Università
Ca' Foscari
Venezia

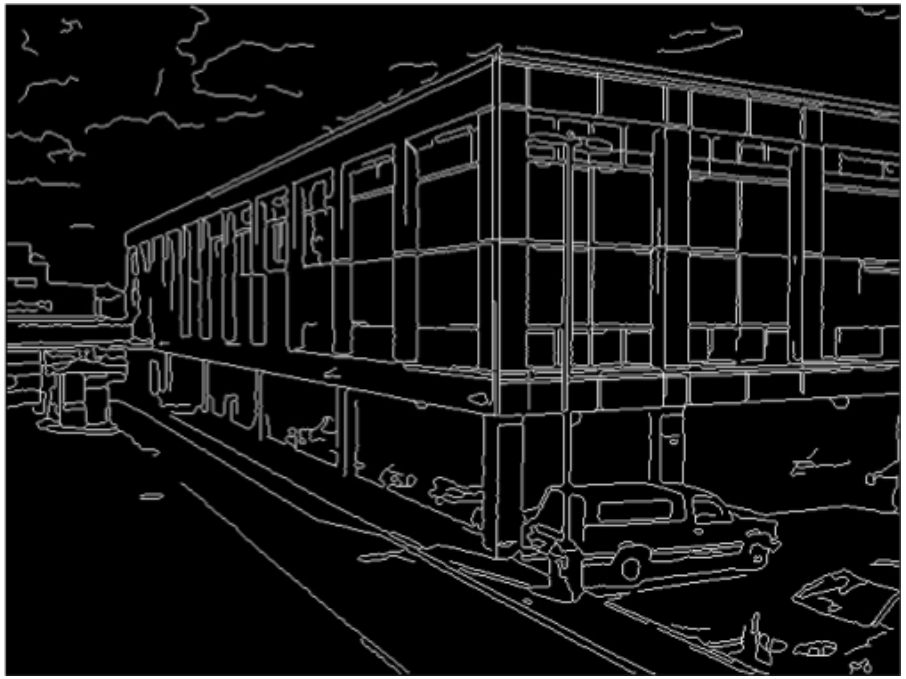
Computer Vision

Finding curves

Filippo Bergamasco (filippo.bergamasco@unive.it)
http://www.dais.unive.it/~bergamasco
DAIS, Ca' Foscari University of Venice
Academic year 2018/2019

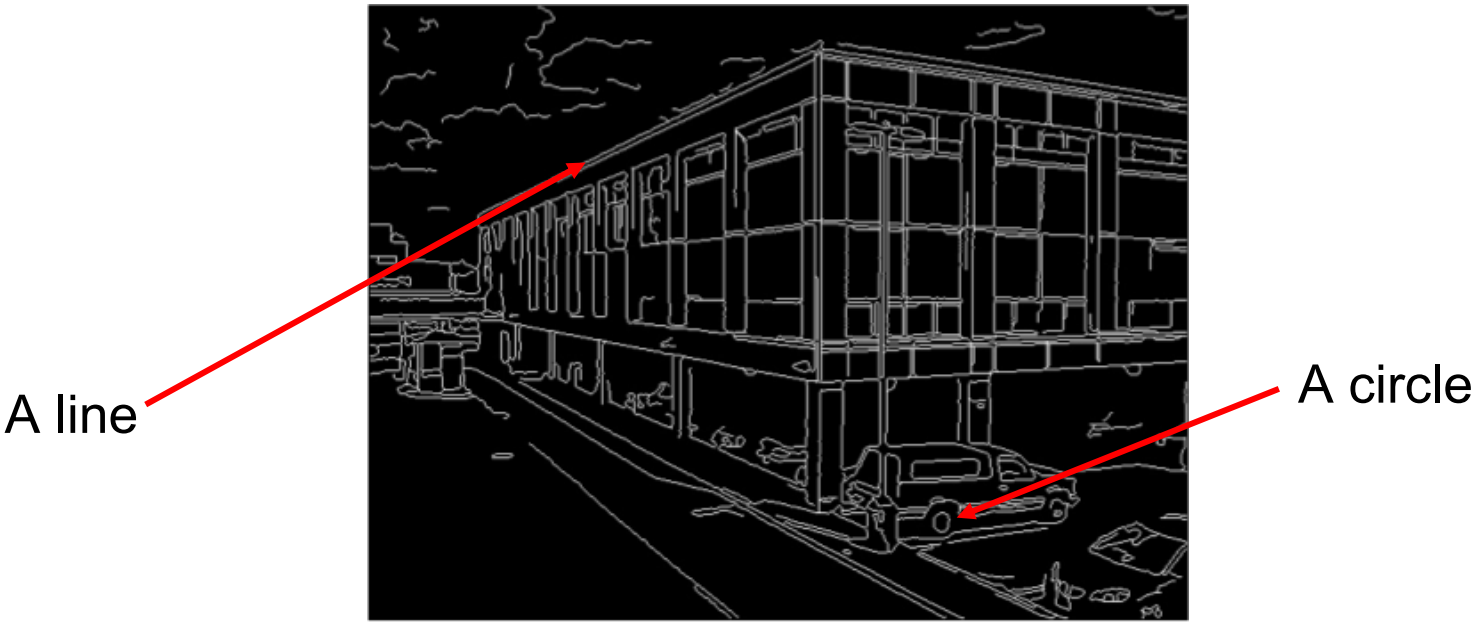
Fitting

Often, we have to work with unstructured environments in which all we have is an edge image and no knowledge about where objects of interest might be.



Fitting

Fitting is the process to decompose an image or a set of tokens (ie. pixels, isolated points, sets of edge points, etc.) into components that belong to one or another simple family (ie. circles, lines, etc)

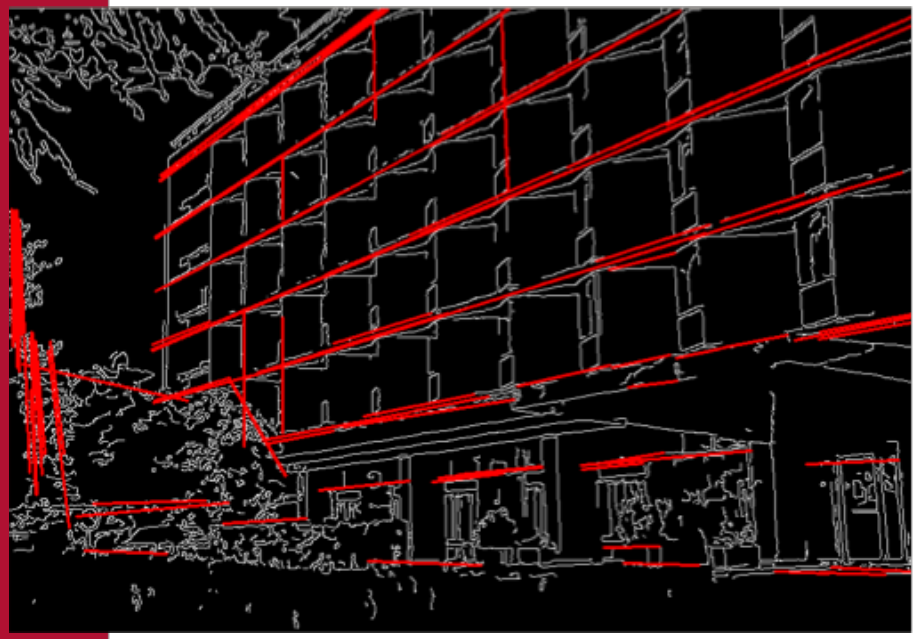


Università
Ca' Foscari
Venezia

Why fitting curves?

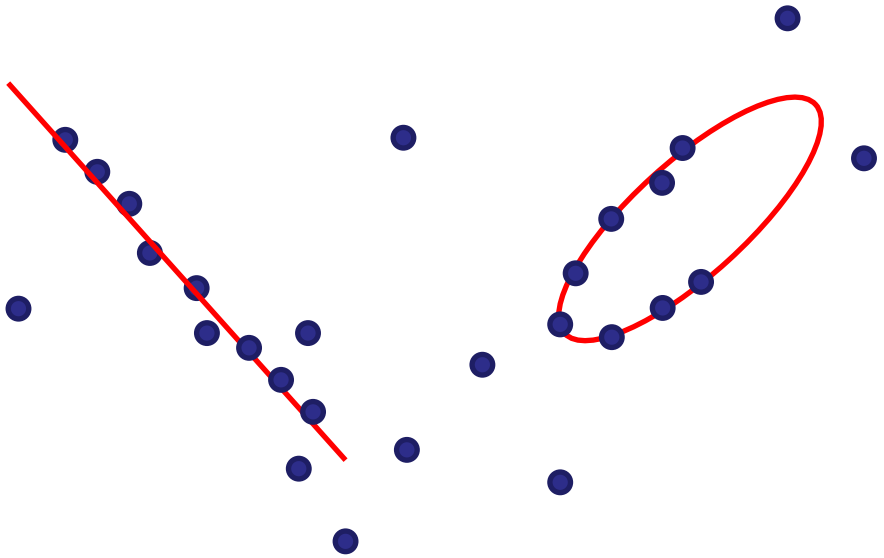
As part of the segmentation process, because it uses a model to produce compact representations that emphasize the relevant image structures.

To analyze and make measurements of man made or geometrical objects



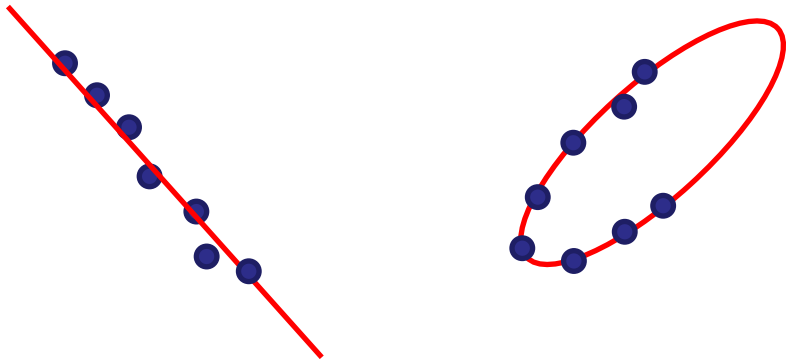
The general view

Generally, fitting involves determining what possible curves could have given rise to a set of tokens observed in an image.



The general view

Generally, fitting involves determining what possible curves could have given rise to a set of tokens observed in an image.

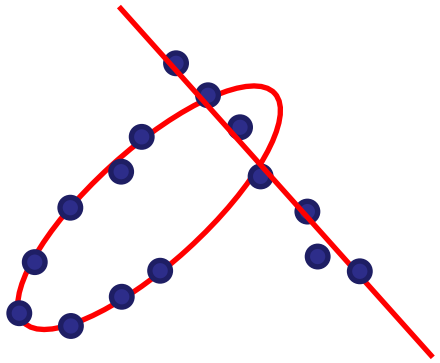


Subproblems:

Parameter estimation: we already know the association between tokens and curves. We need to recover the parameter of each curve

The general view

Generally, fitting involves determining what possible curves could have given rise to a set of tokens observed in an image.

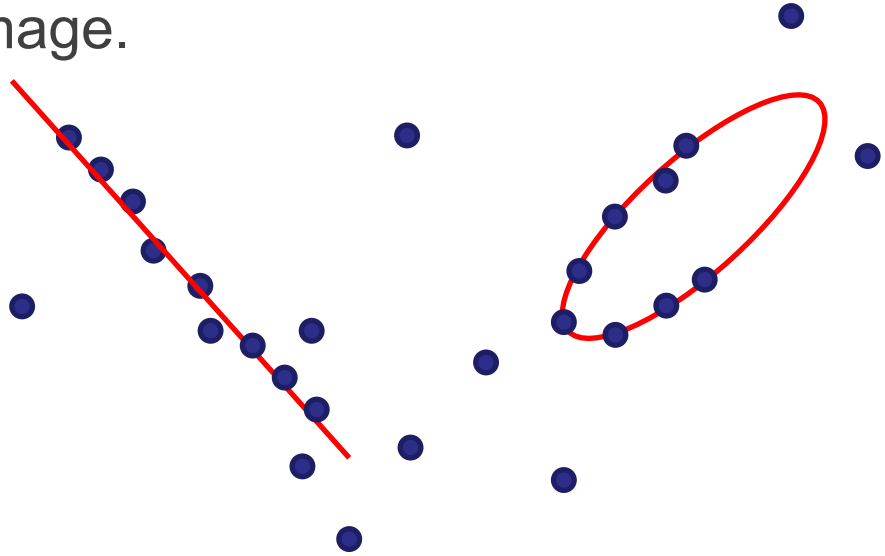


Subproblems:

Token-curve association: we assume to know only how many curves are present but not which token came from which curve. The association must be solved together with parameter estimation

The general view

Generally, fitting involves determining what possible curves could have given rise to a set of tokens observed in an image.



Subproblems:

counting: we have no prior knowledge on the data, so we must figure out (i) how many curves are present, (ii) the association between tokens and curves and (iii) curve parameters

Parameter estimation

We assume to have observed a set of points generated by a certain curve model with unknown parameters.

Goal: find the best set of parameters that justify the observations

Two approaches:

- Minimize a loss function accounting for the distances between each point and the curve
- Describe the curve as a generative model and find the best parameters maximizing the probability of generating the observed data

In some cases (like 2D lines) the two approaches yield to the same result

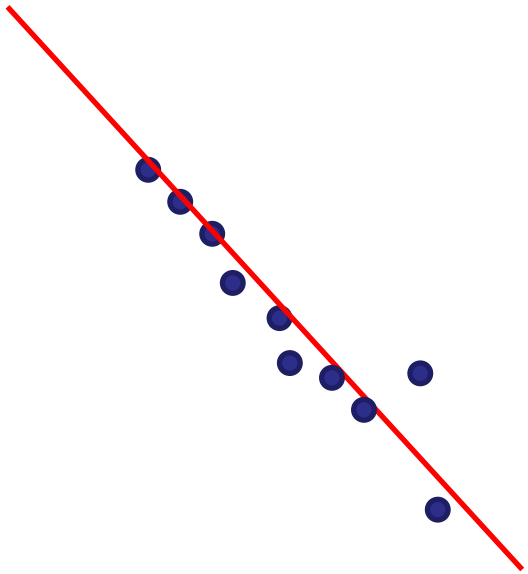
Super-simple line model

N observed data points (x_1,y_1), (x_2,y_2) … (x_n,y_n)

Model: $Y = ax + b + \epsilon$

$$\epsilon \sim N(0, \sigma^2)$$

We also assume that ϵ is independent from x_i and across observations



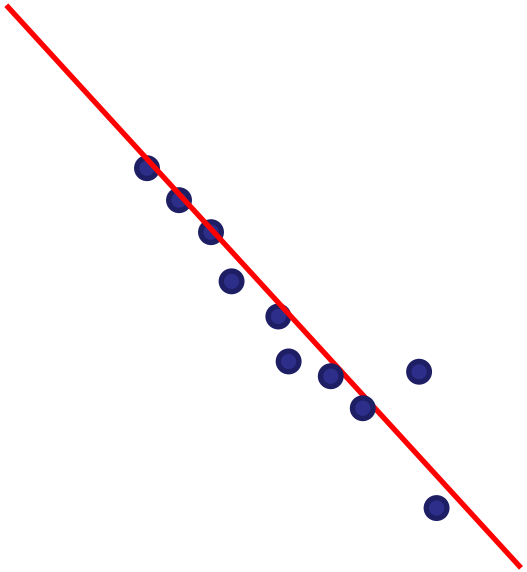
Super-simple line model

N observed data points (x_1,y_1), (x_2,y_2) … (x_n,y_n)

Model: $Y = ax + b + \epsilon$

$$\epsilon \sim N(0, \sigma^2)$$

We also assume that ϵ is independent from x_i and across observations



Y is a random variable depending by x (which can or can not be a random variable) and an additive Gaussian noise ϵ

Università Ca' Foscari Venezia

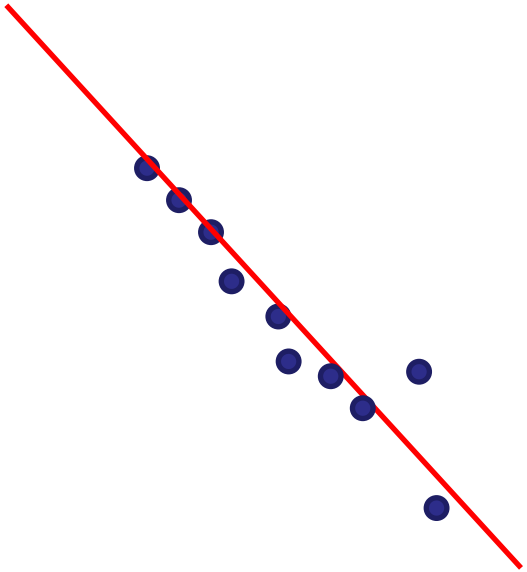
Super-simple line model

N observed data points (x_1,y_1), (x_2,y_2) … (x_n,y_n)

Model: $Y = ax + b + \epsilon$

$$\epsilon \sim N(0, \sigma^2)$$

We also assume that ϵ is independent from x_i and across observations



This means that only the y-coordinate of each measurement is affected by noise, which is why it is a rather dubious model.

Observations PDF

Since the noise model is Gaussian, it is quite simple to write the PDF of observing a certain y_i given x_i and the model parameters

$$p(y_i|x_i;a,b,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y_i-(ax_i+b))^2}{2\sigma^2}}$$

Note: recall that if $Y = A + X, \quad X \sim N(\mu, \sigma^2)$

Then $Y \sim N(A+\mu, \sigma^2)$

Observations PDF

Since Y is independent across observations, the probability to observe the whole set of data points given the model parameters is:

$$\prod_{i=1}^{n} p(y_i|x_i;a,b,\sigma^2) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y_i-(ax_i+b))^2}{2\sigma^2}}$$

Maximum Likelihood

When we see the data, we do not know the true parameters (a, b, σ^2), but any guess at them gives us the likelihood function:

$$L(a, b, \sigma^2) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y_i - (ax_i + b))^2}{2\sigma^2}}$$

In the method of **Maximum Likelihood**, we pick the best values of (a, b, σ^2) so that $L(a, b, \sigma^2)$ is maximized.

Log Likelihood

Maximizing the Likelihood is equivalent to maximize the logarithm of the Likelihood

$$\log \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y_i-(ax_i+b))^2}{2\sigma^2}}$$

$$= \sum_{i=1}^{N} \log \left(\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y_i-(ax_i+b))^2}{2\sigma^2}} \right)$$

$$= -\frac{n}{2} \log 2\pi - n \log \sigma - \frac{1}{2\sigma^2} \sum_{i=1}^{n} \left(y_i - (ax_i + b) \right)^2$$

Log Likelihood

Maximizing the Likelihood is equivalent to maximize the logarithm of the Likelihood

$$\log \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y_i-(ax_i+b))^2}{2\sigma^2}}$$

$$= \sum_{i=1}^{N} \log \left(\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y_i-(ax_i+b))^2}{2\sigma^2}} \right)$$

$$= -\frac{n}{2}\log 2\pi - n\log\sigma - \frac{1}{2\sigma^2} \boxed{\sum_{i=1}^{n} \left(y_i - (ax_i+b)\right)^2}$$

Linear least squares

Closed-form solution

A closed-form minimizer of:

$$L(a, b, \sigma^2) = -\frac{n}{2}\log 2\pi - n\log\sigma - \frac{1}{2\sigma^2}\sum_{i=1}^{n}\left(y_i - (ax_i + b)\right)^2$$

Is given by:

$$\hat{a} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i$$

$$\hat{b} = \bar{y} - \hat{a}\bar{x}$$

$$\bar{y} = \frac{1}{N}\sum_{i=1}^{N} y_i$$

$$\hat{\sigma}^2 = \frac{1}{n}\sum_{i=1}^{N}\left(y_i - (\hat{a}x_i + \hat{b})\right)^2$$

Linear least squares

$$\text{argmin}_{a,b} \sum_{i=1}^{N} \left(y_i - f(x_i, \beta)\right)^2$$

In our case: $\beta = \begin{pmatrix} a \\ b \end{pmatrix}$ $f(x_i, \beta) = \beta_1 x_i + \beta_2$

Let:

$$\mathbf{X} = \begin{pmatrix} x_1 & 1 \\ x_2 & 1 \\ \vdots & \vdots \\ x_n & 1 \end{pmatrix} \quad \mathbf{y} = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}$$

We get $\text{argmin}_\beta \|\mathbf{y} - \mathbf{X}\beta\|^2$

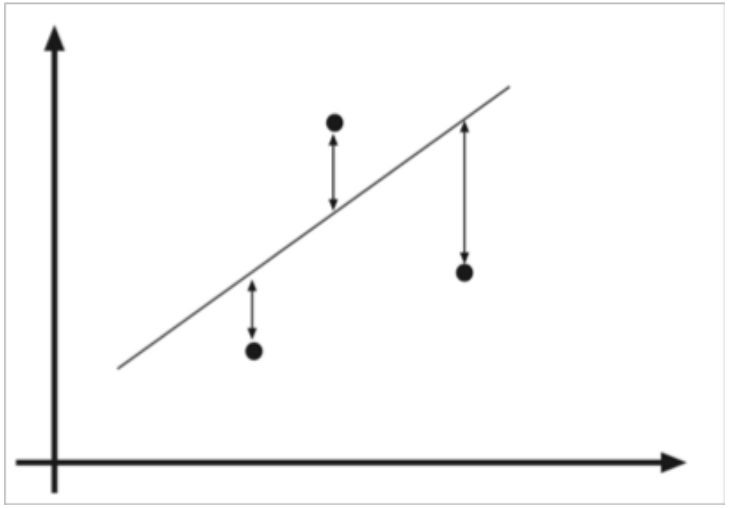
for which a closed-form solution exists:

$$\beta = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

Università
Ca' Foscari
Venezia

Limitation of this simple line model

- We account only **vertical** offsets from points to the line in our error model, this imply that the the measurement error is dependent on the reference frame
- What happens if we try to estimate the parameters of quasi-vertical lines?



A better model

Let's parametrize our line as the locus of points for which the equation $ax + by + c = 0$ holds.

Without loss of generality, let's assume that

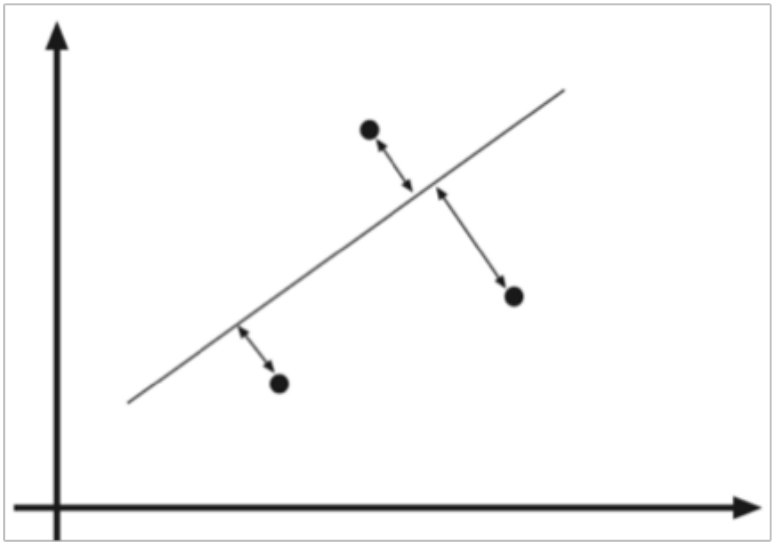
$$a^2 + b^2 = 1$$

The perpendicular distance of a point x_i, y_i to the line is given by:

$$d = |ax_i + by_i + c|$$

A better model

We now assume that our measurements are generated by choosing a point (u,v) along the line, and then perturbing it perpendicular to the line using Gaussian noise.



A better model

We have sequence of N measurements (x_i, y_i) generated by the model

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} u \\ v \end{pmatrix} + n \begin{pmatrix} a \\ b \end{pmatrix}$$

Where $n \sim N(0, \sigma^2), \quad au + bv + c = 0, \quad a^2 + b^2 = 1$

The log Likelihood is now:

$$\frac{\sum_{i=1}^{N} (ax_i + by_i + c)^2}{2\sigma^2} + C$$

Where C is a constant (independent from a,b,c) and with the additional constraint of $a^2 + b^2 = 1$

Università Ca' Foscari Venezia

Optimizing the model

The optimization problem

$$\text{argmax}_{a,b,c} \sum_{i=1}^{N} (ax_i + by_i + c)^2$$

$$\text{subject to } \; a^2 + b^2 = 1$$

lead (via the method of Lagrange multipliers) to the Eigenvalue problem:

$$\begin{pmatrix} \overline{x^2} & \overline{xy} & \overline{x} \\ \overline{xy} & \overline{y^2} & \overline{y} \\ \overline{x} & \overline{y} & 1 \end{pmatrix} \begin{pmatrix} a \\ b \\ c \end{pmatrix} = \lambda \begin{pmatrix} 2a \\ 2b \\ 0 \end{pmatrix}$$

… usually solved numerically

Fitting curves

The problem of fitting general curves is similar conceptually to the one of fitting lines:

We usually assume that data points are generated uniformly at random on the curve, and then perturbed by Gaussian noise normal to the curve

Problem: in many cases is very hard to give an exact formulation for the point-curve distance (hence we often approximate this distance)

Implicit curves

Curve	equation
Line	$ax + by + c = 0$
Circle, center (a, b) and radius r	$x^2 + y^2 - 2ax - 2by + a^2 + b^2 - r^2 = 0$
Ellipses (including circles)	$ax^2 + bxy + cy^2 + dx + ey + f = 0$ where $b^2 - 4ac < 0$
Hyperbolae	$ax^2 + bxy + cy^2 + dx + ey + f = 0$ where $b^2 - 4ac > 0$
Parabolae	$ax^2 + bxy + cy^2 + dx + ey + f = 0$ where $b^2 - 4ac = 0$
General conic sections	$ax^2 + bxy + cy^2 + dx + ey + f = 0$

Note that not all the possible curves are guaranteed for having any real points on them (ex. $x^2 + y^2 + 1 = 0$)

Curve distance

Let's assume that our curve is expressed implicitly as

$$\phi(x, y) = 0$$

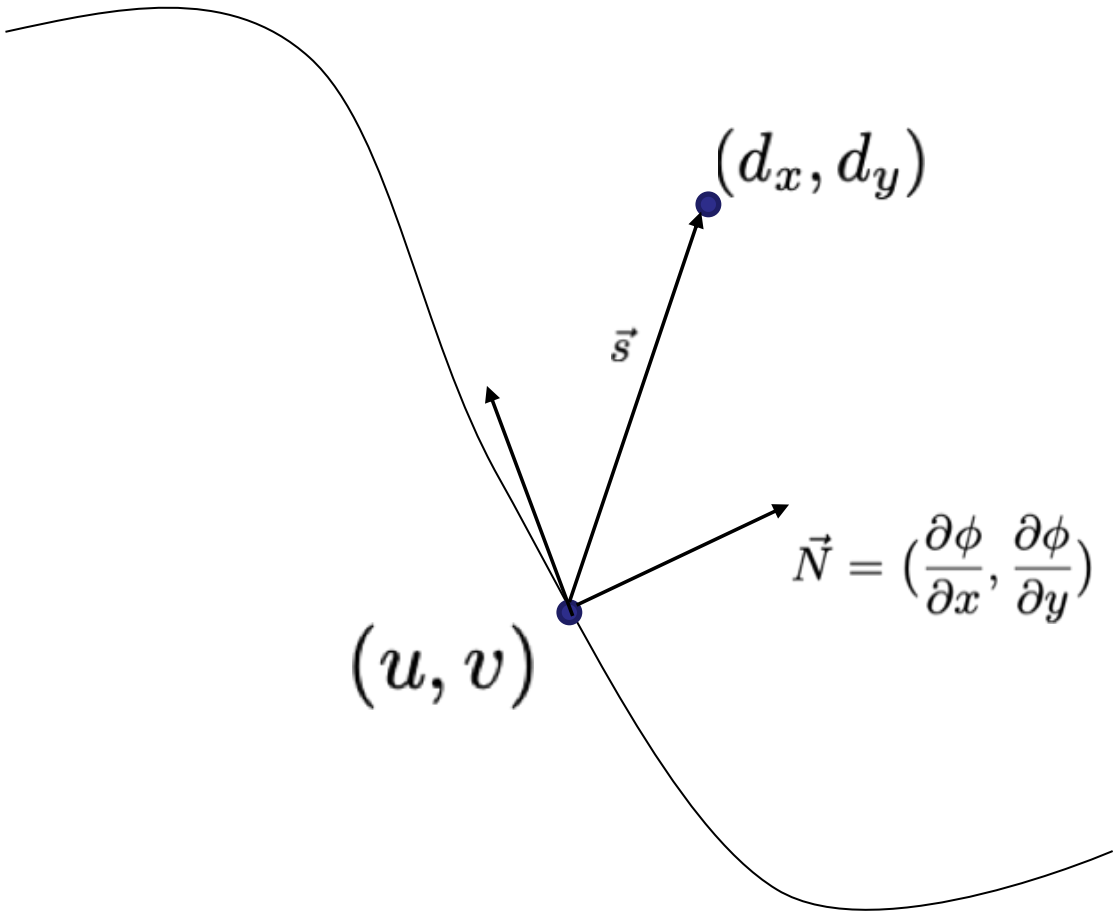
Given a data point (d_x d_y), to find the closest point on a curve we must first enumerate all the pairs of (u,v) such that:

- (u,v) is a point on the curve (ie. $\phi(u, v) = 0$)
- $\vec{s} = (d_x, d_y) - (u, v)$ is normal to the curve.

Then, we take the (u,v) for which $|\vec{s}|$ is minimum

Università
Ca' Foscari
Venezia

Curve distance



We must find all the (u,v) for which:

$$\frac{\partial \phi}{\partial y}(u,v)\{d_x - u\} - \frac{\partial \phi}{\partial x}(u,v)\{d_y - v\} = 0$$

A difficult problem!

Even for simple cases, it can be impossible to obtain a simple closed form solution.

Ex.

Consider the ellipse $2x^2 + y^2 - 1 = 0$

The nearest point on the curve to (d_x, d_y) must satisfy the equations:

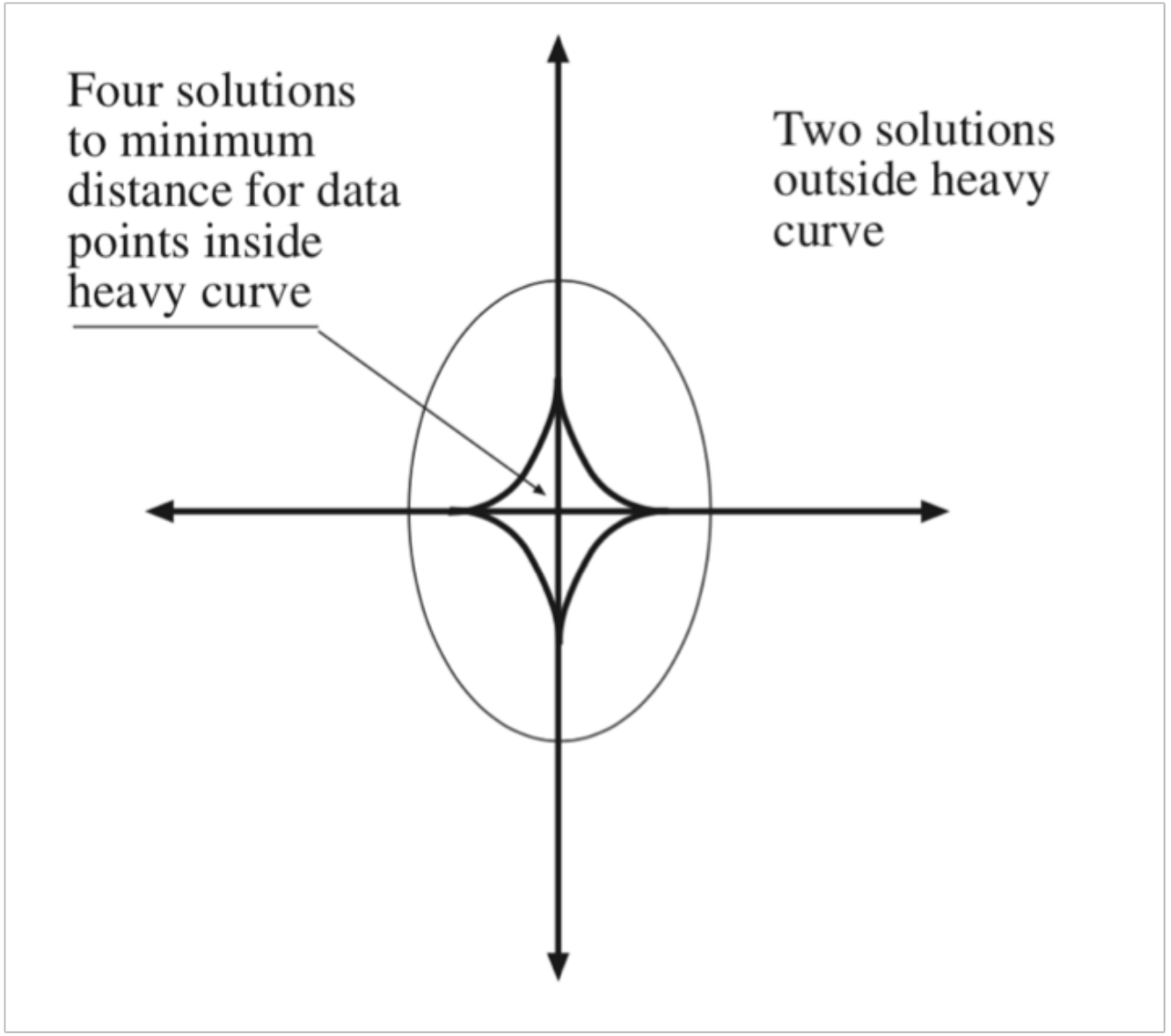
$$2u^2 + v^2 - 1 = 0$$

$$2uv - 4d_y u + 2d_x v = 0$$

Which can result in 2 or 4 solutions depending on the data point

Università
Ca' Foscari
Venezia

A difficult problem!



Distance approximations

If $\phi(x, y)$ is a polynomial with degree k, the closest point on the corresponding curve would be obtained by solving two simultaneous polynomial equations with degree k (up to k^2 solutions to consider)

To solve the problem we can:

- Consider specific properties of the curve
- Use an approximation of the distance function

In both the cases tradeoff must be done to consider numerical stability and computational cost

Algebraic distance

We measure the distance between a curve and a point by evaluating the polynomial equation at that point:

$$\mathrm{dist}(d_x, d_y, \phi) = \phi(d_x, d_y)$$

Problems:

- Accurate **only** if the point is very close to the curve
- Since $\mu\phi(d_x, d_y) = 0$ and $\phi(d_x, d_y) = 0$ represent the same curve, polynomial coefficients should be normalized in some way before evaluating the distance

Algebraic distance

In case of first degree polynomials (ie. lines) the normalization given by $a^2 + b^2 = 1$ lets the algebraic distance and geometric distance coincide

… but it is one of the few lucky situations

Note also that the choice of normalization factor is important and can lead to the exclusion of possible good solutions:

Ex:

If we use $ax^2 + bxy + cy^2 + dx + ey + f = 0$

with the constraint b=1 we cannot fit circles

Normalizing the distance

Another common approximation is to consider the following distance:

$$\mathrm{dist}(d_x, d_y, \phi) = \frac{\phi(d_x, d_y)}{\|\nabla\phi(d_x, d_y)\|}$$

Advantages:

- Does not require to choose a normalization factor for the curve
- More accurate than algebraic distance, because it is normalized by the length of the normal

Can still be numerically inaccurate for points far away from the curve

Token-curve association

Suppose that we want to detect street lanes to develop an autonomous vehicle



Option 1: We can limit the analysis to a specific region and do a parameter fitting of a line

Bad idea if performed in the wild… Why?

Token-curve association

Suppose that we want to detect street lanes to develop an autonomous vehicle

Option 2: We can search for lines at every possible position/orientation to find the "best" ones

Computationally very expensive

Università
Ca' Foscari
Venezia

Token-curve association

Suppose that we want to detect street lanes to develop an autonomous vehicle



Option 3: We can use a consensus-based approach: RANSAC

Università
Ca' Foscari
Venezia

Token-curve association

Suppose that we want to detect street lanes to develop an autonomous vehicle



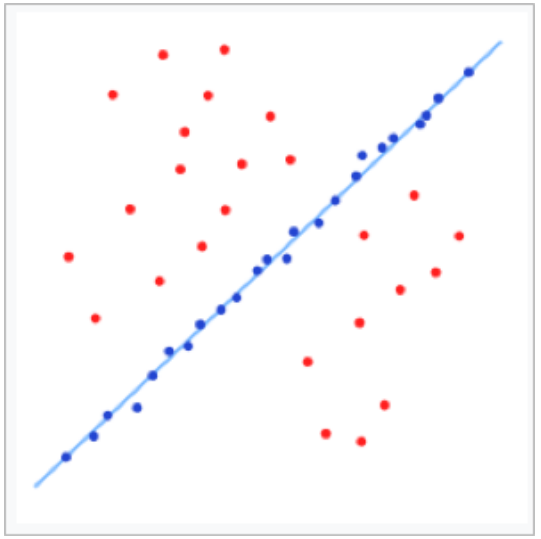
Option 4: We can use a voting scheme:

Hough Transform

RANSAC

The RANSAC algorithm is a learning technique to estimate parameters of a model by random sampling of observed data.

Given a dataset whose data elements contain both inliers and outliers, RANSAC uses a consensus scheme to find the optimal fitting result.



RANSAC

Assumptions:

1. Data consists of **inliers** (i.e., data whose distribution can be explained by some set of model parameters, though may be subject to noise) and **outliers** which are data that do not fit the model

1. Given a (usually small) set of inliers, there exists a procedure which can estimate the parameters of a model that optimally explains or fits this data

   > For example given a set of 2 points we can compute a line model that optimally explains the set

RANSAC

Algorithm:

1. Select a random subset of the original data. Call this subset the hypothetical inliers.
2. Fit the model to the hypothetical inliers
3. Test all other data against the model and mark points either as inliers or outliers according to some loss function. The inliers are called "consensus set"
4. Return to step 1 until a predefined number of iterations **N** is reached
5. The model that produced the largest consensus set is returned

How many iterations?

e: probability that a point is an outlier

s: number of points needed to fit a model

N: number of RANSAC interations

p: desired probability to find a good model given the samples

$$1 - \left(1 - (1 - e)^s\right)^N = p$$

Università Ca' Foscari Venezia

How many iterations?

e: probability that a point is an outlier

s: number of points needed to fit a model

N: number of RANSAC interations

p: desired probability to find a good model given the samples

$$1 - \left(1 - \left(1 - e\right)^{s}\right)^{N} = p$$

Probability to have an inlier by choosing one point

Università Ca' Foscari Venezia

How many iterations?

e: probability that a point is an outlier

s: number of points needed to fit a model

N: number of RANSAC interations

p: desired probability to find a good model given the samples

$$1 - \left(1 - \boxed{(1-e)^s}\right)^N = p$$

Probability to have s inliers in a row

How many iterations?

e: probability that a point is an outlier

s: number of points needed to fit a model

N: number of RANSAC interations

p: desired probability to find a good model given the samples

$$1 - \left(1 - (1 - e)^s\right)^N = p$$

Probability that one or more points in a sample of s points is an outlier

How many iterations?

e: probability that a point is an outlier

s: number of points needed to fit a model

N: number of RANSAC interations

p: desired probability to find a good model given the samples

$$1 - \boxed{\left(1 - (1-e)^s\right)^N} = p$$

Probability that in all the N iterations we fitted models contaminated by outliers

How many iterations?

e: probability that a point is an outlier

s: number of points needed to fit a model

N: number of RANSAC interations

p: desired probability to find a good model given the samples

$$1 - \left(1 - (1 - e)^s\right)^N = p$$

Probability that, in at least one iteration, we fitted a model with just inlier points

How many iterations?

Usually we specify p (ie. The probability of success we expect to have) and compute the corresponding number of iterations:

$$1 - \left(1 - (1 - e)^s\right)^N = p$$

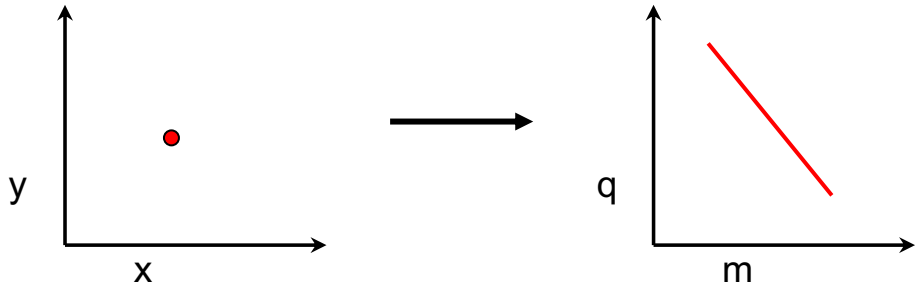
$$\left(1 - (1 - e)^s\right)^N = 1 - p$$

$$N = \frac{\log(1 - p)}{\log\left(1 - (1 - e)^s\right)}$$

Hough transform

The basic idea is to map points from the image space to **the parameter space of the model** (for example the m-q space for lines parameterized as y=mx + q)

For an image point (x,y) can pass infinite lines all satisfying the equation y=mx+q. The equation can be rewritten as q=-mx+y which corresponds to a line in the m-q space

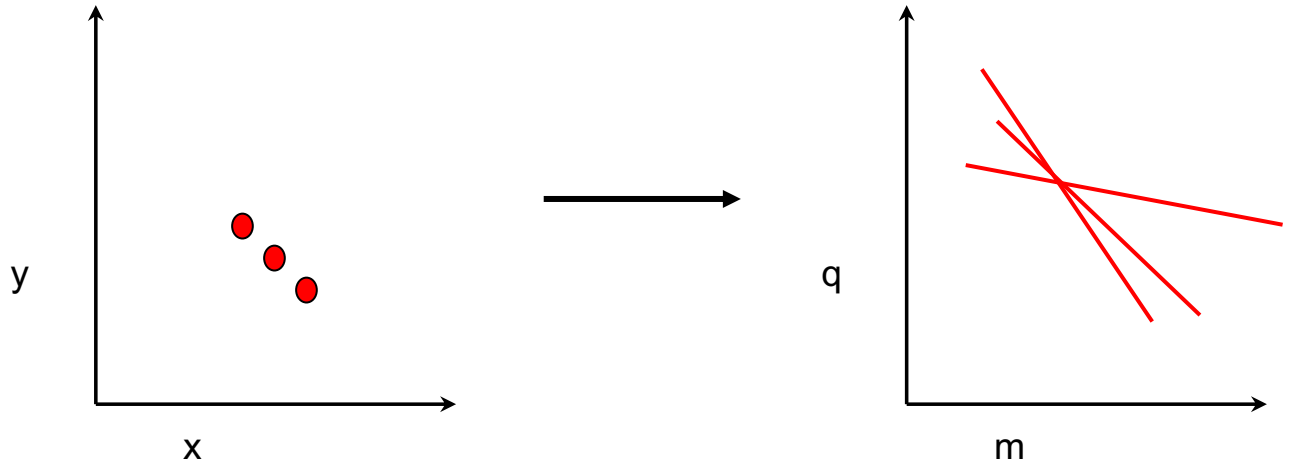


Università Ca' Foscari Venezia

Hough transform

The principal lines in the image plane could be found by identifying points in parameter space where large numbers of parameter-space lines intersect

The parameter space is used as an accumulator of votes



Hough transform

The classical slope-intercept parameterization of the line is not convenient since the slope approach infinity when the line approaches vertical direction

Normal representation of the line:

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

The parameter space is now the r-θ-plane in which the range of values are limited

Hough transform

Algorithm:

1. Initialize H[r,θ]=0
2. For each edge point p=(x,y) in the image
   a. For θ=0 to pi
      i. r = x cos θ + y sin θ
      ii. H[r,θ] += 1
3. Find (r,θ) for which H[r,θ] is maximum
4. The detected line is given by r = x cosθ + y sinθ

Hough Transform

Possible extensions/improvements:

1. Use the image gradient (no need to iterate through angles)
2. Give more votes to strongest edges
3. Change the sampling of (r,θ) to trade-off resolution with computing time
   a. High resolution -> Dispersion of votes
   b. Low resolution -> Cannot distinguish similar lines

Hough Transform

Hough transform is applicable to any function of the form g(**v**, **c**) = 0, where:

- **v** is a vector of coordinates
- **c** is a vector of coefficients

For example the Hough Transform can be used to extract all circles in the scene:

$$(x - c_1)^2 + (y - c_2)^2 = c_3^2$$

A 3D accumulator for (c_1,c_2,c_3) is needed