# Image Analysis - Lecture 7

Fitting

## Lecture 7

### Contents

- Fitting
  - The Hough transform
  - Fitting lines
  - Curve fitting
- Robust fitting
  - M-estimators
  - Ransac

## Line Fitting as Study Problem

**Fitting** involves determining what possible structures could have given rise to a set of tokens in an image. For example, we might have a set of edge points (the tokens) and wish to determine which lines fit them best. There are three increasingly more general problems that occur in fitting:

**1. Parameter estimation:** Assume we know which tokens came from a particular structure, and we want to know what the parameters of the structure are.

- For example, we might have a set of edge points, all of which are known to have come from a line, and we wish to know what line they came from.
- Most interesting case is when criterion is not local - cannot tell whether a set of points lies on a line by looking only at each point and the next.

## Line Fitting as Study Problem (contd' 2)

**2. Data Association:** Assume we know how many structures are present, and we wish to determine which tokens came from which structure.

- For example, we might have a set of edge points, and we need to know the best set of lines fitting these points; this involves (1) determining which points belong together on a line and (2) figuring out what each line is.
- Generally, these problems are not independent (because one good way of knowing whether points belong together on a line is checking how well the best fitting line approximates them).

## Line Fitting as Study Problem (contd' 3)

**3. Model Selection**: We would like to know (1) how many structures are present (2) which points are associated with which structure and (3) what the structures are.

- For example, given a set of edge points, we might want to return a set of lines that fits them well.
- This is, in general, a substantially difficult problem the answer to which depends strongly on the type of model adopted (for example, we could simply pass a line through every pair of edge points – this gives a set of lines that fit extremely well, but are a poor representation).

## Hough Transform

Goal: Finding linear structures in images.
Used on edge data

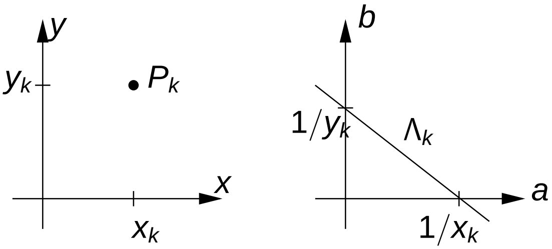
$$l_{a,b}: \quad ax + by = 1 \quad (\text{Assume } 0 \notin l)$$

$$(x_k, y_k) \in l_{a,b} \quad \Leftrightarrow \quad ax_k + by_k = 1$$

Study the set

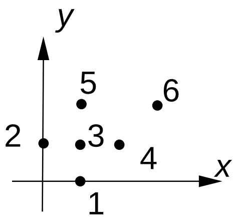
$$\Lambda_k = \{ (a, b) \,|\, (x_k, y_k) \in l_{a,b} \}$$

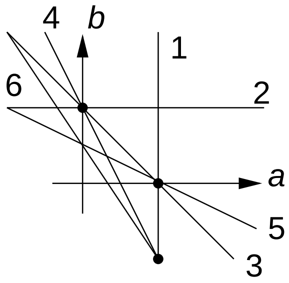
which forms a line in the *ab*-plane.



*Idea: Cover the ab-plane with squares (accumulator cells) and add* 1 *to the cells that contain* $\Lambda_k$.

## Example





$\Lambda_1 : a = 1$ $\quad \Lambda_2 : b = 1$

$\Lambda_3 : a + b = 1$ $\quad \Lambda_4 : 2a + b = 1$

$\Lambda_5 : a + 2b = 1$ $\quad \Lambda_6 : 3a + 2b = 1$

## Example (ctd.)

There are three lines through the points $(0, 1)$, $(1, 0)$ and $(1, -1)$, which corresponds to the three lines

$$y = 1 \quad x = 1 \quad x - y = 1.$$

*One has to threshold and find local maxima in the Hough transform to obtain the 'interesting' lines.*

## Other line representations

**1)** Represent lines as

$$y = kx + l.$$

Each line is a point in the $kl$-plane.
Cannot represent vertical lines.

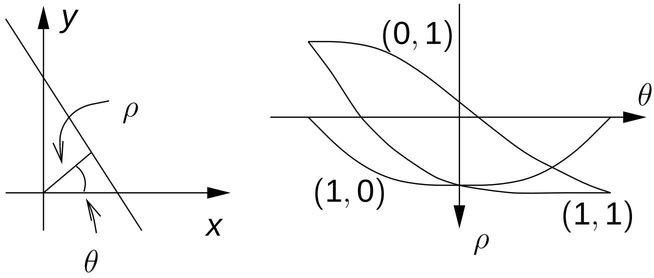
## Other line representations (cont.)

**2)** Represent the lines as

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

Each line is a point in the $\rho\theta$-plane. Represent all lines.
Each point gives a sine-formed curve in the $\rho\theta$-plane.

## Example



The Hough transform can be generalized to arbitrary shapes instead of lines.

## Hough Transform ($\rho, \theta$ representation)

- Construct an array representing $[\rho, \theta]$. For each point, render the curve $\rho\theta$ into this array, adding one at each cell.
- *Difficulties:* **How big should the cells be?** Too big, and we cannot distinguish between quite different lines; too small, and noise causes lines to be missed.
- **How many lines?** Count the peaks in the Hough array.
- **Who belongs to which line?** Tag the votes.

Problems with noise and cell size can defeat it

## The least squares method

**Line fitting**

Assume that the points $(x_i, y_i)$ are measured. Then

$$y_i = kx_i + l$$

for line parameters $(k, l)$.

Assume that:

- the line is not vertical.
- that the errors are only in the $y$-direction.

## Line fitting

Then

$$y = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} = \begin{pmatrix} x_1 & 1 \\ \vdots & \vdots \\ x_n & 1 \end{pmatrix} \begin{pmatrix} k \\ l \end{pmatrix} + \mathbf{n} = Ap + \mathbf{n}$$

If the errors **n** are independent and Gaussian distributed, then it is reasonable to solve $y = Ap$ in least squares sense, i.e. minimizing $|y - Ap|$.

## Least squares solution

This least squares problem was studied in Lecture 2 (and in other courses).
The solution is

$$p = (A^T A)^{-1} A^T y$$

Write this out to obtain

$$\begin{pmatrix} k \\ l \end{pmatrix} = \begin{pmatrix} \bar{x^2} & \bar{x} \\ \bar{x} & 1 \end{pmatrix}^{-1} \begin{pmatrix} \bar{xy} \\ \bar{y} \end{pmatrix}$$

## Least squares in Matlab

In matlab the least squares solution can be obtained using the slash function

$$p = A \backslash y$$

Read the help text 'help slash' for more information about how the 'slash'-operator works.

## 'Total Least Squares'

- One problem with the idea above lies in the two assumptions.
- It cannot handle vertical lines.
- For lines that are close to vertical the assumption that the errors are only in the y-direction gives sub-optimal estimates of the line.
- It is better to minimize the distance between the point and the line.

## Distance between point and line

From linear algebra we know that the distance between the point

$$(x, y)$$

and the line

$$ax + by + c = 0$$

with line parameters

$$l = (a, b, c)$$

is

$$\frac{|ax + by + c|}{\sqrt{a^2 + b^2}}$$

## Optimization problem

Assume that

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

then the distance is

$$d = |ax + by + c|.$$

The line $l = (a, b, c)$ that minimizes the sum of squares of the distance is given by

$$\min_{a,b,c,a^2+b^2=1} f(a, b, c) = \sum_i (ax_i + by_i + c)^2 \ .$$

## Solution

The Lagrange function is

$$L(a,b,c,\lambda) = \sum_i (ax_i + by_i + c)^2 + \lambda(1 - a^2 - b^2)$$

whose stationary points is given by (denote e.g. $\overline{x^2} \leftarrow \sum_i x_i^2$)

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

## Solution

We can solve for $c$ using the last equation:

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

Then we can substitute $c = -a\bar{x} - b\bar{y}$ in the equations above to obtain

$$\begin{pmatrix} \overline{x^2} - \bar{x}\bar{x} & \overline{xy} - \bar{x}\bar{y} \\ \overline{xy} - \bar{x}\bar{y} & \overline{y^2} - \bar{y}\bar{y} \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \mu \begin{pmatrix} a \\ b \end{pmatrix}$$

## Solution as eigenvalue problem

$$\begin{pmatrix} \overline{x^2} - \bar{x}\bar{x} & \overline{xy} - \bar{x}\bar{y} \\ \overline{xy} - \bar{x}\bar{y} & \overline{y^2} - \bar{y}\bar{y} \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \mu \begin{pmatrix} a \\ b \end{pmatrix}$$

is an eigenvalue problem.
There are two solutions, up to scale. Can be obtained in closed form.
The two solutions are orthogonal. One maximizes the likelihood, the other minimizes it.
It is straightforward to test which one of the two minimize $f(a, b, c)$.

## Incremental line fitting

- It is common to detect many edge points along a line.
- One may then want to incrementally update the line parameters as more points are ’added’ to the line’.
- Read chapter 15.2.2 in the book.

## K-means and fitting

Assume that there are points from many lines.
Assume also that the number of lines is known.
Then one can use $k - means$ for clustering points to lines.
Algorithm 15.2

1. Randomly choose $k$ lines or a correspondence function $c = \{1 \ldots, n\} \rightarrow \{1, \ldots, k\}$
2. Update $c$, i.e. assign points to the closest line.
3. Update $l$, i.e. fit lines to corresponding points.

## Curve fitting

Similar ideas can be used to fit conics to points

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

or even higher order algebraic curves.
Read chapter 15.3!

## Inference

Assumes that we measure random points $(x, y)$ along a line $(a, b, c)$ with an error in the normal direction $(a, b)$ that is Gaussian distributed.
Then the logarithm of the likelihood function is

$$\frac{1}{2\sigma^2}\sum_i (ax_i + by_i + c)^2$$

with constraint $a^2 + b^2 = 1$, and $\sigma$ denoting the standard deviation of the noise.

## Problems with inference

Some practical problems:

- Robustness - often there are points there that do not belong to the object.
- There might be missing data
- It is difficult to establish correct correspondence between points and objects.

How can one make the fitting less sensitive to such errors?

## Outliers: M-estimators

A common method is to use an error function which is quadratic for small errors, but large for larger errors.
Then large errors (outliers) will not affect the fitting as much.
Instead of minimizing

$$\sum_i (ax_i + by_i + c)^2$$

we minimize

$$\sum_i \rho(ax_i + by_i + c, \sigma)$$

where e.g. one could use

$$\rho(u, \sigma) = \frac{u^2}{\sigma^2 + u^2}$$

## M-estimators

- How should $\sigma$ be chosen?
- Read in the book. Study the examples.
- Problem with convergence to local optima.
- How do we obtain an initial guess?

## Outliers: RANSAC

Another popular method to deal with outliers is RANSAC. It is an alternative to M-estimators (where we modified the underlying noise model to have heavier tails):

1. Randomly choose a minimal set of points needed for fitting.
2. Study how many points that now lie close to the line.
3. If there are sufficiently many, stop
4. Iterate 1-3 until stop, but at most k times.

## Masters thesis suggestion of the day: City guide, recognition and 3D reconstruction

Recognition and model based reconstruction for generating 3D models.
Images of a house - finding walls, roofs, windows and doors automatically.
Fit models to the objects found.
Localization (city guide). Using one or more images from one position and a 3D model of the surrounding one will determine where the photo was taken. Here recognition and modeling could be important components.

## Review - Lecture 7

- Segmentation and Fitting
  - Hough transform
  - Lines
  - Curves (in particular conics)
  - as inference
- Robustness
  - M-estimation
  - RANSAC