P6 - Contours

6.1 Intro

A contour is a list of points that are connected, ordered to follow the shape.

Representation types:

- Explicit $y = f(x)$
- Implicit $f(x, y) = 0$
- Parametric $p(u) = (x(u), y(u))$

Concepts:

- Tangent unitary vector:

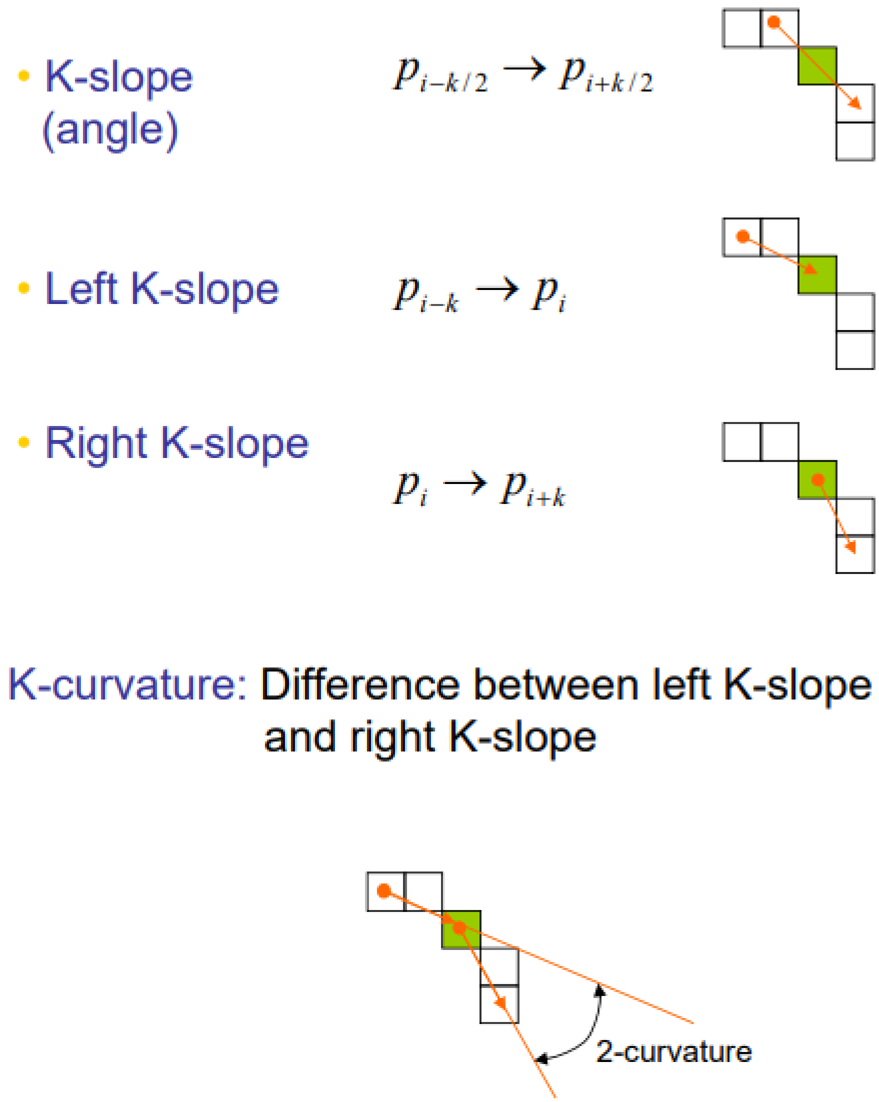
$$t(u) = \frac{p'(u)}{|p'(u)|}$$

- Normal vector, it is perpendicular to the tangent unitary vector, and has a magnitude equal to the curvature radius:

$$n(u) = p''(u)$$

- Curvature = $k = \frac{1}{r}$
- Curve length = $\int_{u1}^{u2} (\frac{dx}{du})^2 + (\frac{dy}{du})^2$

6.1.1 Digital curves

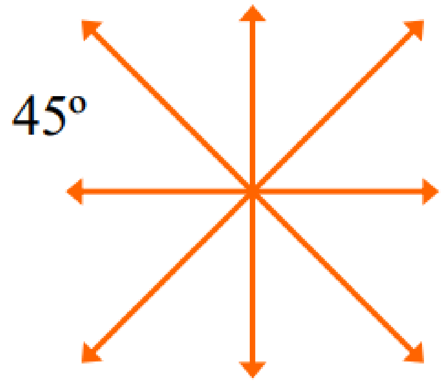


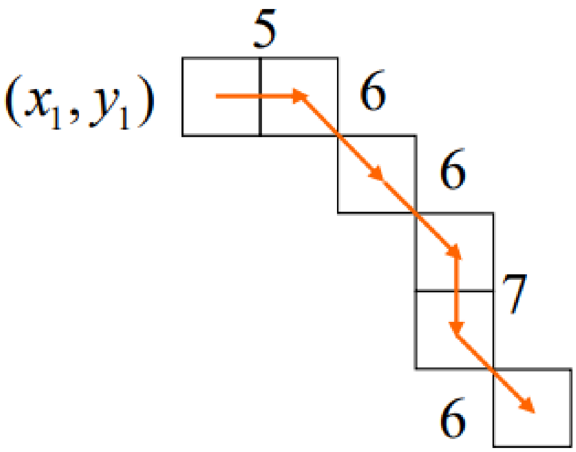
- Contour length:$$
  S = \sum*{i=2}^n\sqrt{(x_i - x*{i-1}) + (y*i - y*{i-1})}

![[Pastedimage20241007154201.png]]

6.2 Chain code

2	3	4
1	·	5
8	7	6





$$(x_1, y_1) \rightarrow 5, 6, 6, 7, 6$$

The **problem** with this is that the possible angles are all multiples of 45°; this is solved by methods of curve fitting:

6.2.1 Curve fitting:

- **Interpolation**: Method that allows the curve to go through all real points
- **Approximation**: Curve is approximated to regular curves, but as a result, curve does not have to pass through any point (if it does, it will be by pure casuality)
- **Mixed**: Passes through some and it is approximated on others

IF ALL POINTS ARE CONSIDERED VALID, the curve model can be based on:

- Linear segments
- Regular curves (such as circular arcs, conic sections, cubic "*splines*")

AS FOR OUTLIERS, we can remove them by:

- Least median squares
- *Ransac*

ERROR

- Euclidean distance from each point to the curve.

$$d_i(type : vector)$$

- Maximum Absolute Error

$$MAE = max(|d_i|)$$

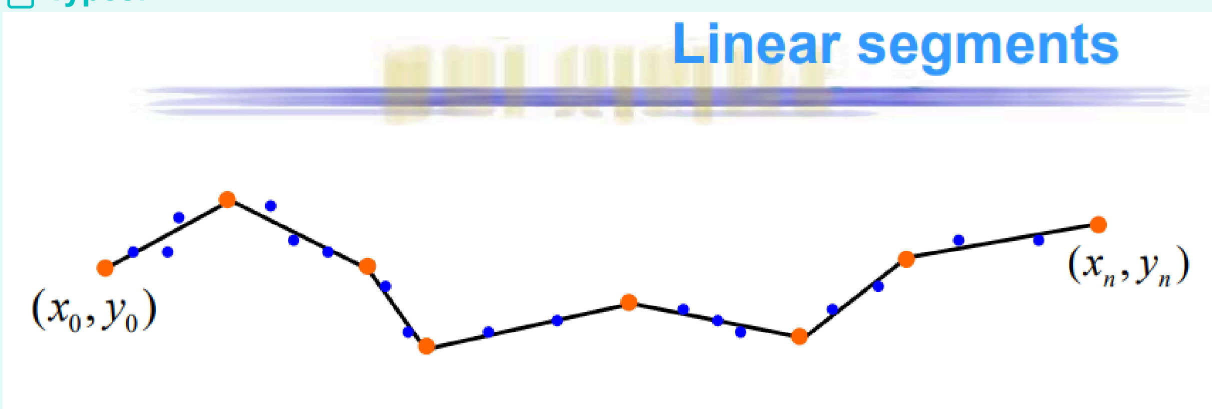
- Normalized Maximum Error:

$$\epsilon = \frac{max(|d_i|)}{S}$$

- Mean Square Error:

$$\frac{1}{n}\sum_{i=1}^{n} d_i^2$$

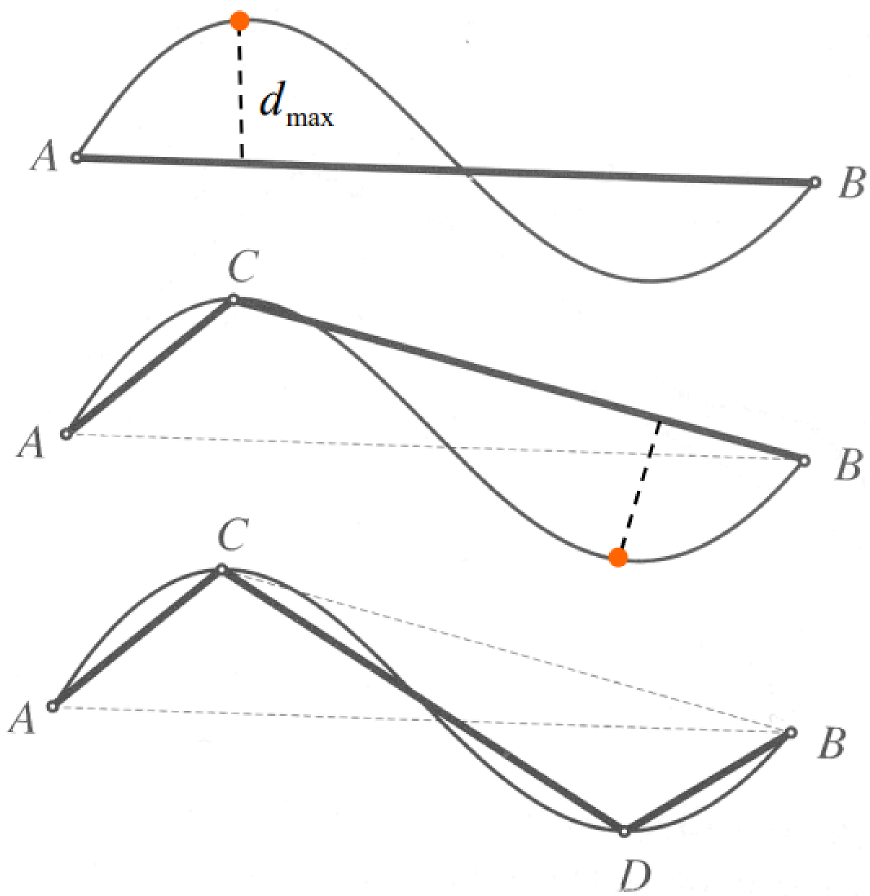
Types:



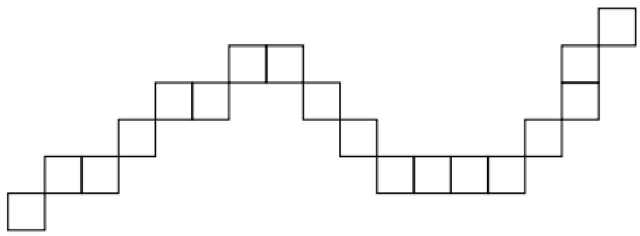
For two vertices

$$\frac{y - y_1}{y_2 - y_1} = \frac{x - x_1}{x_2 - x_1}$$

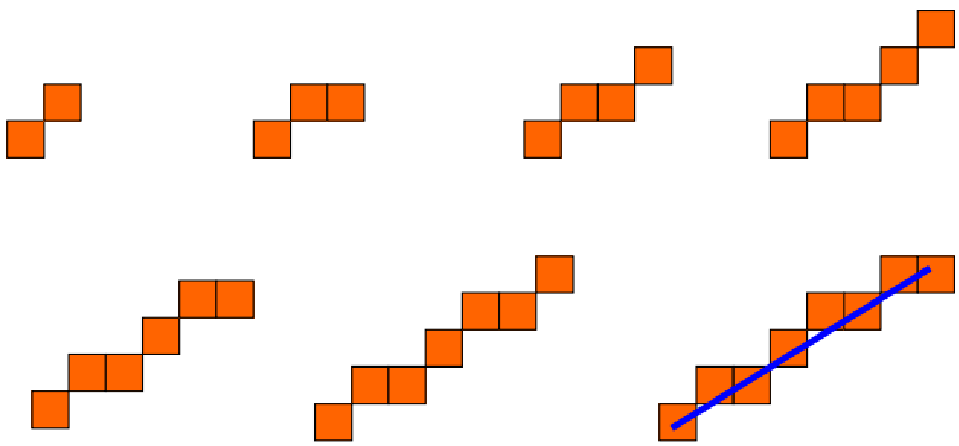
Vertices selection (Top-down)



Vertices selection (Bottom-up)



Add píxels to the segment while fitting error is less than a threshold

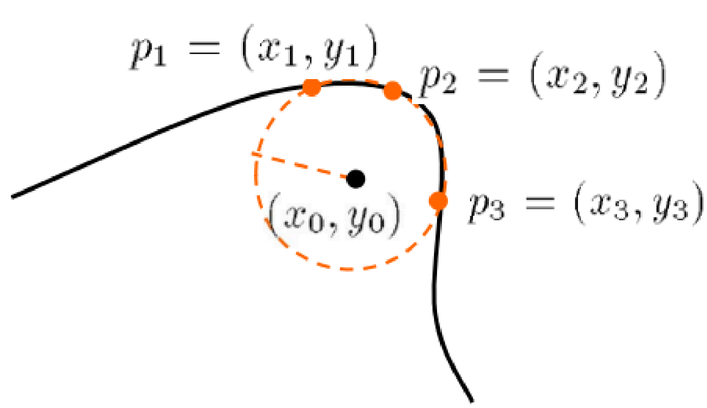


Split & merge

- Algorithm:
  - Top-down division
  - Union of adjacent segments using the “normalized error” measure
  - Repeat until no changes
- Observations:
  - After a union step, a segment could be divided at a different point
  - The use of normalized error in the union step allows merging into a single segment

Circular arcs

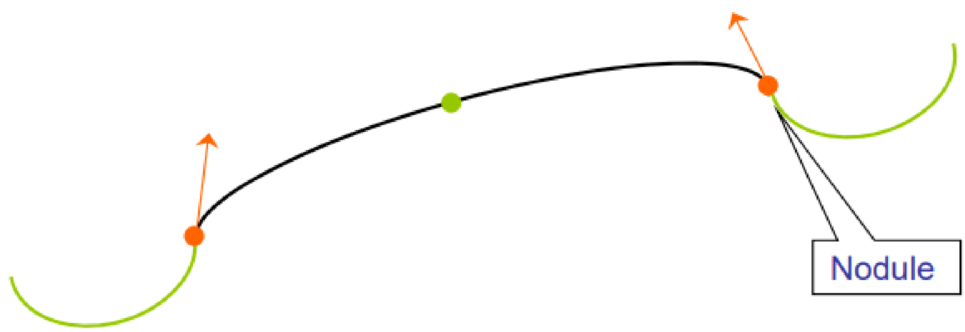
Circumference: $(x - x_0)^2 + (y - y_0)^2 = r^2$



Same points as in the lineal example (orange) can be used in groups of 3 to make circular arcs

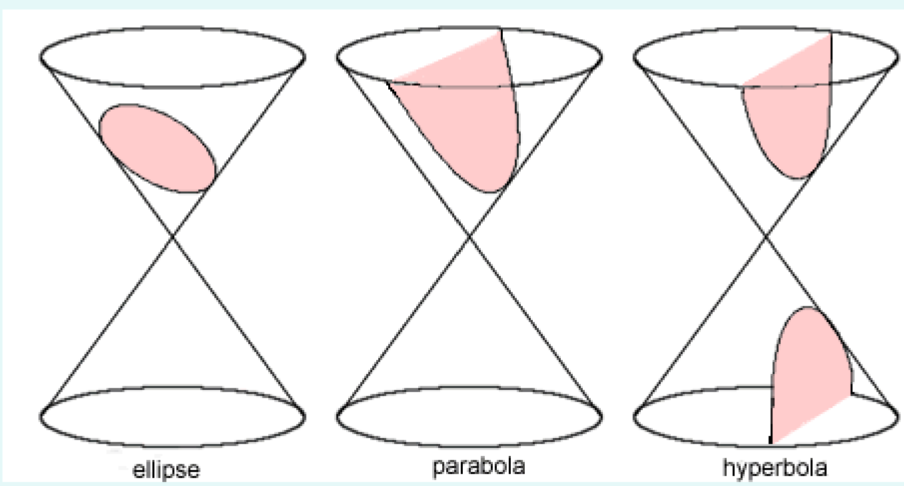
Conic sections

$$f(x,y) = ax^2 + 2hxy + by^2 + 2ex + 2gy + c = 0$$

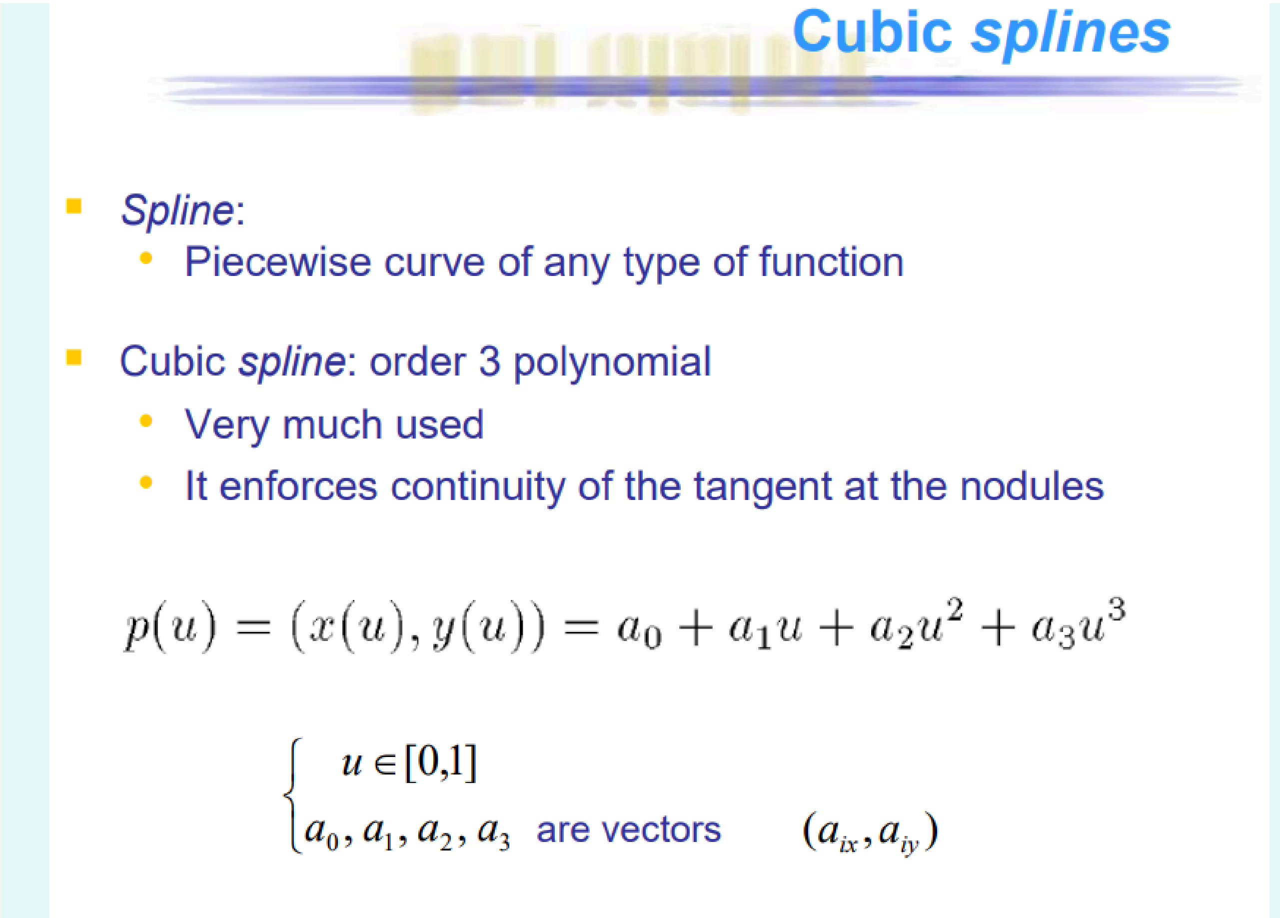


The advantage of a conic curve over a simple cicular arc, is that we can force the defining vector at a point from one side to be the same as for the other side (as in the picture above), so the contour can be continuous.

Possible Conic curves:



- Circle
- Parabola
- Hiperbola
- Ellipse



Any order polinomic

6.3 Regression

Previous steps:

- Define a model
- Define error measurement
- Solution ➪ error minimization

6.3.1 Robust regression

Least-squares adjustment

WIKIPEDIA

① Caluclate MSE

$$F = MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - (ax_i + b))$$

② Knowing that we want to get the minimum value for F, let's find the values of a, b that minimize the value of F, by making a system of equations like:

$$\theta = (a,b) = \begin{cases} \frac{\partial F}{\partial a} = 0 \\ \frac{\partial F}{\partial b} = 0 \end{cases}$$

③ Knowing a, b, we have the line that better approximates the points like:

$$y = ax + b$$

6.3.2 *RanSac* (Random sample consensus)

Selection of random points, multiple times, to avoid the influence of outliers.

PROCESS:

① Chose **random subset of points** ($len < n$; where n is the size of the whole set)

② By defining a maximum error, check **how many points** have an *error* < *defined error* -> Estimating error using *K-Inliers*

③ **Repeat until K is big** enough

④ (*Optional*) **Recompute the curve** using all *K-inliers* that fit inside the best adjust

NOTES:

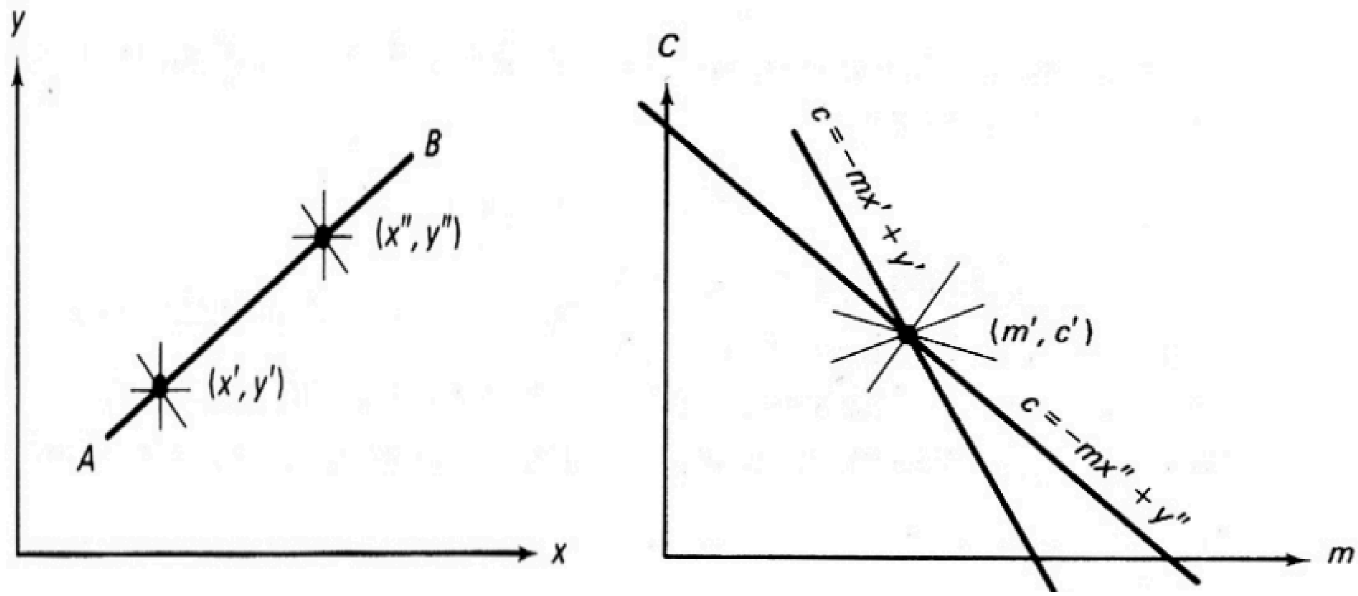
- How many points for each subset? -> minimum 2 -> The more the better but has to represent a small part of the set

6.3.2.1 *Hough* transform:

Hough Transform

$$y = mx + c$$

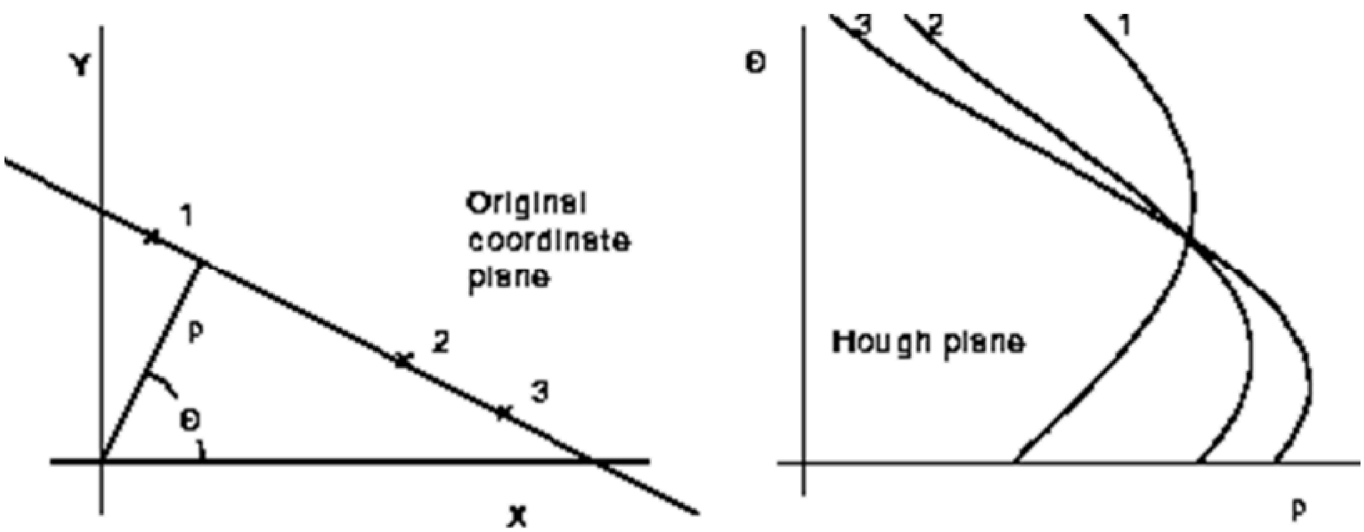
$$c = -mx + y$$



Problem: $m \in [-\infty, \infty]$

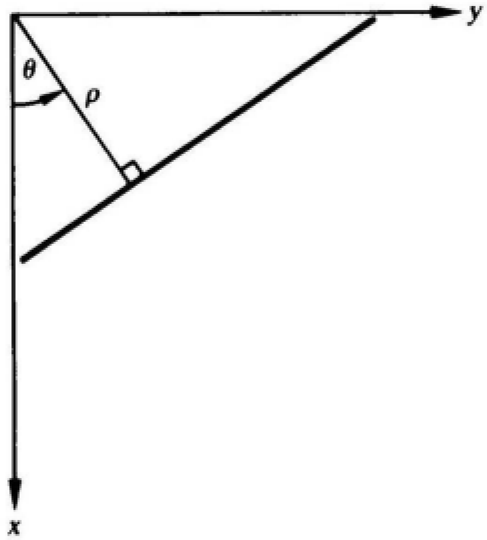
Solution: use normal form of a line equation

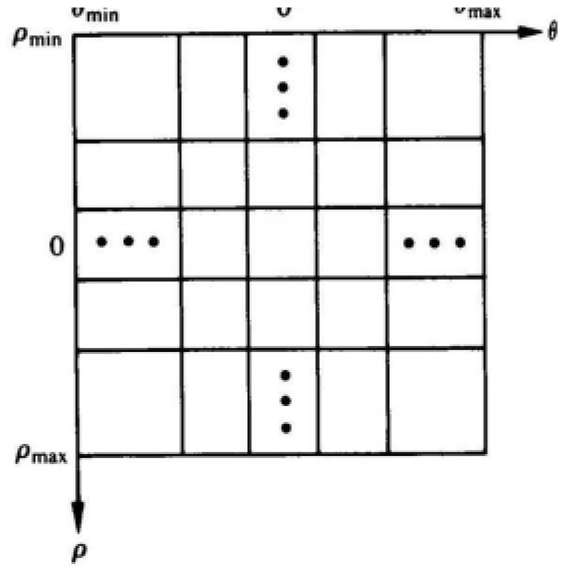
$$x\cos\theta + y\sin\theta = \rho$$



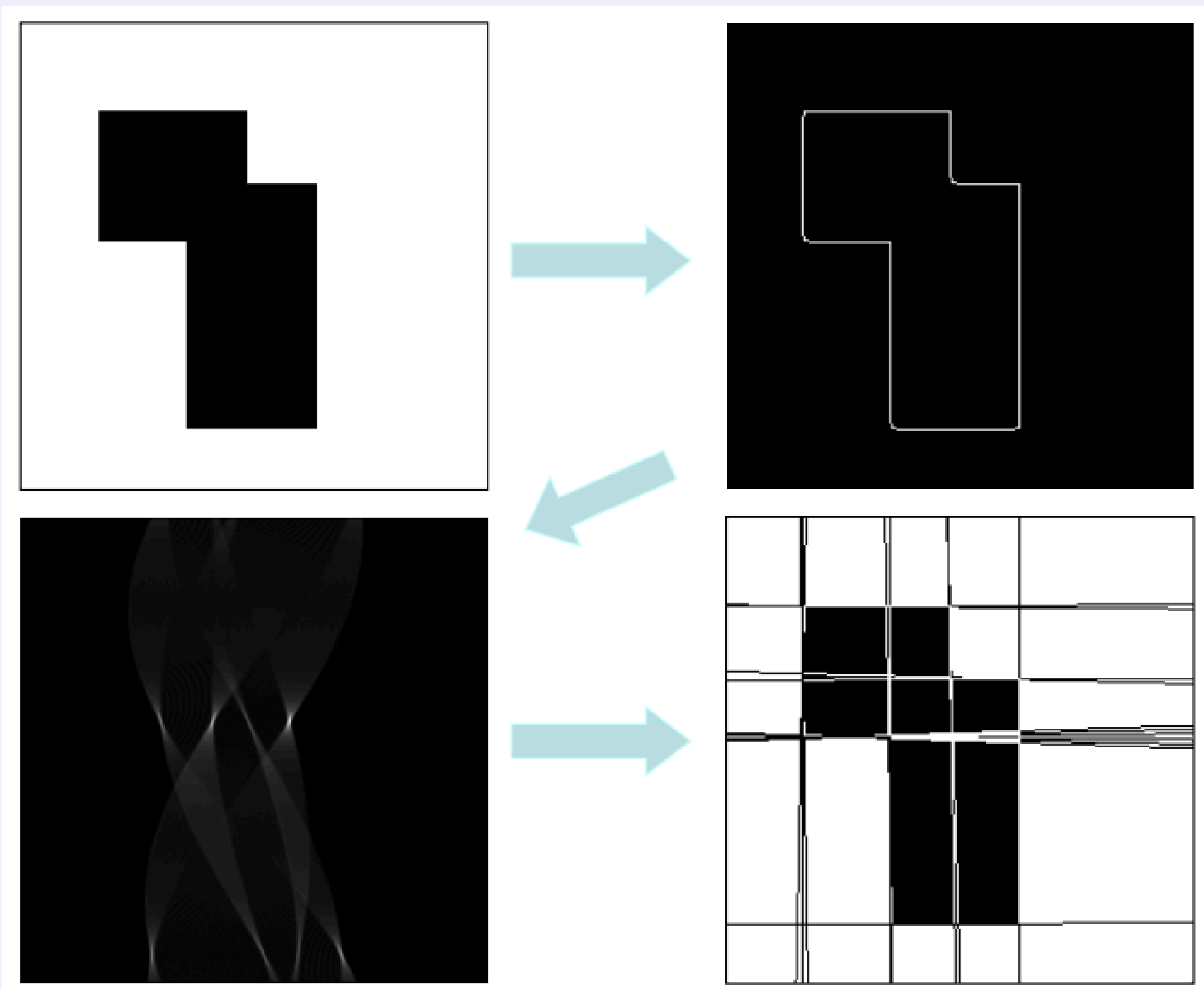
DISCRETIZATION:

- Discretize both ρ and θ
- Consider each cell as an accumulator; each line increases the value of a cell by one unit
- The cell accumulators with maximum values define the parameters of the model.





Simple example



① First, by means of a edge detection algorithm, we detect the edges

② Then, represent the ρ, θ diagram, and we can roughly see 8 local maxima (some less intense), that represent each line that forms the shape

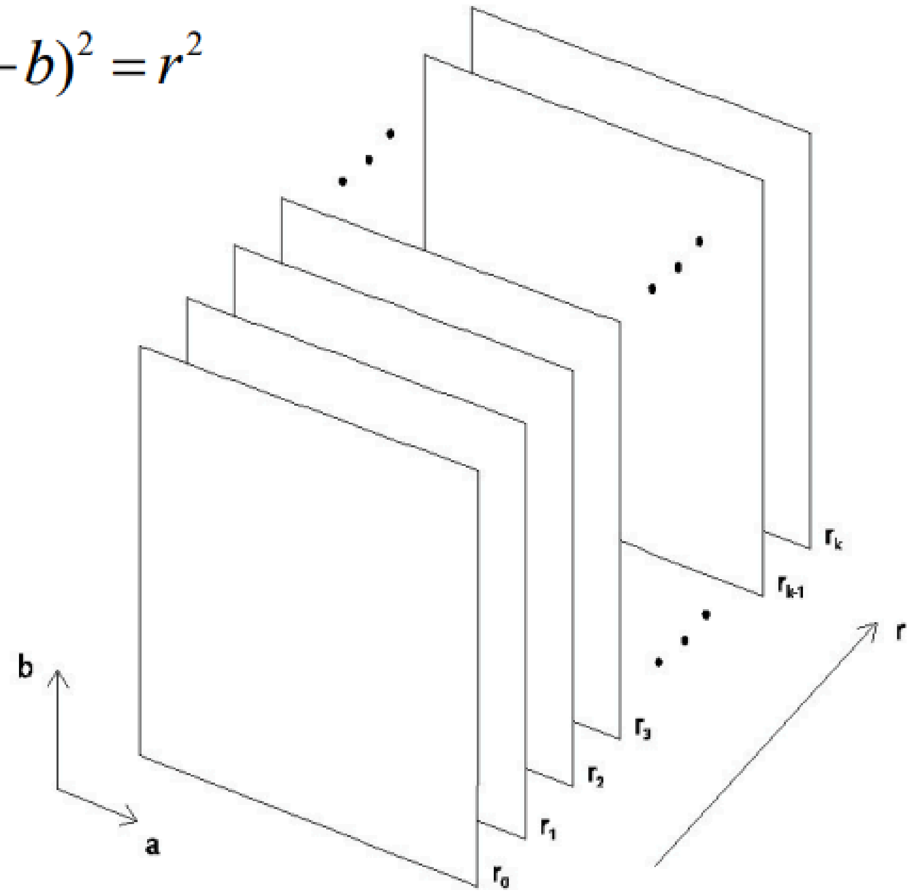
③ We could also store which point "*voted*" for each intersection point to know the points that belong to each line

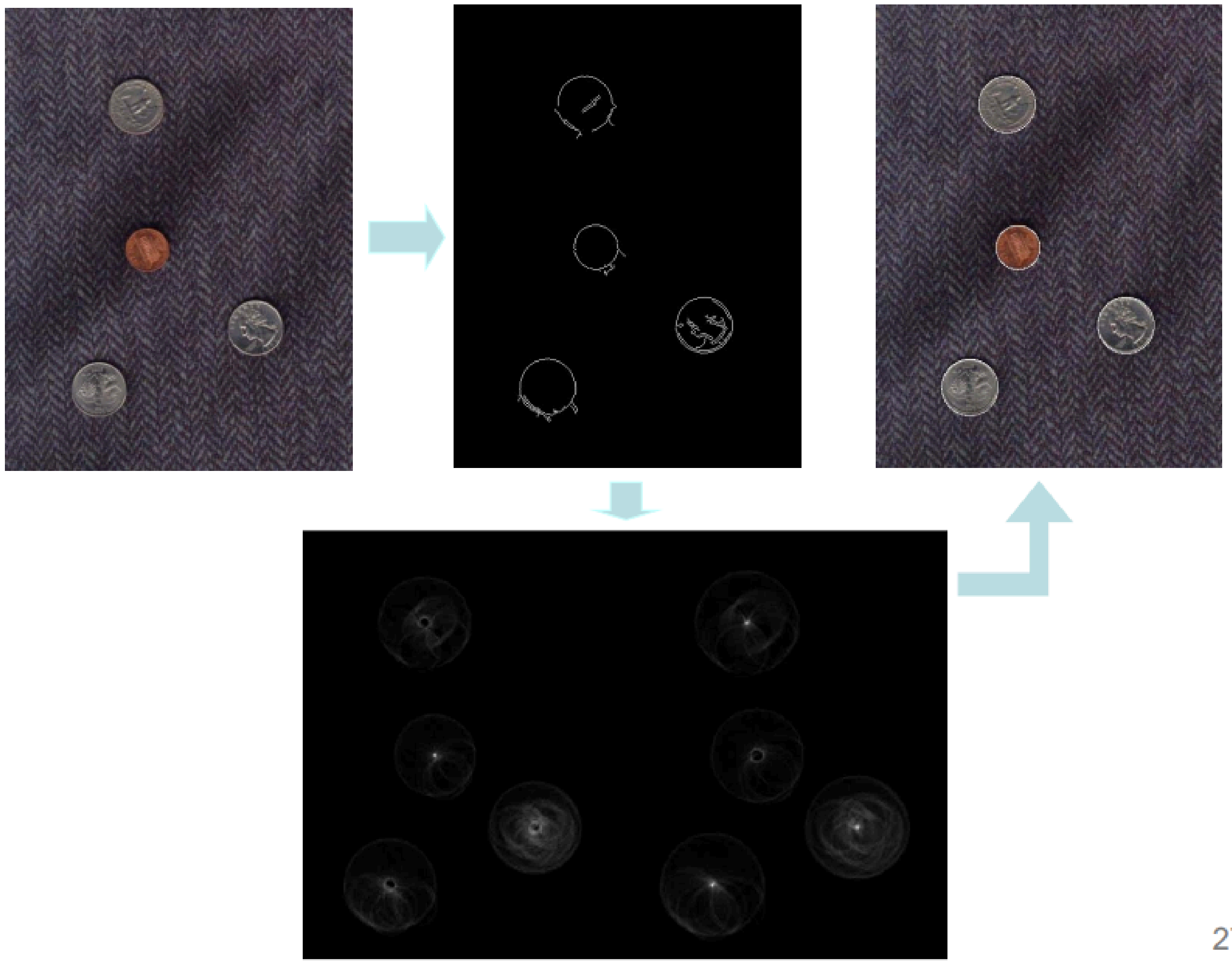
Hough can be applied to **other geometries**, but keep in mind that each additional parameter needs its own dimension:

- Circumference $(x-a)^2+(y-b)^2=r^2$
- Ellipsis $$\frac{(x-x_0)^2}{a^2}+\frac{(y-y_0)^2}{b^2}=1$$
- ...
- Any type of curve that can be analytically expressed

Circumferences:

$$(x-a)^2+(y-b)^2=r^2$$





27

Ellipsis:

Original

Borders

Detected ellipsis (in white)

ADVANTAGES:

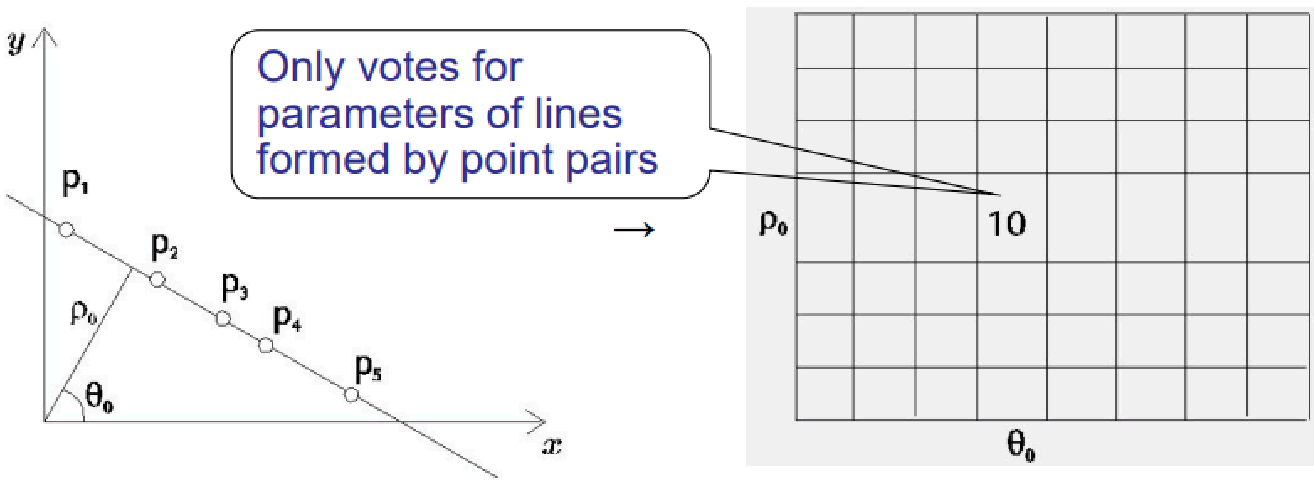
- Robust to noise
- Robust to occlusions (missing points/segments)
- Robust to presence of other forms
- Detection of multiple instances at the same time

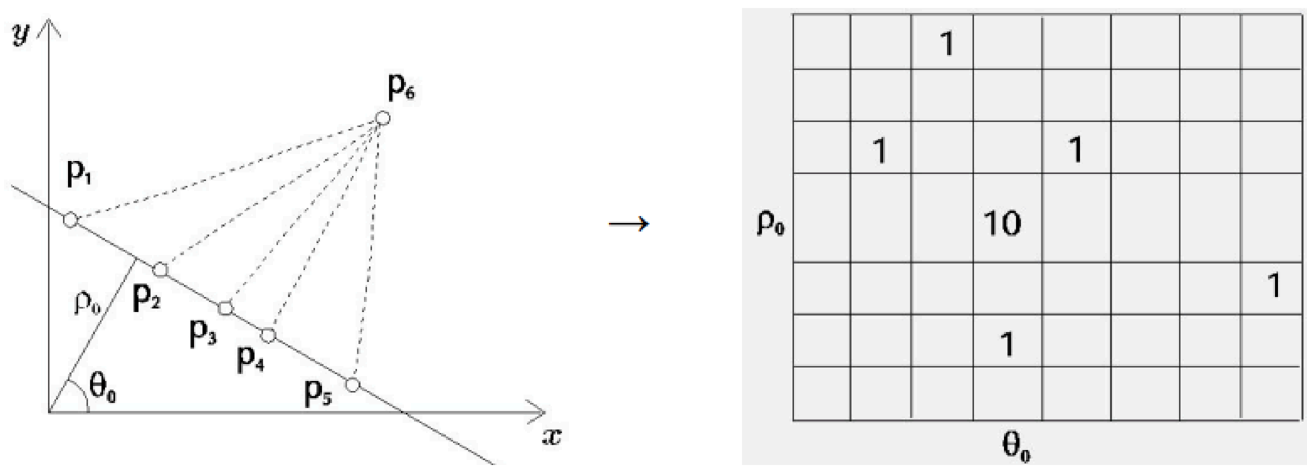
DISADVANTAGES

- False positives
- Computational cost: depending on the number of parameters, the size of the image, and the amount of noise, can take a lot of memory, and grows exponentially. This can be solved

by:
  - Pre-calculate sinus and co-sinus values
  - Multi-resolution: start with little resolution
  - Divide image into sub-images
  - Combinatorial *Hough* transform
  - Parallelize the implementation
- Resolution of accumulator space
- Peak localization: How do we detect local peaks? Which ones are sufficiently good?
  - Smooth accumulator space before peak search
  - We can perform clustering algorithm to the points in the accumulator that pass a certain threshold
  - “Eliminate” detected peaks after each iteration
  - How many peaks? Which ones are “true” peaks?
    - Set a threshold for cell votes
    - Prior knowledge
    - Problem constrains

COMBINATORY HOUGH: Avoiding outliers. If instead of "*voting*" point-by-point, we set combinations of two points to vote for only one cell in the accumulator, this way, outlier points get immediately discarded:





- IN THE FIRST EXAMPLE: We have all the points along one line, the combinatory of any two of them will vote for the same "line" (cell with value 10)
- IN THE SECOND EXAMPLE: We have one outlier and we can see that the previous points combinatories vote for the same cell, giving it a value of 10; while the combinations including p_6 only add values at random points.