Contents

- Segmentation using Graph Cuts
- Fitting
  - The Hough transform
  - Fitting lines
  - Curve fitting
- Robust fitting: M-estimators, Ransac
Cut-methods for segmentation

- Popular method: Normalized cuts. Similar to graph method in Lecture 6, but objective is modified to avoid small clusters. See Section 14.5.5. Works poorly sometimes.
- Even more popular method: Graph cuts. Works very well. See "Graph cuts homepage" on the internet for recent applications and tutorials.
- Very efficient methods exist to find global minima for graph cuts
- Example on blackboard.
Hough Transform

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

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

\[ (x_k, y_k) \in l_{a,b} \iff ax_k + by_k = 1 \]

Study the set

\[ \Lambda_k = \{ (a, b) \mid (x_k, y_k) \in l_{a,b} \} \]

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

\[ \Lambda_1 : a = 1 \]
\[ \Lambda_2 : b = 1 \]
\[ \Lambda_3 : a + b = 1 \]
\[ \Lambda_4 : 2a + b = 1 \]
\[ \Lambda_5 : a + 2b = 1 \]
\[ \Lambda_6 : 3a + 2b = 1 \]
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.
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.
The Hough transform can be generalized to arbitrary shapes instead of lines.
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.
Then

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

If the errors \( n \) are independent and Gaussian distributed, then it is reasonable to solve \( y = Ap \) in least squares sense, i.e. minimizing \( |y - Ap| \).
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
\[
\begin{pmatrix} k \\ l \end{pmatrix} = \begin{pmatrix} \bar{x}^2 & \bar{x} \\ \bar{x} & 1 \end{pmatrix}^{-1} \begin{pmatrix} \bar{x} \bar{y} \\ \bar{y} \end{pmatrix}
\]
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.
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

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

\[
\min_{a,b,c, a^2 + b^2 = 1} f(a, b, c) = \sum_i (ax_i + by_i + c)^2.
\]
The Lagrange function is

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

whose stationary points is given by

\[
\begin{pmatrix}
\bar{x}^2 & \bar{x}\bar{y} & \bar{x} \\
\bar{x}\bar{y} & \bar{y}^2 & \bar{y} \\
\bar{x} & \bar{y} & 1
\end{pmatrix}
\begin{pmatrix}
a \\
b \\
c
\end{pmatrix}
= \lambda
\begin{pmatrix}
a \\
b \\
0
\end{pmatrix}
\]
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

$$
\begin{pmatrix}
\bar{x}^2 - \bar{x}\bar{x} & \bar{x}\bar{y} - \bar{x}\bar{y} \\
\bar{x}\bar{y} - \bar{x}\bar{y} & \bar{y}^2 - \bar{y}\bar{y}
\end{pmatrix}
\begin{pmatrix}
a \\
b
\end{pmatrix}
= \mu
\begin{pmatrix}
a \\
b
\end{pmatrix}
$$
Solution as eigenvalue problem

\[
\begin{pmatrix}
\bar{x}^2 - \bar{x}\bar{x} & \bar{x}\bar{y} - \bar{x}\bar{y} \\
\bar{x}\bar{y} - \bar{x}\bar{y} & \bar{y}^2 - \bar{y}\bar{y}
\end{pmatrix}
\begin{pmatrix}
a \\
b
\end{pmatrix}
= \mu
\begin{pmatrix}
a \\
b
\end{pmatrix}
\]

is an eigenvalue problem.
There are two solutions.
The correct solution and a line that is perpendicular to the correct one.
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 an edge.
- 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 $k - \text{means}$ 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.
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!
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 does 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.
- How do we obtain an initial guess?
Another popular method to deal with outliers is RANSAC

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.
Modelling of animal vision

Collaboration with the vision group at the department of Biology. Modelling of animal visual computation and motor feedback. Jellyfish - with 24 eyes and simple motoric output. Generate data by filming the Jellyfish and giving it different visual stimuli.
Review - Lecture 8

- Segmentation and Fitting
  - Hough transform
  - Lines
  - Curves (in particular conics)
  - as inference

- Robustness
  - M-estimation
  - RANSAC