

---

# TSBB15

# Computer Vision

## Lecture 10

Recap. of TSBB06,  
Maximum Likelihood and RANSAC



# Prelude to Project 2



Image credit: Bundler home page <http://www.cs.cornell.edu/~snaveily/bundler/>

- Project 2: Start with two views, and successively add more.
- Finally perform simultaneous refinement of 3D point positions and camera positions (Bundle adjustment).

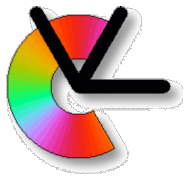


# Prelude to Project 2

---

Highlights of differences compared to Project 1:

- Less programming
- More math skills
- More ways to get lost in 3D  
Unit testing (also of external functions) even more important



# Recap: 2D homogeneous coordinates

---

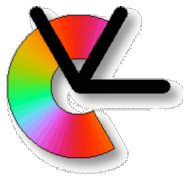
- A 2D point  $(y_1, y_2)$  is given a homogeneous representation as:  $\mathbf{y} \in \mathbb{R}^3$

$$\mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ 1 \end{pmatrix}$$

Canonical form of homogeneous coordinates for a 2D point

- Any scalar multiple of this  $\mathbf{v}$  is also a homogeneous representation of  $(y_1, y_2)$
- We will use  $\mathbf{y}$  to refer to the 2D point as well as its homogeneous coordinates!





# Recap: 2D homogeneous coordinates

---

- A 2D line, at distance  $\rho$  from the origin, and with  $(\cos \alpha, \sin \alpha)$  as a normal vector is given a (dual) homogeneous representation as  $\mathbf{l} \in \mathbb{R}^3$ :

$$\mathbf{l} = \begin{pmatrix} \cos \alpha \\ \sin \alpha \\ -\rho \end{pmatrix}$$

Canonical form of homogeneous coordinates for a 2D line

- Any scalar multiple of this  $\mathbf{l}$  is also a homogeneous representation of the same line
- We will use  $\mathbf{l}$  to refer to the 2D line as well as its homogeneous coordinates!



# Recap: 2D homogeneous coordinates

---

- From these homogeneous representations follow:
- Point  $\mathbf{y}$  lies on line  $\mathbf{l} \Leftrightarrow \mathbf{y} \cdot \mathbf{l} = 0$
- Point  $\mathbf{y}$  intersects lines  $\mathbf{l}_1$  and  $\mathbf{l}_2 \Leftrightarrow \mathbf{y} = \mathbf{l}_1 \times \mathbf{l}_2$
- Line  $\mathbf{l}$  intersects points  $\mathbf{y}_1$  and  $\mathbf{y}_2 \Leftrightarrow \mathbf{l} = \mathbf{y}_1 \times \mathbf{y}_2$
  
- The distance from point  $\mathbf{y}$  to line  $\mathbf{l}$  is given by  $|\mathbf{y} \cdot \mathbf{l}|$  *if they both are in canonical form*



# Recap: 2D homogeneous coordinates

- Rigid transformations: translation + rotation
- Scaling
- Affine transformations
- Projective transformations (homographies)

Can be represented as linear mappings of the homogeneous coordinates:

$$\mathbf{y}' = \mathbf{H}\mathbf{y} \quad \mathbf{l}' = \mathbf{H}^{-T}\mathbf{l} \quad \leftarrow$$

Dual  
transformations  
of lines!



# Recap: Cross product operator

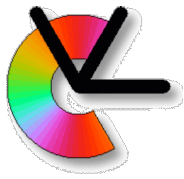
---

The cross product between vectors **a** and **b**:

$$\mathbf{a} \times \mathbf{b}$$

can sometimes be written more conveniently as a 3×3 matrix  $[\mathbf{a}]_{\times}$  applied to **b**:

$$\mathbf{a} \times \mathbf{b} = [\mathbf{a}]_{\times} \mathbf{b}$$



# Recap: 3D homogeneous coordinates

---

- Homogeneous coordinates for 2D points and lines can be extended in a straightforward way to homogeneous coordinates for 3D points  $\mathbf{x}$  and planes  $\mathbf{p}$
- Also 3D lines can be given a homogeneous representation
  - Parameter form:  $t \mathbf{x}_1 + (t - 1) \mathbf{x}_2$
  - Plücker coordinates:  $\mathbf{L} = \mathbf{x}_1 \mathbf{x}_2^T - \mathbf{x}_2 \mathbf{x}_1^T$



# Recap: Pinhole camera

---

- The pinhole camera maps 3D points to a 2D image:

$$\mathbf{y} \sim \mathbf{C}\mathbf{x}$$

- $\mathbf{C}$  is the  $3 \times 4$  camera (projection) matrix
- Does not represent geometric distortion from the camera lens



# Recap: Pinhole camera

- The camera matrix can be decomposed as

$$\mathbf{C} = \begin{pmatrix} f_1 & \gamma & c_1 \\ 0 & f_2 & c_2 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Intrinsic camera  
calibration matrix  
3×3

Normalized  
camera projection  
matrix  
3×4

Extrinsic camera  
calibration matrix  
(a rigid transformation)  
4×4



# Recap: Pinhole camera

---

In a more compact form:

$$\mathbf{C} = \mathbf{K} [\mathbf{R} \mid \mathbf{t}] \quad [\mathbf{R} \mid \mathbf{t}] \text{ is } 3 \times 4$$

$\mathbf{K}$  = **intrinsic calibration** matrix ( $3 \times 3$ )

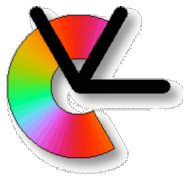
Constant in this project

Camera calibration

$\mathbf{R}, \mathbf{t}$  = **extrinsic calibration**

The camera pose for each image in the sequence





# Recap: Camera center

---

The pinhole camera projects 3D point onto the image plane through its center  $\mathbf{n}$ :

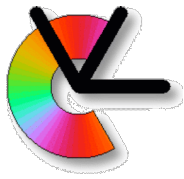
The camera centre  $\mathbf{n} \in \mathbb{R}^4$  satisfies  $\mathbf{C} \mathbf{n} = \mathbf{0}$

If  $\mathbf{C} = \mathbf{K} [\mathbf{R} \mid \mathbf{t}] \Rightarrow$

The 3D coordinates of  $\mathbf{n}$  is  $\mathbf{t}' = -\mathbf{R}^T \mathbf{t}$

Alternatively: if  $\mathbf{C} = \mathbf{K} [\mathbf{R} \mid -\mathbf{R} \mathbf{t}] \Rightarrow$

The 3D coordinates of  $\mathbf{n}$  is  $\mathbf{t}$



# A practical issue

---

The numerical values of a specific camera matrix always **refer to a specific coordinate system** in the image and in 3D space

There may be more than one coordinate system that is used for particular problem:

- A camera centred coordinate system
- Another camera coordinate system
- The world coordinate system

**We need to know which coordinate system it refers to** in order to use it practically

Maybe transform it to a standard coordinate system



# Recap: Equivalent cameras

---

- Two cameras  $\mathbf{C}_1$  and  $\mathbf{C}_2$  with identical camera center  $\mathbf{n}$ :  $\mathbf{C}_1 \mathbf{n} = \mathbf{C}_2 \mathbf{n} = 0$  are called **equivalent**
- The images of two equivalent cameras can always be made identical by means of a homography transformation  
Disregarding that images are of finite size
- Cameras that are not equivalent are called **distinct**



# Recap: Estimation

---

Many problems where we estimate some type of geometric object can be formulated as

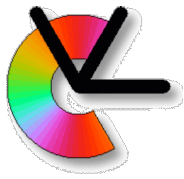
$$\mathbf{A} \mathbf{x} = \mathbf{0}$$

$\mathbf{x}$  is a representation of the unknown geometric object that we want to determine

Usually in terms of homogeneous coordinates

Or, at least, in terms of a projective element

$\mathbf{A}$  is data dependent



# Recap: Estimation

---

In practice, the data in  $\mathbf{A}$  includes noise

The equation  $\mathbf{A} \mathbf{x} = \mathbf{0}$  is not satisfied exactly

Solution (homogeneous method):

minimise:

$$\epsilon(\mathbf{x}) = \|\mathbf{A}\mathbf{x}\| \text{ subject to } \|\mathbf{x}\| = 1$$

This approach minimises an **algebraic error**

Optimal for Gaussian residuals, i.e.  $\mathbf{A}\mathbf{x} \sim \mathcal{N}(\mathbf{0}, \sigma\mathbf{I})$

This is **not** the case in general...



# Recap: Estimation

---

- Algebraic errors lead to linear solution methods (e.g. using SVD), they are simple to use
- The actual data noise, however, occurs in the Euclidean geometric space (2D or 3D)
- Geometric errors are almost always non-linear functions of the free parameters that we optimise
  - No closed form solutions exist
  - Iterative minimisation methods must be used
  - Good initial solutions are critical (use linear estimators!)
  - Use optimisation tools, e.g., in **scipy.optimize**
  - To be continued...



# Recap: Epipolar geometry

A 3D point  $\mathbf{x}$  is viewed by two distinct cameras  $\mathbf{C}_1$  and  $\mathbf{C}_2$

$$\mathbf{y}_1 \sim \mathbf{C}_1 \mathbf{x}$$

$$\mathbf{y}_2 \sim \mathbf{C}_2 \mathbf{x}$$

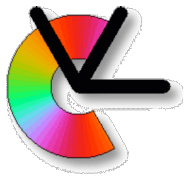


$\mathbf{y}_1$  and  $\mathbf{y}_2$  are  
*corresponding  
points*

Either:

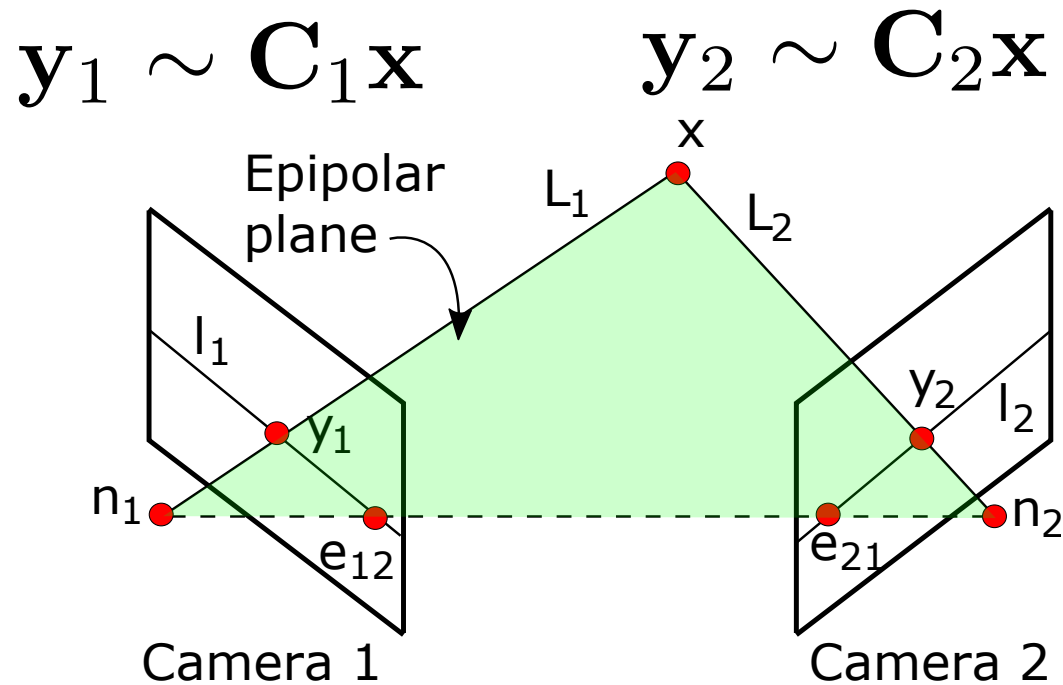
Two physically distinct cameras

The same camera that is moving over time

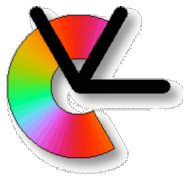


# Recap: Epipolar geometry

- A 3D point  $\mathbf{x}$  is viewed by two distinct cameras  $\mathbf{C}_1$  and  $\mathbf{C}_2$







# Recap: Epipolar points

---

The image of camera center  $\mathbf{n}_1$  in camera 2 is called the **epipole** (or *epipolar point*)  $\mathbf{e}_{21}$ :

$$\mathbf{e}_{21} = \mathbf{C}_2 \mathbf{n}_1$$

Correspondingly, the image of  $\mathbf{n}_2$  in camera 1 defines the epipole  $\mathbf{e}_{12}$ :

$$\mathbf{e}_{12} = \mathbf{C}_1 \mathbf{n}_2$$



# Recap: Epipolar constraint

In this case, it follows that there exists a  $3 \times 3$  matrix  $\mathbf{F}$ , the **fundamental matrix**, such that

$$\mathbf{y}_1^T \mathbf{F} \mathbf{y}_2 = 0$$

The epipolar constraint between *corresponding image points*  $\mathbf{y}_1$  and  $\mathbf{y}_2$

$\mathbf{F}$  depends only on  $\mathbf{C}_1$  and  $\mathbf{C}_2$ :

$$\mathbf{F} = [\mathbf{e}_{12}]_{\times} \mathbf{C}_1 \mathbf{C}_2^+$$

Pseudo-inverse

Calibrated case:  $\mathbf{F}$  is determined from known  $\mathbf{C}_1$  and  $\mathbf{C}_2$



# Recap: epipolar lines

---

$\mathbf{F}$  maps  $\mathbf{y}_2$  to a line  $\mathbf{l}_1 = \mathbf{F}\mathbf{y}_2$  in the first view

$\mathbf{l}_1$  is an **epipolar line**:

it goes through the epipole,  $\mathbf{e}_{12} \cdot \mathbf{l}_1 = 0$

If  $\mathbf{y}_1$  corresponds to  $\mathbf{y}_2$ :

then  $\mathbf{y}_1$  lies on  $\mathbf{l}_1$ :  $\mathbf{y}_1 \cdot \mathbf{l}_1 = 0$

Similarly for the other epipolar line  $\mathbf{l}_2 = \mathbf{F}^T \mathbf{y}_1$



# Recap: 8-point algorithm

---

**F** can also easily be estimated from a set of 8 (or more) corresponding image points:

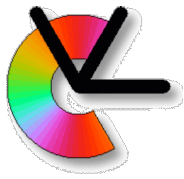
Form a data matrix **A**

Determine right null vector **f** of **A** (SVD)

Reshape **f** to **F**

Enforce internal constraint  $\det \mathbf{F} = 0$

Important to use Hartley normalisation to increase accuracy!



# Recap: triangulation

---

Given two corresponding image points  $\mathbf{y}_1$  and  $\mathbf{y}_2$  we want to determine the 3D point  $\mathbf{x}$

Mid-point method

Algebraic method

Probabilistic method ("optimal" triangulation)

In general, different methods give slightly different results if  $\mathbf{y}_1$  and  $\mathbf{y}_2$  are noisy



# BREAK





# Maximum Likelihood

---

- For normally distributed noise we can often define a maximum likelihood solution to estimation problems
- Consider a direct observation  $\mathbf{x}$ :

$$x \sim \mathcal{N}(\mu, \sigma)$$

- with the observation **likelihood**:

$$Pr(x|\mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-0.5(x-\mu)^2/\sigma^2}$$



# Maximum Likelihood

---

- If we have several **independent** observations:

$$x_1, \dots, x_N \sim \mathcal{N}(\mu, \sigma)$$

- Their joint likelihood becomes:

$$Pr(x_1|\mu, \sigma) \cdot \dots \cdot Pr(x_N|\mu, \sigma) = \prod_{n=1}^N Pr(x_n|\mu, \sigma)$$

- We can now consider the estimation problem as one of finding the parameters that maximise this joint likelihood.





# Maximum Likelihood

---

- Maximise the joint likelihood:

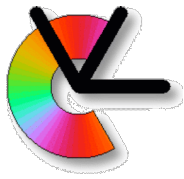
$$\{\mu^*, \sigma^*\} = \arg \max_{\mu, \sigma} \prod_{n=1}^N Pr(x_n | \mu, \sigma)$$

- Take the negative log of this expression:

$$\{\mu^*, \sigma^*\} = \arg \min_{\mu, \sigma} J(\mu, \sigma)$$

- For:

$$J(\mu, \sigma) = \frac{1}{2} \sum_{n=1}^N (x_n - \mu)^2 / \sigma^2 - \frac{N}{2} \log(2\pi\sigma^2)$$



# Maximum Likelihood

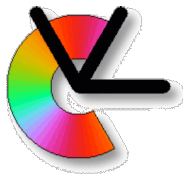
- Setting partial derivatives to zero gives us estimates:

$$\frac{\partial J(\mu, \sigma)}{\partial \mu} = 0 \quad \Rightarrow \quad \mu = \frac{1}{N} \sum_{n=1}^N x_i$$

- and

$$\frac{\partial J(\mu, \sigma)}{\partial \sigma} = 0 \quad \Rightarrow \quad \sigma^2 = \frac{1}{N} \sum_{n=1}^N (x_i - \mu)^2$$

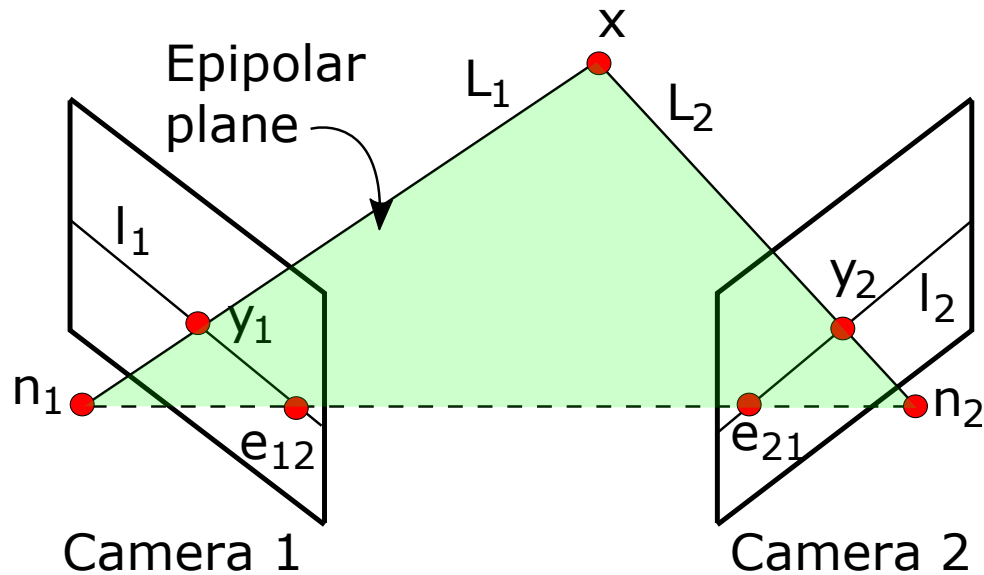
True  $\mu$ !

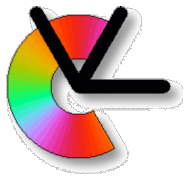


# Optimal triangulation

- Given two cameras and two observations of the same 3D point:

$$\mathbf{y}_1 \sim \mathbf{C}_1 \mathbf{x} \quad \text{and} \quad \mathbf{y}_2 \sim \mathbf{C}_2 \mathbf{x}$$





# Optimal triangulation

---

- Given two cameras and two observations of the same 3D point:

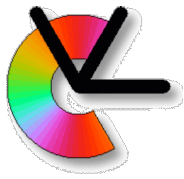
$$\mathbf{y}_1 \sim \mathbf{C}_1 \mathbf{x} \quad \text{and} \quad \mathbf{y}_2 \sim \mathbf{C}_2 \mathbf{x}$$

- Gives us the joint likelihood to maximize:

$$\mathbf{x}^* = \arg \max_{\mathbf{x}} Pr(\mathbf{y}_1 | \mathbf{C}_1 \mathbf{x}) Pr(\mathbf{y}_2 | \mathbf{C}_2 \mathbf{x})$$

- Negative log gives us a least squares problem:

$$\mathbf{x}^* = \arg \min_{\mathbf{x}} d^2(\mathbf{y}_1, \mathbf{C}_1 \mathbf{x}) + d^2(\mathbf{y}_2, \mathbf{C}_2 \mathbf{x})$$



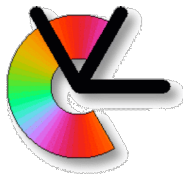
# Optimal triangulation

---

- A least squares problem:

$$\mathbf{x}^* = \arg \min_{\mathbf{x}} d^2(\mathbf{y}_1, \mathbf{C}_1 \mathbf{x}) + d^2(\mathbf{y}_2, \mathbf{C}_2 \mathbf{x})$$

- Each 3D point  $\mathbf{x}$  defines an epipolar plane, which defines two epipolar lines.
- We can look for an epipolar line in one image, and transfer it to the other image.
- This leads to a 6th degree polynomial.
- For the best line pair, find the closest auxiliary points ( $\mathbf{y}_1' = \mathbf{C}_1 \mathbf{x}$  and  $\mathbf{y}_2' = \mathbf{C}_2 \mathbf{x}$ )
- Triangulate these, using linear triangulation (OK as these satisfy the EG perfectly, and have zero reprojection error)



# Optimal triangulation

---

- You will use optimal triangulation in CE3.
- Source code is provided.
- Full algorithm is described in IREG 16.3
- More details on the derivation are given there.



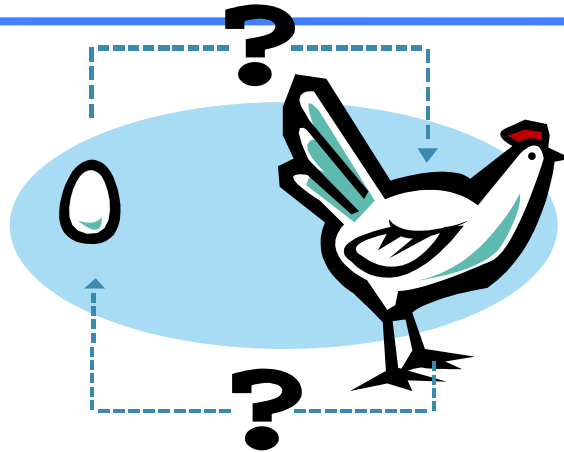
# Robust estimation

---

- We often have two categories of errors:
  - **Measurement noise**: we cannot determine numerical values of the data with high accuracy
  - **Outliers**: all measurements in the dataset cannot be fitted to a consistent model
- We refer to data that can be fitted to the model as **inliers**
- **Robust estimation**: to determine a model from a dataset that contains significant amount of outliers



# A chicken and egg problem



We need corresponding points to estimate  $F$

Point correspondences can be verified if we know  $F$

Can we determine  $F$  and verify correspondences at the same time?

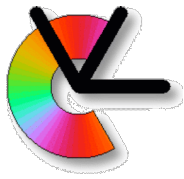




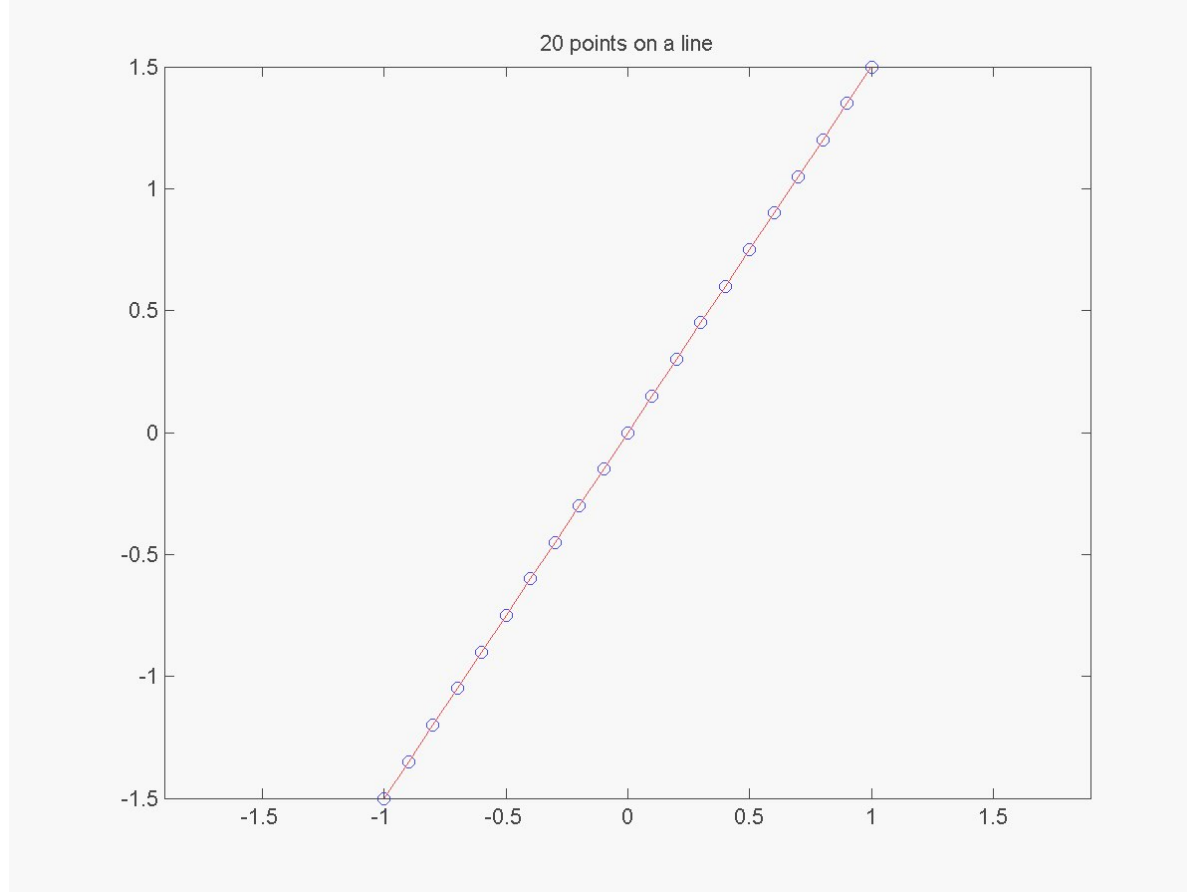
# Robust estimation

Model estimation from measured data where some of the data items are incorrect. E.g.

Data	Model
Points in 2D or in 3D	A 2D line, A 3D plane
Lines in 2D or planes in 3D	Point of intersection in 2D or 3D
Corresponding points in 2 views	Homography
Corresponding points in 2 views	Fundamental matrix

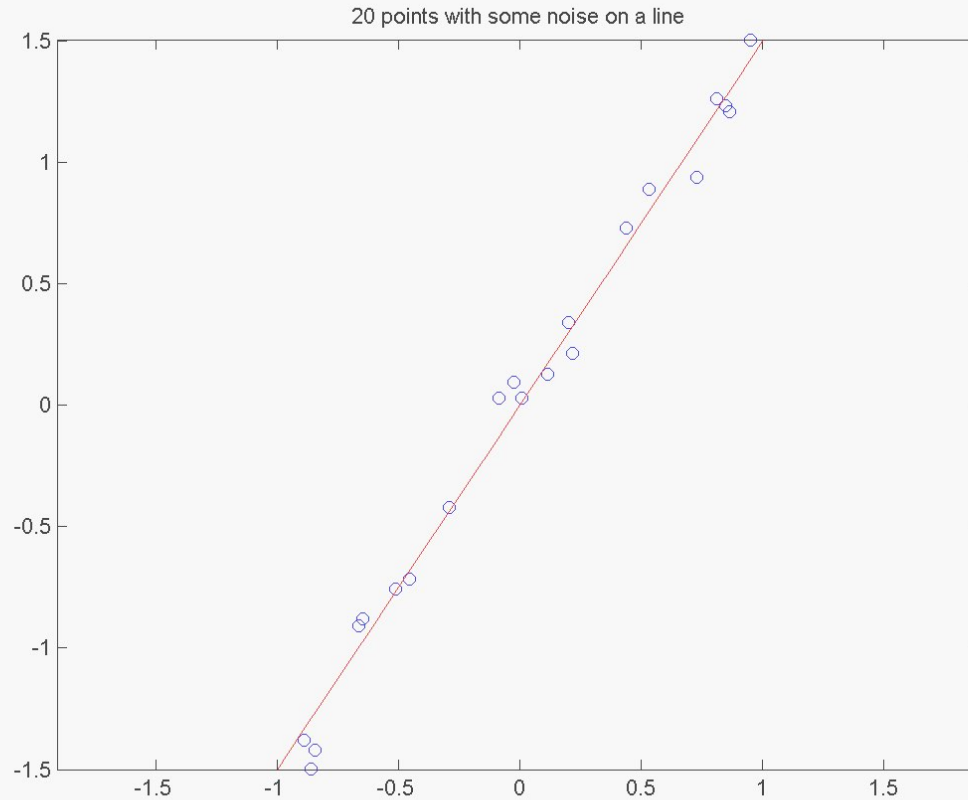


# Example: estimation of a line from points



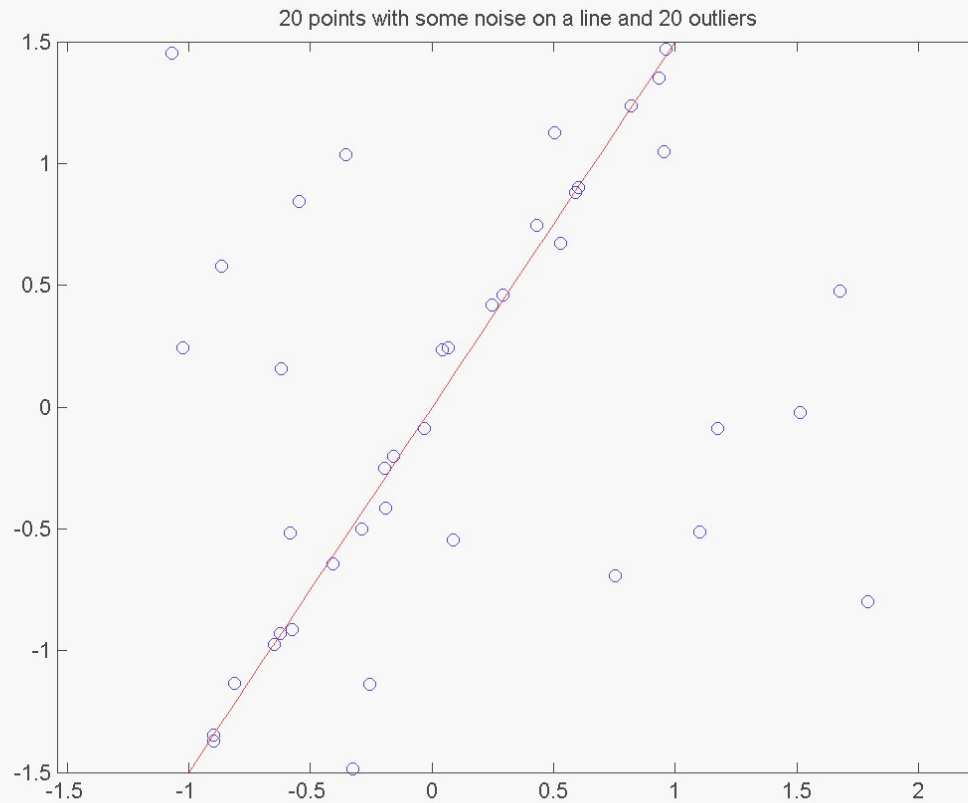


# Example: estimation of a line from points





# Example: estimation of a line from points





# Observations

---

We need (in this case!) a minimum of 2 points to determine a line

Given such a line  $\mathbf{l}$ , we can determine how well any other point  $\mathbf{y}$  fits the line  $\mathbf{l}$

For example: distance between  $\mathbf{y}$  and  $\mathbf{l}$

If we pick 2 **random** points from the dataset:

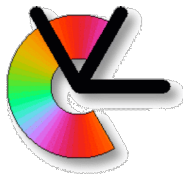
We can easily determine a line  $\mathbf{l}$

$\mathbf{l}$  is the correct line with some probability  $p_{\text{LINE}}$

$p_{\text{LINE}}$  is related to the chance of picking only inliers

$p_{\text{LINE}}$  is larger the fewer points that are used to determine  $\mathbf{l}$

In general: if  $\mathbf{l}$  is supported by the data there are more additional points that lie on it.



# Probabilities

---

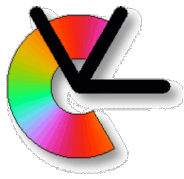
Let  $S$  be a set of  $|S|=M$  points in total and  $M_0$  of them are inliers

There are  $M(M-1)$  ways to draw 2 distinct points in a certain order, ( $\approx M^2$  if  $M$  is large)

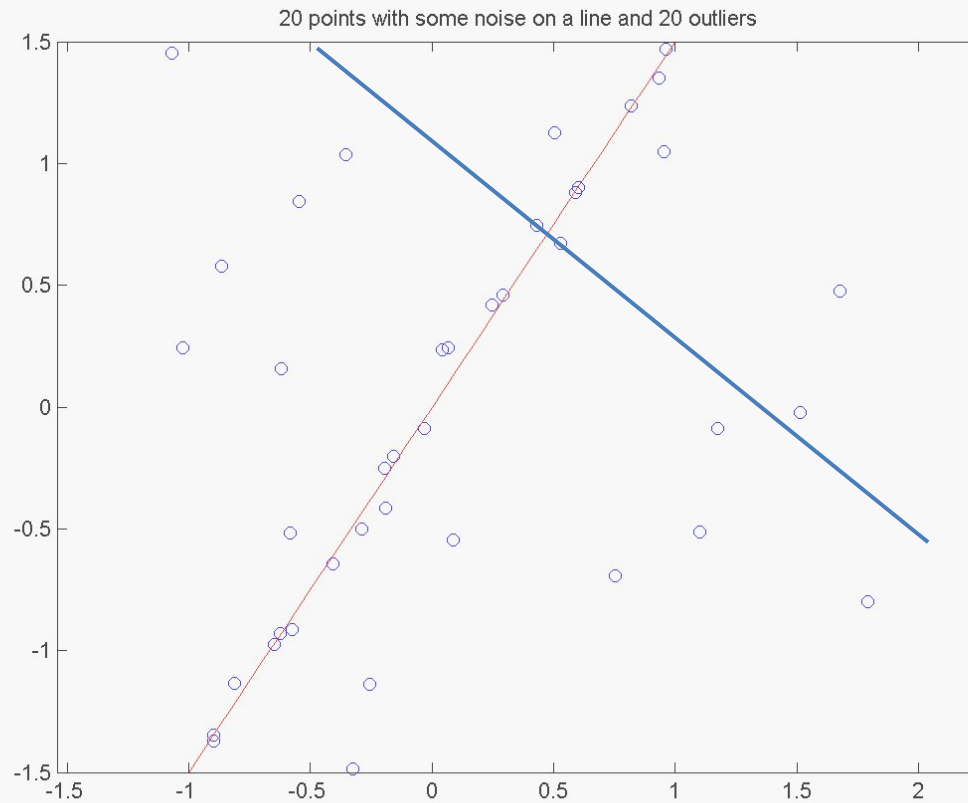
This means  $p_{\text{LINE}} \approx p_{\text{POINT}}^2 = (M_0/M)^2$

This is in practice **an upper bound**:

Even if we pick two inliers, the corresponding line may not be supported by the remaining inliers



# Line estimated from 2 inliers



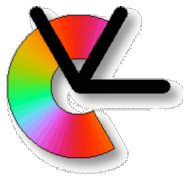


# Basic iteration

---

1. Draw 2 random points from  $S$
2. Fit a line  $l$  to the points
3. Determine how many other points<sup>2</sup> in  $S$  that support the line  $l$  within some error bound.  
<sup>2</sup> These form the **consensus set**  $C$
4. If  $C$  is sufficiently large, then the found line is probably OK. Keep it

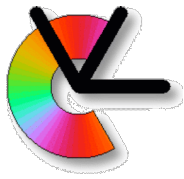




# Basic algorithm

---

- Iterate  $r$  times
  1. Draw 2 random points from  $S$
  2. Fit a line  $l$  to the points
  3. Form the consensus set  $C$ , together and  
Count the number of points in  $C$   
(or  $p(S|l)$  average likelihood of the dataset given the line)
  4. If the consensus set is sufficiently large, then the found line is OK. In particular, keep  $l$  if  $N$  or  $p(S|l)$  is the best one this far.
- Each iteration increases  $p_{\text{SUCCESS}}$  = the probability that the correct line has been found
- We need to iterate sufficiently many time to raise  $p_{\text{SUCCESS}}$  to a useful level



# RANSAC

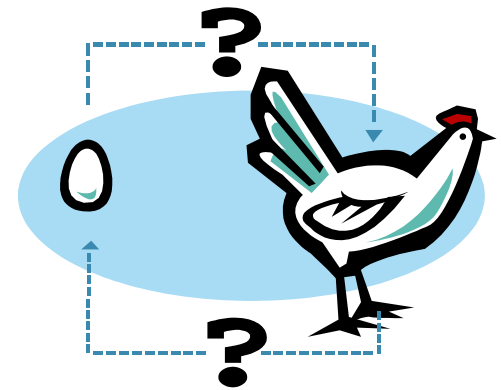
---

- This algorithm is called **RANSAC**
  - RANdOm SAmple Consensus
- Published by Fischler & Bolles in 1981
  - "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography". *Comm. of the ACM* **24**: 381–395.
- Several extensions / variations in the literature
  - Preemptive RANSAC, MLESAC, PROSAC, ...



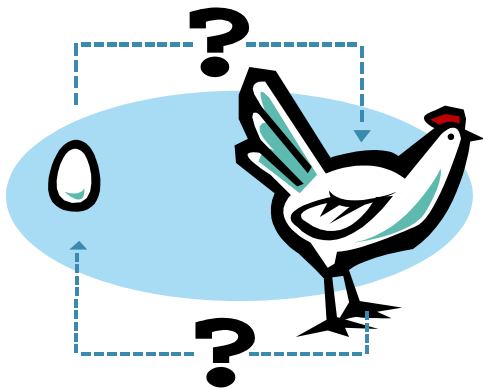
# RANSAC for $F$

- For estimation of the fundamental matrix  $F$ , points that satisfy an epipolar constraint for some  $F$  are **assumed to be inliers**
- The points that violate the epipolar constraint are **outliers**





# Chicken and egg revisited



- Let there be two views with a point set  $P_1$  in one view and  $P_2$  in the other view
- Drawing correspondences randomly from the two point sets is futile:  
 $\Pr(\text{correct}) = \min(1/|P_1|, 1/|P_2|)$ .
- We need to find a subset  $\mathcal{S} \subset \mathcal{P}_1 \times \mathcal{P}_2$  of likely correspondences, or **tentative** correspondences
- Typically done using some heuristic



# RANSAC for $\mathbf{F}$

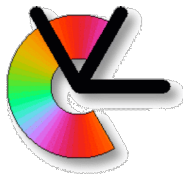
---

Pick 8 random correspondences from  $S$

We do not know if they really correspond, but this can be tested:

1. Use the 8-point algorithm to estimate  $\mathbf{F}$
2. Check how well  $\mathbf{F}$  matches each pair in  $S$
3. Collect those that fit well into the consensus set  $C$
4. If  $C$  is sufficiently large:  $\mathbf{F}$  is OK: keep  $\mathbf{F}$  and  $C$

Iterate  $r$  times



# Probability of success

---

Let  $w$  be the fraction of inliers in  $S$

In each iteration we pick  $N$  points that are all inliers with probability  $w^N$  (approximately)

The probability of not all  $N$  points are inliers is then given by  $1 - w^N$

The probability of not all  $N$  points are inliers in  $r$  iterations is  $(1 - w^N)^r$

The probability that in iteration  $r$ , at least once, all  $N$  points are inliers:  $p = 1 - (1 - w^N)^r$

Solve for  $r$ :

$$r = \frac{\log(1 - p)}{\log(1 - w^N)}$$



# Minimising geometric errors

---

- The 8-point algorithm uses an algebraic error. We would like to also estimate  $\mathbf{F}$  using ML.
- This requires us to use an error that corresponds to a likelihood.
- We know that image coordinates from detection are well modelled as normally distributed. Via the negative log we can then obtain a least-squares problem.
- Thus, we can look for a geometric error, but will all geometric errors lead to independent normally distributed likelihoods?



# Geometric errors for $\mathbf{F}$

Given a set of  $N$  corresponding image points  
 $\{\mathbf{y}_{1k}, \mathbf{y}_{2k}\}, k = 1, \dots, N$

We can formulate a geometric error as

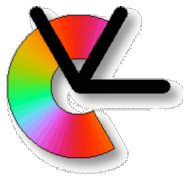
$$\epsilon_1 = \sum_{k=1}^N d_{PL}(\mathbf{y}_{1k}, \mathbf{F}\mathbf{y}_{2k})^2 + d_{PL}(\mathbf{y}_{2k}, \mathbf{F}^T\mathbf{y}_{1k})^2$$

Epipolar lines!

where  $d_{PL}$  is the Euclidean distance between a point and a line

Referred to as an  $L_2$  error





# Geometric errors for $\mathbf{F}$

This error does not take into account that the epipolar lines also, are perturbed by noise

As an alternative, we want to find

A fundamental matrix  $\mathbf{F}$

A set of auxiliary points  $\{\mathbf{y}'_{1k}, \mathbf{y}'_{2k}\}$

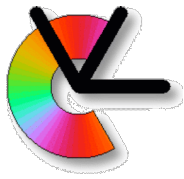
where  $(\mathbf{y}'_{1k})^T \mathbf{F} \mathbf{y}'_{2k} = 0$  (exactly!) for all  $k$

such that

$$\epsilon = \sum_{k=1}^N d_{PP}(\mathbf{y}_{1k}, \mathbf{y}'_{1k})^2 + d_{PP}(\mathbf{y}_{2k}, \mathbf{y}'_{2k})^2$$

is minimised

$d_{pp}$  is the Euclidean point-to-point distance



# Re-parameterisation of $\mathbf{F}$

---

This means that we want to optimise over combinations of  $\mathbf{F}$  *and* the auxiliary points that always satisfy the epipolar constraint!

Cannot be done directly in a simple way

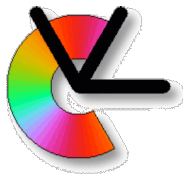
Instead:

Let  $\mathbf{F}$  be determined from  $\mathbf{C}_1$  and  $\mathbf{C}_2$

Let the auxiliary points be the images of some *virtual* 3D points projected through  $\mathbf{C}_1$  and  $\mathbf{C}_2$

The auxiliary points and  $\mathbf{F}$  are then always consistent

Vary the 3D points and  $\mathbf{C}_1$  and  $\mathbf{C}_2$



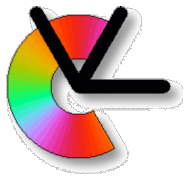
# Cameras from $\mathbf{F}$

---

Each camera matrix has 11 degrees of freedom: two cameras have 22 DOF

$\mathbf{F}$  has 7 DOF

There are many combinations of two cameras that produce the same  $\mathbf{F}$



# Cameras from $\mathbf{F}$

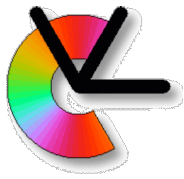
Choose  $\mathbf{C}_1 = [\mathbf{I} \mid \mathbf{0}]$  (set WCS=CCS for camera 1)

A convenient choice of  $\mathbf{C}_2$  is then given by

$$\mathbf{C}_2 = \left[ [\mathbf{e}_{21}]_{\times} \mathbf{F}^T \mid \mathbf{e}_{21} \right] \quad (\text{why?})$$

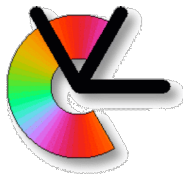
Note that  $\mathbf{e}_{21}$  is given by  $\mathbf{F} \mathbf{e}_{21} = \mathbf{0}$

$\mathbf{C}'_1 = \mathbf{C}_1 \mathbf{H}$  and  $\mathbf{C}'_2 = \mathbf{C}_2 \mathbf{H}$ , for arbitrary 3D homography  $\mathbf{H}$ , also gives the same  $\mathbf{F}$  (why?)



# Gold Standard estimation of $\mathbf{F}$

- Determine an initial  $\mathbf{F}_0$  using a linear method
- From  $\mathbf{F}_0$ : determine initial  $\mathbf{C}_1$  and  $\mathbf{C}_2$
- From  $\{\mathbf{y}_{1k}, \mathbf{y}_{2k}\}$  and  $\mathbf{C}_1, \mathbf{C}_2$ , triangulate virtual 3D points  $\mathbf{x}_k$ 
  - Relative to some arbitrary 3D coordinate system
  - Projective triangulation
- Re-project the 3D points through  $\mathbf{C}_1$  and  $\mathbf{C}_2$  to the auxiliary points  $\{\mathbf{y}'_{1k}, \mathbf{y}'_{2k}\}$ 
  - These are consistent with  $\mathbf{F}_0$
- Minimise  $\mathcal{E}$  over the 3D points  $\mathbf{x}_k$  and  $\mathbf{C}_2$  ( $\mathbf{C}_1$  is fixed)
  - Use a non-linear optimisation tool, e.g., **lsqnonlin**
  - $\mathcal{E}$  is referred to as an  $L_2$  **reprojection error**
- The resulting  $\mathbf{C}_2$  (+  $\mathbf{C}_1$ ) gives  $\mathbf{F}_{\text{optimal}}$  that
  - Minimises a geometric error
  - Maximises the likelihood of the correspondences under Gaussian noise
  - This is referred to as **Gold Standard estimation** of  $\mathbf{F}$
- Computer Exercise 3



# Summary

---

- The math introduced here and in the next two lectures builds on **TSBB06 Multidimensional Signal Analysis**
- The **Maximum Likelihood**(ML) trick is to look for models that **make observations likely**.
- **Optimal Triangulation**, and the **Gold Standard method** to find **F** are two examples of ML. More examples will follow...
- The **RANSAC algorithm** is an important tool for handling data with **outliers**. Other tools are clustering and mode finding (LE6).