Lecture 08
Multiple View Geometry 3

Prof. Dr. Davide Scaramuzza
sdavide@ifi.uzh.ch
Today’s outline

• Review of last lecture
• RANSAC for robust Structure from Motion
• Visual Odometry
• Use case: SVO: Semi-direct Visual Odometry (developed by us)
Review of last lecture
Problem formulation: Given many points in correspondence across two images, \{\( (u^i_1, v^i_1), (u^i_2, v^i_2) \)\}, simultaneously compute the 3D location \( P_i \), the camera relative-motion parameters \( (R, t) \), and camera intrinsic \( K_{1,2} \) that satisfy

\[
\begin{align*}
\lambda_1 \begin{bmatrix} u^i_1 \\ v^i_1 \\ 1 \end{bmatrix} &= K_1[I|0] \cdot \begin{bmatrix} X^i_w \\ Y^i_w \\ Z^i_w \\ 1 \end{bmatrix} \\
\lambda_2 \begin{bmatrix} u^i_2 \\ v^i_2 \\ 1 \end{bmatrix} &= K_2[R|T] \cdot \begin{bmatrix} X^i_w \\ Y^i_w \\ Z^i_w \\ 1 \end{bmatrix}
\end{align*}
\]
Structure from Motion (SFM)

- Two variants exist:
  - **Uncalibrated** camera(s) -> $K$ is unknown
    - Uses the Fundamental Matrix
  - **Calibrated** camera(s) -> $K$ is known
    - Uses the Essential Matrix

\[
\begin{align*}
R, T &= ? \\
P_i &= ?
\end{align*}
\]
Structure from Motion (SFM)

- Let’s study the case in which the camera(s) is «calibrated»
- For convenience, let’s use normalized image coordinates
- Thus, we want to find $R, T, P_i$ that satisfy

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

$$\begin{cases} 
\lambda_1 \begin{bmatrix} \bar{u}^i_1 \\ \bar{v}^i_1 \\ 1 \end{bmatrix} = [I | 0] \cdot \begin{bmatrix} X^i_w \\ Y^i_w \\ Z^i_w \\ 1 \end{bmatrix} \\
\lambda_2 \begin{bmatrix} \bar{u}^i_2 \\ \bar{v}^i_2 \\ 1 \end{bmatrix} = [R | T] \cdot \begin{bmatrix} X^i_w \\ Y^i_w \\ Z^i_w \\ 1 \end{bmatrix}
\end{cases}$$

$$P_i = ?$$

$$R, T = ?$$
Structure from Motion (SfM)

• How many knowns and unknowns?
  – \(4n\) knowns:
    • \(n\) correspondences; each one \((u^i_1, v^i_1)\) and \((u^i_2, v^i_2)\), \(i = 1 \ldots n\)
  – \(5 + 3n\) unknowns
    • 5 for the motion up to a scale (rotation-> 3, translation->2)
    • \(3n\) = number of coordinates of the \(n\) 3D points

• Does a solution exist?
  – If and only if
    \[\text{number of independent equations} \geq \text{number of unknowns}\]
    \[\Rightarrow 4n \geq 5 + 3n \Rightarrow n \geq 5\]
Epipolar Geometry

\[ p_1 = \begin{bmatrix} \bar{u}_1 \\ \bar{v}_1 \\ 1 \end{bmatrix} \quad p_2 = \begin{bmatrix} \bar{u}_2 \\ \bar{v}_2 \\ 1 \end{bmatrix} \]

\[ p_1, p_2, T \text{ are coplanar:} \]

\[ p_2^T \cdot n = 0 \Rightarrow p_2^T \cdot (T \times p_1') = 0 \Rightarrow p_2^T \cdot (T \times (R p_1)) = 0 \]

\[ \Rightarrow p_2^T [T]_\times R p_1 = 0 \Rightarrow p_2^T E p_1 = 0 \quad \text{epipolar constraint} \]

\[ E = [T]_\times R \quad \text{essential matrix} \]
Epipolar Geometry

\[
p_1 = \begin{bmatrix}
\bar{u}_1 \\
\bar{v}_1 \\
1
\end{bmatrix}
\quad p_2 = \begin{bmatrix}
\bar{u}_2 \\
\bar{v}_2 \\
1
\end{bmatrix}
\text{Normalized image coordinates}
\]

\[
p_2^T E p_1 = 0 \quad \text{Epipolar constraint}
\]

\[
E = [T]_x R \quad \text{Essential matrix}
\]

- The Essential Matrix can be computed from 5 image correspondences [Kruppa, 1913]. The more points, the higher accuracy.

- The Essential Matrix can be decomposed into \( R \) and \( T \) by recalling that \( E = [T]_x R \). Two distinct solutions for \( R \) and \( T \) are possible (i.e., 4-fold ambiguity).
The eight-point algorithm

\[ p_1 = (\bar{u}_1, \bar{v}_1, 1)^T, \quad p_2 = (\bar{u}_2, \bar{v}_2, 1) \quad p_2^T E p_1 = 0 \]

\[
\begin{bmatrix}
\bar{u}_2 & \bar{v}_2 & 1
\end{bmatrix}
\begin{bmatrix}
e_{11} & e_{12} & e_{13} \\
e_{21} & e_{22} & e_{23} \\
e_{31} & e_{32} & e_{33}
\end{bmatrix}
\begin{bmatrix}
\bar{u}_1 \\
\bar{v}_1 \\
1
\end{bmatrix} = 0
\]

\[
\begin{bmatrix}
u_2 u_1 & u_2 v_1 & u_2 & v_2 u_1 & v_2 v_1 & v_2 & u_1 & v_1 & 1
\end{bmatrix}
\begin{bmatrix}
e_{11} \\
e_{12} \\
e_{13} \\
e_{21} \\
e_{22} \\
e_{23} \\
e_{31} \\
e_{32} \\
e_{33}
\end{bmatrix} = 0
\]

Minimize:

\[
\sum_{i=1}^{N} (p_i^T E p_i)^2
\]

under the constraint \( \|E\|^2 = 1 \)

A linear least-square solution is given by the eigenvector of \( Q \) corresponding to its smallest eigenvalue (which is the unit vector that minimizes \( |Q \cdot E|^2 \)). We use Singular Value Decomposition.
8-point algorithm: Matlab code

- function F = 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);
- F = V(:,9);

- F = reshape(V(:,9),3,3)';
The eight-point algorithm

Meaning of the linear least-square error \( \sum_{i=1}^{N} (p_{2}^{iT} E p_{1}^{i})^2 \):

sum of squared *algebraic* distances between points \( p_2 \) and epipolar lines \( E p_1 \) (or points \( p_1 \) and epipolar lines \( E^T p_2 \))
Extract \( R \) and \( T \) from \( E \)
(this slide will not be asked at the exam)

- **Singular Value Decomposition**

\[
E = U \Sigma V^T
\]

Enforcing rank-2 constraint

\[
\hat{T} = U \begin{bmatrix}
0 & \mp 1 & 0 \\
\pm 1 & 0 & 0 \\
0 & 0 & 0
\end{bmatrix} \Sigma V^T
\]

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

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

\[
t = K_2 \hat{t}
\]

\[
R = K_2 \hat{R} K_1^{-1}
\]
Structure from Motion (SFM)

- Two variants exist:
  - **Calibrated** camera(s) -> $K$ is known
    - Uses the Essential Matrix
  - **Uncalibrated** camera(s) -> $K$ is unknown
    - Uses the Fundamental Matrix
The Fundamental Matrix

- So far, we had the camera intrinsic parameters and we used normalized image coordinates

\[
\begin{bmatrix}
    p_2^T E & p_1 = 0 \\
    u_2^i & v_2^i & E & u_1^i & v_1^i = 0
\end{bmatrix}
\]

\[
\begin{bmatrix}
    u_1^i \\
    v_1^i \\
    1
\end{bmatrix} = K^{-1}
\begin{bmatrix}
    u_1^i \\
    v_1^i \\
    1
\end{bmatrix}
\]

\[
\begin{bmatrix}
    u_2^i \\
    v_2^i \\
    1
\end{bmatrix} = K^{-1}
\begin{bmatrix}
    u_2^i \\
    v_2^i \\
    1
\end{bmatrix}
\]

\[
F = K_2^{-T} E K_1^{-1}
\]

\[
E = [T]_x R
\]

\[
\Rightarrow F = K_2^{-T} [T]_x R K_1^{-1}
\]
Today’s outline

• Review of last lecture
• **RANSAC for robust Structure from Motion**
• Visual Odometry
• Use case: SVO: Semi-direct Visual Odometry (developed by us)
“Robust” Structure from Motion

- All Structure-from-Motion algorithms (including the 8-point algorithms) assume that **image correspondences are correct**
- However, finding the correct correspondences is not always successful
  - We call false image correspondences **outliers**
  - We call correct image correspondences **inliers**
“Robust” Structure from Motion

- All Structure-from-Motion algorithms (including the 8-point algorithms) assume that image correspondences are correct.
- However, finding the correct correspondences is not always successful.
  - We call false image correspondences outliers.
  - We call correct image correspondences inliers.
RANSAC (RAndon SAmple Consensus)

- Let’s review what we have said about it in the previous lectures...
RANSAC (RAndon SAmple Consensus)

- RANSAC has become the **standard method for model fitting in the presence of outliers** (very noisy points or wrong data)
- It can be applied to line fitting but also to thousands of different problems where the goal is to **estimate the parameters of a model from the data** (e.g., camera calibration, Structure from Motion, DLT, homography, etc.)
- Let’s review RANSAC for line fitting and see how we can adapt it to SfM

RANSAC
RANSAC

- Select sample of 2 points at random
- Select sample of 2 points at random
- Calculate model parameters that fit the data in the sample
RANSAC

- Select sample of 2 points at random
- Calculate model parameters that fit the data in the sample
- Calculate error function for each data point
RANSAC

- Select sample of 2 points at random
- Calculate model parameters that fit the data in the sample
- Calculate error function for each data point
- Select data that supports current hypothesis
RANSAC

- Select sample of 2 points at random
- Calculate model parameters that fit the data in the sample
- Calculate error function for each data point
- Select data that supports current hypothesis
- Repeat sampling
RANSAC

- Select sample of 2 points at random
- Calculate model parameters that fit the data in the sample
- Calculate error function for each data point
- Select data that supports current hypothesis
- Repeat sampling
RANSAC

Set with the maximum number of inliers obtained within $k$ iterations

$$k = \frac{\log(1 - p)}{\log(1 - (1 - \varepsilon)^s)}$$

- $\varepsilon$ is the fraction of outliers in the data (e.g., 50%)
- $p$ is the requested probability of success (e.g., 99%)
- $s$ is the minimum number of points required to estimate the model (e.g., for a line $s = 2$)
RANSAC applied to general model fitting

1. Initial: let $A$ be a set of $N$ points

2. repeat

3. Randomly select a sample of $s$ points from $A$

4. Fit a model from the $s$ points

5. Compute the distances of all other points from this model

6. Construct the inlier set (i.e. count the number of points whose distance $< d$)

7. Store these inliers

8. until maximum number of iterations $k$ reached

9. The set with the maximum number of inliers is chosen as a solution to the problem

In order to implement RANSAC in SfM, we need three ingredients:

1. What’s the model in SfM?
2. What’s the minimum number of points to estimate the model?
3. How do we compute the distance of a point from the model? In other words, can we define a distance that measures how well a point fits the model?
1. **What’s the model** in SfM?
   1. Possible models are:
      1. \( R, T \)
      2. \( E \) (i.e., essential Matrix, for calibrated cameras) or \( F \) (Fundamental matrix, for uncalibrated cameras)

2. **What’s the minimum number of points** to estimate the model?
   1. We know that 5 points is the theoretical minimum number of points
   2. However, if we use the 8-point algorithm, then, 8 is the minimum

3. **How do we compute the error**, i.e., the **distance** of a point from the model?
   1. If we use \( E \) for the model, then we can use the epipolar constraint \( p_2^T Ep_1 = 0 \) to measure how well a correspondence pair \((p_1, p_2)\) verifies the model \( E \). For instance, we can use a threshold \( th \) and count as inliers all correspondence pairs that satisfy \( |p_2^T Ep_1| < th \)
   2. However, in which units is \( th \) defined? Therefore, a better distance is the **reprojection error** \( err \) (measured in pixels). We can then count as inliers all correspondence pairs that satisfy \( err < th \), where \( th \) is now defined in pixels.

3. **In the next three slides, we give an overview of four different popular error measures:**
   1. Algebraic error
   2. Directional error
   3. Epipolar-Line distance
   4. Reprojection error
Algebraic Error and Directional Error

Least-square error

It minimizes \[ \sum_{i=1}^{N} (p_{i2}^T Ep_{i1})^2 \]

However, it does not have any geometric meaning per se. However, using the definition of dot product, it can be observed that

\[ p_{1}^T \cdot Ep_{2} = \|p_{1}^T\|\|Ep_{2}\|\cos(\theta) \]

It can be observed that this product is non zero when, \( p_{1}^T, p_{2}, \) and \( T \) are not coplanar.

Directional Error (angular distance to the Epipolar plane)

A better error measure is to maximize \( \cos(\theta) \). From above, we obtain:

\[ \cos(\theta) = \frac{p_{1}^T \cdot Ep_{2}}{\|p_{1}^T\|\|Ep_{2}\|} \]
Epipolar-Line Distance

Minimize sum of squared epipolar distances

\[ \sum_{i=1}^{N} \left[ d^2(p_2, l_2) + d^2(p_1, l_1) \right] \]
Reprojection Error

- Definition: is the sum of the squared distances between the observed image points and the reprojection of the triangulated 3D point

\[ SSRE = d^2(p_1, M_1P) + d^2(p_2, M_2P) \]

\[ d(p_1, M_1P) = \| p_1 - M_1P \| \]
Example: 8-point RANSAC applied to SfM

- Let’s consider the following image pair and its image correspondences (e.g., obtained by Harris, SIFT, BRISK, etc.)
Example: 8-point RANSAC applied to SfM

- Let’s consider the following image pair and its image correspondences (e.g., obtained by Harris, SIFT, BRISK, etc.)
- For convenience, we overlay the features of the second image in the first image and use arrows to denote the motion vectors of the features
Example: 8-point RANSAC applied to SfM

Let’s consider the following image pair and its image correspondences (e.g., obtained by Harris, SIFT, BRISK, etc.)

For convenience, we overlay the features of the second image in the first image and use arrows to denote the motion vectors of the features.

1. Randomly select 8 point correspondences
Example: 8-point RANSAC applied to SfM

• Let’s consider the following image pair and its image correspondences (e.g., obtained by Harris, SIFT, BRISK, etc.)
• For convenience, we overlay the features of the second image in the first image and use arrows to denote the motion vectors of the features

1. Randomly select 8 point correspondences
2. Fit the model to all other points and count the inliers

Image 1
Example: 8-point RANSAC applied to SfM

• Let’s consider the following image pair and its image correspondences (e.g., obtained by Harris, SIFT, BRISK, etc.)
• For convenience, we overlay the features of the second image in the first image and use arrows to denote the motion vectors of the features

1. Randomly select 8 point correspondences
2. Fit the model to all other points and count the inliers
3. Repeat from 1

Image 1
Example: 8-point RANSAC applied to SfM

- Let’s consider the following image pair and its image correspondences (e.g., obtained by Harris, SIFT, BRISK, etc.)
- For convenience, we overlay the features of the second image in the first image and use arrows to denote the motion vectors of the features
Example: 8-point RANSAC applied to SfM

• Let’s consider the following image pair and its image correspondences (e.g., obtained by Harris, SIFT, BRISK, etc.)
• For convenience, we overlay the features of the second image in the first image and use arrows to denote the motion vectors of the features

1. Randomly select 8 point correspondences
Example: 8-point RANSAC applied to SfM

- Let’s consider the following image pair and its image correspondences (e.g., obtained by Harris, SIFT, BRISK, etc.)
- For convenience, we overlay the features of the second image in the first image and use arrows to denote the motion vectors of the features

1. Randomly select 8 point correspondences
2. Fit the model to all other points and count the inliers
Example: 8-point RANSAC applied to SfM

• Let’s consider the following image pair and its image correspondences (e.g., obtained by Harris, SIFT, BRISK, etc.)
• For convenience, we overlay the features of the second image in the first image and use arrows to denote the motion vectors of the features

1. Randomly select 8 point correspondences
2. Fit the model to all other points and count the inliers
3. Repeat from 1 for \( k \) times

\[ k = \frac{\log(1 - p)}{\log(1 - (1 - \varepsilon)^8)} \]
RANSAC iterations $k$ vs. $s$

$k$ is exponential in the number of points $s$ necessary to estimate the model:

- **8-point RANSAC**
  - Assuming
    - $p = 99\%$,
    - $\varepsilon = 50\%$ (fraction of outliers)
    - $s = 8$ points (8-point algorithm)
  
  \[ k = \frac{\log(1 - p)}{\log(1 - (1 - \varepsilon)^s)} = 1177 \text{ iterations} \]

- **5-point RANSAC**
  - Assuming
    - $p = 99\%$,
    - $\varepsilon = 50\%$ (fraction of outliers)
    - $s = 5$ points (5-point algorithm of David Nister (2004))
  
  \[ k = \frac{\log(1 - p)}{\log(1 - (1 - \varepsilon)^s)} = 145 \text{ iterations} \]

- **2-point RANSAC (e.g., line fitting)**
  - Assuming
    - $p = 99\%$,
    - $\varepsilon = 50\%$ (fraction of outliers)
    - $s = 2$ points
  
  \[ k = \frac{\log(1 - p)}{\log(1 - (1 - \varepsilon)^s)} = 16 \text{ iterations} \]

<table>
<thead>
<tr>
<th>Number of points ($s$):</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>2</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of iterations ($N$):</td>
<td>1177</td>
<td>587</td>
<td>292</td>
<td>145</td>
<td>71</td>
<td>16</td>
<td>7</td>
</tr>
</tbody>
</table>
RANSAC iterations $k$ vs. $\varepsilon$

- $k$ is increases exponentially with the fraction of outliers $\varepsilon$
RANSAC iterations

- As observed, $k$ is exponential in the number of points $s$ necessary to estimate the model.
- The 8-point algorithm is extremely simple and was very successful; however, it requires more than 1177 iterations.
- Because of this, there has been a large interest by the research communities in using smaller motion parameterizations.
- The first efficient solution to the minimal-case solution (5-point algorithm) took more than 100 years.
- The 5-point RANSAC only requires 145 iterations; however:
  - The 5-point algorithm can return up to 10 solutions of $E$.
  - The 8-point algorithm only returns a unique solution of $E$.

Can we use less than 5 points?
Yes, if you use motion constraints!
Planar Motion

Planar motion is described by three parameters: θ, φ, ρ

\[
R = \begin{bmatrix}
\cos \theta & -\sin \theta & 0 \\
\sin \theta & \cos \theta & 0 \\
0 & 0 & 1 \\
\end{bmatrix}
\]

\[
T = \begin{bmatrix}
\rho \cos \phi \\
\rho \sin \phi \\
0 \\
\end{bmatrix}
\]

Let's compute the Epipolar Geometry

\[
E = [T]_x R \quad \text{Essential matrix}
\]

\[
p_2^T E \, p_1 = 0 \quad \text{Epipolar constraint}
\]
Planar Motion

Planar motion is described by three parameters: $\theta$, $\phi$, $\rho$

\[
R = \begin{bmatrix}
\cos \theta & -\sin \theta & 0 \\
\sin \theta & \cos \theta & 0 \\
0 & 0 & 1
\end{bmatrix}
\]

\[
T = \begin{bmatrix}
\rho \cos \phi \\
\rho \sin \phi \\
0
\end{bmatrix}
\]

Let's compute the Epipolar Geometry

Recall that: $[a_x] =
\begin{bmatrix}
0 & -a_z & a_y \\
a_z & 0 & -a_x \\
-a_y & a_x & 0
\end{bmatrix}$

$E = [T] \times R =
\begin{bmatrix}
0 & 0 & \rho \sin \phi \\
0 & 0 & -\rho \cos \phi \\
-\rho \sin \phi & \rho \cos \phi & 0
\end{bmatrix}
\begin{bmatrix}
\cos \theta & -\sin \theta & 0 \\
\sin \theta & \cos \theta & 0 \\
0 & 0 & 1
\end{bmatrix}$
Planar Motion

Planar motion is described by three parameters: $\theta$, $\phi$, $\rho$

$$R = \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad T = \begin{bmatrix} \rho \cos \phi \\ \rho \sin \phi \\ 0 \end{bmatrix}$$

Let's compute the Epipolar Geometry

Recall that: $[a_x] = \begin{bmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{bmatrix}$

$$E = [T] \times R = \begin{bmatrix} 0 & 0 & \rho \sin \phi \\ 0 & 0 & -\rho \cos \phi \\ -\rho \sin (\phi - \theta) & \rho \cos (\phi - \theta) & 0 \end{bmatrix}$$
Planar Motion

Planar motion is described by three parameters: $\theta$, $\varphi$, $\rho$

\[
R = \begin{bmatrix}
\cos \theta & -\sin \theta & 0 \\
\sin \theta & \cos \theta & 0 \\
0 & 0 & 1
\end{bmatrix}
\]

\[
T = \begin{bmatrix}
\rho \cos \varphi \\
\rho \sin \varphi \\
0
\end{bmatrix}
\]

By applying the epipolar constraint, every image correspondence provides one equation.

Thus, 2 correspondences are sufficient to estimate $\theta$ and $\varphi$

[“2-Point RANSAC”, Ortin, 2001]
Can we use an even smaller (<2) number of points?

Yes, if we exploit ground, wheeled vehicles with non-holonomic constraints
Planar & Circular Motion (e.g., cars)

Wheeled vehicles, like cars, follow locally-planar circular motion about the Instantaneous Center of Rotation (ICR).

Example of Ackerman steering principle

Locally-planar circular motion
Planar & Circular Motion (e.g., cars)

Wheeled vehicles, like cars, follow locally-planar circular motion about the Instantaneous Center of Rotation (ICR).

\[ \varphi = \theta/2 \Rightarrow \text{only 2 parameters } (\theta, \rho) \text{ need to be estimated} \]

and therefore only 1 point is needed

This is the smallest parameterization possible and results in the most efficient algorithm for removing outliers

Planar & Circular Motion (e.g., cars)

\[
R = \begin{bmatrix}
\cos \theta & -\sin \theta & 0 \\
\sin \theta & \cos \theta & 0 \\
0 & 0 & 1 \\
\end{bmatrix}
\]

\[
T = \begin{bmatrix}
\rho \cos \frac{\theta}{2} \\
\rho \sin \frac{\theta}{2} \\
0 \\
\end{bmatrix}
\]

Let’s compute the Epipolar Geometry

\[
E = [T]_x R \quad \text{Essential matrix}
\]

\[
p_2^T E p_1 = 0 \quad \text{Epipolar constraint}
\]
Planar & Circular Motion (e.g., cars)

\[ R = \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad T = \begin{bmatrix} \rho \cos \frac{\theta}{2} \\ \rho \sin \frac{\theta}{2} \\ 0 \end{bmatrix} \]

Let’s compute the Epipolar Geometry

Recall that: \([a_x] = \begin{bmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{bmatrix}\)

\[ E = [T] \times R = \begin{bmatrix} 0 & 0 & \rho \sin \frac{\theta}{2} \\ 0 & 0 & -\rho \cos \frac{\theta}{2} \\ -\rho \sin \frac{\theta}{2} & \rho \cos \frac{\theta}{2} & 0 \end{bmatrix} \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \]
Let’s compute the Epipolar Geometry

Recall that: $[\mathbf{a}_x] = \begin{bmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{bmatrix}$

$E = [T] \times R = \begin{bmatrix} 0 & 0 & \rho \sin \frac{\theta}{2} \\ 0 & 0 & \rho \cos \frac{\theta}{2} \\ \rho \sin \frac{\theta}{2} & -\rho \cos \frac{\theta}{2} & 0 \end{bmatrix}$
Planar & Circular Motion (e.g., cars)

Let’s compute the Epipolar Geometry

\[ p_2^T E p_1 = 0 \quad \Rightarrow \quad \sin\left(\frac{\theta}{2}\right) \cdot (u_2 + u_1) + \cos\left(\frac{\theta}{2}\right) \cdot (v_2 - v_1) = 0 \]

\[ \theta = -2 \tan^{-1}\left(\frac{v_2 - v_1}{u_2 + u_1}\right) \]
1-Point RANSAC algorithm

- Point RANSAC is ONLY used to find the inliers.
- Motion is then estimated from them in 6DOF!

Only 1 iteration!

The most efficient algorithm for removing outliers, up to 1000 Hz!

$\theta = -2 \tan^{-1} \left( \frac{v_2 - v_1}{u_2 + u_1} \right)$

Compute $\theta$ for every point correspondence.
Comparison of RANSAC algorithms

\[ N = \frac{\log(1 - p)}{\log(1 - (1 - \varepsilon)^s)} \] where we typically use \( p = 99\% \)

<table>
<thead>
<tr>
<th>Numb. of iterations</th>
<th>5-Point RANSAC [Nister’03]</th>
<th>2-Point RANSAC [Ortin’01]</th>
<th>1-Point RANSAC [Scaramuzza, IJCV’10]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>&gt;145</td>
<td>&gt;16</td>
<td>1</td>
</tr>
</tbody>
</table>
Visual Odometry with 1-Point RANSAC

Work in different environments

Urban

Today’s outline

• Review of last lecture
• RANSAC for robust Structure from Motion
• Visual Odometry
• Use case: SVO: Semi-direct Visual Odometry (developed by us)
Visual Odometry (VO)

VO is the process of incrementally estimating the pose of the vehicle by examining the changes that motion induces on the images of its onboard cameras.

**Input**

Image sequence (or video stream) from one or more cameras attached to a moving vehicle.

**Output**

\[ R_0, R_1, \ldots, R_i \]
\[ t_0, t_1, \ldots, t_i \]

Camera trajectory (3D structure is a plus):
Example 1: VO for Phones

Application to Augmented Reality for smartphones
Example 2: VO for Flying Robots

[Scaramuzza et al., Vision-Controlled Micro Flying Robots: from System Design to Autonomous Navigation and Mapping in GPS-denied Environments, IEEE RAM, September, 2014]
Example 3: VO for Mouse Scanners

World-first mouse scanner

Currently distributed by LG: SmartScan LG LSM100
Assumptions

- Sufficient illumination in the environment
- Dominance of static scene over moving objects
- Enough texture to allow apparent motion to be extracted
- Sufficient scene overlap between consecutive frames

Is any of these scenes good for VO? Why?
Why VO?

• Contrary to wheel odometry, VO is not affected by wheel slip on uneven terrain or other adverse conditions.

• More accurate trajectory estimates compared to wheel odometry (relative position error 0.1% – 2%)

• VO can be used as a complement to
  – wheel odometry
  – GPS
  – inertial measurement units (IMUs)
  – laser odometry

• In GPS-denied environments, such as underwater and aerial, VO has utmost importance
**VO work flow**

- VO computes the camera path incrementally (pose after pose)

1. Image sequence
2. Feature detection
3. Feature matching (tracking)
4. Motion estimation
   - 2D-2D
   - 3D-3D
   - 3D-2D
5. Local optimization

SIFT features tracks
VO or Structure from Motion (SFM)?

SFM is more general than VO and tackles the problem of 3D reconstruction of both the structure and camera poses from unordered image sets.

Reconstruction from 3 million images from Flickr.com
Cluster of 250 computers, 24 hours of computation!
Paper: “Building Rome in a Day”, ICCV’09
VO or Structure from Motion (SFM)?

- VO is a particular case of SFM.

- VO focuses on estimating the 3D motion of the camera sequentially (as a new frame arrives) and in real time.

- Terminology: sometimes SFM is used as a synonym of VO.
A Brief history of VO


- 1980 to 2000: The VO research was dominated by NASA/JPL in preparation of 2004 Mars mission (see papers from Matthies, Olson, etc. From JPL)

- 2004: VO used on a robot on another planet: Mars rovers Spirit and Opportunity

- 2004: VO was revived in the academic environment by Nister «Visual Odometry» paper. The term VO became popular.
Motion Estimation

• Motion estimation is the core computation step performed for every image in a VO system
• It computes the camera motion $T_k$ between the previous and the current image:

$$T_k = \begin{bmatrix} R_{k,k-1} & t_{k,k-1} \\ 0 & 1 \end{bmatrix}$$

• By concatenation of all these single movements, the full trajectory of the camera can be recovered
Motion Estimation

- Motion estimation is the core computation step performed for every image in a VO system.
- It computes the camera motion $T_k$ between the previous and the current image:
  \[
  T_k = \begin{bmatrix}
  R_{k,k-1} & t_{k,k-1} \\
  0 & 1
  \end{bmatrix}
  \]
- By concatenation of all these single movements, the full trajectory of the camera can be recovered.
- An iterative refinement over the last $m$ poses can be performed to get a more accurate estimate of the local trajectory.
Triangulated 3D points are determined by intersecting backprojected rays from 2D image correspondences of at least two image frames.

In reality, they never intersect due to:
- image noise,
- camera model and calibration errors,
- and feature matching uncertainty.

The point at minimal distance from all intersecting rays can be taken as an estimate of the 3D point position.
Triangulation and Keyframe Selection

- When frames are taken at nearby positions compared to the scene distance, 3D points will exhibit large uncertainty.
Triangulation and Keyframe Selection

• One way to avoid this consists of skipping frames until the average uncertainty of the 3D points decreases below a certain threshold. The uncertainty can be computed by intersecting back-projected cones. The selected frames are called keyframes.

• Keyframe selection is a very important step in VO and should always be done before updating the motion.
Camera-Pose Optimization

- So far we assumed that the transformations are between consecutive frames.

\[ T_{e_{ij}} \]

- Transformations can be computed also between non-adjacent frames \( T_{e_{ij}} \) and can be used as additional constraints to improve cameras poses by minimizing the following:

\[
\sum_{e_{ij}} \| C_i - T_{e_{ij}} C_j \|^2
\]

- For efficiency, only the last \( m \) keyframes are used.
- Gauss-Newton or Levenberg-Marquadt can be used.
Bundle Adjustment (BA)

• Similar to pose-optimization but it also optimizes 3D points

\[
\arg\min_{X^i, C_k} \sum_{i,k} \| p^i_k - g(X^i, C_k) \|^2
\]

• In order to not get stuck in local minima, the initialization should be close to the minimum

• Gauss-Newton or Levenberg-Marquadt can be used
References: a two-part tutorial


Today’s outline

• Review of last lecture
• RANSAC for robust Structure from Motion
• Visual Odometry

• Use case: SVO: Semi-direct Visual Odometry (developed by us)
Use case: SVO: Semi-direct Visual Odometry

- Developed between 2013 and 2014 at the Robotics and Perception Group at the University of Zurich by Christian Forster, Matia Pizzoli, and Davide Scaramuzza

Keyframe-based Visual Odometry

Keyframe 1

Keyframe 2

Current frame

New keyframe

Initial pointcloud

New triangulated points
**Feature-based vs. direct**

**Feature-based (e.g., PTAM)**

1. Feature extraction
2. Feature matching
3. RANSAC + P3P
4. Reprojection error minimization

\[ T_{k, k-1} = \arg \min_T \sum_i \| I_k(u'_i) - I_{k-1}(u_i) \|^2 \]

**Direct approaches**

\[ T_{k, k-1} = \arg \min_T \sum_i \| I_k(u'_i) - I_{k-1}(u_i) \|^2 \]

**Our solution:**
Combine feature-based and direct methods

**SVO: Semi-direct Visual Odometry**

[Soatto’95, Meillard and Comport, IROS 2013], DVO [Kerl et al., ICRA 2013], DTAM [Newcombe et al., ICCV ‘11], ...
SVO: Semi-direct Visual Odometry

Depth-Filter:

- Depth-filter for every new feature
- Recursive Bayesian depth estimation
- Epipolar search using SSD

Measurement Likelihood models outliers:

\[
p(d_i^k | d_i, \rho_i) = \rho_i \mathcal{N}(\tilde{d}_i^k | d_i, \tau_i^2) + (1 - \rho_i) \mathcal{U}(\tilde{d}_i^k | d_i^{\text{min}}, d_i^{\text{max}})
\]

[Forster, Pizzoli, Scaramuzza, «SVO: Semi Direct Visual Odometry», ICRA’14]
SVO Results: Depth Estimation

[Forster, Pizzoli, Scaramuzza, «SVO: Semi Direct Visual Odometry», ICRA’14]
Visual Odometry Results

[Forster, Pizzoli, Scaramuzza, «SVO: Semi Direct Visual Odometry», ICRA’14]
Onboard computer for image analysis
Quad Core Odroid (ARM Cortex A-9):
Same of Samsung Galaxy S4 phones

170 grams!
Autonomous Vision-based Flight over a Disaster Zone
Firefighters’ training area, Zurich
Flight Results: Indoor, aggressive flight
Speed: 4 m/s, height: 1.5 m – Down-looking camera
Flight Results: Indoor, aggressive flight
Speed: 4 m/s, height: 1.5 m – Down-looking camera

Open Source available at: github.com/uzh-rpg/rpg_svo
Works with and without ROS
Closed-Source professional edition available for companies