Introduction to Visual Odometry

Karel Zimmermann

Czech Technical University in Prague
Faculty of Electrical Engineering, Department of Cybernetics
Center for Machine Perception
http://cmp.felk.cvut.cz/~zimmerk, zimmerk@fel.cvut.cz

Some images and codes taken from D.Scaramuzza
Localization sensors I

There are many localization-suitable sensors. Which pros, cons?
Localization sensors I

There are many localization-suitable sensors. Which pros, cons?

- **GPS**: Global Positioning System
  - Inaccurate.
  - Does not work indoor
Localization sensors I

There are many localization-suitable sensors. Which pros, cons?

- **GPS**: Global Positioning System
  - Inaccurate.
  - Does not work indoor

- **IMU**: Accelerometers + Gyroscope + Magnetometers.
  - Fuse all sensors data to get 6DOF (dead-reckoning).
  - Suffers from drift.
Localization sensors I

There are many localization-suitable sensors. Which pros, cons?

- **GPS**: Global Positioning System
  - Inaccurate.
  - Does not work indoor

- **IMU**: Accelerometers + Gyroscope + Magnetometers.
  - Fuse all sensors data to get 6DOF (dead-reckoning).
  - Suffers from drift.

- **Wi-fi**:
  - Map Wi-Fi routers in advance, than guess the location based on IDs and signal strength.
  - Does not work when routers are unavailable.
  - Inaccurate.
Localization sensors II

- **Odometry**: Integrate wheel motion and steering via (kinematic) model.
  - Does not work on slippage terrain.
  - Drift + Inaccurate (show video !!!).
Localization sensors II

- **Odometry**: Integrate wheel motion and steering via (kinematic) model.
  - Does not work on slippage terrain.
  - Drift + Inaccurate (show video !!!).

- **What about camera?**
  - cheap + light
  - every mobile device has a camera.
  - high frame-rate.
Localization sensors II

- **Odometry:** Integrate wheel motion and steering via (kinematic) model.
  - Does not work on slippage terrain.
  - Drift + Inaccurate (show video !!!).

- **What about camera?**
  - cheap + light
  - every mobile device has a camera.
  - high frame-rate.

- **What about depth sensors?** - show video !!!
  - stereo camera
  - structured light approach (e.g. Kinect)
  - time-of-flight approach (e.g. Velodyne)
Task definition

- **Input:** Camera images.
- **Output:** Real-time robot location.
Main principle

- Find correspondences in two consecutive frames.
Main principle

- Find correspondences in two consecutive frames.
- If the world is static and correspondences are correct then estimate camera motion $R$ and $t$ (and 3D reconstruction $X$) minimizing reprojection error.
Main principle

- Find correspondences in two consecutive frames.
- If the world is **static** and correspondences are **correct** then estimate camera motion $R$ and $t$ (and 3D reconstruction $X$) minimizing reprojection error.
- Since the world is **dynamic** and most of the correspondences are **incorrect** we need robust method - show RANSAC presentation!!
Algorithm at glance

1. Get image $I_k$.

2. Compute correspondences between $I_{k-1}$ and $I_k$ (either feature matching or tracking).

3. Find correct correspondences and compute essential matrix $E$.

4. Decompose $E$ into $R_k$ and $t_k$.

5. Compute 3D model (points $X$).

6. Rescale $t_k$ according to relative scale $r$.

7. $k = k + 1$
Feature point detection

- Which points are suitable?
Feature point detection

Feature points must be well distinguishable from its neighbourhood.

\[
E(u, v) = \sum_{x,y} \left( I(x+u, y+v) - I(x, y) \right)^2 \approx [u \ v] \ M \ [u \ v]
\]

\(\lambda_1 \text{ and } \lambda_2 \text{ are large}\)
Feature point detection

- Feature points must be well distinguishable from its neighbourhood.

\[ E(u, v) = \sum_{x, y} \left( I(x + u, y + v) - I(x, y) \right)^2 \approx [u \ v] \ M \ [u \ v] \]

large $\lambda_1$, small $\lambda_2$
Feature point detection

Feature points must be well distinguishable from its neighbourhood.

\[ E(u, v) = \sum_{x, y} \left( I(x + u, y + v) - I(x, y) \right)^2 \approx [u \ v] \mathbf{M} \begin{bmatrix} u \\ v \end{bmatrix} \]
Feature point detection

- Detected feature points
- Show video Vodom_video_Prato_roof.avi
Estimating correspondences

- There are two ways:
  - Tracking - for high **temporal** resolution
    - OpenCV Lucas-Kanade tracker
  - Descriptor and matching - for high **spatial** resolution
    - OpenCV: SIFT, SURF etc ...

- show video vo_ros_PR2.avi
1. Get image $I_k$.

2. Compute correspondences between $I_{k-1}$ and $I_k$ (either feature matching or tracking).

3. **Find correct correspondences and compute essential matrix $E$.**

4. Decompose $E$ into $R_k$ and $t_k$.

5. Compute 3D model (points $X$).

6. Rescale $t_k$ according to relative scale $r$.

7. $k = k + 1$
Compute essential matrix

- Forget for a moment that we have already estimated correspondences.
Compute essential matrix

- Forget for a moment that we have already estimated correspondences.
- Given cameras $C_1, C_2$ (including their rotations) and point $u$, where is corresponding point $v$?
Compute essential matrix

- Forget for a moment that we have already estimated correspondences.
- Given cameras $C_1, C_2$ (including their rotations) and point $u$, where is corresponding point $v$?
- That depends on 3D point $X$. 
Compute essential matrix

- Forget for a moment that we have already estimated correspondences.
- Given cameras $C_1, C_2$ (including their rotations) and point $u$, where is corresponding point $v$?
- That depends on 3D point $X$. 

![Diagram with points u, v, X, X', and camera configurations C1, C2]
**Compute essential matrix**

- Forget for a moment that we have already estimated correspondences.
- Given cameras $C_1, C_2$ (including their rotations) and point $u$, where is corresponding point $v$?
- That depends on 3D point $X$.
- All possible correspondences lie on epipolar line $\{v \mid u^T E v = 0\}$.
Compute essential matrix

- Forget for a moment that we have already estimated correspondences.
- Given cameras $C_1, C_2$ (including their rotations) and point $u$, where is corresponding point $v$?
- That depends on 3D point $X$.
- All possible correspondences lie on epipolar line $\{v \mid u^T E v = 0\}$.
Compute essential matrix

- All possible correspondences lie on epipolar line \( \{ v \mid u^T E v = 0 \} \).

- Note:
  - This holds only if \( K \) is identity matrix (or \( u, v \) are decalibrated).
  - Putting projection rays of all possible 3D points \( X \) together yields epipolar plane (yellow).
Compute essential matrix

- All possible correspondences lie on epipolar line $\{v \mid u^T E v = 0\}$.
- Note:
  - This holds only if $K$ is identity matrix (or $u, v$ are decalibrated).
  - Putting projection rays of all possible 3D points $X$ together yields epipolar plane (yellow).
  - There exist unique decomposition $E = [t] \times R \Rightarrow$ camera motion!
Compute essential matrix

- Let us assume that we have several correspondences.
Compute essential matrix

- Let us assume that we have several correspondences.
- Essential matrix $E$ is just a solution of (overdetermined) homogeneous system of linear equations.
Compute essential matrix

Let us assume that we have several correspondences.

Essential matrix $E$ is just a solution of (overdetermined) homogeneous system of linear equations.

For each correspondence pair $u, v$, the following holds:

$$u^T E v = u^T \begin{bmatrix} e_1^T \\ e_2^T \\ e_3^T \end{bmatrix} v = u^T \begin{bmatrix} e_1^T v \\ e_2^T v \\ e_3^T v \end{bmatrix} = [u_1 e_1^T v + u_2 e_2^T v + u_3 e_3^T v] =$$
Compute essential matrix

- Let us assume that we have several correspondences.
- Essential matrix $E$ is just a solution of (overdetermined) homogeneous system of linear equations.
- For each correspondence pair $u, v$, the following holds:
  \[
  u^T E v = u^T \begin{bmatrix} e_1^T \\ e_2^T \\ e_3^T \end{bmatrix} v = u^T \begin{bmatrix} e_1^T v \\ e_2^T v \\ e_3^T v \end{bmatrix} = [u_1 e_1^T v + u_2 e_2^T v + u_3 e_3^T v] = \\
  = [u_1 v^T u_2 v^T u_3 v^T] \begin{bmatrix} e_1 \\ e_2 \\ e_3 \end{bmatrix} = 0
  \]
- It must hold for all correspondence pairs $u_i, v_i$, therefore:
  \[
  \begin{bmatrix}
  u_{11} v_1^T & u_{12} v_1^T & u_{13} v_1^T \\
  u_{21} v_2^T & u_{22} v_2^T & u_{23} v_2^T \\
  \vdots & \vdots & \vdots 
  \end{bmatrix} \begin{bmatrix} e_1 \\ e_2 \\ e_3 \end{bmatrix} = 0
  \]
Compute essential matrix

- It is just homogeneous set of linear equations:

\[
\begin{bmatrix}
  u_{11}v_1^\top & u_{12}v_1^\top & u_{13}v_1^\top \\
  u_{21}v_2^\top & u_{22}v_2^\top & u_{23}v_2^\top \\
  \vdots & \vdots & \vdots
\end{bmatrix}
\begin{bmatrix}
e_1 \\
e_2 \\
e_3
\end{bmatrix}
= 0
\]

\[
A
\]
\[
e
\]

- Again we want to avoid trivial solution \( e_1 = e_2 = e_3 = 0 \).

- We solve the following optimization task (constrained LSQ)

\[
\arg \min_e \| Ae \| \text{ subject to } \| e \| = 1
\]

- the solution is singular vector of matrix \( A \) corresponding to the smallest singular value (can be found via SVD or eigenvectors/eigenvalues of \( AA^\top \))
Compute essential matrix

- Since L2 is sensitive to outliers, RANSAC is used to find inliers (i.e. correct correspondences).

- Since algebraic error is minimized (instead of geometric error), coordinates have to be normalized.
Algorithm at glance

1. Get image $I_k$.

2. Compute correspondences between $I_{k-1}$ and $I_k$ (either feature matching or tracking).

3. Find correct correspondences and compute essential matrix $E$.

4. Decompose $E$ into $R_k$ and $t_k$.

5. Compute 3D model (points $X$).

6. Rescale $t_k$ according to relative scale $r$.

7. $k = k + 1$
Decompose $E$ into $R$ and $t$

- Once you find $E$, you can estimate camera motion by decomposing $E = [t]_\times R$ via SVD ($E = U\Sigma V^T$) as follows: $[t]_\times = VW\Sigma V^T$, $R = UW^{-1}V^T$, but !!!:
Decompose $E$ into $R$ and $t$

Once you find $E$, you can estimate camera motion by decomposing $E = [t] \times R$ via SVD ($E = U\Sigma V^\top$) as follows: $[t] \times = VW\Sigma V^\top$, $R = UW^{-1}V^\top$, but !!!:

- Scale is unknown (if something is equal to zero, then any scalar multiplication of it is equal to zero as well).
Once you find $E$, you can estimate camera motion by decomposing $E = [t] \times R$ via SVD ($E = U\Sigma V^T$) as follows: $[t] \times = VW\Sigma V^T$, $R = UW^{-1}V^T$, but !!!: Scale is unknown (if something is equal to zero, then any scalar multiplication of it is equal to zero as well).
Estimating camera motion

- Once you find $E$, you can estimate camera motion by decomposing $E = [t] \times R$ via

- SVD ($E = U \Sigma V^\top$) as follows: $[t] \times = VW\Sigma V^\top$, $R = UW^{-1}V^\top$, but !!!:
  - Scale is unknown (if something is equal to zero, then any scalar multiplication of it is equal to zero as well).
  - We search for 8 unknowns ($\dim(e) = 9$ minus scale) $\Rightarrow$ at least 8 correspondences needed $\Rightarrow$ 8-point algorithm.
Estimating camera motion

- Once you find $E$, you can estimate camera motion by decomposing $E = [t]_\times R$ via SVD ($E = U\Sigma V^\top$) as follows: $[t]_\times = VW\Sigma V^\top$, $R = UW^{-1}V^\top$, but !!!:
  - Scale is unknown (if something is equal to zero, then any scalar multiplication of it is equal to zero as well).
  - We search for 8 unknowns ($\dim(e) = 9$ minus scale) $\Rightarrow$ at least 8 correspondences needed $\Rightarrow$ 8-point algorithm.
  - However you want to find only camera translation (3 DoFs) and rotation (3 DoFs) minus scale $\Rightarrow$ 5-point algorithm [Nister 2003].
Estimating camera motion

- Once you find $E$, you can estimate camera motion by decomposing $E = [t] \times R$ via SVD ($E = U \Sigma V^\top$) as follows: $[t] \times = VW \Sigma V^\top$, $R = UW^{-1}V^\top$, but !!!:
  - Scale is unknown (if something is equal to zero, then any scalar multiplication of it is equal to zero as well).
  - We search for 8 unknowns ($\text{dim}(e) = 9$ minus scale) $\Rightarrow$ at least 8 correspondences needed $\Rightarrow$ 8-point algorithm.
  - However you want to find only camera translation (3 DoFs) and rotation (3 DoFs) minus scale $\Rightarrow$ 5-point algorithm [Nister 2003].
  - Why is it such a big deal???
Estimating camera motion

- Once you find $E$, you can estimate camera motion by decomposing $E = [t] \times R$ via

- SVD ($E = U \Sigma V^\top$) as follows: $[t] \times = V W \Sigma V^\top$, $R = U W^{-1} V^\top$, but !!!:
  
  - Scale is unknown (if something is equal to zero, then any scalar multiplication of it is equal to zero as well).
  
  - We search for 8 unknowns ($\text{dim}(e) = 9$ minus scale) $\Rightarrow$ at least 8 correspondences needed $\Rightarrow$ 8-point algorithm.

  - However you want to find only camera translation (3 DoFs) and rotation (3 DoFs) minus scale $\Rightarrow$ 5-point algorithm [Nister 2003].

  - Why is it such a big deal???

  - Because, usually you have 80% outliers and you need to use RANSAC (explain in the end of the lecture).
Algorithm at glance

1. Get image $I_k$.

2. Compute correspondences between $I_{k-1}$ and $I_k$ (either feature matching or tracking).

3. Find correct correspondences and compute essential matrix $E$.

4. Decompose $E$ into $R_k$ and $t_k$.

5. Compute 3D model (points $X$).

6. Rescale $t_k$ according to relative scale $r$.

7. $k = k + 1$
Compute 3D model

- Scene point \( X \) is observed by two cameras \( P \) and \( Q \).
- Let \( \mathbf{u} = [u_1 \ u_2]^{\top} \) and \( \mathbf{v} = [v_1 \ v_2]^{\top} \) are projections of \( X \) in \( P \) and \( Q \).
- Then

\[
u_1 = \frac{\mathbf{p}_1^{\top} \mathbf{X}}{\mathbf{p}_3^{\top} \mathbf{X}} \quad \Rightarrow \quad u_1 \mathbf{p}_3^{\top} \mathbf{X} - \mathbf{p}_1^{\top} \mathbf{X} = 0
\]
Compute 3D model

- Scene point \( X \) is observed by two cameras \( P \) and \( Q \).

- Let \( u = [u_1 \ u_2]^\top \) and \( v = [v_1 \ v_2]^\top \) are projections of \( X \) in \( P \) and \( Q \).

- Then

\[
\begin{align*}
    u_1 &= \frac{p_1^\top X}{p_3^\top X} \implies u_1 p_3^\top X - p_1^\top X = 0 \\
    u_2 &= \frac{p_2^\top X}{p_3^\top X} \implies u_2 p_3^\top X - p_2^\top X = 0 \\
    v_1 &= \frac{q_1^\top X}{q_3^\top X} \implies v_1 q_3^\top X - q_1^\top X = 0 \\
    v_2 &= \frac{q_2^\top X}{q_3^\top X} \implies v_2 q_3^\top X - q_2^\top X = 0
\end{align*}
\]
Compute 3D model

- Which is $4 \times 4$ homogeneous system of linear equations:

$$
\begin{bmatrix}
u_1 p_3^\top - p_1^\top \\
u_2 p_3^\top - p_2^\top \\
v_1 q_3^\top - q_1^\top \\
v_2 q_3^\top - q_2^\top
\end{bmatrix} \times = 0
$$
Compute 3D model

- Which is $4 \times 4$ homogeneous system of linear equations:

$$\begin{bmatrix}
u_1 p_3^\top - p_1^\top \\
u_2 p_3^\top - p_2^\top \\
u_1 q_3^\top - q_1^\top \\
u_2 q_3^\top - q_2^\top \\
\end{bmatrix} X = 0$$

- Algebraic error is minimized (it often yields small geometric error).

- To obtain a better 3D model, the reprojection (geometric) error is locally minimized by Levenberg-Marquardt method:

$$\arg \min_{p^i, X_j} \sum_{i,j} d(P^i X_j, u_j^i)^2$$

- It is often called the bundle adjustment.
Algorithm at glance

1. Get image $I_k$.

2. Compute correspondences between $I_{k-1}$ and $I_k$ (either feature matching or tracking).

3. Find correct correspondences and compute essential matrix $E$.

4. Decompose $E$ into $R_k$ and $t_k$.

5. Compute 3D model (points $X$).

6. Rescale $t_k$ according to relative scale $r$.

7. $k = k + 1$
Estimating camera motion - relative scale

1. You cannot get absolute scale (without calibration object).
Estimating camera motion - relative scale

1. You cannot get absolute scale (without calibration object).

2. If you estimate motion (and 3D model) from $C_1, C_2$
1. You cannot get absolute scale (without calibration object).

2. If you estimate motion (and 3D model) from $C_1, C_2$ and then from $C_2, C_3$ you can have completely different scale.
Estimating camera motion - relative scale

1. You cannot get absolute scale (without calibration object).
2. If you estimate motion (and 3D model) from $C_1, C_2$ and than from $C_2, C_3$ you can have completely different scale.
3. You want to keep the same relative scale $r$ by rescaling $t$ (and 3D)

\[ r = \frac{d_k}{d_{k-1}} = \frac{\|X_k - Y_k\|}{\|X_{k-1} - Y_{k-1}\|} \]
Drift

- Error accumulates over time $\Rightarrow$ drift $\Rightarrow$ loop-closure needed.

- Keyframe detection (avoid motion estimation for small motion or pure rotation) - show video vo_ros_PR2.avi
Visual Odometry (VO), Structure from Motion (SFM),
Visual Simultaneous Localisation and Mapping (VSLAM)

- SFM (3D from unordered set of images) show video sfm_colloseum.avi
**Visual Odometry (VO), Structure from Motion (SFM), Visual Simultaneous Localisation and Mapping (VSLAM)**

- SFM (3D from unordered set of images) show video sfm_colloseum.avi
- VO sequential and real-time camera motion estimation.
Visual Odometry (VO), Structure from Motion (SFM), Visual Simultaneous Localisation and Mapping (VSLAM)

- SFM (3D from unordered set of images) show video sfm_colloseum.avi
- VO sequential and real-time camera motion estimation.
- VSLAM is VO with loop closer (bundle adjustment) - show video vo_kinect.avi

Before loop closing  

After loop closing