Model Fitting

Faisal Qureshi
faisal.quareshi@uoit.ca
Model Fitting

Least squares
Robust least squares
RANSAC
Hough Transform
Application: Image Stitching
2D Line Fitting

\[ y = mx + c \]

Unknowns

Under-constrained system
2D Line Fitting

\[
\begin{bmatrix}
    x_1 \\
    x_2
\end{bmatrix}
\begin{bmatrix}
    m \\
    c
\end{bmatrix}
= 
\begin{bmatrix}
    y_1 \\
    y_2
\end{bmatrix}
\Rightarrow
\begin{bmatrix}
    m \\
    c
\end{bmatrix} = 
\begin{bmatrix}
    x_1 & 1 \\
    x_2 & 1
\end{bmatrix}^{-1}
\begin{bmatrix}
    y_1 \\
    y_2
\end{bmatrix}
\]

\[
\begin{align*}
y_1 &= mx_1 + c \\
y_2 &= mx_2 + c
\end{align*}
\]

System of linear equations

\[
Ap = y
\]
2D Line Fitting

\[ \begin{pmatrix} x_1 & 1 \\ x_2 & 1 \\ \vdots & \vdots \\ x_n & 1 \end{pmatrix} \begin{pmatrix} m \\ c \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} \]

\[ \text{Ap} = \mathbf{y} \]

Over-constrained system
2D Line Fitting

\[
A^T A \in \mathbb{R}^{2 \times 2}
\]

\[
A \in \mathbb{R}^{n \times 2}
\]

\[
p \in \mathbb{R}^{2 \times 1}
\]

\[
y \in \mathbb{R}^{n \times 1}
\]

\[
A^T y \in \mathbb{R}^{2 \times n}
\]

\[
A^T \in \mathbb{R}^{2 \times n}
\]

\[
A^T A \in \mathbb{R}^{2 \times 2}
\]

\[
p = (A^T A)^{-1} A^T y
\]

\[
A^T A \in \mathbb{R}^{2 \times 2}
\]

\[
A^T \in \mathbb{R}^{2 \times 1}
\]

\[
\text{Pseudo Inverse}
\]
Least Squares

Find the line that minimizes the overall sum of squared errors

\[
\epsilon_i = y_i - mx_i - c
\]

Sum of Squared Errors

\[
\epsilon = \sum_{i=1}^{n} (y_i - mx_i - c)^2
\]
Minimize \( \epsilon = \sum_{i=1}^{n} (y_i - mx_i - c)^2 \)

Set partial derivatives equal to 0 and solve for \( m \) and \( c \)

\[
\frac{\partial \epsilon}{\partial m} = 0 \\
\frac{\partial \epsilon}{\partial c} = 0
\]
Minimize \( \epsilon = \sum_{i=1}^{n} (y_i - mx_i - c)^2 \)

\[
\begin{align*}
\epsilon &= \left\| \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} - \begin{bmatrix} x_1 & 1 \\ x_2 & 1 \\ \vdots & \vdots \\ x_n & 1 \end{bmatrix} \begin{bmatrix} m \\ c \end{bmatrix} \right\|^2 \\
&= \|y - Ap\|^2 \\
\end{align*}
\]
\[ E = \min \left\| \begin{bmatrix} y_1 - x_1 m - c \\ \vdots \\ y_n - x_n m - c \end{bmatrix} \right\|_2 \]
Minimize \[ \epsilon = \sum_{i=1}^{n} (y_i - mx_i - c)^2 \]

\[ \epsilon = y^T y - 2(Ap)^T y + (Ap)^T Ap \]

\[ \frac{\partial \epsilon}{\partial p} = -2A^T y + 2A^T Ap \]

Set \[ \frac{\partial \epsilon}{\partial p} \] to 0 and solve for \[ p \]

\[ -2A^T y + 2A^T Ap = 0 \]

\[ \Rightarrow A^T Ap = A^T y \]

\[ \Rightarrow p = (A^T A)^{-1} A^T y \]

Same as before. We were performing least square fitting all along.
Least Squares

How do we compute $\epsilon_i$?

Linear least square fitting cannot handle vertical lines!

Also not rotation invariant
Total Least Squares

Eq. of a line: $ax + by + c = 0$

Find $(a, b, c)$ to minimize the perpendicular distances

$(a, b)$ is the unit normal of the line
Total Least Squares

Minimize $\epsilon = \sum_{i=1}^{n} (ax_i + by_i + c)^2$ s.t. $(a, b)^T (a, b) = 1$

Set $\frac{\partial \epsilon}{\partial c} = 2 \sum_{i=1}^{n} (ax_i + by_i + c)$ to 0

\[ \sum_{i=1}^{n} (ax_i + by_i + c) = 0 \]

\[ \implies a \sum_{i=1}^{n} x_i + b \sum_{i=1}^{n} y_i + nc = 0 \]

\[ \implies c = -\frac{a}{n} \sum_{i=1}^{n} x_i - \frac{b}{n} \sum_{i=1}^{n} y_i = -a\hat{x} - b\hat{y} \]
Total Least Squares

Substituting

\[ c = -\frac{a}{n} \sum_{i=1}^{n} x_i - \frac{b}{n} \sum_{i=1}^{n} y_i \]

\[ \epsilon = \sum_{i=1}^{n} (ax_i + by_i - a\hat{x} - b\hat{y})^2 \]

\[ = \sum_{i=1}^{n} (a(x_i - \hat{x}) + b(y_i - \hat{y}))^2 \]

\[ = \left\| \begin{bmatrix} x_1 - \hat{x} & y_1 - \hat{y} \\ \vdots & \vdots \\ x_n - \hat{x} & y_n - \hat{y} \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} \right\|^2 = \|Ap\|^2 \]

Notice not the same as least squares.
Total Least Squares

Solution

\[ p^* = \arg \min_{p} \| A p \|^2 \text{ s.t. } p^T p = 1 \]

Solution is the eigenvector corresponding to the smallest eigenvalue of \( A^T A \)
Singular Value Decomposition

The Singular Value Decomposition (SVD) of a matrix $A \in \mathbb{R}^{m \times n}$ is defined as

$$A = UDV^T$$

such that

- $U$ is a $m \times n$ matrix with orthogonal columns
- $D$ is a $n \times n$ diagonal matrix. Its entries are called singular values
- $V$ is an $n \times n$ orthogonal matrix
Singular Value Decomposition

The Singular Value Decomposition (SVD) of a matrix \( A \in \mathbb{R}^{m \times n} \) is defined as

\[
A = U D V^T
\]

such that

- \( A \) is a matrix with orthogonal columns
- \( U \) is an orthogonal matrix
- \( D \) is a diagonal matrix. Its entries are called singular values
- \( V \) is an orthogonal matrix

If diagonal entries of \( D \) are sorted as

\[
\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n \geq 0
\]

Last column of \( V \) is the smallest eigenvector of \( A^T A \)
Robust Estimation

Problem: Least Square Estimation is sensitive to outliers

Outliers: points that don’t fit that data
Least Squares

Solution: Use robust least square fitting, RANSAC
Robust Least Squares

\[ \sum_{i=1}^{n} \rho(\epsilon_i, \sigma) \]

where

\[ \rho(\epsilon, \sigma) = \frac{\epsilon^2}{\epsilon^2 + \sigma^2} \]

- Favors a configuration with small residuals
- Constant penalty for large residuals
Robust Least Squares

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

Scale parameter
Robust Least Squares

When the scale is selected just right, the effects of outliers are eliminated.

Scale is too small, so the error value is almost the same for each point. This results in a poor fit.

When the scale is too large, the system behaves like a least squares. It is sensitive to outliers.
Robust Least Squares

- Robust least square fitting is a non-linear optimization problem, which must be solved iterative.
  - The iterative problem can be initialized using least square fitting.
- Scale parameters can be chosen adaptively as well
  - Scale parameter is typically selected to 1.5 times the median residual
Robust Estimation

Which data points are responsible for which model?
Model Fitting

How can we be sure that all relevant data points are available?
RANSAC
RANSAC

• General approach
  • Classify points as inliers and outliers
  • Estimated model parameters using inliers, ignoring outliers
  • Repeat if necessary

RANSAC
RANSAC

Count = 6
RANSAC

Count = 2
RANSAC

Count = 8
RANSAC

Count = 4
RANSAC
Algorithm 15.4: RANSAC: fitting lines using random sample consensus

Determine:

- **s** — the smallest number of points required
- **N** — the number of iterations required
- **d** — the threshold used to identify a point that fits well
- **T** — the number of nearby points required to assert a model fits well

Until **N** iterations have occurred

1. Draw a sample of **s** points from the data uniformly and at random
2. Fit to that set of **s** points
3. For each data point outside the sample
   - Test the distance from the point to the line against **d** if the distance from the point to the line is less than **d** the point is close
   - If there are **T** or more points close to the line then there is a good fit. Refit the line using all these points.

4. Use the best fit from this collection, using the fitting error as a criterion
RANSAC

• How many iterations $N$ are needed to succeed probability equal to $p$?

Probability of sampling an outlier: $e$

Samples required for each model fitting: $s$
RANSAC

• Prob. of selecting an inlier: \(1 - e\)

• Prob. of selecting \(s\) inliers in a row: \((1 - e)^s\)

• Prob. of getting a bad sample: \(1 - (1 - e)^s\)
  
  At least one of those \(s\) selections was an outlier

• Prob. of getting \(N\) bad samples in a row: \((1 - (1 - e)^s)^N\)

• Prob. of getting at least one good sample after \(N\) tries: \(1 - (1 - (1 - e)^s)^N\)
RANSAC

So

\[
p = 1 - (1 - (1 - e)^s)^N
\]

\[
\implies (1 - (1 - e)^s)^N = 1 - p
\]

\[
\implies \log (1 - (1 - e)^s)^N = \log (1 - p)
\]

\[
\implies N \log (1 - (1 - e)^s) = \log (1 - p)
\]

\[
\implies N = \frac{\log (1 - p)}{\log (1 - (1 - e)^s)}
\]
RANSAC

- Pros
  - Robust to outliers
  - Applicable to large number of parameters (than hough transform approaches)
  - Parameters are easier to choose than hough transform

- Cons
  - Not good for getting multiple fits
    - Computational times grows quickly with fraction of outliers and number of parameters that need to be estimated

- Uses
  - Estimating homography (image stitching)
  - Estimating fundamental matrix (relating two views)