Classification by Hand-Crafted Features

"Image Classification: Modern Approaches"

Giacomo Boracchi, Alessandro Giusti
DEIB, Politecnico di Milano

February, 14th, 2018

giacomo.boracchi@polimi.it
home.deib.polimi.it/boracchi/

Image Classification By Feature Extraction

The Feature Extraction Perspective

Images can not be directly fed to a classifier

We need some intermediate step to:

- Extract meaningful information (to our understanding)
- Reduce data-dimension

We need to extract features:

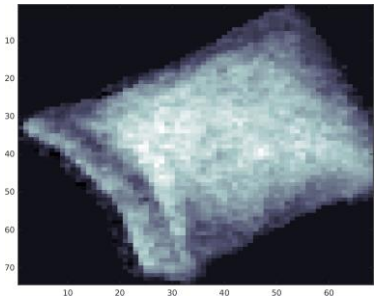
- The better our features, the better the classifier

These are the options we will consider:

- **Hand Crafted Features:**
- **Computer Vision Features:**
- **Learned Features:**

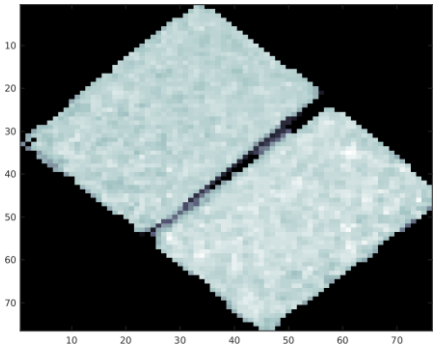
The Feature Extraction Perspective

Input image



$I_1 \in \mathbb{R}^{r_1 \times c_1}$

Input image



$I_1 \in \mathbb{R}^{r_2 \times c_2}$

Feature Extraction Algorithms

$\mathbf{x} \in \mathbb{R}^d$

$\mathbf{x} \in \mathbb{R}^d$

$(d \ll r \times c)$

Classifier

"parcel"

$l \in \Lambda$

"double"

$l \in \Lambda$

Hand Crafted Featues

Hand Crafted Features: engineers (you)

- know what's meaningful in an image (e.g. a specific colour/shape, the area, the size)
- can implement algorithms to map these information in numbers to be fed to a classifier

Hand Crafted Featues

Advantages:

- Exploit a priori / expert information
- Features are interpretable (you might understand why they are not working)
- You can adjust features to improve your performance
- Limited amount of training data needed
- You can interact with the system to give more relevance to some features in some cases, disregarding the training data

Hand Crafted Featues

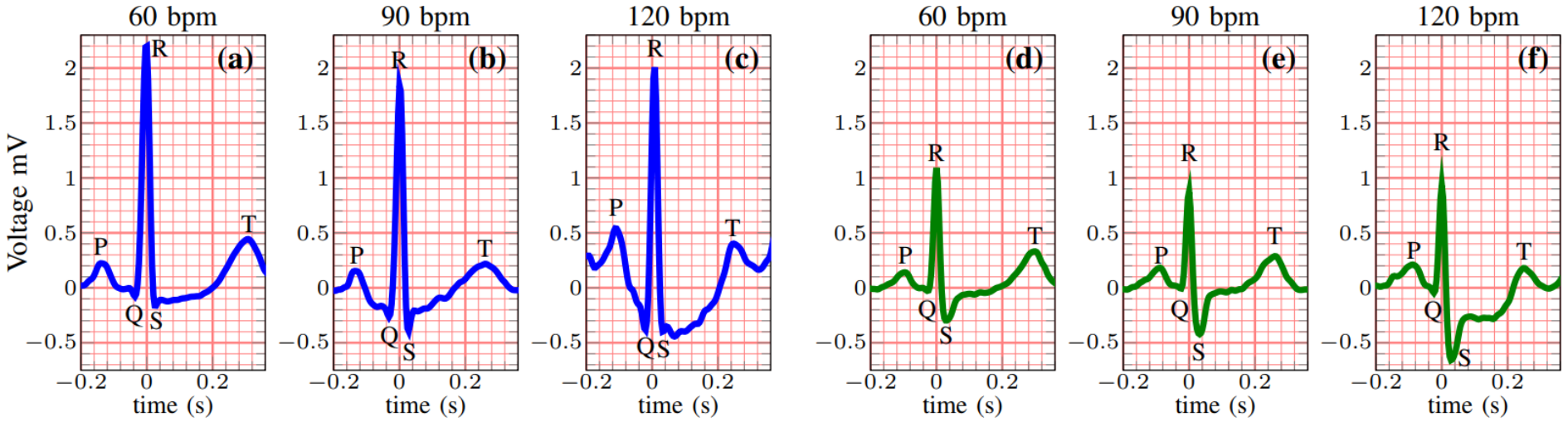
Cons:

- Requires a lot of effort
- Not a viable option in many visual recognition tasks (e.g. on natural images) which are easily performed by humans
- Risk of overfitting the training set used in the design
- Not very general and "portable"

Example: ECG Classification

ECG monitoring raises relevant machine learning challenges:

- Anomaly detection
- Heart-beat classification (arrhythmias)
- Person identification



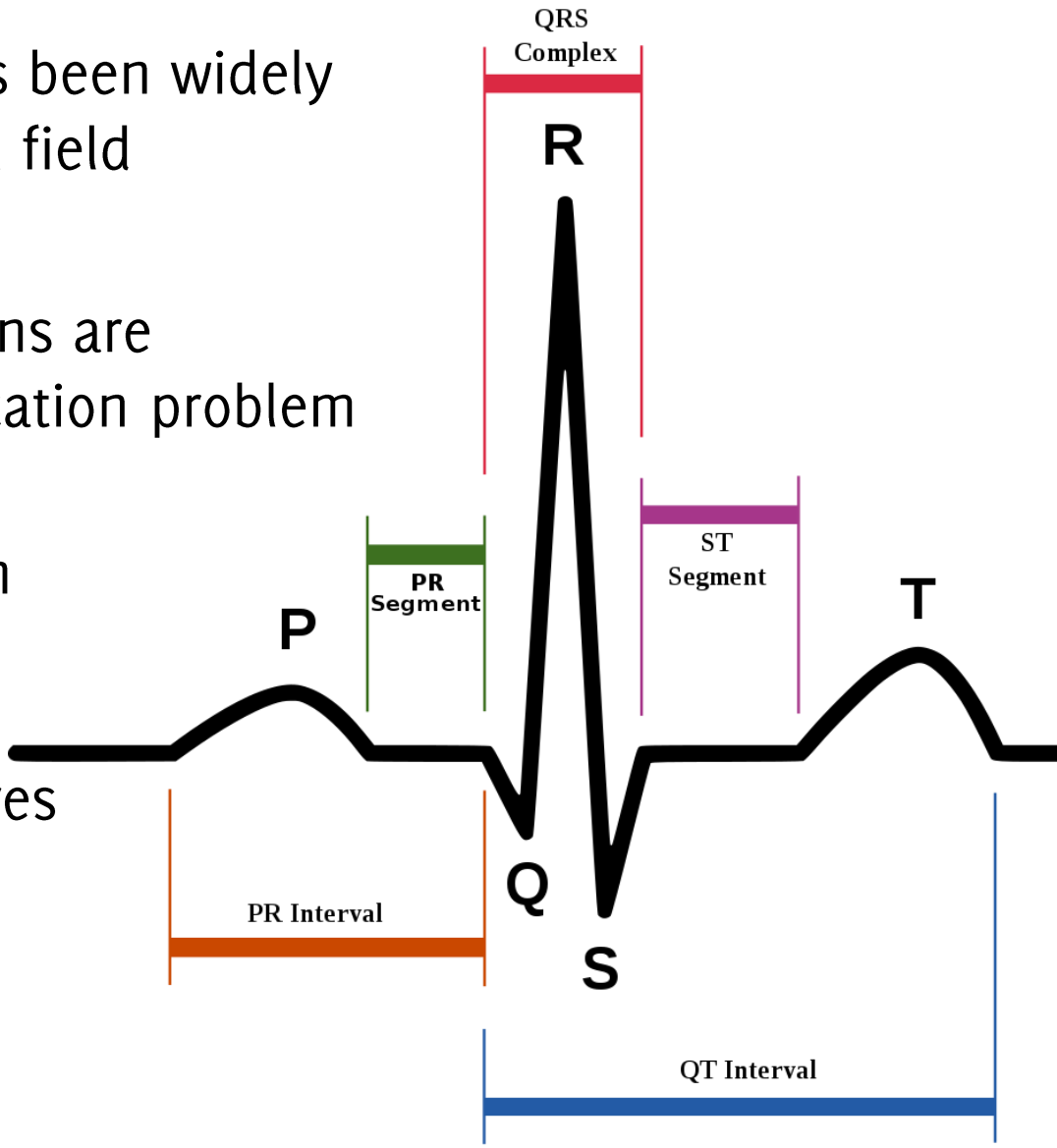
Carrera, Rossi, Fragneto,Boracchi "Domain Adaptation for Online ECG Monitoring " ICDM 2017

Example: parcel classification

Heartbeats morphology has been widely investigated in the medical field

Doctors know which patterns are meaningful for the classification problem

There are some well known landmarks in an heartbeat which can be used to design expert-driven features



Example: ECG Classification

A data-driven approach ignores all the medical background and simply tries to infer doctors decisions from TR

E.g. you might learn an unsupervised model of the heartbeat and compare HB against this to detect anomalies

Carrera, Rossi, Fragneto,Boracchi “Domain Adaptation for Online ECG Monitoring “ ICDM 2017

Your First IC Challenge

Parcel classification (Double vs Non-Double, or Double vs Parcel vs Envelope)

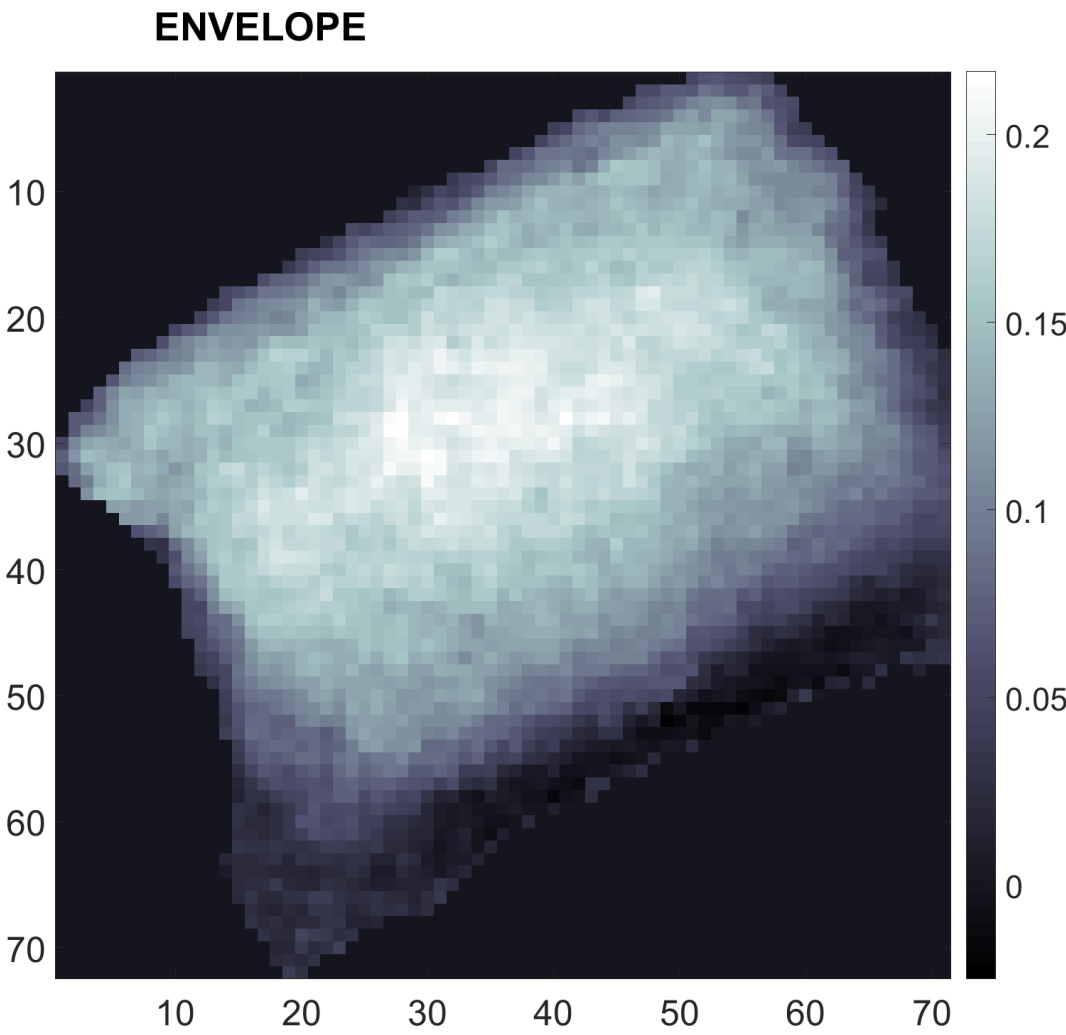
Download the dataset

http://home.deib.polimi.it/boracchi/teaching/ImageClassification.htm

The Application Scenario: Parcel Classification

Images acquired from a RGB-D sensor:

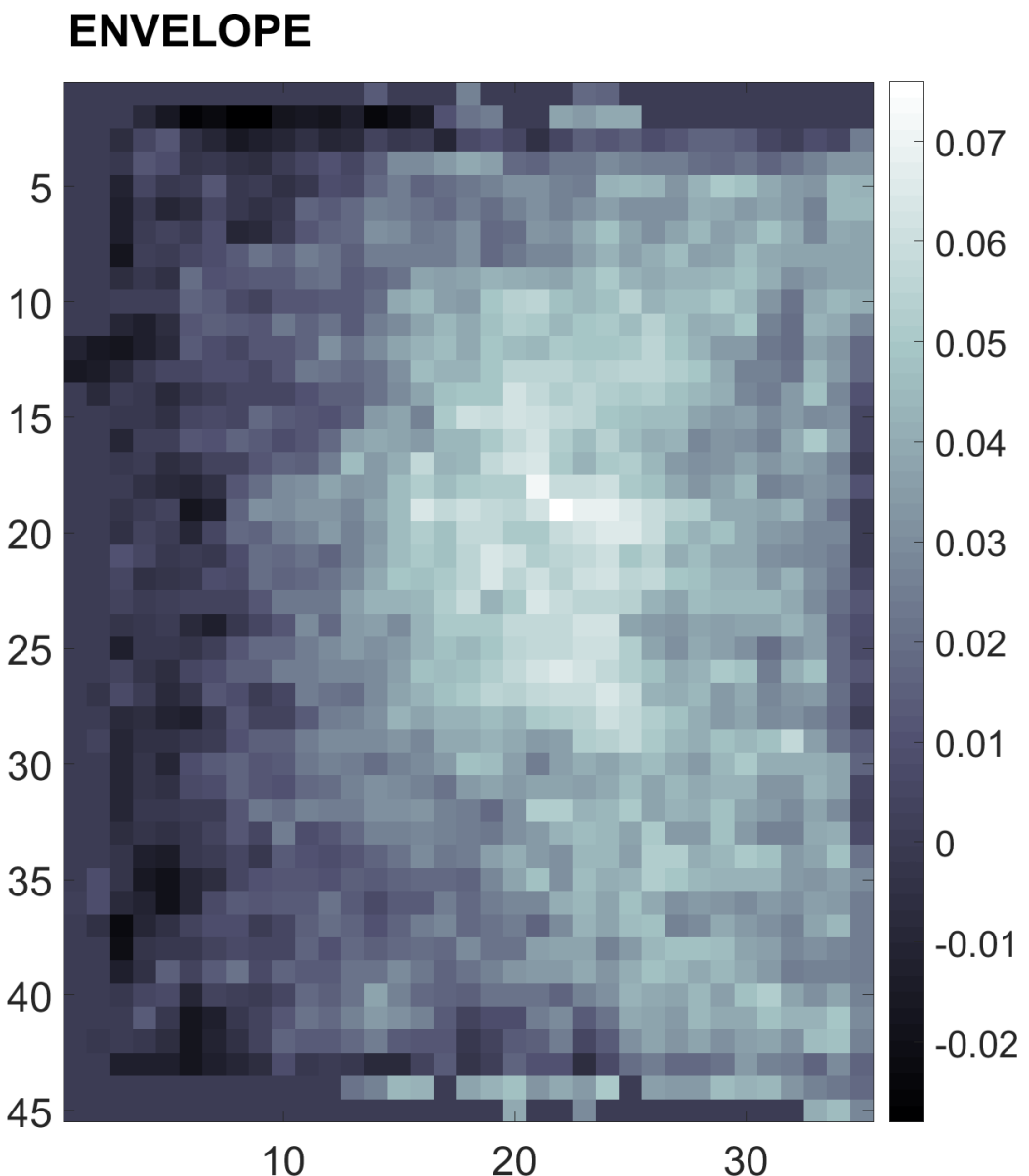
- No color information provided
- A few pixels report depth measures
- Images of 3 classes
  - ENVELOPE
  - PARCEL
  - DOUBLE



The Application Scenario: Parcel Classification

Images acquired from a RGB-D sensor:

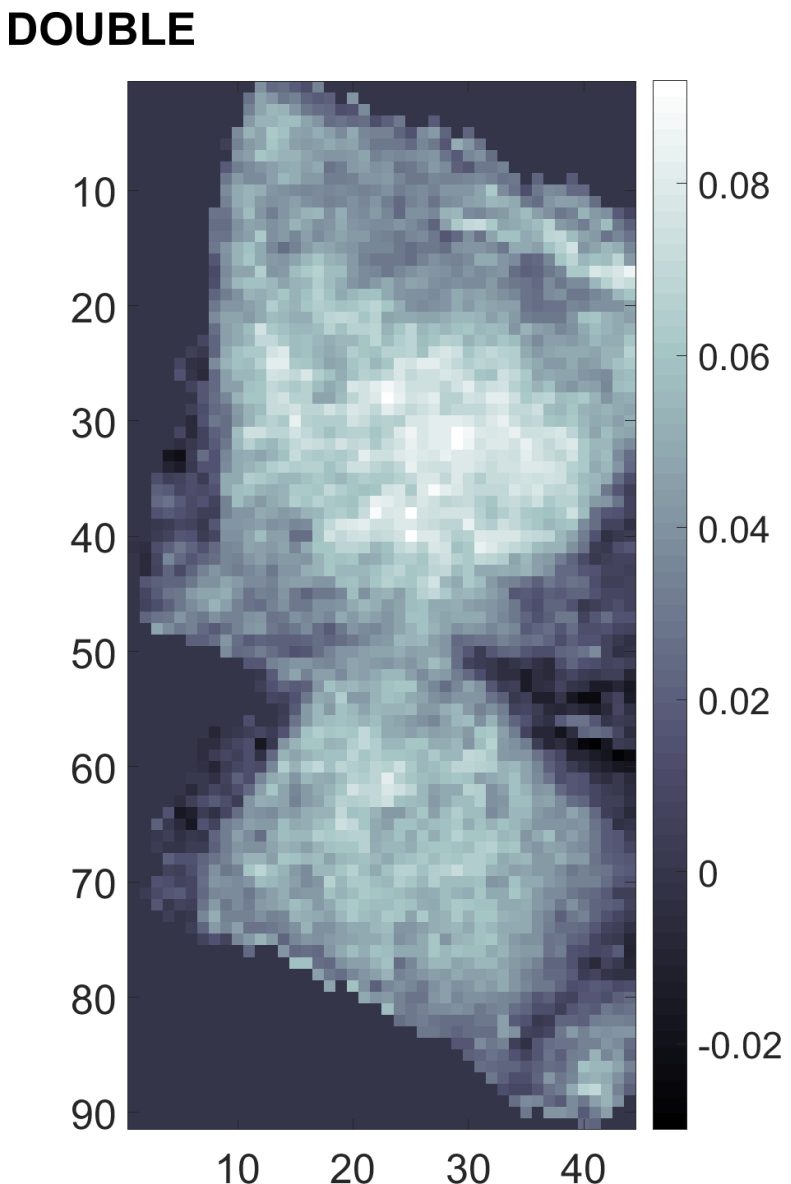
- No color information provided
- A few pixels report depth measures
- Images of 3 classes
  - ENVELOPE
  - PARCEL
  - DOUBLE



The Application Scenario: Parcel Classification

Images acquired from a RGB-D sensor:

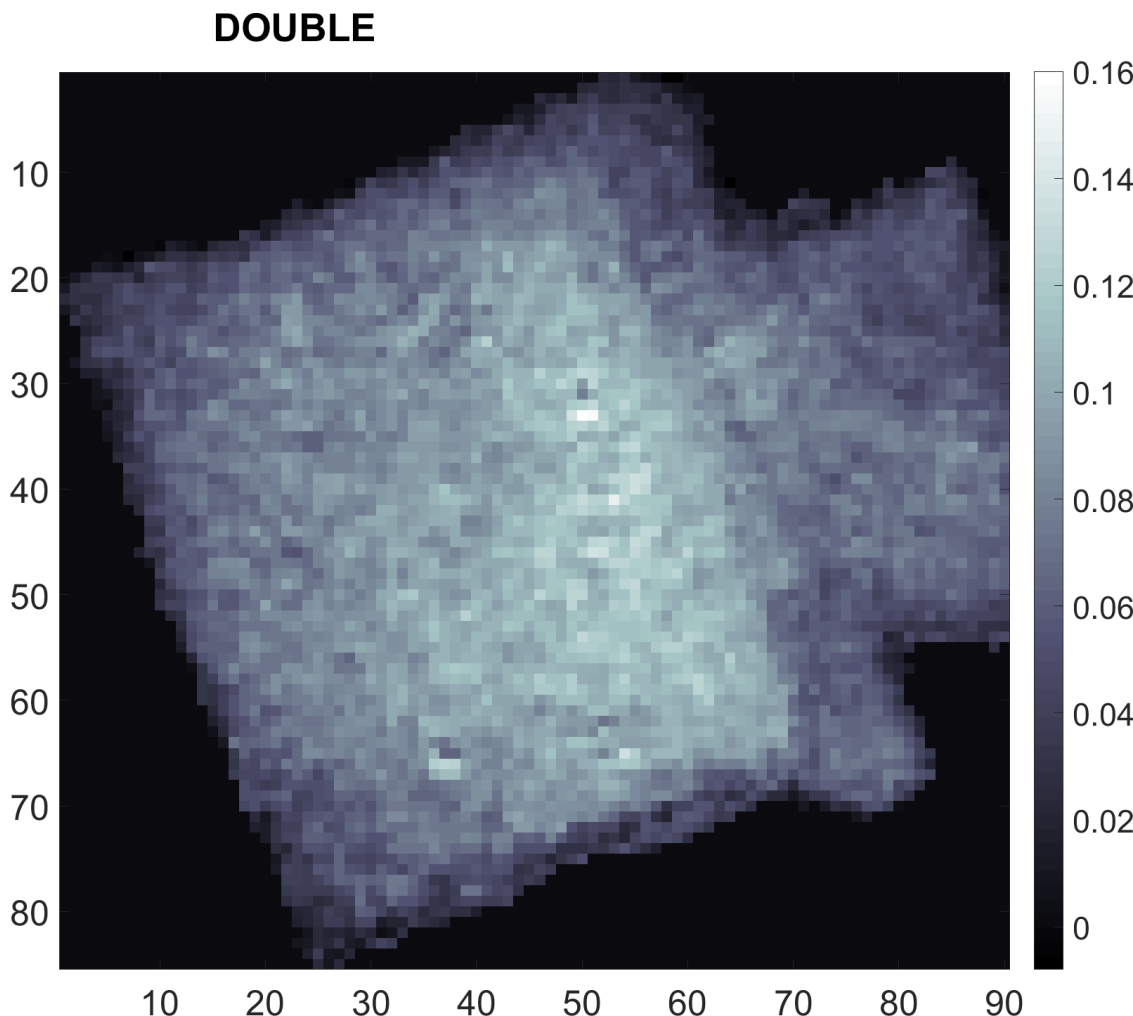
- No color information provided
- A few pixels report depth measures
- Images of 3 classes
  - ENVELOPE
  - PARCEL
  - DOUBLE



The Application Scenario: Parcel Classification

Images acquired from a RGB-D sensor:

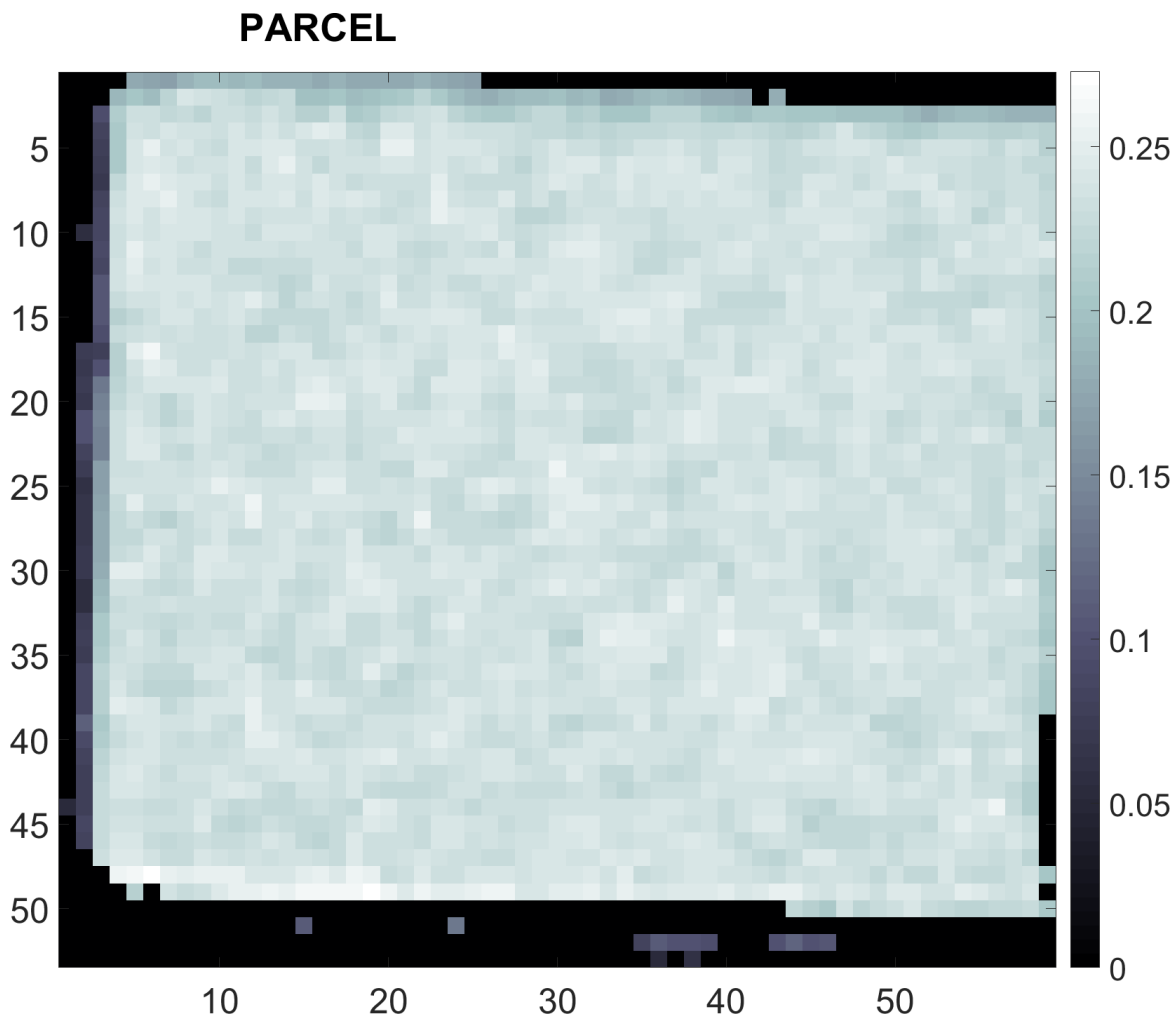
- No color information provided
- A few pixels report depth measures
- Images of 3 classes
  - ENVELOPE
  - PARCEL
  - DOUBLE



The Application Scenario: Parcel Classification

Images acquired from a RGB-D sensor:

- No color information provided
- A few pixels report depth measures
- Images of 3 classes
  - ENVELOPE
  - PARCEL
  - DOUBLE



TODO

Which features do you think are meaningful for separating these images into classes?

- Implement your feature extraction
- train your favourite classifier on your features
- assess the classification performance in a k-fold cross-validation procedure

Dataset available for download on the course website

Outline

Outline

Nonlinear and local image transformation

Nonlinear Local Transformations:

Most important image processing operations for classification problems

Local (Spatial) Transformation

In general, these can be written as

$$G(r,c) = T_U[I(r,c)]$$

Where

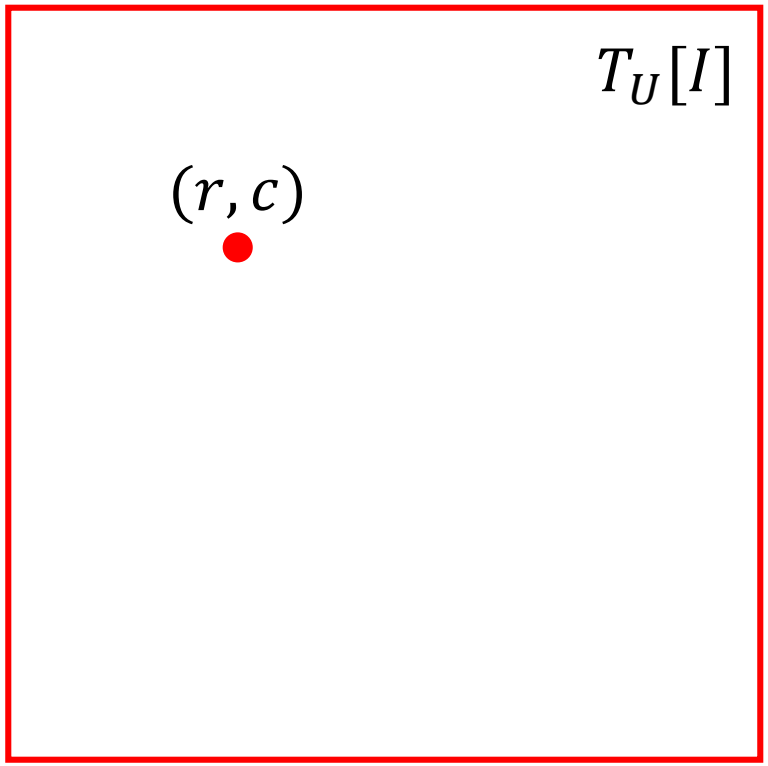
- I is the input image to be transformed
- G is the output
- T_U: $\mathbb{R}^3 \rightarrow \mathbb{R}^3$ or T_U: $\mathbb{R}^3 \rightarrow \mathbb{R}$ is a function
- U is a neighbourhood, identifies a region of the image that will concur in the output definition

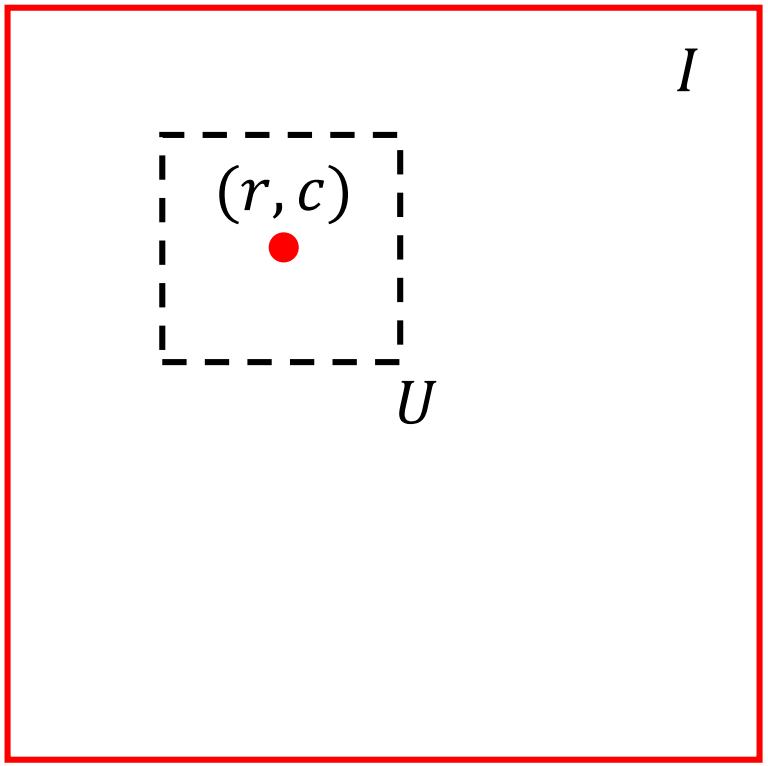
T operates on I "around" U

The output at pixel (r,c) i.e., $T_U[I(r,c)]$ is defined by all $\{I(x,y), \quad (x-r, y-c) \in U\}$

Local (Spatial) Filters

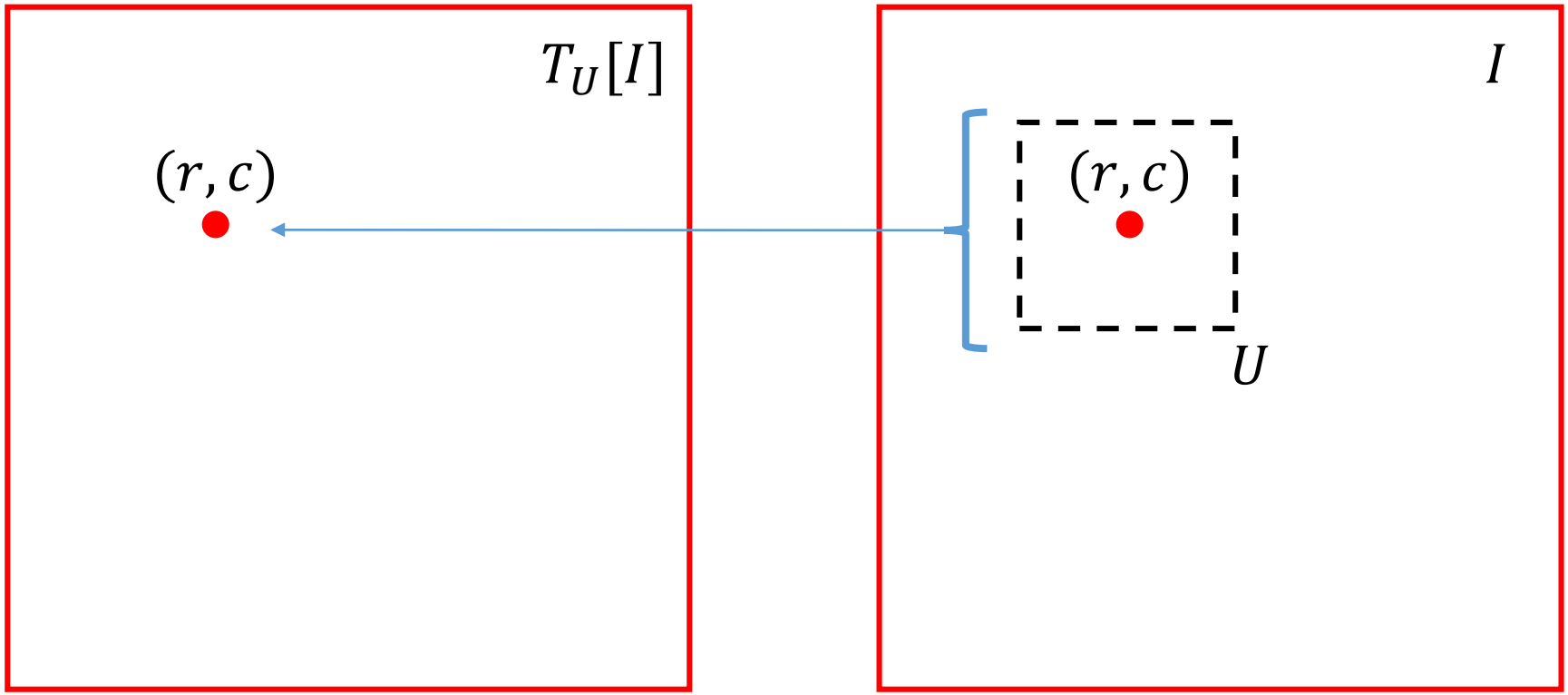
The dashed square represents $\{I(x,y),\ \ (x-r,y-c) \in U\}$





Local (Spatial) Filters

The dashed square represents $\{I(x,y),\ \ (x-r,y-c) \in U\}$



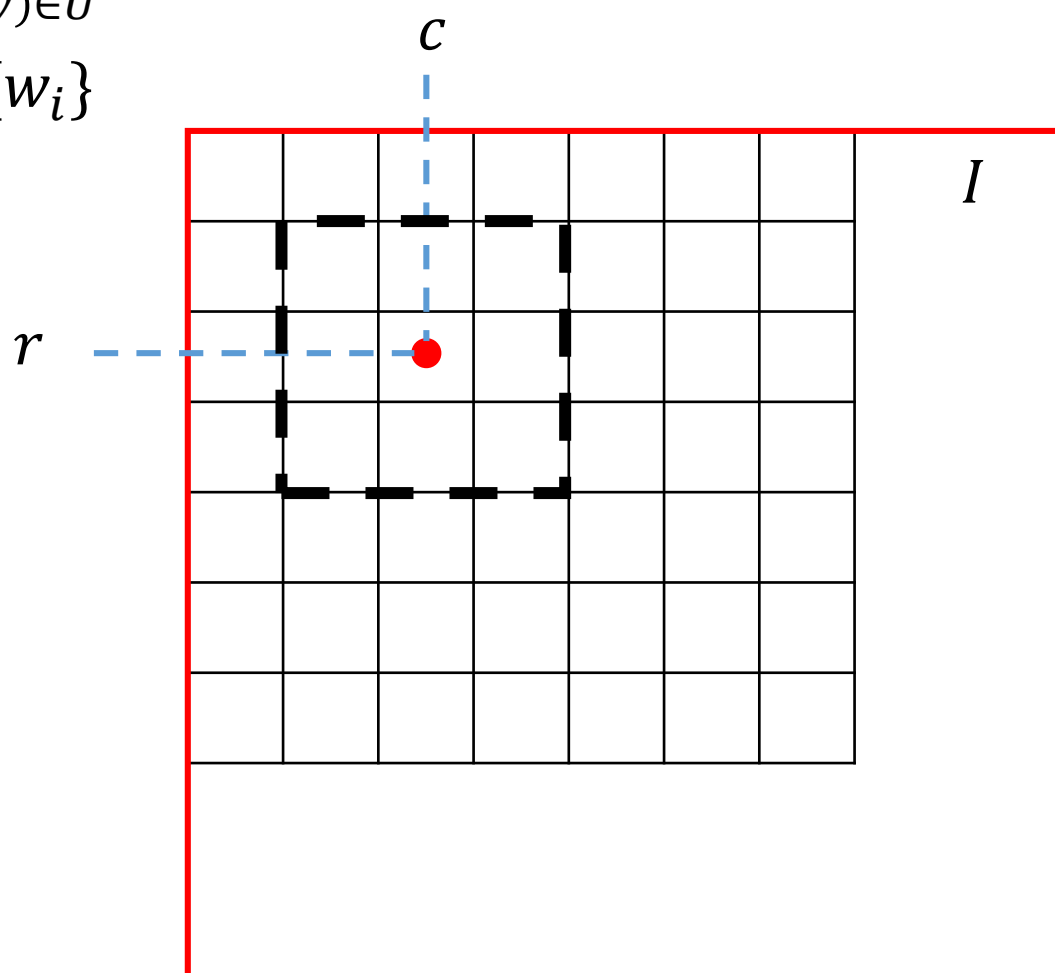
- The location of the output does not change
- The operation is repeated for each pixel
- T can be either linear or nonlinear

Local Linear Filters

Linear Transformation: Linearity implies that

$$T\big(I(r,c)\big) = \sum_{(x,y)\in U} w_i * I(r+x, c+y)$$

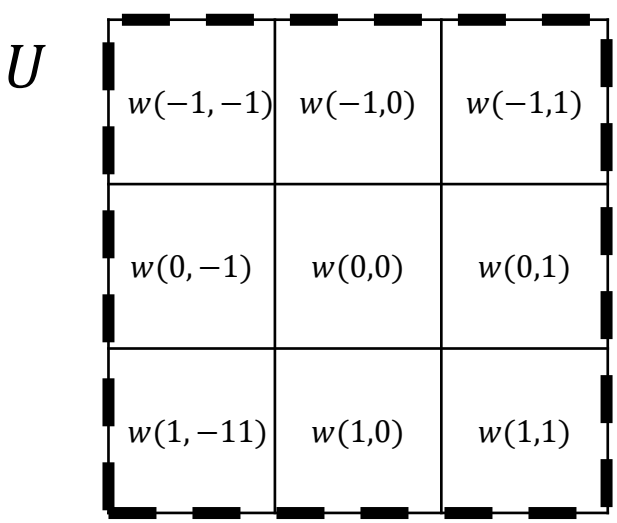
Considering some weights $\{w_i\}$



Local Linear Filters

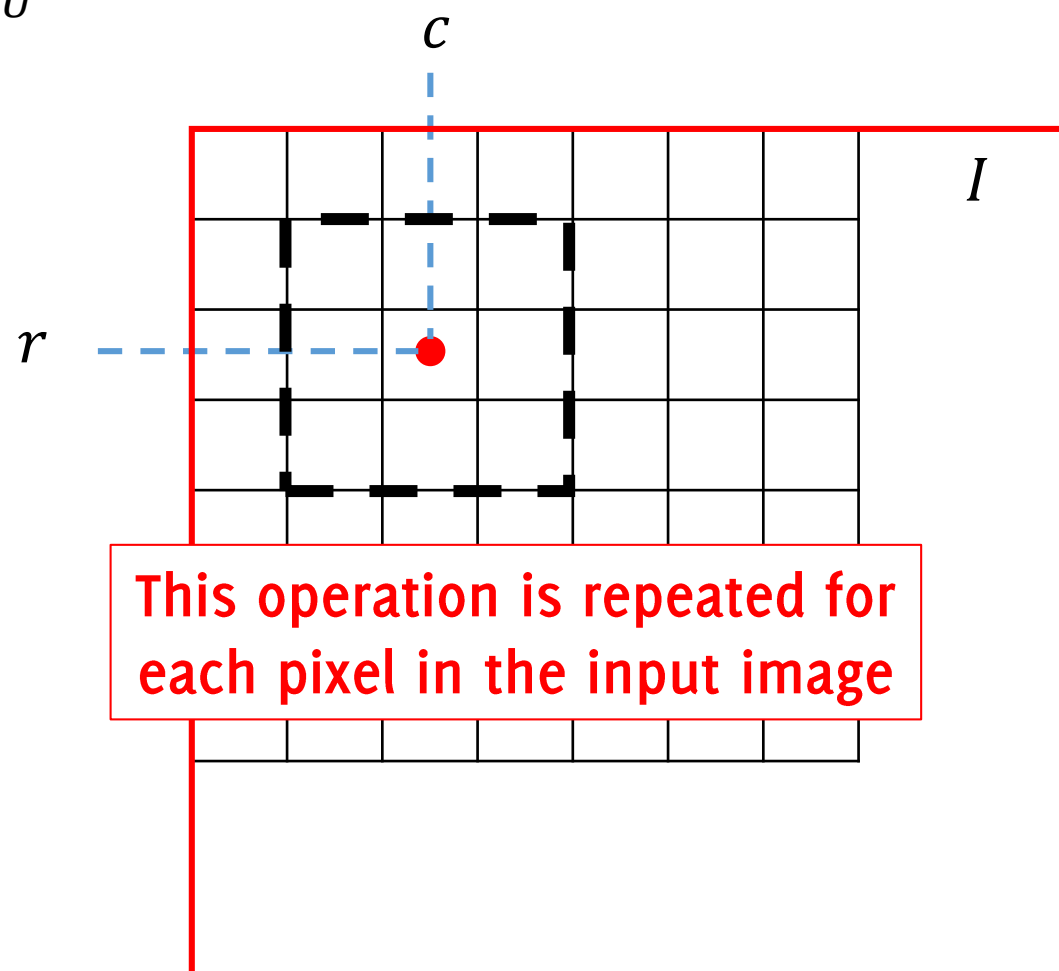
Linear Transformation: Linearity implies that

$$T\big(I(r,c)\big) = \sum_{(x,y)\in U} w(x,y) * I(r+x, c+y)$$



We can consider weights as an image, or **a filter** h

The filter h entirely defines this operation



This operation is repeated for each pixel in the input image

Non Linear Filters

For Linear Filters **it does not hold:**

$$H[\lambda f(t) + \mu g(t)] = \lambda H[f](t) + \mu H[g](t)$$

does not hold, at least for some value of λ, μ, f, g

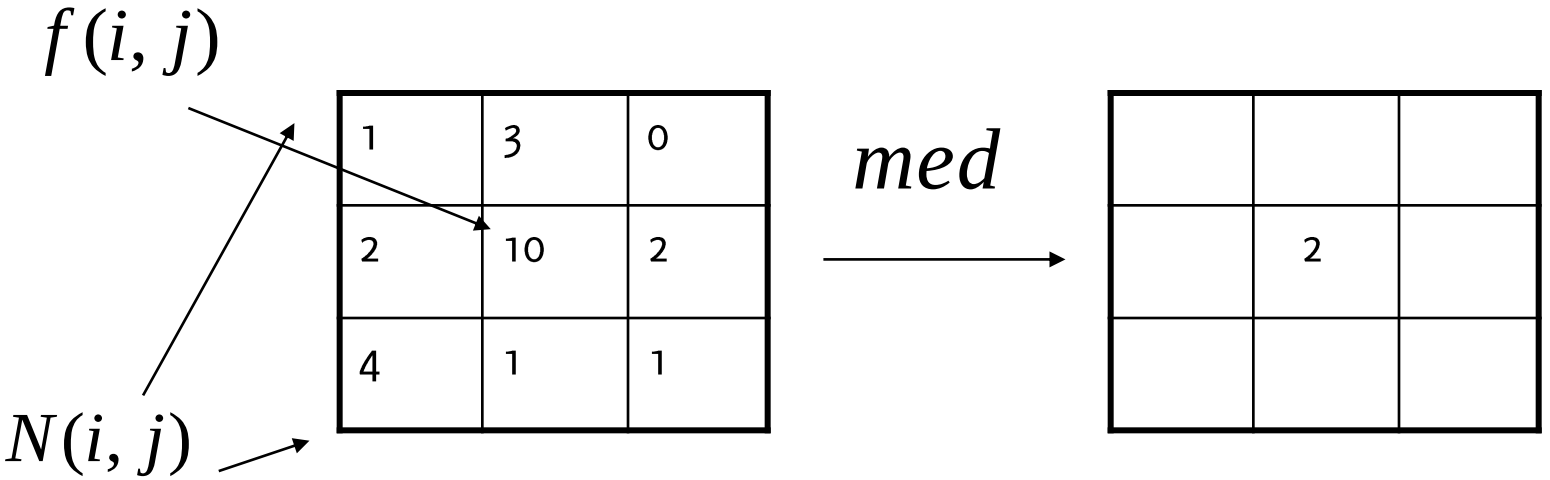
Examples of nonlinear filter are

- Median Filter (Weighted Median)
- Ordered Statistics based Filters
- Threshold, Shrinkage

There are many others, such as data adaptive filtering procedures (e.g LPA-ICI)

Blockwise Median

Block-wise median: replaces each pixel with the median of its neighborhood



$$m = median(1,3,0,2,10,2,4,1,1) = 2$$

In Python: skimage.filters.median

Denoisng using local smoothing 3x3

Denoisng using local smoothing 3x3

Denoisng with median 3x3

Salt and Pepper (Impulsive) noise

Denoisng with median 3x3

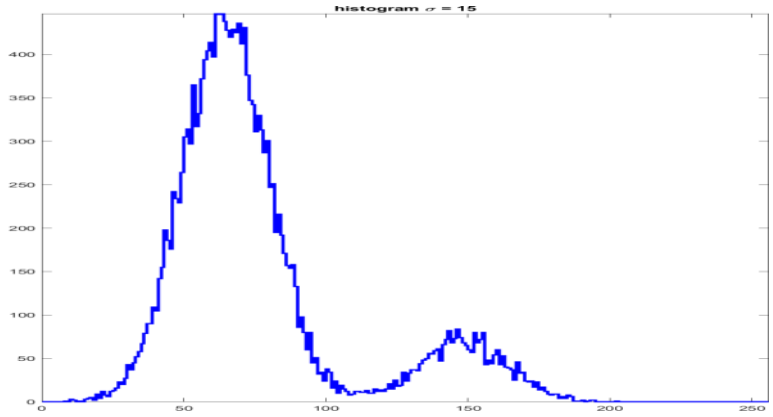
Salt and Pepper (Impulsive) noise

Going to Binary Images

Plenty morphological information can be extracted from binary images

Threshold definition by Otsu Method

Assume your (grayscale) image has a bi-modal histogram



Goal: Find a threshold T to suitably binarieze your image.

This threshold has to be computed efficiently since this operation might be repeated in different image regions

TODO DEFINIRE GLI ISTOGRAMMI, SONO IMPORTANTI ANCHE PER LE FEATURES

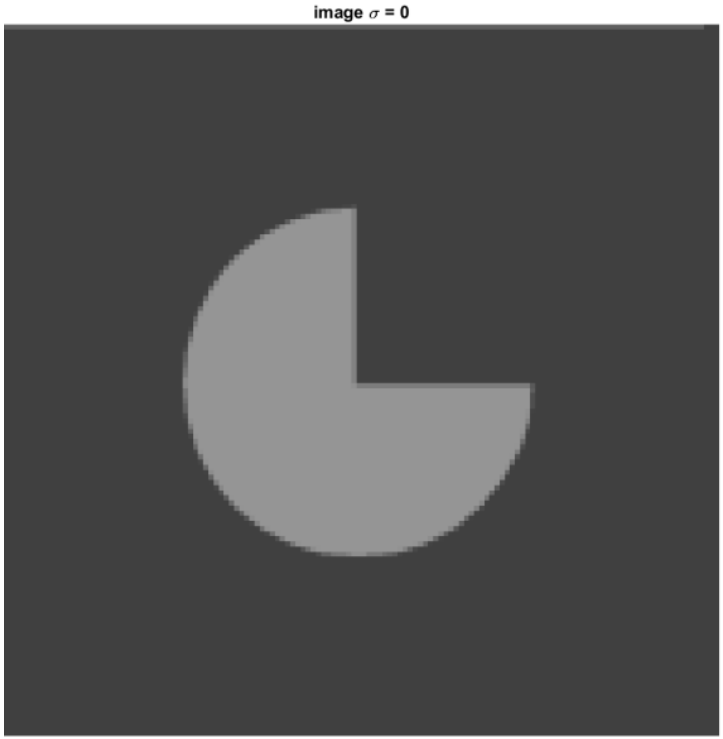
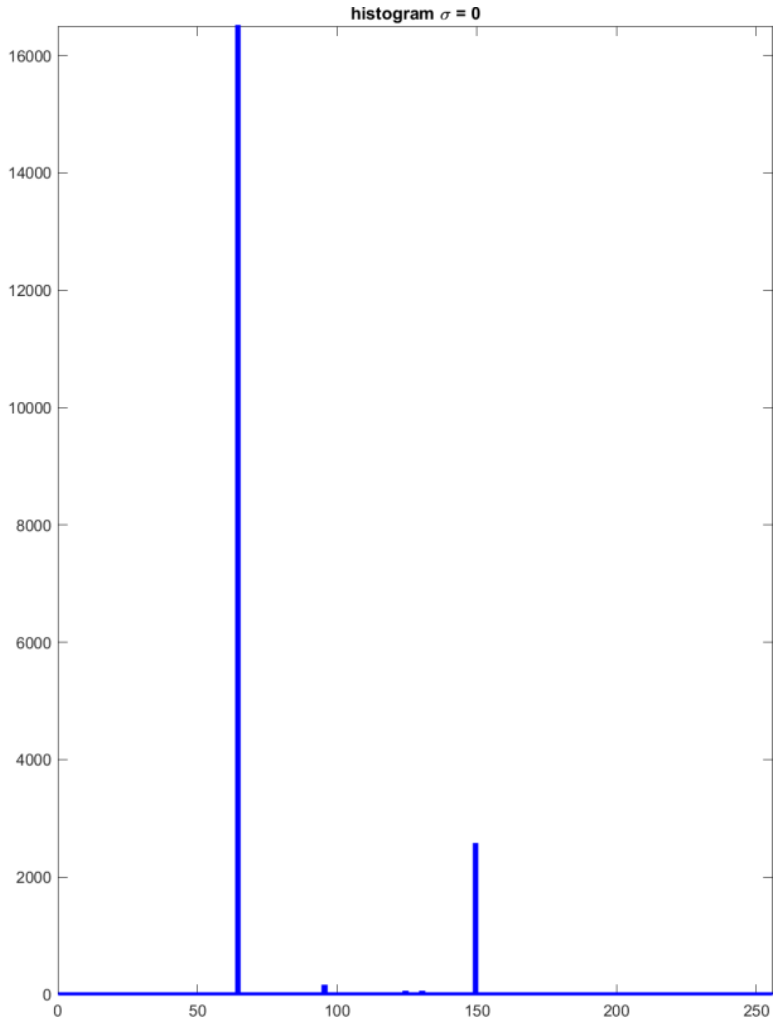


image σ = 0


histogram σ = 0
16000
14000
12000
10000
8000
6000
4000
2000
0
0
50
100
150
200
250

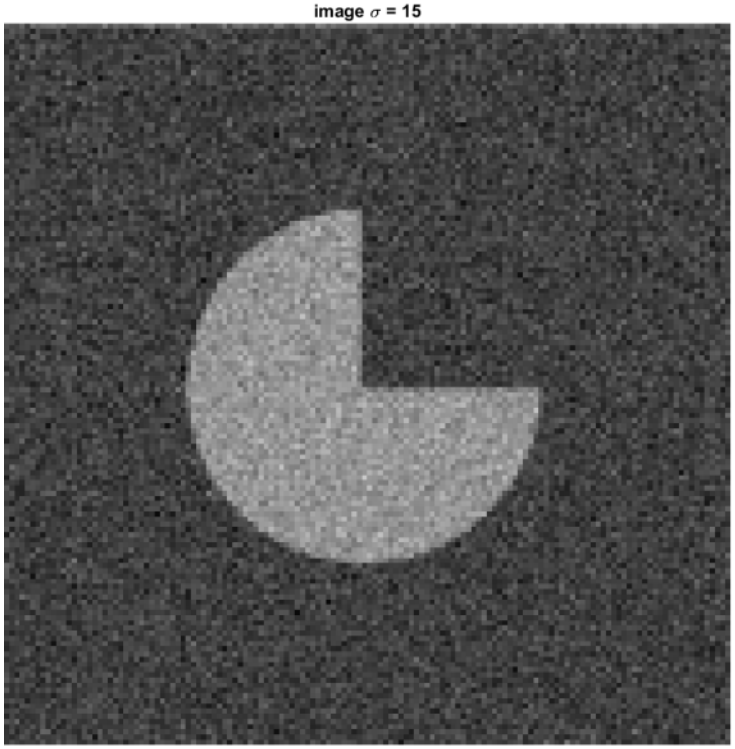
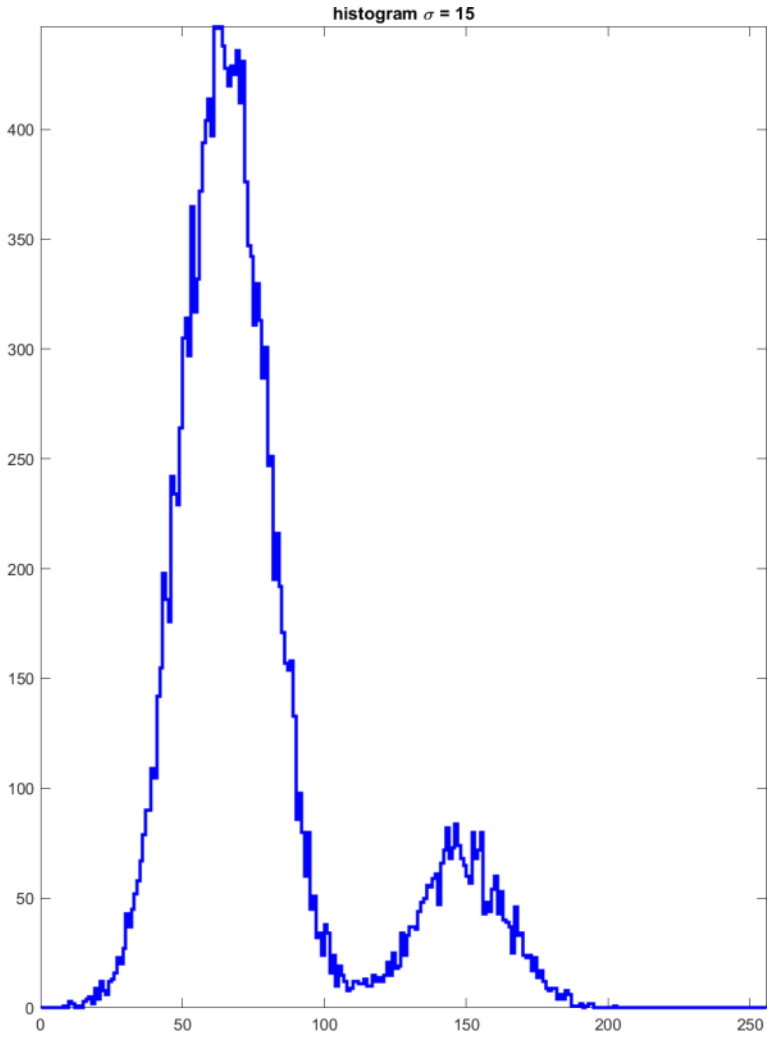


image σ = 15


histogram σ = 15
400
350
300
250
200
150
100
50
0
0
50
100
150
200
250

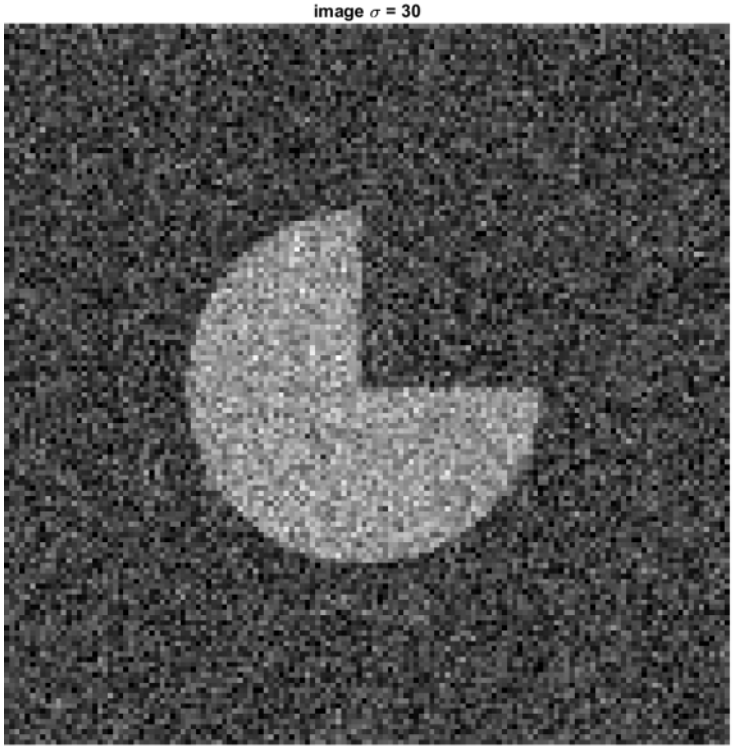
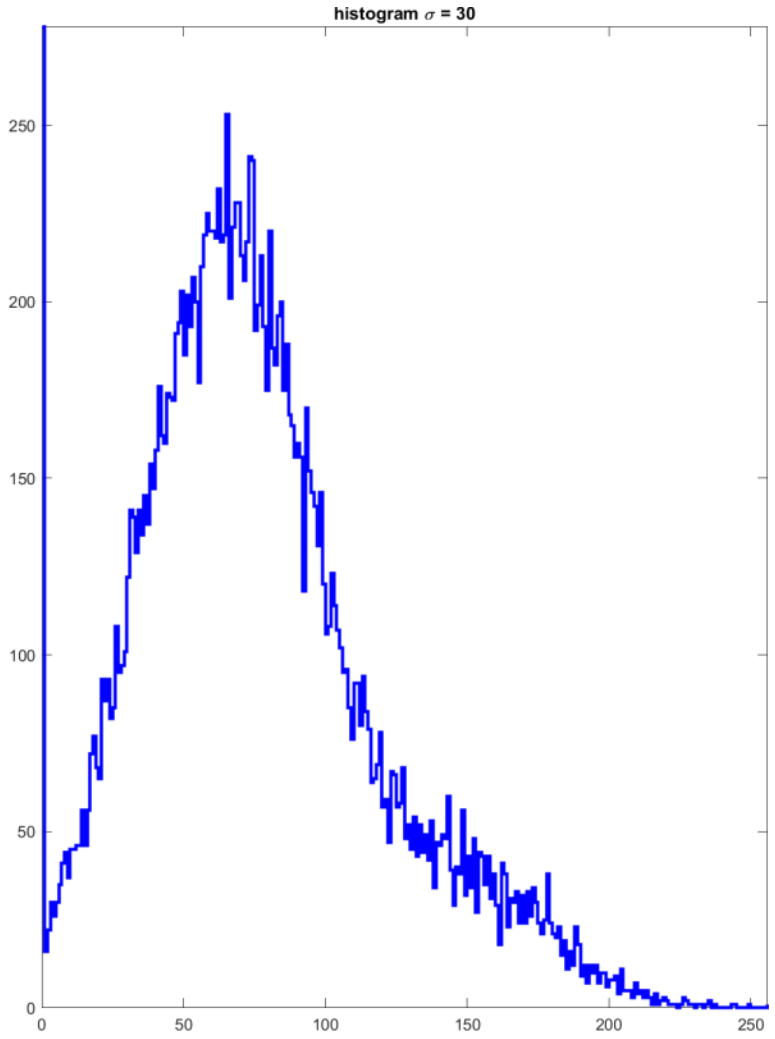
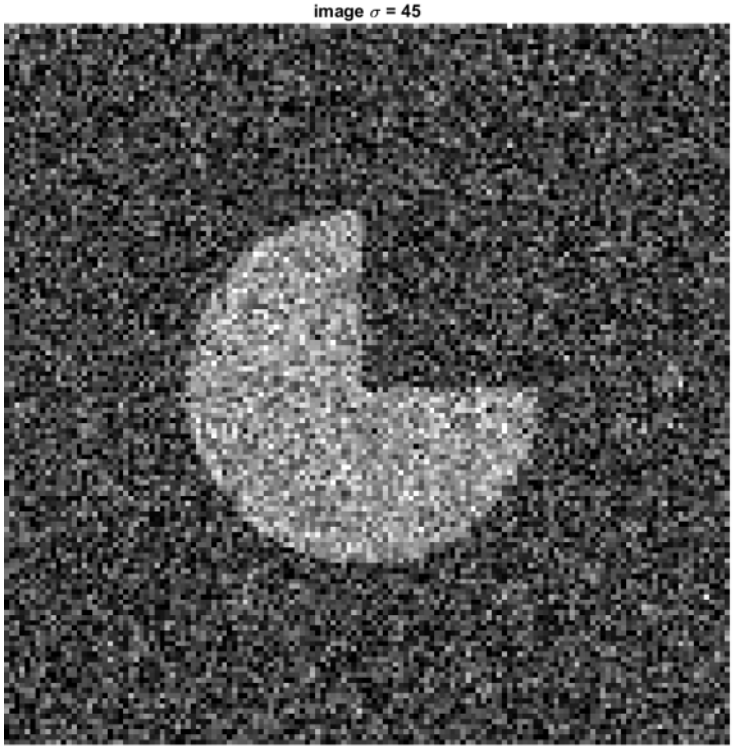
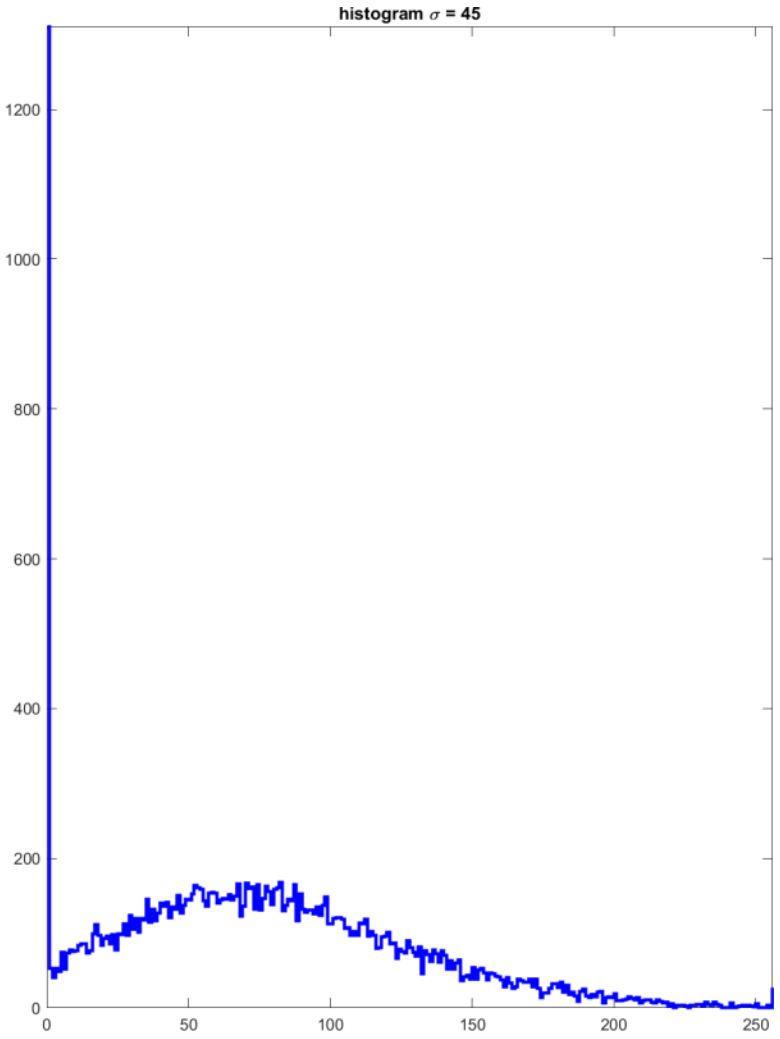


image σ = 30


histogram σ = 30
250
200
150
100
50
0
0
50
100
150
200
250





In Python skimage.filters.threshold_otsu

Otsu Metdod

A method to compute thresholds very efficiently

The threshold estimation problem is formulated as the problem of producing groups that are “as tight as possible”

The criteria to select the threshold is: minimizing the intra-class distribution

This is equivalent to maximizing the inter-class variance, which can be done more efficiently using iterative expressions.

(see the blackboard)

N. Otsu, *”A threshold selection method from gray-level histograms”, IEEE Trans. Sys., Man., Cyber.*, vol. 9, 1979

Morphological Operations

Ordered Statitiscs and Blob Labeling

An overview on morphological operations

Erosion, Dilation

Open, Closure

We assume the image being processed is binary, as these operators are typically meant for refining “mask” images.

Erosion

General definition:
Nonlinear Filtering procedure that replace to each pixel value the minimum on a given neighbor

As a consequence on binary images:

E(x)=1 iff the image in the neighbour is constantly 1

This operation reduces thus the boundaries of binary images

It can be interpreted as an AND operation of the image and the neighbour overlapped at each pixel

Dilation

General definition:
Nonlinear Filtering procedure that replace to each pixel value the maximum on a given neighbor

As a consequence on binary images

$E(x)=1$ iff at least a pixel in the neighbour is 1

This operation grows fat the boundaries of binary images

It can be interpreted as an OR operation of the image and the neighbour overlapped at each pixel

Open and Closure

Open Erosion followed by a Dilation

Closure Dilation followed by an Erosion

Open

Open Erosion followed by a Dilation

- Smooths the contours of an object
- Typically eliminate thin protrusions

Closure

Closure Dilation followed by an Erosion

- Smooths the contours of an object, typically creates bridges
- Generally fuses narrow breaks

There are several other Non Linear Filters

Ordered Statistic based

- Median Filter
- Weight Ordered Statistic Filter
- Trimmed Mean
- Hybrid Median

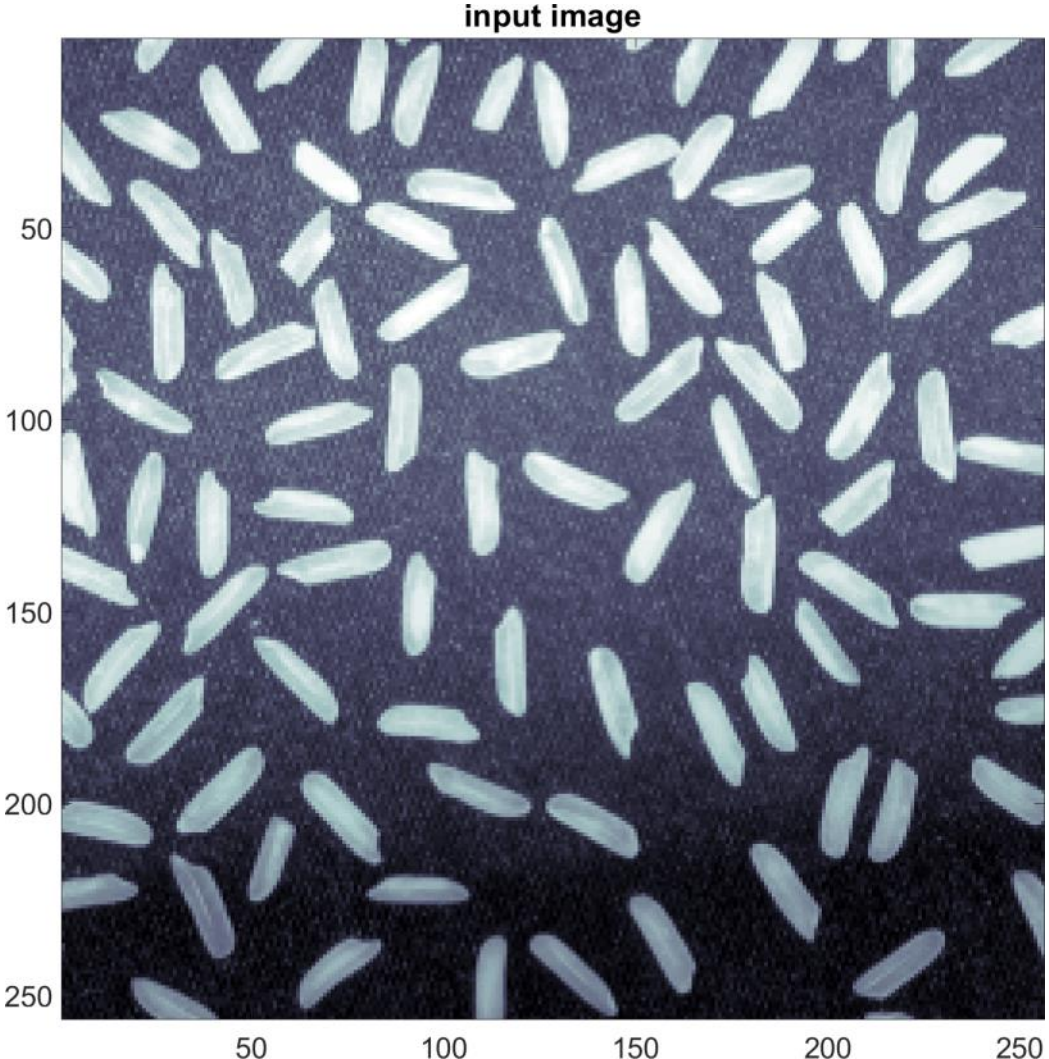
In Python: skimage.morphology

Other Morphological Operations

Blob Labeling, Connected Components

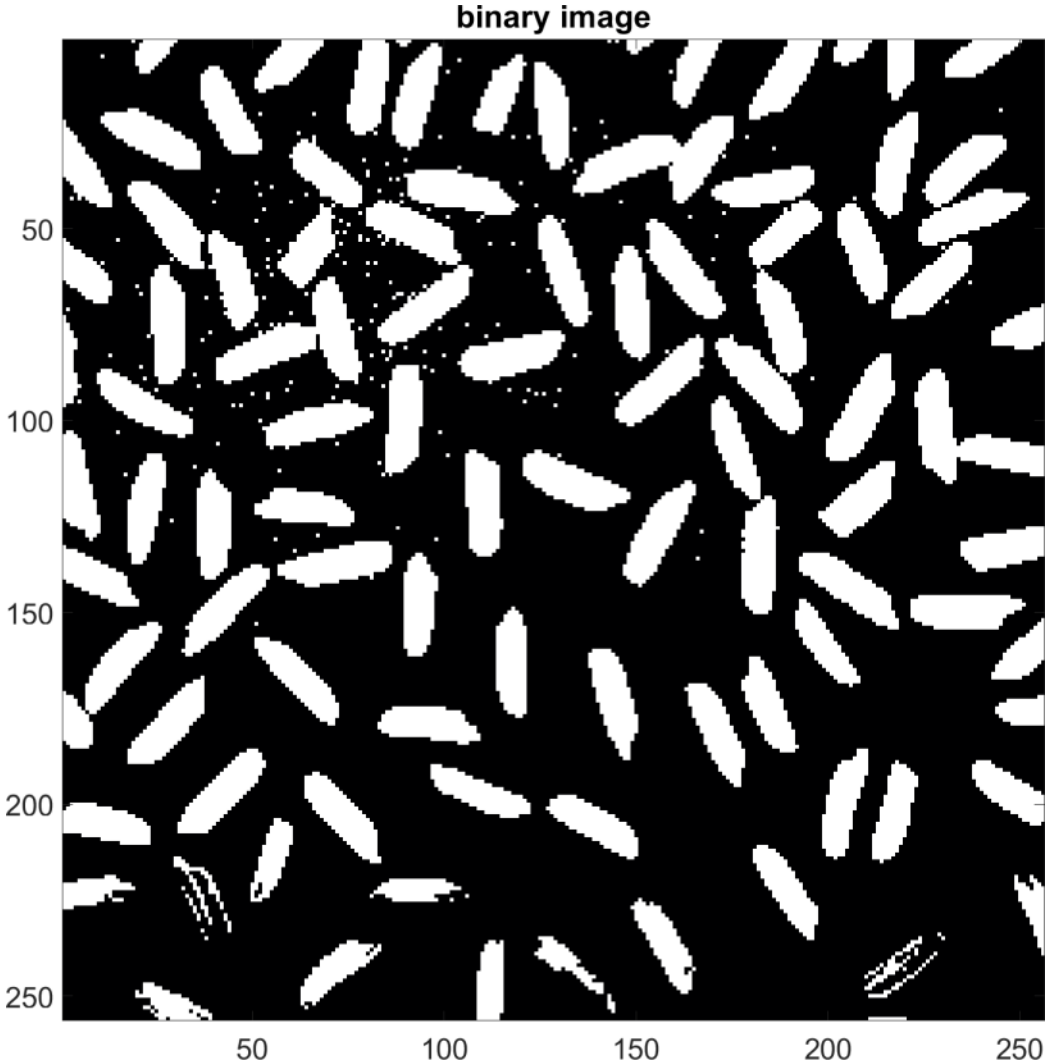
Extraction of connected components

Extract subsets of pixels that are connected according to 4-pixel connectivity or 8-pixel connectivity



Connected components

Extract subsets of pixels that are connected according to 4-pixel connectivity or 8-pixel connectivity

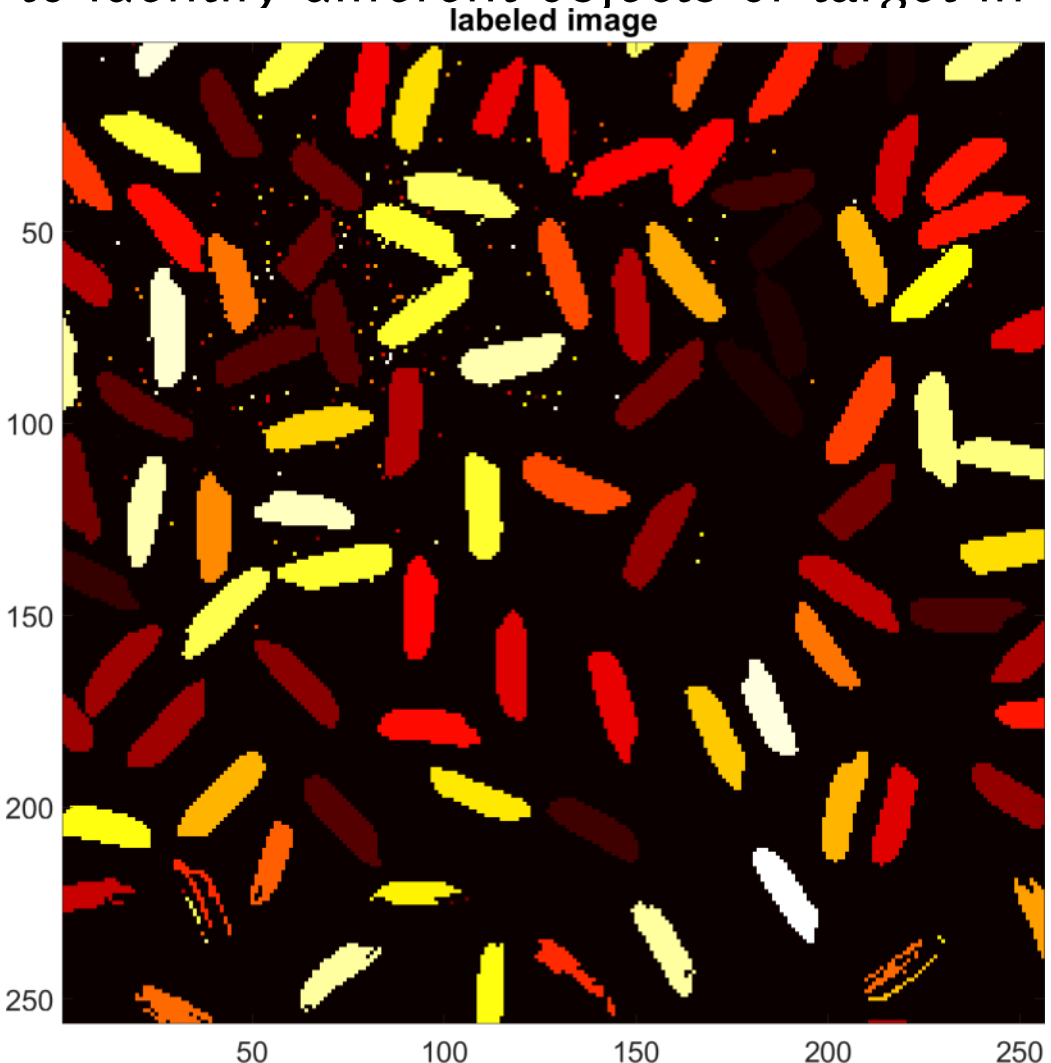


Connected components

Here, each color denotes a different number, i.e. a label.

This allows to identify different objects or target in the scene

Each color corresponds to a different label



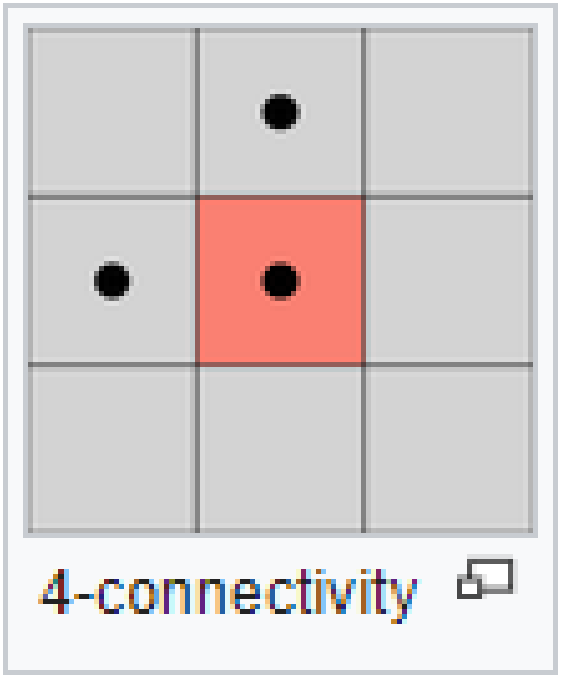
Connected components

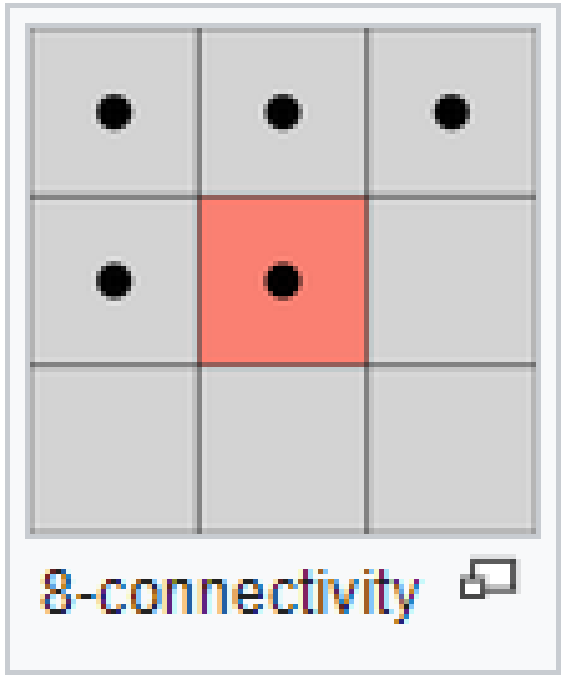
Here, each color denotes a different number, i.e. a label.

This allows to identify different objects or target in the scene

Connected components

Extract subsets of pixels that are connected according to 4-pixel connectivity or 8-pixel connectivity





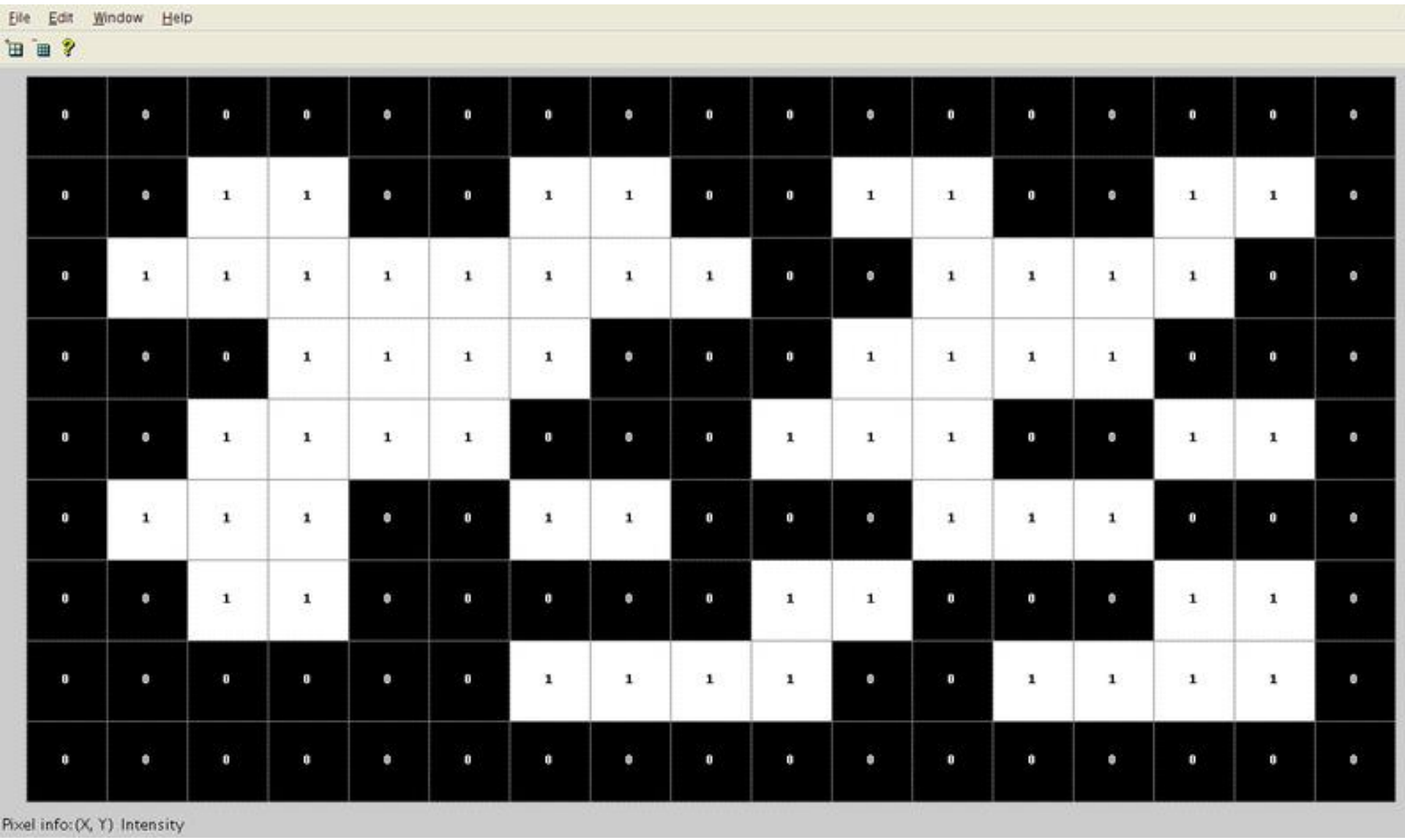
Two Pass Algorithm: First Pass

Iterate through each pixel (r, c)

If $I(r, c) == 1$

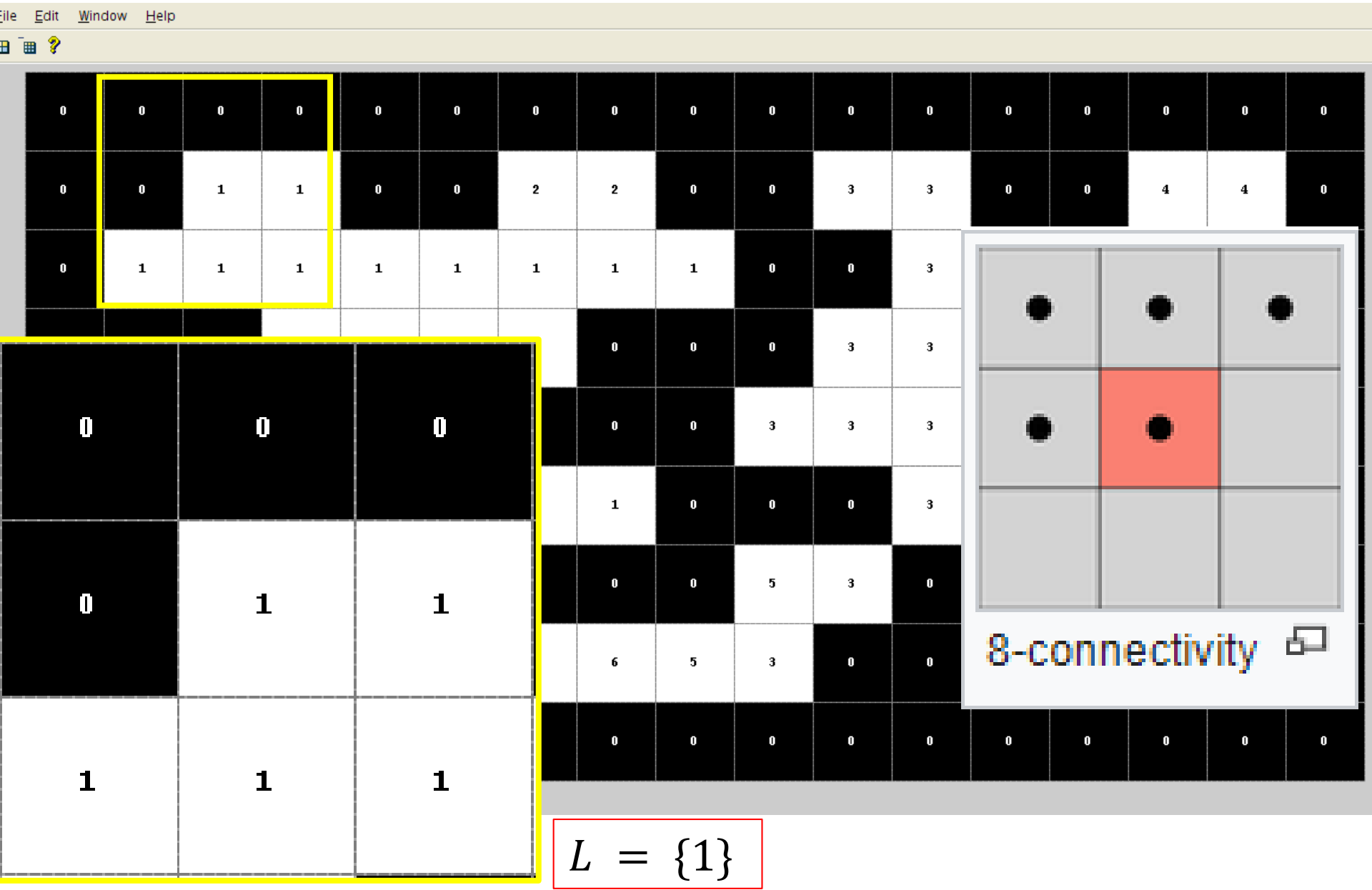
- Get a neighbor $U_{(r,c)}$ of (r, c)
- If $I(u, v) == 0 \ \forall (u, v) \in U_{(r,c)}$
  - Assign a new label $L(r, c)$
- Else $L(r, c) = \min\big(L(u, v)\big)$ over $U_{(r,c)}$
  - If there are different labels in $U_{(r,c)}$
    - Record they are equivalent in a table

Binary input image

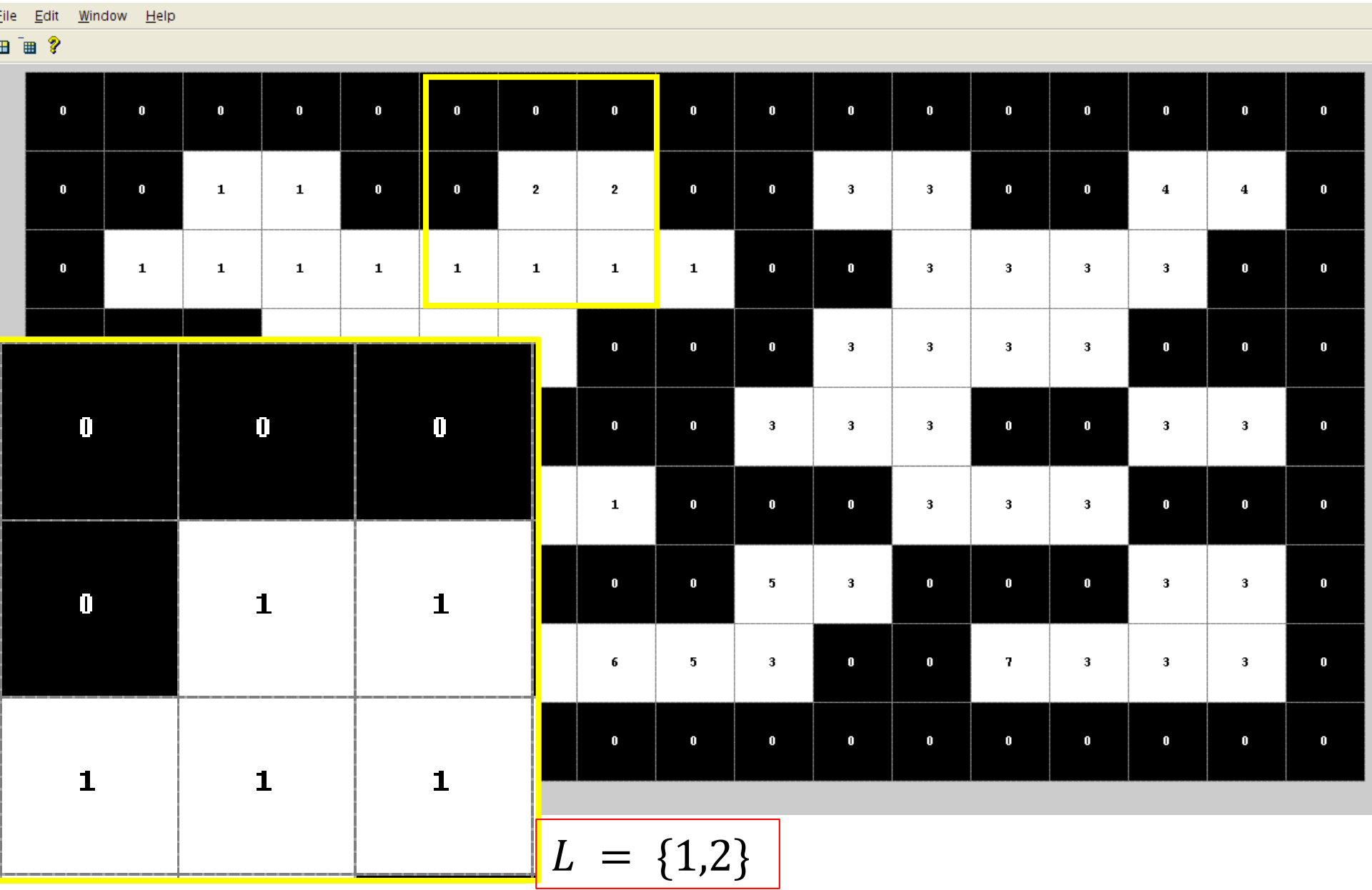


By Dhulloo3 - Own work, Public Domain, https://commons.wikimedia.org/w/index.php?curid=10166860

Iterations of the first pass

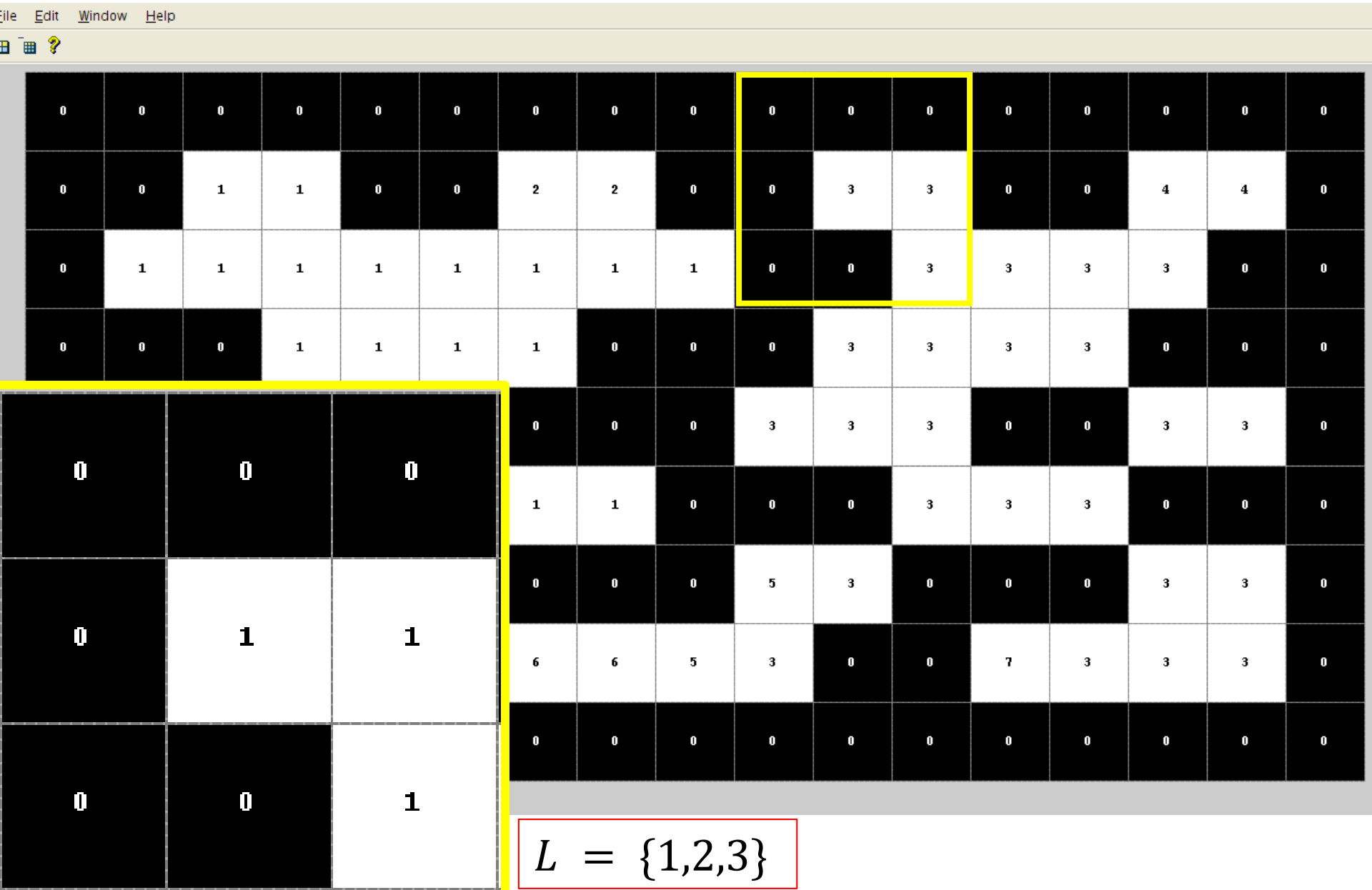


Iterations of the first pass

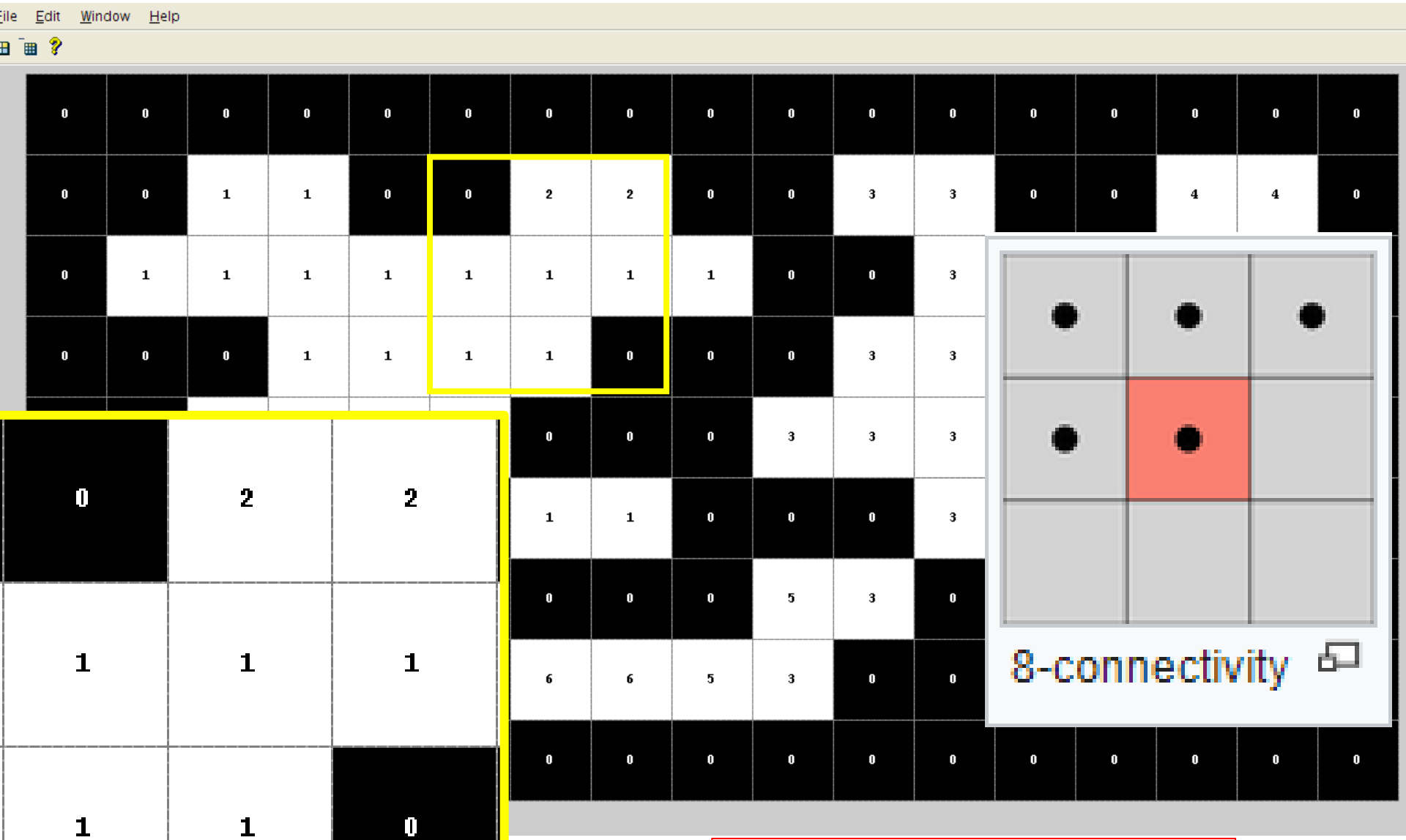


$L = \{1,2\}$

Iterations of the first pass



Iterations of the first pass



$L = \{1,2,3,4\}$ store 1 = 2

Output of the first pass

File Edit Window Help

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	1	0	0	2	2	0	0	3	3	0	0	4	4	0
0	1	1	1	1	1	1	1	1	0	0	3	3	3	3	0	0
0	0	0	1	1	1	1	0	0	0	3	3	3	3	0	0	0
0	0	1	1	1	1	0	0	0	3	3	3	0	0	3	3	0
0	1	1	1	0	0	1	1	0	0	0	3	3	3	0	0	0
0	0	1	1	0	0	0	0	0	5	3	0	0	0	3	3	0
0	0	0	0	0	0	6	6	5	3	0	0	7	3	3	3	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

xel info:(X, Y) Intensity

$L = \{1,2,3,4,5,6,7\}$ equivalence sets $\{1,2\}$, $\{3,4,5,6,7\}$

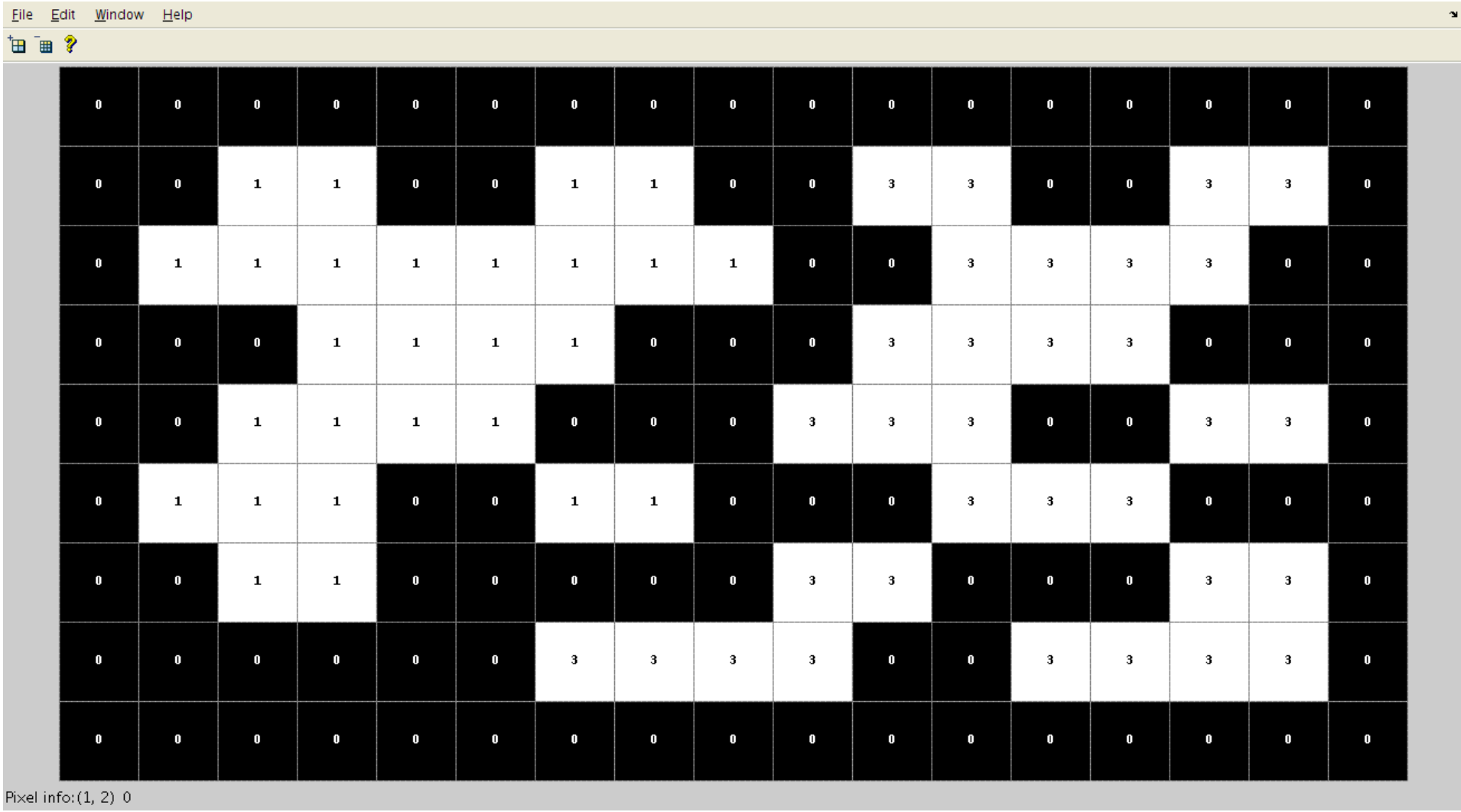
Two Pass Algorithm: First Pass

Iterate through each pixel (r, c)

If $I(r, c) == 1$

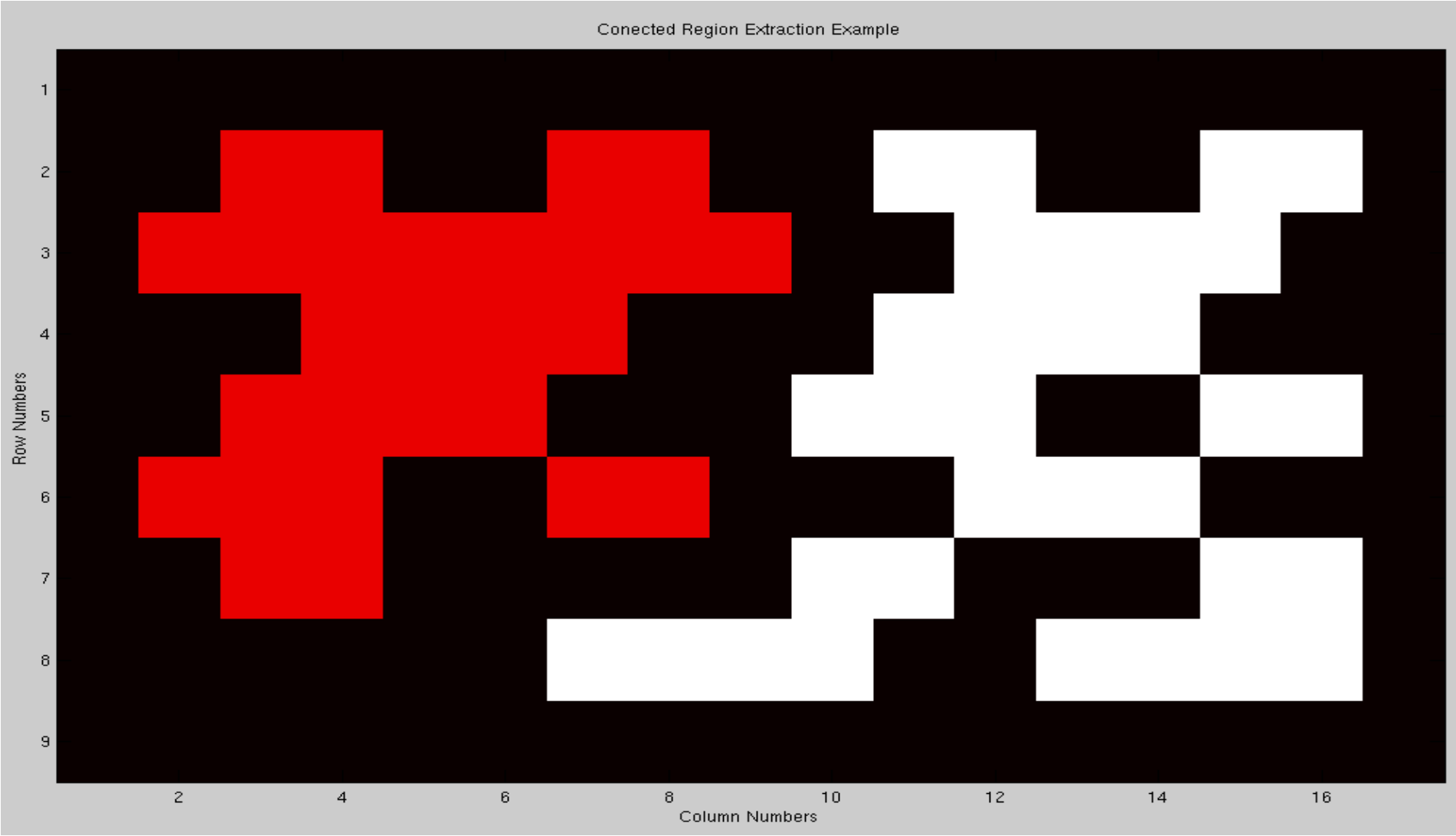
Relabel the element with the lowest equivalent label

Output of the Second Pass



By Dhull003 - Own work, Public Domain, https://commons.wikimedia.org/w/index.php?curid=10166888

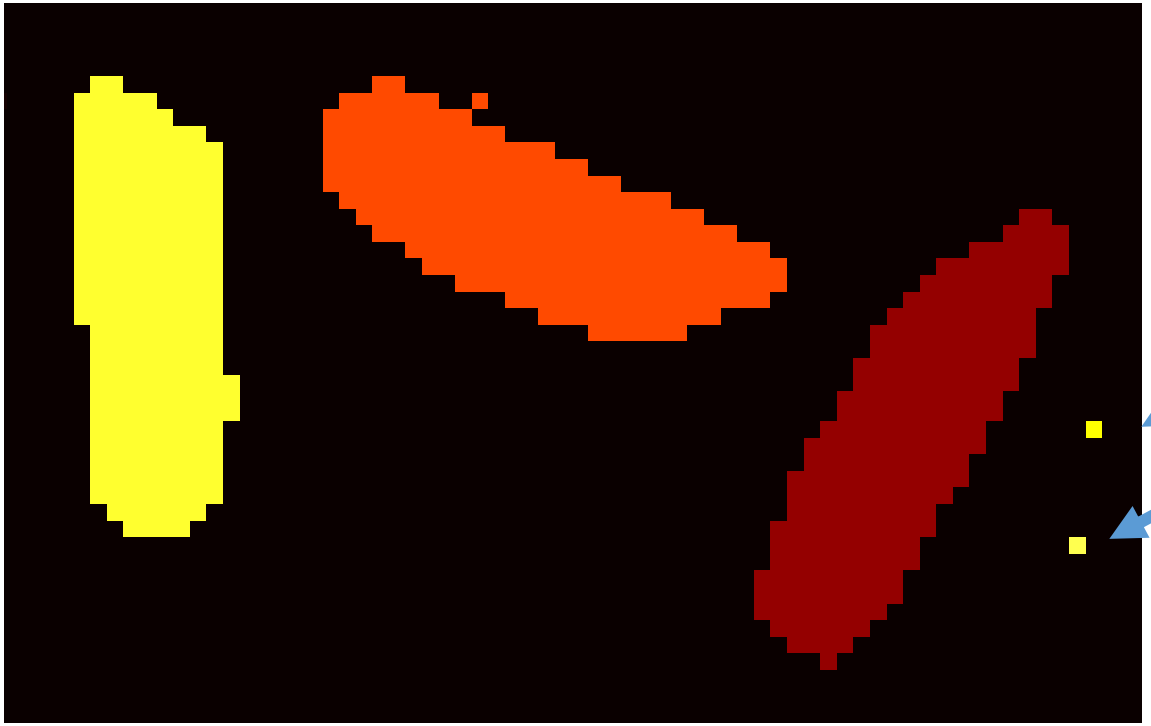
Output of the Second Pass



By Dhull003 - Own work, Public Domain, https://commons.wikimedia.org/w/index.php?curid=10166888

Bounding Box vs Axis

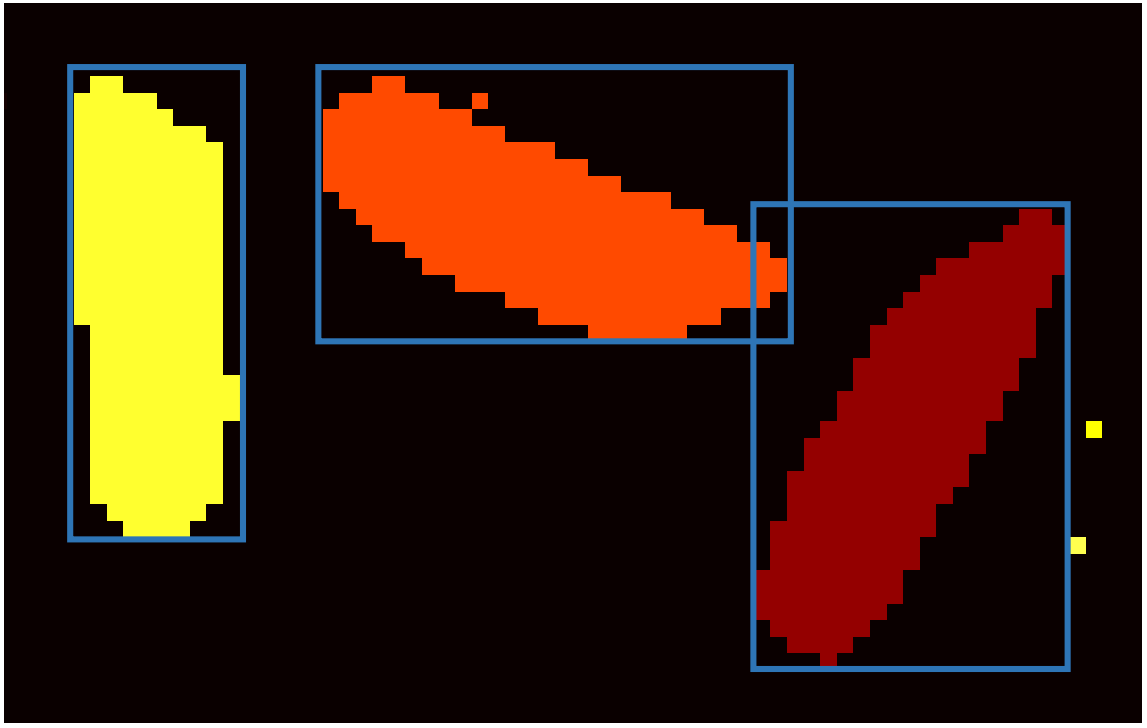
These provide information about size and orientation of the object



Too small, can be removed analyzing the area

Bounding Box vs Axis

These provide information about size and orientation of the object

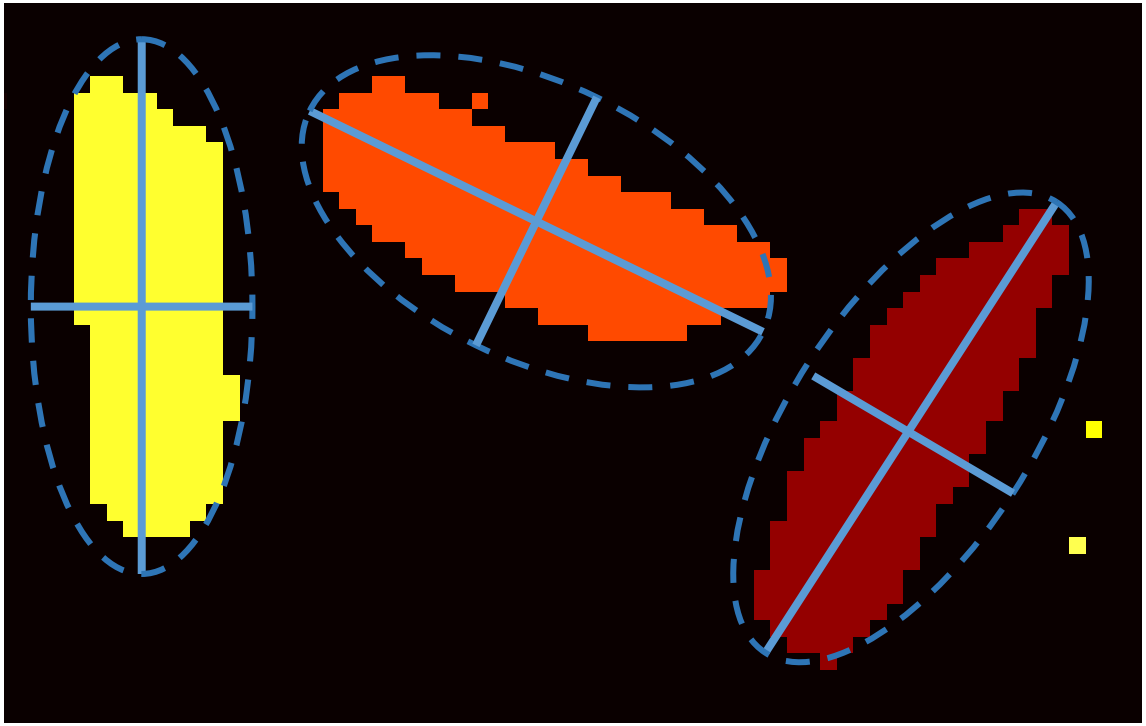


Bounding box allow to crop separate images for each component

(these are defined as the range of values for each coordinate)

Bounding Box vs Axis

These provide information about size and orientation of the object



Blob axis are computer as axis the of the ellipse that has the same second-moments as the region.

In Python: skimage.measure

Features from Morphological Image Processing

Now operations can be performed on each blob individually

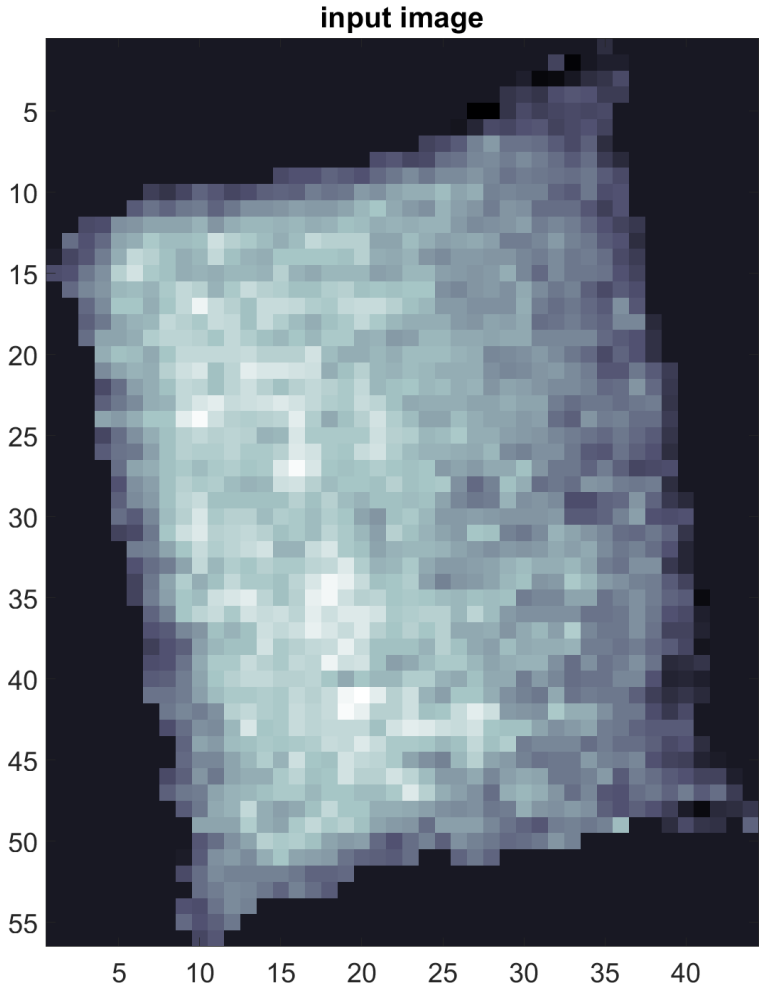
- Remove blobs that are too small
- Ehnance boundaries by morphological operation

For instance, you can for each blob

- Compute the area
- Compute different image statistics over each blob (average intensity, standard deviation, squared error w.r.t. regression model) and over central / perimetral area
- Compute Bounding Box and the aspect ratio
- Compute the Axis
- Analyze the location in the image
- …. compute the convex hull…

Boundary Extraction

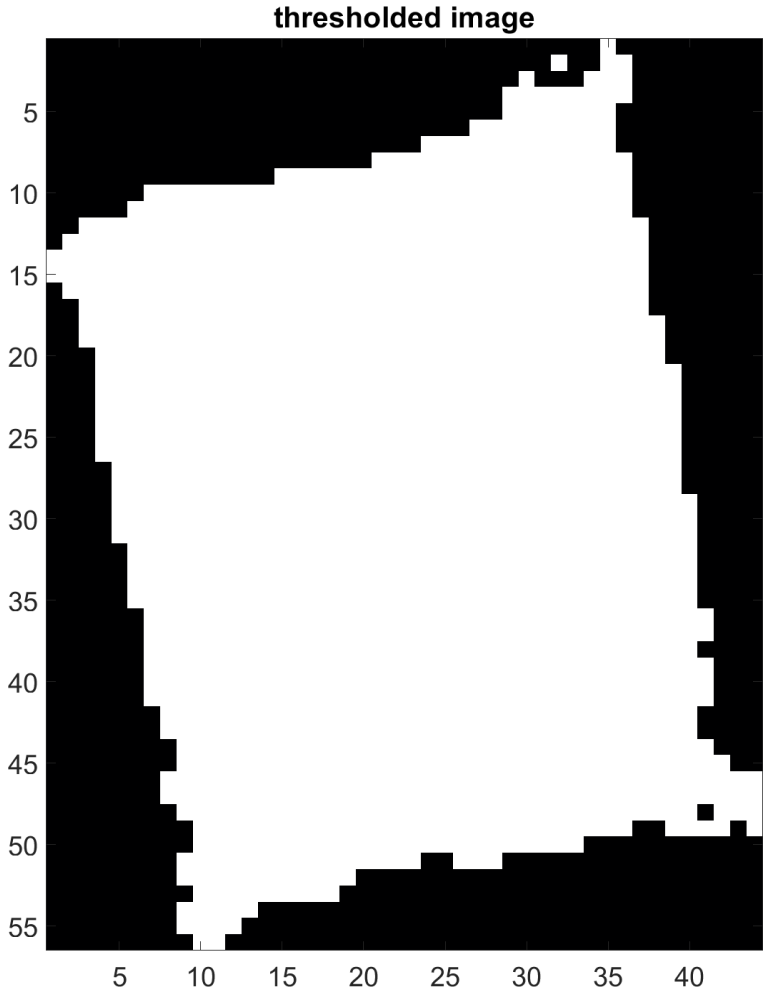
The simplest way to extract boundaries of an image is to subtract from a binary image its eroded version



I

Boundary Extraction

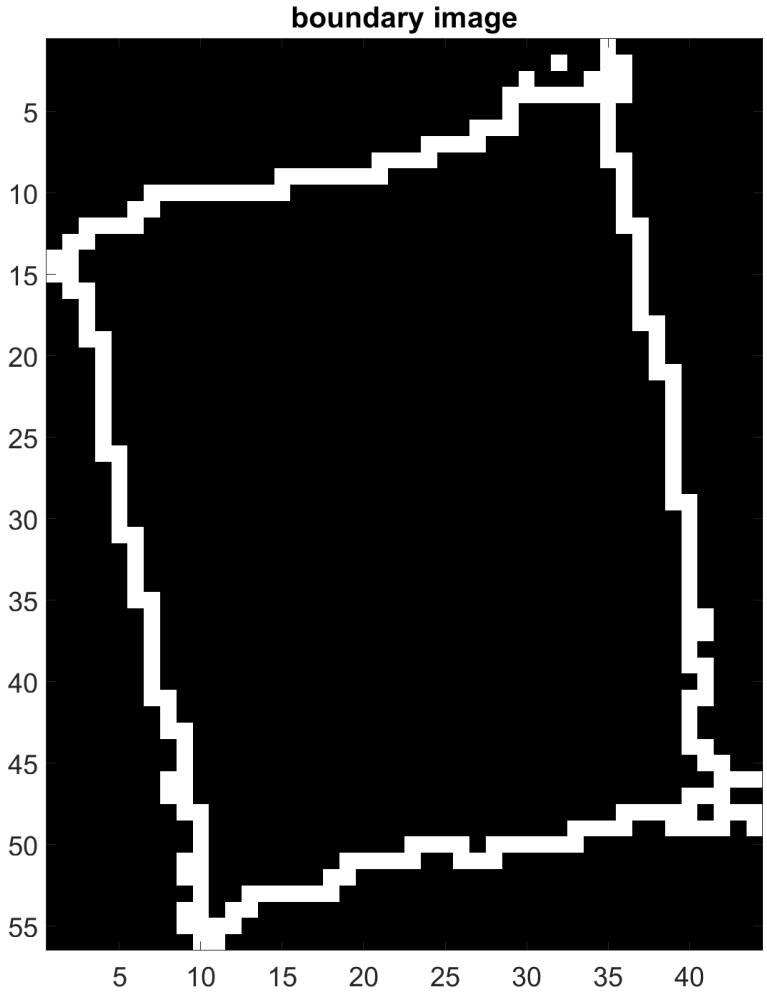
The simplest way to extract boundaries of an image is to subtract from a binary image its eroded version



$$T = I > \Gamma$$

Boundary Extraction

The simplest way to extract boundaries of an image is to subtract from a binary image its eroded version



$$B = T \ominus h$$

$$h = \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$

Feature Extraction From Boundaries

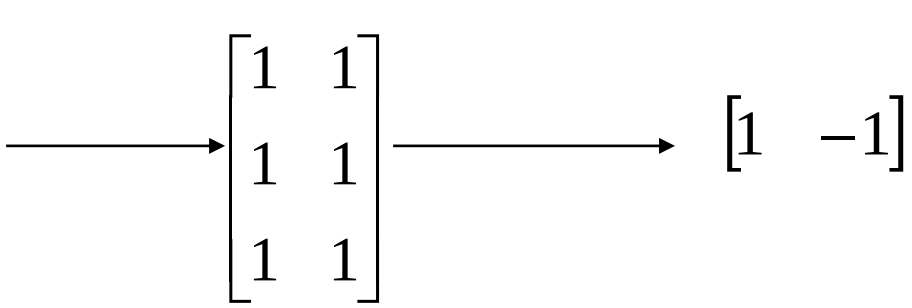
For each blob, extract

- Perimeter
- Area
- Perimeter-area ratio
- Edges orientation (it would be good to suitably rotate the image before)

Edge Detection and Line Detection

Classical Operators: Prewitt

Horizontal derivative

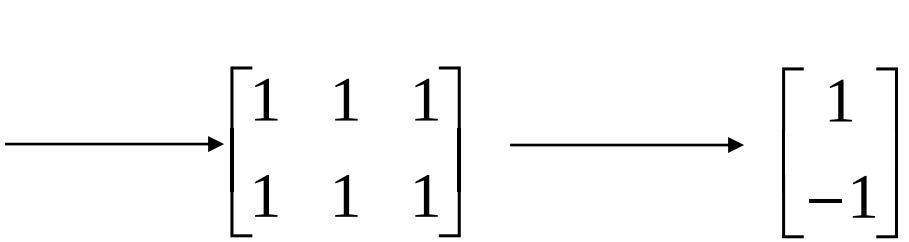


Smooth

Differentiate

$$h = \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}$$

Vertical derivative



$$h' = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

Classical Operators: Sobel

Horizontal derivative

$$\rightarrow \begin{bmatrix} 1 & 1 \\ 2 & 2 \\ 1 & 1 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & -1 \end{bmatrix} \qquad h = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$$

Smooth Differentiate

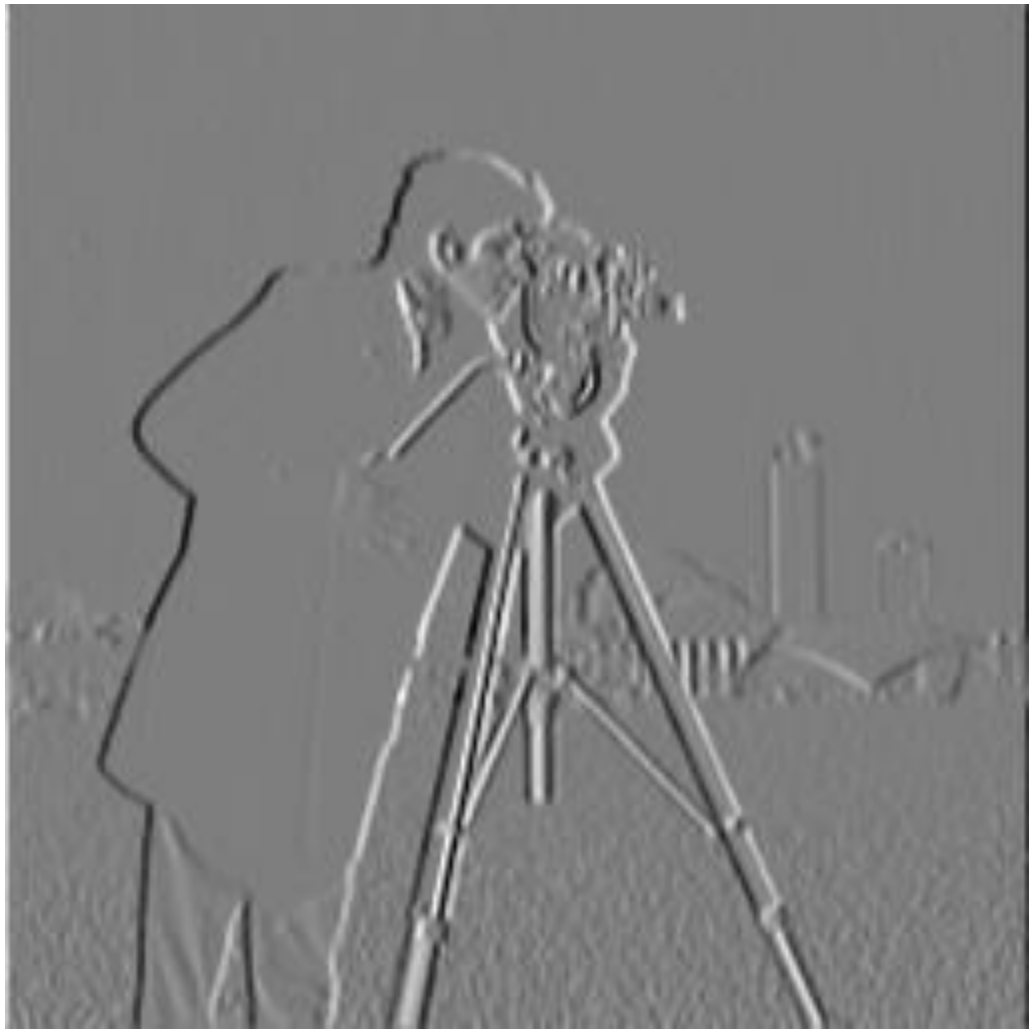
Vertical derivative

$$\rightarrow \begin{bmatrix} 1 & 2 & 1 \\ 1 & 2 & 1 \end{bmatrix} \rightarrow \begin{bmatrix} 1 \\ -1 \end{bmatrix} \qquad h' = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Another famous test image - cameraman

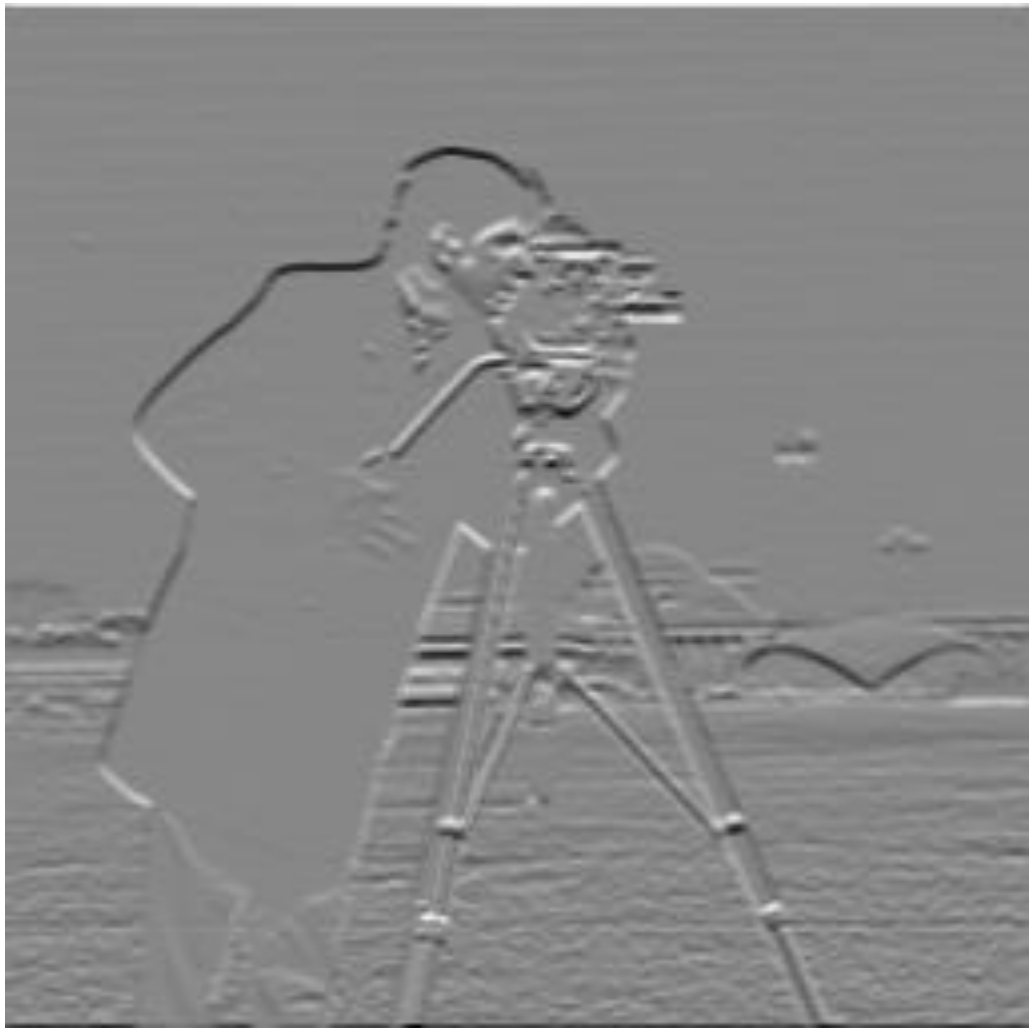
Horizontal Derivatives using Sobel

$$(I \circledast h_x)$$



Vertical Derivatives using Sobel

$$\left(I \circledast h_y\right)$$



Gradient Magnitude

$$\|\nabla I\| = \sqrt{(I \circledast d_x)^2 + (I \circledast d_y)^2}$$

$$\nabla I(r,c) = \begin{bmatrix} I \circledast d_x \\ I \circledast d_y \end{bmatrix}(r,c)$$

The Gradient Orientation

Like for continuous function, the gradient in each pixel points at the **steepest growth/decrease direction.**

$$\angle \nabla I(r,c) = \text{atand}\left(\frac{\nabla I_2(r,c)}{\nabla I_1(r,c)}\right) = \text{atand}\left(\frac{(I \circledast d_x)(r,c)}{(I \circledast d_x)(r,c)}\right)$$

The gradient norm indicates the strength of the intensity variation

Gradient Magnitude and edge detectors

The gradient Magnitute is not a binary image

We can see edges but we cannot identify them, yet

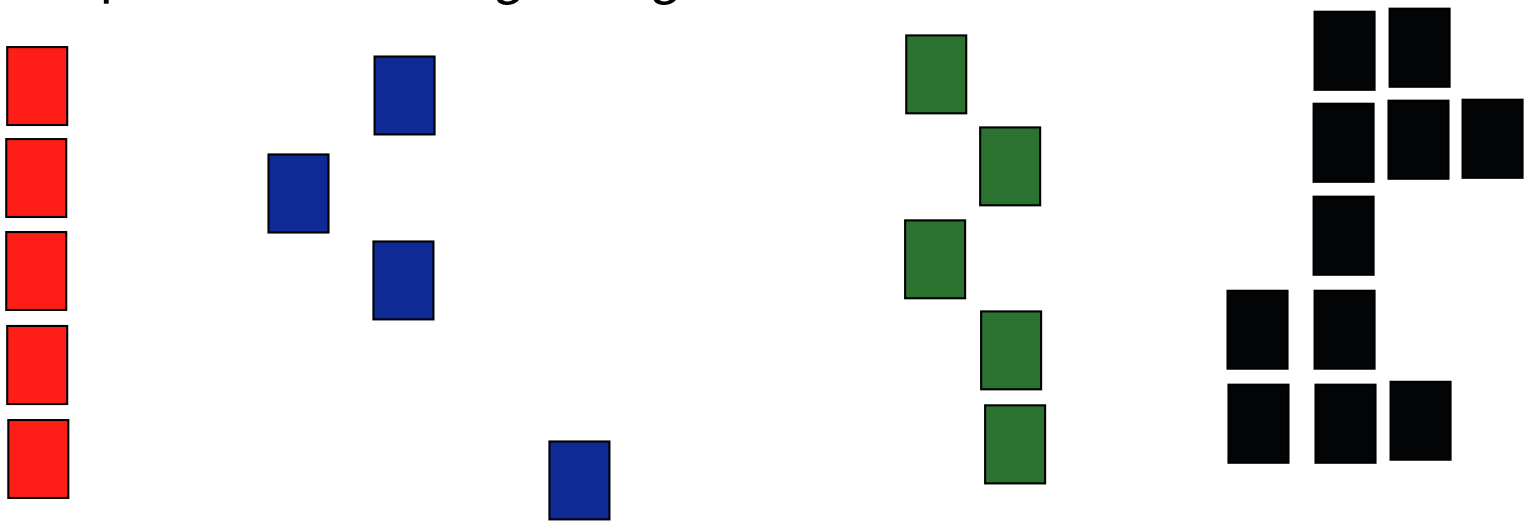
This is a serious limitation as, for instance, you cannot directly compute the number of pixels (perimeter?) or other information as if this was an estimate of the edge

Canny Edge Detector Criteria

Good Detection: The optimal detector must minimize the probability of false positives as well as false negatives.

Good Localization: The edges detected must be as close as possible to the true edges.

Single Response Constraint: The detector must return one point only for each edge point. similar to good detection but requires an ad-hoc formulation to get rid of multiple responses to a single edge



True Edge | Poor singnal-to-noise ratio | Poor localization | Too many responses

Canny Edge Detector

It is characterized by 3 important steps

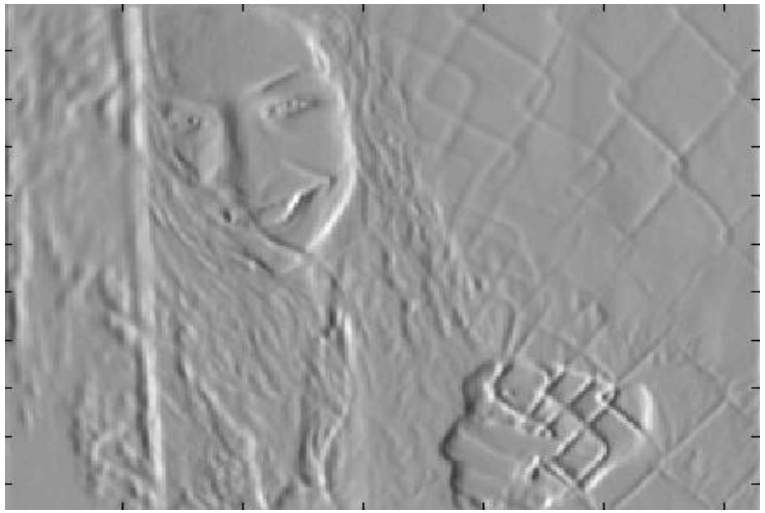
- Convolution with derivative of Gaussian before computing image derivatives
- Non-maximum Suppression
- Hysteresis Thresholding

J. Canny *"A Computational Approach to Edge Detection"* IEEE PAMI vol 8, no. 6, Nov. 1986

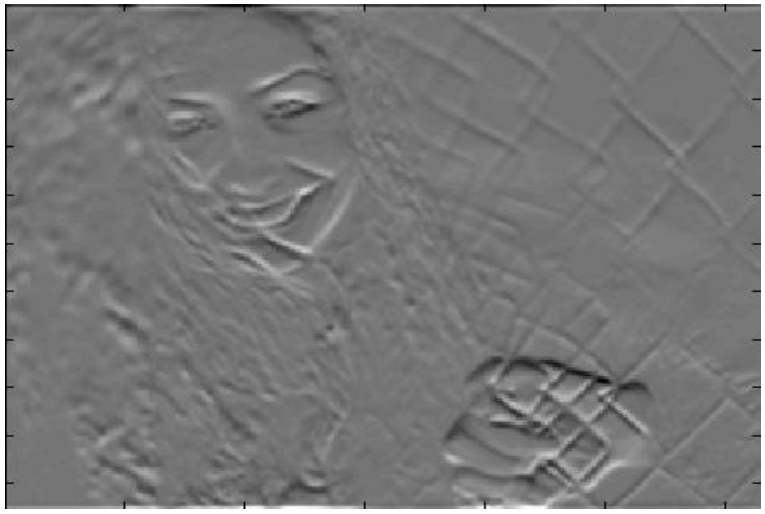
Canny Edge Detector

I

∇I_x



∇I_y



Canny Edge Detector

$\left|\left|\nabla I\right|\right|_2$

I

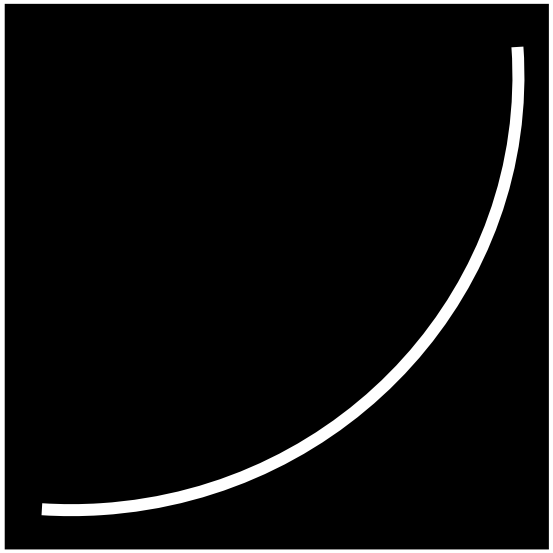
$\left|\left|\nabla I\right|\right|_2 > 25$

Still, there are too many responses

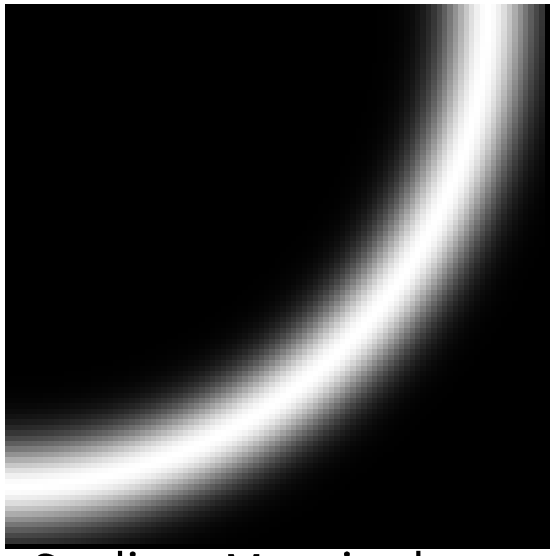
Non-Maximum Suppression: The Idea

We wish to determine the points along the curve where the gradient magnitude is largest.

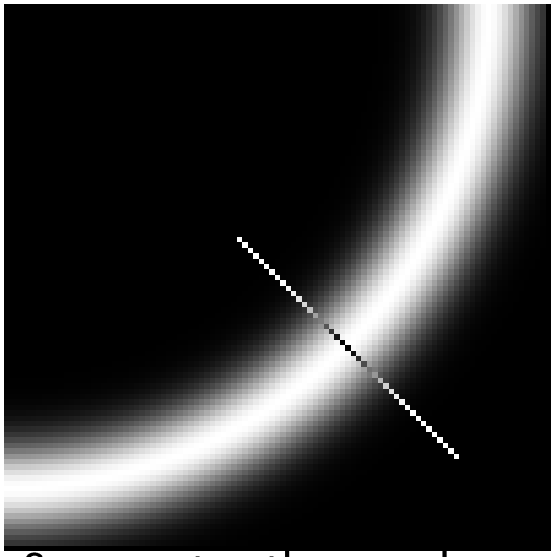
Non-maximum suppression: we look for a maximum along a slice orthogonal to the curve. These points form a 1D signal.



Original Image

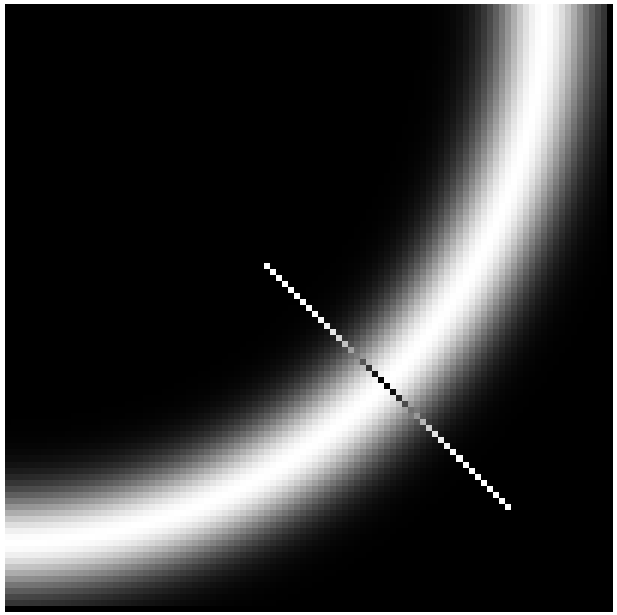


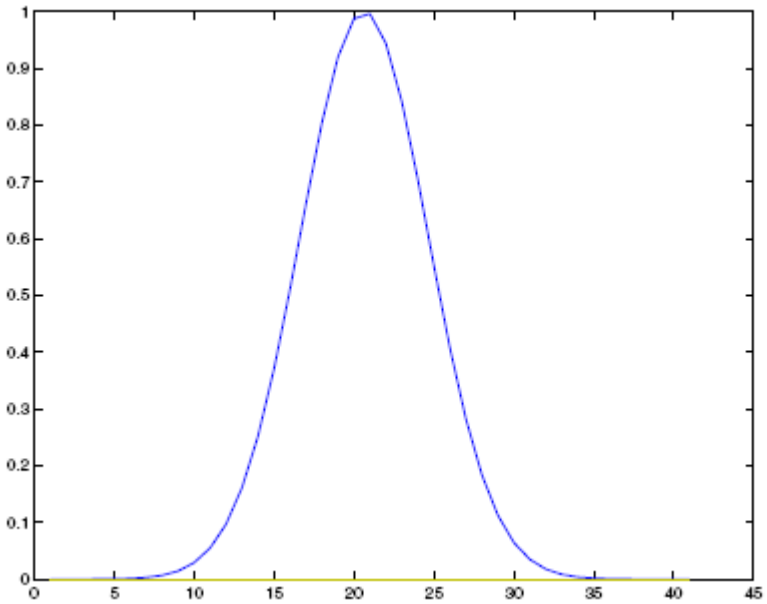
Gradient Magnitude



Segment orthogonal

Non-Maximum Suppression – Idea





Non-Maximum Suppression: The Idea

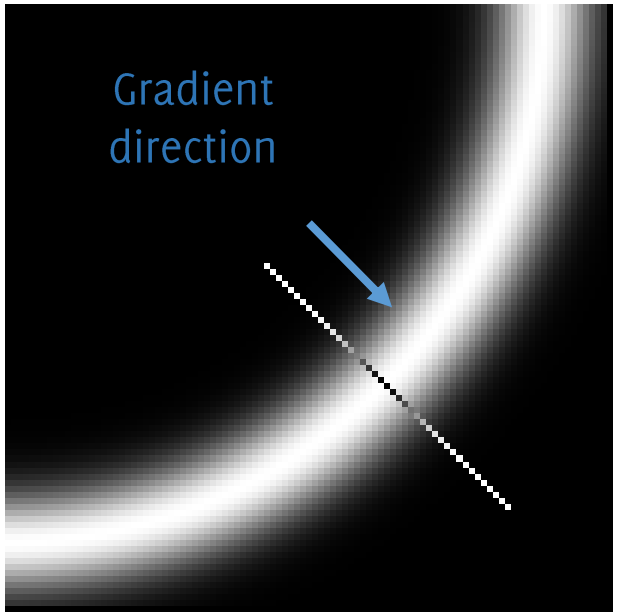
We wish to determine the points along the curve where the gradient magnitude is largest.

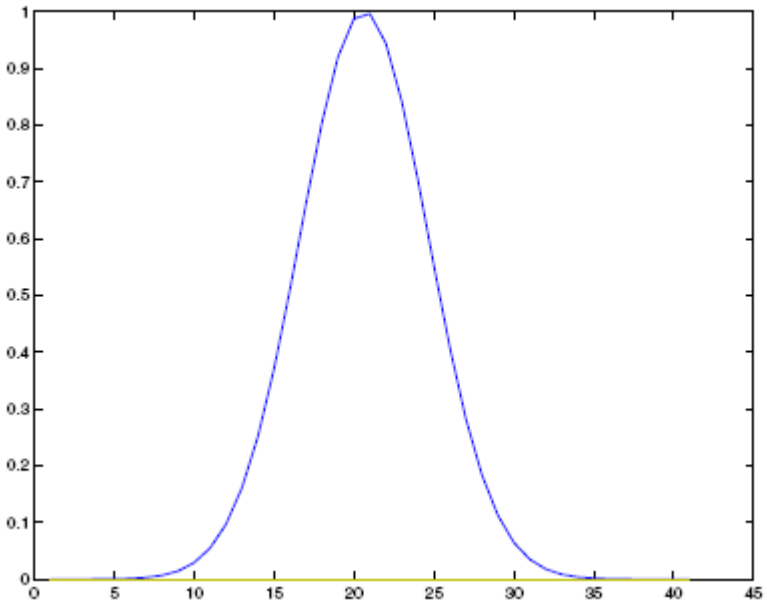
Non-maximum suppression: we look for a maximum along a slice orthogonal to the curve. These points form a 1D signal.

There are two issues:

- which slice to select? Which pixels to consider to extract the maximum?
- once an edge pixel has been found, which pixels to consider next?

Non-Maximum Suppression – Idea (II)





In each pixel, the gradient indicates the direction of the steepest variation: thus, the gradient is orthogonal to the edge direction (no variation along the edge).
We have to consider pixels on a segment following the gradient direction

The intensity profile along the segment.
We can easily identify the location of the maximum.

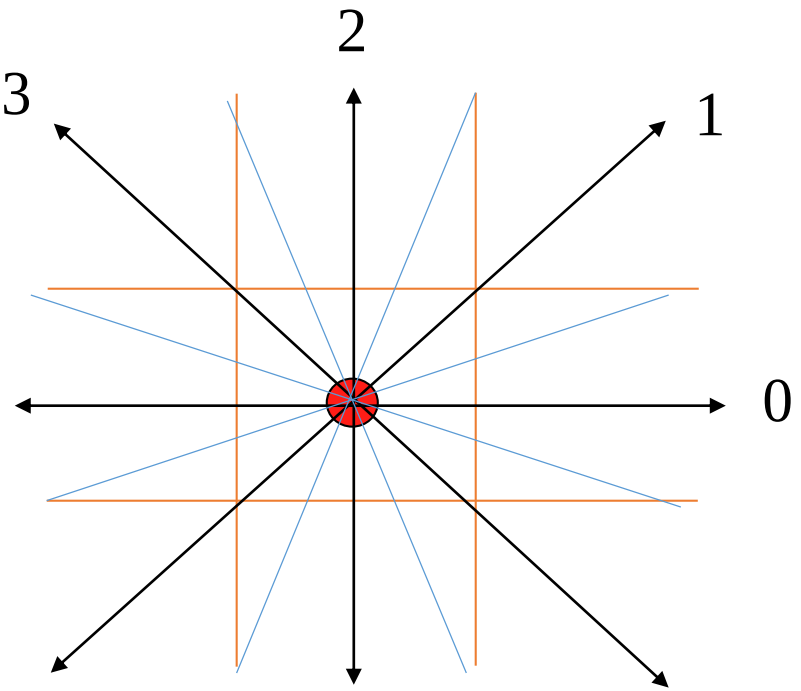
Non-Maximum Suppression: Quantize Gradient Directions

In practice the gradient directions are quantized according to 4 main directions, each covering 45°

Thus, only diagonal or vertical line segments are considered

We consider 4 directions 0,1,2, 3

$$\theta(\boldsymbol{x_0}) = \text{atan}\left(\frac{\partial/\partial y\, I(\boldsymbol{x_0})}{\partial/\partial x\, I(\boldsymbol{x_0})}\right)$$



Orientation is of irrelevant for segment extraction

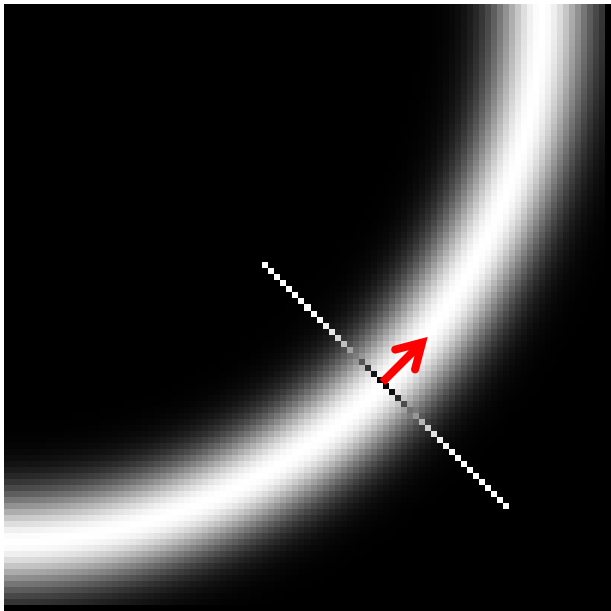
Tracking the edge direction

The direction orthogonal to the gradient follows the edge

Once a local maxima is found, we consider the direction orthogonal to the gradient in that point,

The direction is quantized as for extracting the 1D segment for nonmaximum suppression

We move one step in the quantized direction to determine another point where to extract 1D segments



Non-Maximum Suppression

$$\left|\left|\nabla I\right|\right|_2$$

$$M > 25$$

Hysteresis Thresholding

- Use of two different thresholds High and Low for
  - For new edge starting point
  - For continuing edges

- In such a way the edges continuity is preserved

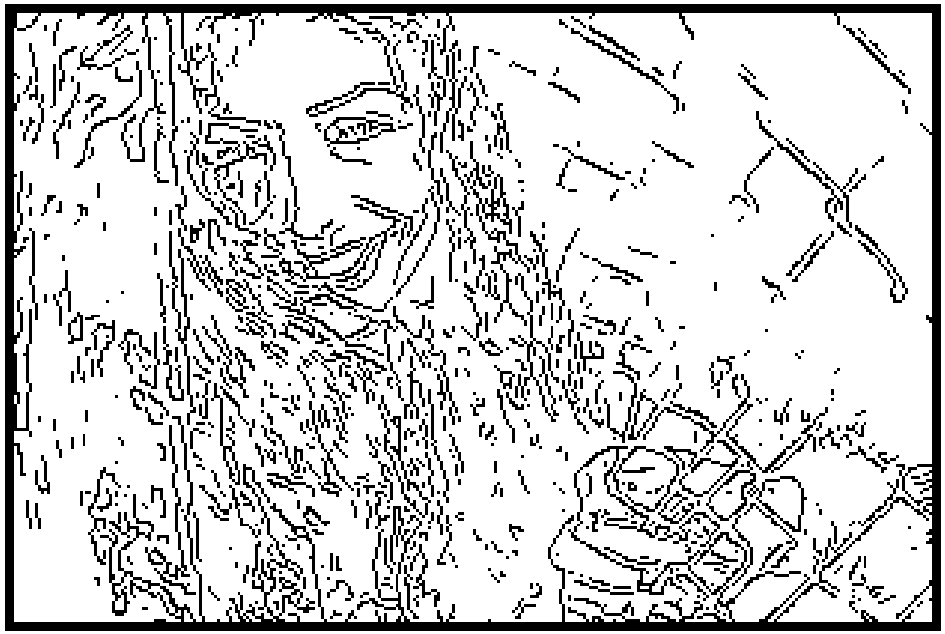
Hysteresis Thresholding

If the gradient at a pixel is above ‘High’, declare it an ‘edge pixel’.

If the gradient at a pixel is below ‘Low’, declare it a ‘non-edge-pixel’.

If the gradient at a pixel is between ‘Low’ and ‘High’ then declare it an ‘edge pixel’ if and only if it is connected to an ‘edge pixel’ directly or via pixels between ‘Low’ and ‘ High’.

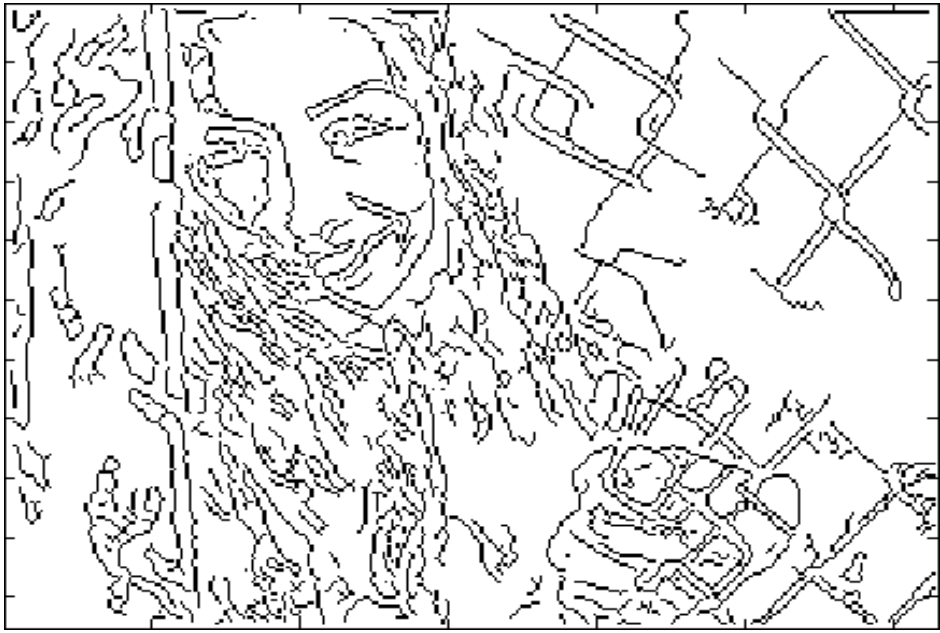
Hysteresis Thresholding



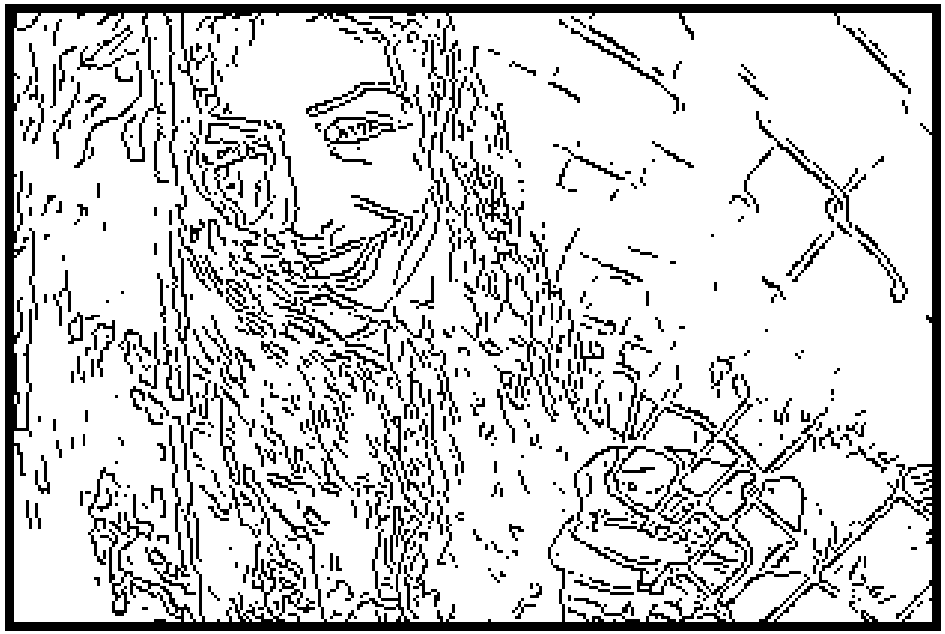
M

$High = 35$

$Low = 15$



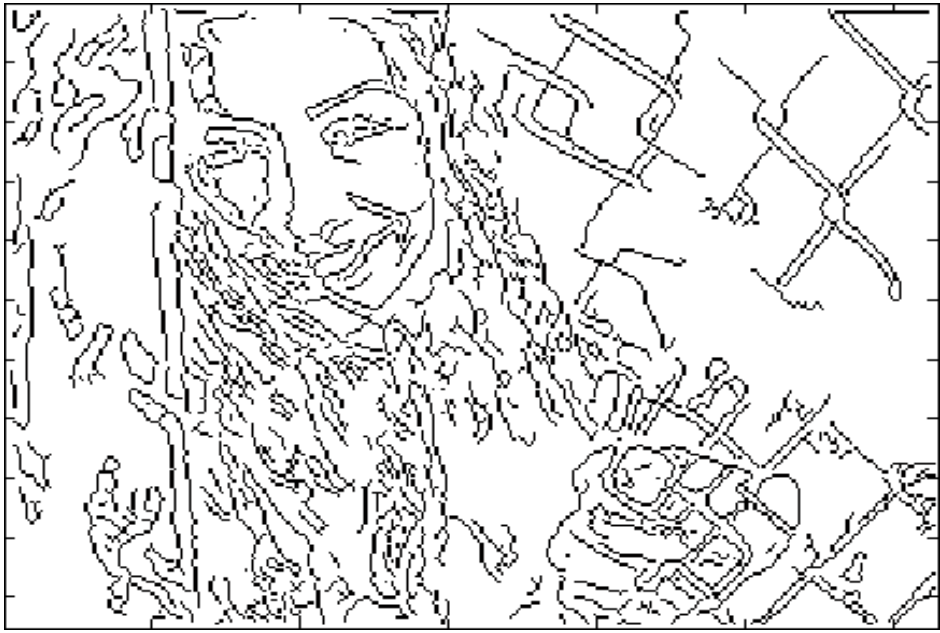
Hysteresis Thresholding



M

$High = 35$

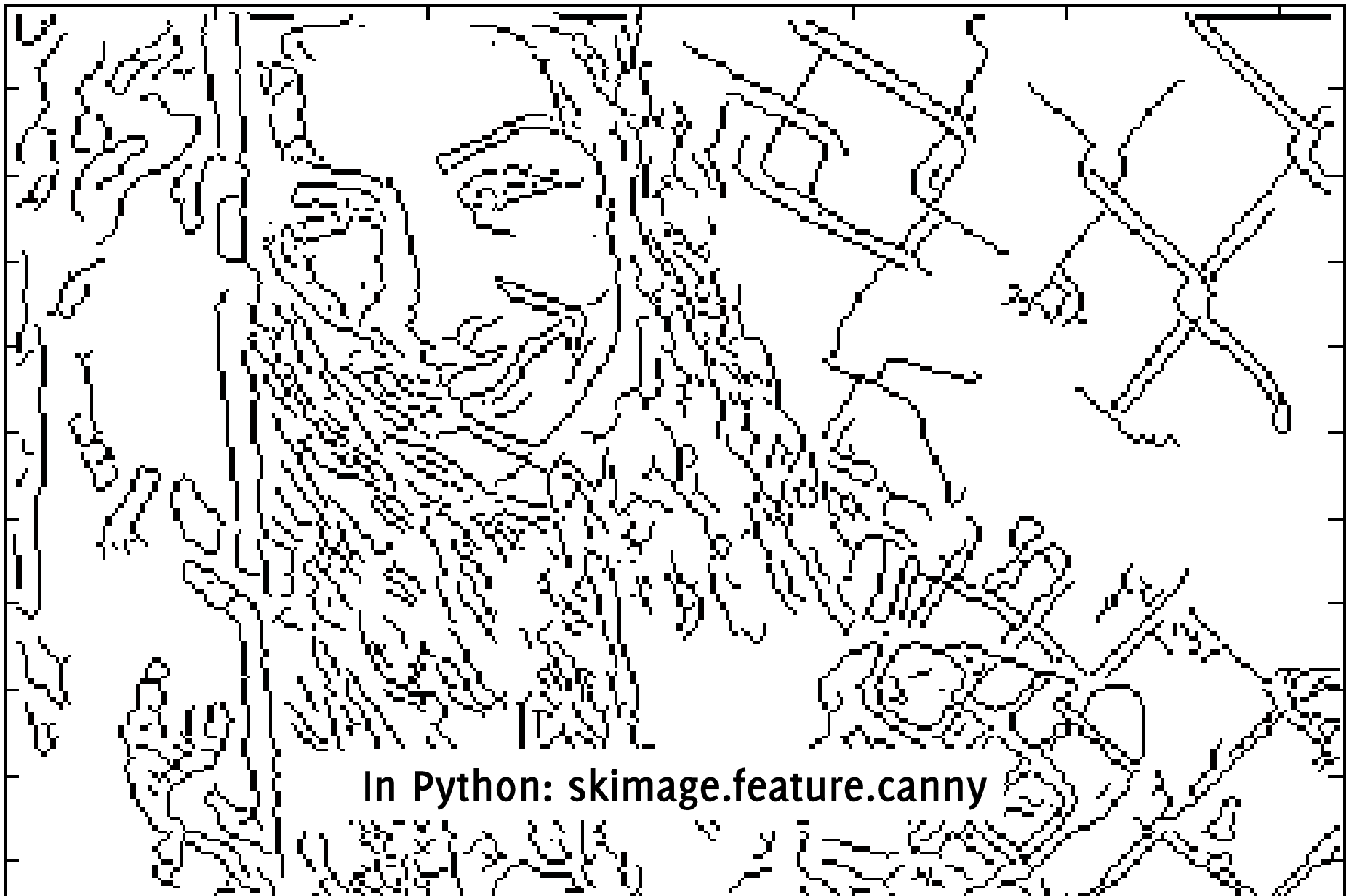
$Low = 15$



Original Image

Canny Edge Detection Results

Canny Edge Detection Results

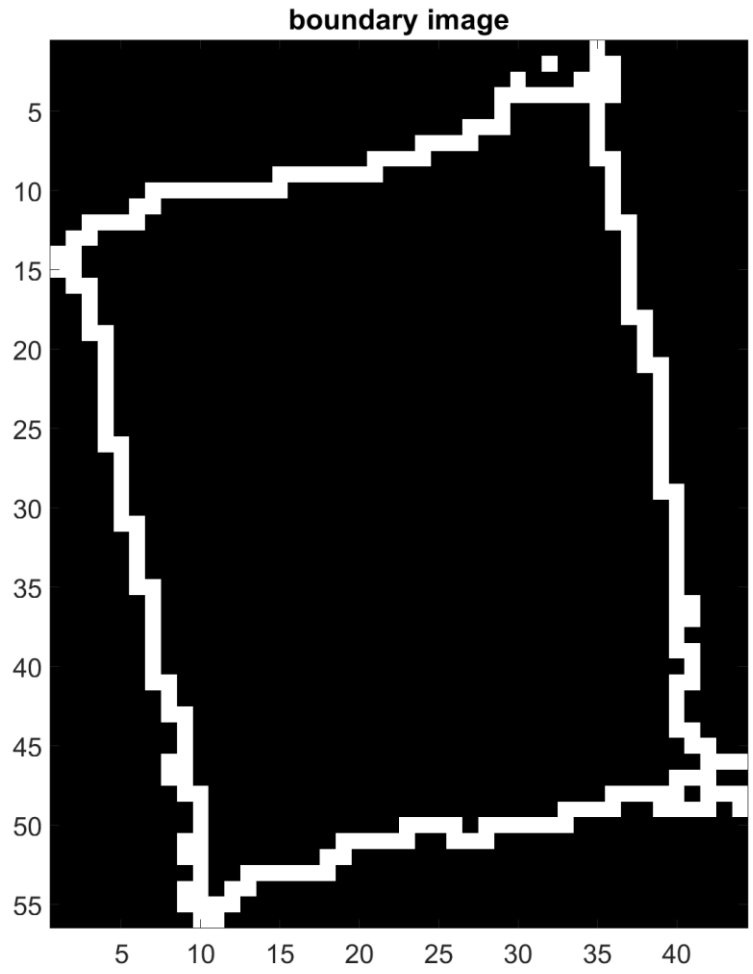


Line Detection: Hough Transform

It is important to find pit

Line Detection: The problem

Finding all the lines passing through points in (a binary) image

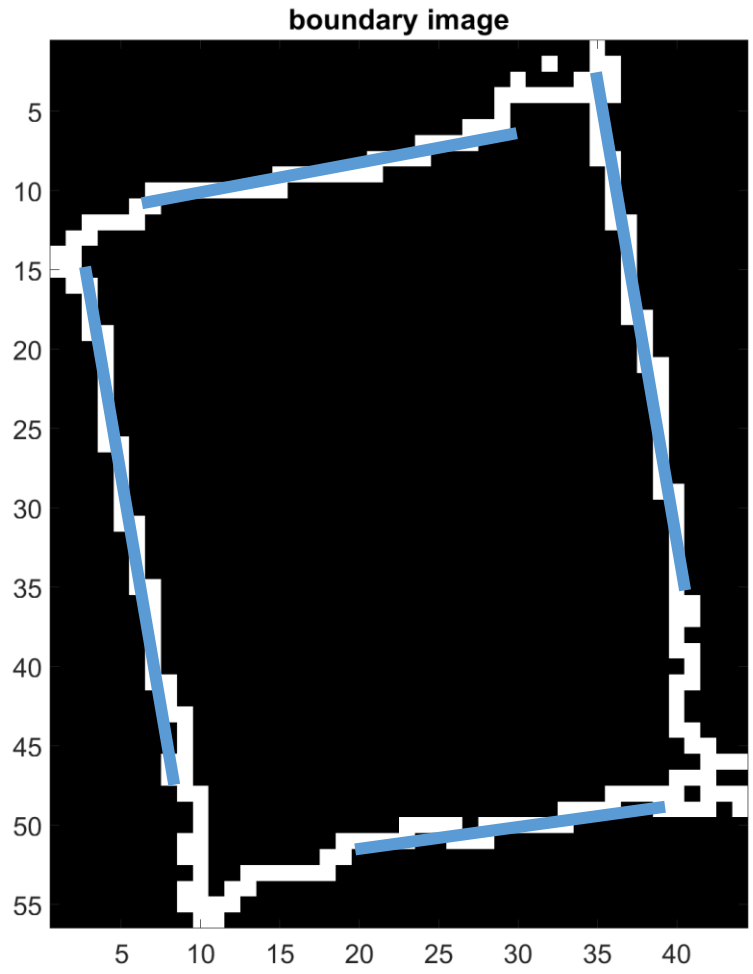


Line Detection: The problem

Finding all the lines passing through points in (a binary) image

Finding lines means

- Having an analytical expression for each line
- Thus estimating its direction



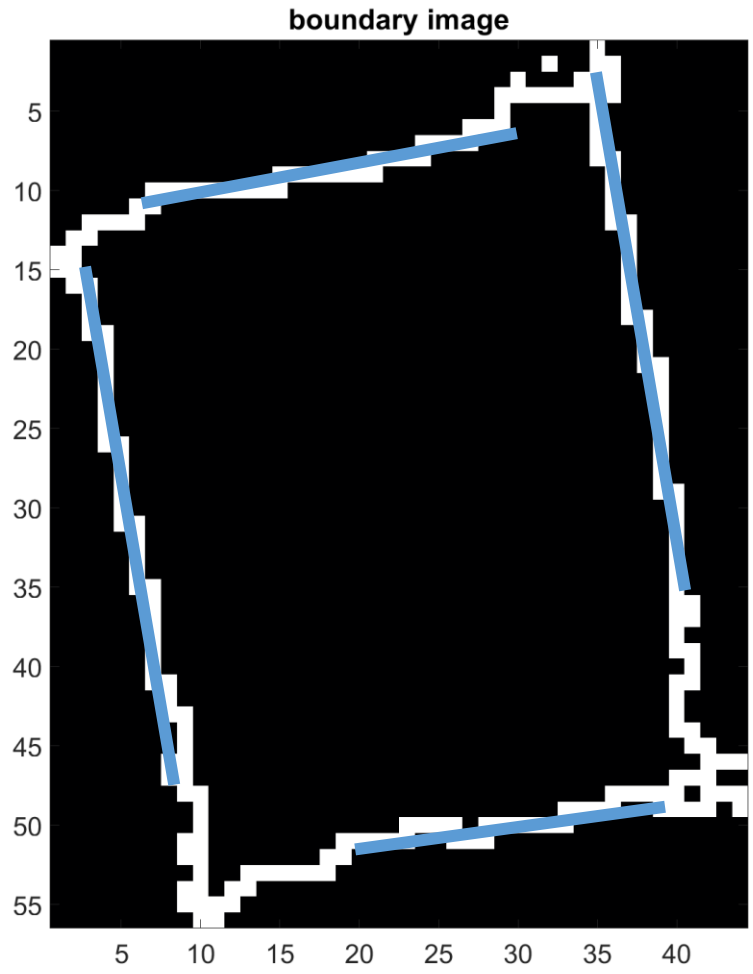
Line Detection: The problem

The Brut-force attempt: Given n points in an image, find subsets of these that lie on straight lines

- Compute all the lines passing through any pair of points
- Check subsets of points that are close to these lines

This requires computing

- $\frac{n(n-1)}{2}$ strainght lines
- $n\left(\frac{n(n-1)}{2}\right)$ comparisons
- Computationally prohibitive task in all but the most trivial applications



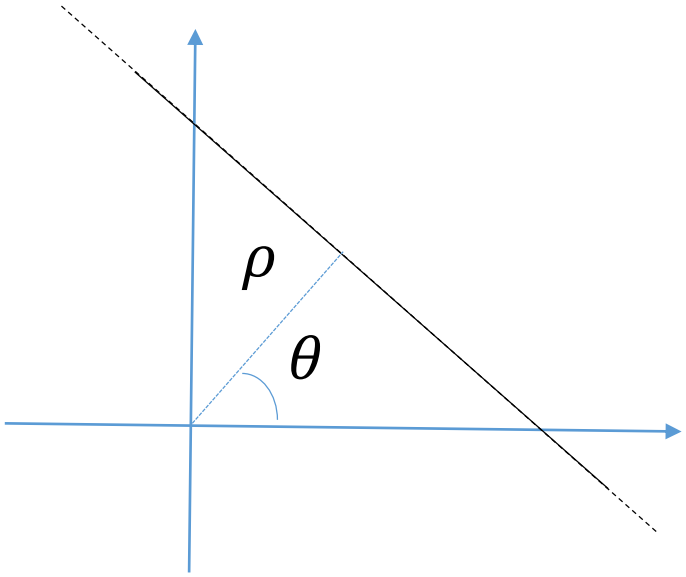
Hough Transform

Identify lines in the “parameter space” i.e. in the space of the parameters identifying lines. These are:

$$\{(\rho, \theta), \rho \in [-L, L], \theta \in \left[-\frac{\pi}{2}, \frac{\pi}{2}\right]\}$$

A line is then represented as:

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



Hough Transform

Identify lines in the “parameter space” i.e. in the space of the parameters identifying lines. These are:

$$\{(\rho, \theta), \rho \in [-L, L], \theta \in \left[-\frac{\pi}{2}, \frac{\pi}{2}\right]\}$$

A line is then represented as:

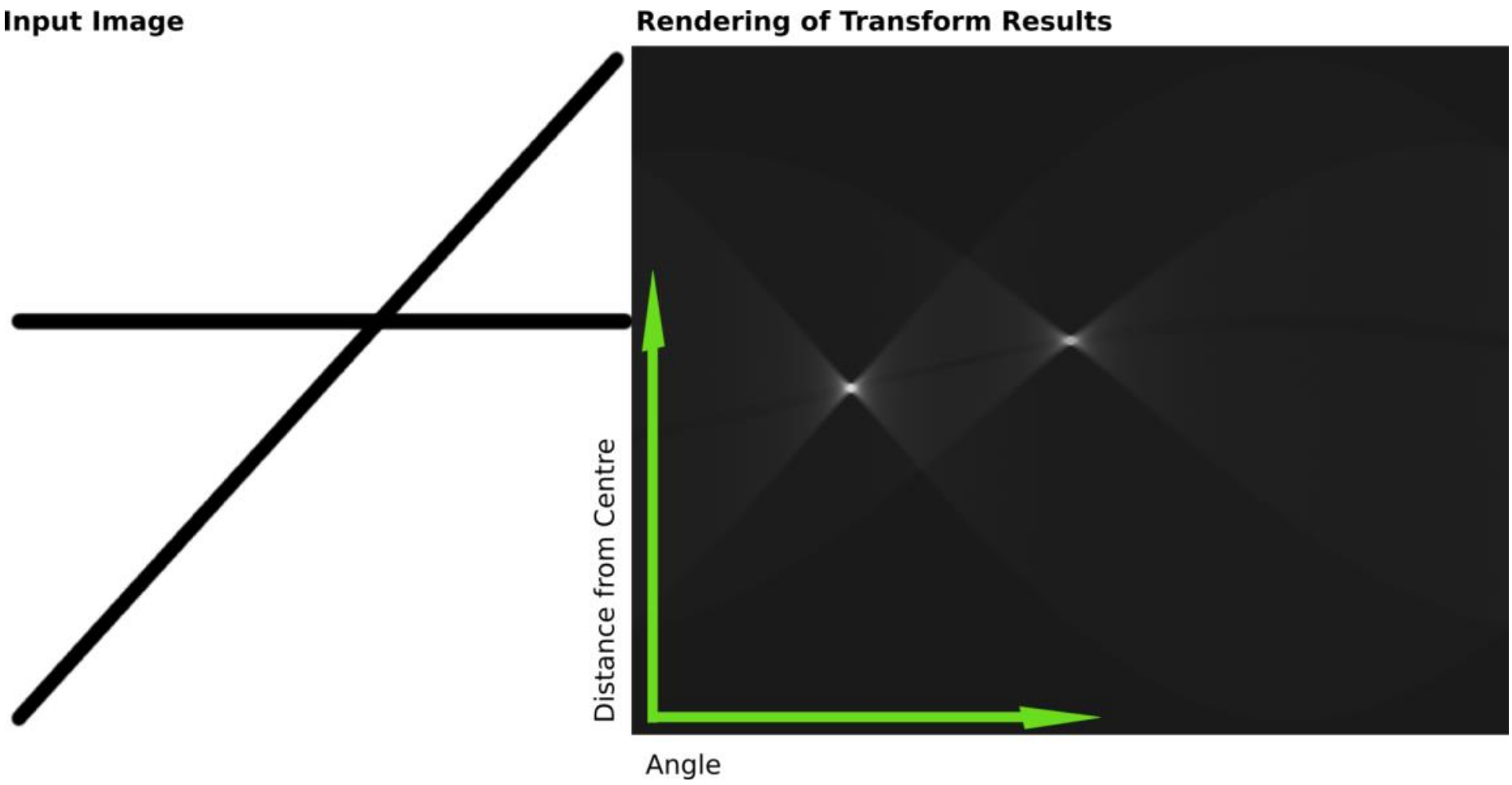
$$x \cos(\theta) + y \sin(\theta) = \rho$$

The Hough transform identifies **through an optimized voting procedure** the most represented lines

The voting procedure is performed in the «accumulator space» which is a grid in (ρ, θ)-domain, for all the possible values.

From the Hough transform we can extract pairs (ρ, θ) corresponding to lines passing through most of points

Hough Transform

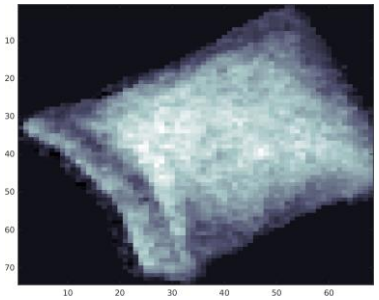


How to use Hand-Crafted Features

D.I.Y the classifier as well?

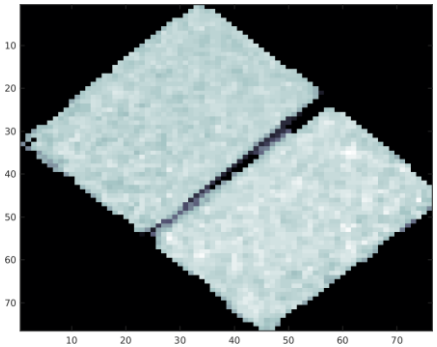
The Feature Extraction Perspective

Input image



$I_1 \in \mathbb{R}^{r_1 \times c_1}$

Input image



$I_1 \in \mathbb{R}^{r_2 \times c_2}$

Feature Extraction Algorithms

$\mathbf{x} \in \mathbb{R}^d$

$\mathbf{x} \in \mathbb{R}^d$

$(d \ll r \times c)$

Classifier

"parcel"

$l \in \Lambda$

"double"

$l \in \Lambda$

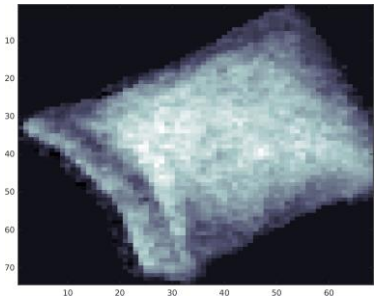
IMPORTANT REMARK

Do not forget to:

- Normalize your features. These might have very different average intensity / range and it is often useful to bring them to 0 mean and standard deviation 1.
- Split your dataset in training and testing, and ideally implement a cross-validation procedure to define the best parameters.
- In binary classification problems, use AUC as a figure of merit to define the optimal threshold for the classifier

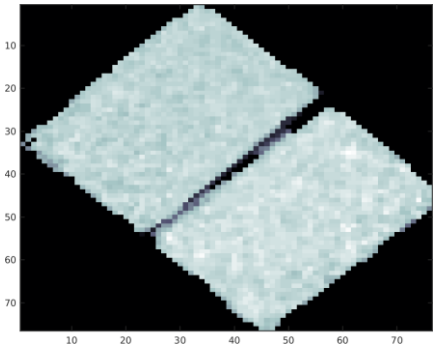
What about D.I.Y solution?

Input image



$I_1 \in \mathbb{R}^{r_1 \times c_1}$

Input image



$I_1 \in \mathbb{R}^{r_2 \times c_2}$

Feature Extraction Algorithms

$\mathbf{x} \in \mathbb{R}^d$

$\mathbf{x} \in \mathbb{R}^d$

$(d \ll r \times c)$

Rule-based Classification

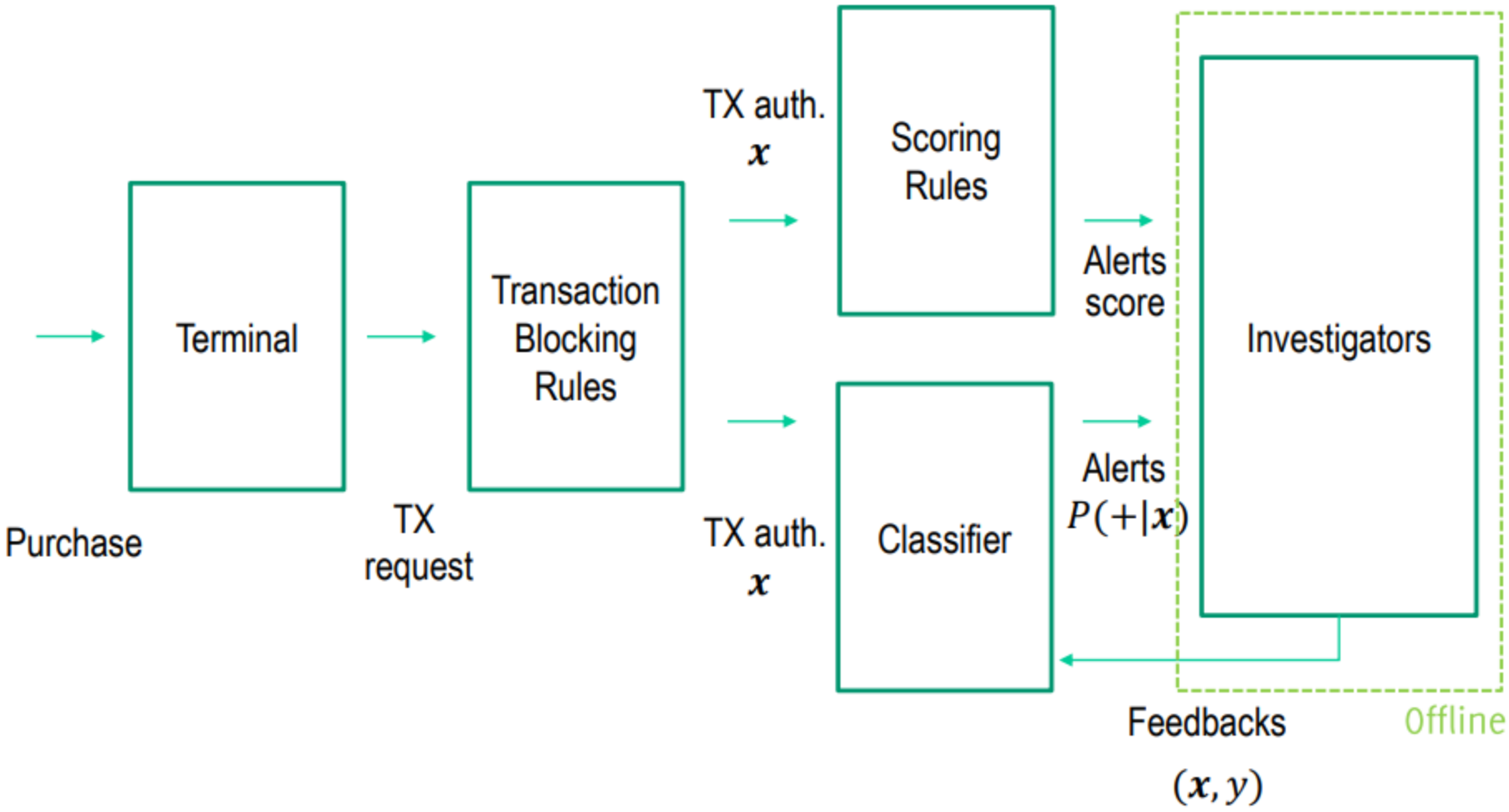
"parcel"

$l \in \Lambda$

"double"

$l \in \Lambda$

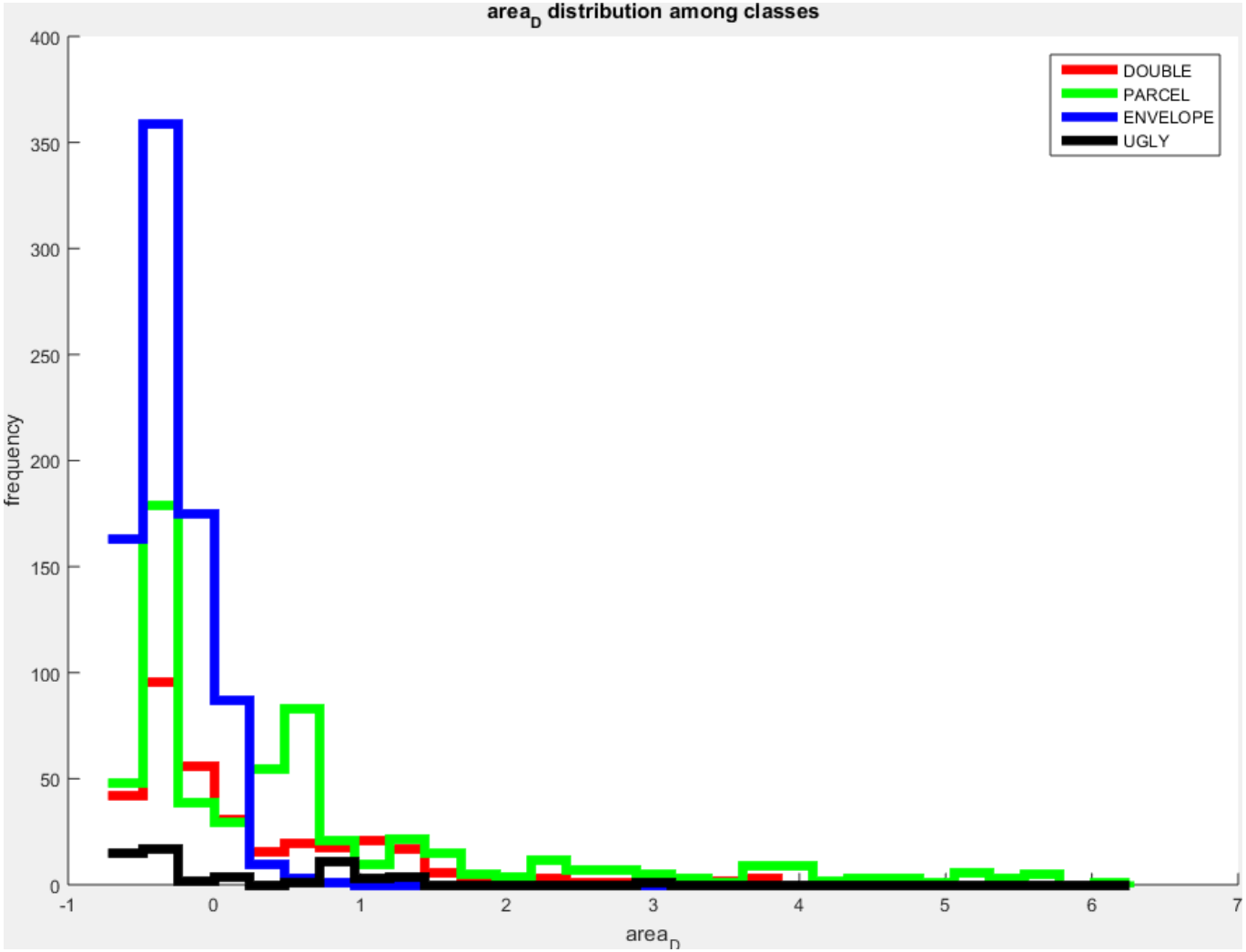
Rule-based classifiers are being used in practical applications



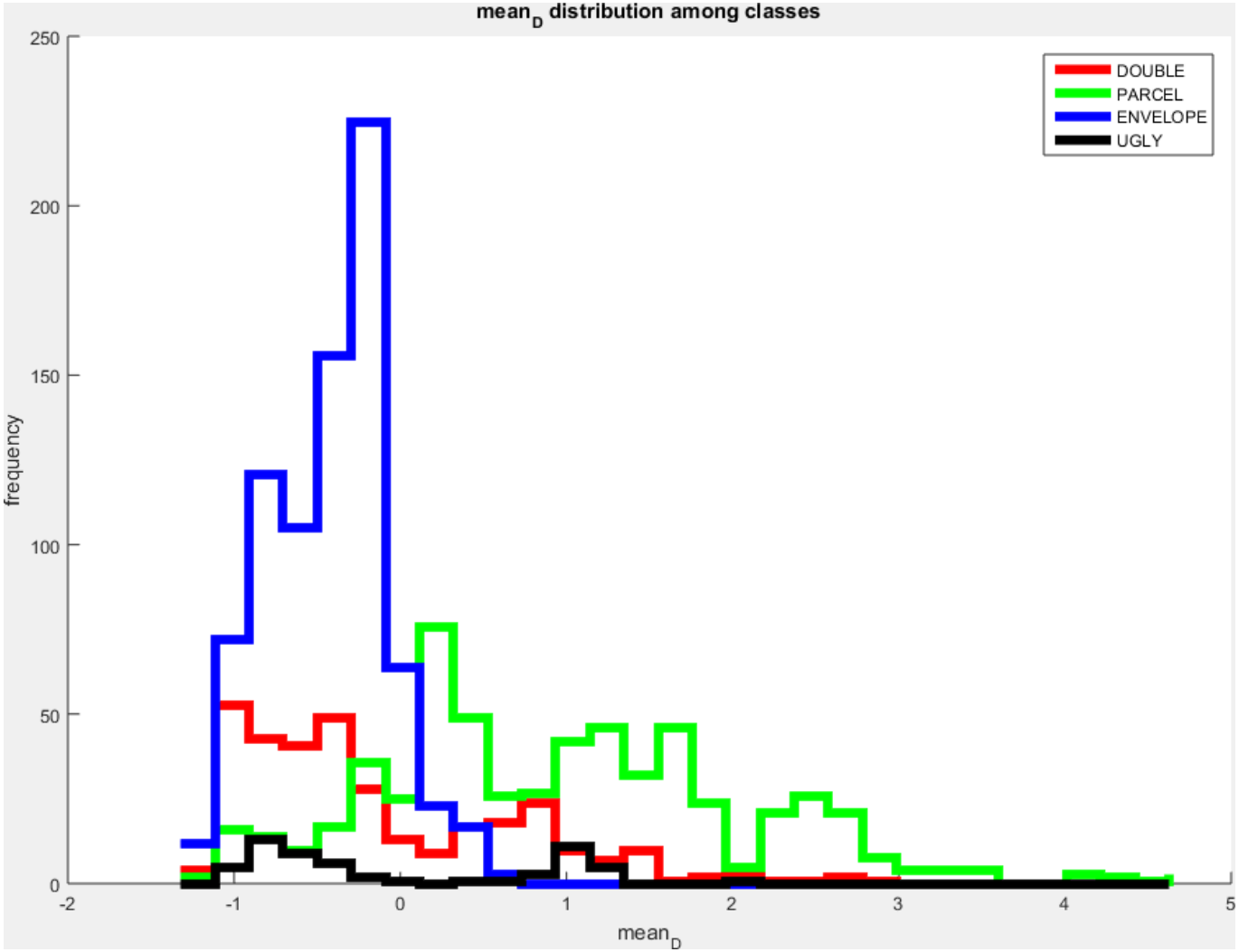
Dal Pozzolo et al. **Credit Card Fraud Detection: a Realistic Modeling and a Novel Learning Strategy**, TNNLS 2017

Distribution of the marginal features

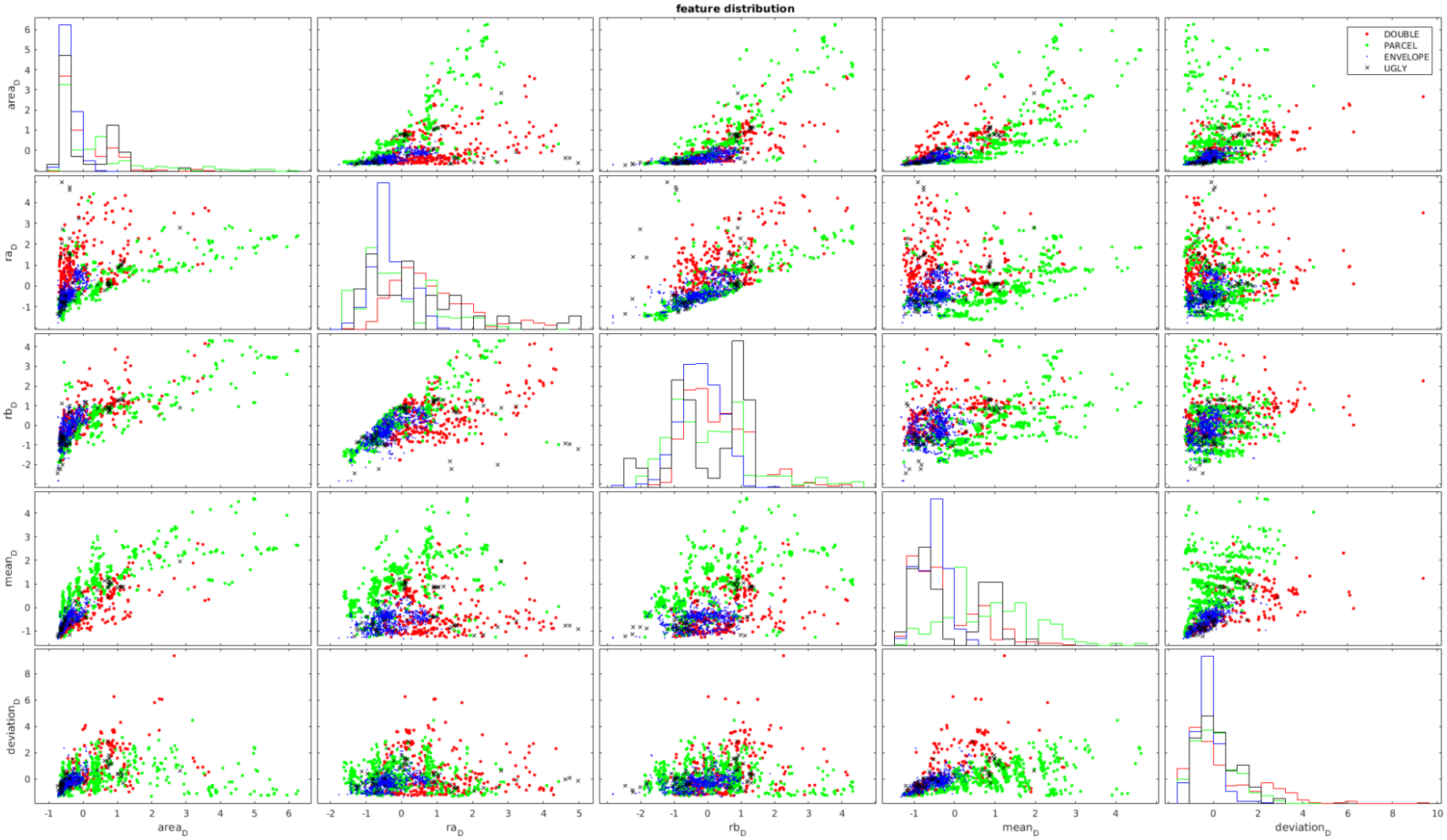
Rule based classification combines a-priori information with exploratory data analysis… with intrinsic limitations



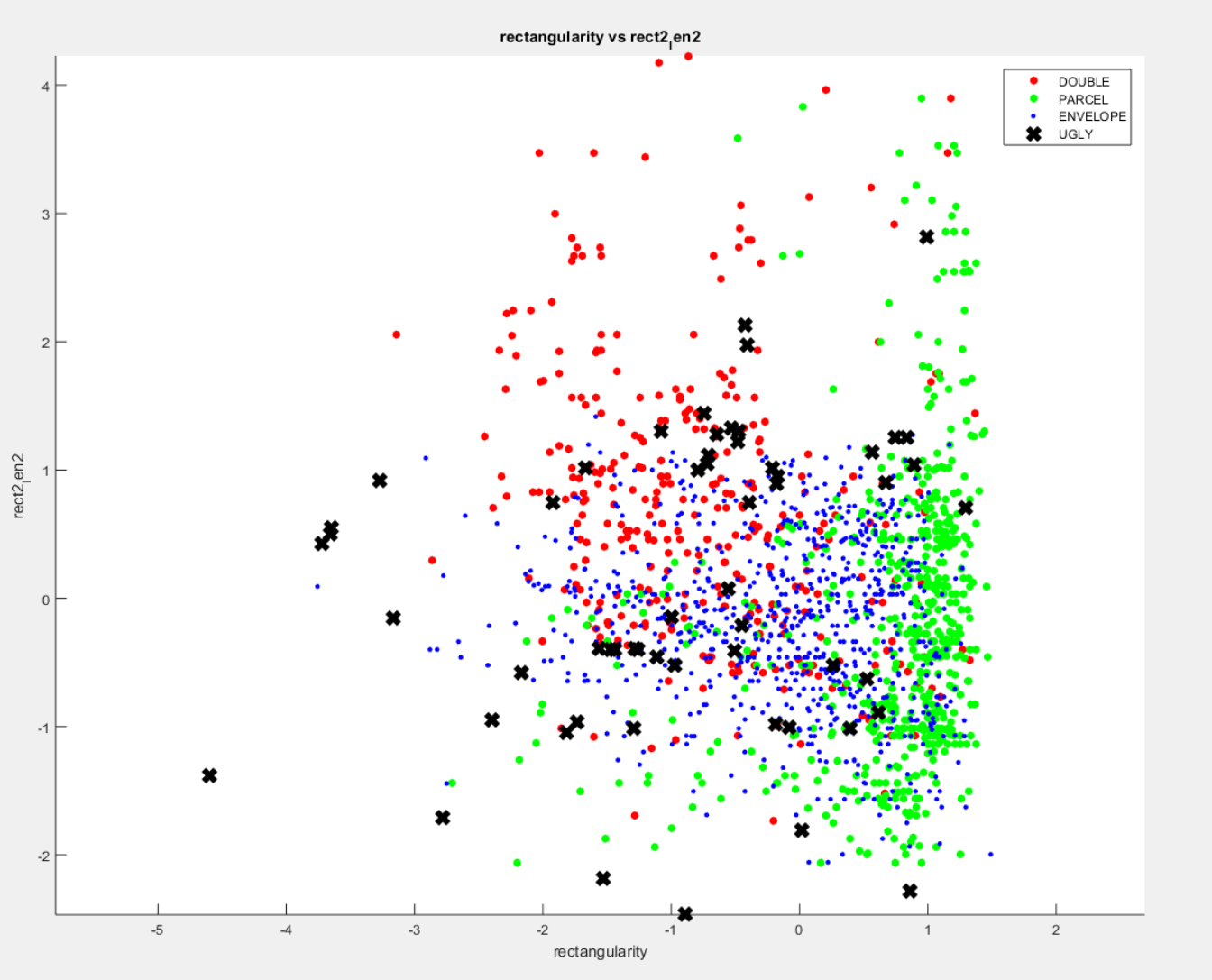
Distribution of the marginal features



Exploratory Data-Analysis



Exploratory Data-Analysis



D.I.Y

Given $x = [f_1, f_2, f_3]$

If $(f_1 > T_1)$

$\hat{y} = 1$

Else

If $(f_1 > T_2)$

$\hat{y} = 2$

Else

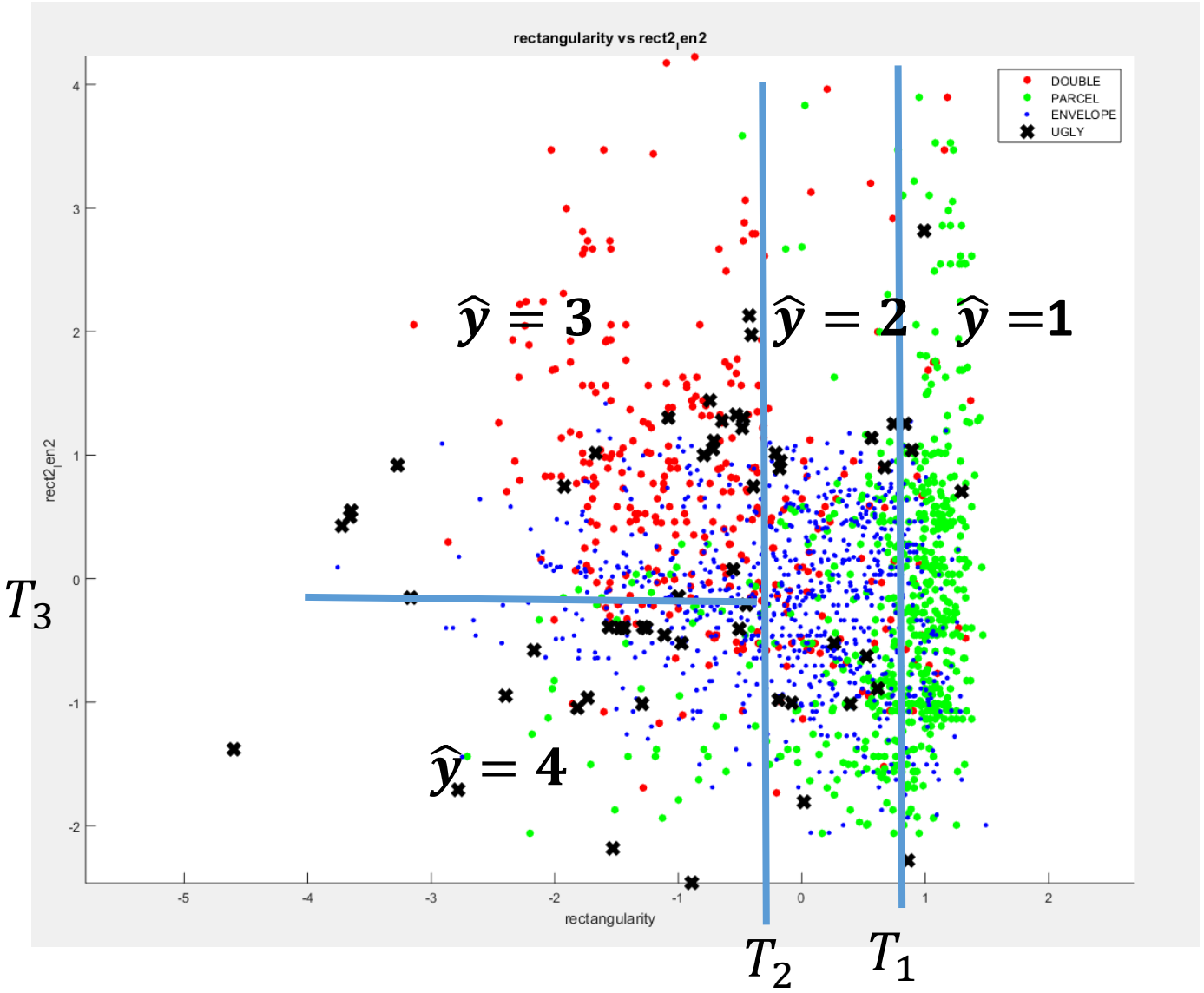
If $(f_2 > T_3)$

$\hat{y} = 4$

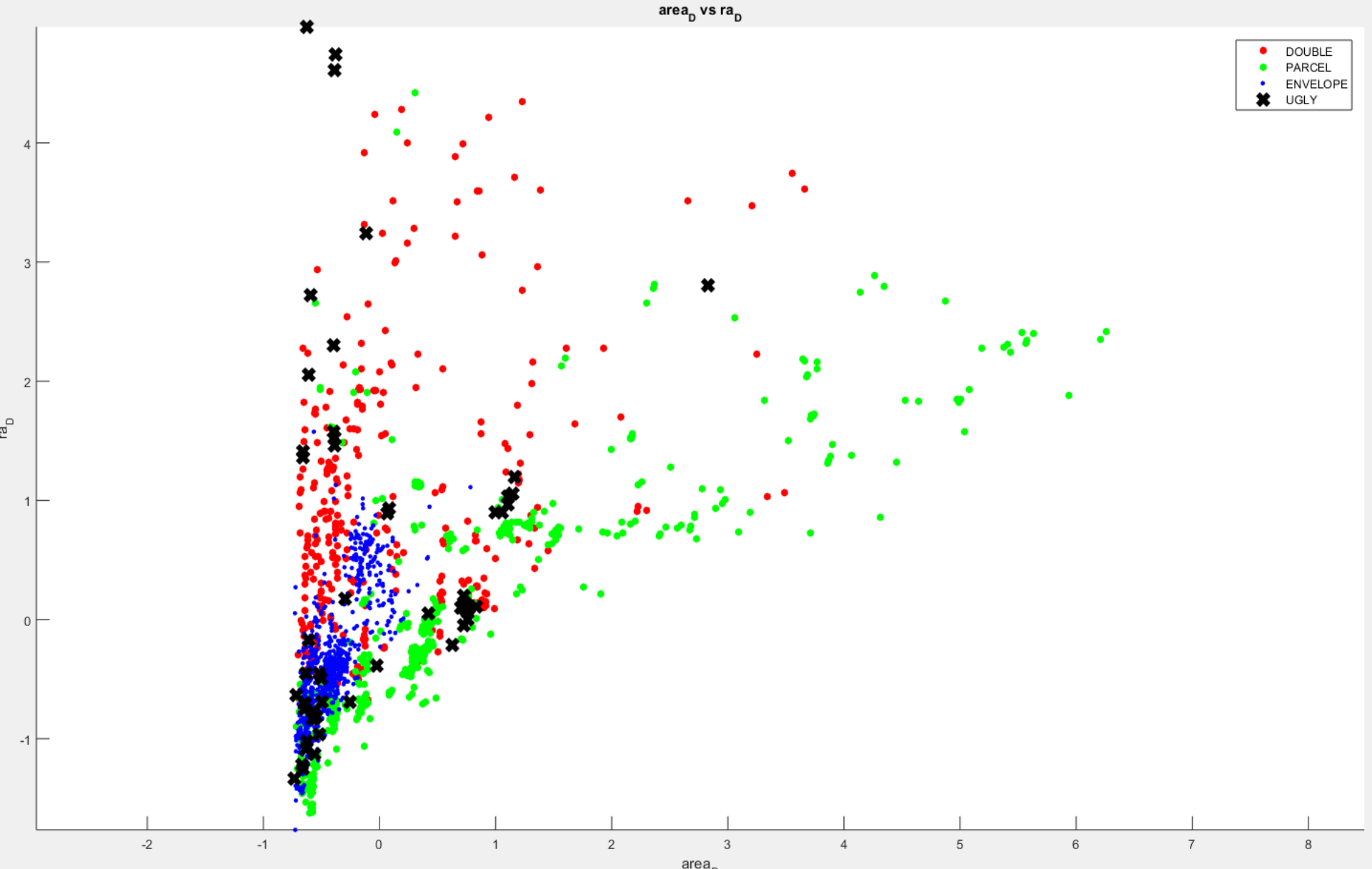
Else

$\hat{y} = 3$

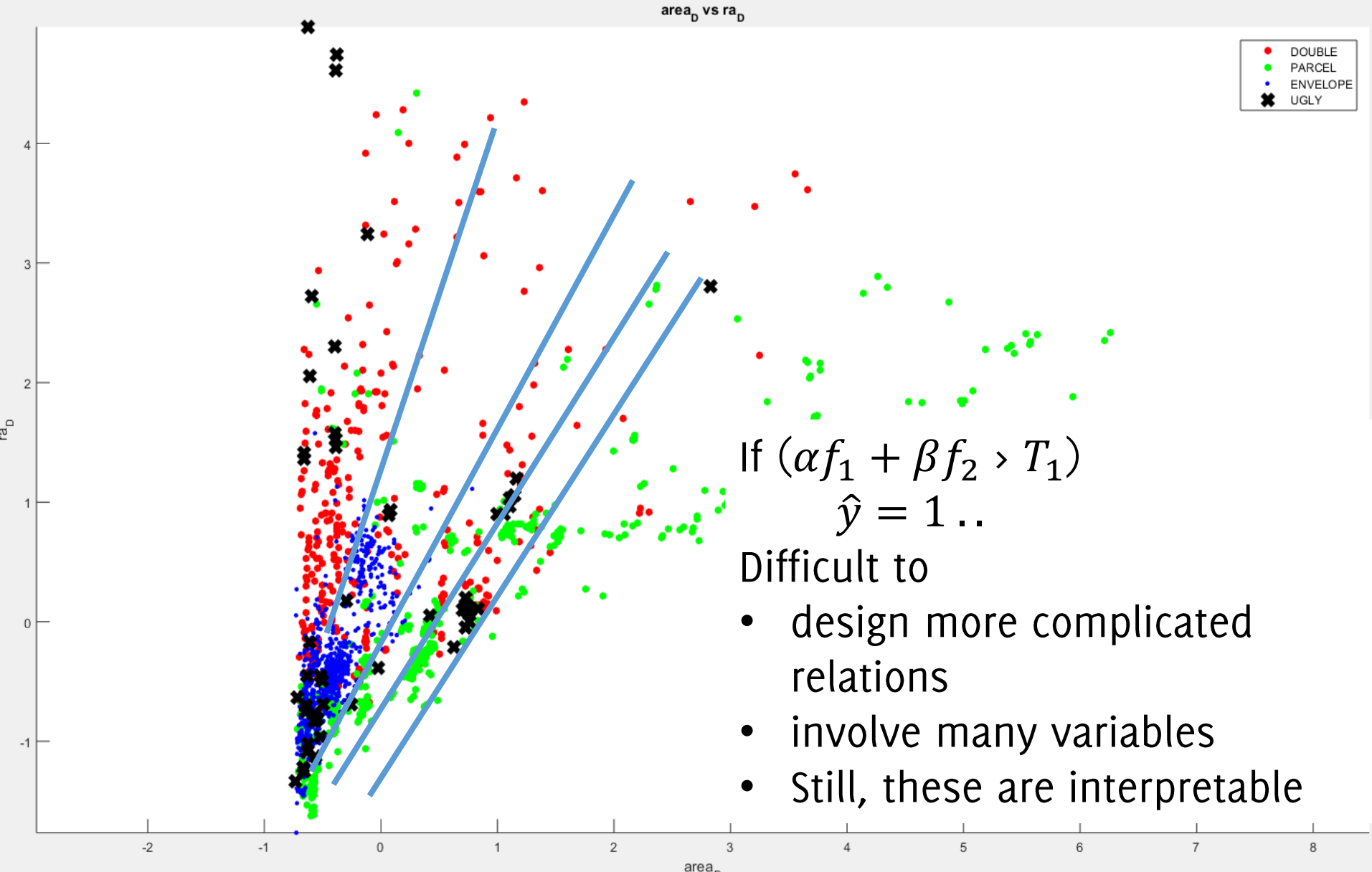
What a rule-based classifier does?



Limitations of rule-based classification



These separation lines are difficult to design by hand



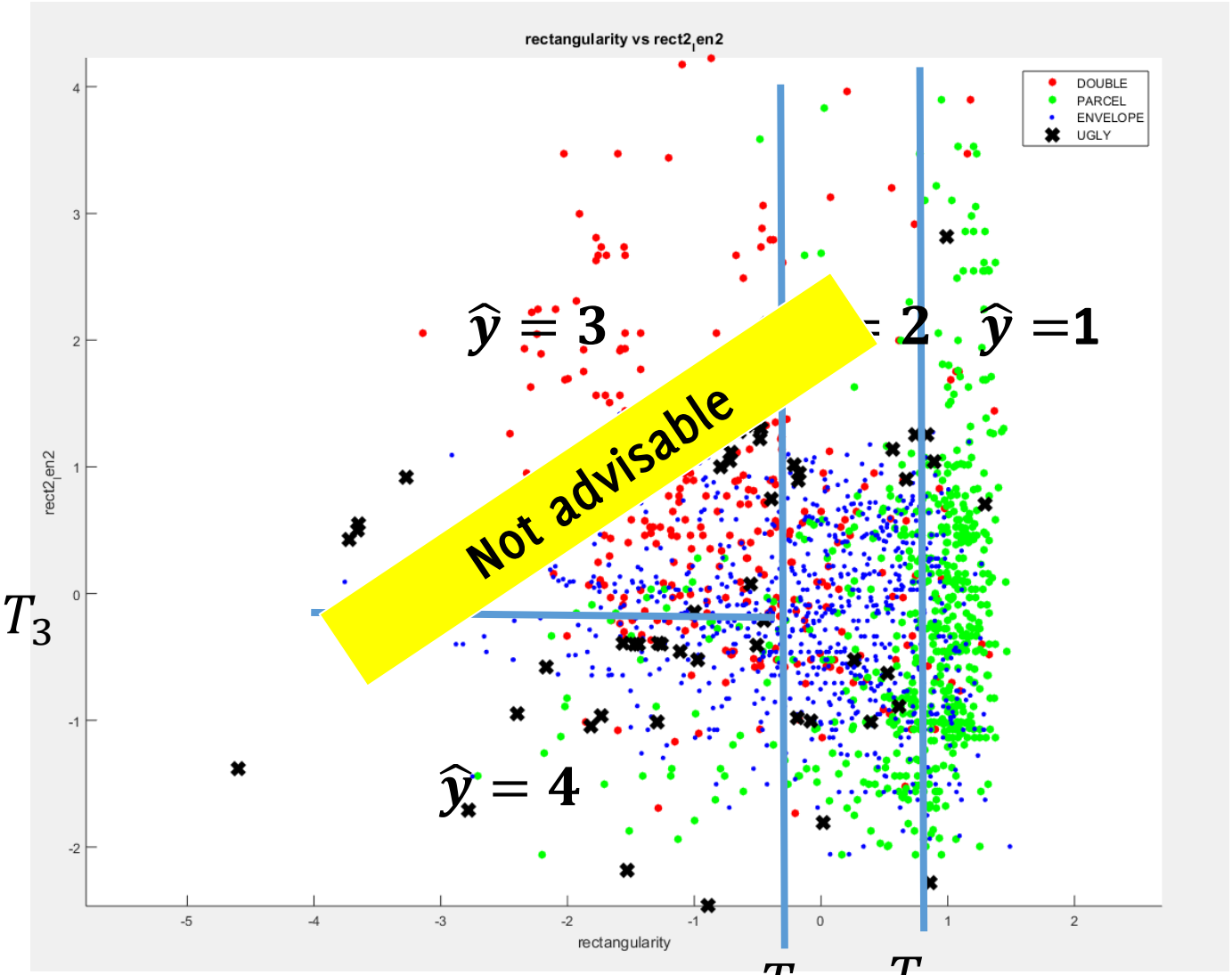
If $(\alpha f_1 + \beta f_2 \rangle T_1)$

$\hat{y} = 1$ ..

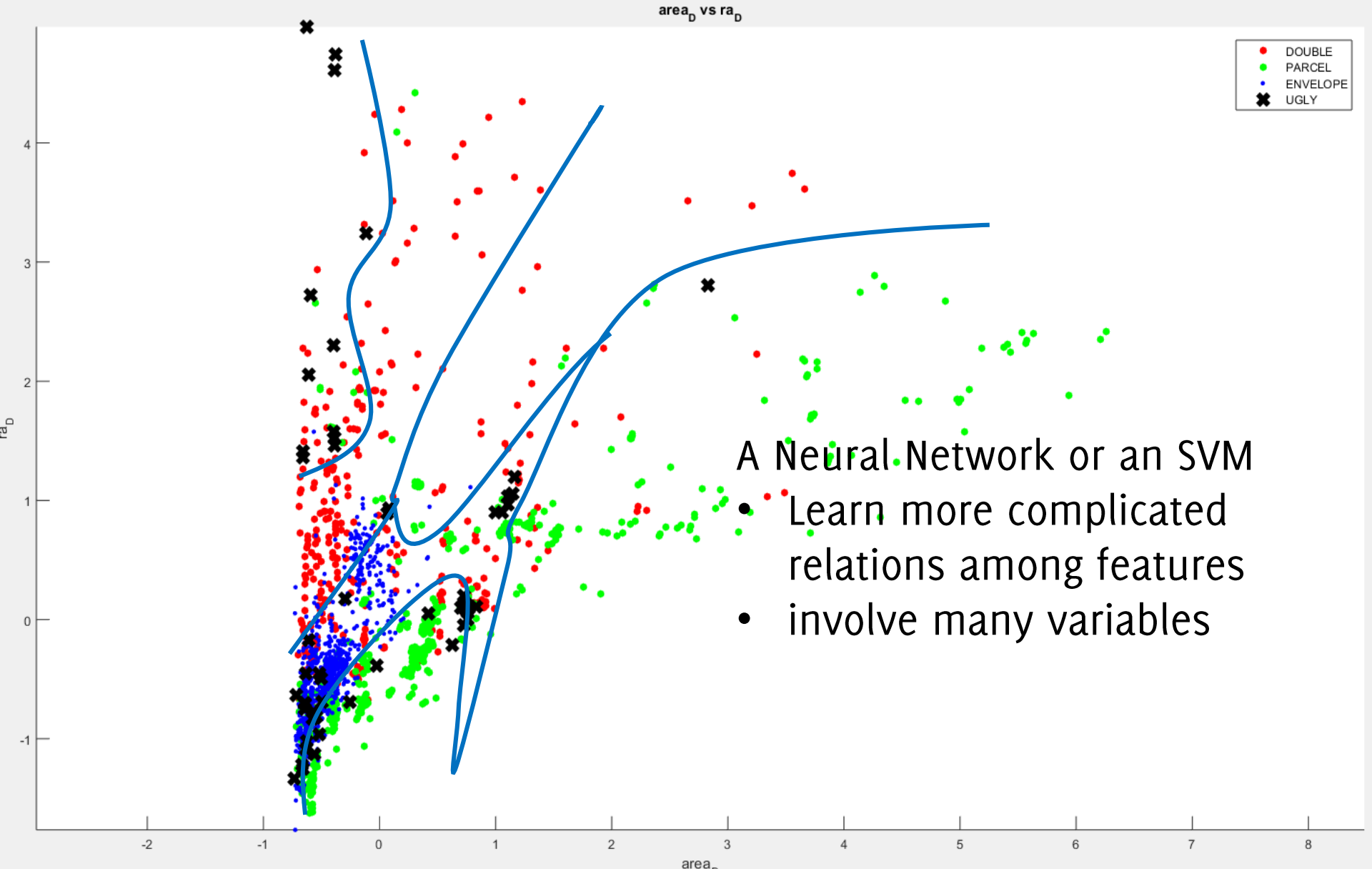
Difficult to

- design more complicated relations
- involve many variables
- Still, these are interpretable

What a rule-based classifier does?



A learned, non-linear classifier might



A Neural Network or an SVM

- Learn more complicated relations among features
- involve many variables