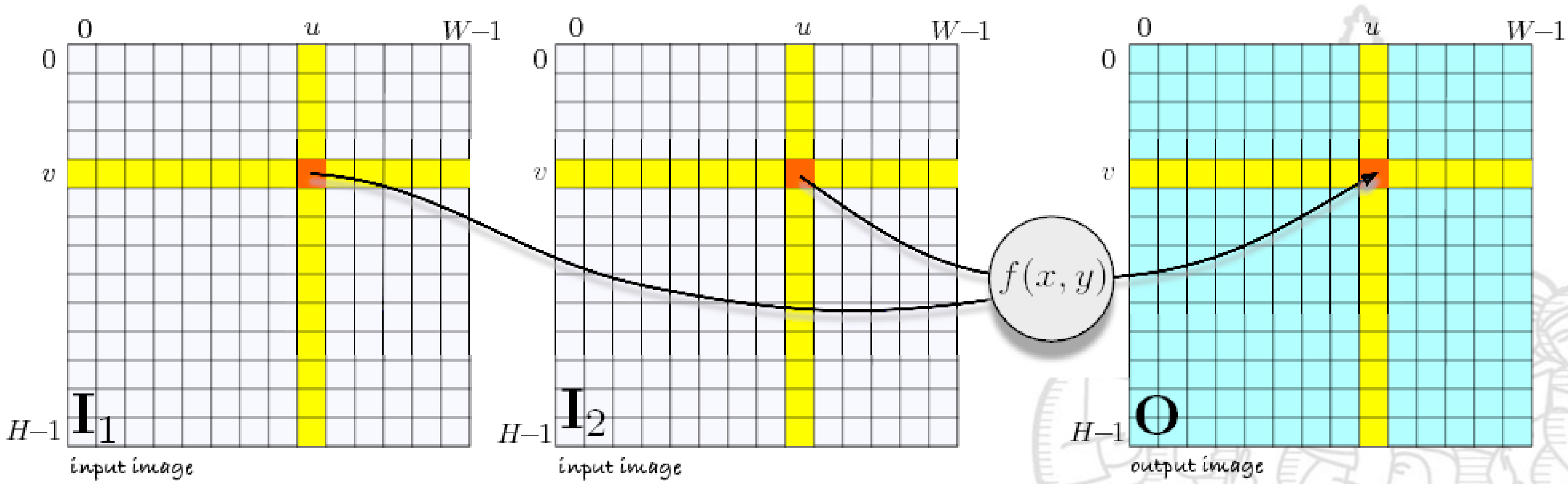
Merge several adjacent levels together

Code sample >

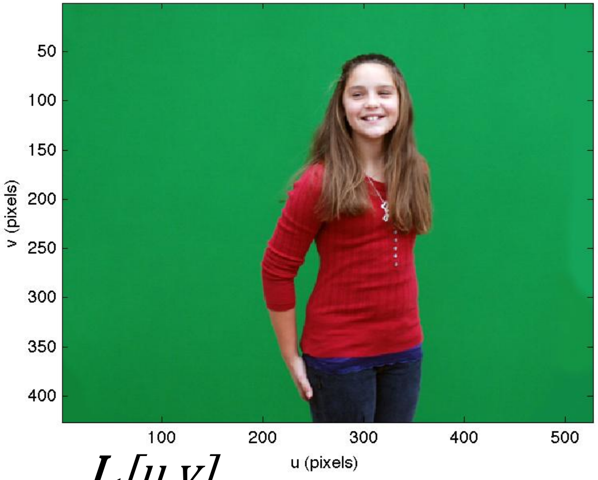
```
 %hist equalization
[n,v]=ihist(xwing_grey);
plot(v,n)
cd=zeros(length(v),1);
cd(1)=v(1)/(r*c);
    for l=2:length(v)
       cd(l)=cd(l-1)+1/(r*c)*n(l); % cumulative distribution
    end
xwing_equalized=zeros(r,c);
    for i=1:r
       for j=1:c
           xwing_equalized(i,j)=255*cd(xwing_grey(i,j)+1);    % Equalization
       end
    end
 idisp(xwing_equalized)
```

Diadic operations

$$\boldsymbol{O}[u,v] = f(\boldsymbol{I}_1[u,v], \boldsymbol{I}_2[u,v]), \qquad \forall (u,v) \in \boldsymbol{I}_1$$



Green screen



$I_1[u,v]$

$I_2[u,v]$

If $\boldsymbol{I}_1[u, v]$ isGreen

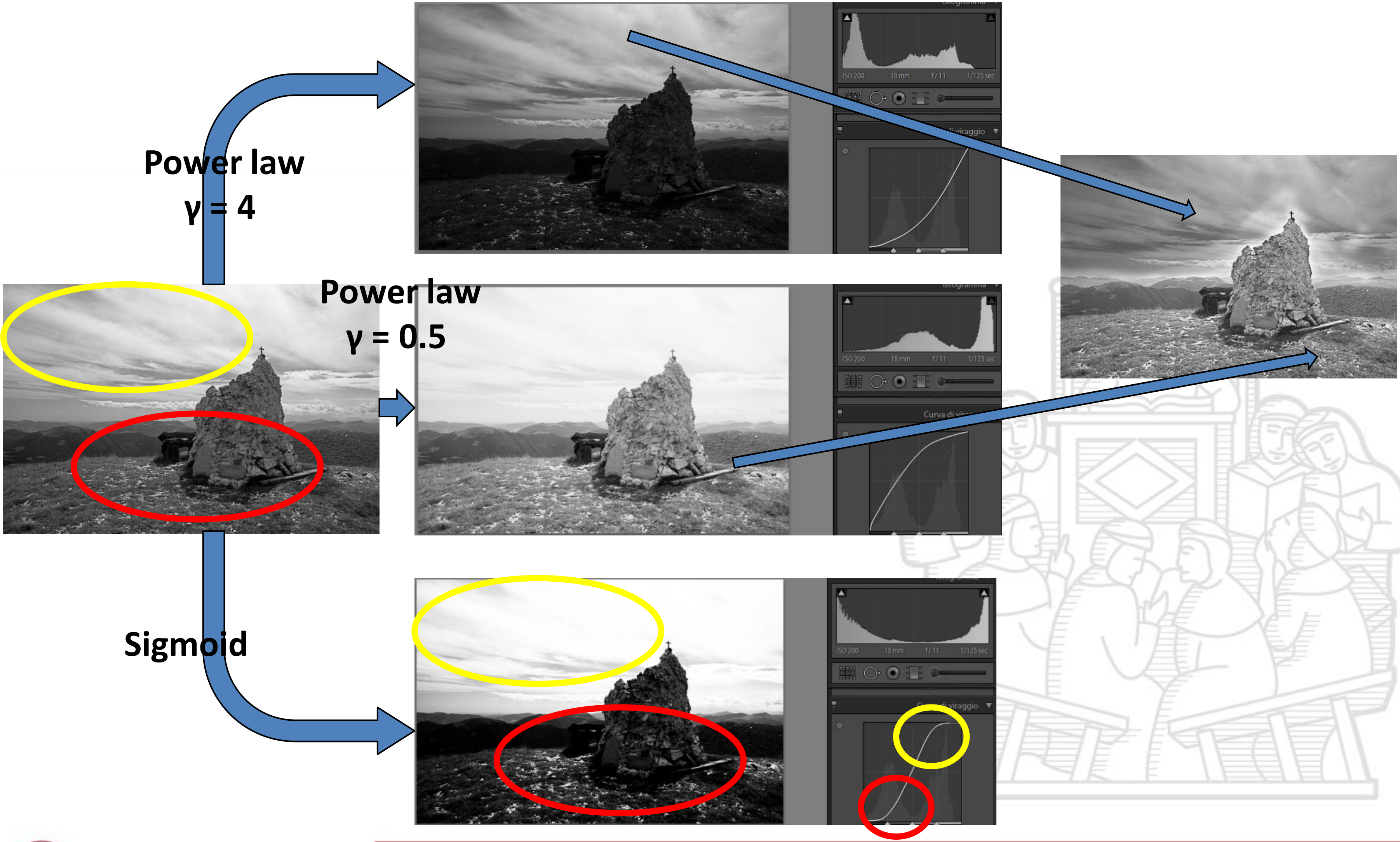
$$\boldsymbol{O}[u, v] = \boldsymbol{I}_2[u, v]$$

Else

$$\boldsymbol{O}[u, v] = \boldsymbol{I}_1[u, v]$$

$O[u,v]$

High Dynamic Range



Background subtraction

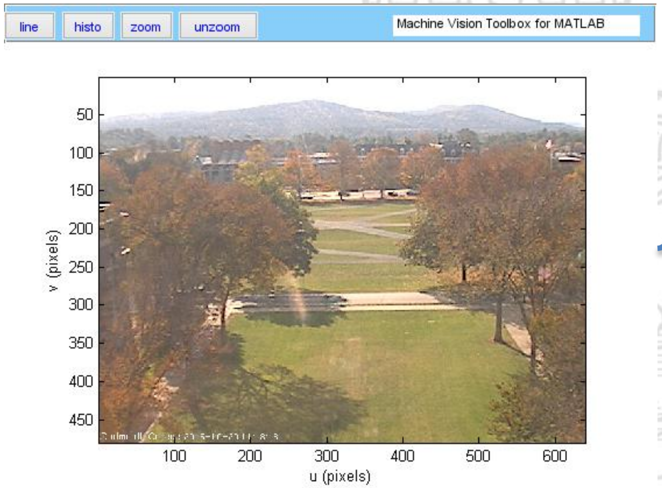
Another important diadic operation is the background subtraction to find novel elements (foreground) of a scene.

$$\boldsymbol{O}[u,v] = \boldsymbol{I}_1[u,v] - \boldsymbol{I}_2[u,v] = \boldsymbol{I}_1[u,v] - \boldsymbol{B}[u,v]$$

background

How we estimate the background $\boldsymbol{B}[u,v]$?

We can take a shoot when we know that only background is visible

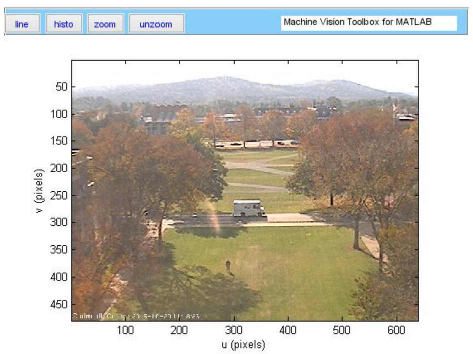


'http://wc2.dartmouth.edu', 05:19 p.m., Rome time

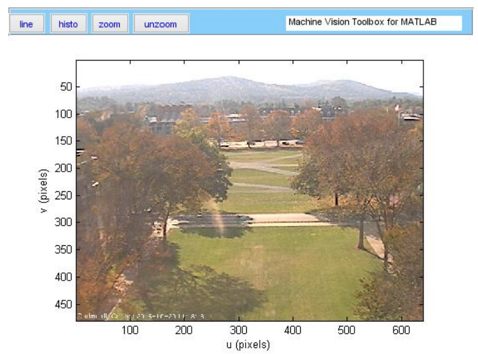
Background subtraction

'http://wc2.dartmouth.edu‘, 05:19 p.m., Rome time

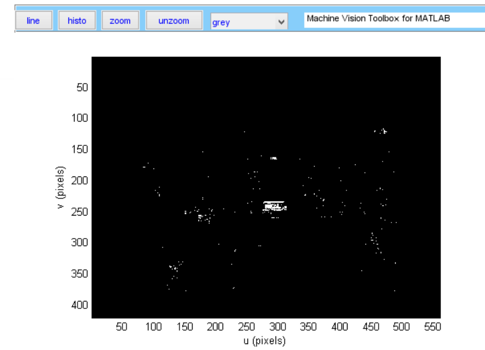
$$\boldsymbol{I}_1[u,v] - \boldsymbol{B}[u,v] = \boldsymbol{O}[u,v]$$



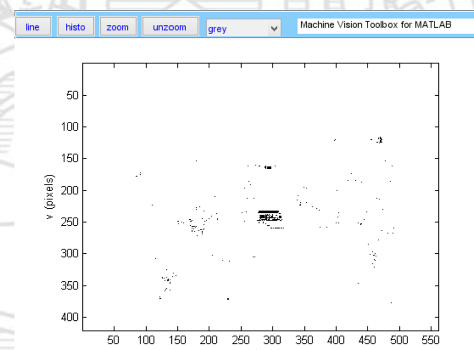
−



=



foreground

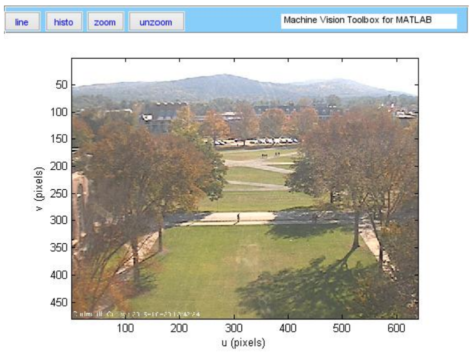


background

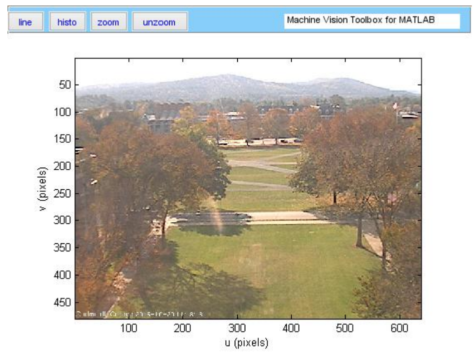
Background subtraction

'http://wc2.dartmouth.edu', 07:48 p.m., Rome time

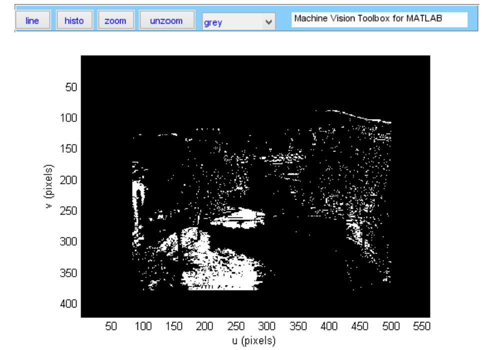
$$\boldsymbol{I}_1[u,v] - \boldsymbol{B}[u,v] = \boldsymbol{O}[u,v]$$



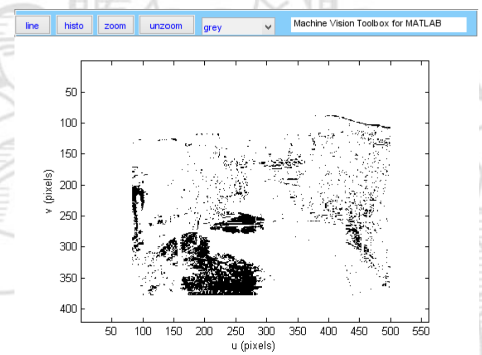
—



=



foreground



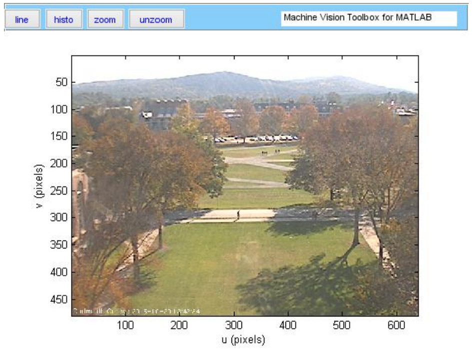
background

What went wrong?

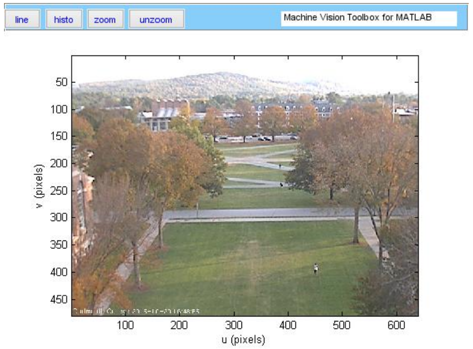
Background subtraction

'http://wc2.dartmouth.edu', 10:55 p.m., Rome time

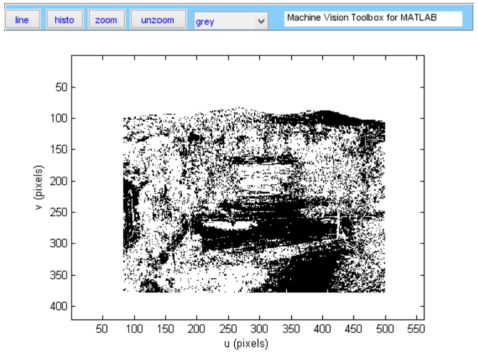
$$\boldsymbol{I}_1[u,v] - \boldsymbol{B}[u,v] = \boldsymbol{O}[u,v]$$



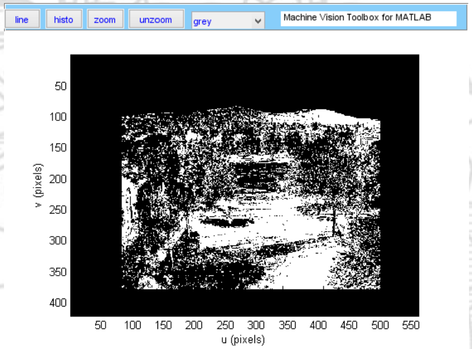
—



=



foreground



background

What went wrong?

Sant'Anna
Scuola Universitaria Superiore Pisa

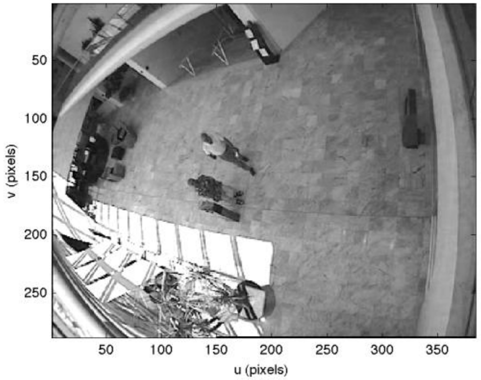
Background estimation

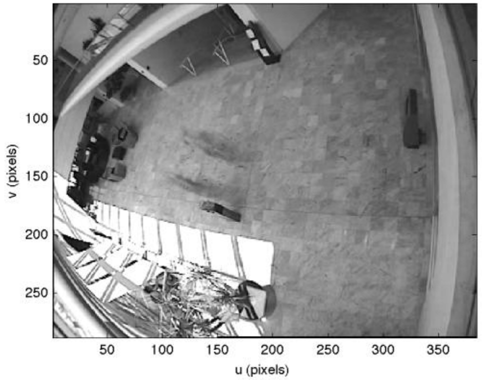
We require a progressive adaptation to small, persistent changes in the background.

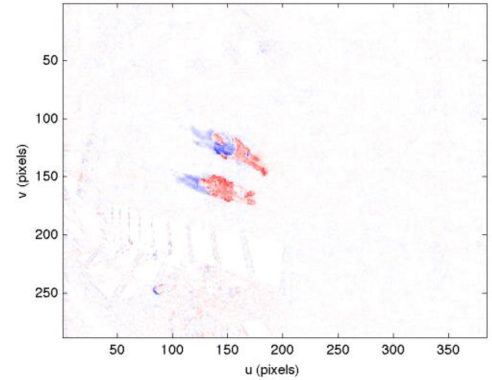
Rather than take a static image as background, we estimated it as follow:

$$\boldsymbol{B}\langle k+1\rangle = \boldsymbol{B}\langle k\rangle + c(\boldsymbol{I}\langle k\rangle - \boldsymbol{B}\langle k\rangle)$$

$$c(x) = \begin{cases} \sigma, & x > \sigma \\ x, & -\sigma \le x \le \sigma \\ -\sigma, & x < -\sigma \end{cases}$$







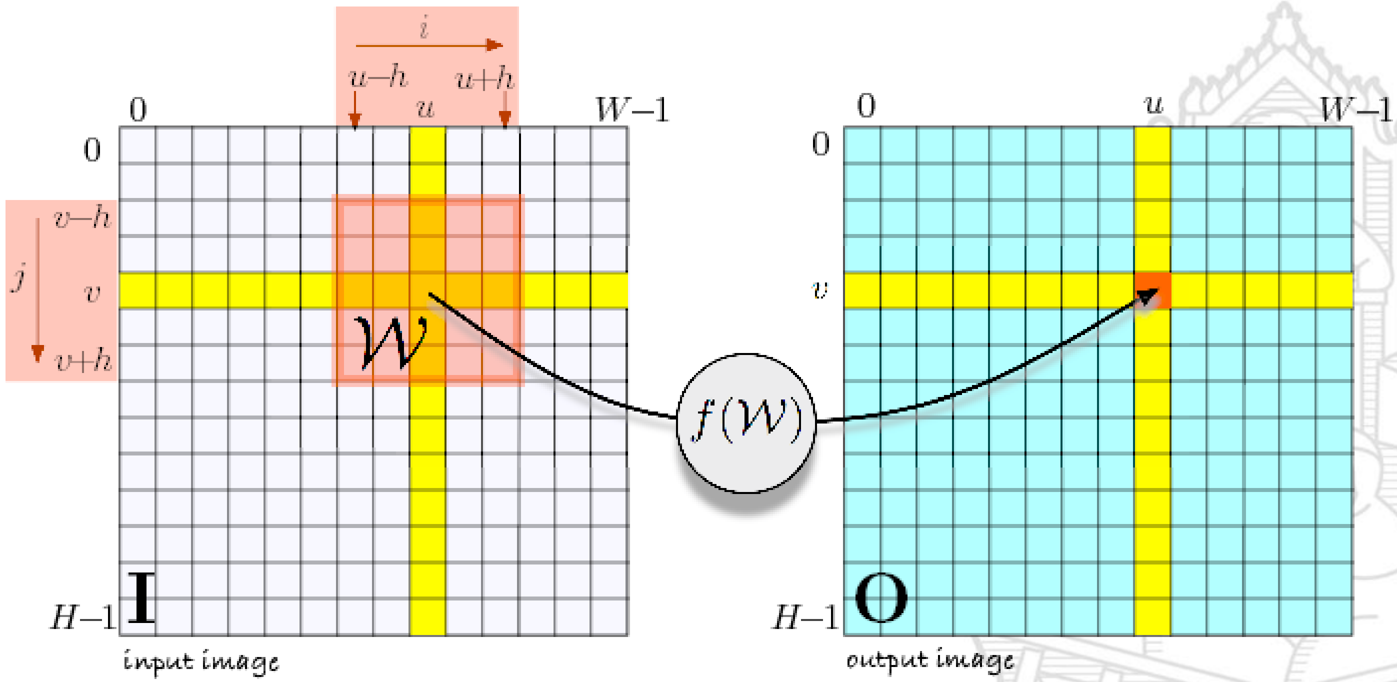
Background subtraction

Code sample >

```
% backgorund estimation
sigma=0.01;
vid = videoinput('winvideo', 1);
bg=getsnapshot(vid);
bg_small=idouble(imono(bg));
while 1
      img=getsnapshot(vid);
      img_small=idouble(imono(img));
      if isempty(img), break; end
      d=img_small-bg_small;
      d=max(min(d,sigma), -sigma);
      bg_small=bg_small+d;
      idisp(bg_small); drawnow
end
```

Spatial operation (local operators)

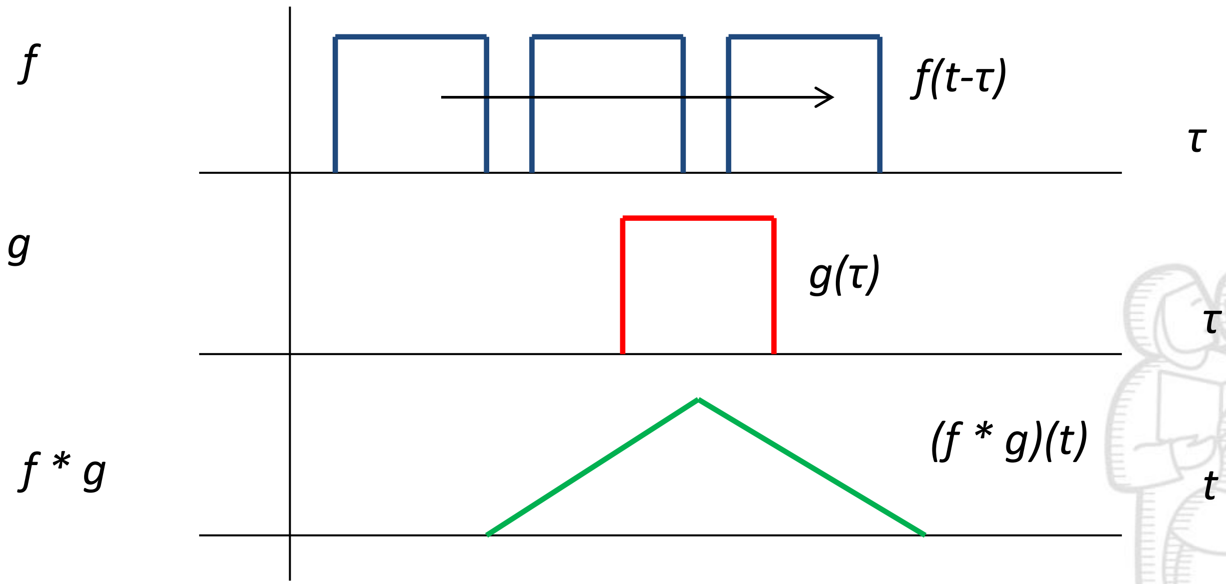
$$\boldsymbol{O}[u,v] = f(\boldsymbol{I}[u+i, v+j]), \qquad \forall (i,j) \in \boldsymbol{W}, \forall (u,v) \in \boldsymbol{I}$$

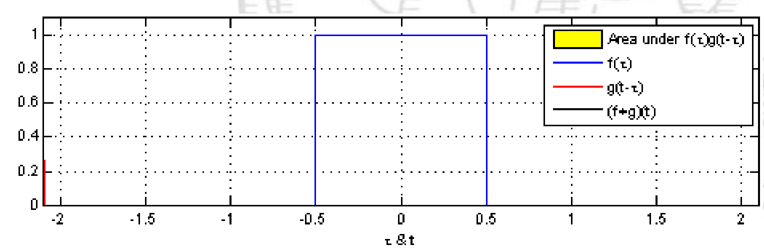


1D Convolution

One important local operator is the convolution:

$$(f * g)(t) := \int_{-\infty}^{\infty} f(t-\tau)g(\tau)d\tau$$



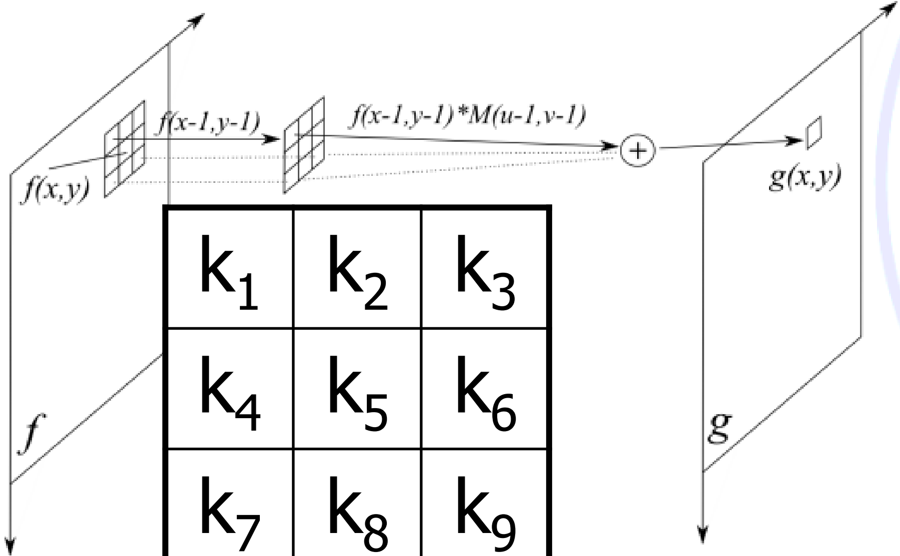


wikipedia

2D Convolution

$$\boldsymbol{O}[u,v] = \sum_{i,j \in \boldsymbol{K}} \boldsymbol{I}[u-i, v-j]\boldsymbol{K}[i,j], \qquad \forall (u,v) \in \boldsymbol{I}$$

$$\boldsymbol{O} = \boldsymbol{K} \otimes \boldsymbol{I}$$



k_1	k_2	k_3
k_4	k_5	k_6
k_7	k_8	k_9

Convolution mask

2D Convolution

kernel

·1+	·1+	·1+
·1+	·1+	·1+
·1+	·1+	·1

Input image

0	1	2	0	12	5	0	1
5	2	6	0	0	1	1	1
5	0	0	4	5	6	1	0
12	25	0	24	56	8	2	3
1	2	6	0	0	1	5	2
1	2	0	2	1	2	1	0
12	0	12	25	3	5	0	1
1	1	1	35	57	5	3	1

Output image

	21						

Convolution

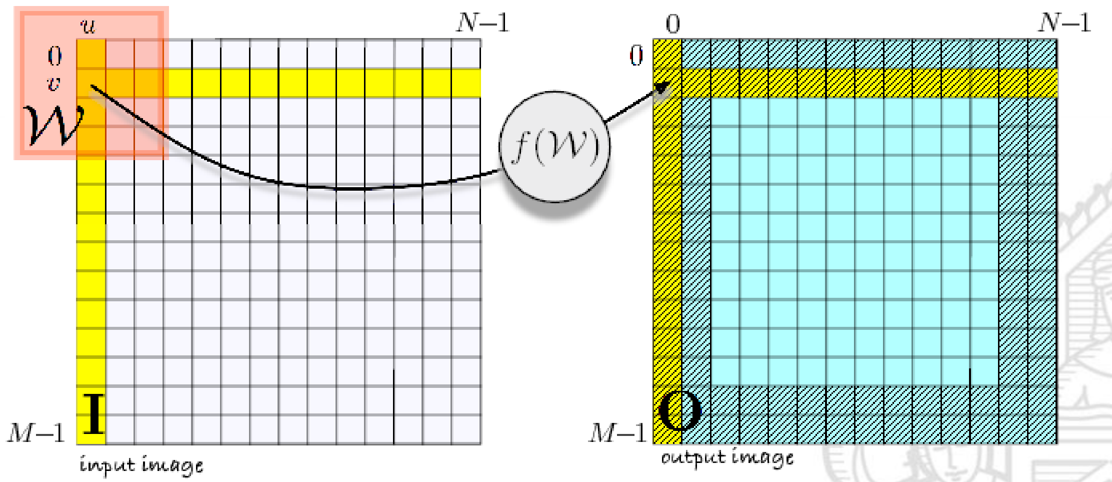
Input image

0·1+	1·1+	2·1+	0	12	5	0	1
5·1+	2·1+	6·1+	0	0	1	1	1
5·1+	0·1+	0·1	4	5	6	1	0
12	25	0	24	56	8	2	3
1	2	6	0	0	1	5	2
1	2	0	2	1	2	1	0
12	0	12	25	3	5	0	1
1	1	1	35	57	5	3	1

Output image

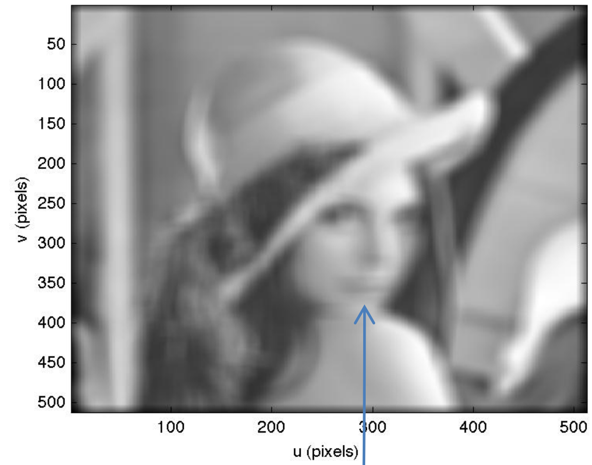
	21	15					

Boundary effect



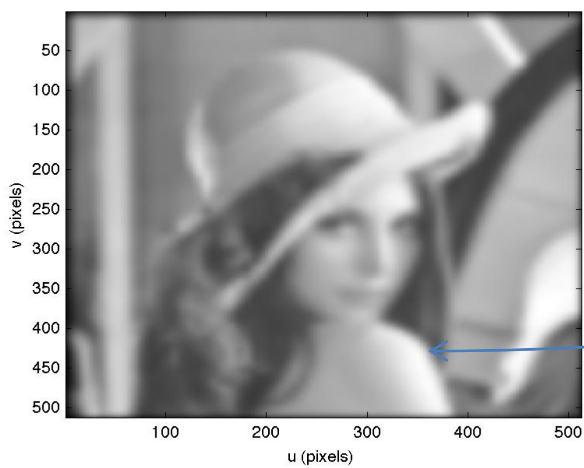
- Duplicate
- All black
- Reduce size
- …

Smoothing



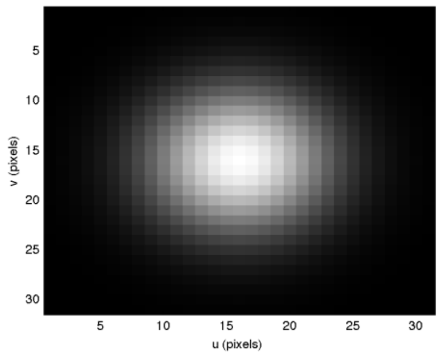
$$K = ones(21,21)/21^2$$

$$O = K \otimes I$$

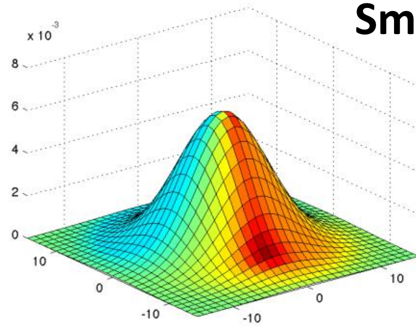


$$G[u,v] = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$

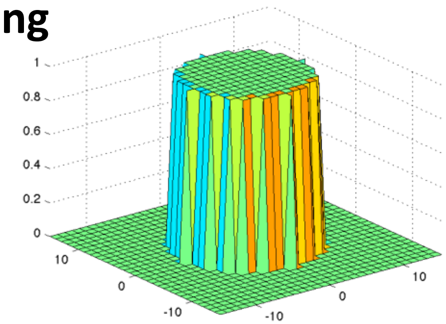
Kernel examples



Smoothing

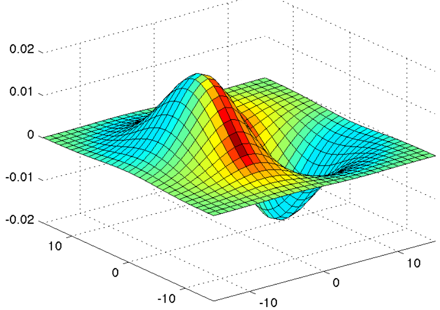


Gaussian



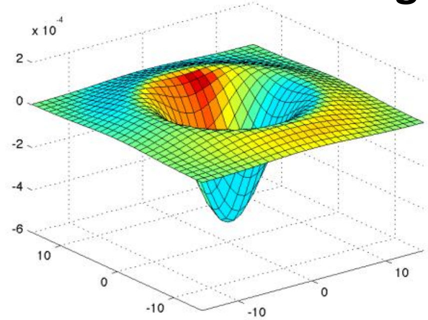
Top hat

Gradient

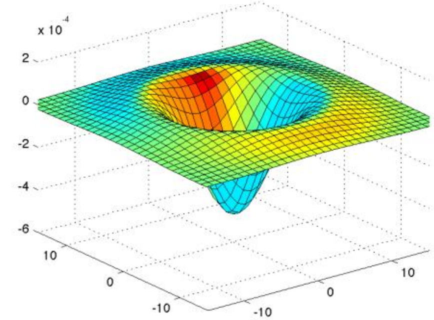


Derivative of Gaussian
(DoG)

Edge detection

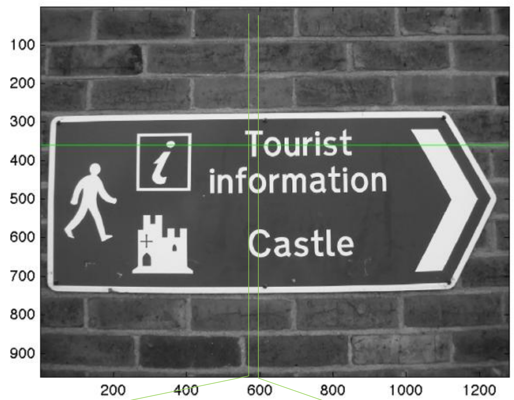


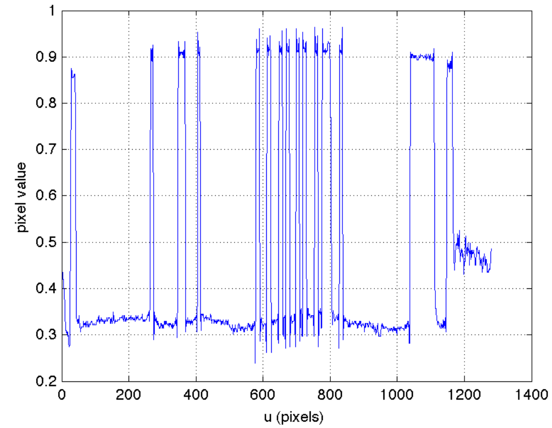
Laplacian of Gaussian
(LoG)



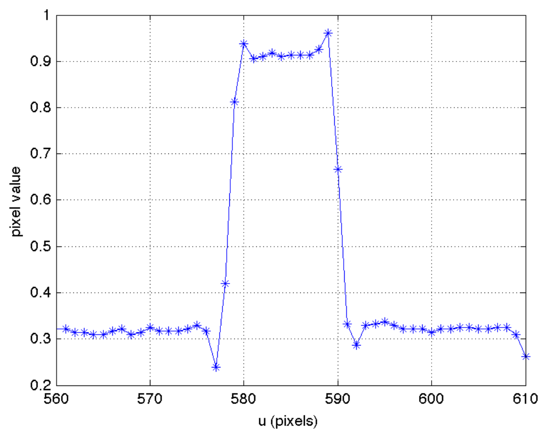
Difference of Gaussian
(DiffG)

Edge detection





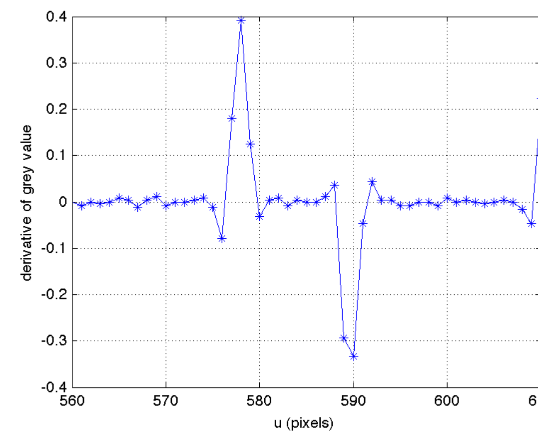
Horizontal profile of the image at v=360



$$p'[u] = p[u] - p[u-1]$$

$$p'[u] = \frac{1}{2}(p[u+1] - p[u-1])$$

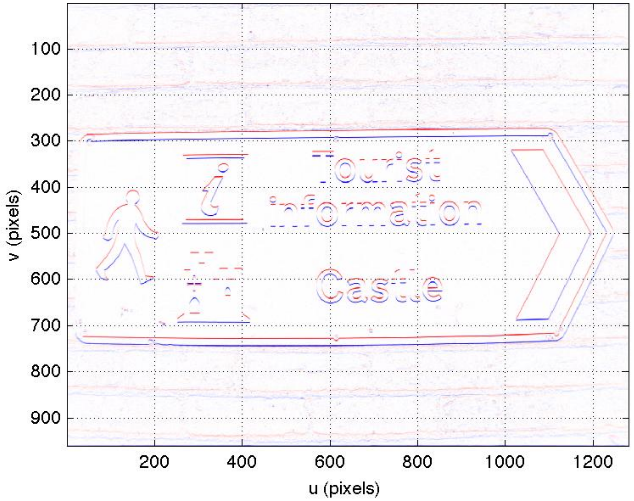
$$\boldsymbol{K} = \left[-\frac{1}{2} \;\; 0 \;\; \frac{1}{2}\right]$$



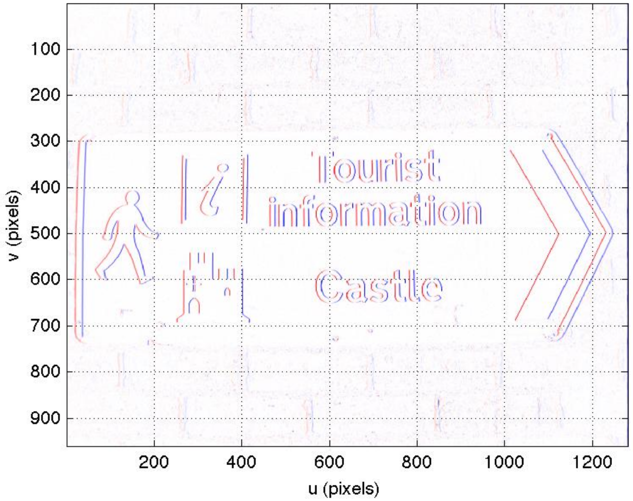
Gradient computation

Common convolution kernel: Sobel, Prewitt, Roberts, …

Sobel $\boldsymbol{D}_v = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$ $\boldsymbol{D}_u = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$



$\boldsymbol{I}_v = \boldsymbol{D}_v \otimes \boldsymbol{I}$

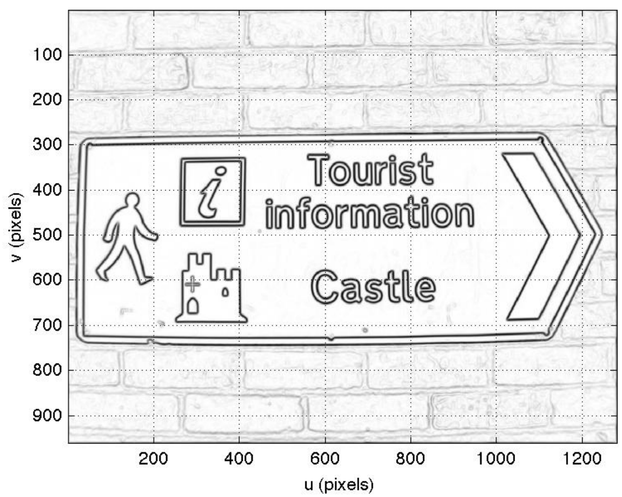


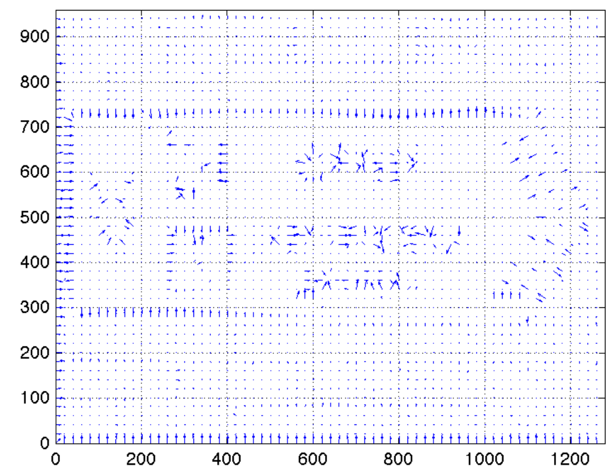
$\boldsymbol{I}_u = \boldsymbol{D}_u \otimes \boldsymbol{I}$

Direction and magnitude

$$m = \sqrt{\boldsymbol{I}_v^2 + \boldsymbol{I}_u^2}$$

$$\theta = \operatorname{atan}(\boldsymbol{I}_v, \boldsymbol{I}_u)$$





Noise amplification

Derivative amplifies high-frequency noise. So, firstly we can smooth the image, after that we can take the derivative:

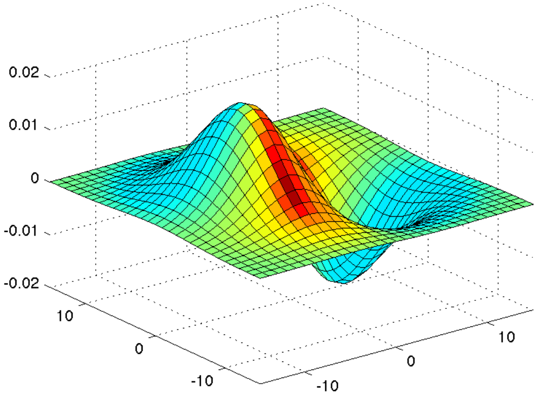
$$\boldsymbol{I}_u = \boldsymbol{D}_u \otimes (\boldsymbol{G} \otimes \boldsymbol{I})$$

Associative property:

$$\boldsymbol{I}_u = \underbrace{(\boldsymbol{D}_u \otimes \boldsymbol{G})} \otimes \boldsymbol{I}$$

Derivative of Gaussian (DoG)

$$\boldsymbol{G}_u = -\frac{u}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$



Derivative of Gaussian (DoG)

<<DoG acts as a bandpass filter!>>

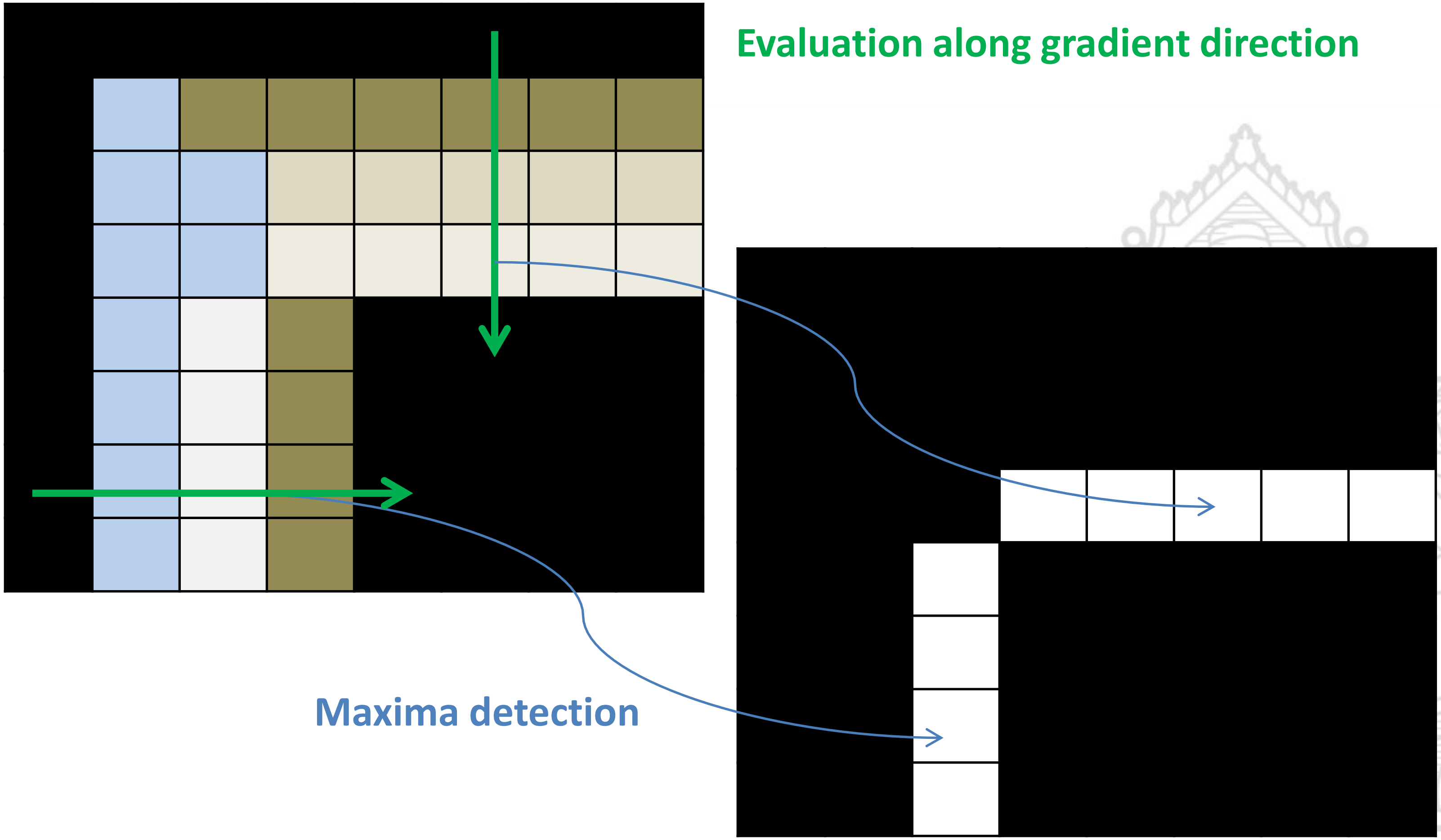
Canny edge detection

The algorithm is based on a few steps:

1. Gaussian filtering
2. Gradient intensity and direction
3. non-maxima suppression (edge thinning)
4. hysteresis threshold

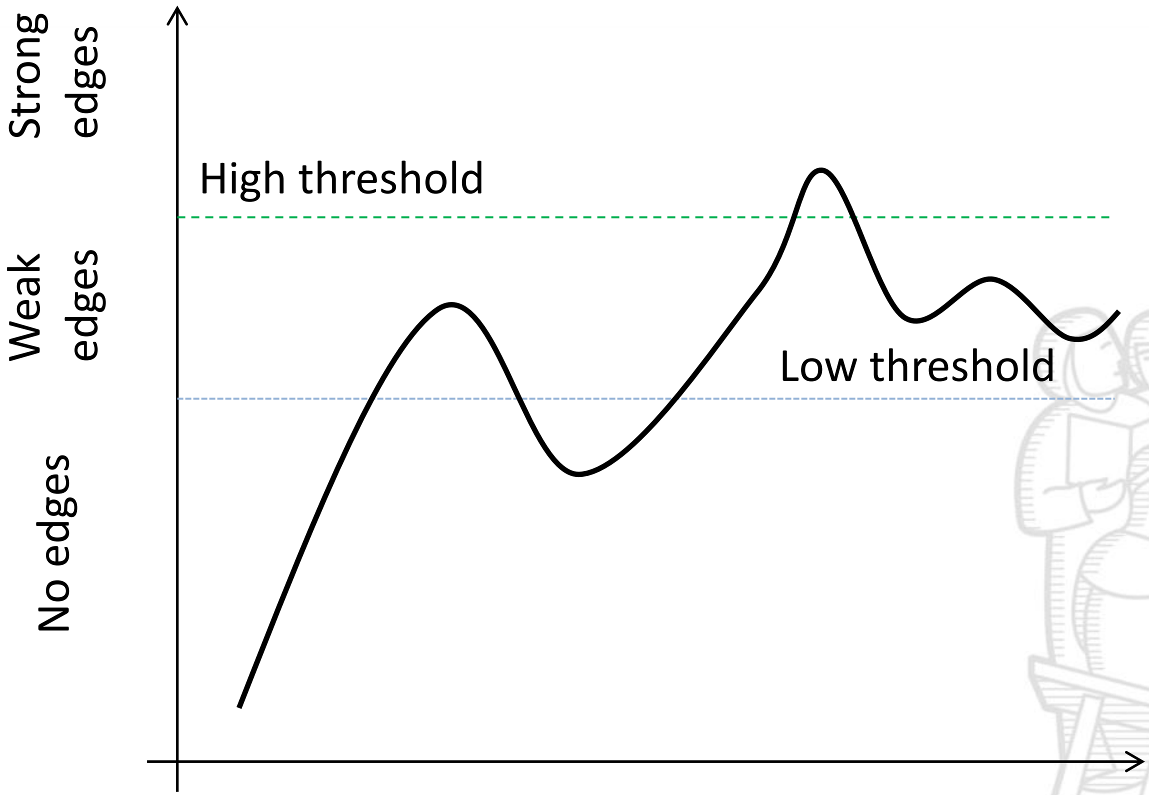
Canny edge detection

3. Non local maxima suppression

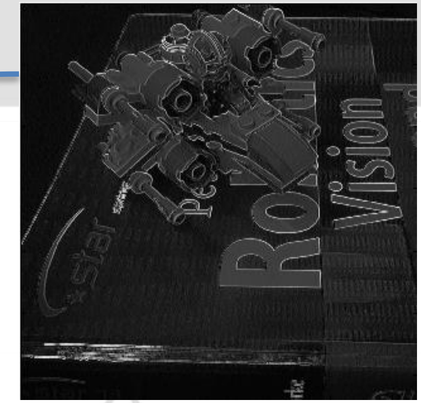


Canny edge detection

4. hysteresis threshold

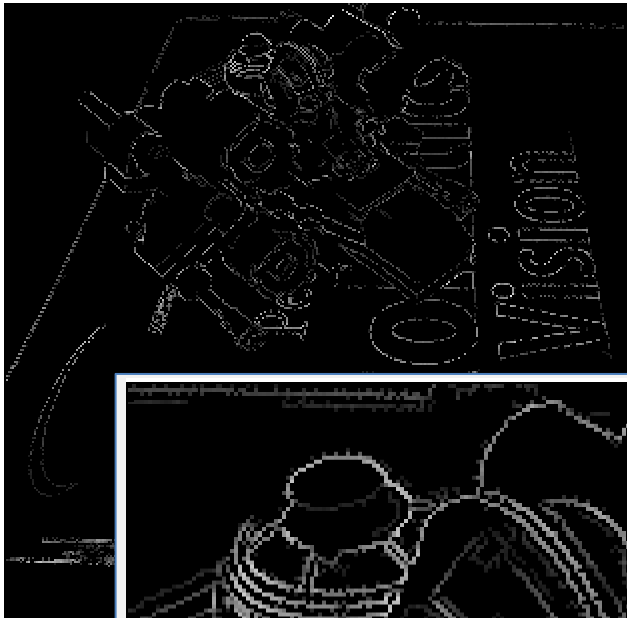


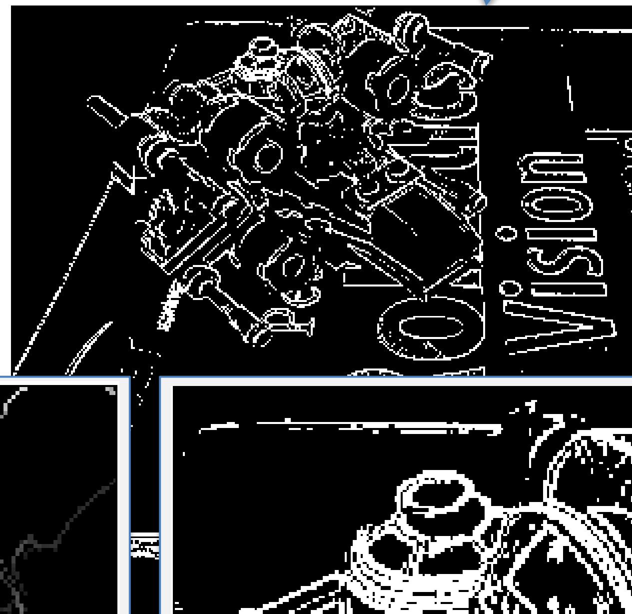
Magnitude of the gradient

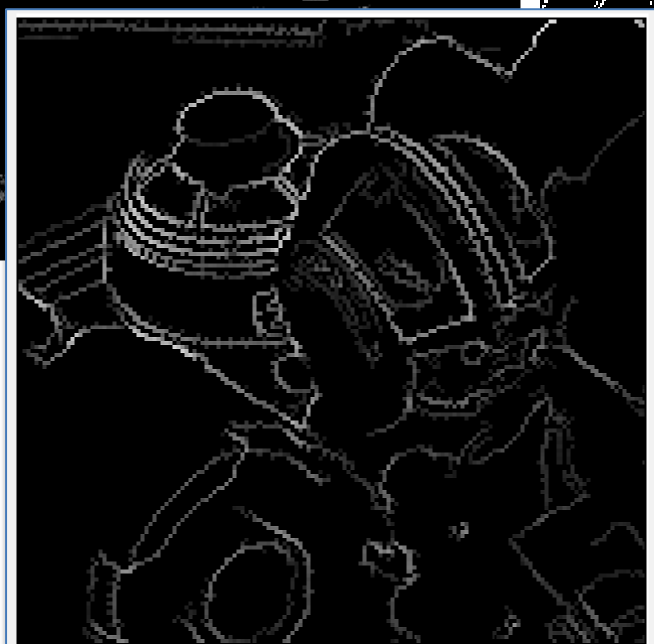


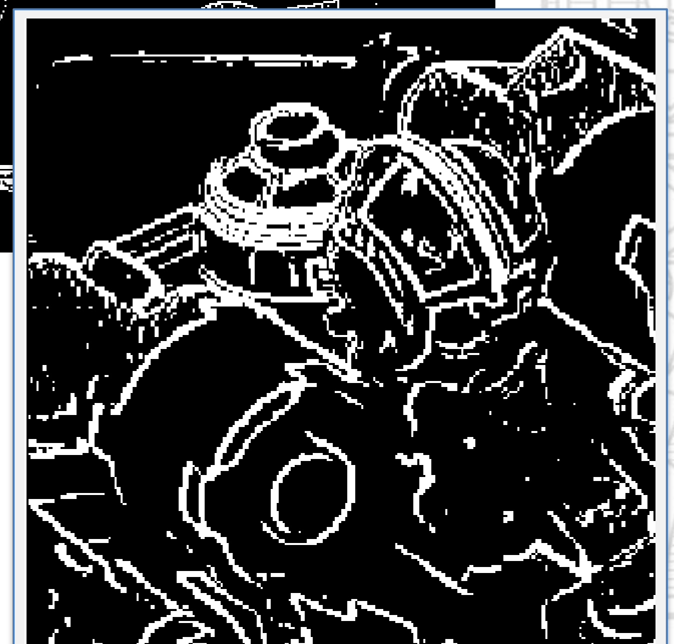
Thresholding

canny









Edge detection

High gradient → Local maxima

Alternative approach is to use second derivative and to find where there is a zero

Laplacian operator

$$\nabla \boldsymbol{I}^2 = \frac{\partial^2 \boldsymbol{I}}{\partial u^2} + \frac{\partial^2 \boldsymbol{I}}{\partial v^2} = \boldsymbol{I}_{uu} + \boldsymbol{I}_{vv} = \boldsymbol{L} \otimes \boldsymbol{I}$$

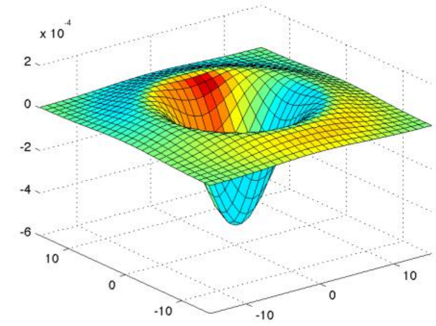
$$\boldsymbol{L} = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

Noise sensitivity

Again, derivative amplifies high-frequency noise. So firstly we can smooth the image, after that we take the derivative:

$$\boldsymbol{L} \otimes (\boldsymbol{G} \otimes \boldsymbol{I}) = \underbrace{(\boldsymbol{L} \otimes \boldsymbol{G})}_{\text{Laplacian of Gaussian (LoG)}} \otimes \boldsymbol{I}$$

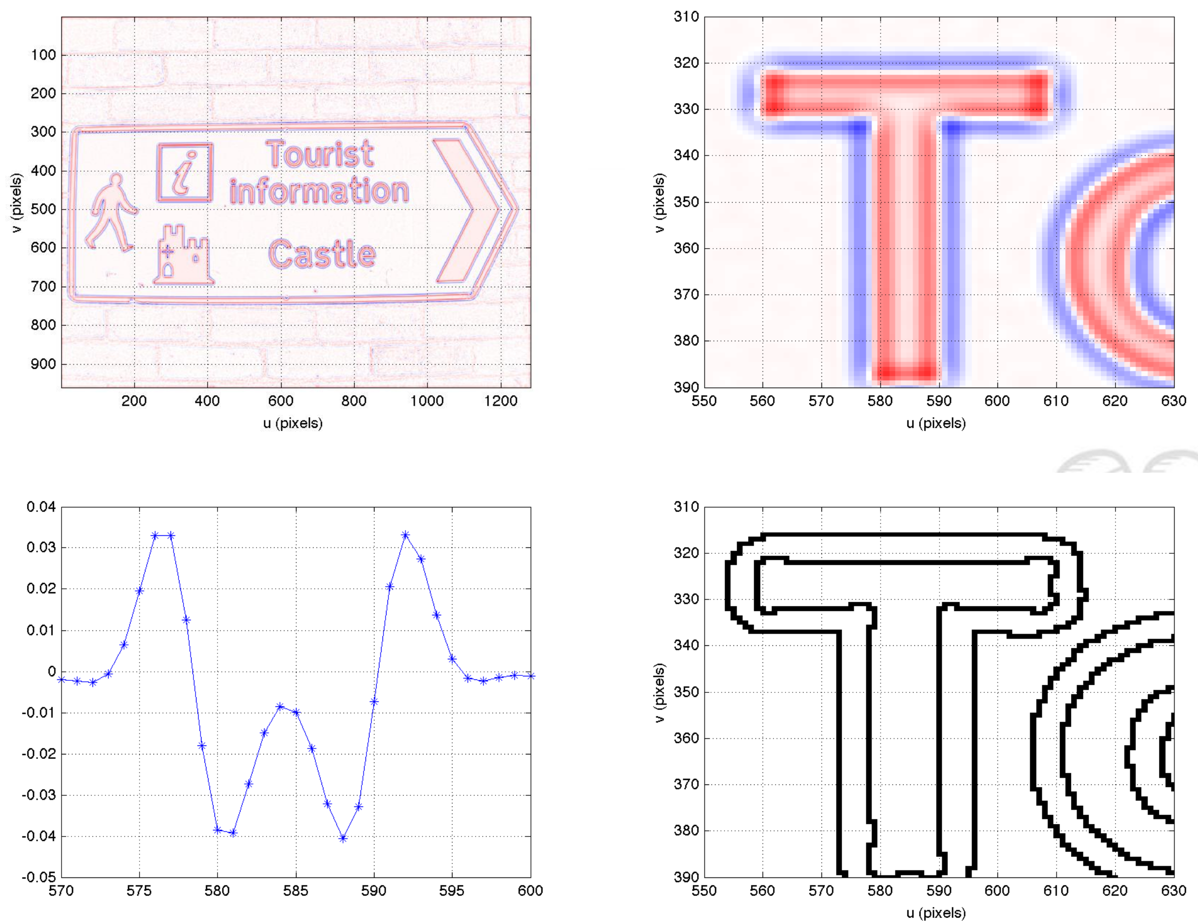
$$\boldsymbol{LoG}(u, v) = \frac{1}{\pi\sigma^4}\left(\frac{u^2 + v^2}{2\sigma^2} - 1\right) e^{-\frac{u^2+v^2}{2\sigma^2}}$$



Laplacian of Gaussian
(LoG)

Marr-Hildreth operator or the Mexican hat kernel

Edge detection



Gradient and Laplacian

Example

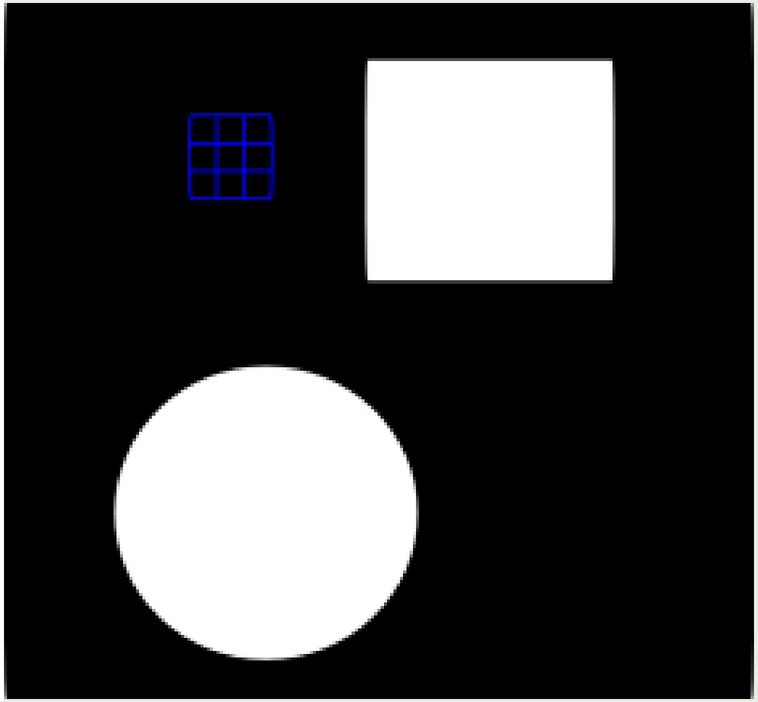


Image window:

$$f(x,y) = \begin{array}{|c|c|c|} \hline 0 & 0 & 0 \\ \hline 0 & 0 & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array}$$

$$\nabla^2 = \begin{array}{|c|c|c|} \hline 0 & -1 & 0 \\ \hline -1 & 4 & -1 \\ \hline 0 & -1 & 0 \\ \hline \end{array}$$

$$G_x = \begin{array}{|c|c|c|} \hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline \end{array}$$

Products are:

$$\nabla^2 \otimes f(x,y) = 0$$

$$G_x \otimes f(x,y) = 0$$

Gradient and Laplacian

Example

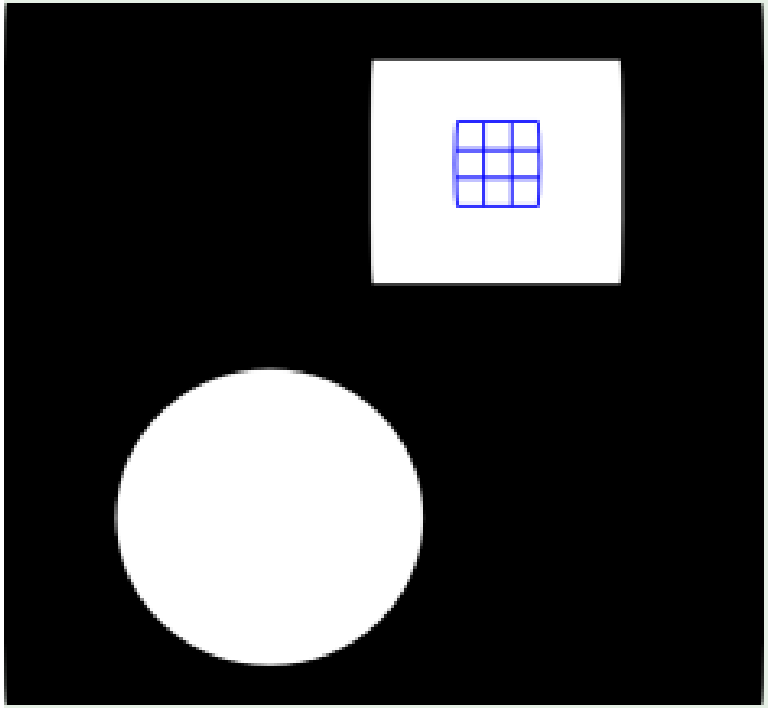


Image window:

$$f(x,y) = \begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$

$$\nabla^2 = \begin{array}{|c|c|c|}\hline 0 & -1 & 0 \\ \hline -1 & 4 & -1 \\ \hline 0 & -1 & 0 \\ \hline \end{array}$$

$$G_x = \begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline \end{array}$$

Products are:

$$\nabla^2 \otimes f(x,y) = 0$$

$$G_x \otimes f(x,y) = 0$$

Gradient and Laplacian

Example

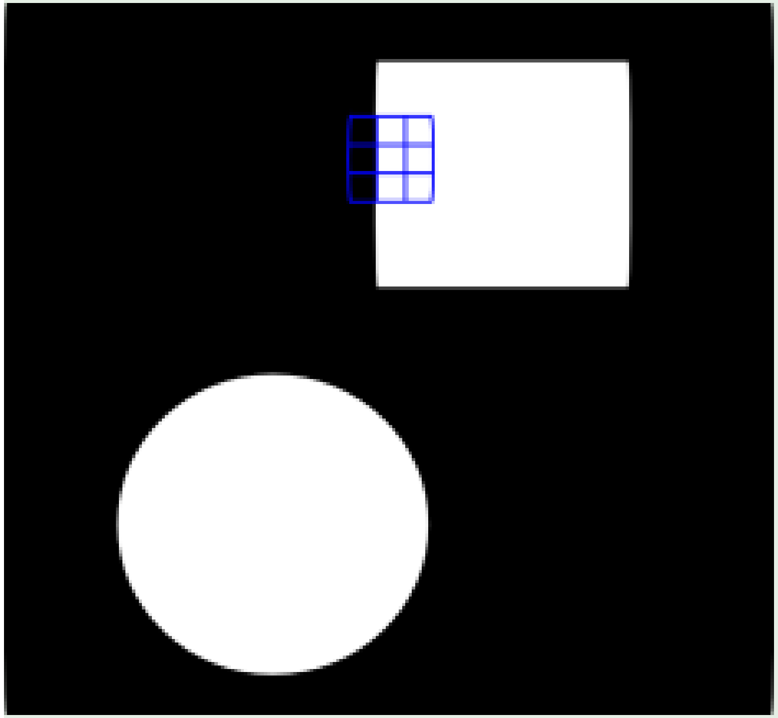


Image window:

$$f(x,y) = \begin{array}{|c|c|c|} \hline 0 & 1 & 1 \\ \hline 0 & 1 & 1 \\ \hline 0 & 1 & 1 \\ \hline \end{array}$$

$$\nabla^2 = \begin{array}{|c|c|c|} \hline 0 & -1 & 0 \\ \hline -1 & 4 & -1 \\ \hline 0 & -1 & 0 \\ \hline \end{array}$$

$$G_x = \begin{array}{|c|c|c|} \hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline \end{array}$$

Products are:

$$\nabla^2 \otimes f(x,y) = 1$$

$$G_x \otimes f(x,y) = 4$$

Gradient and Laplacian

Example

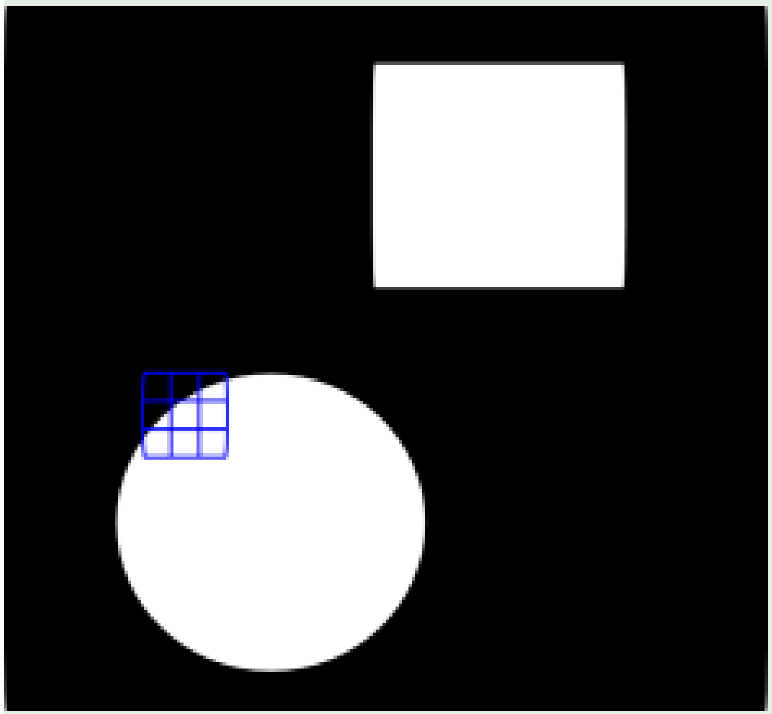


Image window:

$$f(x,y) = \begin{array}{|c|c|c|} \hline 0 & 0 & 1 \\ \hline 0 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$

$$\nabla^2 = \begin{array}{|c|c|c|} \hline 0 & -1 & 0 \\ \hline -1 & 4 & -1 \\ \hline 0 & -1 & 0 \\ \hline \end{array}$$

$$G_x = \begin{array}{|c|c|c|} \hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline \end{array}$$

Products are:

$$\nabla^2 \otimes f(x,y) = 2$$

$$G_x \otimes f(x,y) = 3$$

Gradient and Laplacian

Example

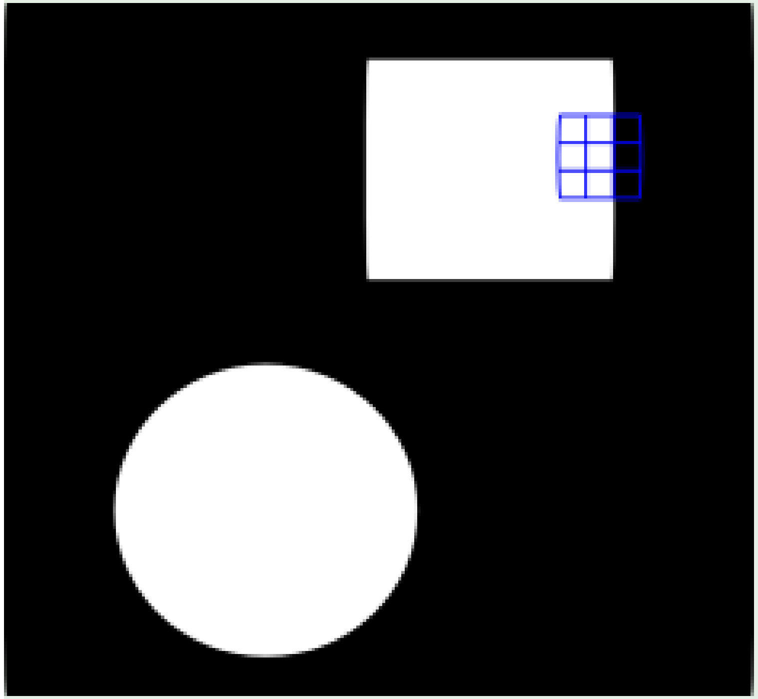


Image window:

$$f(x,y) = \begin{array}{|c|c|c|}\hline 1 & 1 & 0 \\ \hline 1 & 1 & 0 \\ \hline 1 & 1 & 0 \\ \hline \end{array}$$

$$\nabla^2 = \begin{array}{|c|c|c|}\hline 0 & -1 & 0 \\ \hline -1 & 4 & -1 \\ \hline 0 & -1 & 0 \\ \hline \end{array}$$

$$G_x = \begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline \end{array}$$

Products are:

$$\nabla^2 \otimes f(x,y) = 1$$

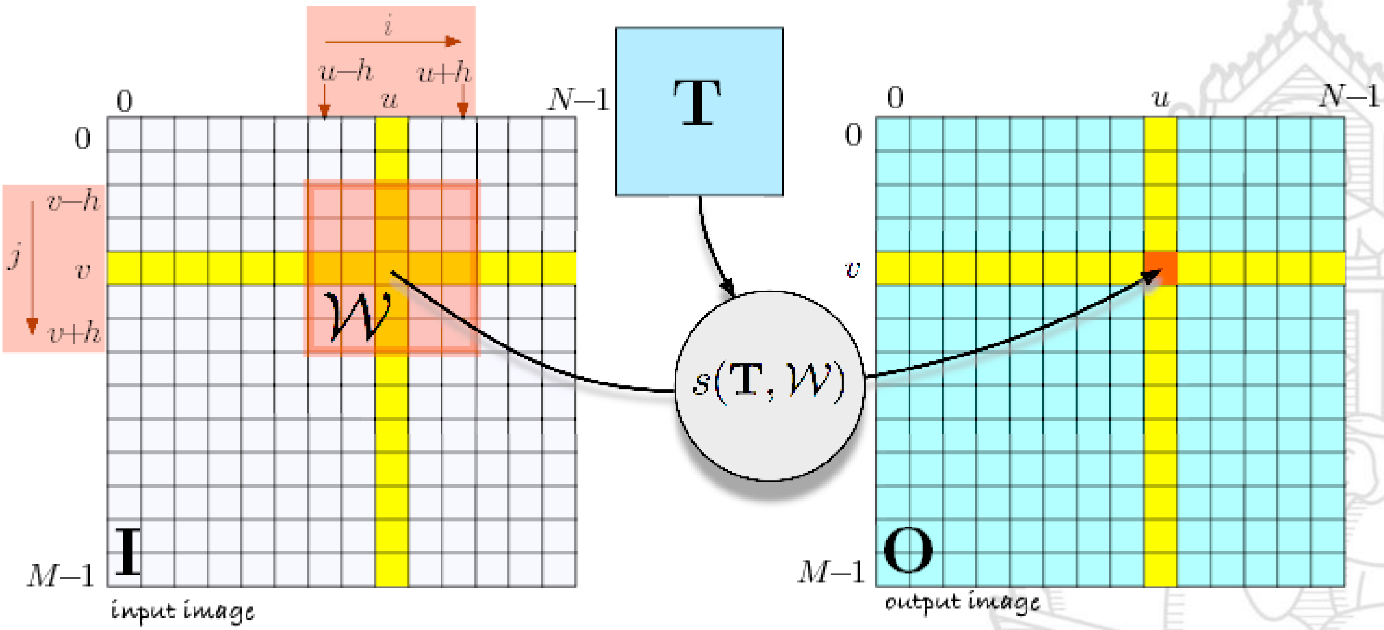
$$G_x \otimes f(x,y) = -4$$

Code sample >

```
% denoising/edge detection
dx=[-1 0 1;-2 0 1; -1 0 1];
dy=[-1 -2 -1;0 0 0;1 2 1];
K=kgauss(3);
K1=ones(19,19).*1/(19*19);
xwingDenoisMean=iconv(K1,xwing_grey);
idisp(xwingDenoisMean)
xwingDenoisGaus=iconv(K,xwing_grey);
idisp(xwingDenois)
xwingIx=iconv(dx,xwing_grey);
idisp(xwingIx)
xwingIy=iconv(dy,xwing_grey);
idisp(xwingIy)
magnGrad=sqrt(xwingIx.^2+xwingIy.^2);
idisp(magnGrad)
edgeGrad=magnGrad>250;

edgeLapl=iconv(klog(2),xwing_grey);
idisp(iint(edgeLapl)>250);

edgeLapl=iconv(klog(1),xwing_grey);
idisp(iint(edgeLapl)>250);

edgeLapl=iconv(klog(3),xwing_grey);
idisp(iint(edgeLapl)>250);
```

Template matching

$$\boldsymbol{O}[u, v] = s(\boldsymbol{T}, W), \qquad \forall (u, v) \in \boldsymbol{I}$$



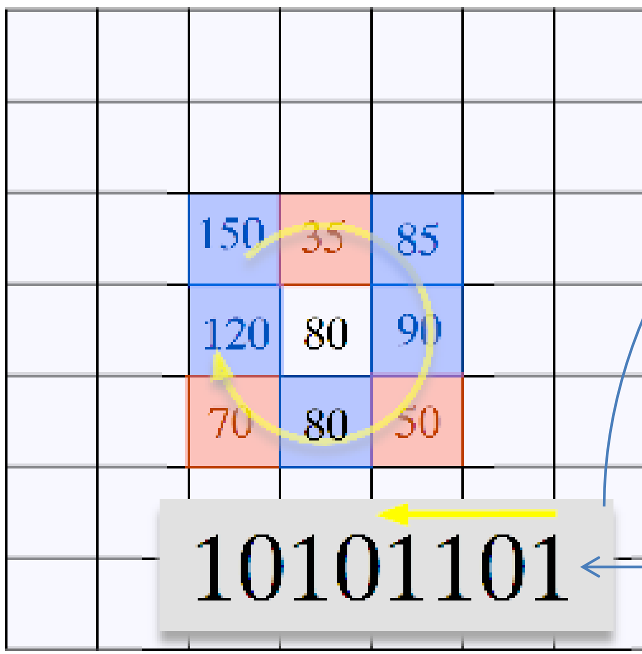
Template matching

Similarity measures

Sum of absolute differences	
SAD	$$s = \sum_{(u,v)\in \boldsymbol{I}} \lvert \boldsymbol{I}_1[u,v] - \boldsymbol{I}_2[u,v] \rvert$$
ZSAD	$$s = \sum_{(u,v)\in \boldsymbol{I}} \lvert (\boldsymbol{I}_1[u,v] - \bar{\boldsymbol{I}}_1) - (\boldsymbol{I}_2[u,v] - \bar{\boldsymbol{I}}_2) \rvert$$
Sum of squared differences	
SSD	$$s = \sum_{(u,v)\in \boldsymbol{I}} (\boldsymbol{I}_1[u,v] - \boldsymbol{I}_2[u,v])^2$$
ZSSD	$$s = \sum_{(u,v)\in \boldsymbol{I}} ((\boldsymbol{I}_1[u,v] - \bar{\boldsymbol{I}}_1) - (\boldsymbol{I}_2[u,v] - \bar{\boldsymbol{I}}_2))^2$$
Cross correlation	
NCC	$$s = \frac{\sum_{(u,v)\in \boldsymbol{I}} \boldsymbol{I}_1[u,v] \cdot \boldsymbol{I}_2[u,v]}{\sqrt{\sum_{(u,v)\in \boldsymbol{I}} \boldsymbol{I}_1^2[u,v] \cdot \sum_{(u,v)\in \boldsymbol{I}} \boldsymbol{I}_2^2[u,v]}}$$
ZNCC	$$s = \frac{\sum_{(u,v)\in \boldsymbol{I}} (\boldsymbol{I}_1[u,v] - \bar{\boldsymbol{I}}_1) \cdot (\boldsymbol{I}_2[u,v] - \bar{\boldsymbol{I}}_2)}{\sqrt{\sum_{(u,v)\in \boldsymbol{I}} (\boldsymbol{I}_1[u,v] - \bar{\boldsymbol{I}}_1)^2 \cdot \sum_{(u,v)\in \boldsymbol{I}} (\boldsymbol{I}_2[u,v] - \bar{\boldsymbol{I}}_2)^2}}$$

Non-parametric similarity measures

Census



$$s(x) = \begin{cases} 1, if\ x > R \\ 0, otherwise \end{cases}$$

Census representation

Hamming distance

Rank transform = 5

Sant'Anna
Scuola Universitaria Superiore Pisa

Non-parametric similarity measures

Rank transform is more compact but does not encode position information

50	10	205
1	25	2
102	250	240

Census: 01110101

Rank: 5

Hamming distance: 6!

10	26	2
101	25	202
1	250	214

Census: 10111010

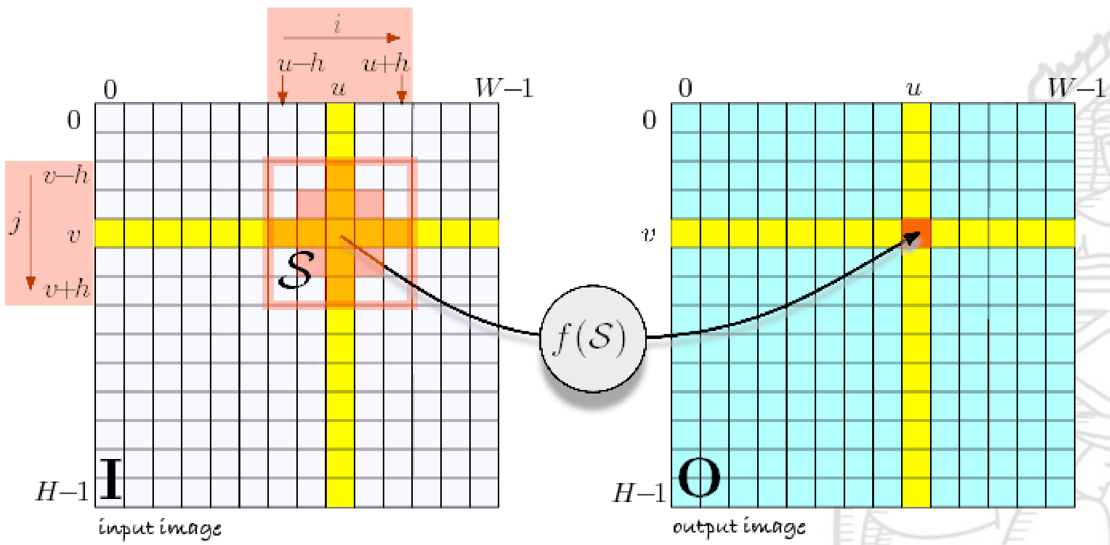
Rank: 5

Non-linear operators

- Variance measure (on windows): Edge detection
- Median filter: noise removal
- Rank transform: non-local maxima suppression

Mathematical morphology

$$\boldsymbol{O}[u,v] = f(\boldsymbol{I}[u+i, v+j]), \qquad \forall (i,j) \in S, \forall (u,v) \in \boldsymbol{I}$$



Erosion

Erosion is a specific procedure of the more general Morphological Image Processing techniques.

It belongs to the concept of mathematical morphology and it is strictly related to the set theory.

Here the concept is roughly introduced to understand the basis of erosion.

Notation

Let consider A as a set in Z^2

Definition $a = (a_1,a_2)$ **belongs to** A

We write:

$$a \in A$$

Definition $a = (a_1,a_2)$ **does not belong to** A

We write:

$$a \notin A$$

A set is represented by the parenthesis$\{\cdot\}$.

In our case, the elements of a set are the pixels belonging to a certain area or object of an image. When we write:

Example

$$C = \{w \mid w = -d,\ per\ d \in D\}$$

This means that C is composed by all the elements w which are obtained by scalar product of the elements of D and the value -1.

Definition

When all elements of A are also elements of B, we say that A is a subset of B.

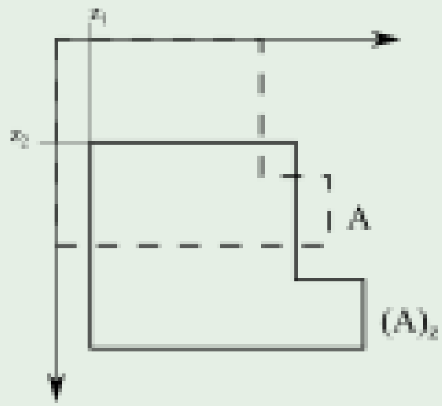
$$A \subseteq B$$

Definition

The translation of a set A by an element z, is represented as $(A)_z$ and is defined by:

$$(A)_z = \{c \mid c = a + z,\ per\ a \in A\}$$

Example



Now we can write the morphological operation which interest us, thus:

Definition

$$A \ominus B = \{z | (B)_z \subseteq A\}$$

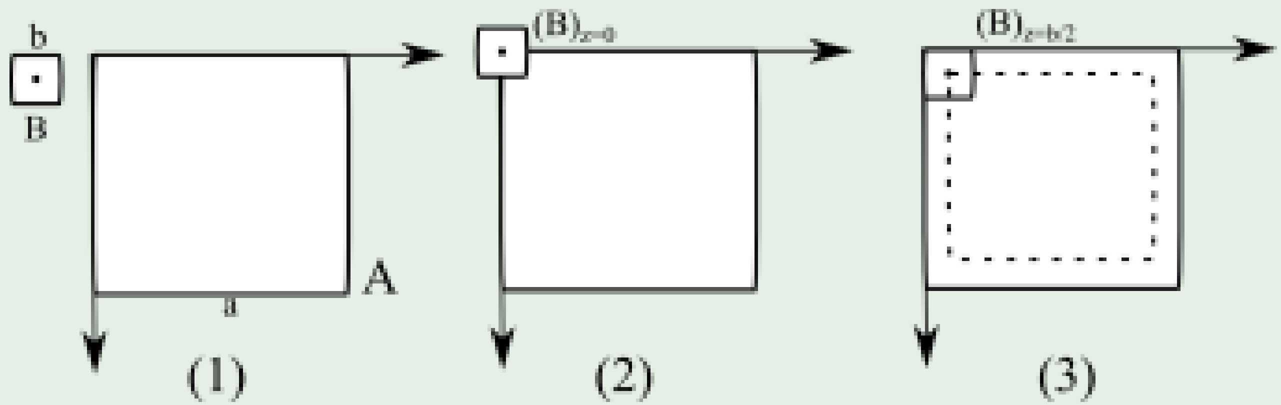
This definition represents an:

erosion

The verbose definition is: the erosion of *A* through *B* is the set of all the points *z* whom the translation of *B* by *z* is a subset of *A*.

It’s simple to see that graphically:

Example



1. I due insiemi A e B.
2. $(B)_0 \subseteq A$? No z=0
3. $(B)_{b/2} \subseteq A$? Sì z=b/2

In this case the eroded set will be;

$$A \ominus B = [\frac{b}{2} : a - \frac{b}{2}; \frac{b}{2} : a - \frac{b}{2}]$$

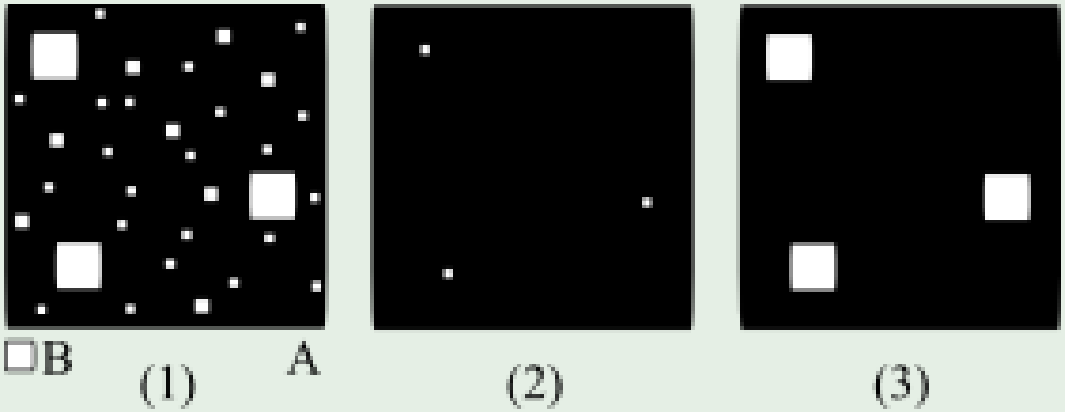
We can figure the erosion as a “shape-cutting” of the most external part of the set.

Dilation is the «opposite» operation, but formally they are related by:

$$A \oplus B = \overline{\bar{A} \ominus B}$$

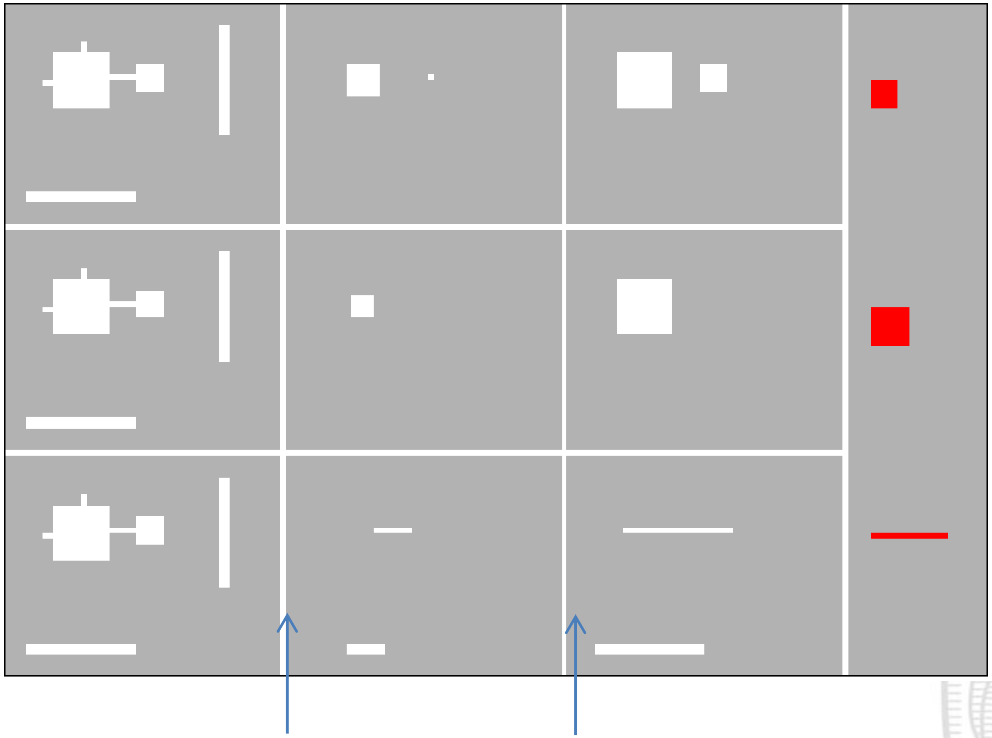
Which means that eroding the white pixels is the same as dilating the dark pixels, and vice versa.

Example



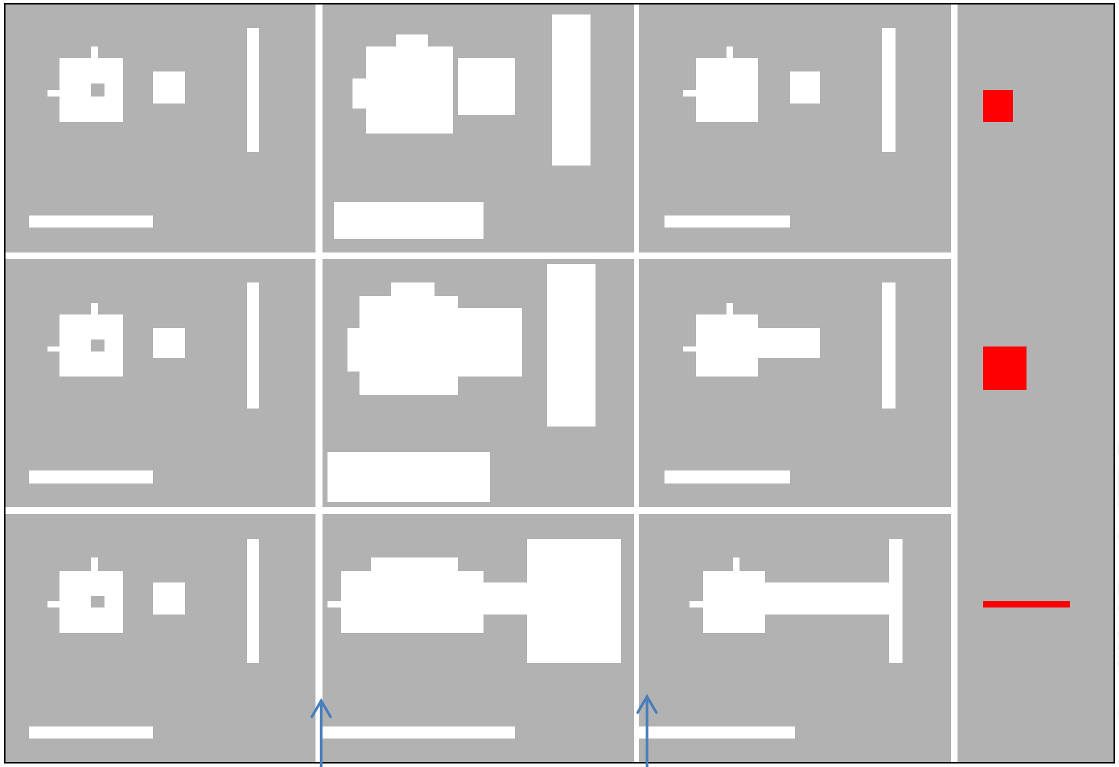
1) Original image
2) Erosion by the element B
3) Dilatation (the opposite procedure of the Erosion)

Example: opening



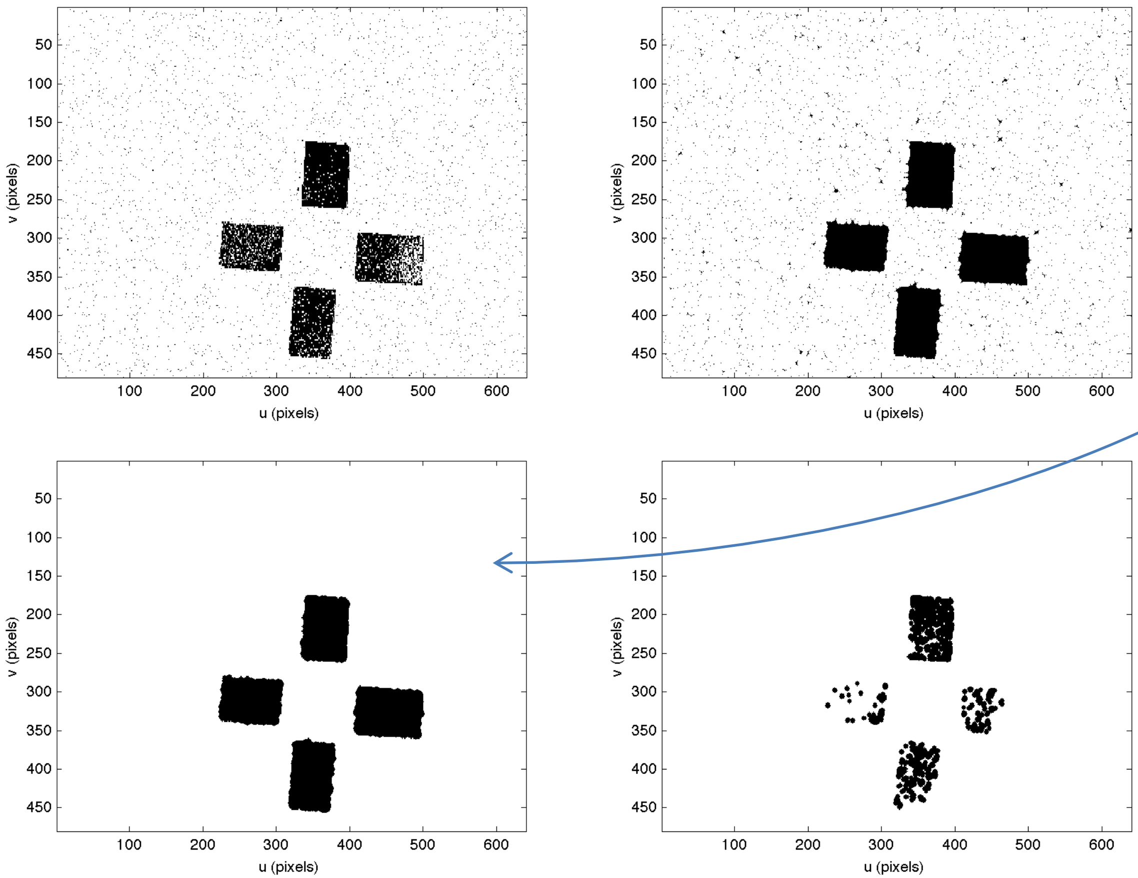
$$O = I \ominus S \qquad O = I \oplus S$$

Example: closing



$O = I \oplus S$ $\qquad$ $O = I \ominus S$

Noise removal & boundary detection



$$O = I \ominus S$$

$$O_2 = I - O$$

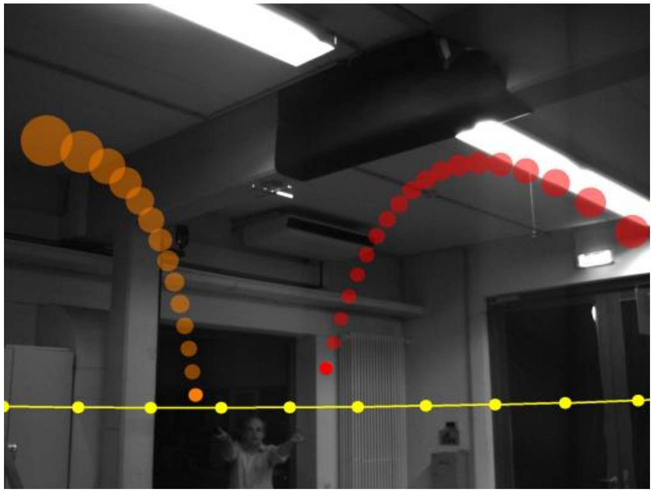
?

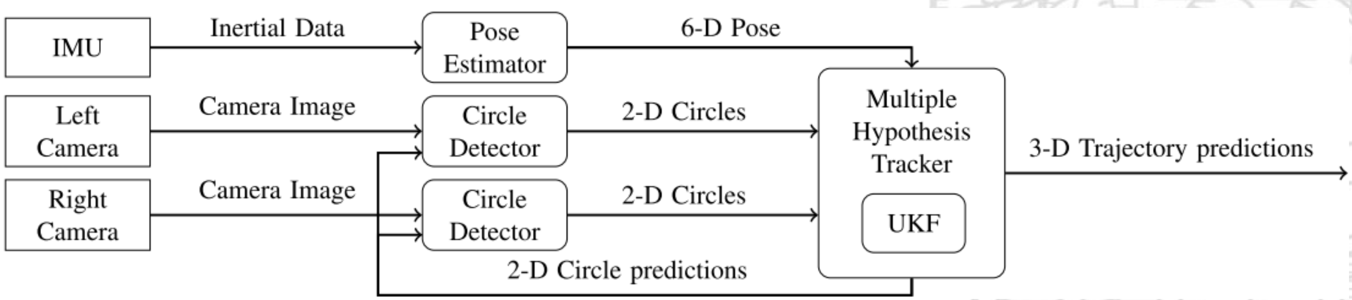
Matlab sample of buondary detection

Shape changing

- Cropping
- Reisizing
- Rotating
- …

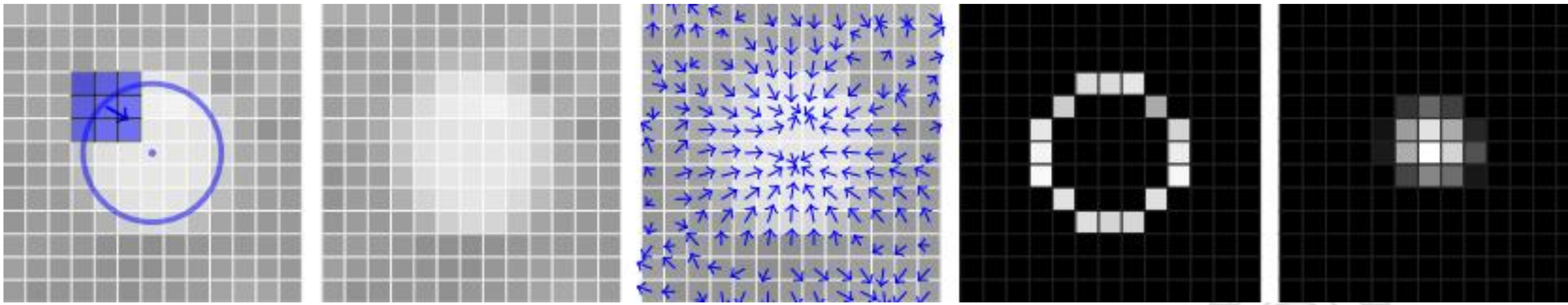
Example DLR





Oliver Birbach, Udo Frese and Berthold Bauml, (2011) ‘Realtime Perception for Catching a Flying Ball with a Mobile Humanoid’

Example DLR



$$C = \frac{\sqrt{2}\left(\begin{pmatrix}+1 & 0 & -1\\ +2 & 0 & -2\\ +1 & 0 & -1\end{pmatrix} * I,\ \begin{pmatrix}+1 & +2 & +1\\ 0 & 0 & 0\\ -1 & -2 & -1\end{pmatrix} * I\right)^T}{\sqrt{16\begin{pmatrix}1 & 2 & 1\\ 2 & 4 & 2\\ 1 & 2 & 1\end{pmatrix} * I^2 - \left(\begin{pmatrix}1 & 2 & 1\\ 2 & 4 & 2\\ 1 & 2 & 1\end{pmatrix} * I\right)^2 + \varepsilon^2}}$$

$$R(x,y,\alpha) = \left(\begin{pmatrix}\cos\alpha\\ \sin\alpha\end{pmatrix} \cdot C(x,y)\right)^2 = |C(x,y)|^2 \cdot \cos^2\delta$$

$$CR(x_c,y_c,r) = \frac{1}{2\pi}\int_{\alpha=0}^{2\pi} R(x_c + r\cos\alpha, y_c + r\sin\alpha, \alpha)\, d\alpha$$

Part 2

Feature extraction

Image feature extraction

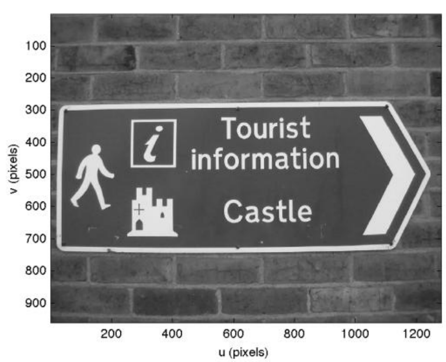
We need to be able to answer pithy questions such as what is the pose of the object? what type of object is it? how fast is it moving? how fast am I moving? and so on. The answers to such questions are measurements obtained from the image and which we call image features. Features are the gist of the scene and the raw material that we need for robot control.

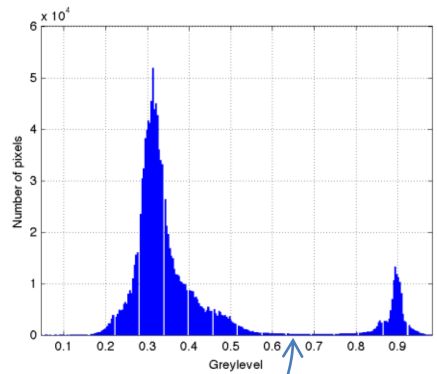
The image processing operations from the last chapter operated on one or more input images and returned another image. In contrast feature extraction operates on an image and returns one or more image features.

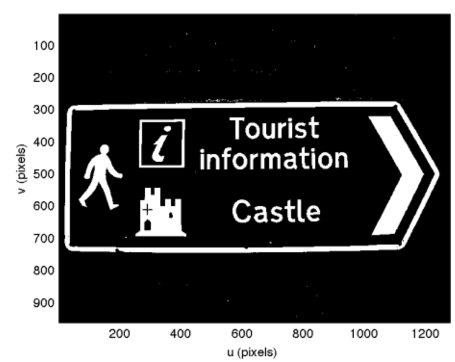
Image feature extraction is a necessary first step in using image data to control a robot. It is an information concentration step that reduces the data rate from 10 6 −10 8 bytes s –1 at the output of a camera to something of the order of tens of features per frame that can be used as input to a robot's control system.

Region-features classification

Thresholding







$t = 0.65$

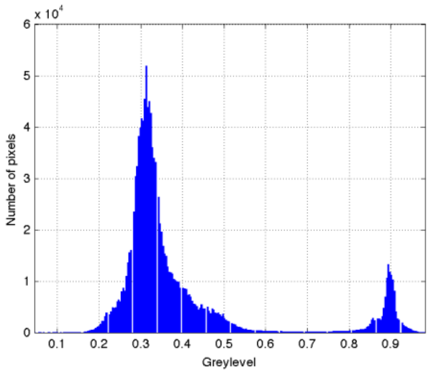
$$\boldsymbol{O}[u,v] = \begin{cases} 1, & if\ \boldsymbol{I}[u,v] > t \\ 0, & if\ \boldsymbol{I}[u,v] \leq t \end{cases}$$

How we select this value?

Otsu

Background and object can be described as classes of the image histogram

Otsu thresholding method maximize the variance between classes.



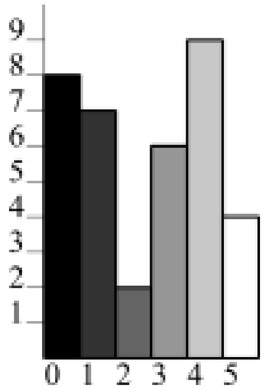
One implementation can be defined as follow:

1. Obtaining the image histogram
2. For each threshold value, $t = 0, \ldots, L-1$ the following variables should be derived
3. Compute:

$$
\begin{aligned}
\mu_s{}^t &= \frac{\sum_{i=0}^{t} \#(i) \cdot i}{\#s} \\
\mu_o{}^t &= \frac{\sum_{i=t+1}^{L-1} \#(i) \cdot i}{\#o} \\
W_s &= \frac{\#s}{\#s + \#o} \\
W_o &= \frac{\#o}{\#s + \#o} \\
\sigma_b^2 &= W_s W_o (\mu_s{}^t - \mu_o{}^t)^2
\end{aligned}
$$

4. Maximum $\sigma^2{}_b\,(t)$ defines the correct threshold t.

Otsu



Per $t = 0$:

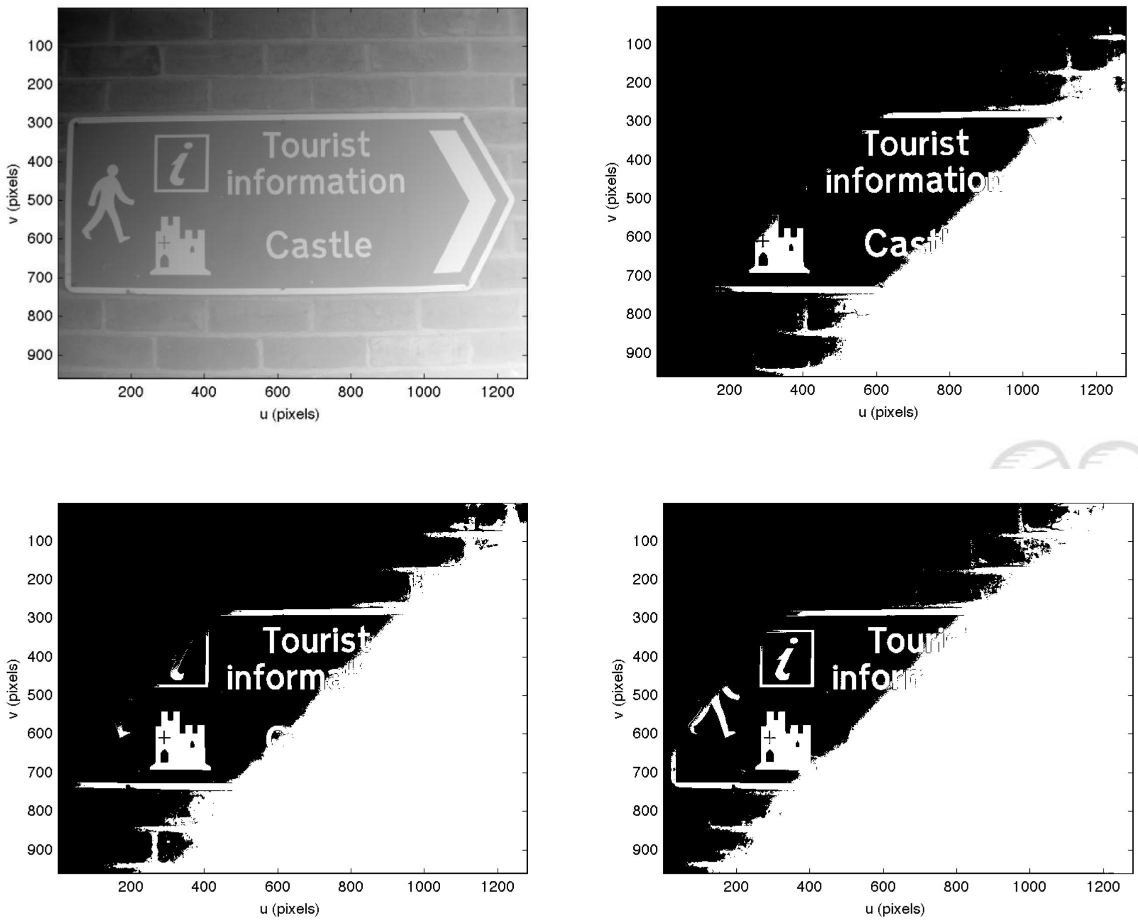
$$
\begin{aligned}
W_s &= \frac{8}{36} = 0.22 \\
\mu_s &= \frac{8 \cdot 0}{8} = 0 \\
W_o &= \frac{7+2+6+9+4}{36} = \frac{28}{36} = 0.78 \\
\mu_o &= \frac{7 \cdot 1 + 2 \cdot 2 + 6 \cdot 3 + 9 \cdot 4 + 4 \cdot 5}{28} = 3.04 \\
\sigma_b^2 &= 0.22 \cdot 0.78(3.04 - 0)^2 = 1.59
\end{aligned}
$$

Threshold	t=0	t=1	t=2	t=3	t=4
Variance	1.59	2.56	2.63	2.14	0.87

Code sample >

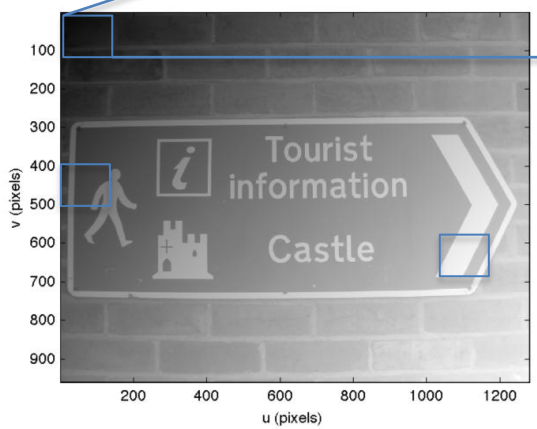
```
%OTSU
street=iread('street.png');
idisp(street);
idisp(street>t);
[rig,col]=size(street);
[n,v]=ihist(street);
plot(v,n)

max=0;
occ=n;
for t=1:256
sum_tmp=0;
sum_sf=0;
media_sf=0;
sum_ogg=0;
media_ogg=0;
peso_sf=0;
peso_ogg=0;
for i=1:t
     sum_tmp=occ(i)*i+sum_tmp;
     sum_sf=occ(i)+sum_sf;
  end

  media_sf=sum_tmp/sum_sf;
  sum_tmp=0;

  for j=t+1:256
    sum_tmp=occ(j)*j+sum_tmp;
    sum_ogg=occ(j)+sum_ogg;
  end
  media_ogg=sum_tmp/sum_ogg;

  peso_sf=sum_sf/(sum_ogg+sum_sf);
  peso_ogg=sum_ogg/(sum_ogg+sum_sf);

  var=peso_sf*peso_ogg*(media_sf-media_ogg)^2;

  if var>max
     max=var;
     soglia=t;
  end

end
```

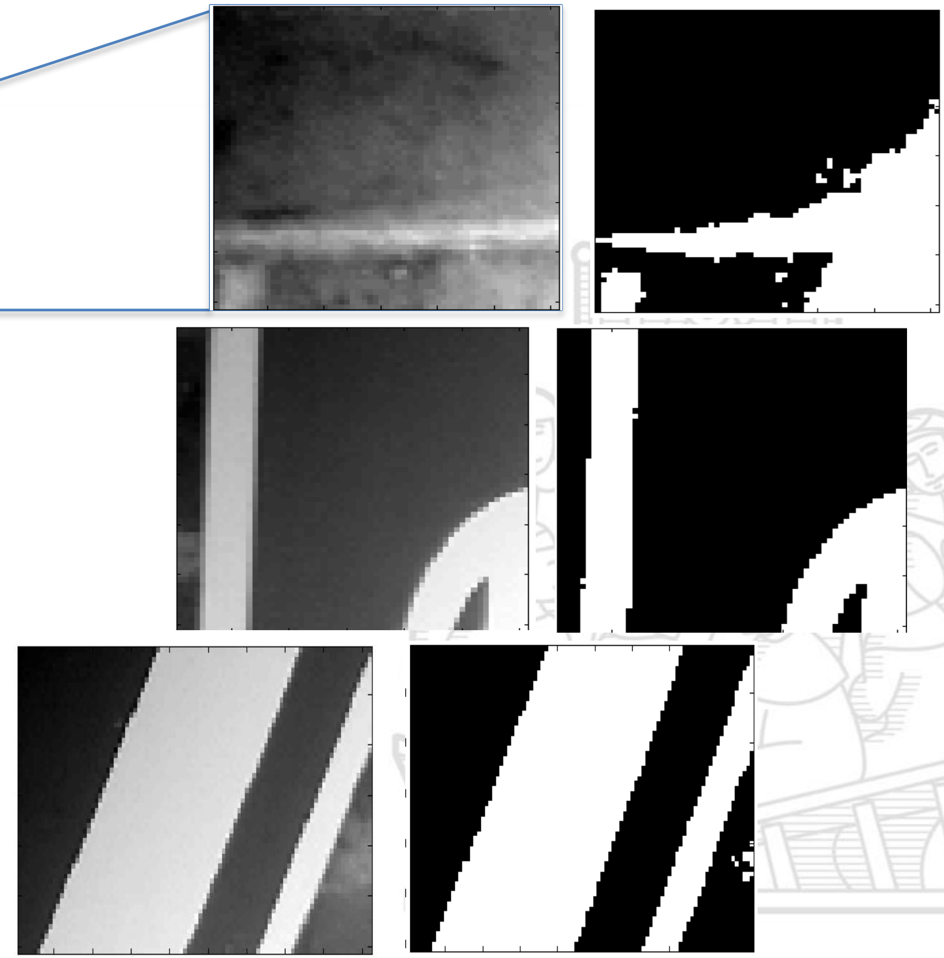
Illumination problem



Local thresholding

We can split the image in smaller ones, and thresholding **locally** the various portions.



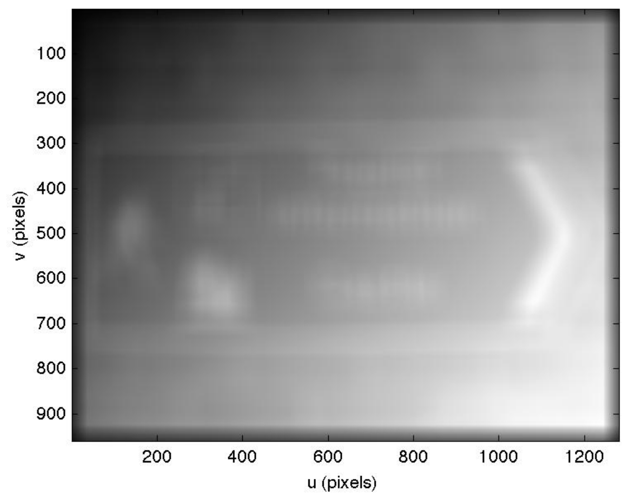


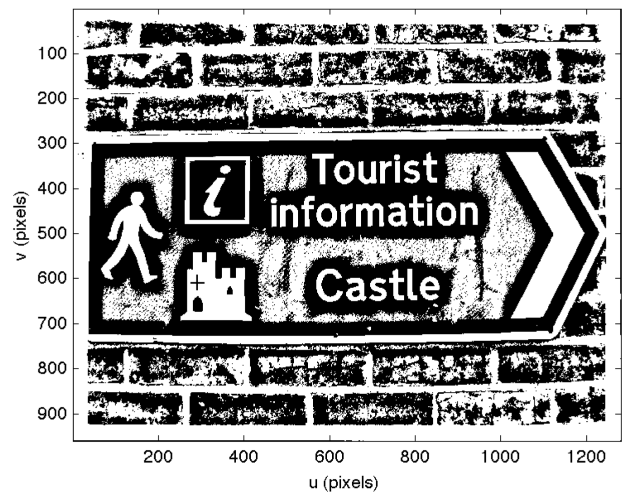
The final image is the collection of the smaller ones.

Local thresholding

Niblack algorithm used a local threshold

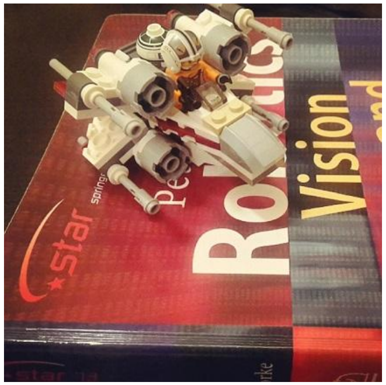
$$\boldsymbol{t}[u,v] = \mu(W) + k\sigma(W)$$

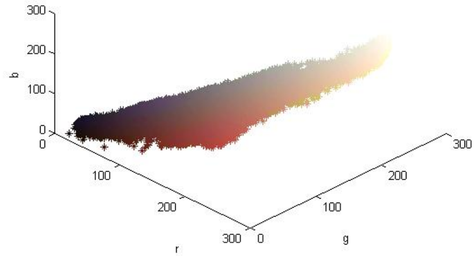


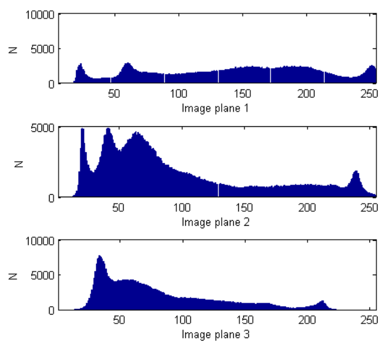


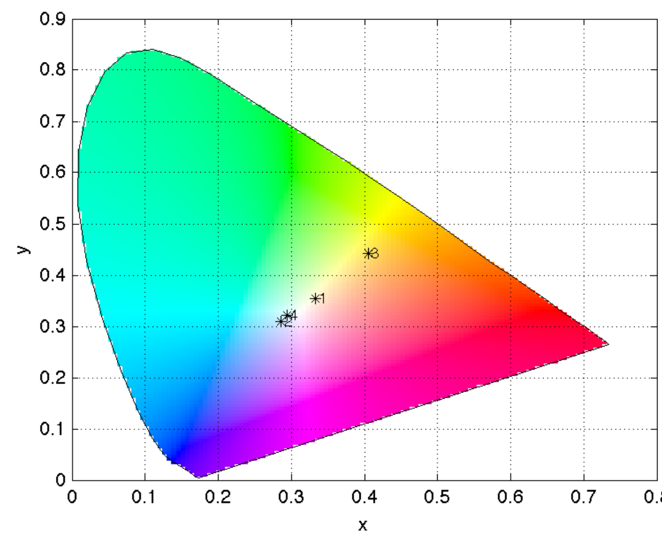
k=-2

Colour classification



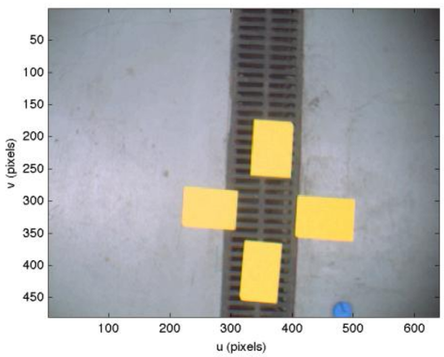




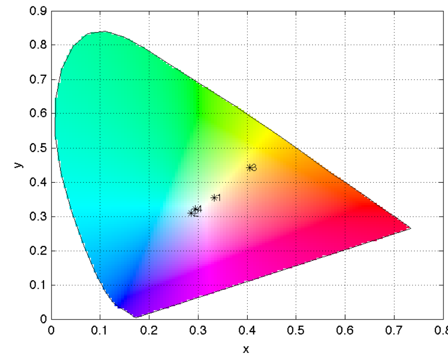


K-means classification

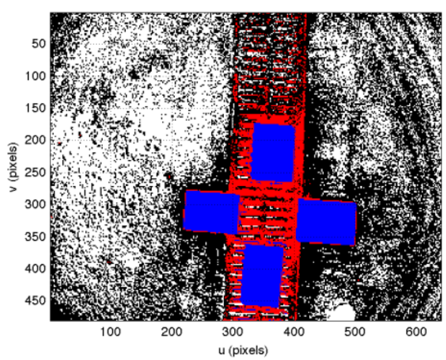
To begin with, *k*-centre points (which define *k* clusters) are randomly initialized into a *n*-dimensional points space. Each unknown point is assigned to the closest centre point (thus to belong to the corresponding cluster), then the centre point positions are updated to be the mean of all points assigned to the cluster

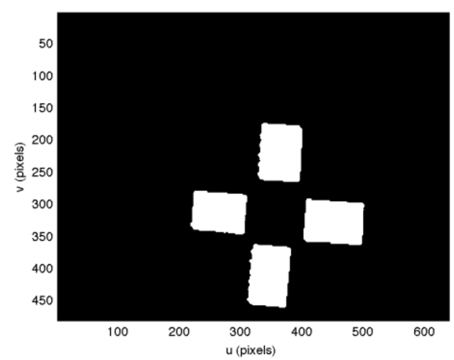


Original image

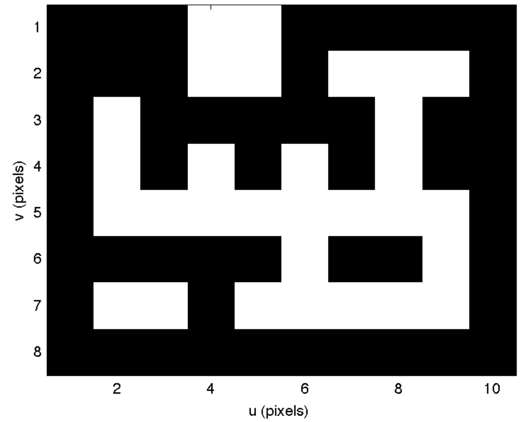


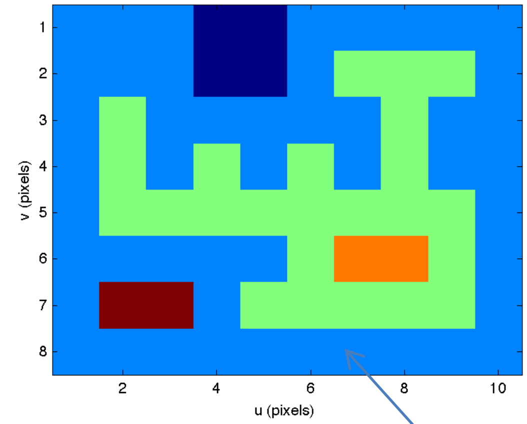
Cluster after k-means iterations





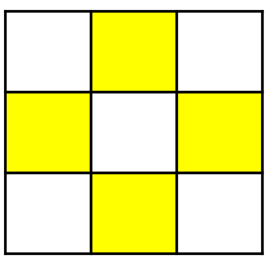
Connected components



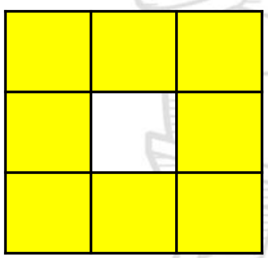


5 labels

4 - neighbourhood

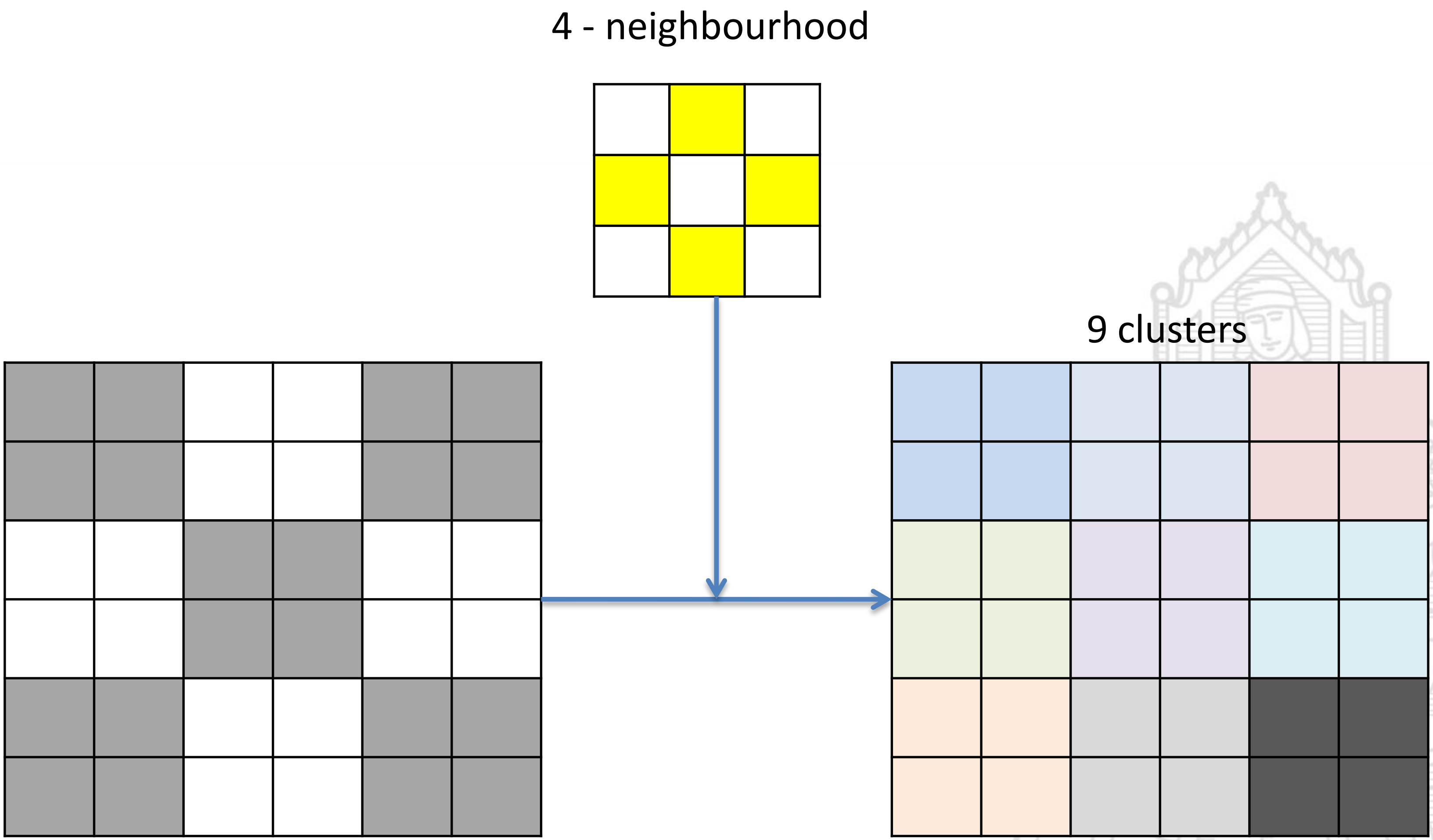


8 - neighbourhood

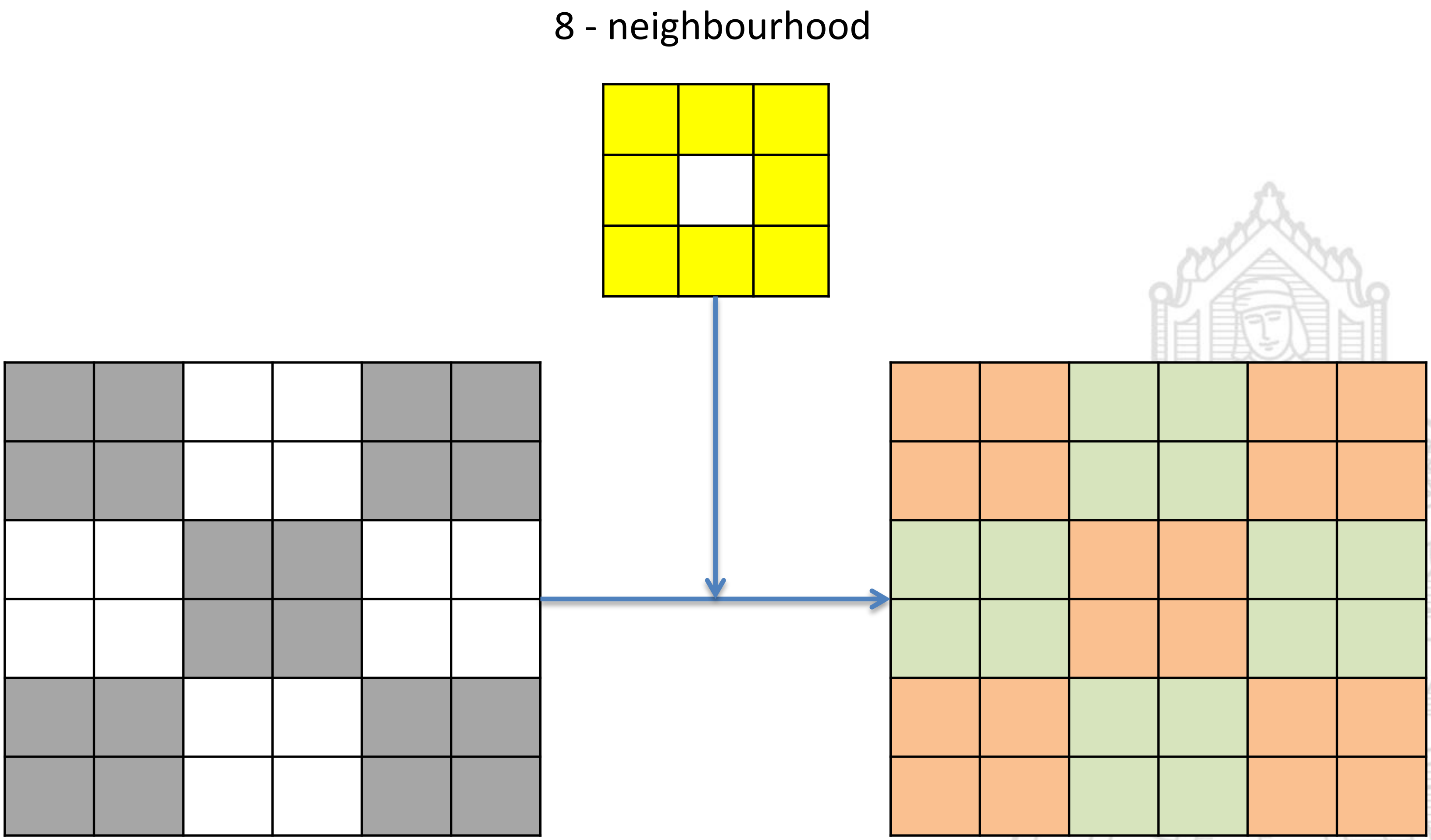


Sant'Anna
Scuola Universitaria Superiore Pisa

Connected components

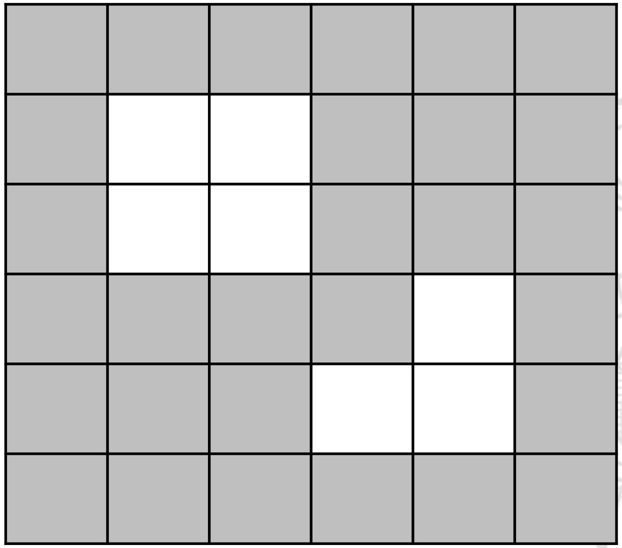


Connected components



The motivated student algorithm

To compute the connected components of an image, we first (conceptually) split the image into horizontal runs of adjacent pixels, and then color the runs with unique labels, re-using the labels of vertically adjacent runs whenever possible. In a second phase, adjacent runs of different colors are then merged.



The motivated student algorithm

To compute the connected components of an image, we first (conceptually) split the image into horizontal runs of adjacent pixels, and then color the runs with unique labels, re-using the labels of vertically adjacent runs whenever possible. In a second phase, adjacent runs of different colors are then merged.

1	1	1	1	1	1
2	3	3	4	4	4
5	6	6	7	7	7
8	8	8	8	9	10
11	11	11	12	12	13
14	14	14	14	14	15

16	17	20	23	26	29
16	18	21	23	26	29
16	18	21	23	26	29
16	19	22	23	27	29
16	19	22	24	27	29
16	19	22	25	28	29

The motivated student algorithm

3 -18	6 -18	3 -21	6 -21	→	18 = 3
3 -3	6 -3	3 -21	6 -21	→	6 = 3
3 -3	3 -3	3 -21	3 -21	→	21 = 3
3 -3	3 -3	3 -3	3 -3	→	No more merges required

The motivated student algorithm

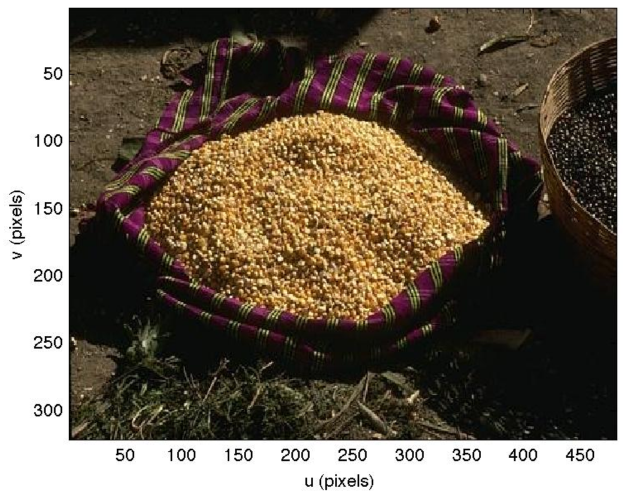
$\boxed{9\text{-}27}$	12-24	12-27
9-9	$\boxed{12\text{-}24}$	12-9
9-9	12-12	$\boxed{12\text{-}9}$
9-9	9-9	9-9

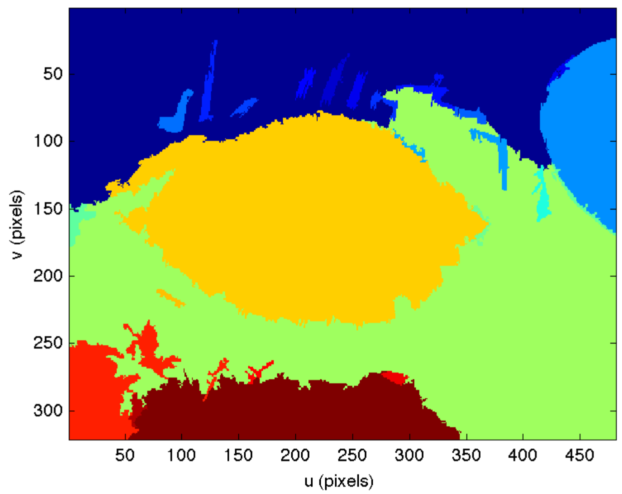
9-27 12-24 12-27

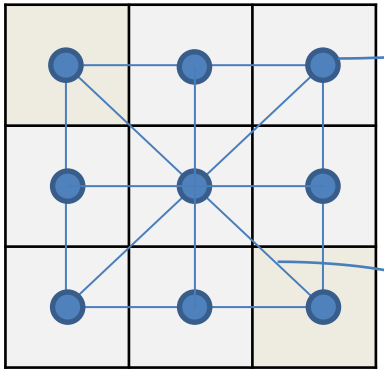
1	1	1	1	1	1
2	3	3	4	4	4
5	3	3	7	7	7
8	8	8	8	9	10
11	11	11	12	12	13
14	14	14	14	14	15

16	17	20	23	26	29
16	3	3	23	26	29
16	3	3	23	26	29
16	19	22	23	27	29
16	19	22	24	27	29
16	19	22	25	28	29

Graph-based segmentation







Nodes v_n

Edge e_q

$q = 1 \dots 16$

$n = 1 \dots 9$

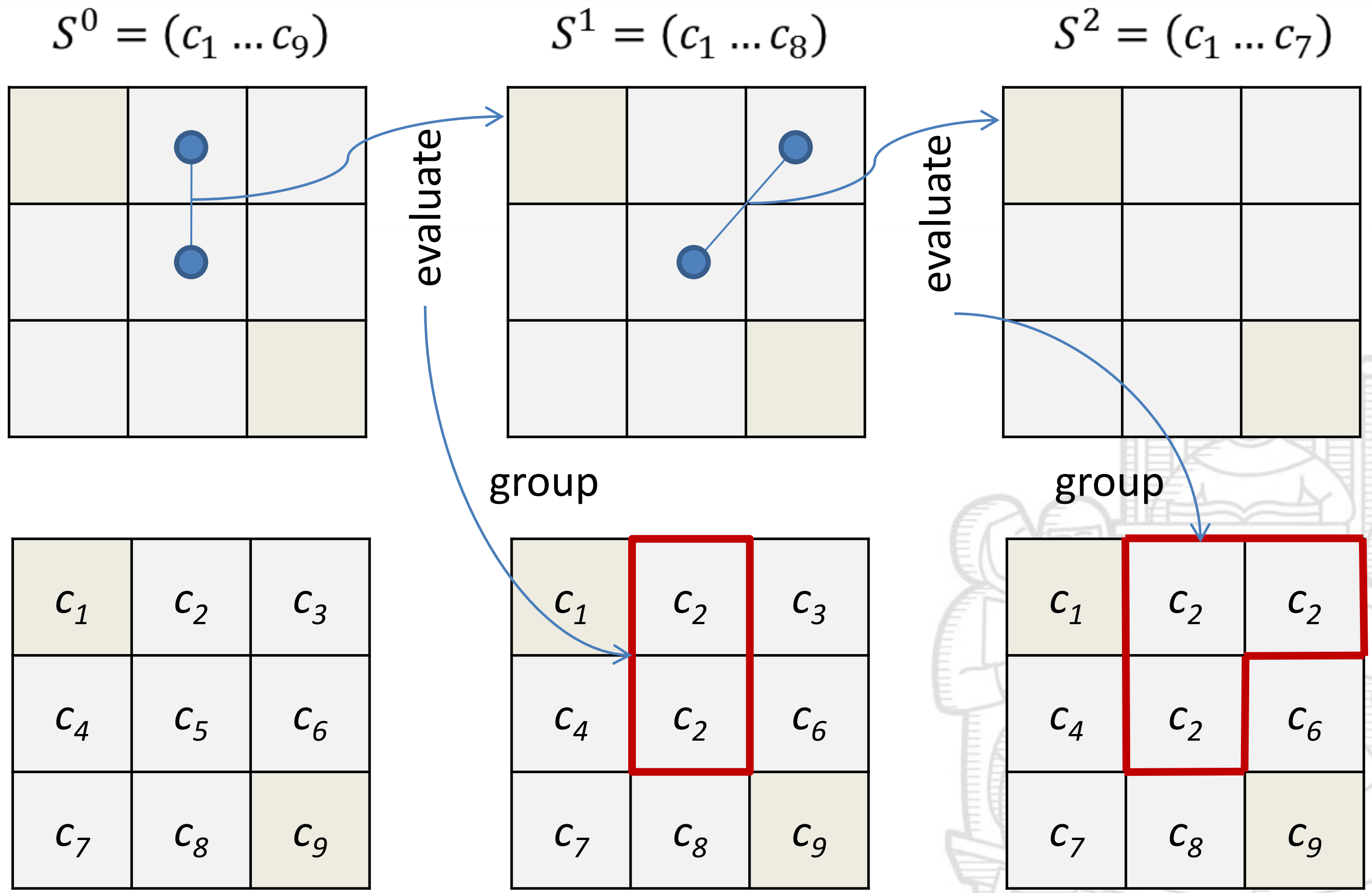
weight= $\left|color\ v_i - color\ v_j\right|$

Graph-based segmentation

The input is a graph $G = (V, E)$, with n vertices and m edges. The output is a segmentation of V into components $S = (C_1, \ldots, C_r)$.

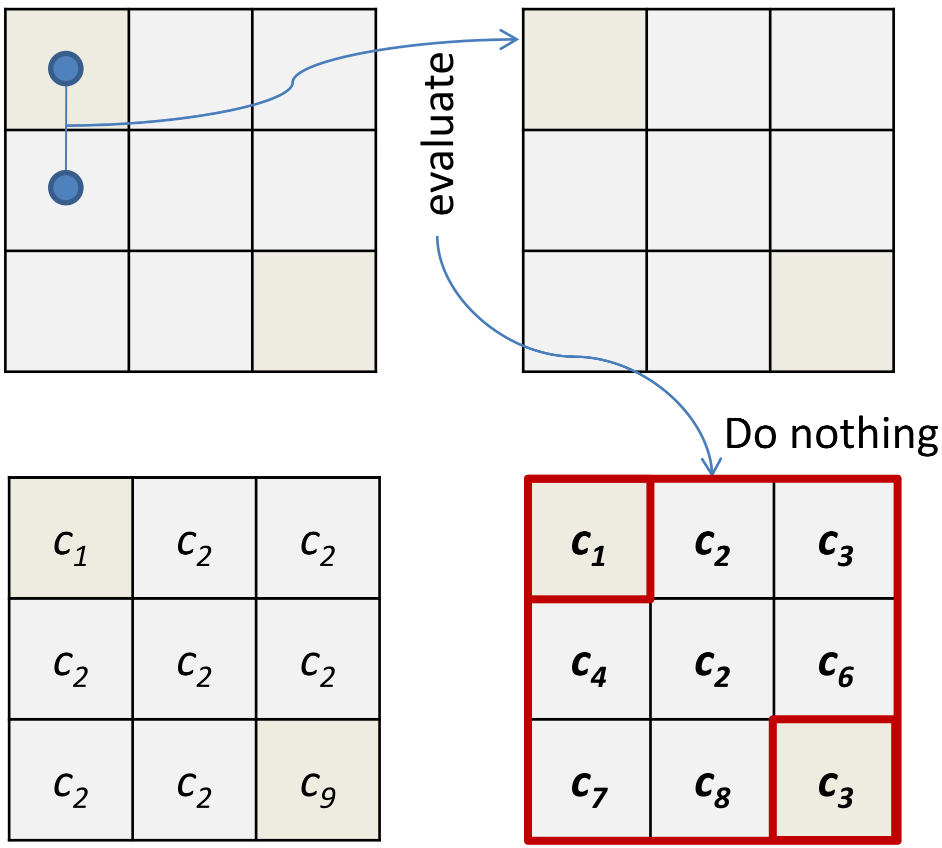
0. Sort E into $\pi = (o_1, \ldots, o_m)$, by non-decreasing edge weight.
1. Start with a segmentation S^0, where each vertex v_i is in its own component.
2. Repeat step 3 for $q = 1, \ldots, m$.
3. Construct S^q given S^{q-1} as follows. Let v_i and v_j denote the vertices connected by the q-th edge in the ordering, i.e., $o_q = (v_i, v_j)$. If v_i and v_j are in disjoint components of S^{q-1} and $w(o_q)$ is small compared to the internal difference of both those components, then merge the two components otherwise do nothing. More formally, let C_i^{q-1} be the component of S^{q-1} containing v_i and C_j^{q-1} the component containing v_j. If $C_i^{q-1} \neq C_j^{q-1}$ and $w(o_q) \leq MInt(C_i^{q-1}, C_j^{q-1})$ then S^q is obtained from S^{q-1} by merging C_i^{q-1} and C_j^{q-1}. Otherwise $S^q = S^{q-1}$.
4. Return $S = S^m$.

Pedro F. Felzenszwalb, Daniel P. Huttenlocher, International Journal of Computer Vision , September 2004, Volume 59, Issue 2, pp 167-181



$S^0 = (c_1 \dots c_9)$
$S^1 = (c_1 \dots c_8)$
$S^2 = (c_1 \dots c_7)$
evaluate
evaluate
group
group
c_1 c_2 c_3
c_4 c_5 c_6
c_7 c_8 c_9
c_1 c_2 c_3
c_4 c_2 c_6
c_7 c_8 c_9
c_1 c_2 c_2
c_4 c_2 c_6
c_7 c_8 c_9

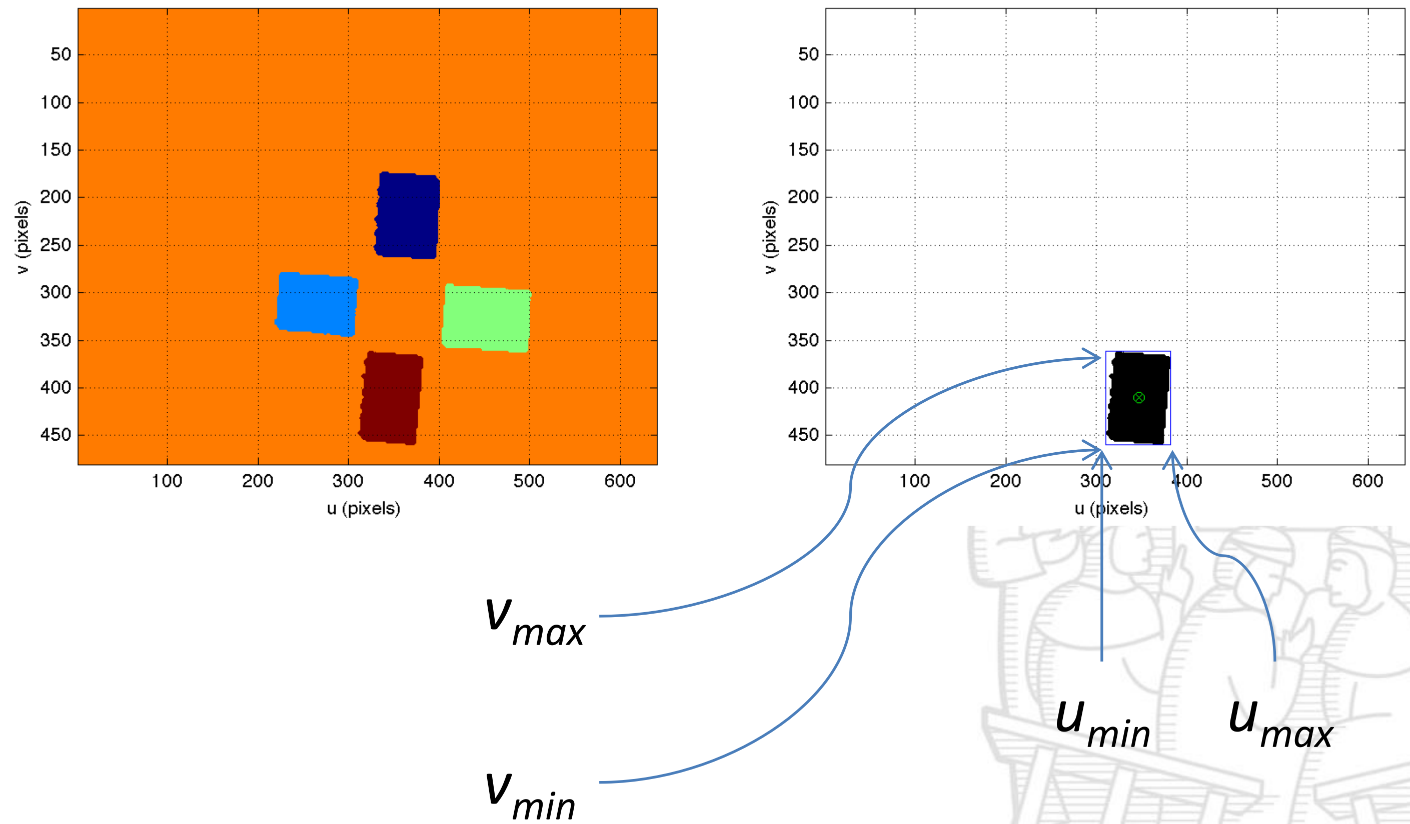
$$S^j = (c_1, c_2, c_3)$$



Final labelling

Concise description:

Bounding boxes



Concise description:

Moments

$$m_{pq} = \sum_{(u,v)\in \boldsymbol{I}} u^p v^q \boldsymbol{I}[u,v]$$

Where *(p+q)* is the order of the moment

$m_{00} = \sum \boldsymbol{I}[u,v]$ is the area of a region

Centroid of the region is located in:

$$u_c = \frac{m_{10}}{m_{00}} \qquad v_c = \frac{m_{01}}{m_{00}}$$

Central moments μ_{pq} are computed with respect to the centroid

$$\mu_{pq} = \sum_{(u,v)\in I} (u - u_c)^p (v - v_c)^q \boldsymbol{I}[u, v]$$

Central moments are related to moments m_{pq} by:

$$\mu_{10} = 0 \qquad \mu_{01} = 0 \qquad \mu_{11} = m_{11} - \frac{m_{10}m_{01}}{m_{00}}$$

$$\mu_{20} = m_{20} - \frac{m_{10}^2}{m_{00}} \qquad \mu_{02} = m_{02} - \frac{m_{01}^2}{m_{00}}$$

By using a thin plate analogy we can write the inertia matrix:

$$\boldsymbol{J} = \begin{bmatrix} \mu_{20} & \mu_{11} \\ \mu_{11} & \mu_{02} \end{bmatrix}$$

About axis parallel to *u*-*v*-axes and intersecting the centroid

Computing eigenvalues λ_1, λ_2 , we can compute an equivalent ellipse from $\boldsymbol{J}$:

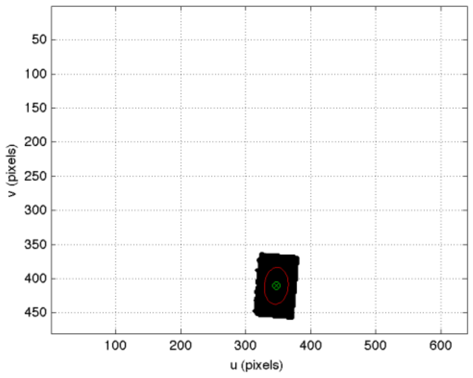
$$a = 2\sqrt{\frac{\lambda_2}{m_{00}}} \qquad b = 2\sqrt{\frac{\lambda_1}{m_{00}}}$$

Principal axis with $\lambda_2 > \lambda_1$

$$\theta = \tan^{-1}\frac{v_y}{v_x}$$

Eigenvectors

Orientation



Features invariance

Some region features are invariant with respect to certain transformations.

$$\frac{a}{b}$$

$$\rho = \frac{4\pi m_{00}}{p^2}$$

$$\phi_1 = \eta_{20} + \eta_{02}$$
$$\phi_2 = (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2$$
$$\phi_3 = (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2$$
$$\phi_4 = (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2$$
$$\phi_5 = (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})\left[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2\right] + (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})\left[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2\right]$$
$$\phi_6 = (\eta_{20} - \eta_{02})\left[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2\right] + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03})$$
$$\phi_7 = (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})\left[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2\right] + (3\eta_{12} - \eta_{30})(\eta_{21} + \eta_{03})\left[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2\right]$$

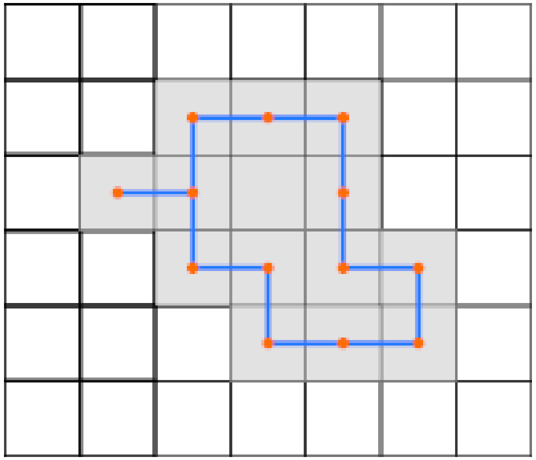
	Translation	Rotation	Scale
Area	Y	Y	N
Centroid	N	Y	Y
Aspect ratio	Y	Y	Y
Orientation	Y	N	Y
Circularity	Y	Y	Y
Hu moments	Y	Y	Y

With normalized moments:

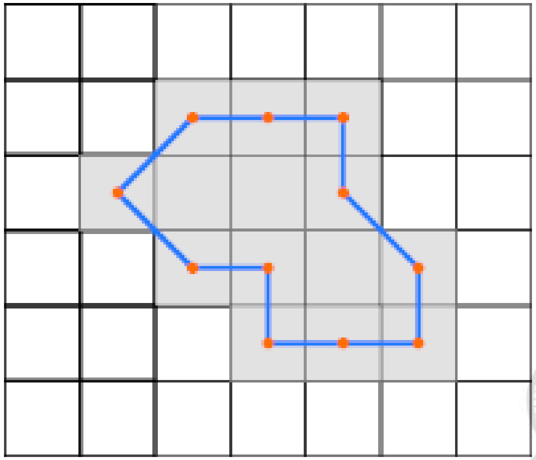
$$\eta_{pq} = \frac{\mu_{pq}}{\mu_{00}^{\gamma}}, \gamma = \frac{1}{2}(p+q) + 1, p + q = 2, 3, \dots$$

Boundary representation

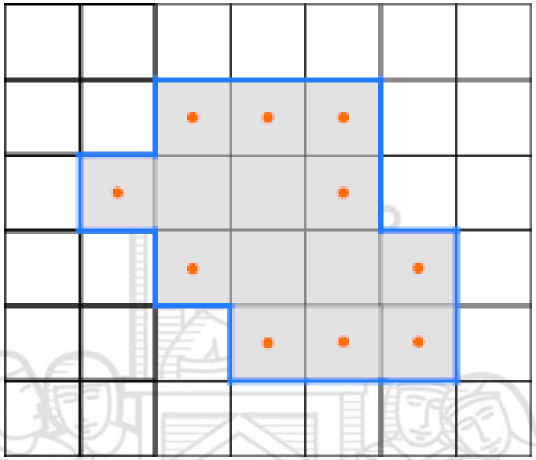
The region is described by the shape of its perimeter



Chain code 4 direction



Chain code 8 direction



Crack code

A chain code is a list of the outermost pixels linked by short line segments, with different orientations depending on the chain code used.

The crack code has its segment between the region and the pixels outside.

Note that chain codes representations underestimate the actual perimeter.

Line features

Hough transformation

It transforms the original image in an accumulation matrix in the parameters plane.

Every points that belong to the researched function (line, circle,. . .) increase the accumulation value.

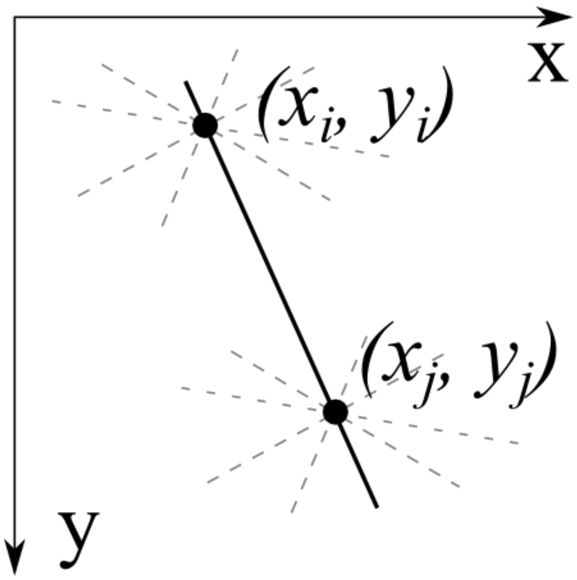
Few disadvantages

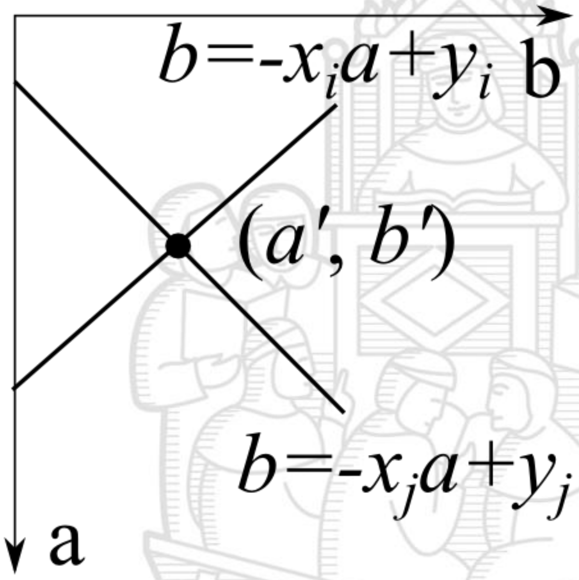
- high computational cost
- requires perfect shapes or produces several wrong detections

Line in parameters plane

$$y = a \cdot x + b \rightarrow b = -x_i \cdot a + y_i$$

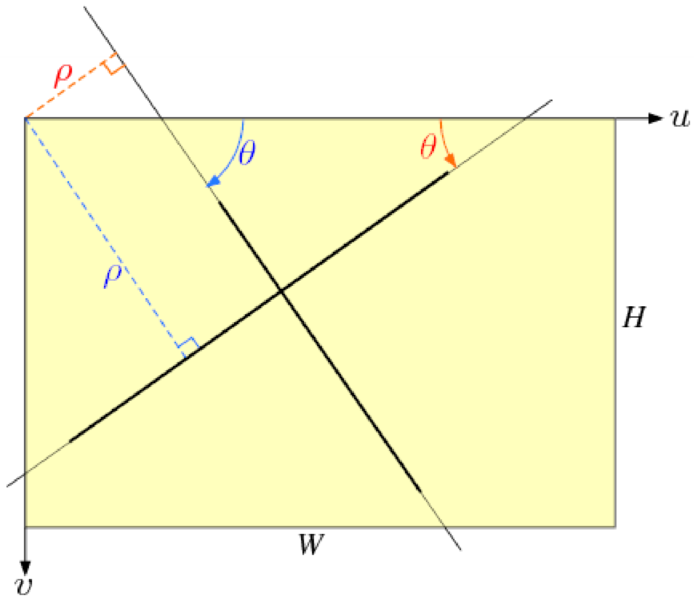
The linear function $b = -x_i \cdot a + y_i$ in the parameters plane represents all the linear functions that belong to the generic point (x_i , y_i).





Each point of the same function increase the accumulation *(a', b')*.

Polar parametrization



- The line shown can be described by the function

$$y=ax+b$$

  and identified by the couple of parameters:

  (a,b)=(-0.5,0.5)

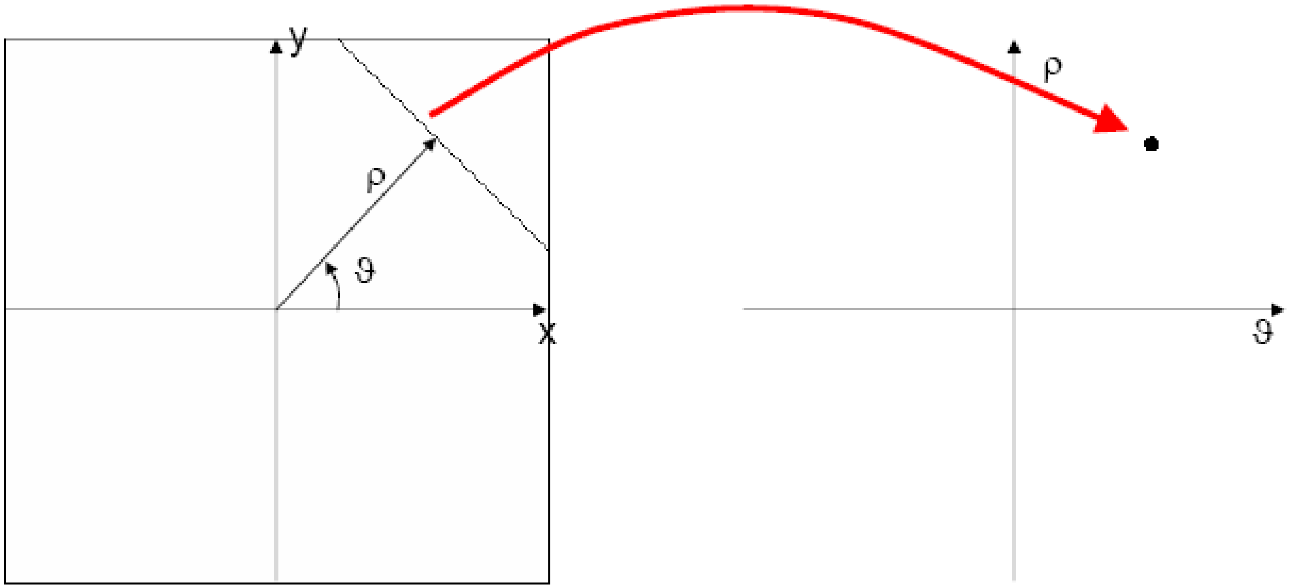
- or by the function

$$\rho = x \cos\vartheta + y \sin \vartheta$$

- and identified by the couple:

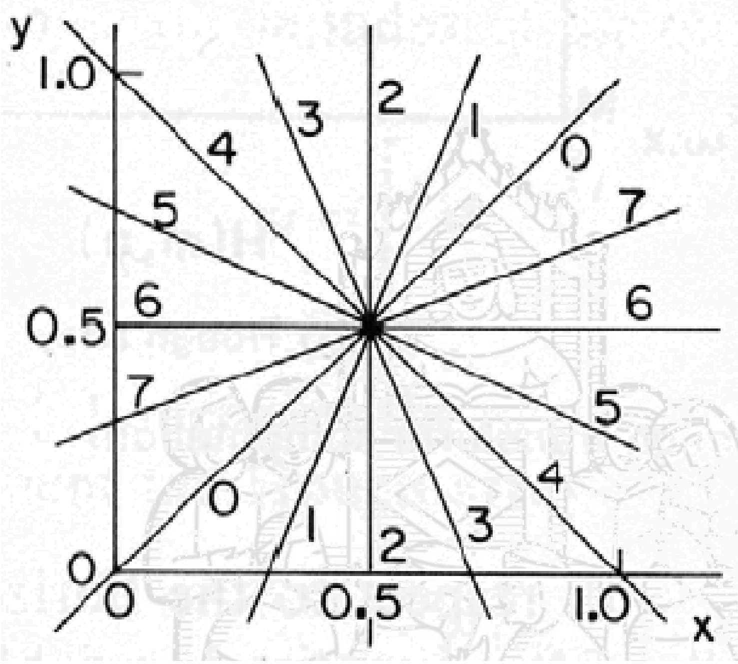
  (ρ,ϑ)=(0.447,1.107)

Transformation of the plane

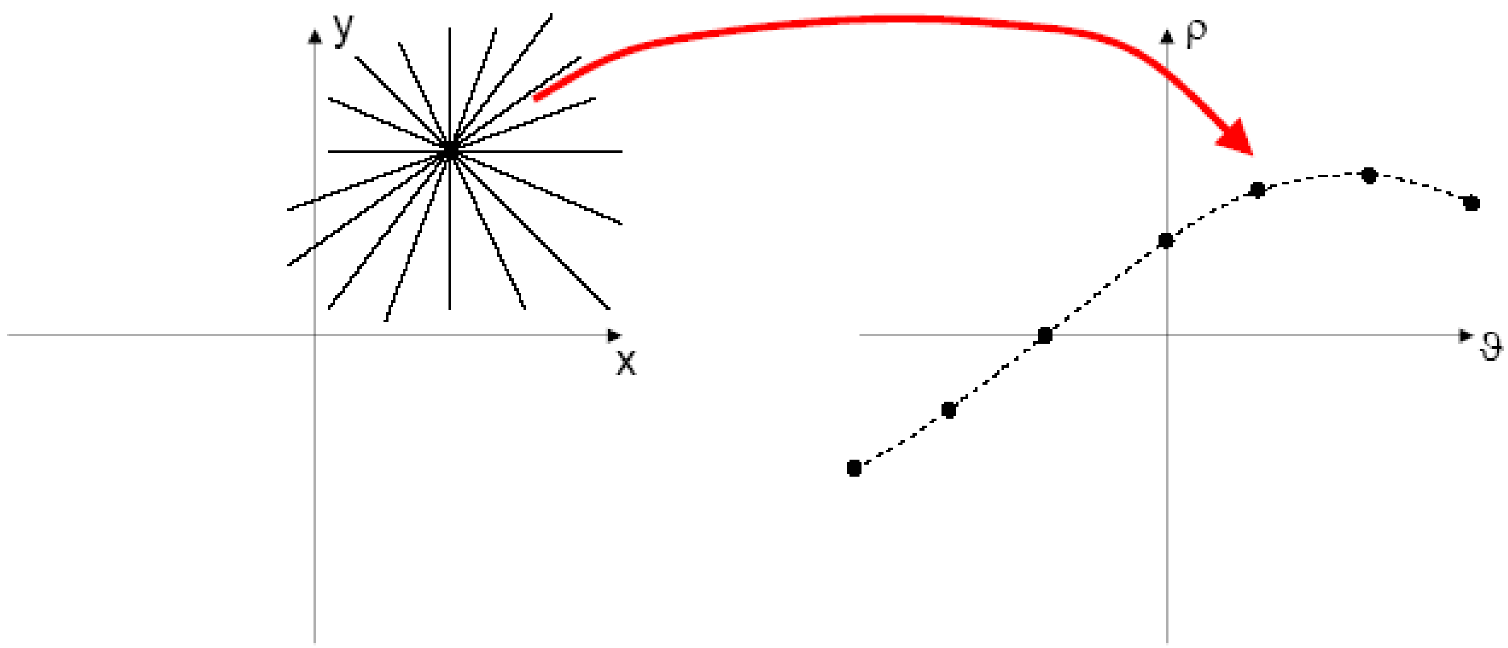


- In the image plane, one point is identified by the intersection of lines.

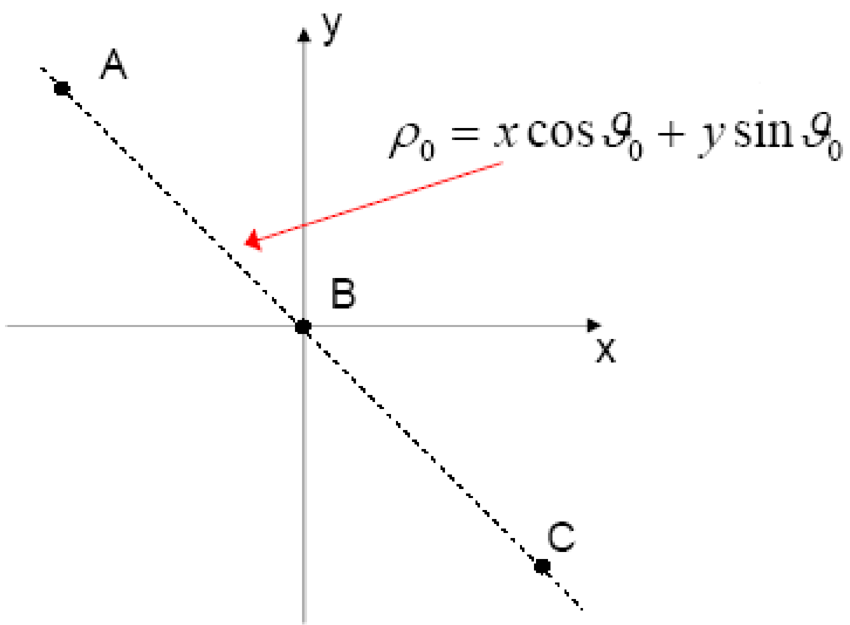
- Each point P corresponds, in the parameter plane, to the curve given by the image points of the lines passing through P

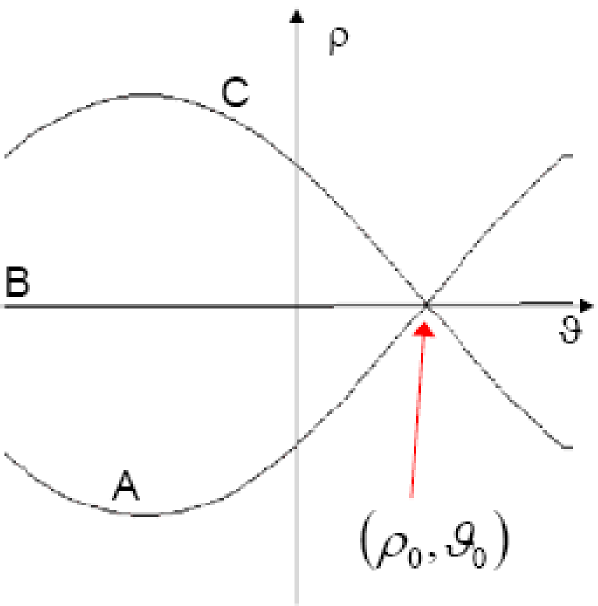


Transformation of a point



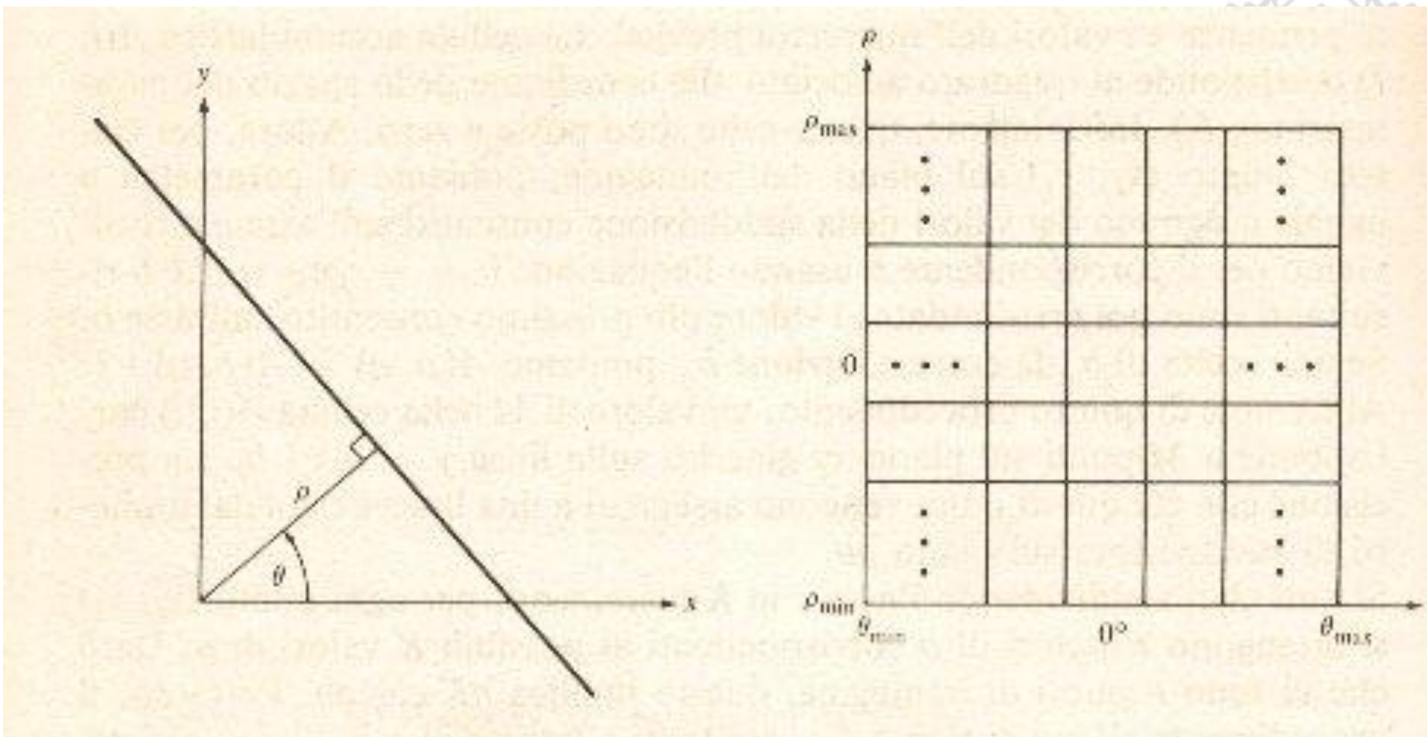
Detection of a lines on the transformed plane





Hough's method with the polar representation of the line

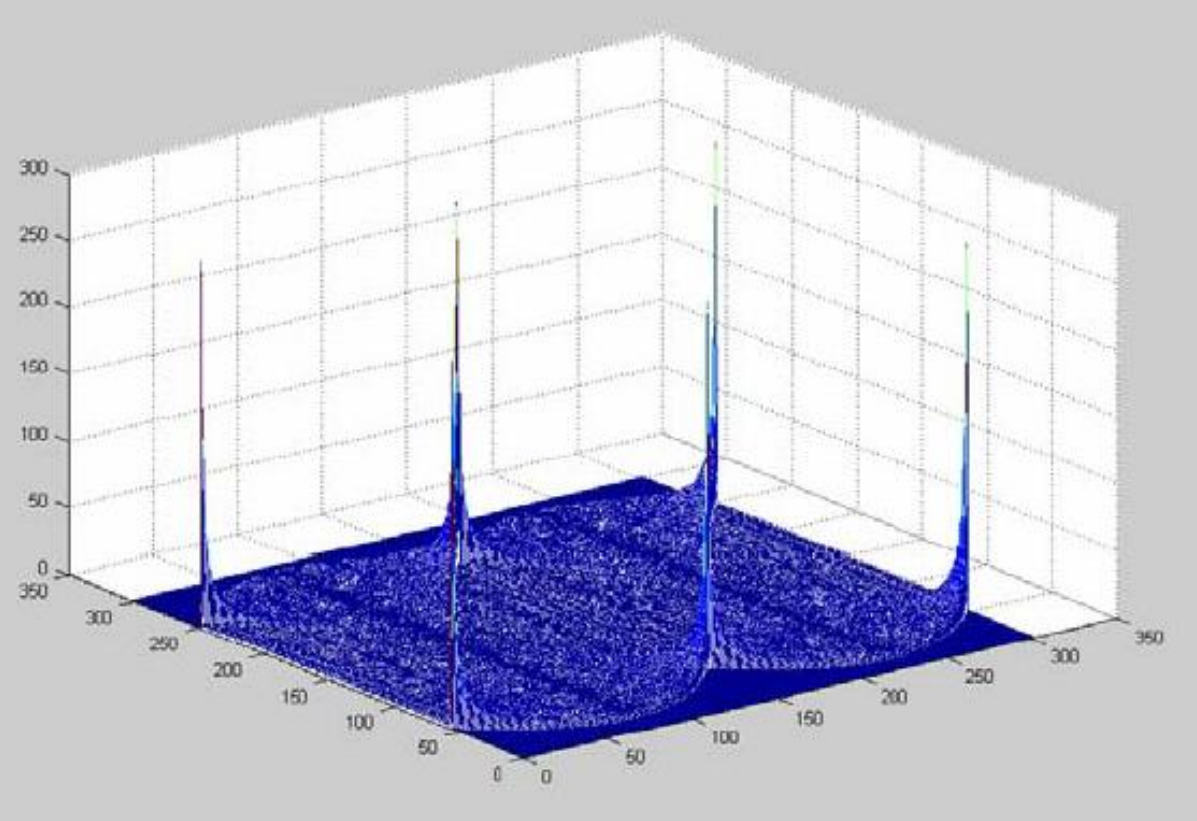
$\rho = x \cos \theta + y \sin \theta$



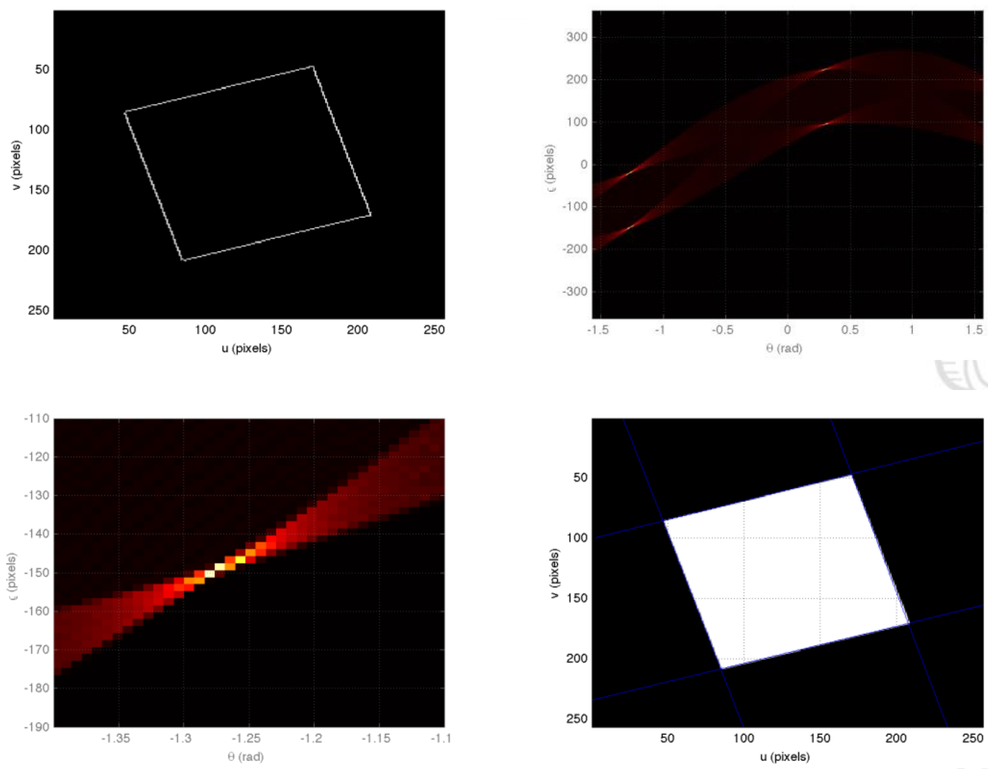
Hough's algorithm

1. Quantize the parameter space between appropriate minimum and maximum values
2. Create an accumulation array with size equal to the number of parameters, initialized to 0
3. For each edge in the image, increment of the element of the accumulation array corresponding to the parameter values of the curves on which the edge lays
4. The local maxima in the accumulation array represent the parameter values of the curves that better approximate the boundary

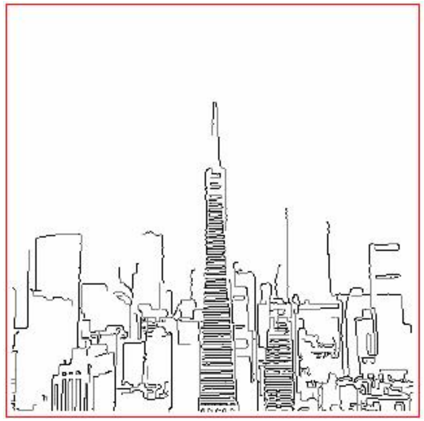
Example of accumulation matrix

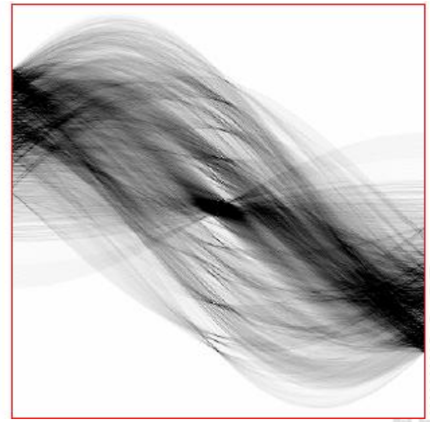


Example of the Hough's method to a rectangle

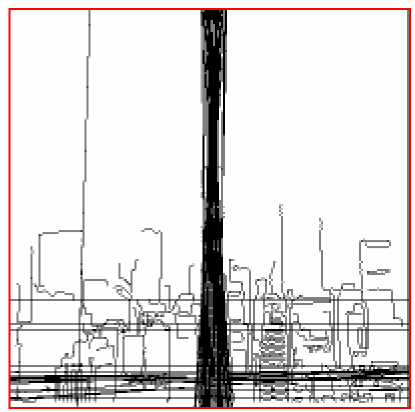


The problem of selecting the 'right' curves

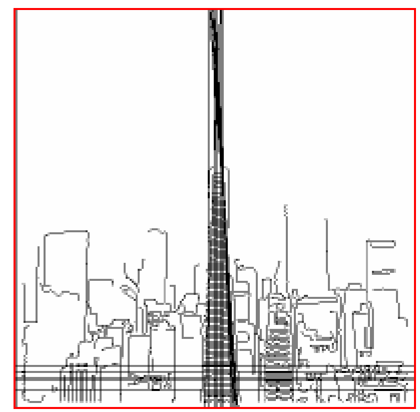




Soglia: 101

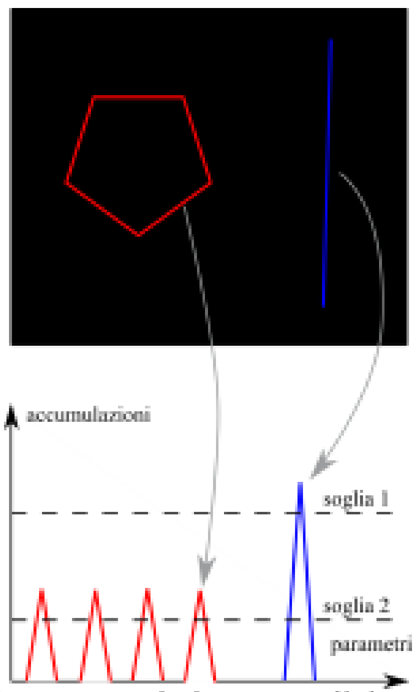


Soglia: 140



Soglia: 160

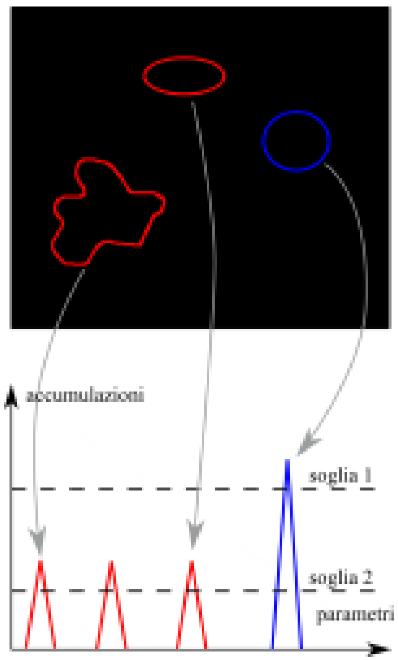
Square and lines



- depending on the length of the line, the accumulation has different values (lower or higher)
- a threshold should be selected to separate the key features from the other elements

In this case, it is possible to lower the threshold and each lines will be detected.

Ellipsis and circles



- same procedure can be used to detect circles
- in this case is difficult to identify the right threshold

Tracking

Matlab: Jtracking example

Point features

They are often called interest points, salient points, keypoints or corner points (even they not necessarily belong to corner of the image or the scene).

Since interest points are quite distinct from the other points in a local neighbourhood, they can be reliably find in different views of the same scene.

The earliest interest point's detector was developer by Moravec to aid tracking algorithm. It is based on the assumption that if an image region *W* should be detected from different views, it should be sufficiently different to all adjacent regions. Defining the similarity among a region at (*u*, *v*) and adjacent regions displaced in the 8 cardinal direction by (δ_u, δ_v) as:

$$s(u, v, \delta_u, \delta_v) = \sum_{i,j \in \boldsymbol{W}} (\boldsymbol{I}[u + \delta_u + i, v + \delta_v + j] - \boldsymbol{I}[u + i, v + j])^2$$

The maximum of the interest measure:

$$C_M(u, v) = \min_{(\delta_u, \delta_v)} s(u, v, \delta_u, \delta_v)$$

Define the interest point.

Point features

Maravec detector is non-isotropic, so strong responses generate also from point on a line (which is not desired).

To provide a rotationally invariant description of the neighbourhood and capturing the intensity structure, a symmetric 2x2 matrix is derived. This matrix is often referred to as structure tensor, auto-correlation matrix or second moment matrix. It form is the following:

$$\boldsymbol{A} = \begin{pmatrix} \boldsymbol{G}(\sigma_I) \otimes \boldsymbol{I}_u^2 & \boldsymbol{G}(\sigma_I) \otimes \boldsymbol{I}_u \boldsymbol{I}_v \\ \boldsymbol{G}(\sigma_I) \otimes \boldsymbol{I}_u \boldsymbol{I}_v & \boldsymbol{G}(\sigma_I) \otimes \boldsymbol{I}_v^2 \end{pmatrix}$$

Where $\boldsymbol{G}(\sigma_I)$ is a gaussian kernel, $\boldsymbol{I}_u$ and $\boldsymbol{I}_v$ are respectively the gradient of the image I in the u and v directions.

Common evaluations

Based on the auto-correlation matrix, it is possible to obtain different corner strength: the more common are reported hereafter.

Shi- Tomasi

$$C_{ST}(u,v) = \min(\lambda_1, \lambda_2)$$

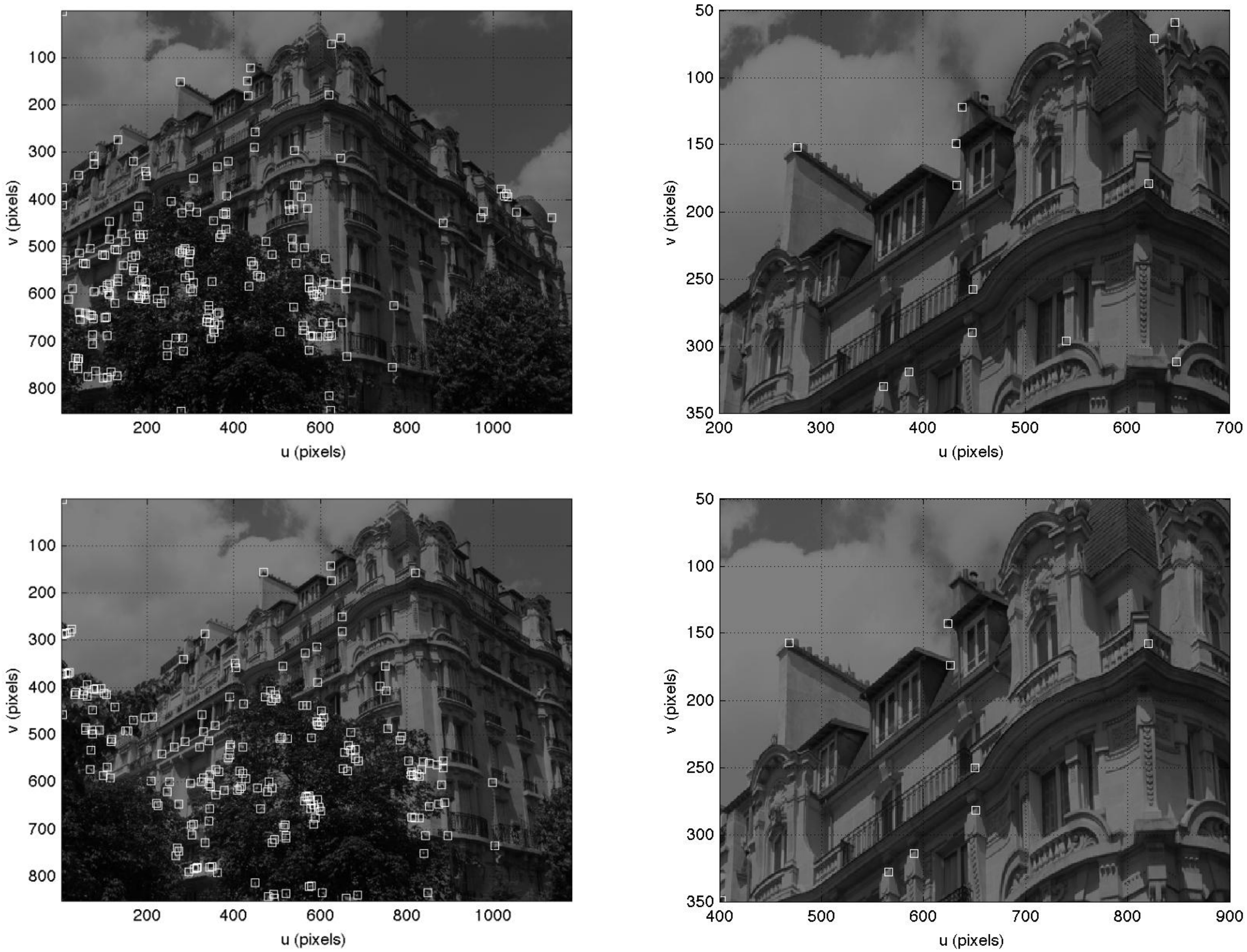
Harris

$$C_H(u,v) = det(A) - k\ tr(A)^2$$

Noble

$$C_n(u,v) = \frac{det(A)}{tr(A)}$$

Corners by Harris



Example use on tracking

Code sample >

test_tracker_lesson2015

Optic flow

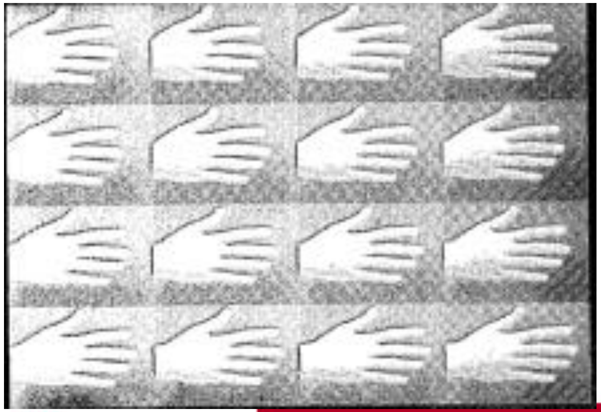
The motion is a significant part of our visual process, and it is used for several purposes:

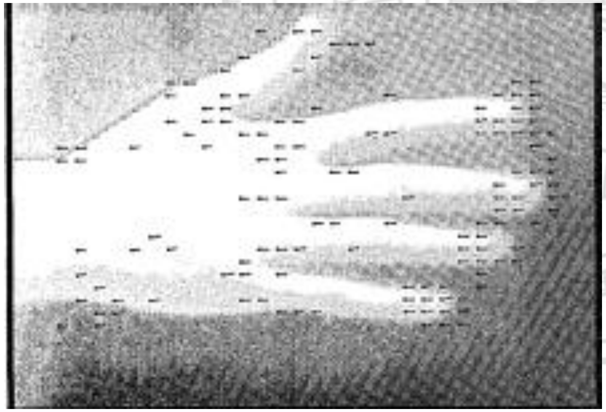
- to recognize tridimensional shapes
- to control the body by the oculomotor control
- to organize perception
- to recognize object
- to predict actions
- …

Optic flow

A surface or object moving in the space projects in the image plane a bidimensional path of speeds, *dx/dt* and *dy/dt* that is often referred to as *bidimensional motor field*.

The aim of the *optic flow* is to approximate the variation over time of the intensity levels of the image.





We consider that the intensity *I* of a pixel *(x, y)* at the instant *t*, moves to a neighbor pixel in the instant *t+dt*, thus:

$$I(x, y, t) = I(x + dx, y + dy, t + dt) \tag{7}$$

Expanding in Taylor serie:

$$I(x + dx, y + dy, t + dt) = I(x, y, t) + \frac{\partial I}{\partial x}dx + \frac{\partial I}{\partial y}dy + \frac{\partial I}{\partial t}dt$$

and taking as a reference (7) :

$$\frac{\partial I}{\partial x}dx + \frac{\partial I}{\partial y}dy + \frac{\partial I}{\partial t}dt = 0$$

This is the *gradient constraint equation* :

$$\frac{\partial I}{\partial x}\dot{x} + \frac{\partial I}{\partial y}\dot{y} + \frac{\partial I}{\partial t} = 0$$

That can be rewritten as:

$$\nabla I \cdot V^T + \frac{\partial I}{\partial t} = 0$$

with $V=(\dot{x}, \dot{y})$

Since the gradient constraint equation has two variables, it cannot be solved directly. This is called the *aperture problem*.

Lucas-Kanade hypothesis

To solve the aperture problem, they hypothesize:

- the motion of the intensity of pixel among two subsequent frames is small
- the motion in a small local neighbor of the pixel is constant

This is equivalent to say that the optical flow is constant for each pixel centered in *p*, thus:

$$
\begin{aligned}
I_x(q_1)\dot{x} + I_y(q_1)\dot{y} &= -I_t(q_1) \\
I_x(q_1)\dot{x} + I_y(q_1)\dot{y} &= -I_t(q_1) \\
&\vdots \\
I_x(q_1)\dot{x} + I_y(q_1)\dot{y} &= -I_t(q_1)
\end{aligned}
$$

Where $q_1, \ldots, q_n$ are pixel of the window centered in *p* and $I_{x,y,t}$ are the derivative with respect to *x,y,t*

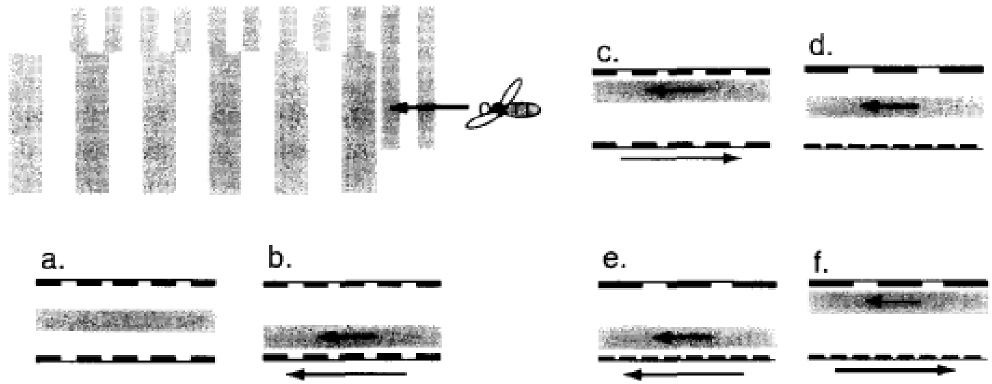
By writing the equations in vectorial form ***A·v* = *b*** where:

$$A = \begin{bmatrix} I_x(q_1) & I_y(q_1) \\ I_x(q_2) & I_y(q_2) \\ \vdots & \vdots \\ I_x(q_n) & I_y(q_n) \end{bmatrix}, \quad v = \begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix}, \quad b = \begin{bmatrix} -I_t(q_1) \\ -I_t(q_2) \\ \vdots \\ -I_t(q_n) \end{bmatrix}$$

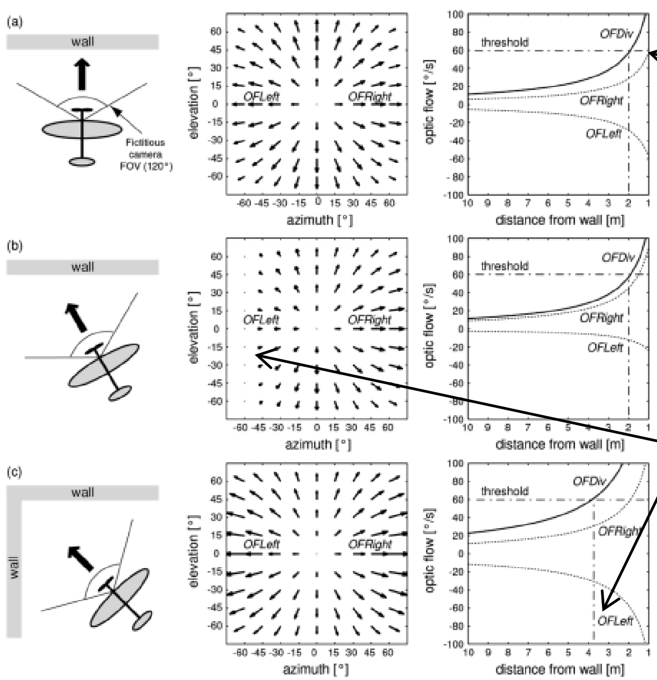
This system can be solved with the least square method:

$$v = (A^T A)^{-1} A^T b$$

Bee-inspired navigation control



The optic flow provides information for wall following and landing



The difference is inversely proportional to the distance from an obstacle

The absolute difference defines the turning behaviour