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+Magnetomers.
  - 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+Magnetomers.
  - 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.

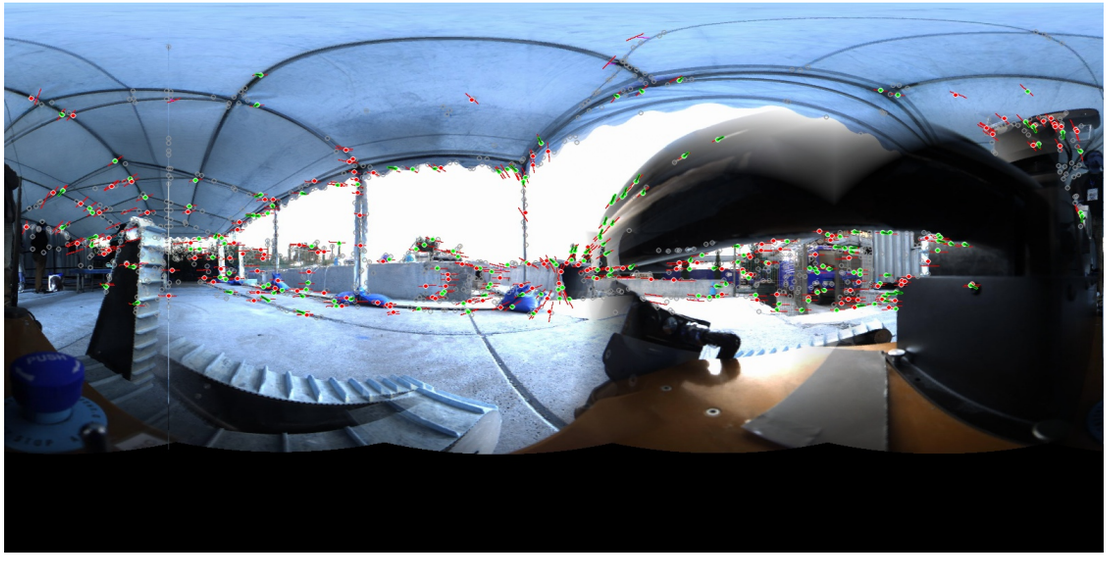
Localization sensors II

- **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)

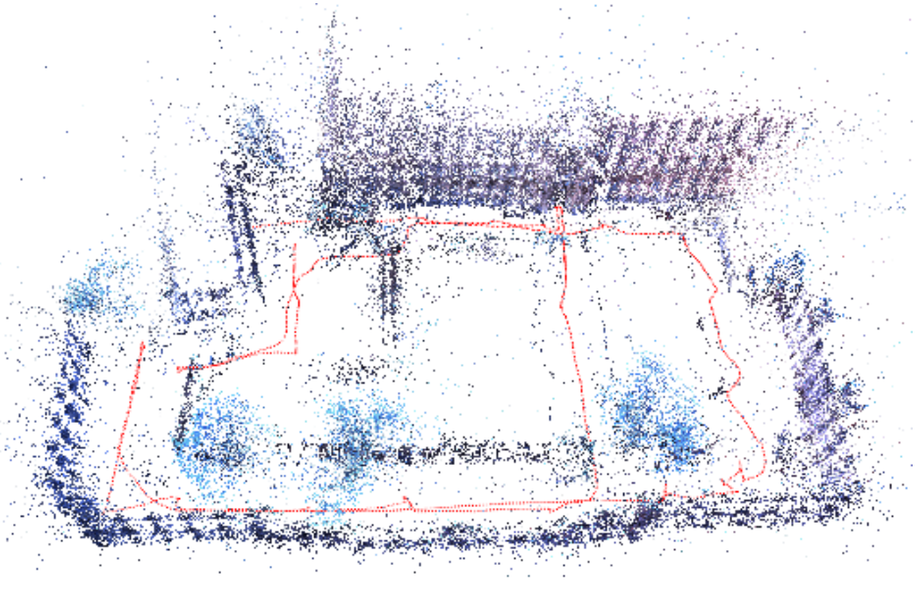
Task definition

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

input

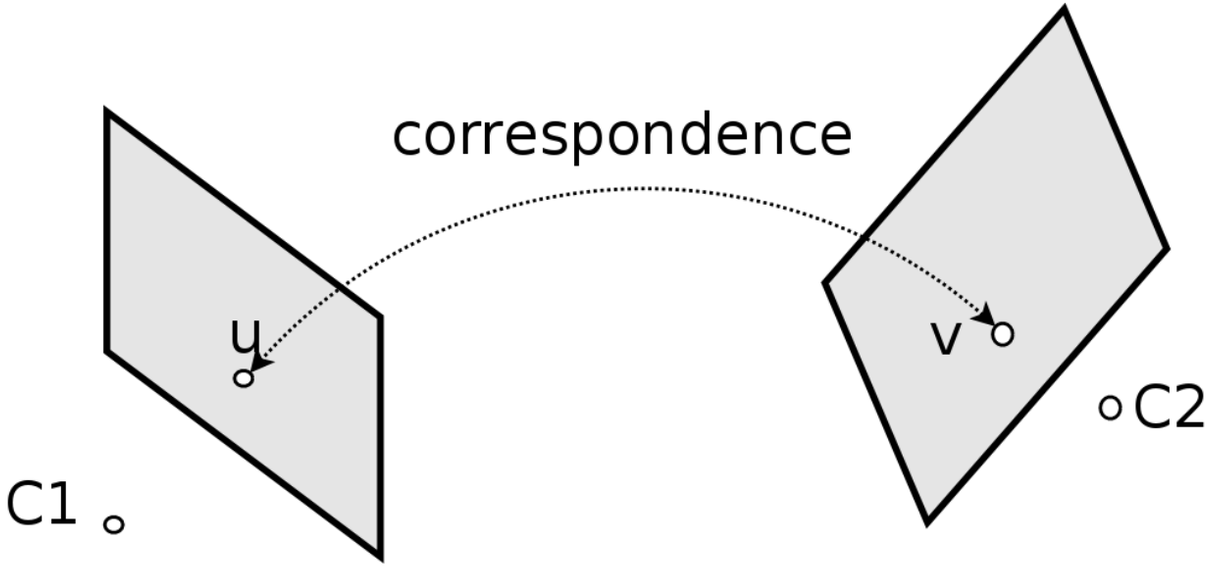


output



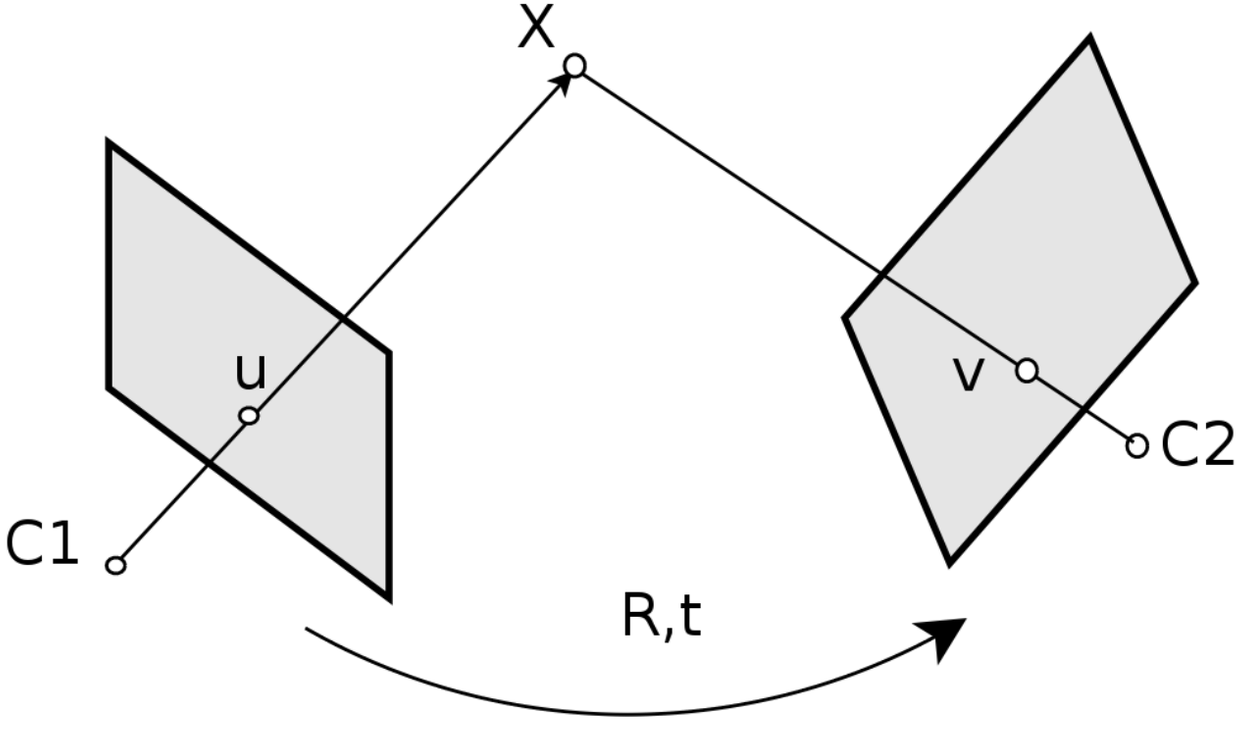
Main principle

- Find correspondences in two consecutive frames.



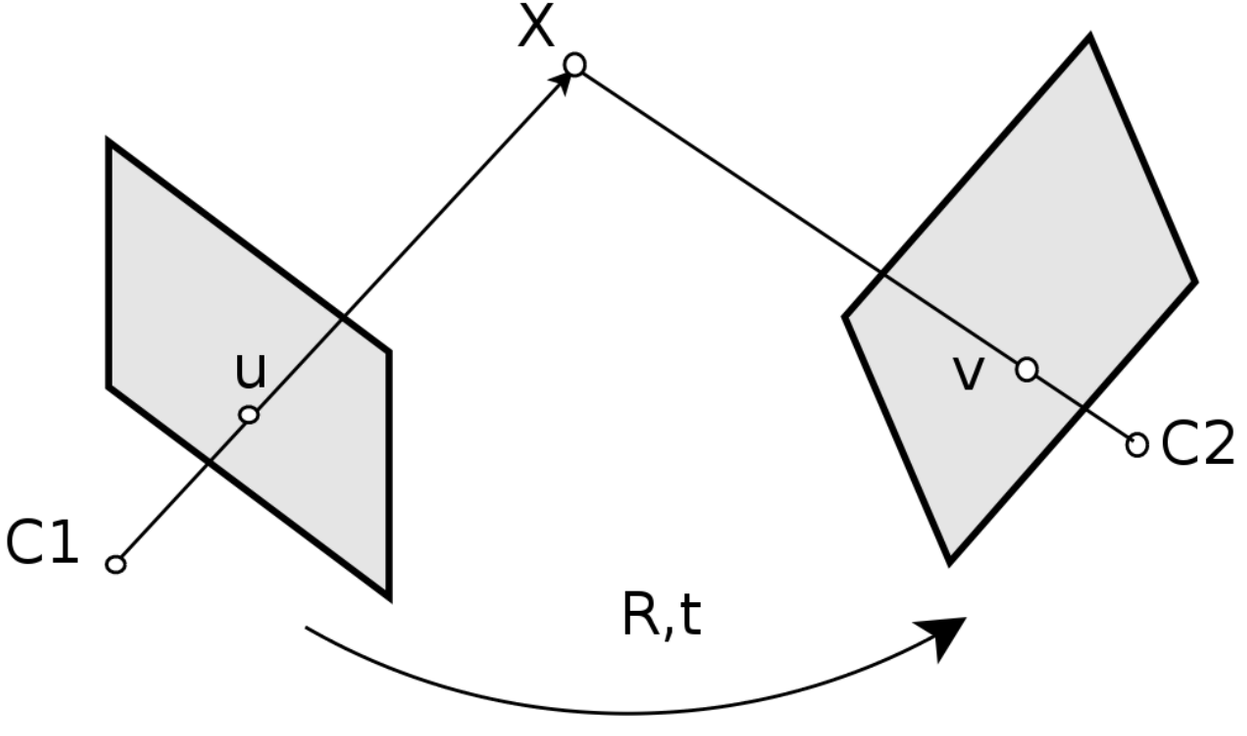
Main principle

- 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 $\mathtt{E}$
4. Decompose $\mathtt{E}$ into $\mathtt{R}_k$ and $\mathbf{t}_k$.
5. Compute 3D model (points X).
6. Rescale t_k according to relative scale r.
7. $k = k + 1$

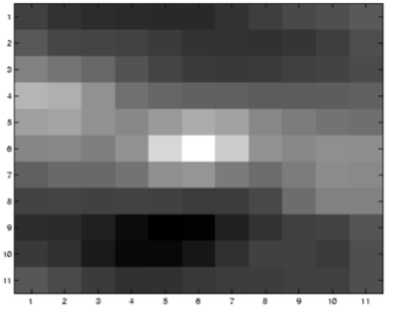
Feature point detection

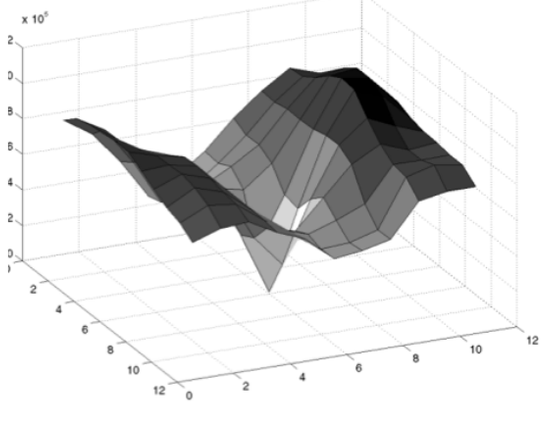
- Which points are suitable?

Feature point detection

- Feature points must be well distinguishable from its neighbourhood.

$$E(u,v) = \sum_{x,y} \Big(I(x+u, y+v) - I(x,y) \Big)^2 \approx [u\ v]\ \mathtt{M} \begin{bmatrix} u \\ v \end{bmatrix}$$



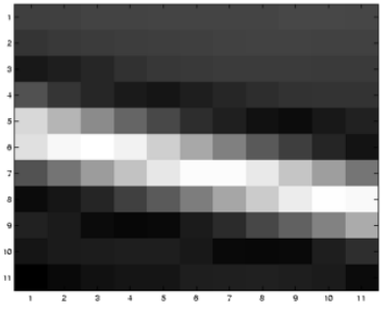


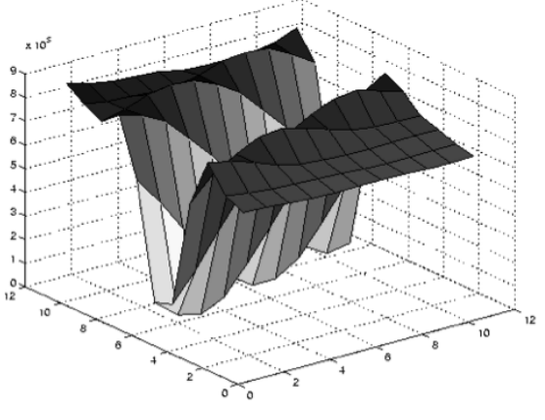
λ_1 and λ_2 are large

Feature point detection

- Feature points must be well distinguishable from its neighbourhood.

$$E(u,v) = \sum_{x,y} \Big(I(x+u, y+v) - I(x,y) \Big)^2 \approx [u\ v]\ \mathtt{M} \begin{bmatrix} u \\ v \end{bmatrix}$$



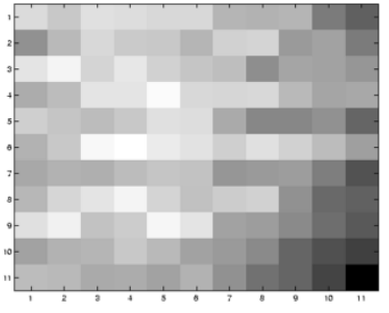


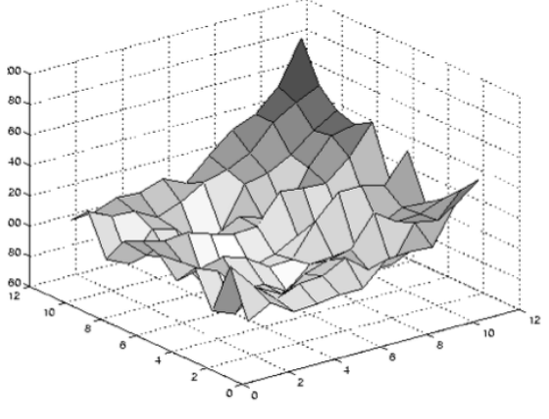
large λ_1, small λ_2

Feature point detection

- Feature points must be well distinguishable from its neighbourhood.

$$E(u,v) = \sum_{x,y} \Big(I(x+u, y+v) - I(x,y) \Big)^2 \approx [u\ v]\ \mathtt{M} \begin{bmatrix} u \\ v \end{bmatrix}$$

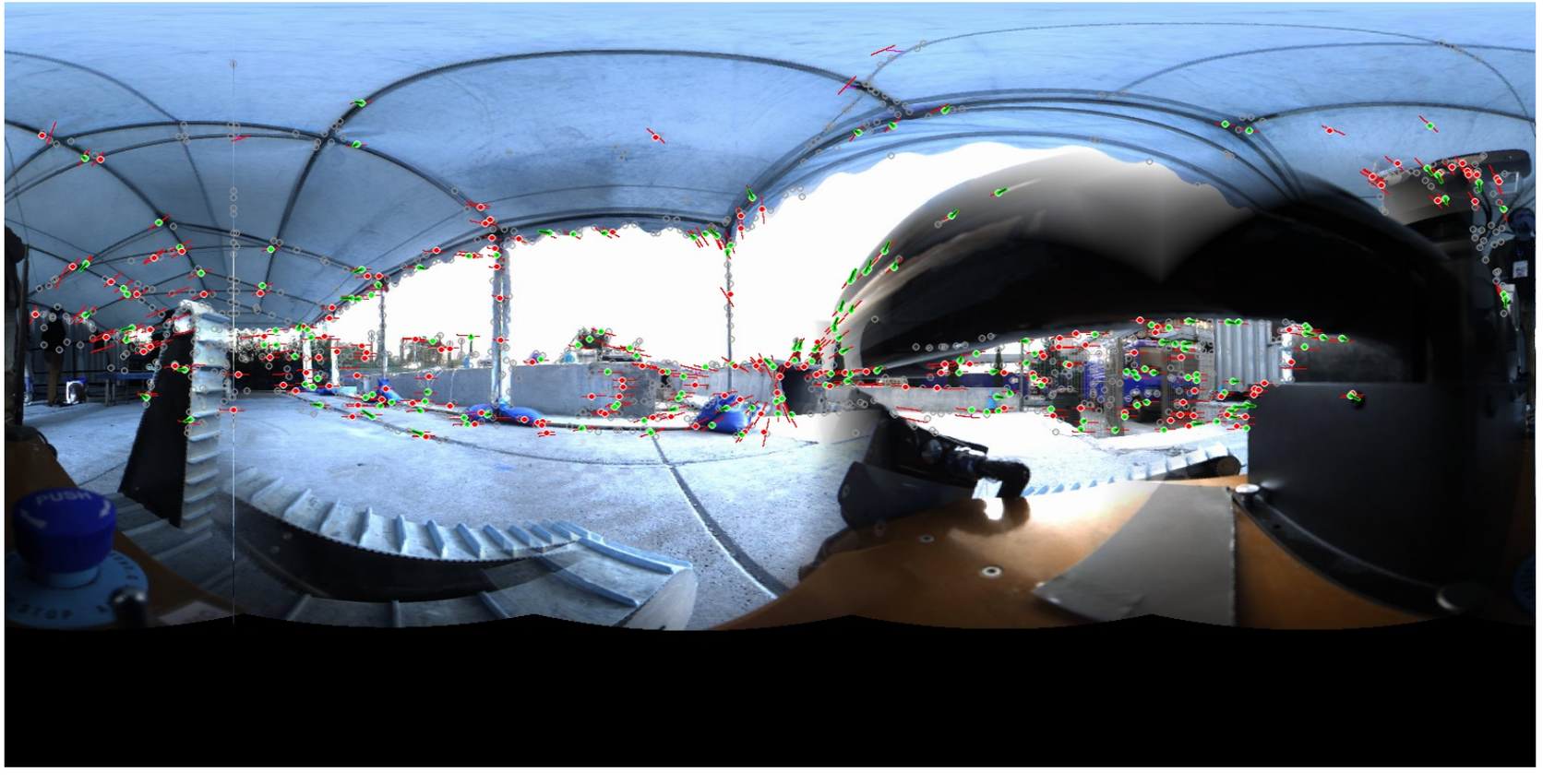




small λ_1, small λ_2

Feature point detection

- Detected feature points
- Show video `Vodom_video_Prato_roof.avi`

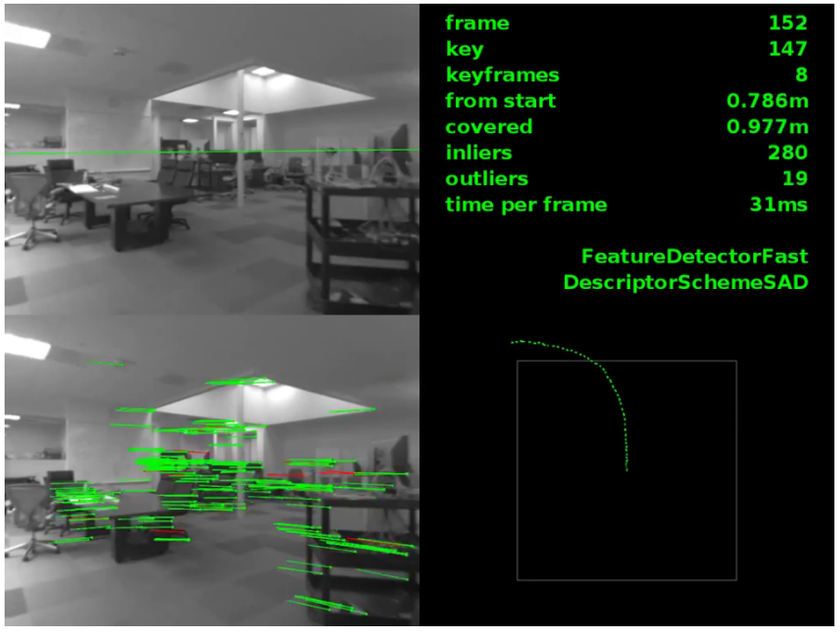


Estimating correspondences

- 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`



Algorithm at glance

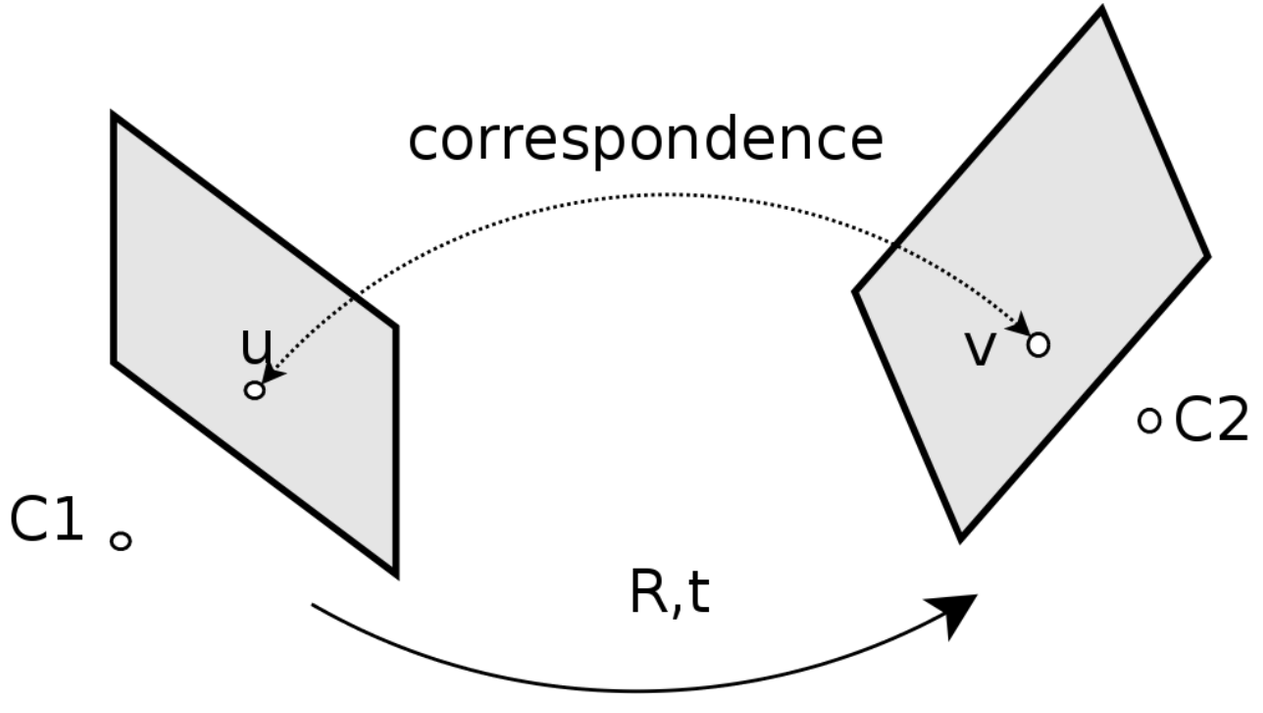
1. Get image I_k.
2. Compute correspondences between I_{k-1} and I_k (either feature matching or tracking).
3. Find correct correspondences and compute essential matrix $\mathtt{E}$.
4. Decompose $\mathtt{E}$ into $\mathtt{R}_k$ and $\mathbf{t}_k$.
5. Compute 3D model (points X).
6. Rescale t_k according to relative scale r.
7. $k = k + 1$

Compute essential matrix

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

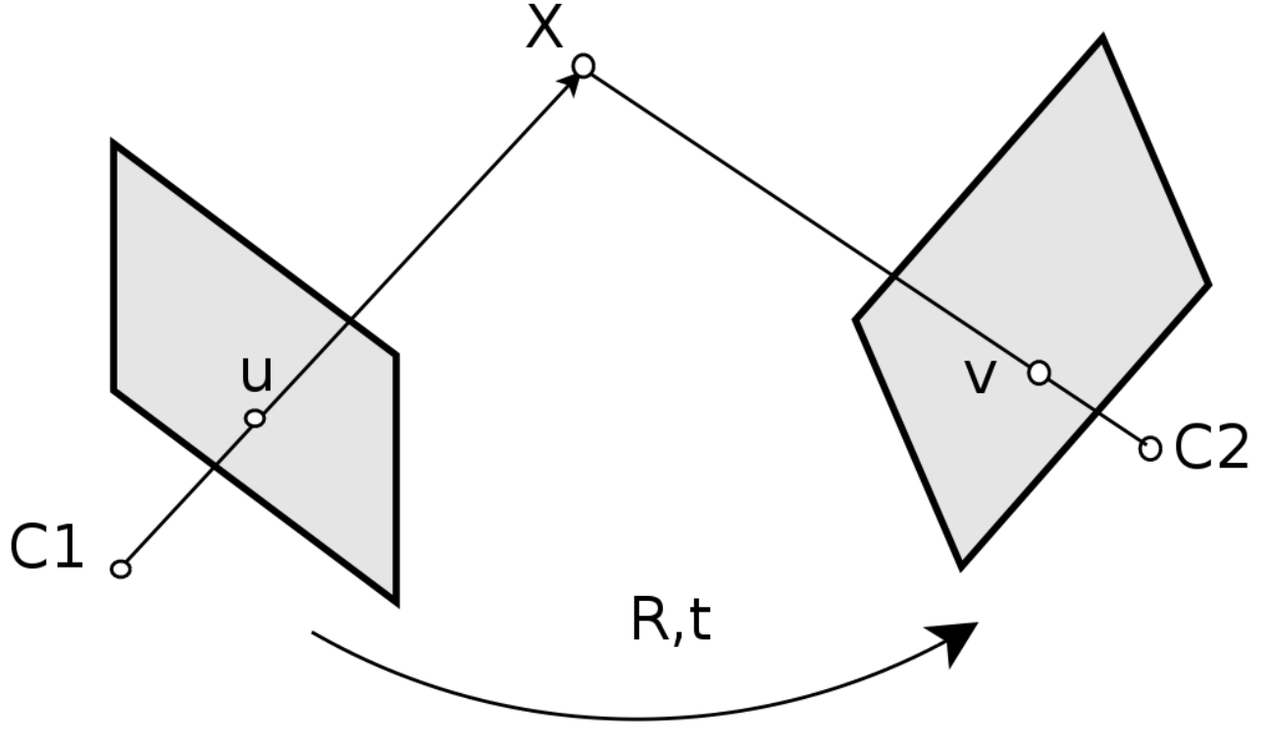
Compute essential matrix

- 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?



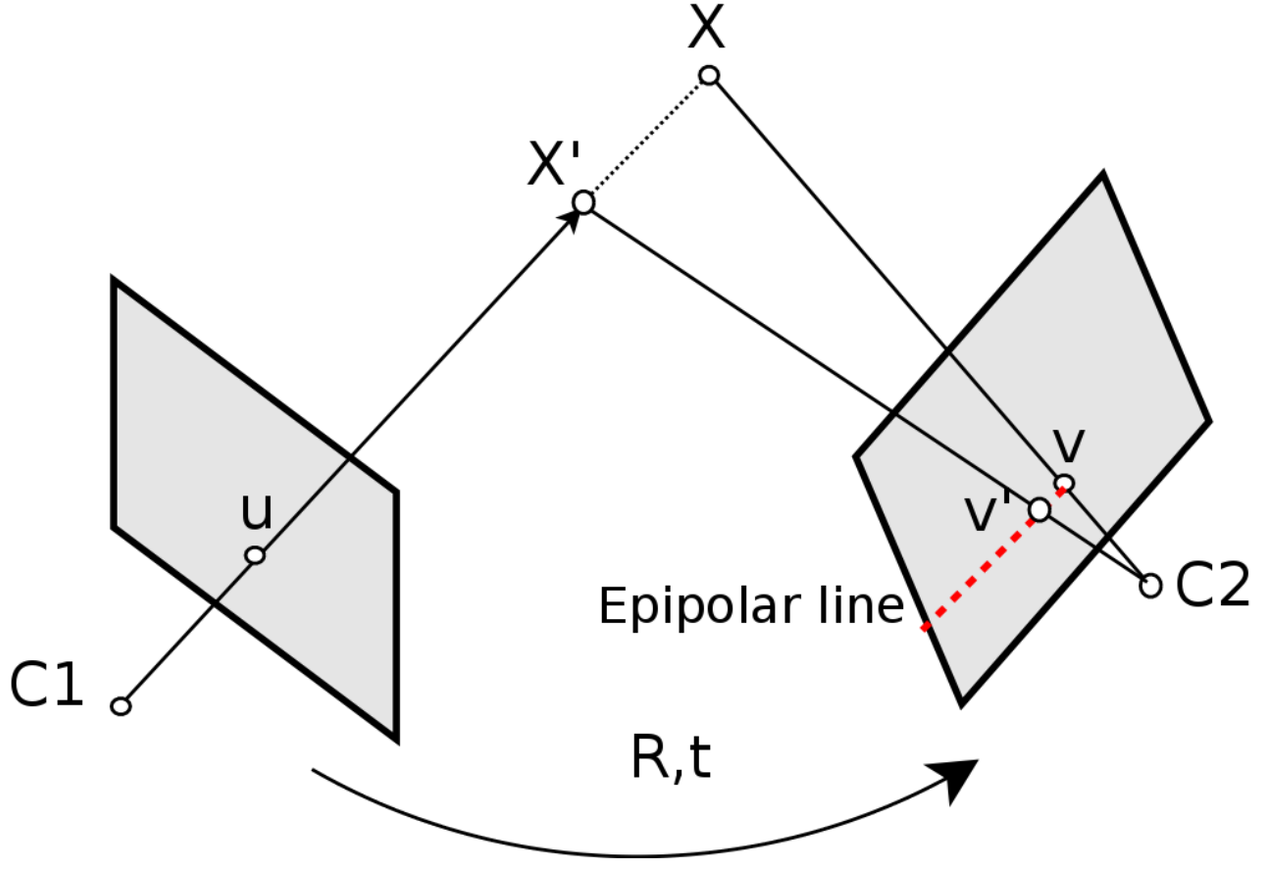
Compute essential matrix

- 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.



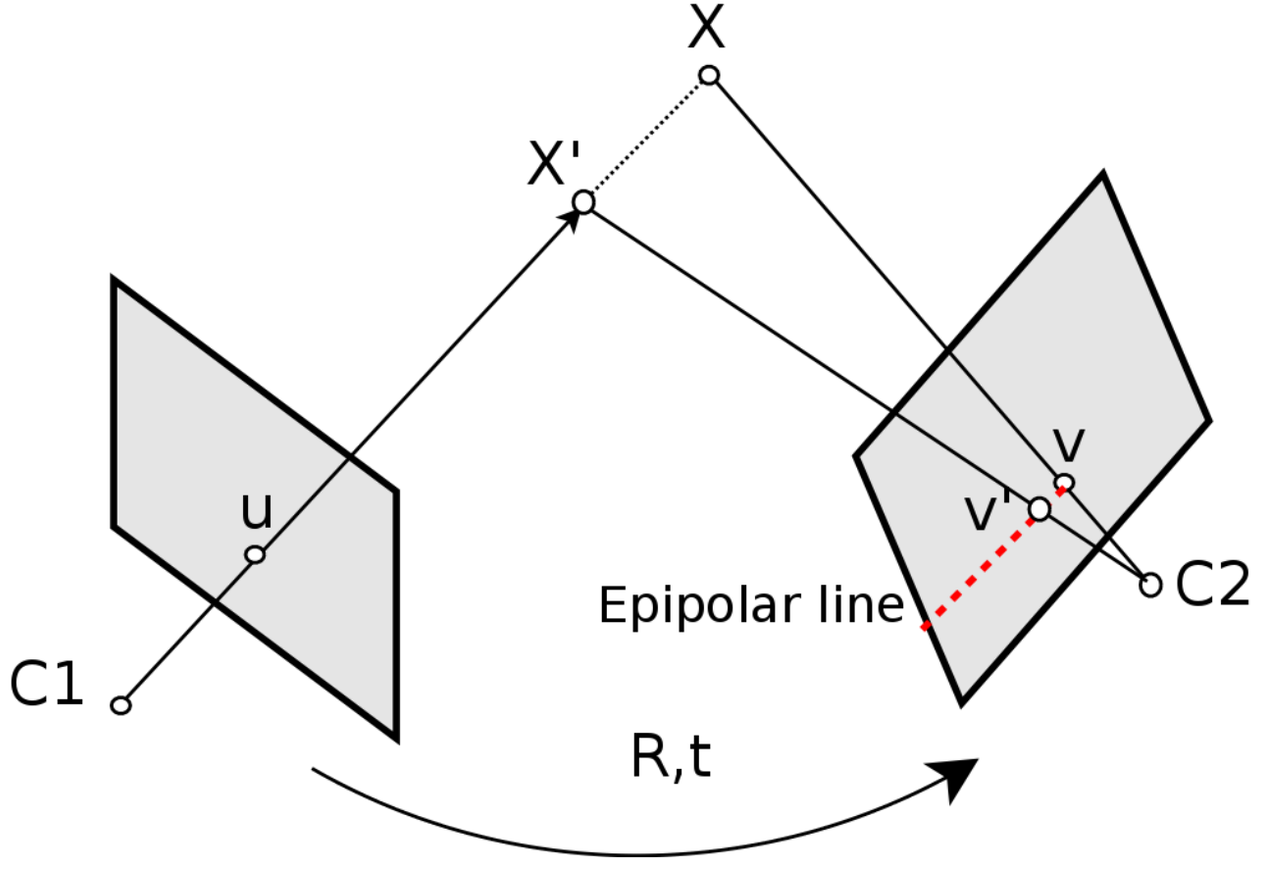
Compute essential matrix

- 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.



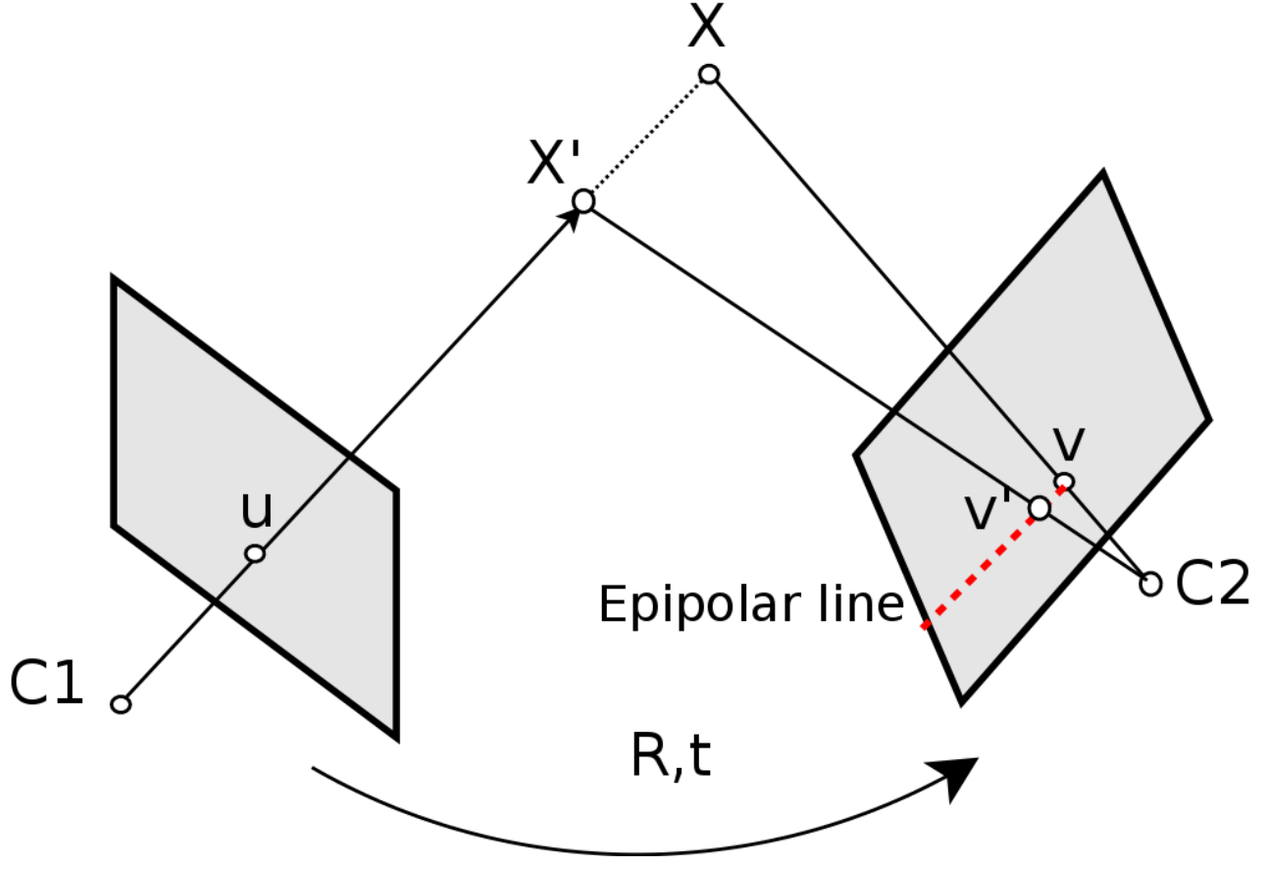
Compute essential matrix

- 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 $\{\mathbf{v} \mid \mathbf{u}^\top \mathbf{E}\, \mathbf{v} = 0\}$.



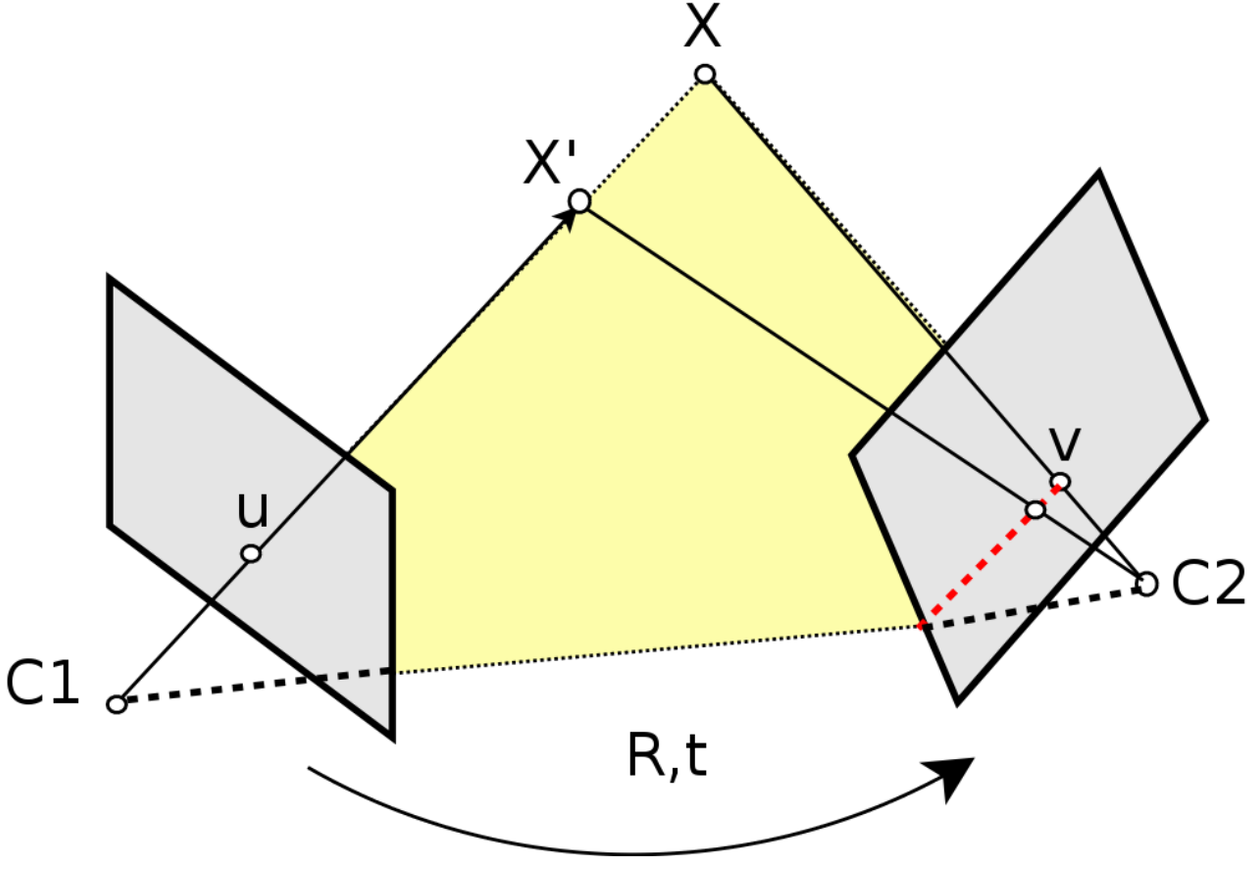
Compute essential matrix

- 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 $\{\mathbf{v} \mid \mathbf{u}^\top \mathbf{E}\, \mathbf{v} = 0\}$.



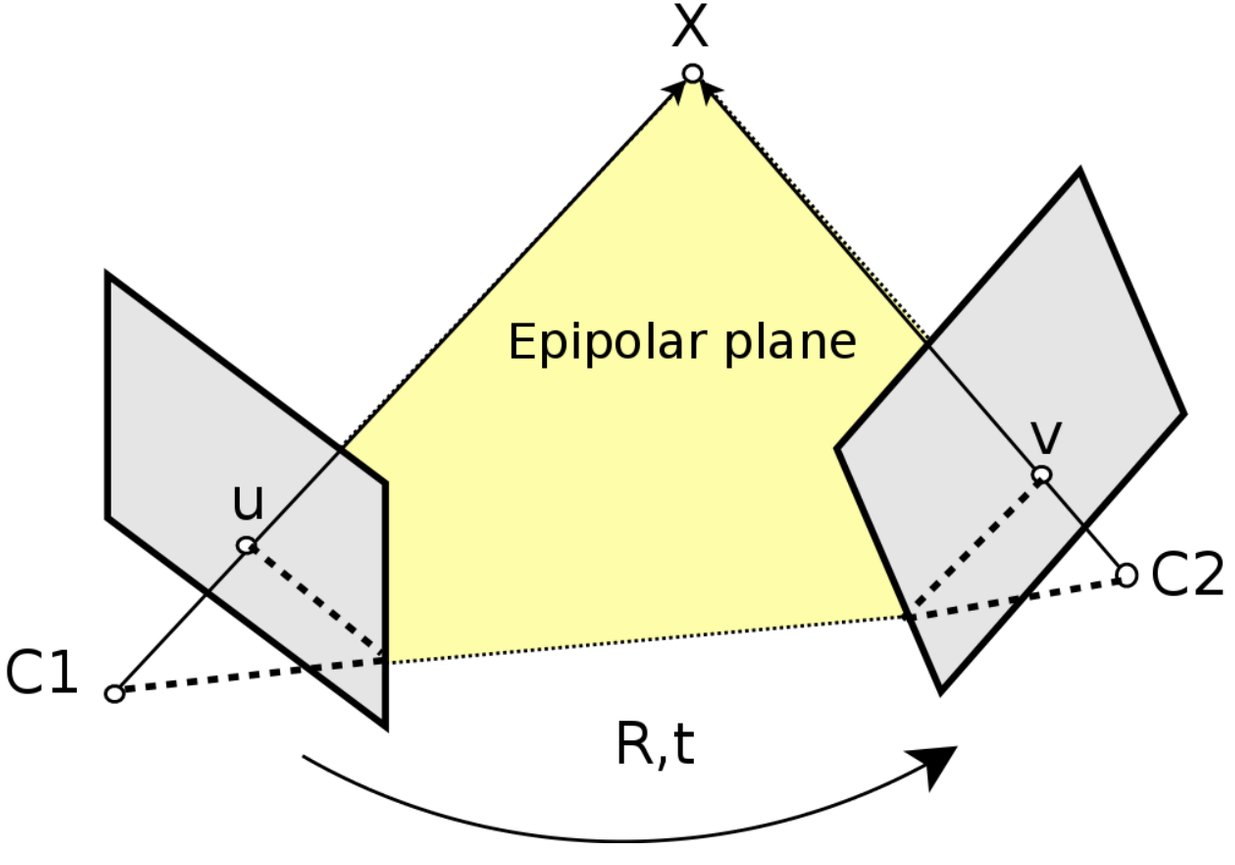
Compute essential matrix

- All possible correspondences lie on epipolar line $\{\mathbf{v} \mid \mathbf{u}^\top \mathbf{E}\,\mathbf{v} = 0\}$.
- Note:
  - This holds only if $\mathtt{K}$ is identity matrix (or $\mathbf{u}, \mathbf{v}$ are decalibrated).
  - Putting projection rays of all possible 3D points X together yiedls epipolar plane (yellow).



Compute essential matrix

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



Compute essential matrix

- Let us assume that we have several correspondences.

Compute essential matrix

- Let us assume that we have several correspondences.
- Essetial matrix $\mathtt{E}$ is just a solution of (overdetermined) homogeneous system of linear equations.

Compute essential matrix

- Let us assume that we have several correspondences.
- Essetial matrix $\mathbf{E}$ is just a solution of (overdetermined) homogeneous system of linear equations.
- For each correspondence pair $\mathbf{u}, \mathbf{v}$, the following holds:

$$\mathbf{u}^\top \mathbf{E}\, \mathbf{v} = \mathbf{u}^\top \begin{bmatrix} \mathbf{e}_1^\top \\ \mathbf{e}_2^\top \\ \mathbf{e}_3^\top \end{bmatrix} \mathbf{v} = \mathbf{u}^\top \begin{bmatrix} \mathbf{e}_1^\top \mathbf{v} \\ \mathbf{e}_2^\top \mathbf{v} \\ \mathbf{e}_3^\top \mathbf{v} \end{bmatrix} = [u_1 \mathbf{e}_1^\top \mathbf{v} + u_2 \mathbf{e}_2^\top \mathbf{v} + u_3 \mathbf{e}_3^\top \mathbf{v}] =$$

Compute essential matrix

- Let us assume that we have several correspondences.
- Essetial matrix $\mathbf{E}$ is just a solution of (overdetermined) homogeneous system of linear equations.
- For each correspondence pair $\mathbf{u}, \mathbf{v}$, the following holds:

$$\mathbf{u}^\top \mathbf{E}\, \mathbf{v} = \mathbf{u}^\top \begin{bmatrix} \mathbf{e}_1^\top \\ \mathbf{e}_2^\top \\ \mathbf{e}_3^\top \end{bmatrix} \mathbf{v} = \mathbf{u}^\top \begin{bmatrix} \mathbf{e}_1^\top \mathbf{v} \\ \mathbf{e}_2^\top \mathbf{v} \\ \mathbf{e}_3^\top \mathbf{v} \end{bmatrix} = [u_1 \mathbf{e}_1^\top \mathbf{v} + u_2 \mathbf{e}_2^\top \mathbf{v} + u_3 \mathbf{e}_3^\top \mathbf{v}] =$$

$$= [u_1 \mathbf{v}^\top \, u_2 \mathbf{v}^\top \, u_3 \mathbf{v}^\top] \begin{bmatrix} \mathbf{e}_1 \\ \mathbf{e}_2 \\ \mathbf{e}_3 \end{bmatrix} = 0$$

- It must hold for all correspondece pairs $\mathbf{u}_i$, $\mathbf{v}_i$, therefore:

$$\begin{bmatrix} u_{11}\mathbf{v}_1^\top & u_{12}\mathbf{v}_1^\top & u_{13}\mathbf{v}_1^\top \\ u_{21}\mathbf{v}_2^\top & u_{22}\mathbf{v}_2^\top & u_{23}\mathbf{v}_2^\top \\ & \vdots\vdots & \end{bmatrix} \begin{bmatrix} \mathbf{e}_1 \\ \mathbf{e}_2 \\ \mathbf{e}_3 \end{bmatrix} = \mathbf{0}$$

Compute essential matrix

- It is just homogeneous set of linear equations:

$$\underbrace{\begin{bmatrix} u_{11}\mathbf{v}_1^\top & u_{12}\mathbf{v}_1^\top & u_{13}\mathbf{v}_1^\top \\ u_{21}\mathbf{v}_2^\top & u_{22}\mathbf{v}_2^\top & u_{23}\mathbf{v}_2^\top \\ \vdots & & \end{bmatrix}}_{\mathtt{A}} \underbrace{\begin{bmatrix} \mathbf{e}_1 \\ \mathbf{e}_2 \\ \mathbf{e}_3 \end{bmatrix}}_{\mathbf{e}} = \mathbf{0}$$

- Again we want to avoid trivial solution $\mathbf{e}_1 = \mathbf{e}_2 = \mathbf{e}_3 = \mathbf{0}$.
- We solve the following optimization task (constrained LSQ)

$$\arg\min_{\mathbf{e}} \|\mathtt{A}\mathbf{e}\| \;\; \text{subject to} \;\; \|\mathbf{e}\| = 1$$

- the solution is singular vector of matrix $\mathtt{A}$ corresponding to the smallest singular value (can be found via SVD or eigenvectors/eigenvalues of $\mathtt{A}\mathtt{A}^\top$)

Compute essential matrix

- Since L2 is sensitive to ouliers, 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

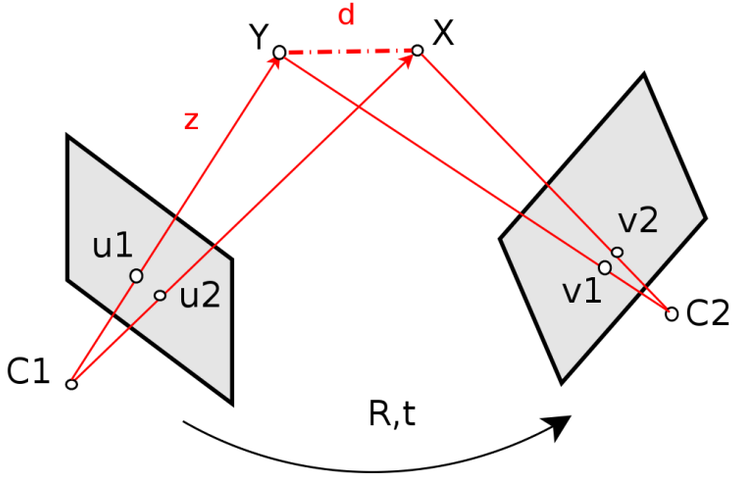
1. Get image I_k.
2. Compute correspondences between I_{k-1} and I_k (either feature matching or tracking).
3. Find correct correspondences and compute essential matrix $\mathtt{E}$.
4. Decompose $\mathtt{E}$ into $\mathtt{R}_k$ and $\mathbf{t}_k$.
5. Compute 3D model (points X).
6. Rescale t_k according to relative scale r.
7. $k = k + 1$

Decompose $\mathtt{E}$ into $\mathtt{R}$ and $\mathtt{t}$

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

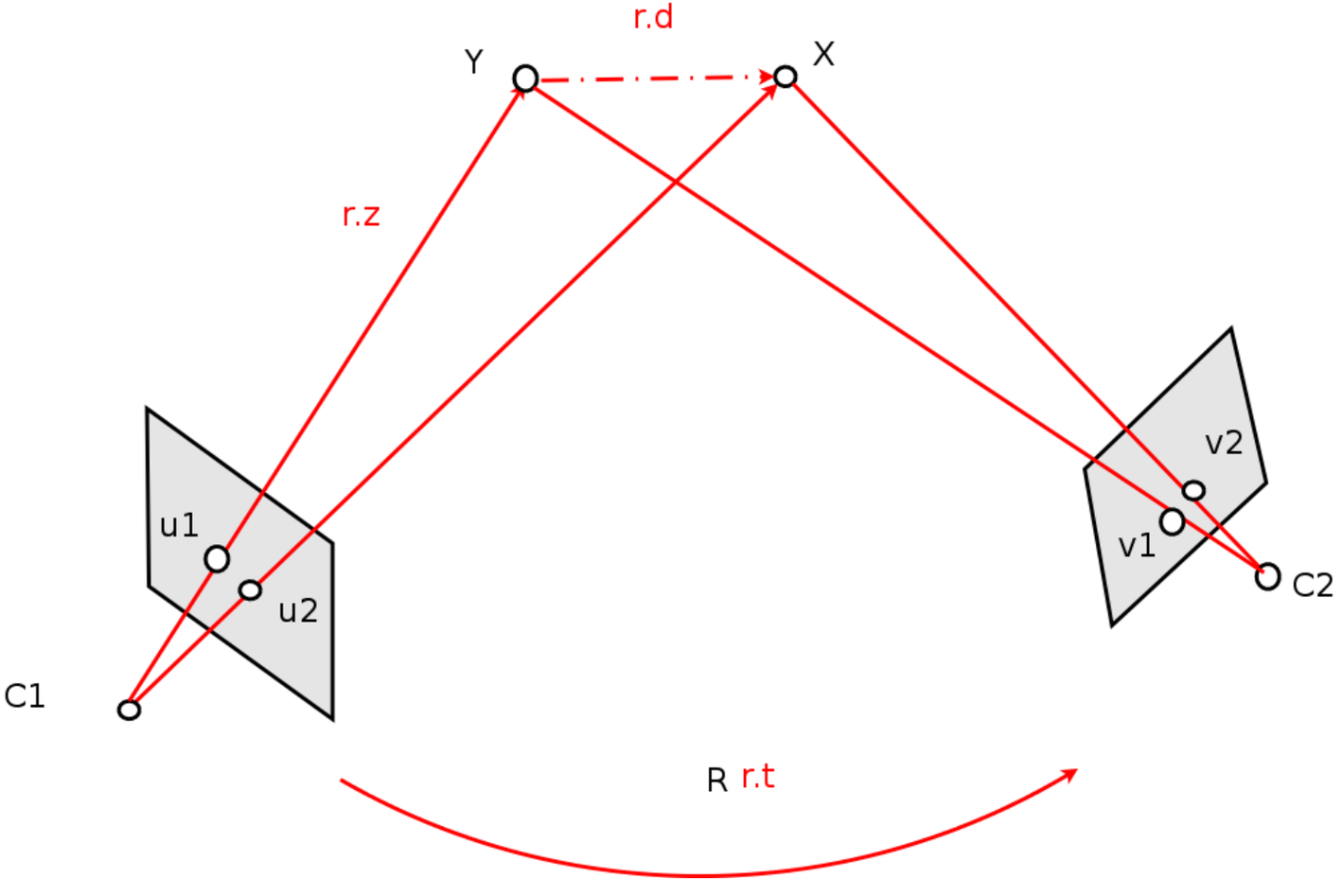
Decompose $\mathtt{E}$ into $\mathtt{R}$ and $\mathtt{t}$

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



Estimating camera motion

- Once you find $\mathbf{E}$, you can estimate camera motion by decomposing $\mathbf{E} = [\mathbf{t}]_\times \mathbf{R}$ via
- SVD ($\mathbf{E} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$) as follows: $[\mathbf{t}]_\times = \mathbf{V}\mathbf{W}\mathbf{\Sigma}\mathbf{V}^\top$, $\mathbf{R} = \mathbf{U}\mathbf{W}^{-1}\mathbf{V}^\top$, **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 $\mathtt{E}$, you can estimate camera motion by decomposing $\mathtt{E} = [\mathbf{t}]_\times \mathtt{R}$ via
- SVD ($\mathtt{E} = \mathtt{U}\Sigma\mathtt{V}^\top$) as follows: $[\mathbf{t}]_\times = \mathtt{V}\mathtt{W}\Sigma\mathtt{V}^\top$, $\mathtt{R} = \mathtt{U}\mathtt{W}^{-1}\mathtt{V}^\top$, **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 unknows ($\dim(\mathbf{e}) = 9$ minus scale) $\Rightarrow$ at least 8 correspondences needed $\Rightarrow$ 8-point algorithm.

Estimating camera motion

- Once you find $\mathtt{E}$, you can estimate camera motion by decomposing $\mathtt{E} = [\mathbf{t}]_\times \mathtt{R}$ via
- SVD ($\mathtt{E} = \mathtt{U}\Sigma\mathtt{V}^\top$) as follows: $[\mathbf{t}]_\times = \mathtt{V}\mathtt{W}\Sigma\mathtt{V}^\top$, $\mathtt{R} = \mathtt{U}\mathtt{W}^{-1}\mathtt{V}^\top$, **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 unknows ($\dim(\mathbf{e}) = 9$ 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 $\mathtt{E}$, you can estimate camera motion by decomposing $\mathtt{E} = [\mathbf{t}]_\times \mathtt{R}$ via
- SVD ($\mathtt{E} = \mathtt{U}\Sigma\mathtt{V}^\top$) as follows: $[\mathbf{t}]_\times = \mathtt{V}\mathtt{W}\Sigma\mathtt{V}^\top$, $\mathtt{R} = \mathtt{U}\mathtt{W}^{-1}\mathtt{V}^\top$, **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 unknows ($\dim(\mathbf{e}) = 9$ 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 $\mathtt{E}$, you can estimate camera motion by decomposing $\mathtt{E} = [\mathbf{t}]_\times \mathtt{R}$ via
- SVD ($\mathtt{E} = \mathtt{U}\Sigma\mathtt{V}^\top$) as follows: $[\mathbf{t}]_\times = \mathtt{V}\mathtt{W}\Sigma\mathtt{V}^\top$, $\mathtt{R} = \mathtt{U}\mathtt{W}^{-1}\mathtt{V}^\top$, **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 unknows ($\dim(\mathbf{e}) = 9$ 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).

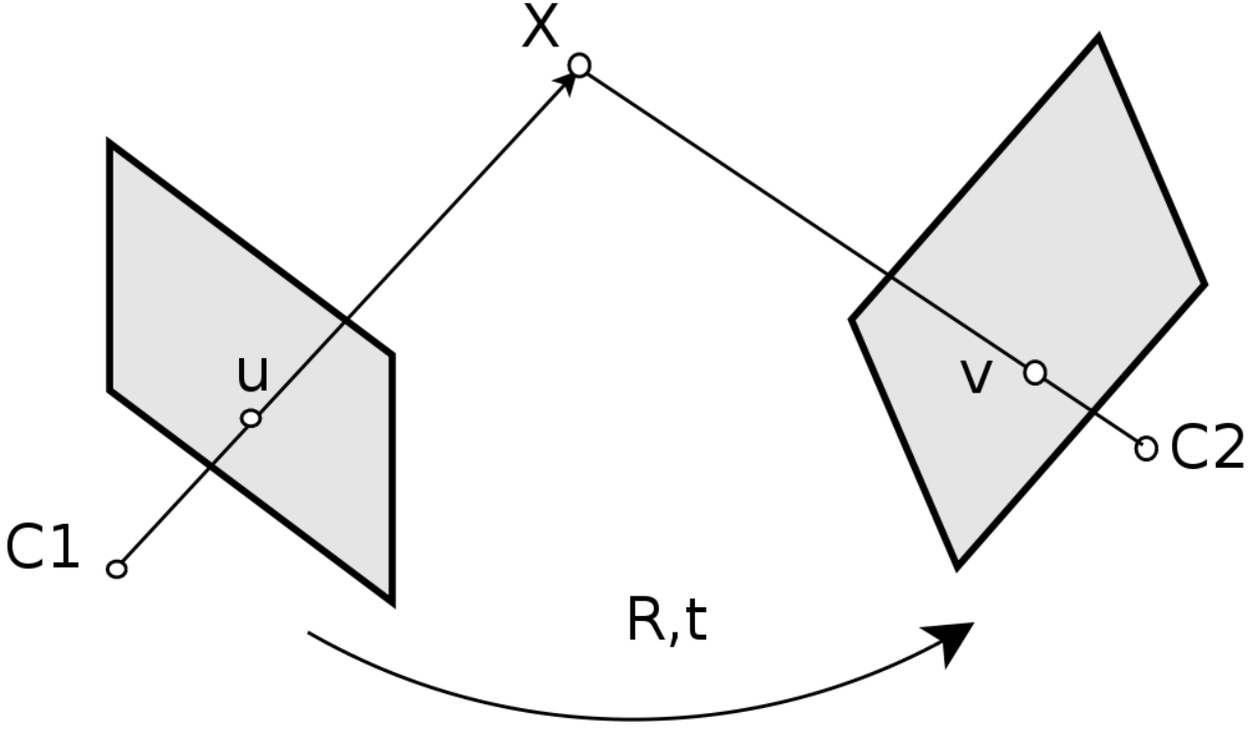
Algorithm at glance

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 $\mathtt{E}$.
4. Decompose $\mathtt{E}$ into $\mathtt{R}_k$ and $\mathbf{t}_k$.
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 $\mathbf{u} = [u_1\ u_2]^\top$ and $\mathbf{v} = [v_1\ v_2]^\top$ are projections of X in P and Q.
- Then

$$u_1 = \frac{\mathbf{p}_1^\top \mathbf{X}}{\mathbf{p}_3^\top \mathbf{X}} \quad \Rightarrow \quad u_1 \mathbf{p}_3^\top \mathbf{X} - \mathbf{p}_1^\top \mathbf{X} = 0$$

- and similarly ...

$$u_2 = \frac{\mathbf{p}_2^\top \mathbf{X}}{\mathbf{p}_3^\top \mathbf{X}} \quad \Rightarrow \quad u_2 \mathbf{p}_3^\top \mathbf{X} - \mathbf{p}_2^\top \mathbf{X} = 0$$

$$v_1 = \frac{\mathbf{q}_1^\top \mathbf{X}}{\mathbf{q}_3^\top \mathbf{X}} \quad \Rightarrow \quad v_1 \mathbf{q}_3^\top \mathbf{X} - \mathbf{q}_1^\top \mathbf{X} = 0$$

$$v_2 = \frac{\mathbf{q}_2^\top \mathbf{X}}{\mathbf{q}_3^\top \mathbf{X}} \quad \Rightarrow \quad v_2 \mathbf{q}_3^\top \mathbf{X} - \mathbf{q}_2^\top \mathbf{X} = 0$$

Compute 3D model

- Which is 4×4 homogeneous system of linear equations:

$$\begin{bmatrix} u_1\mathbf{p}_3^\top - \mathbf{p}_1^\top \\ u_2\mathbf{p}_3^\top - \mathbf{p}_2^\top \\ v_1\mathbf{q}_3^\top - \mathbf{q}_1^\top \\ v_2\mathbf{q}_3^\top - \mathbf{q}_2^\top \end{bmatrix} \mathbf{X} = \mathbf{0}$$

Compute 3D model

- Which is 4×4 homogeneous system of linear equations:

$$\begin{bmatrix} u_1\mathbf{p}_3^\top - \mathbf{p}_1^\top \\ u_2\mathbf{p}_3^\top - \mathbf{p}_2^\top \\ v_1\mathbf{q}_3^\top - \mathbf{q}_1^\top \\ v_2\mathbf{q}_3^\top - \mathbf{q}_2^\top \end{bmatrix} \mathbf{X} = \mathbf{0}$$

- 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:

$$\arg\min_{\mathrm{P}^i, \mathbf{X}_j} \sum_{i,j} d(\mathrm{P}^i X_j, \mathbf{u}_j^i)^2$$

- It is often called the bundle adjustment.

Algorithm at glance

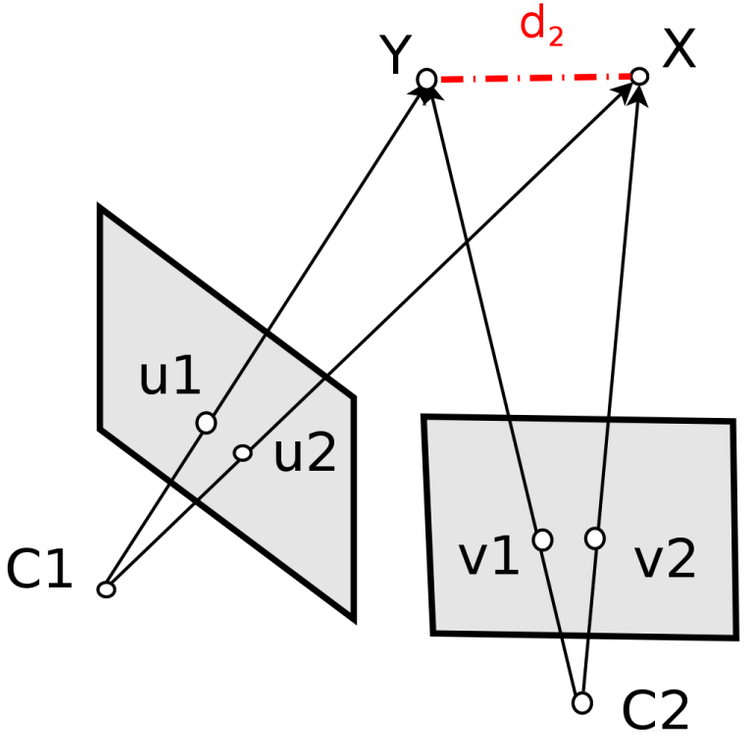
1. Get image I_k.
2. Compute correspondences between I_{k-1} and I_k (either feature matching or tracking).
3. Find correct correspondences and compute essential matrix $\mathtt{E}$.
4. Decompose $\mathtt{E}$ into $\mathtt{R}_k$ and $\mathbf{t}_k$.
5. Compute 3D model (points X).
6. Rescale t_k according to relative scale r.
7. $k = k + 1$

Estimating camera motion - relative scale

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

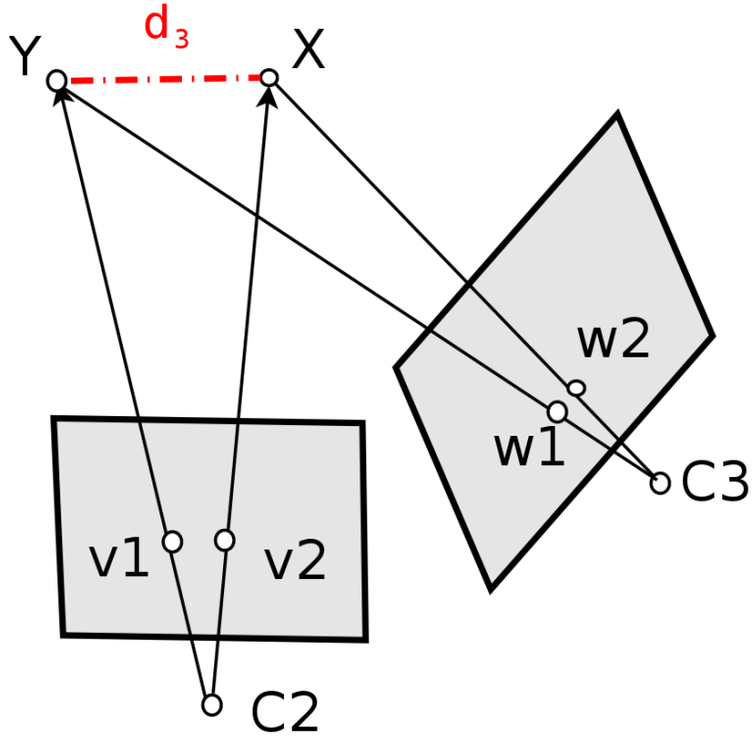
Estimating camera motion - relative scale

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



Estimating camera motion - relative scale

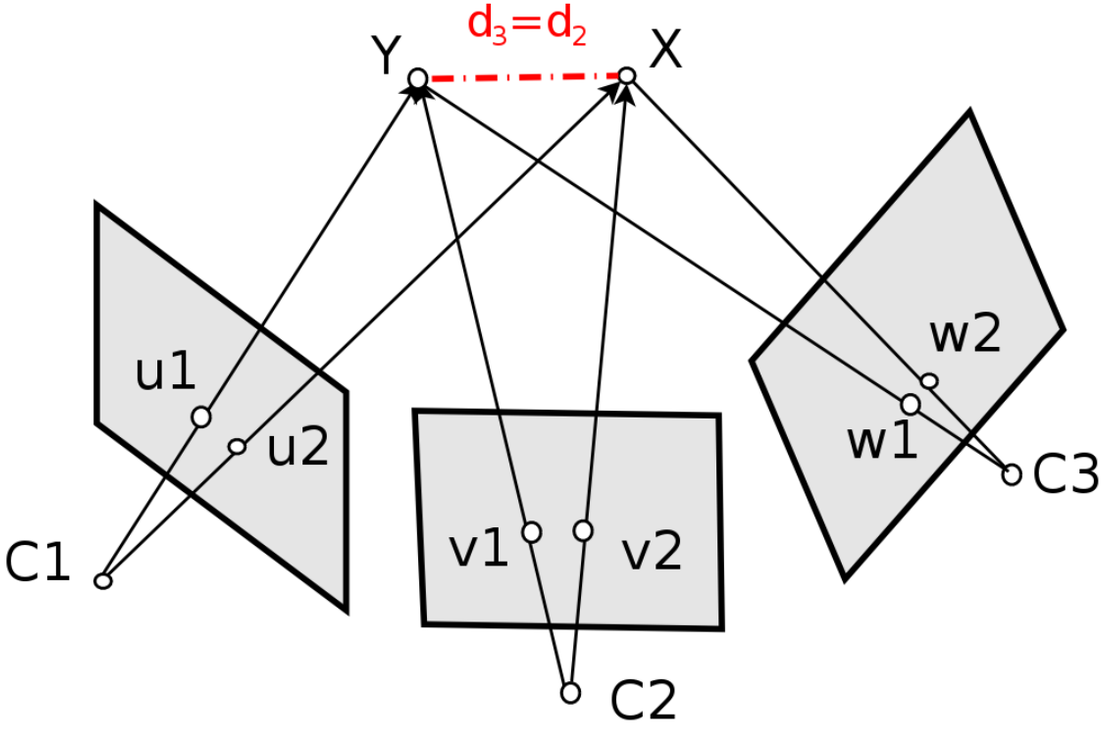
1. You cannot get absolute scale (without calibration object).
2. If you estimate motion (and 3D model) from C_1, C_2 and than from C_2, C_3 you can have completely different scale.



Estimating camera motion - relative scale

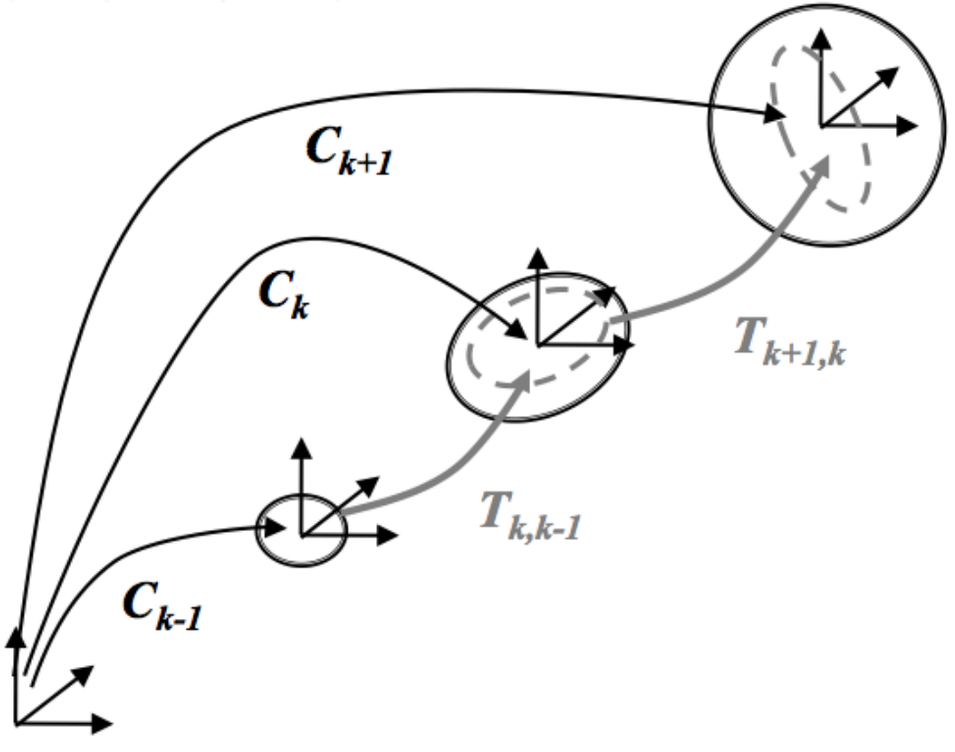
1. You cannot get absolute scale (without calibration object).
2. If you estimate motion (and 3D model) from C_1, C_2 and than from C_2, C_3 you can have completely different scale.
3. You want to keep the same relative scale r by rescaling $\mathbf{t}$ (and 3D)

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



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)

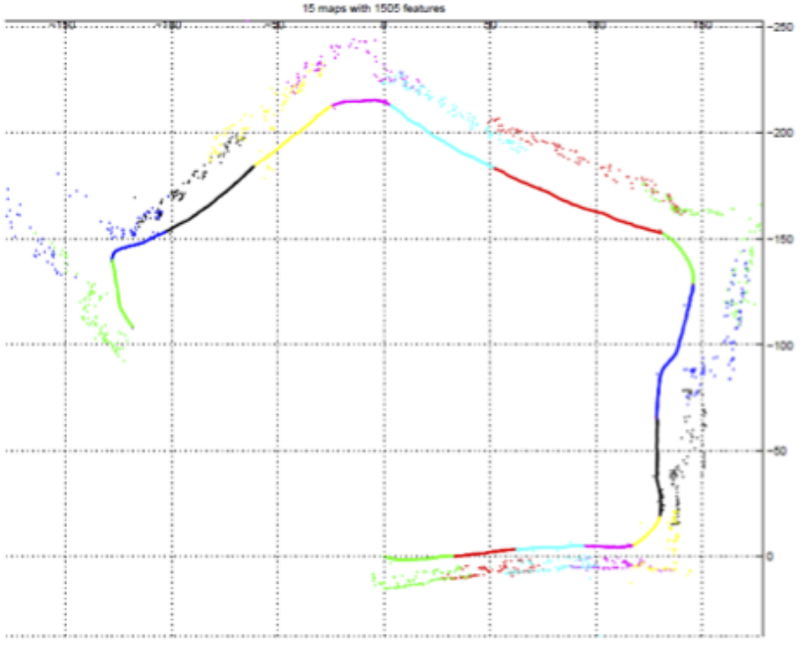
- SFM (3D from unordered set of images) show video `sfm_colloseum.avi`

Visual Odometry (VO), Structure from Motion (SFM), Visual Simultaneous Localisation and Mapping (VSLAM)

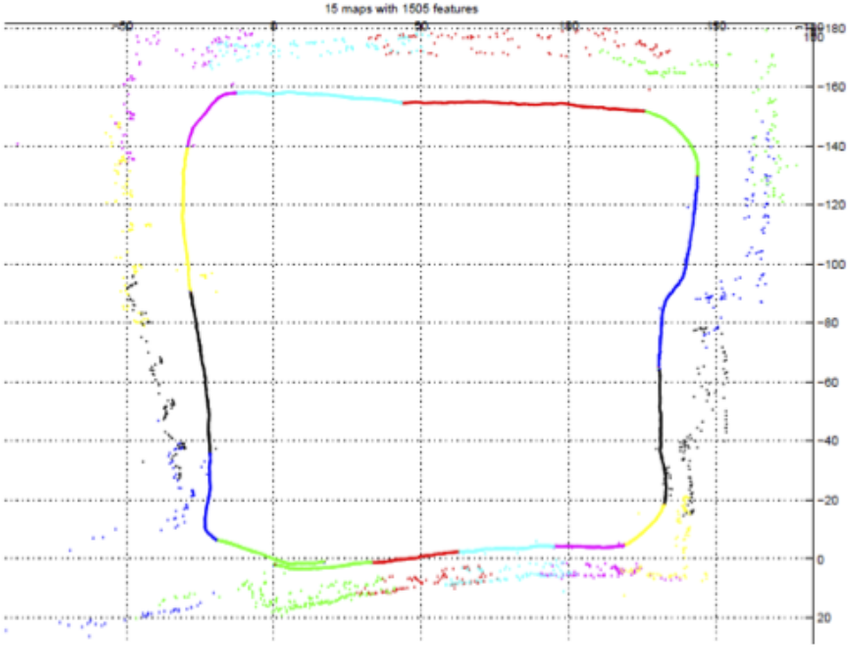
- SFM (3D from unordered set of images) show video `sfm_colloseum.avi`
- VO sequential and real-time camera motion estimation.

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