



---

# TSBB15

# Computer Vision

## Lecture 11

## More RANSAC and Calibrated Geometry



# RANSAC: Random Sample Consensus

---

Iterate  $r$  times

1. Pick  $n$  random points from  $D$
2. Determine a model  $M$  from these points
3. Form the consensus set  $C$ , together with
  - Number of points in  $C$  (i.e.  $|C|$ )
  - Average likelihood of the elements in set  $C$  given the model  $M$
4. If  $C$  is larger than ever before, then keep this model.

After the iterations: the best kept model is the RANSAC model estimate



# RANSAC: Random Sample Consensus

Iterate  $r$  times

1. Pick  $n$  random points from  $D$  ← **n should be small!**
2. Determine a model  $M$  from these points ← **Must be fast!**
3. Form the consensus set  $C$ , together with
  - Number of points in  $C$  (i.e.  $|C|$ )
  - Average likelihood of the elements in set  $C$  given the model  $M$
4. If  $C$  is larger than ever before, then keep this model.

After the iterations: the best kept model is the RANSAC model estimate



# RANSAC: Random Sample Consensus

---

- A **minimal solver** is an algorithm that finds a solution to a geometric problem using the smallest possible number of points.
- E.g. for line fitting, draw 2 points, and use cross-product to find a line.
- There is a large body of work that studies efficient minimal solvers for specific problems. These are intended for use with RANSAC.
- The 8pt algorithm is not minimal (7 pts is enough).



# Variations/Extensions

---

After RANSAC is done, we can:

1. optionally re-estimate the found model from  $C$ , using a **more accurate** estimation method.

**Beware:** Re-estimation of  $F$  using the 8-point algorithm may degrade the solution (the size of  $C$  could shrink).

2. use the found model to look for more correspondences.



# The 8-point algorithm revisited

---

- The epipolar constraint

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

- defines one linear constraint on  $\mathbf{F}$  for each pair of corresponding points  $\mathbf{y}_1$  and  $\mathbf{y}_2$
- The 8-point algorithm uses  $n \geq 8$  such constraints to determine  $\mathbf{F}$
- If  $n = 8$ , the data matrix  $\mathbf{A}$  has a well-defined 1-dimensional null space that contains  $\mathbf{F}$ 
  - May not satisfy  $\det \mathbf{F} = 0$
  - This condition can be enforced!



# The 7-point algorithm

- If there are only 7 point constraints  
⇒ The null space  $Null(\mathbf{A})$  of  $\mathbf{A}$  is 2-dimensional

- $Null(\mathbf{A})$  is spanned by  $\mathbf{f}_1$  and  $\mathbf{f}_2$  ← Two 9-dim vectors

- $\text{vec}(\mathbf{F})$  lies somewhere in this null space, i.e.:

$$\mathbf{F} = \alpha \mathbf{F}_1 + (1 - \alpha) \mathbf{F}_2$$

- Use *internal constraint*  $\det \mathbf{F} = 0$  to determine  $\mathbf{F}$ :

$$\det(\alpha \mathbf{F}_1 + (1 - \alpha) \mathbf{F}_2) = 0$$

- This is a third order polynomial in  $\alpha$  (why?)  
⇒ 1 or 3 real solutions



# The 7-point algorithm

---

In summary:

Use exactly 7 correspondences to build  $\mathbf{A}$

Determine basis  $\mathbf{f}_1, \mathbf{f}_2$  of  $\text{Null}(\mathbf{A})$  (SVD)

Reshape, and solve  $\det(\alpha \mathbf{F}_1 + (1 - \alpha) \mathbf{F}_2) = 0$

Gives 1 or 3 real solutions

For each solution  $\alpha$  assemble  $\mathbf{F}$  as:

$$\mathbf{F} = \alpha \mathbf{F}_1 + (1 - \alpha) \mathbf{F}_2$$





# The 7-point algorithm

---

Pros and cons:

- + Only 7 correspondences needed
  - ⇒ Smaller  $r$  for same  $w$
- + No constraint enforcement needed (**why?**)
- + No Hartley normalization needed (**why?**)
- + Slightly more accurate solution
- Slightly more complicated calculations
- Multiple solutions (1 – 3)



# RANSAC speedup

---

**Example:**  $w = 0.5$  and  $p = 0.99$

8-point:  $r = \log(1-p) / \log(1-w^8) \approx 1180$

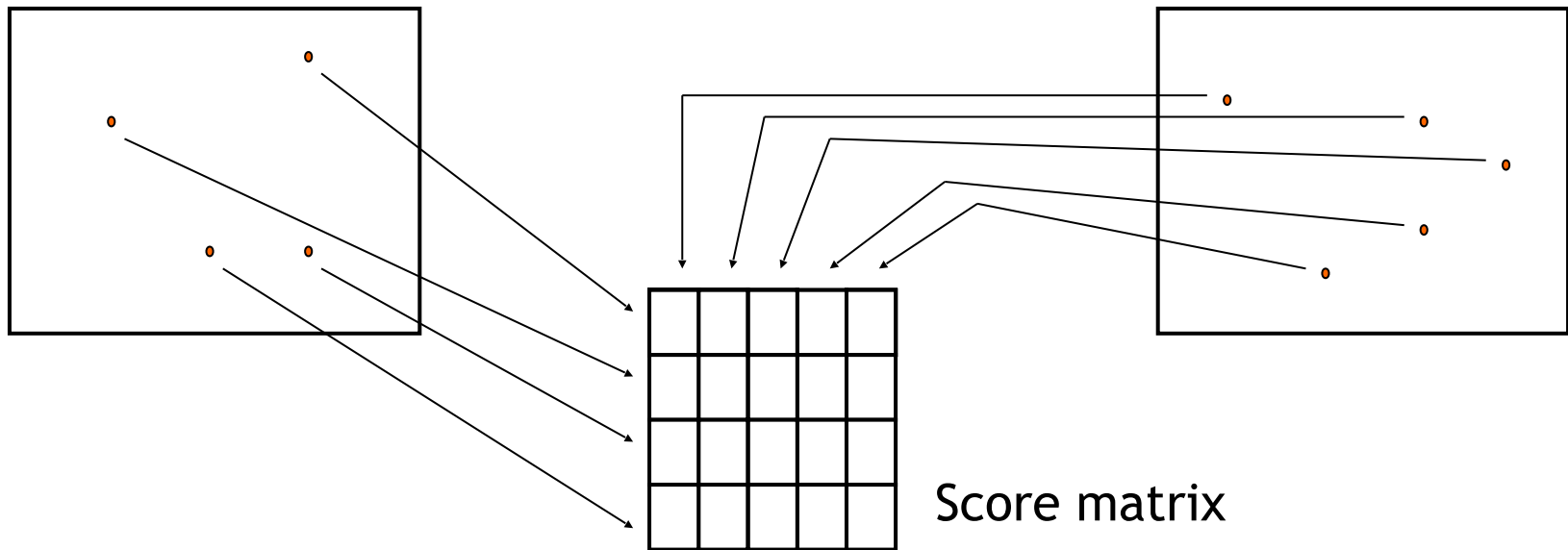
7-point:  $r = \log(1-p) / \log(1-w^7) \approx 590$

Caveats:

1. In the 7-point case we must test up to 3 possible  $F$  which makes each iteration slightly more computationally expensive  $\Rightarrow$  less than 50% speedup
2. The 7pt method is more accurate, and thus the  $r$  value is underestimated more for the 8pt method



# Tentative Matches



Each entry in the matching matrix describes how well a certain point in image 1 matches another point in image 2.  
For example: high score = good match



# Brute force matching

---

- Given  $P_1$  points in image 1 and  $P_2$  points in image 2
  - Form a  $P_1 \times P_2$  *matching matrix*
  - Each entry  $(i,j)$  is a hypothetical correspondence between point  $i$  in image 1 and point  $j$  in image 2
- Set entry  $(i,j) =$   
a matching score between point  $i$  and point  $j$
- For each column and/or row: keep only the largest entry
  - $w$  increases  $\Rightarrow r$  decreases for fixed  $p$
- Such **tentative correspondences** are the input to RANSAC [See CE3]



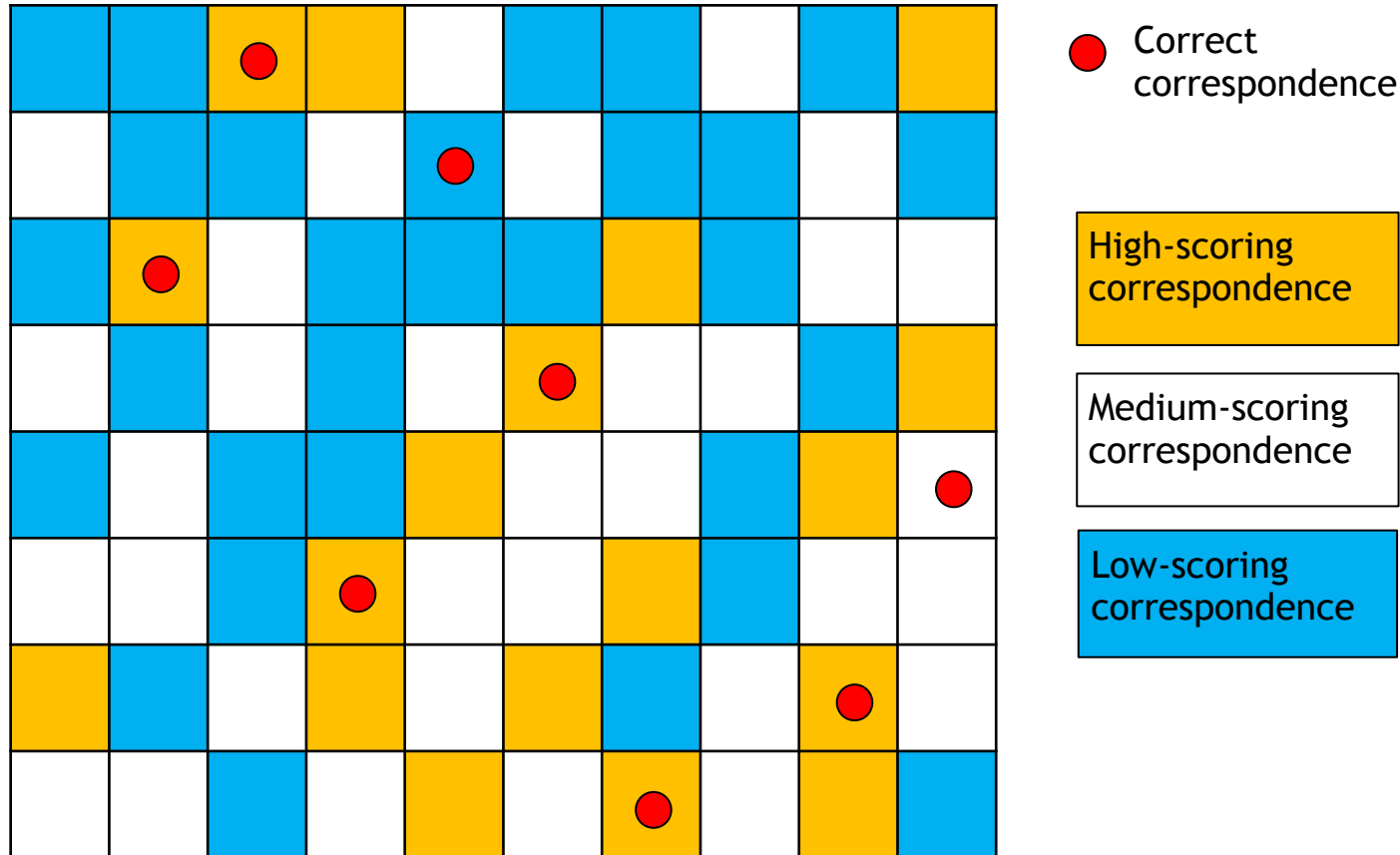
# Tentative matches

---

- The matching score is usually based on **similarity of visual appearance**, not geometric properties (**why?**)
- For example
  - SIFT descriptor, BRIEF/BRISK/ORB etc [See LE8]
  - Color descriptors
  - Tracking quality score
- For most matching matrices, brute force matching is **needlessly expensive** (and the Hungarian method is even worse!).
- Instead a **search tree** can be formed for one of feature sets
- If  $\min()$  along one dimension is used, one could also compute the **ratio score** instead [LE 8]



# Two-threshold RANSAC





# Two-threshold RANSAC

---

- Use a high threshold on correspondences to get a set  $D_0$ , which is used in the sampling stage of RANSAC  
⇒ fewer iterations are needed
- Use a low threshold (or none) to obtain a bigger set  $D$ , which is used to check for inliers in the consensus set  $C$ .  
⇒ more correspondences are found



# PROSAC

A variant of this idea is to:

- First sort the matching scores
- Remove low-probability correspondences (as before)
- Set  $D_0$  = the  $n_0$  best ranked correspondences
  - In principle,  $n_0 = n$  works here
  - $D_1$  = remaining part of the correspondences
- Do RANSAC as before (selecting from  $D_0$  and matching against  $D_1$ )
- If a good solution cannot be found with this  $S_0$ , extend it with more of the best correspondences and do RANSAC again
- Iterate this extension of  $D_0$  until a sufficiently good solution is found
- A more systematic approach along these lines is referred to as PROSAC
  - Chum & Matas: *Matching with PROSAC – Progressive Sample Consensus*, CVPR 2005





# RANSAC speedups

---

In summary:

- Using matching of visual appearance is a very effective way of pruning the set  $D$  of tentative correspondences
- This leads to
  - Increased  $w$  (= prob. of picking an inlier)
  - Reduced  $r$  (= number of RANSAC iterations for a specific  $p$ )
  - Faster RANSAC algorithm / higher  $p$  possible



# Epipolar geometry

- Given two camera matrices  $\mathbf{C}_1$  and  $\mathbf{C}_2$  we can compute  $\mathbf{F}$ :

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

$$\mathbf{F} = (\mathbf{C}_1^+)^T \mathbf{C}_2^T [\mathbf{e}_{21}]_{\times}$$

- Assuming we know the camera projection matrices, we can instead apply RQ-factorisation to them...



# Epipolar geometry

In this case we can write:

$$\mathbf{C}_1 = \mathbf{K}_1 [ \mathbf{R}_1 \ \mathbf{t}_1 ]$$

$$\mathbf{C}_2 = \mathbf{K}_2 [ \mathbf{R}_2 \ \mathbf{t}_2 ]$$

Internal  
camera  
parameters

Now, use the first camera to define the 3D coordinate system:

$$\mathbf{C}_1 = \mathbf{K}_1 [ \mathbf{I} \ \mathbf{0} ]$$

$$\mathbf{C}_2 = \mathbf{K}_2 [ \mathbf{R} \ \mathbf{t} ]$$



# Epipolar geometry

---

Finally, we assume that the two cameras are identical:  $\mathbf{K}_1 = \mathbf{K}_2 = \mathbf{K}$

$$\mathbf{C}_1 = \mathbf{K} [ \mathbf{I} \ \mathbf{0} ]$$

$$\mathbf{C}_2 = \mathbf{K} [ \mathbf{R} \ \mathbf{t} ]$$

$\mathbf{K}$  is known from the calibration, and constant  $\mathbf{R}$ ,  $\mathbf{t}$  change as the camera moves



# Relative camera transformation

---

From

$$\mathbf{C}_1 = \mathbf{K} [\mathbf{I} \ \mathbf{0}]$$

$$\mathbf{C}_2 = \mathbf{K} [\mathbf{R} \ \mathbf{t}]$$

follows that  $\mathbf{C}_2 = \mathbf{C}_1 \mathbf{T}$ , where  $\mathbf{T}$  is a  $(4 \times 4)$  rigid transformation:

$$\mathbf{T} = \begin{pmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{pmatrix}$$



# Relative camera transformation

Apply  $\mathbf{T}$  to the homogeneous coordinates of a 3D point  $\mathbf{x}$ :

$$\mathbf{T} \begin{pmatrix} \bar{\mathbf{x}} \\ 1 \end{pmatrix} = \begin{pmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{pmatrix} \begin{pmatrix} \bar{\mathbf{x}} \\ 1 \end{pmatrix} = \begin{pmatrix} \mathbf{R}\bar{\mathbf{x}} + \mathbf{t} \\ 1 \end{pmatrix}$$

The result are the homogeneous coordinates of  $\mathbf{x}$  after **first** being rotated by  $\mathbf{R}$  and **then** translated by  $\mathbf{t}$



# Relative camera transformation

- $\mathbf{T}$  transforms from the camera centred coordinate system (CCS) of camera 1 to the CCS of camera 2
- $\mathbf{T}^{-1}$  transforms in the other way:

$$\mathbf{T}\mathbf{T}^{-1} = \begin{pmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{pmatrix} \begin{pmatrix} \mathbf{R}^T & -\mathbf{R}^T\mathbf{t} \\ \mathbf{0} & 1 \end{pmatrix} = \begin{pmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & 1 \end{pmatrix}$$

- Example: the camera centre of camera 2 has 3D coordinates  $\mathbf{0}$  in CCS2
  - Its coordinates in CCS1 are given by  $-\mathbf{R}^T\mathbf{t}$



# Normalised image coordinates

Pixel coordinates are given by

$$\mathbf{y}_1 \sim \mathbf{C}_1 \mathbf{x} = \mathbf{K} [\mathbf{I} \ \mathbf{0}] \mathbf{x}$$

$$\mathbf{y}_2 \sim \mathbf{C}_2 \mathbf{x} = \mathbf{K} [\mathbf{R} \ \mathbf{t}] \mathbf{x}$$

**Normalised image coordinates** remove the influence of the internal camera parameters:

$$\mathbf{y}'_1 \sim \mathbf{K}^{-1} \mathbf{y}_1 = [\mathbf{I} \ \mathbf{0}] \mathbf{x} = \mathbf{C}'_1 \mathbf{x}$$

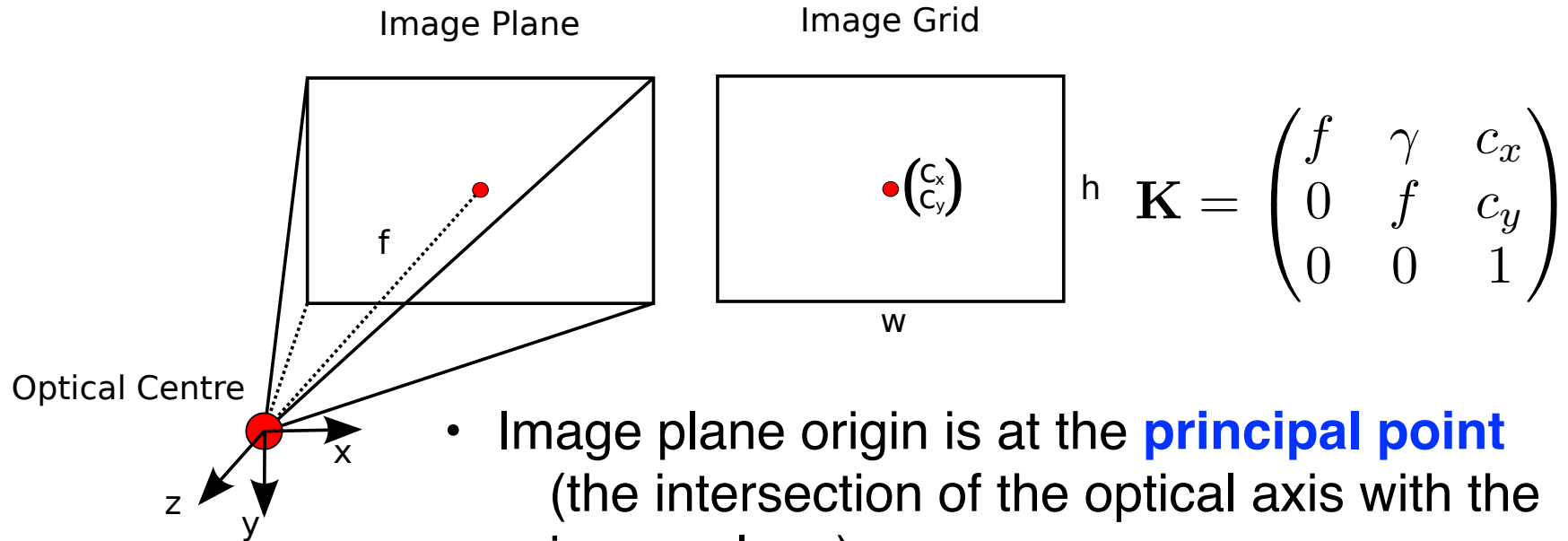
$$\mathbf{y}'_2 \sim \mathbf{K}^{-1} \mathbf{y}_2 = [\mathbf{R} \ \mathbf{t}] \mathbf{x} = \mathbf{C}'_2 \mathbf{x}$$

Normalised  
cameras





# Normalised image coordinates



- Image plane origin is at the **principal point** (the intersection of the optical axis with the image plane)
- Usually this puts the origin in the image centre, so negative normalized image coordinates are perfectly normal.



# Two types of normalised coordinates

---

Normalised image coordinates are sometimes also discussed in relation to **Hartley normalisation**

These two normalisations are unrelated!  
H-normalisation vs C-normalisation

Here: **camera normalisation** is used to refer to image coordinates that are normalised relative to the camera coordinate system



# Normalised image coordinates

---

When  $\mathbf{K}$  (and lens distortion) are known it is often more efficient to work with camera normalized image coordinates

## Calibrated epipolar geometry

The epipolar constraint becomes

$$0 = \mathbf{y}_1^T \mathbf{F} \mathbf{y}_2 = (\mathbf{K} \mathbf{y}'_1)^T \mathbf{F} \mathbf{K} \mathbf{y}'_2 = \mathbf{y}'_1{}^T \mathbf{K}^T \mathbf{F} \mathbf{K} \mathbf{y}'_2$$



# The essential matrix

We can define a matrix  $\mathbf{E} = \mathbf{K}^T \mathbf{F} \mathbf{K}$

$\mathbf{E}$  is called the **essential matrix**

Inherits the properties of  $\mathbf{F}$ , but refers specifically to C-normalised image coordinates

For example, the epipolar constraint becomes:

$$0 = \mathbf{y}'_1{}^T \mathbf{E} \mathbf{y}'_2$$

$\mathbf{F}$  and  $\mathbf{E}$  represent the same constraint but in different coordinate systems



# The essential matrix

In the same way as for  $F$ ,  $E$  is given by

$$\mathbf{E} = [\mathbf{e}_{12}]_{\times} \mathbf{C}'_1 \mathbf{C}'_2{}^+ = \mathbf{C}'_1{}^{+T} \mathbf{C}'_2{}^T [\mathbf{e}'_{21}]_{\times}$$

$\mathbf{e}'_{12}$  and  $\mathbf{e}'_{21}$  are the epipoles

$\mathbf{C}'_1$  and  $\mathbf{C}'_2$  are the camera matrices

In camera  
normalised  
image  
coordinates



# The essential matrix

In this case we get

$$\mathbf{C}'_1 = [\mathbf{I} \ \mathbf{0}] \Rightarrow \mathbf{C}'_1{}^{+T} = [\mathbf{I} \ \mathbf{0}]$$

$$\mathbf{C}'_2 = [\mathbf{R} \ \mathbf{t}] \Rightarrow \mathbf{C}'_1{}^{+T} \mathbf{C}'_2{}^T = \mathbf{R}^T, \mathbf{e}_{21} = \mathbf{t}$$

leading to

$$\mathbf{E} = \mathbf{R}^T [\mathbf{t}]_{\times}$$

E encodes the relative rotation and translation between the two cameras



# BREAK





# The essential matrix

---

- **E** is defined only from the rotation **R** and the translation **t**
- In practice **E** is a projective element (**why?**)  
⇒ **t** and  $\lambda \mathbf{t}$  refer to the same essential matrix

$$\mathbf{E}_1 = \mathbf{R}^T [ \mathbf{t} ]_x \sim \mathbf{R}^T [ \lambda \mathbf{t} ]_x = \mathbf{E}_2$$

- **E** has 5 degrees of freedom (**why?**)
  - Compare to **F** with 7 DOF





# Internal constraints on $\mathbf{E}$

---

- $\det \mathbf{E} = 0$  applies (similar to  $\mathbf{F}$ )
- Since  $\mathbf{E}$  has fewer DOF than  $\mathbf{F}$  there must be additional constraints
- They can be summarized as:
  - Singular values of  $\mathbf{E} = (\sigma, \sigma, 0)$  (**why?**)
  - Or:  $\mathbf{E} \mathbf{E}^T \mathbf{E} - \frac{1}{2} \text{tr}(\mathbf{E}^T \mathbf{E}) \mathbf{E} = \mathbf{0}$  (**why?**)



# E in the literature

---

- **E** was introduced in  
    Longuet-Higgins: *A computer algorithm for reconstructing a scene from two projections*,  
    Nature (1981)
- Remained more or less unnoticed by the  
    computer vision community until **F**  
    was introduced some 12 years later



# Estimation of $\mathbf{E}$

- $\mathbf{E}$  can be estimated from image data in the same way as  $\mathbf{F}$ , e.g. the 8-point algorithm
- The difference is that normalized image coordinates must be used
  - Hartley-normalization can still be used to increase the accuracy.
- Alternatively, estimate  $\mathbf{F}$  from pixel coordinates and transform:  $\mathbf{E} = \mathbf{K}^T \mathbf{F} \mathbf{K}$ 
  - 8/7-point algorithm for  $\mathbf{F}$
  - Gold Standard estimation of  $\mathbf{F}$
  - $\mathbf{K}$  must be known!
- Best is of course to use Gold Standard directly on normalized image coordinates, with  $\mathbf{R}, \mathbf{t}$  (in  $\mathbf{C}'_2$ ), and 3D points as unknowns.



# The 5-point algorithm

---

- The internal constraints on **E** imply that it can be estimated from only 5 corresponding points
  - Gives up to 10 solutions
- Nistér: *An efficient solution to the five-point pose problem*, CVPR 2003
  - Relatively complex algorithm, e.g., finding roots of 10th order polynomials, root polishing etc.
- RANSAC speedup
  - **E** has  $n = 5$  instead of 7 (for **F**)  $\Rightarrow r$  decreases considerably
- Reduced sensitivity to dominant planes, compared to **F**



# Relative pose

---

- Given an  $\mathbf{E}$  that satisfies the internal constraints we know:

$$\mathbf{E} = \mathbf{R}^T [ \mathbf{t} ]_x$$

- what can be said about  $\mathbf{R}$  and  $\mathbf{t}$ ?
- $(\mathbf{R}, \mathbf{t})$  is referred to as the *pose* of camera 2 relative to camera 1
  - i.e. the **relative pose** of  $C_1$  and  $C_2$



# Relative pose from $E$

---

The translation  $\mathbf{t}$ :

$\mathbf{t}$  is a right null vector of  $E$

$\mathbf{t}$  can be determined from  $\text{svd}(E)$

but only up to an undetermined scaling  
including an unknown sign of  $\mathbf{t}$

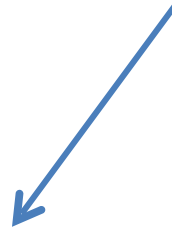


# Relative pose

- SVD of  $\mathbf{E}$  gives

$$\mathbf{E} = \mathbf{U} \mathbf{S} \mathbf{V}^T$$

Always possible!



- with  $\mathbf{U}$  and  $\mathbf{V} \in \text{SO}(3)$ , and  $\mathbf{S} = \text{diag}(\sigma, \sigma, 0)$
- $\mathbf{t} \sim \mathbf{v}_3 = 3^{\text{rd}}$  column of  $\mathbf{V}$  (why?)



# Cross product operator

---

$[\mathbf{v}_3]_{\times}$  expressed in the basis system of the columns in  $\mathbf{V}$ :

$$[\mathbf{v}_3]_{\times} = \pm \mathbf{V} \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \mathbf{V}^T = \pm \mathbf{V} [\mathbf{e}_3]_{\times} \mathbf{V}^T$$

$$\mathbf{e}_3 = (0, 0, 1)^T$$






# Relative pose

Set  $\mathbf{W} = \begin{pmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$

Try setting  $\mathbf{R} = \mathbf{V} \mathbf{W}^T \mathbf{U}^T$ , leading to

$$\begin{aligned} \mathbf{R}^T [\mathbf{t}]_x &= \mathbf{U} \mathbf{W} \mathbf{V}^T [\mathbf{v}_3]_x = \mathbf{U} \mathbf{W} \mathbf{V}^T \mathbf{V} [\mathbf{e}_3]_x \mathbf{V}^T = \\ &= \mathbf{U} \mathbf{W} [\mathbf{e}_3]_x \mathbf{V}^T \sim \mathbf{U} \mathbf{S} \mathbf{V}^T = \mathbf{E} \end{aligned}$$

 (why?)



# Ambiguity in R

However, also  $R = V W U^T$  gives

$$R^T [ \mathbf{t} ]_x = \mathbf{E} \quad (\text{check this!})$$

Consequently, there are two possible rotations that give the same  $\mathbf{E}$ :

$$R_1 = V W^T U^T$$

$$R_2 = V W U^T$$

These two rotations are always distinct!  
They form a *twisted pair*



# Relative pose: summary

---

- Given an essential matrix  $\mathbf{E} = \mathbf{R}^T [ \mathbf{t} ]_{\times}$
- There are two possible rotations  $\mathbf{R}$ :  $\mathbf{R}_1$  and  $\mathbf{R}_2$
- $\mathbf{t}$  is determined up to scale: two possible directions are opposite :  $\pm \mathbf{t}$
- In total we have 4 possible camera configurations:

$$\mathbf{C}_2 = [ \mathbf{R}_1 \quad \mathbf{t} ]$$

$$\mathbf{C}_2 = [ \mathbf{R}_1 \quad -\mathbf{t} ] \quad \text{with } \mathbf{C}_1 = [ \mathbf{I} \quad \mathbf{0} ]$$

$$\mathbf{C}_2 = [ \mathbf{R}_2 \quad \mathbf{t} ]$$

$$\mathbf{C}_2 = [ \mathbf{R}_2 \quad -\mathbf{t} ]$$



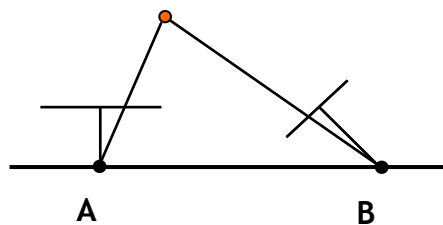
# Reality check

---

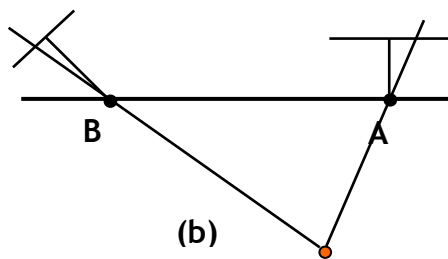
- Take a pair of corresponding points in the two images and determine the corresponding 3D point
  - Triangulation
- The 3D point **is in front of both cameras** for **only one** of the 4 configurations
  - See figure
- Thus: only **one** of the 4 possible configurations corresponds to a real stereo rig
- **t** is then determined in terms of direction, but still not in terms of scale



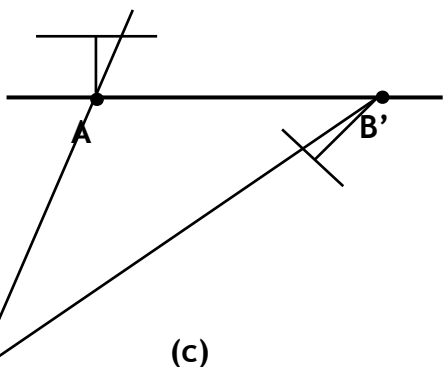
# The camera matrices $C_1$ and $C_2$



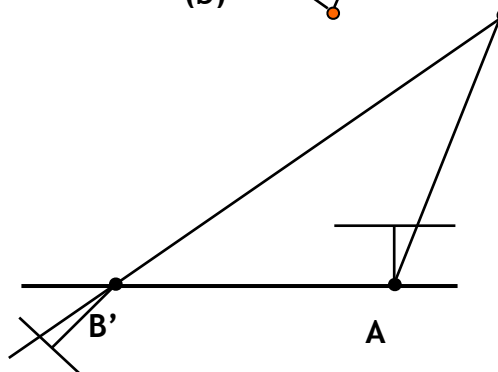
(a)



(b)



(c)



(d)

From Hartley & Zisserman,  
*Multiple View Geometry in  
Computer Vision*



# Relative pose: summary (II)

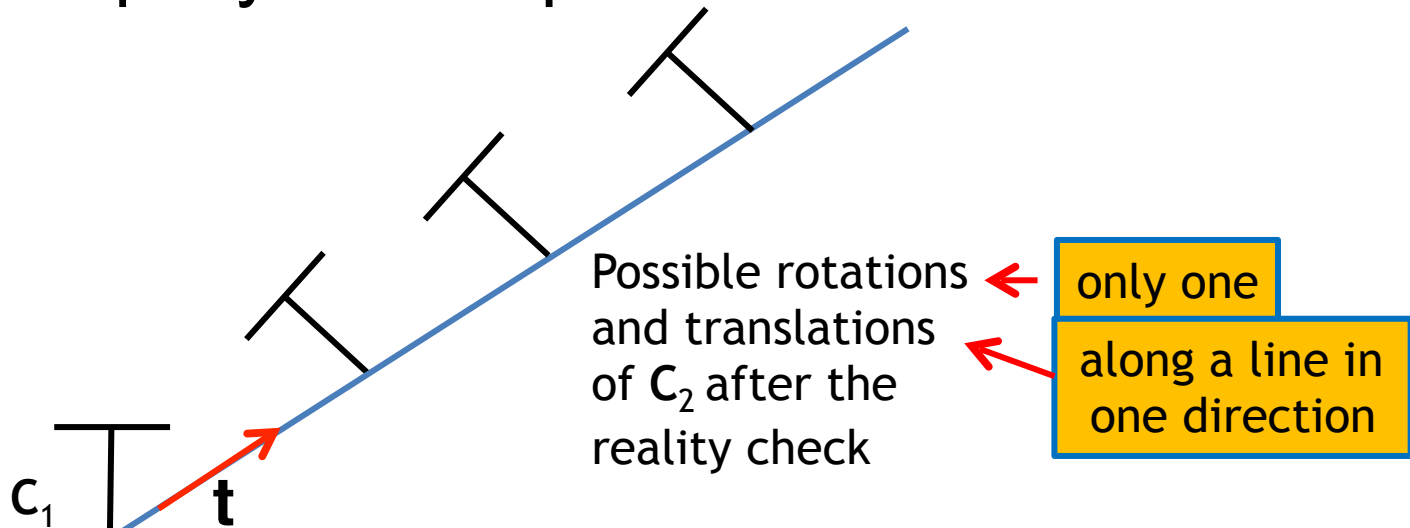
---

- Given a set of corresponding points,  $\mathbf{E}$  can be estimated
  - 8-point algorithm, 7-point alg., ..., 5-point alg.
  - Possibly multiple solutions
  - Test the epipolar constraint with more points to get only one solution for  $\mathbf{E}$
- For this  $\mathbf{E}$ , we can determine the 4 possible camera configurations
- Reality check:
  - Triangulate a 3D point to determine which one of the four configurations that has the point in front of both cameras
- This determines  $\mathbf{R}$  and sign of  $\mathbf{t}$  but not its absolute scale



# Relative pose: two cameras

- Given corresponding points in camera normalized coordinates, we can determine  $R$  uniquely and  $\mathbf{t}$  up to scale





# E or F?

If **K** is known there are several advantages of using **E** instead of **F**:

- Relative camera pose (rotation and translation) can be determined from **E**
  - Not possible from **F**
- Fewer points are needed in RANSAC to determine **E**
  - $r$  is reduced  $\Rightarrow$  faster RANSAC
- Reduced sensitivity to dominant planes
- Also lens distortion can be undone





# PnP

We can add another camera to an already estimated EG using the **perspective n-point problem** (PnP):

**given:** a set of 2D  $\leftrightarrow$  3D correspondences

$$\{\mathbf{y}_n \leftrightarrow \mathbf{x}_n\}_{n=1}^N$$

**sought:** the absolute camera pose, such that

$$\mathbf{y}_n \sim [\mathbf{R}|\mathbf{t}] \mathbf{x}_n, \text{ for } n = 1, \dots, N$$



# PnP

- Geometric loss:

$$J(\mathbf{R}, \mathbf{t}) = \sum_{n=1}^N d_{\text{PP}}^2(\mathbf{y}_n, [\mathbf{R} | \mathbf{t}] \mathbf{x}_n)$$

- Feed loss to a non-linear solver e.g.  
`scipy.optimize.least_squares`
- Use  $\text{SO}(3)$  parametrization of  $\mathbf{R}$  to restrict the problem to 6dof



# Robust PnP

- The minimal case for PnP is  $N=3$ , i.e. P3P has 1-4 real solutions. See e.g. Mikael Persson, Klas Nordberg, *Lambda Twist: An Accurate Fast Robust Perspective Three Point (P3P) Solver*, **ECCV18**
- With a minimal solver the problem can be solved robustly using RANSAC
- In general, robust PnP is better at removing outliers than a robust estimation of **E**. (**why?**)
- Many efficient solvers exist (see pointers on project page), and you are encouraged to use one in project #2.



# Summary

---

- Fast **minimal solvers** are important for RANSAC
- RANSAC for correspondences can be improved by **sampling according to similarity** (PROSAC)
- Use **calibrated epipolar geometry** when you can, it is faster and more accurate
- The **essential matrix** encodes calibrated EG, a relative rotation and a translation up to scale.
- **perspective n-point (PnP)** estimation can be used to add more views to an existing EG