Vision Algorithms for Mobile Robotics

Lecture 08
Multiple View Geometry 2

Davide Scaramuzza
http://rpg.ifi.uzh.ch
Lab Exercise 6 - This afternoon

Implement the 8-point algorithm

Estimated poses and 3D structure
2-View Geometry: recap

Depth from stereo (i.e., stereo vision):

• **Assumptions**: K, T and R are known.
• **Goal**: Recover the 3D structure from two images

2-view Structure From Motion:

• **Assumptions**: none (K, T, and R are unknown).
• **Goal**: Recover simultaneously 3D scene structure and camera poses (up to scale) from two images
Structure from Motion (SFM)

**Problem formulation:** Given a set of $n$ point *correspondences* between two images, \( \{ p_{1}^{i} = (u_{1}^{i}, v_{1}^{i}), p_{2}^{i} = (u_{2}^{i}, v_{2}^{i}) \} \), where $i = 1 \ldots n$, the goal is to simultaneously

- estimate the 3D points $P^{i}$,
- the camera relative-motion parameters $(R, T)$,
- and the camera intrinsics $K_{1}, K_{2}$ that satisfy:

\[
\begin{align*}
\lambda_{1}^{i} \begin{bmatrix} u_{1}^{i} \\ v_{1}^{i} \\ 1 \end{bmatrix} &= K_{1} I [0] \cdot \begin{bmatrix} X_{w}^{i} \\ Y_{w}^{i} \\ Z_{w}^{i} \\ 1 \end{bmatrix} \\
\lambda_{2}^{i} \begin{bmatrix} u_{2}^{i} \\ v_{2}^{i} \\ 1 \end{bmatrix} &= K_{2} [R | T] \cdot \begin{bmatrix} X_{w}^{i} \\ Y_{w}^{i} \\ Z_{w}^{i} \\ 1 \end{bmatrix}
\end{align*}
\]
Structure from Motion (SFM)

Two variants exist:

- **Calibrated** camera(s) ⇒ $K_1, K_2$ are known
- **Uncalibrated** camera(s) ⇒ $K_1, K_2$ are unknown
Structure from Motion (SFM)

- Let’s study the case in which the cameras are **calibrated**
- For convenience, let’s use *normalized image coordinates* →
- Thus, we want to find $R, T, P^i$ that satisfy:

$$
\begin{bmatrix}
\bar{u}_1 \\
\bar{v}_1 \\
1
\end{bmatrix} = \begin{bmatrix} I | 0 \end{bmatrix} \cdot \begin{bmatrix}
X^i_w \\
Y^i_w \\
Z^i_w
\end{bmatrix} $$

$$
\begin{bmatrix}
\bar{u}_2 \\
\bar{v}_2 \\
1
\end{bmatrix} = [R | T] \cdot \begin{bmatrix}
X^i_w \\
Y^i_w \\
Z^i_w
\end{bmatrix} $$

$$
\begin{bmatrix}
\bar{u} \\
\bar{v} \\
1
\end{bmatrix} = K^{-1} \begin{bmatrix}
u \\
v
1
\end{bmatrix} $$
Scale Ambiguity

If we rescale the entire scene and camera views by a constant factor (i.e., similarity transformation), the projections (in pixels) of the scene points in both images remain exactly the same:
Scale Ambiguity

• In Structure from Motion, it is therefore **not possible** to recover the absolute scale of the scene!
  • What about stereo vision? Is it possible? Why?
• Thus, only **5 degrees of freedom** are measurable:
  • 3 parameters to describe the **rotation**
  • 2 parameters for the **translation up to a scale** (we can only compute the direction of translation but not its length)
Structure From Motion (SFM)

- How many knowns and unknowns?
  - \(4n\) knowns:
    - \(n\) correspondences; each one \((u_{i1}, v_{i1})\) and \((u_{i2}, v_{i2})\), \(i = 1 \ldots n\)
  - \(5 + 3n\) unknowns
    - 5 for the motion up to a scale (3 for rotation, 2 for translation)
    - \(3n\) = number of coordinates of the \(n\) 3D points

- Does a solution exist?
  - If and only if the number of independent equations \(\geq\) number of unknowns
    \[4n \geq 5 + 3n \Rightarrow n \geq 5\]
  - First attempt to identify the solutions by Kruppa in 1913 (see historical note on slide 16).

---
Structure From Motion (SFM)

• Can we solve the estimation of relative motion \((R, T)\) independently of the estimation of the 3D points? Yes! The next couple of slides prove that this is possible.

• Once \((R, T)\) are known, the 3D points can be triangulated using the triangulation algorithm from Lecture 7 (i.e., least square approximation plus reprojection error minimization)
The Epipolar Constraint: Recap from Lecture 07

• The camera centers $C_1$, $C_2$ and one image point $p_1$ (or $p_2$) determine the so called epipolar plane.
• The intersections of the epipolar plane with the two image planes are called epipolar lines.
• Corresponding points must therefore lie along the epipolar lines: this constraint is called epipolar constraint.
• An alternative way to formulate the epipolar constraint is to notice that two corresponding image vectors plus the baseline must be coplanar.
Epipolar Geometry

\[ \bar{p}_1 = \begin{bmatrix} \bar{u}_1 \\ \bar{v}_1 \\ 1 \end{bmatrix}, \quad \bar{p}_2 = \begin{bmatrix} \bar{u}_2 \\ \bar{v}_2 \\ 1 \end{bmatrix} \]

\( \bar{p}_1, \bar{p}_2, T \) are coplanar:
\[ \bar{p}_2^T \cdot n = 0 \Rightarrow \bar{p}_2^T (T \times (R\bar{p}_1)) = 0 \Rightarrow \bar{p}_2^T [T_x]R \bar{p}_1 = 0 \Rightarrow \bar{p}_2^T E \bar{p}_1 = 0 \]

Epipolar constraint

\[ E = [T_x]R \quad \text{essential matrix} \]
Epipolar Geometry

\[
\begin{bmatrix}
\bar{u}_1 \\
\bar{v}_1 \\
1
\end{bmatrix}
\quad \bar{p}_1 =
\begin{bmatrix}
\bar{u}_2 \\
\bar{v}_2 \\
1
\end{bmatrix}
\quad \bar{p}_2 =
\]

Normalized image coordinates

\[\bar{p}_2^T E \bar{p}_1 = 0\]  
*Epipolar constraint or Longuet-Higgins equation (1981)*

\[E = [T_x]R\]  
*Essential matrix*

\[R\] and \[T\] can be computed from \[E\] recalling that:

\[E = [T_x]R\]

Example: Essential Matrix of a Camera Translating along $\mathbf{x}$

\[ E = [T_x]R \]

\[ [T_x] = \begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & b \\ 0 & -b & 0 \end{bmatrix} \]

\[ R = I_{3 \times 3} \]

\[ T = \begin{bmatrix} -b \\ 0 \\ 0 \end{bmatrix} \]

\[ \rightarrow E = [T_x]R = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & b \\ 0 & -b & 0 \end{bmatrix} \]
How to compute the Essential Matrix?

• If we don’t know \((R, T)\) can we estimate \(E\) from two images?
• Yes, given at least 5 correspondences
A Note of History

- **Kruppa showed in 1913 that 5 image correspondences is the minimal case** and that there can be at up to 11 solutions.
- However, in **1988, Demazure** showed that there are actually at most **10 distinct solutions**.
- In **1996**, Philipp proposed an **iterative algorithm to find these solutions**.
- In **2004**, Nister proposed the **first efficient and non iterative solution**. It uses Groebner basis decomposition.
- The first popular solution uses 8 points and is called the **8-point algorithm** or **Longuet-Higgins algorithm** (1981). Because of its ease of implementation, it is still used today (e.g., NASA rovers).


The 8-point algorithm

• Each pair of point correspondences \( \bar{p}_1 = (\bar{u}_1, \bar{v}_1, 1)^T, \quad \bar{p}_2 = (\bar{u}_2, \bar{v}_2, 1)^T \) provides a linear equation:

\[
\bar{p}_2^T \ E \ \bar{p}_1 = 0
\]

\[
E = \begin{bmatrix}
e_{11} & e_{12} & e_{13} \\
e_{21} & e_{22} & e_{23} \\
e_{31} & e_{32} & e_{33}
\end{bmatrix}
\]

\[
\bar{u}_2 \bar{u}_1 e_{11} + \bar{u}_2 \bar{v}_1 e_{12} + \bar{u}_2 e_{13} + \bar{v}_2 \bar{u}_1 e_{21} + \bar{v}_2 \bar{v}_1 e_{22} + \bar{v}_2 e_{23} + \bar{u}_1 e_{31} + \bar{v}_1 e_{32} + e_{33} = 0
\]

The 8-point algorithm

• For $n$ points, we can write

$$
\begin{bmatrix}
\bar{u}_2 \bar{u}_1^1 & \bar{u}_2 \bar{v}_1^1 & \bar{u}_2 \bar{v}_1^1 & \bar{v}_2 \bar{u}_1^1 & \bar{v}_2 \bar{v}_1^1 & \bar{v}_2 \bar{v}_1^1 & \bar{u}_1^1 & \bar{v}_1^1 & 1 \\
\bar{u}_2^2 \bar{u}_1 & \bar{u}_2^2 \bar{v}_1^2 & \bar{u}_2^2 \bar{v}_1^2 & \bar{v}_2 \bar{u}_1^2 & \bar{v}_2 \bar{v}_1^2 & \bar{v}_2 \bar{v}_1^2 & \bar{u}_1^2 & \bar{v}_1^2 & 1 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\bar{u}_2^n \bar{u}_1^n & \bar{u}_2^n \bar{v}_1^n & \bar{u}_2^n \bar{v}_1^n & \bar{v}_2 \bar{u}_1^n & \bar{v}_2 \bar{v}_1^n & \bar{v}_2 \bar{v}_1^n & \bar{u}_1^n & \bar{v}_1^n & 1
\end{bmatrix}
= 0
$$

Q (this matrix is known)

$\bar{E}$ (this matrix is unknown)
The 8-point algorithm

### Minimal solution
- $Q(n \times 9)$ should have rank 8 to have a unique (up to a scale) non-trivial solution $\bar{E}$
- Each point correspondence provides 1 independent equation
- Thus, 8 point correspondences are needed

### Over-determined solution
- $n > 8$ points
- A solution is to minimize $||Q \bar{E}||^2$ subject to the constraint $||\bar{E}||^2 = 1$.
  The solution is the eigenvector corresponding to the smallest eigenvalue of the matrix $Q^T Q$ (because it is the unit vector $x$ that minimizes $||Qx||^2 = x^T Q^T Qx$).
- It can be solved through Singular Value Decomposition (SVD). Matlab instructions:
  ```matlab
  [U,S,V] = svd(Q);
  Ev = V(:,9);
  E = reshape(Ev,3,3)';
  ```

### Degenerate Configurations
- The solution of the 8-point algorithm is **degenerate when the 3D points are coplanar**.
- Conversely, the 5-point algorithm works also for coplanar points
8-point algorithm: Matlab code

A few lines of code. In today’s exercise you will learn how to implement it

```matlab
function E = calibrated_eightpoint( p1, p2)

p1 = p1'; % 3xN vector; each column = [u;v;1]
p2 = p2'; % 3xN vector; each column = [u;v;1]

Q = [p1(:,1).*p2(:,1), ...
p1(:,2).*p2(:,1), ...
p1(:,3).*p2(:,1), ...
p1(:,1).*p2(:,2), ...
p1(:,2).*p2(:,2), ...
p1(:,3).*p2(:,2), ...
p1(:,1).*p2(:,3), ...
p1(:,2).*p2(:,3), ...
p1(:,3).*p2(:,3) ] ;

[U,S,V] = svd(Q);
Eh = V(:,9);
E = reshape(Eh,3,3)';
```
Extract R and T from E

- Singular Value Decomposition: $E = U \sum V^T$
- Enforcing rank-2 constraint: set smallest singular value of $\sum$ to 0:

$\sum = \begin{bmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \\ 0 & 0 & \times \end{bmatrix} = \begin{bmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \\ 0 & 0 & 0 \end{bmatrix}$

$\hat{T} = U \begin{bmatrix} 0 & \mp 1 & 0 \\ \pm 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \sum V^T$

$\hat{R} = U \begin{bmatrix} 0 & \mp 1 & 0 \\ \pm 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} V^T$

$\hat{T} = \begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & t_x \\ -t_y & t_x & 0 \end{bmatrix} \Rightarrow \hat{t} = \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}$

$T = K_2 \hat{t}$

$R = K_2 \hat{R} K_1^{-1}$
4 possible solutions of $R$ and $T$

There exists **only one solution** where points are in front of both cameras

(a) These two views are flipped by $180^\circ$ around the optical axis

(b)
Structure from Motion (SFM)

Two variants exist:

- **Calibrated** camera(s) ⇒ $K_1, K_2$ are known
  - Uses the Essential matrix

- **Uncalibrated** camera(s) ⇒ $K_1, K_2$ are unknown
  - Uses the Fundamental matrix

$P_i = ?

C_1 \quad R, T \quad C_2$
The Fundamental Matrix

So far, we have assumed to know the camera intrinsic parameters and we have used normalized image coordinates to get the epipolar constraint for calibrated cameras:

\[
\begin{bmatrix}
\overline{u}_1^i \\
\overline{v}_1^i \\
1
\end{bmatrix} = K_1^{-1}
\begin{bmatrix}
\overline{u}_1^i \\
\overline{v}_1^i \\
1
\end{bmatrix} = K_2^{-1}
\begin{bmatrix}
\overline{u}_2^i \\
\overline{v}_2^i \\
1
\end{bmatrix}
\]

\[
\overline{p}_2^T E \overline{p}_1 = 0
\]

\[
\begin{bmatrix}
\overline{u}_2^i^T \\
\overline{v}_2^i \\
1
\end{bmatrix} E
\begin{bmatrix}
\overline{u}_1^i \\
\overline{v}_1^i \\
1
\end{bmatrix} = 0
\]
The Fundamental Matrix

So far, we have assumed to know the camera intrinsic parameters and we have used normalized image coordinates to get the epipolar constraint for calibrated cameras:

\[
\begin{bmatrix}
\bar{u}_1^i \\
\bar{v}_1^i \\
1
\end{bmatrix} = K_1^{-1} \begin{bmatrix}
u_1^i \\
v_1^i \\
1
\end{bmatrix} \quad \begin{bmatrix}
\bar{u}_2^i \\
\bar{v}_2^i \\
1
\end{bmatrix} = K_2^{-1} \begin{bmatrix}
u_2^i \\
v_2^i \\
1
\end{bmatrix}
\]

\[
\bar{p}_2^T E \bar{p}_1 = 0
\]

\[
\begin{bmatrix}
u_2^i \\
v_2^i \\
1
\end{bmatrix}^T \begin{bmatrix}
K_2^{-T} E K_1^{-1}
\end{bmatrix} \begin{bmatrix}
u_1^i \\
v_1^i \\
1
\end{bmatrix} = 0
\]
The Fundamental Matrix

So far, we have assumed to know the camera intrinsic parameters and we have used normalized image coordinates to get the epipolar constraint for **calibrated cameras**:

\[
\begin{bmatrix}
\bar{u}_1^i \\
\bar{v}_1^i \\
1
\end{bmatrix} = K_1^{-1}\begin{bmatrix}
u_1^i \\
v_1^i \\
1
\end{bmatrix}
\begin{bmatrix}
\bar{u}_2^i \\
\bar{v}_2^i \\
1
\end{bmatrix} = K_2^{-1}\begin{bmatrix}
u_2^i \\
v_2^i \\
1
\end{bmatrix}
\]

\[
\bar{p}_2^T E \bar{p}_1 = 0
\]

Fundamental Matrix \( F = K_2^{-T} E K_1^{-1} \)
The 8-point Algorithm for the Fundamental Matrix

• The same 8-point algorithm to compute the essential matrix from a set of normalized image coordinates can also be used to determine the Fundamental matrix:

\[
\begin{bmatrix}
  u_2^i \\
v_2^i \\
  1
\end{bmatrix}^T \cdot
\begin{bmatrix}
  u_1^i \\
v_1^i \\
  1
\end{bmatrix} = 0
\]

• However, now the key advantage is that we work directly in pixel coordinates
Problem with 8-point algorithm

\[
\begin{bmatrix}
  u_2^1 u_1^1 & u_2^1 v_1^1 & u_2^1 & v_2^1 u_1^1 & v_2^1 v_1^1 & v_2^1 & u_1^1 & v_1^1 & 1 \\
  u_2^2 u_1^2 & u_2^2 v_1^2 & u_2^2 & v_2^2 u_1^2 & v_2^2 v_1^2 & v_2^2 & u_1^2 & v_1^2 & 1 \\
  \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
  u_2^n u_1^n & u_2^n v_1^n & u_2^n & v_2^n u_1^n & v_2^n v_1^n & v_2^n & u_1^n & v_1^n & 1
\end{bmatrix}
\begin{bmatrix}
f_{11} \\
f_{12} \\
f_{13} \\
f_{21} \\
f_{22} \\
f_{23} \\
f_{31} \\
f_{32} \\
f_{33}
\end{bmatrix} = 0
\]
Problem with 8-point algorithm

- Poor numerical conditioning, which makes results very sensitive to noise
- Can be fixed by rescaling the data: **Normalized 8-point algorithm**

![Normalized 8-point algorithm](image)

Orders of magnitude difference between column of data matrix → least-squares yields poor results
Normalized 8-point algorithm (1/3)

• This can be fixed using a normalized 8-point algorithm [Hartley, 1997], which estimates the Fundamental matrix on a set of *Normalized correspondences* (with better numerical properties) and then unnormalizes the result to obtain the fundamental matrix for the *given (unnormalized) correspondences*.

• **Idea:** Transform image coordinates so that they are in the range $\sim [-1,1] \times [-1,1]$

• One way is to apply the following rescaling and shift:

\[
\begin{bmatrix}
\frac{2}{700} & 0 & -1 \\
0 & \frac{2}{500} & -1 \\
0 & 0 & 1
\end{bmatrix}
\]

Hartley, In defense of the eight-point algorithm, IEEE Transactions of Pattern Analysis and Machine Intelligence, *PDF*
The Normalized 8-point algorithm can be summarized in three steps:

1. **Normalize** the point correspondences: \( \tilde{p}_1 = B_1 p_1 \), \( \tilde{p}_2 = B_2 p_2 \)

2. Estimate **normalized** \( \hat{F} \) with 8-point algorithm using normalized coordinates \( \tilde{p}_1, \tilde{p}_2 \)

3. Compute **unnormalized** \( F \) from \( \hat{F} \):

\[
\tilde{p}_2^T \hat{F} \tilde{p}_1 = 0
\]

\[
F = B_2^T \hat{F} B_1
\]
Normalized 8-point algorithm (2/3)

• In the original 1997 paper, Hartley proposed to rescale the two point sets such that the centroid of each set is 0 and the mean standard deviation $\sqrt{2}$ (equivalent to having the points distributed around a circled passing through the four corners of the $[-1,1] \times [-1,1]$ square).

• This can be done for every point as follows: 
\[
\hat{p}^i = \frac{\sqrt{2}}{\sigma}(p^i - \mu)
\]
where $\mu = (\mu_x, \mu_y) = \frac{1}{N}\sum_{i=1}^{n}p^i$ is the centroid and $\sigma = \frac{1}{N}\sum_{i=1}^{n}\|p^i - \mu\|^2$ is the mean standard deviation of the point set.

• This transformation can be expressed in matrix form using homogeneous coordinates:
\[
\hat{p}^i = \begin{bmatrix}
\frac{\sqrt{2}}{\sigma} & 0 & -\frac{\sqrt{2}}{\sigma}\mu_x \\
0 & \frac{\sqrt{2}}{\sigma} & -\frac{\sqrt{2}}{\sigma}\mu_y \\
0 & 0 & 1
\end{bmatrix}
\]

Hartley, In defense of the eight-point algorithm, IEEE Transactions of Pattern Analysis and Machine Intelligence, PDF
Can $R, T, K_1, K_2$ be extracted from $F$?

• In general no: infinite solutions exist

• However, if the coordinates of the principal points of each camera are known and the two cameras have the same focal length $f$ in pixels, then $R, T, f$ can determined uniquely
Comparison between Normalized and non-normalized algorithm

<table>
<thead>
<tr>
<th></th>
<th>8-point</th>
<th>Normalized 8-point</th>
<th>Nonlinear refinement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avg. Ep. Line Distance</td>
<td>2.33 pixels</td>
<td>0.92 pixel</td>
<td>0.86 pixel</td>
</tr>
</tbody>
</table>
Error Measures

• The **quality of the estimated Essential or Fundamental matrix** can be measured using different error metrics:
  • Algebraic error
  • Directional Error
  • Epipolar Line Distance
  • Reprojection Error

• When is the error 0?

• **These errors will be exactly 0 only if** \( E \) (or \( F \)) **is computed from just 8 points** (because in this case a **non-overdetermined solution** exists).

• **For more than 8 points, it will only be 0 if there is no noise or outliers in the data** (if there is image noise or outliers then it the system becomes overdetermined)
Algebraic Error

- It follows directly from the 8-point algorithm, which seeks to minimize the algebraic error:
  \[ \text{err} = \|QE\|^2 = \sum_{i=1}^{N} (\overline{p}_2^T E \overline{p}_1)^2 \]

- From the proof of the epipolar constraint and using the definition of dot product, it can be observed that:
  \[ \|\overline{p}_2^T E \overline{p}_1\| = \|\overline{p}_2^T \cdot (E \overline{p}_1)\| = \|\overline{p}_2\| \|E \overline{p}_1\| \cos(\theta) \]
  \[ = \|\overline{p}_2\| \|[T_x] R \overline{p}_1\| \cos(\theta) \]

- We can see that this product depends on the angle \(\theta\) between \(\overline{p}_2\) and the normal \(n = E \overline{p}_1\) to the epipolar plane. It is non zero when \(\overline{p}_1, \overline{p}_2,\) and \(T\) are not coplanar.

- What is the drawback of this error measure?
Directional Error

- Sum of squared cosines of the angle from the epipolar plane: $\text{err} = \sum_{i=1}^{N} (\cos(\theta_i))^2$

- It is obtained by normalizing the algebraic error:

$$\cos(\theta) = \frac{\overline{p_2^T E p_1}}{\|p_2\| \|Ep_1\|}$$
Epipolar Line Distance

- Sum of Squared Epipolar-Line-to-point Distances: \( \text{err} = \sum_{i=1}^{N} \left( d(p_{1i}^i, l_{1i}^i) \right)^2 + \left( d(p_{2i}^i, l_{2i}^i) \right)^2 \)

- Cheaper than reprojection error because does not require point triangulation
Reprojection Error

- Sum of the **Squared Reprojection Errors**: $\text{err} = \sum_{i=1}^{N} \| p_i^1 - \pi(P_i^1, K_1, I, 0) \|^2 + \| p_i^2 - \pi(P_i^1, K_2, R, T) \|^2$

- More **expensive** than the previous three errors because it requires to first triangulate the 3D points!

- **However it is the most popular because more accurate**. The reason is that the error is computed directly with the respect the raw input data, which are the image points.
Things to remember

• SFM from 2 view
  • Calibrated and uncalibrated case
  • Proof of Epipolar Constraint
  • 8-point algorithm and algebraic error
  • Normalized 8-point algorithm
  • Algebraic, directional, Epipolar line distance, Reprojection error
Readings

- CH. 7.2 of Szeliski book, 1st edition
- Ch. 14.2 of Corke book
Understanding Check

Are you able to answer the following questions?
• What's the minimum number of correspondences required for calibrated SFM and why?
• Are you able to derive the epipolar constraint?
• Are you able to define the essential matrix?
• Are you able to derive the 8-point algorithm?
• How many rotation-translation combinations can the essential matrix be decomposed into?
• Are you able to provide a geometrical interpretation of the epipolar constraint?
• Are you able to describe the relation between the essential and the fundamental matrix?
• Why is it important to normalize the point coordinates in the 8-point algorithm?
• Describe one or more possible ways to achieve this normalization.
• Are you able to describe the normalized 8-point algorithm?
• Are you able to provide quality metrics and their interpretation for the essential and fundamental matrix estimation?