Augmented Reality

3A SN M

Simone Gasparini

simone.gasparini@enseeiht.fr

Previously…

- Tracking using markers
  - Detect the marker in the image
  - Use the 4 points to estimate the camera pose

- **Tracking by detection** method
  - No history from frame to frame
  - Each frame processed independently
- Some information can be passed to the next frame
  - The pose of the camera as initial guess (small movement)
  - The point position as initial guess
- Can stabilize the tracking
  - Eg deal with (partial) occlusions

Previously…

Another approach

- **Detect and track**
  - Detect once
  - Use the information of the previous frame to find the new position of the points
  - Estimate the pose
  - If no luck, try detection again
- Generalization of the previous problem
  - Track the features (not just markers) over images
  - Estimate the pose using tracked features
- SLAM approach

Previously… (Marker/fiducial approach)

Pros

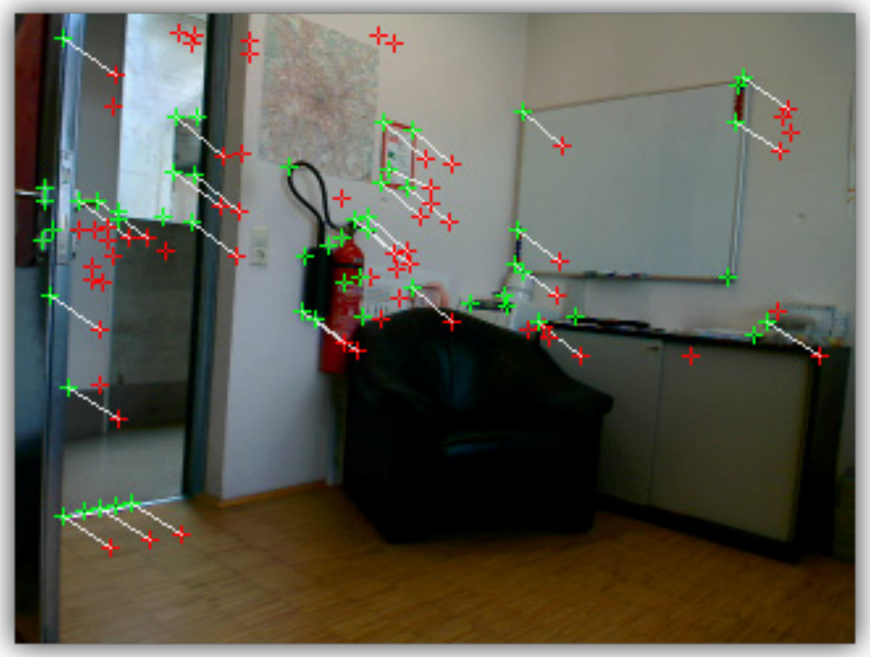
- Easy setup and low cost
  - Even for mobile devices
- Works well with
  - Large movement of the camera
  - Dynamic environments (as long as the marker is visible)
  - Texture-less scenes
- Encodes additional info
- Always correct scale for 3D object

Cons

- Requires setup
- Markers must be always (at least partially) visible
- May require marker digital removal

Feature-based camera tracking

- More general method
- Instead of markers we can track features
  - Harris, SIFT, SURF…
- Tracking feature using KLT
- Estimate the pose
  - Relax the coplanar point method
  - More general approach



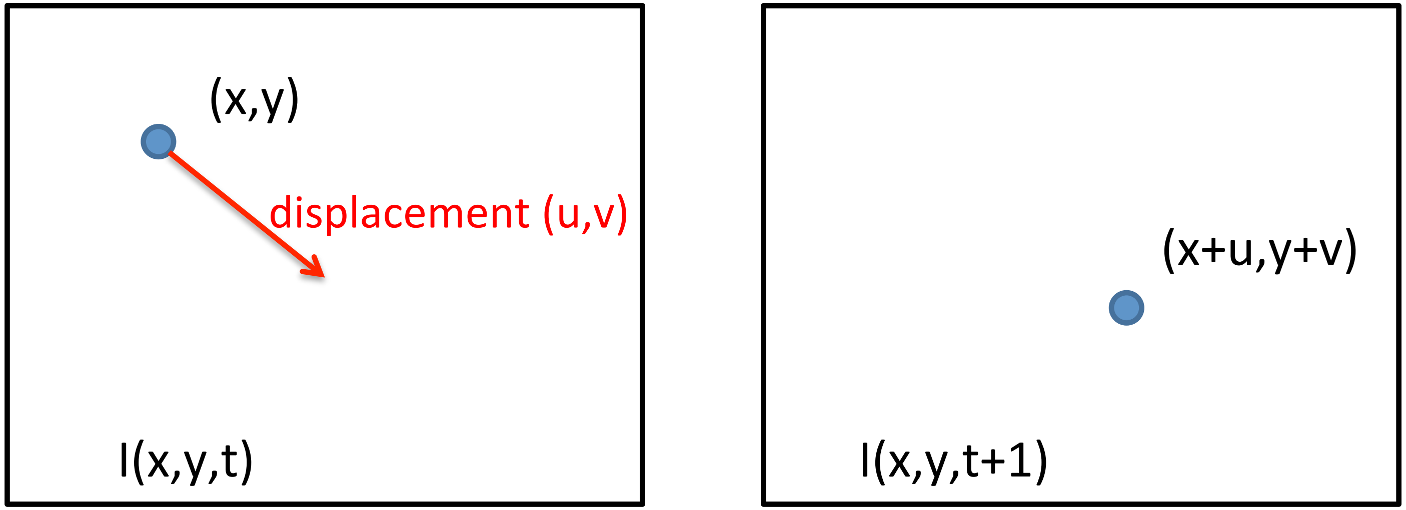
Kanade-Lucas-Tomasi (KLT) Tracker

- Find a good point to track (harris corner)
- Find displacement by solving the optical flow equation in a window around the point
- Get the new position of the point

- Window size
  - Small window more sensitive to noise and may miss larger motions
  - Large window more likely to cross an occlusion boundary (and it's slower)
  - Typically 15x15 to 31x31

- OpenCV has it implemented in `calcOpticalFlowPyrLK()`

Feature Tracking



- Given two consecutive frames and a point on the first image, estimate the point translation on the second image

Optical flow

- **Optical flow:** apparent motion of pixels due to the relative motion between the camera and objects in the scene

Main assumptions

- **Small motion**: points do not move very far
- **Brightness constancy**: projection of the same point looks the same in every frame

Optical flow

- Assume the image brightness is a continuous and differentiable function f
  - The discrete formulation is straightforward
- x, y are the coordinates of the points inside the image
- t is time

$$f(x, y, t)$$ Brightness at position x,y at time t

Optical flow

- Assume the image brightness is a continuous and differentiable function f
- x, y are the coordinates of the points inside the image
- t is time

$$f(x, y, t)$$ Brightness at position x,y at time t

Brightness Constancy Equation for dx, dy and dt

$$f(x, y, t) = f(x + dx, y + dy, t + dt)$$

Optical flow

- Assume the image brightness is a continuous and differentiable function f
- x, y are the coordinates of the points inside the image
- t is time

$$f(x, y, t)$$ Brightness at position x,y at time t

Brightness Constancy Equation for dx, dy and dt

$$f(x, y, t) = \boxed{f(x + dx, y + dy, t + dt)}$$

1st order Taylor Series

$$f(x + dx, y + dy, t + dt) = f(x, y, t) + dx\frac{\partial f}{\partial x} + dy\frac{\partial f}{\partial y} + dt\frac{\partial f}{\partial t}$$

Optical flow

$$f(x+dx, y+dy, t+dt) = f(x,y,t) + dx\boxed{\frac{\partial f}{\partial x}} + dy\frac{\partial f}{\partial y} + dt\frac{\partial f}{\partial t}$$

$\frac{\partial f}{\partial x}$ The brightness variation along x (known)
=> image gradient on x

Optical flow

$$f(x+dx, y+dy, t+dt) = f(x,y,t) + dx\frac{\partial f}{\partial x} + dy\boxed{\frac{\partial f}{\partial y}} + dt\frac{\partial f}{\partial t}$$

$\frac{\partial f}{\partial x}$ The brightness variation along x (known)
=> image gradient on x

$\frac{\partial f}{\partial y}$ The brightness variation along y (known)
=> image gradient on y

Optical flow

$$f(x+dx, y+dy, t+dt) = f(x,y,t) + dx\frac{\partial f}{\partial x} + dy\frac{\partial f}{\partial y} + dt\boxed{\frac{\partial f}{\partial t}}$$

$\frac{\partial f}{\partial x}$ The brightness variation along x (known)
=> image gradient on x

$\frac{\partial f}{\partial y}$ The brightness variation along y (known)
=> image gradient on y

$\frac{\partial f}{\partial t}$ The brightness variation between the two frames (known)

Optical flow

Brightness Constancy Equation for dx, dy and dt

$$f(x,y,t) = \boxed{f(x+dx, y+dy, t+dt)}$$

1st order Taylor Series

$$f(x+dx, y+dy, t+dt) = f(x,y,t) + dx\frac{\partial f}{\partial x} + dy\frac{\partial f}{\partial y} + dt\frac{\partial f}{\partial t}$$

Optical flow

Brightness Constancy Equation for *dx, dy* and *dt*

(1) $$f(x,y,t) = \boxed{f(x+dx, y+dy, t+dt)}$$

1st order Taylor Series

(2) $$f(x+dx, y+dy, t+dt) = f(x,y,t) + dx\frac{\partial f}{\partial x} + dy\frac{\partial f}{\partial y} + dt\frac{\partial f}{\partial t}$$

Replace (2) in (1)

$$f(x,y,t) = f(x,y,t) + dx\frac{\partial f}{\partial x} + dy\frac{\partial f}{\partial y} + dt\frac{\partial f}{\partial t}$$

Optical flow

Brightness Constancy Equation for dx, dy and dt

$$f(x,y,t) = \boxed{f(x+dx, y+dy, t+dt)}$$

1st order Taylor Series

$$f(x+dx, y+dy, t+dt) = f(x,y,t) + dx\frac{\partial f}{\partial x} + dy\frac{\partial f}{\partial y} + dt\frac{\partial f}{\partial t}$$

$$f(x,\cancel{y},t) = f(x,\cancel{y},t) + dx\frac{\partial f}{\partial x} + dy\frac{\partial f}{\partial y} + dt\frac{\partial f}{\partial t}$$

$$dx\frac{\partial f}{\partial x} + dy\frac{\partial f}{\partial y} + dt\frac{\partial f}{\partial t} = 0$$

Optical flow

$$dx\frac{\partial f}{\partial x} + dy\frac{\partial f}{\partial y} + dt\frac{\partial f}{\partial t} = 0$$

Optical flow

$$dx\frac{\partial f}{\partial x} + dy\frac{\partial f}{\partial y} + dt\frac{\partial f}{\partial t} = 0$$

rewrite as...

$$f_x dx + f_y dy + f_t dt = 0$$

$$\left\{ \begin{aligned} f_x &= \frac{\partial f}{\partial x} \\ f_y &= \frac{\partial f}{\partial y} \\ f_t &= \frac{\partial f}{\partial t} \end{aligned} \right.$$

Optical flow

$$dx\frac{\partial f}{\partial x} + dy\frac{\partial f}{\partial y} + dt\frac{\partial f}{\partial t} = 0$$

rewrite as...

$$f_x dx + f_y dy + f_t dt = 0$$

$$\begin{cases} f_x = \frac{\partial f}{\partial x} \\ f_y = \frac{\partial f}{\partial y} \\ f_t = \frac{\partial f}{\partial t} \end{cases}$$

divide by dt

$$f_x u + f_y v + f_t = 0$$

where

$u = \frac{dx}{dt}$ Velocity along x (unknown)

$v = \frac{dy}{dt}$ Velocity along y (unknown)

Optical flow

Brightness Constancy Equation

$$\boxed{f_x u + f_y v + f_t = 0}$$

$$\left[\begin{array}{ll} u = \dfrac{dx}{dt} & \text{Velocity along x (unknown) => motion on x for } dt\text{=1} \\ v = \dfrac{dy}{dt} & \text{Velocity along y (unknown) => motion on y for } dt\text{=1} \end{array}\right.$$

$$\left[\begin{array}{ll} f_x = \dfrac{\partial f}{\partial x} & \text{The brightness variation along x (known) => image gradient on x} \\ f_y = \dfrac{\partial f}{\partial y} & \text{The brightness variation along y (known) => image gradient on y} \\ f_t = \dfrac{\partial f}{\partial t} & \text{The brightness variation between two frames (known)} \end{array}\right.$$

For each position we have 1 equation with 2 unknowns…

Optical Flow – Lucas-Kanade

Main assumptions

- **Brightness constancy**: projection of the same point looks the same in every frame
- **Small motion**: points do not move very far
- **Spatial coherence**: points move like their neighbours
  - assume that brightness constancy holds for a small neighbourhood (window) around the point
  - Use the neighbor pixels to solve the equation at least squares

Eg, considering a 3x3 window we get 9 equations

$$
\begin{aligned}
f_{x1}u + f_{y1}v &= -f_{t1} \\
f_{x2}u + f_{y2}v &= -f_{t2} \\
&\vdots \\
f_{x9}u + f_{y9}v &= -f_{t9}
\end{aligned}
$$

Optical Flow – Lucas-Kanade

$$f_{x1}u + f_{y1}v = -f_{t1}$$

$$f_{x2}u + f_{y2}v = -f_{t2}$$

$$\vdots$$

$$f_{x9}u + f_{y9}v = -f_{t9}$$

$$\mathbf{A} = \begin{bmatrix} f_{x1} & f_{y1} \\ f_{x2} & f_{y2} \\ \vdots & \vdots \\ f_{x9} & f_{y9} \end{bmatrix} \qquad \mathbf{B} = \begin{bmatrix} -f_{t1} \\ -f_{t2} \\ \vdots \\ f_{t9} \end{bmatrix} \qquad \mathbf{u} = \begin{bmatrix} u \\ v \end{bmatrix}$$

$$\mathbf{Au} = \mathbf{B}$$

Optical Flow – Lucas-Kanade

$$\mathbf{A}\mathbf{u} = \mathbf{B}$$

$$\mathbf{A} = \begin{bmatrix} f_{x1} & f_{y1} \\ f_{x2} & f_{y2} \\ \vdots & \vdots \\ f_{x9} & f_{y9} \end{bmatrix} \quad \mathbf{B} = \begin{bmatrix} -f_{t1} \\ -f_{t2} \\ \vdots \\ f_{t9} \end{bmatrix} \quad \mathbf{u} = \begin{bmatrix} u \\ v \end{bmatrix}$$

Solving with the pseudo-inverse $\left(\mathbf{A}^{\mathbf{T}}\mathbf{A}\right)^{-1}\mathbf{A}^{\mathbf{T}}$

$$\mathbf{A}^{\mathbf{T}}\mathbf{A}\mathbf{u} = \mathbf{A}^{\mathbf{T}}\mathbf{B}$$

$$\mathbf{u} = \left(\mathbf{A}^{\mathbf{T}}\mathbf{A}\right)^{-1}\mathbf{A}^{\mathbf{T}}\mathbf{B}$$

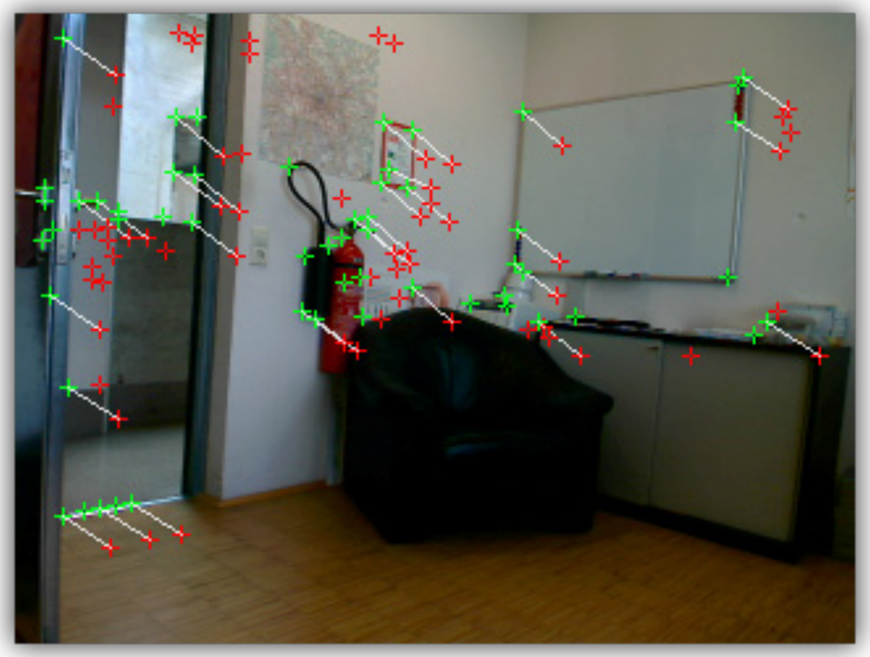
$\mathbf{A}^{\mathbf{T}}\mathbf{A}$ has to be invertible, hence rank(A)=2

Remember the Calibrated Photometric Stereo…

$\widehat{\mathtt{M}}(P)^{\top} = \mathbf{I}(Q)\,\mathtt{S}^{\top}\left(\mathtt{S}\,\mathtt{S}^{\top}\right)^{-1}$

Feature-based camera tracking

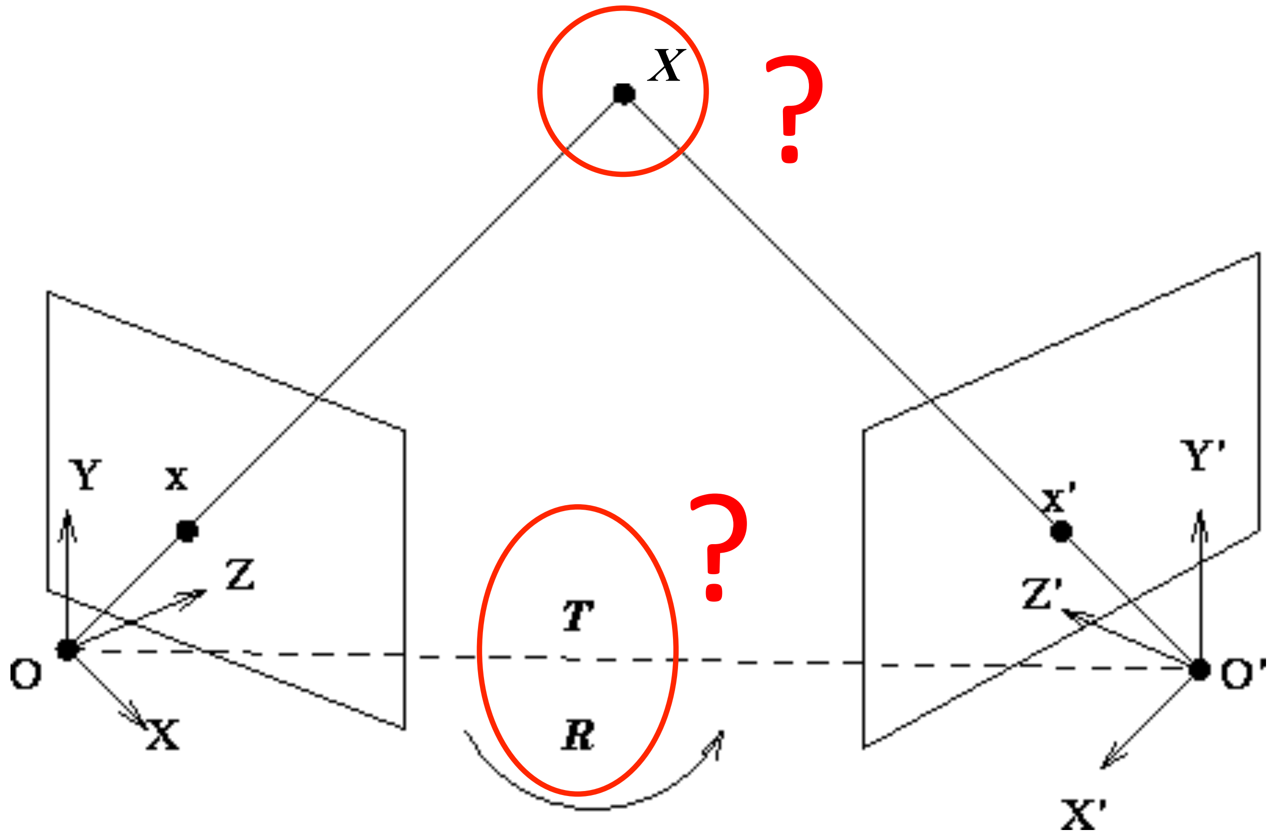
- More general method
- Instead of markers we can track features
  - Harris, SIFT, SURF…
- Tracking feature using KLT
- Estimate the pose
  - Relax the coplanar point method
  - More general approach



Camera tracking with features

Repeat:

- Track the features in the image
- Estimate the camera pose from 2D-3D correspondences



Camera tracking with features

Repeat:

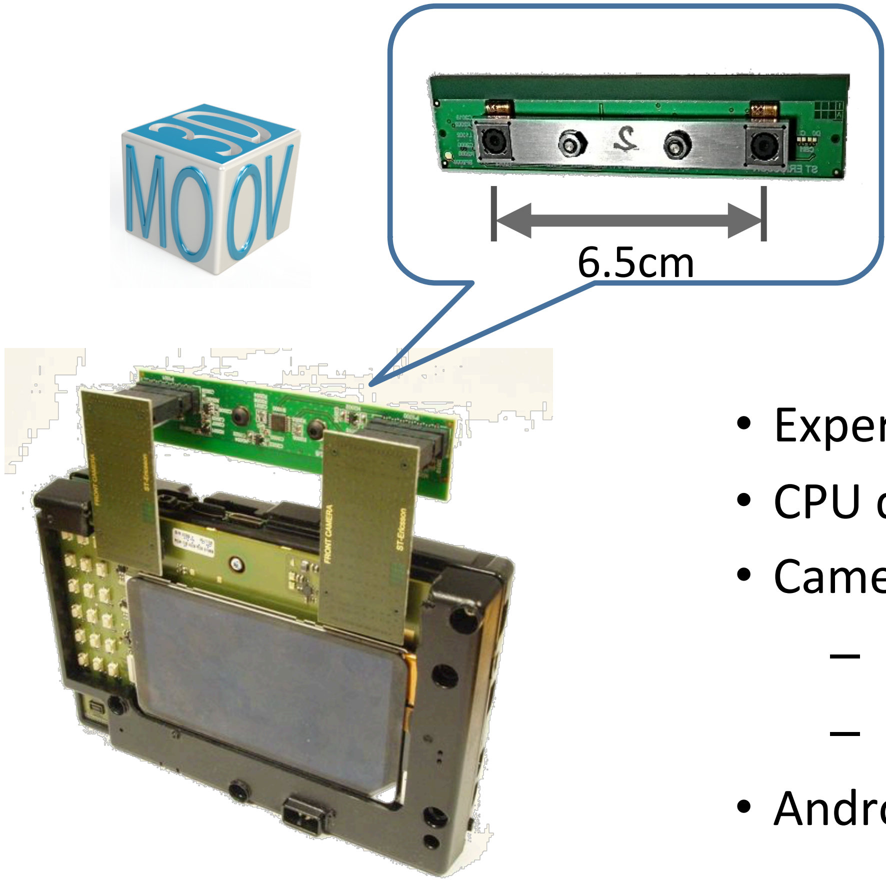
- Track the features in the image
- Estimate the camera pose from 2D-3D correspondences

- What about the 3D points?
- They are known if
  - we have some kind of 3D model (fiducials, makers…)
  - we can have the 3D information with the image (stereocamera, kinect…)
- General case of monocular camera?
  - Use 3D reconstruction

Camera tracking with features

Stereo camera

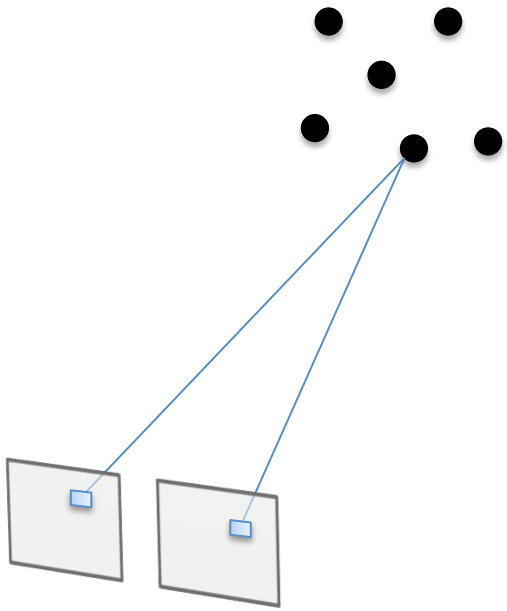
- 2D-3D associations are available at each frame



- Experimental mobile phone
- CPU dual core ARM® CORTEXTM-A9
- Cameras
  - 2x 5.3Mpixel (VGA mode)
  - Baseline 6.5cm
- Android 2.3.4

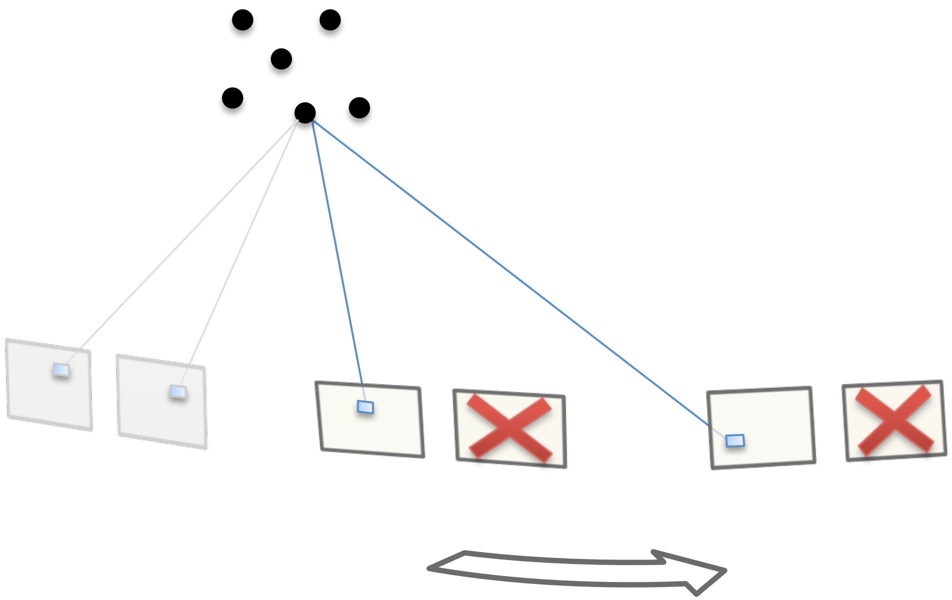
Using a stereo camera

- Use **stereo image** to initialize (3D reconstruction)



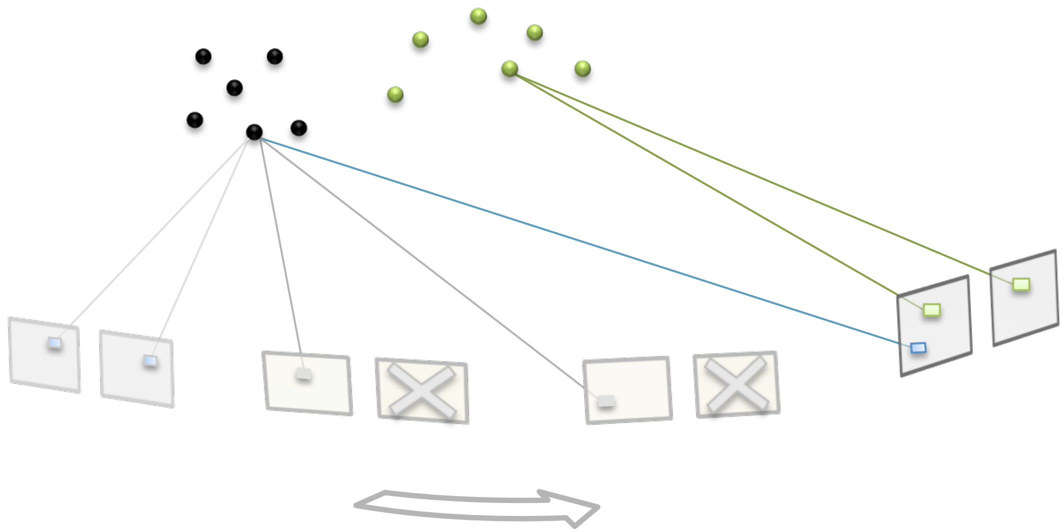
Using a stereo camera

- Use stereo image to initialize (3D reconstruction)
- Track the features only in one image



Using a stereo camera

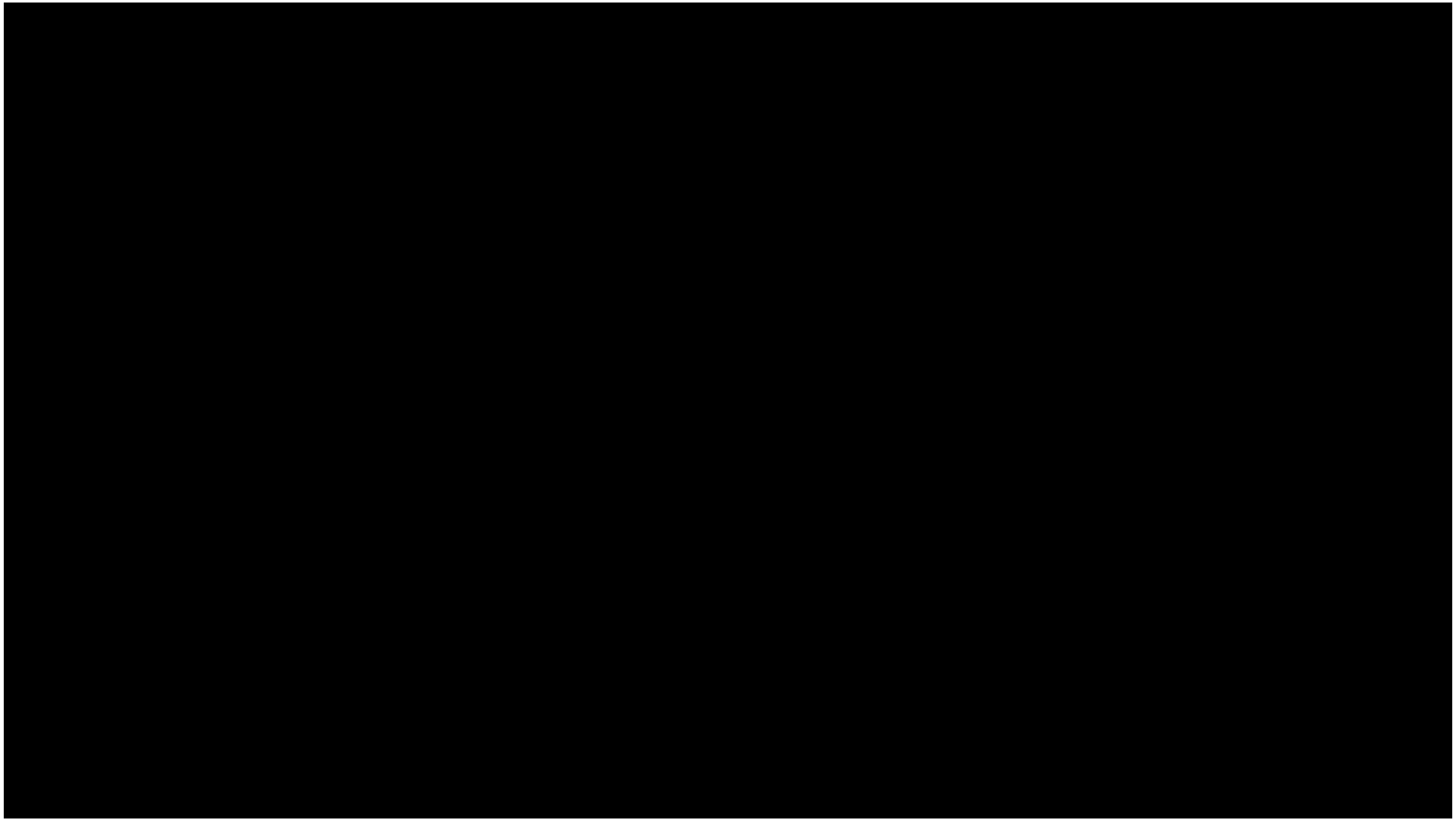
- Use stereo image to initialize (3D reconstruction)
- Track the features only in one image
- Use stereo image to add new features when needed



Using a stereo camera

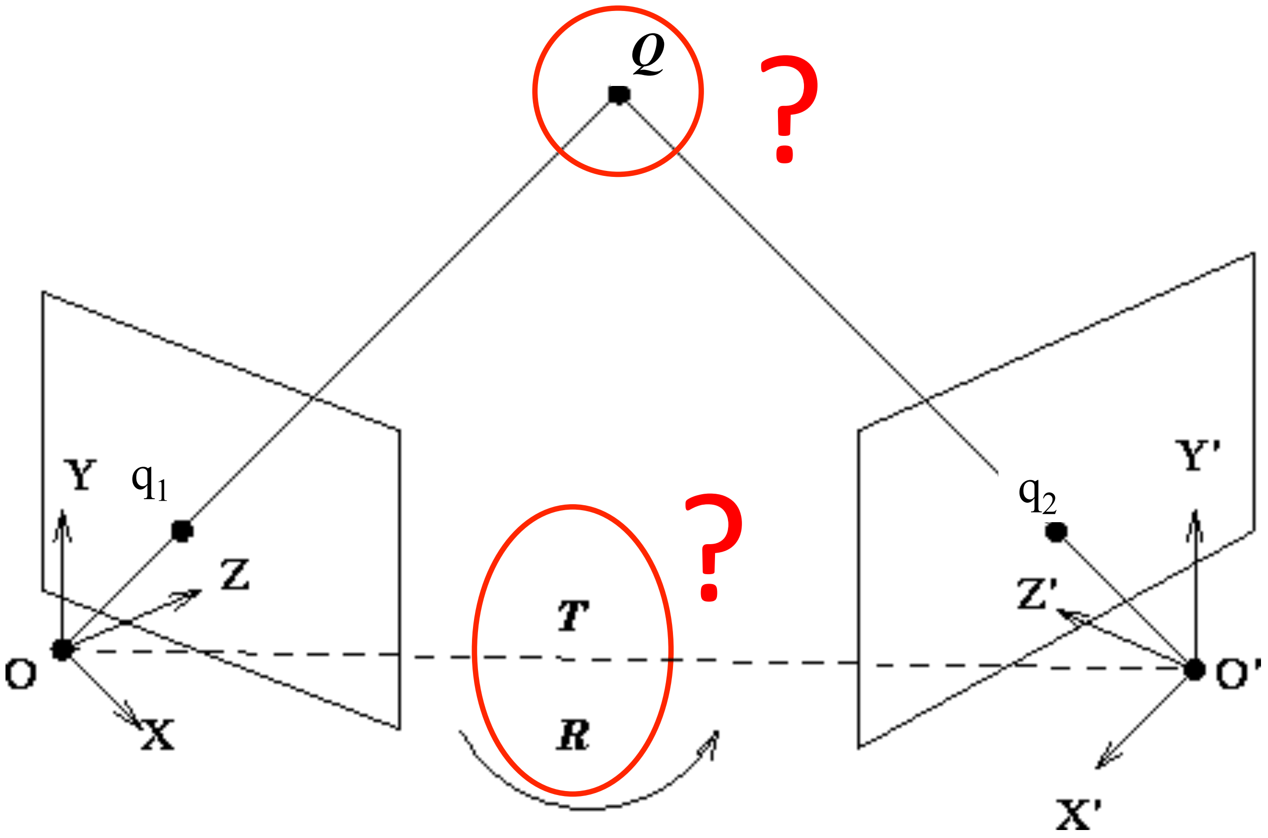
Using a stereo camera

Using a stereo camera

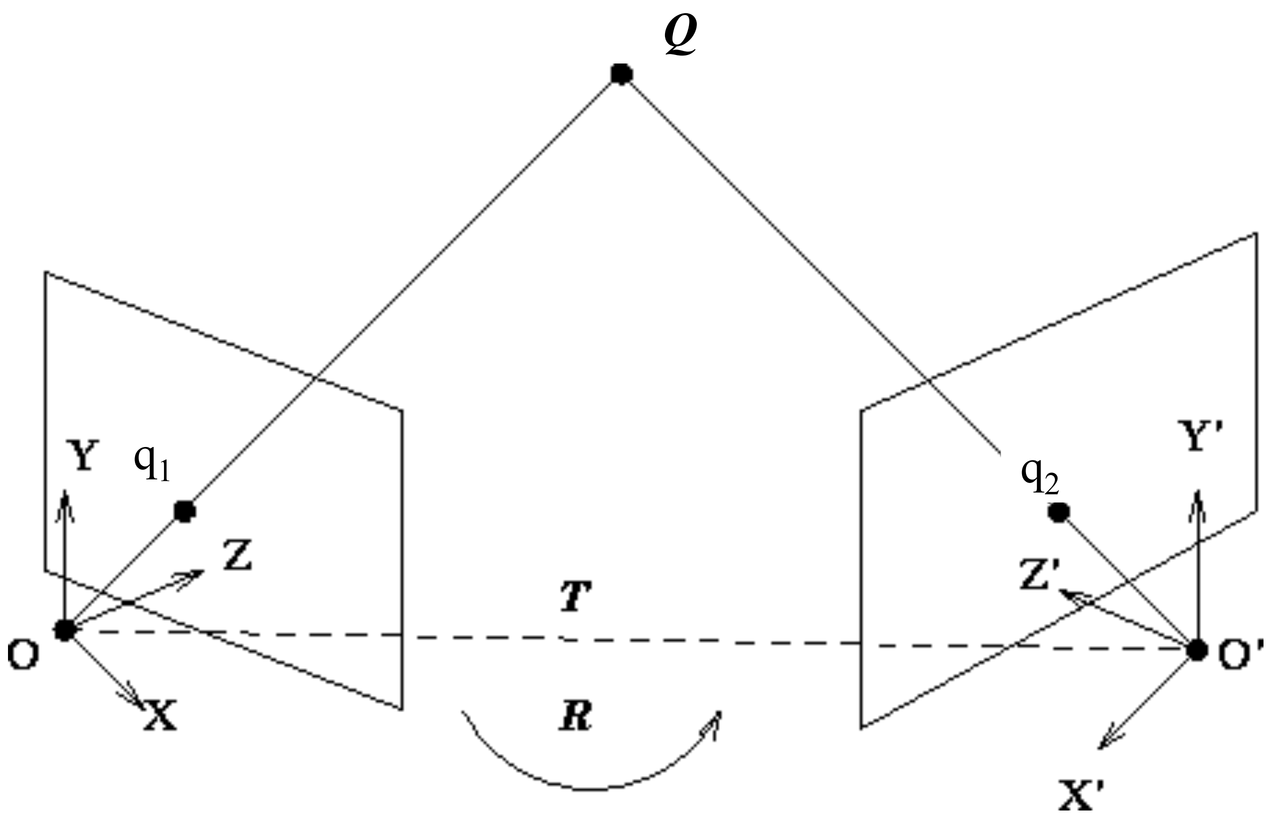


Back to monocular camera

- Stereo camera is a heavy constraint
- How can we track the movement of a single camera?



The epipolar geometry



$$\mathbf{q}_1 \sim \mathbf{K}_1\left[\mathbf{I} \mid \mathbf{0}\right]\begin{bmatrix} \tilde{\mathbf{Q}} \\ 1 \end{bmatrix} \qquad \mathbf{q}_2 \sim \mathbf{K}_2\left[\mathbf{R} \mid \mathbf{t}\right]\begin{bmatrix} \tilde{\mathbf{Q}} \\ 1 \end{bmatrix}$$

The cross matrix

$$\mathbf{a} \times \mathbf{b} = \begin{bmatrix} a_2 b_3 - a_3 b_2 \\ a_3 b_1 - a_1 b_3 \\ a_1 b_2 - a_2 b_1 \end{bmatrix} = [\mathbf{a}]_\times \mathbf{b} = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix} \mathbf{b}$$

Property of skew symmetric matrices

Let **A** be a $n \times n$ skew-symmetric matrix $\Rightarrow \mathbf{A}^T = -\mathbf{A}$

The determinant of A satisfies $\det(\mathbf{A}) = \det(\mathbf{A}^T) = \det(-\mathbf{A}) = (-1)^n \det(\mathbf{A})$

If n is odd the determinant vanishes.

Hence, <u>all odd dimension skew symmetric matrices are singular</u>

The Fundamental and Essential matrices

$$\mathbf{q}_1 \sim \mathbf{K}_1[\mathbf{I} \mid \mathbf{0}]\begin{bmatrix} \tilde{\mathbf{Q}} \\ 1 \end{bmatrix} \qquad \mathbf{q}_2 \sim \mathbf{K}_2[\mathbf{R} \mid \mathbf{t}]\begin{bmatrix} \tilde{\mathbf{Q}} \\ 1 \end{bmatrix}$$

$$\tilde{\mathbf{Q}} \sim \mathbf{K}_1^{-1}\mathbf{q}_1 \quad \Rightarrow \quad \mathbf{K}_2^{-1}\mathbf{q}_2 \sim [\mathbf{R} \mid \mathbf{t}]\begin{bmatrix} \mathbf{K}_1^{-1}\mathbf{q}_1 \\ 1 \end{bmatrix} = \mathbf{R}\mathbf{K}_1^{-1}\mathbf{q}_1 + \mathbf{t}$$

Let's multiply both ends by $[\mathbf{t}]_x$

$$[\mathbf{t}]_x \mathbf{K}_2^{-1}\mathbf{q}_2 \sim [\mathbf{t}]_x \mathbf{R}\mathbf{K}_1^{-1}\mathbf{q}_1 + [\mathbf{t}]_x \mathbf{t} \qquad \text{but } [\mathbf{t}]_x \mathbf{t} = 0 = \mathbf{t} \times \mathbf{t} \quad \Rightarrow [\mathbf{t}]_x \mathbf{K}_2^{-1}\mathbf{q}_2 \sim [\mathbf{t}]_x \mathbf{R}\mathbf{K}_1^{-1}\mathbf{q}_1$$

Let's multiply both ends by $\left(\mathbf{K}_2^{-1}\mathbf{q}_2\right)^{\mathrm{T}}$

$$\mathbf{q}_2^{\mathrm{T}}\mathbf{K}_2^{-\mathrm{T}}[\mathbf{t}]_x \mathbf{K}_2^{-1}\mathbf{q}_2 \sim \mathbf{q}_2^{\mathrm{T}}\mathbf{K}_2^{-\mathrm{T}}[\mathbf{t}]_x \mathbf{R}\mathbf{K}_1^{-1}\mathbf{q}_1$$

but $\mathbf{q}_2^{\mathrm{T}}\mathbf{K}_2^{-\mathrm{T}}[\mathbf{t}]_x \mathbf{K}_2^{-1}\mathbf{q}_2 = 0$ as it is like doing $\mathbf{a}^{\mathrm{T}}[\mathbf{b}]_x \mathbf{a} = \mathbf{a}^{\mathrm{T}}(\mathbf{b} \times \mathbf{a}) = 0$

$$\mathbf{q}_2^{\mathrm{T}} \underbrace{\mathbf{K}_2^{-\mathrm{T}}[\mathbf{t}]_x \mathbf{R}\mathbf{K}_1^{-1}}_{\mathbf{F}} \mathbf{q}_1 = 0 = \mathbf{q}_2^{\mathrm{T}}\mathbf{K}_2^{-\mathrm{T}}\mathbf{E}\mathbf{K}_1^{-1}\mathbf{q}_1 = \mathbf{q}_2^{\mathrm{T}}\mathbf{F}\mathbf{q}_1 \qquad \mathbf{F} = \mathbf{K}_2^{-\mathrm{T}}\mathbf{E}\mathbf{K}_1^{-1}$$

Essential matrix

Fundamental matrix

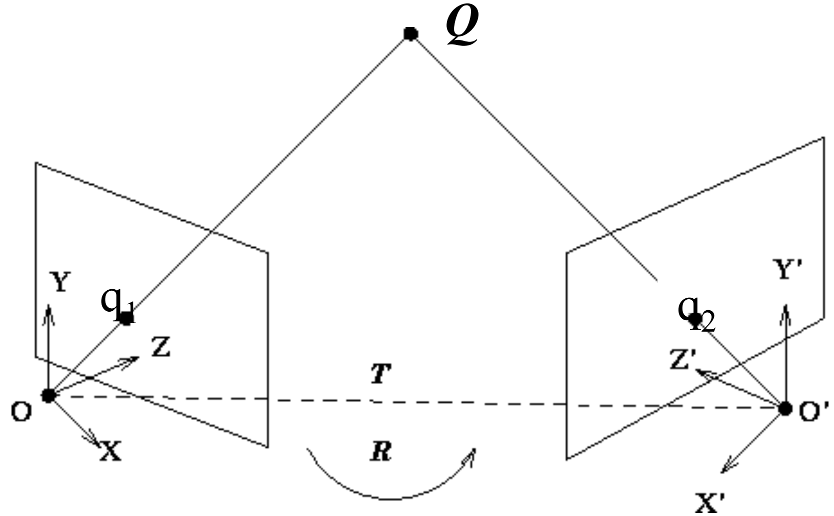
The Essential matrix E

- 3x3 matrix relating corresponding points in 2 views

$$\left.\begin{aligned} \tilde{\mathbf{q}}_2^{\mathrm{T}}\,\mathbf{E}\,\tilde{\mathbf{q}}_2 &= 0 \\ \mathbf{E} &= [\mathbf{t}]_{\mathrm{x}}\,\mathbf{R} \end{aligned}\right\} \quad \text{where} \quad \tilde{\mathbf{q}}_1 \sim \mathbf{K}_1^{-1}\mathbf{q}_1 \quad \text{and} \quad \tilde{\mathbf{q}}_2 \sim \mathbf{K}_2^{-1}\mathbf{q}_2$$

- Calibrated case: **E** depends only on **R** and **t**
- Uncalibrated case: fundamental matrix **F**

$$\mathbf{q}_2^{\mathrm{T}}\,\mathbf{F}\,\mathbf{q}_1 = 0 = \mathbf{q}_2^{\mathrm{T}}\,\mathbf{K}^{-\mathrm{T}}\boxed{[\mathbf{t}]_{\mathrm{x}}\,\mathbf{R}}\mathbf{K}^{-1}\mathbf{q}_1$$



Computing F

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

$$\begin{bmatrix} q_2^x & q_2^y & 1 \end{bmatrix} \begin{bmatrix} \mathbf{f}_1^T \mathbf{q}_1 \\ \mathbf{f}_2^T \mathbf{q}_1 \\ \mathbf{f}_3^T \mathbf{q}_1 \end{bmatrix} = 0 \quad \textit{with} \quad \mathbf{f}_i^T = i\text{-th row of } \mathbf{F}$$

$$q_2^x \mathbf{f}_1^T \mathbf{q}_1 + q_2^y \mathbf{f}_2^T \mathbf{q}_1 + \mathbf{f}_3^T \mathbf{q}_1 = 0$$

$$\begin{bmatrix} q_2^x \mathbf{q}_1^T & q_2^y \mathbf{q}_1^T & \mathbf{q}_1^T \end{bmatrix}_{1\times 9} \begin{bmatrix} \mathbf{f}_1 \\ \mathbf{f}_2 \\ \mathbf{f}_3 \end{bmatrix}_{9\times 1} = 0$$

How many points do we need to compute F?

Computing F

$$\begin{bmatrix} q_2^x \mathbf{q}_1^T & q_2^y \mathbf{q}_1^T & \mathbf{q}_1^T \end{bmatrix}_{1\times 9} \begin{bmatrix} \mathbf{f}_1 \\ \mathbf{f}_2 \\ \mathbf{f}_3 \end{bmatrix}_{9\times 1} = 0$$

For each pair of points we have 1 equation for 9 unknowns

We need at least 8 pairs to solve up to a scale factor

Computing F

change of notation: for each pair $i \rightarrow \mathbf{q}_{2i}^T \mathbf{F} \mathbf{q}_{1i} = 0$

$$\mathbf{A}_{8\times 9}\begin{bmatrix} \mathbf{f}_1 \\ \mathbf{f}_2 \\ \mathbf{f}_3 \end{bmatrix} = \begin{bmatrix} q_{21}^x \mathbf{q}_{11}^T & q_{21}^y \mathbf{q}_{11}^T & \mathbf{q}_{11}^T \\ q_{22}^x \mathbf{q}_{12}^T & q_{22}^y \mathbf{q}_{12}^T & \mathbf{q}_{12}^T \\ \vdots & \vdots & \vdots \\ q_{28}^x \mathbf{q}_{18}^T & q_{28}^y \mathbf{q}_{18}^T & \mathbf{q}_{18}^T \end{bmatrix} \begin{bmatrix} \mathbf{f}_1 \\ \mathbf{f}_2 \\ \mathbf{f}_3 \end{bmatrix}_{9\times 1} = \mathbf{0}$$

Once again, solve a system $\mathbf{Af}=\mathbf{0}$, i.e.

Solve for $|\mathbf{Af}|^2=\mathbf{0}$ subject to $|\mathbf{f}|=\mathbf{1}$

As usual, use SVD s.t. $\mathbf{A} \overset{SVD}{=} \mathbf{UDV}^T$

And set $\mathbf{x}$ as the last column of $\mathbf{V}$

$$\mathbf{A} \overset{SVD}{=} \mathbf{UDV}^T \rightarrow \mathbf{F} = reshape(\mathbf{V}(:,9),3,3)$$

Computing F

- Remember that F must have rank 2
- We need to enforce the constraint
- Solution: set last singular value of F to zero.

$$\mathbf{F} \overset{SVD}{=} \mathbf{UDV}^T \rightarrow \mathbf{D}_{3\times 3} = diag(\sigma_1^2, \sigma_2^2, \sigma_3^2)$$

σ_3^2 should be 0 but in general it is not, then take:

$$\hat{\mathbf{D}}_{3\times 3} = diag(\sigma_1^2, \sigma_2^2, 0) \quad \rightarrow \quad \mathbf{F} \overset{def}{=} \mathbf{U}\hat{\mathbf{D}}\mathbf{V}^T$$

- The same can hold for essential matrix E but... It has some different properties.

The Essential matrix

- Rank(E) = 2 because of $[t]_x$
- Also:

A (3×3) matrix is an essential matrix if and only if two of its singular values are equal and the third one is zero

Proof: based on the fact that $\mathsf{E} = \mathsf{SR}$ with $\mathsf{S} = [\mathbf{t}]_\times$ and $\mathsf{R} \in SO(3)$

- Define:

$$\mathsf{W} = \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad \text{and} \quad \mathsf{Z} = \begin{pmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

- S can be decomposed as $\mathsf{S} = k\mathsf{UZU}^\mathsf{T}$ with $\mathsf{U} \in O(3)$ (Property of skew symmetric matrices)

- We have $\mathsf{Z} = -\ \mathrm{diag}(1,1,0)\mathsf{W}$ and thus:

$$\mathsf{S} \sim \mathsf{U}\ \mathrm{diag}(1,1,0)\mathsf{WU}^\mathsf{T}$$

- A Singular Value Decomposition of E is thus:

$$\boxed{\mathsf{E} \sim \mathsf{U}\ \mathrm{diag}(1,1,0)\left(\mathsf{WU}^\mathsf{T}\mathsf{R}\ \right)}$$

Computation of the Essential matrix E

- Can be estimated with 8 pairs of corresponding pairs
  - Hence the "famous" **8 points algorithm**
  - Similar estimation procedure to fundamental matrix
- With respect to fundamental matrix, **E** has 2 constraints

$$\det \mathbf{E} = 0$$

$$2\mathbf{E}\mathbf{E}^{\mathrm{T}} - tr(\mathbf{E}\mathbf{E}^{\mathrm{T}})\mathbf{E} = 0$$

2 equal eigenvalues and
1 null eigenvalue

- Imposing these constraints allows to reduce the number of corresponding pairs
- <u>5 corresponding pairs are enough !</u>
  - (up to 10 solutions, see later)
- hence the "famous" **5 point algorithm**

Decomposition of E – solving for R and t

- Compute E with with the 5 points algorithm
- We need to decompose **E** in order to find **R** and **t**
- From the property of **E** we know that if we decompose E with a SVD we get:

$$\mathbf{E} \overset{SVD}{\sim} \mathbf{U}\, diag(1,1,0)\, \mathbf{V}^T$$

- Hence also **U** and **V** are known (from svd(E))
- We are going to use them for decomposing **E** into the **R** a**nd** t parts

Decomposition of E – solving for R and t

★ The translation:

$\mathrm{rank}(\mathbf{E}) = 2 \leq \min(\mathrm{rank}(\mathbf{S}), \mathrm{rank}(\mathbf{R})) \Rightarrow \mathrm{rank}(\mathbf{S}) = 2$

- Matrix E has rank 2 and its left nullspace is $\mathbf{u}_3$
- Matrix S is skew symmetric and must have the same nullspace as E, thus:

$$\mathsf{S} \sim [\mathbf{u}_3]_\times \qquad \text{and} \qquad \mathbf{t} \sim \mathbf{u}_3$$

★ The rotation:

$$\mathbf{E} \overset{SVD}{\sim} \mathbf{U}\, diag(1,1,0)\, \mathbf{V}^T$$

- We write $\mathsf{R} = \mathsf{U}\mathsf{X}\mathsf{V}^\mathsf{T}$ with $\mathsf{X} \in O(3)$
- Using $\mathsf{S} \sim \mathsf{U}\mathsf{Z}\mathsf{U}^\mathsf{T}$, we get:

$$\begin{aligned} \mathsf{E} &\sim \mathsf{S}\mathsf{R} \\ &\sim \mathsf{U}\mathsf{Z}\mathsf{U}^\mathsf{T}\mathsf{U}\mathsf{X}\mathsf{V}^\mathsf{T} \\ &\sim \mathsf{U}\mathsf{Z}\mathsf{X}\mathsf{V}^\mathsf{T} \end{aligned}$$

and thus $\mathsf{Z}\mathsf{X} = \mathrm{diag}(1, 1, 0)$ giving:

$\mathbf{Z} = \pm\,\mathrm{diag}(1,1,0)\mathbf{W}$

$$\mathsf{X} = \mathsf{W} \qquad \text{or} \qquad \mathsf{X} = \mathsf{W}^\mathsf{T}$$

Decomposition of E

The Singular Value Decomposition of E is:

$$\mathsf{E} = \mathsf{U}\ \mathrm{diag}(1,1,0)\ \mathsf{V}^\mathsf{T}$$

then the following two solutions are possible for R:

$$\begin{aligned} \mathsf{R} &= \mathsf{U}\mathsf{W}\mathsf{V}^\mathsf{T} \\ \mathsf{R} &= \mathsf{U}\mathsf{W}^\mathsf{T}\mathsf{V}^\mathsf{T} \end{aligned}$$

and for $\mathbf{t}$:

$$\mathbf{t} = \pm\mathbf{u}_3$$

Among the 4 solutions, 1 is feasible

Four possible solutions

★ The sign of $\mathbf{t}$ is undetermined

★ Combining with the two possible rotations, this gives:

$$\mathsf{P}' \sim (\mathsf{UWV}^\mathsf{T} \;\; \mathbf{u}_3) \qquad \text{or} \qquad \mathsf{P}' \sim (\mathsf{UWV}^\mathsf{T} \;\; -\mathbf{u}_3)$$

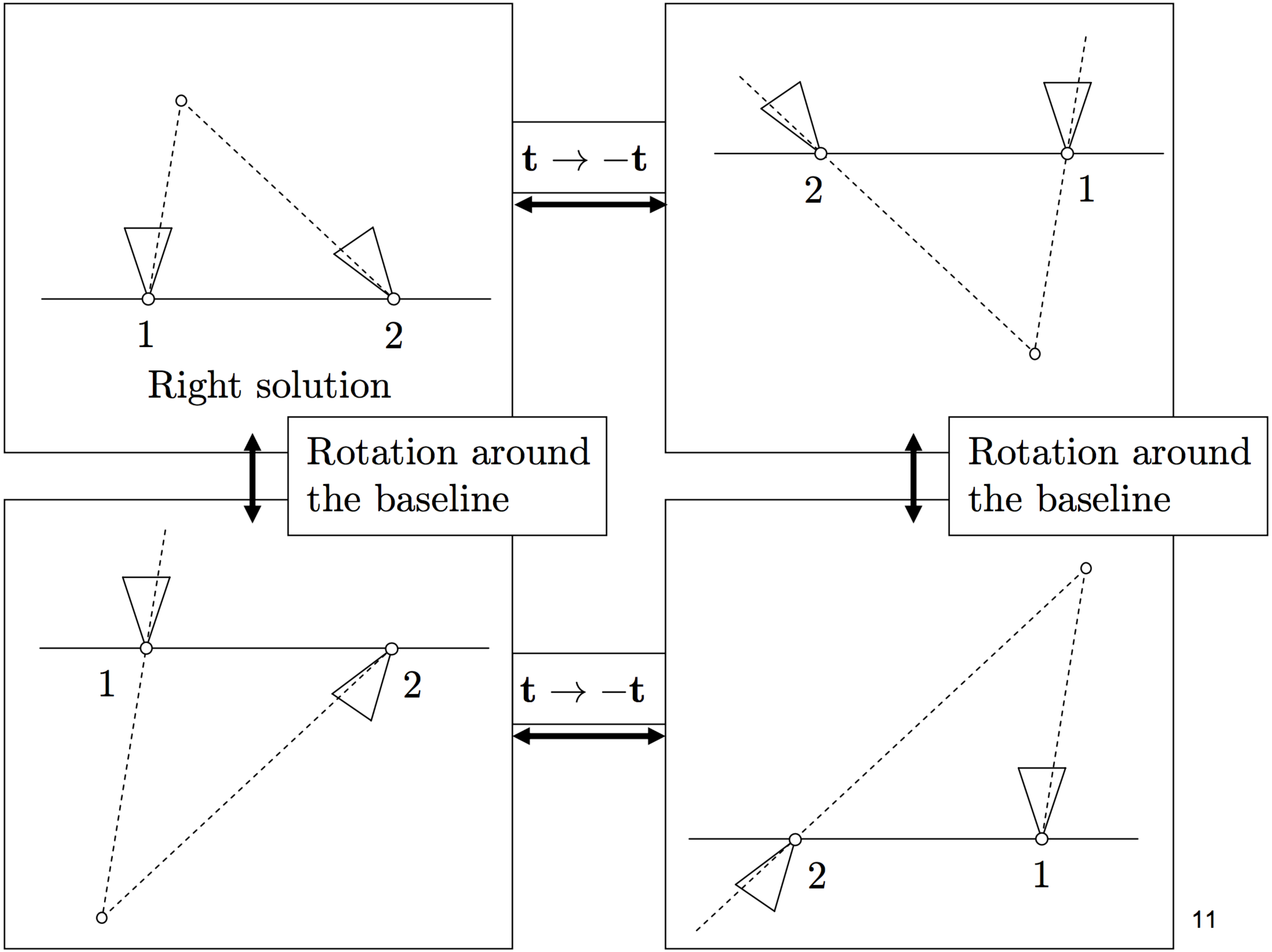
$$\mathsf{P}' \sim (\mathsf{UW}^\mathsf{T}\mathsf{V}^\mathsf{T} \;\; \mathbf{u}_3) \qquad \text{or} \qquad \mathsf{P}' \sim (\mathsf{UW}^\mathsf{T}\mathsf{V}^\mathsf{T} \;\; -\mathbf{u}_3)$$

★ The $\mathbf{u}_3 \to -\mathbf{u}_3$ swaps the position of the cameras

★ The $\mathsf{UWV}^\mathsf{T} \to \mathsf{UW}^\mathsf{T}\mathsf{V}^\mathsf{T}$ makes a rotation of π around the baseline

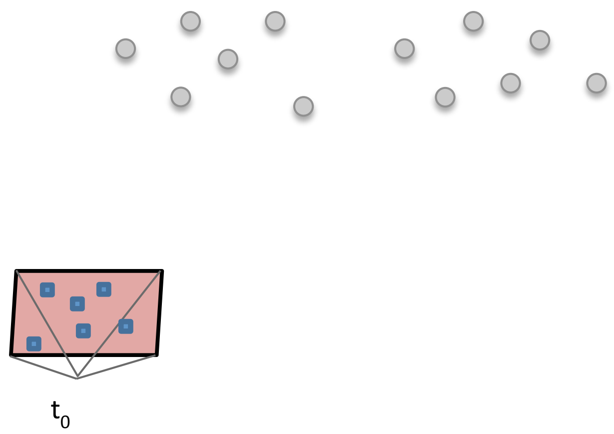
★ Only one solution is feasible

Four possible solutions



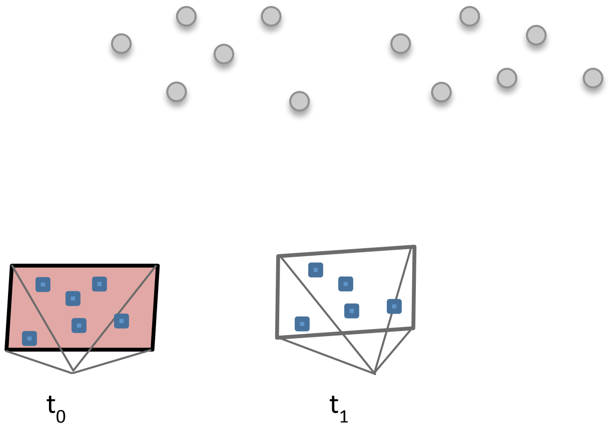
Initialization with monocular camera

- K-frame based tracking
- Main idea:
  - 3D reconstruction only when we have sufficient camera displacement
  - Kframe selection based on
    - Number of lost features
    - Actual movement if available



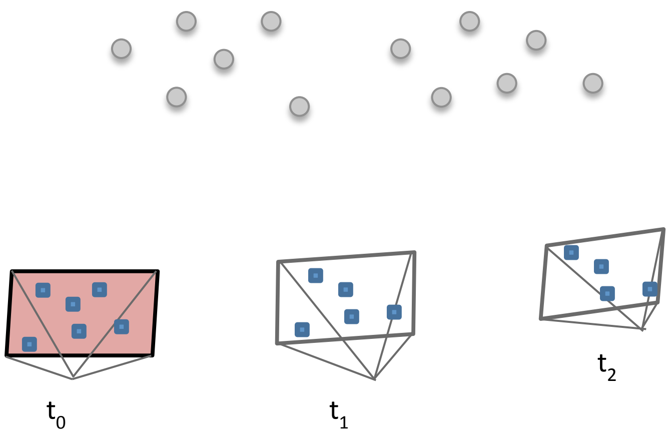
Initialization with monocular camera

- K-frame based tracking
- Main idea:
  - 3D reconstruction only when we have sufficient camera displacement
  - Kframe selection based on
    - Number of lost features
    - Actual movement if available



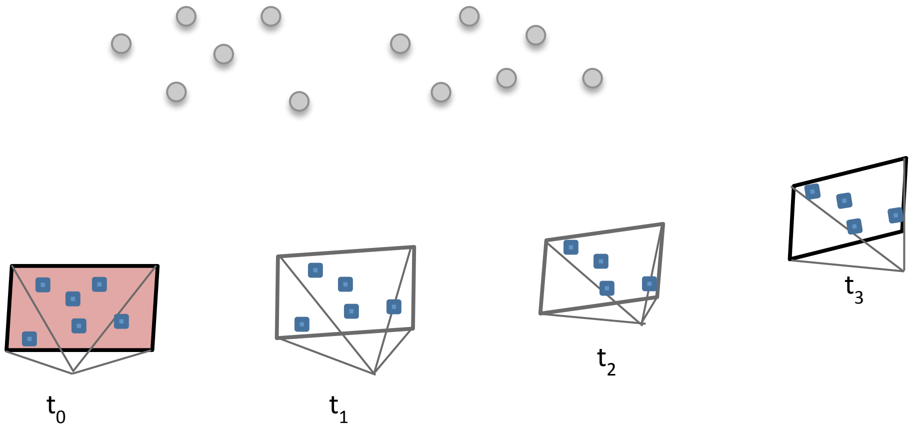
Initialization with monocular camera

- K-frame based tracking
- Main idea:
  - 3D reconstruction only when we have sufficient camera displacement
  - Kframe selection based on
    - Number of lost features
    - Actual movement if available



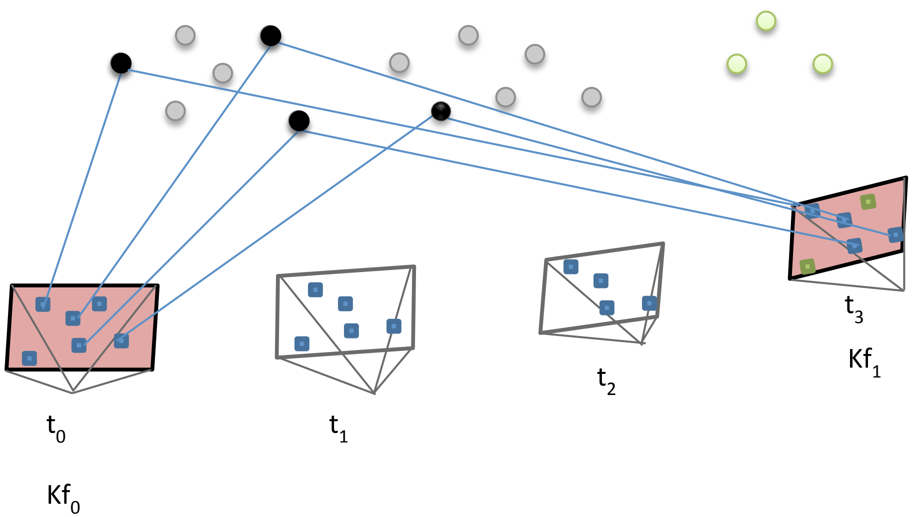
Initialization with monocular camera

- K-frame based tracking
- Main idea:
  - 3D reconstruction only when we have sufficient camera displacement
  - Kframe selection based on
    - Number of lost features
    - Actual movement if available



Initialization with monocular camera

- K-frame based tracking
- Main idea:
  - 3D reconstruction only when we have sufficient camera displacement
  - Kframe selection based on
    - Number of lost features
    - Actual movement if available
  - Then for each new kframe detect and [add new features to track]



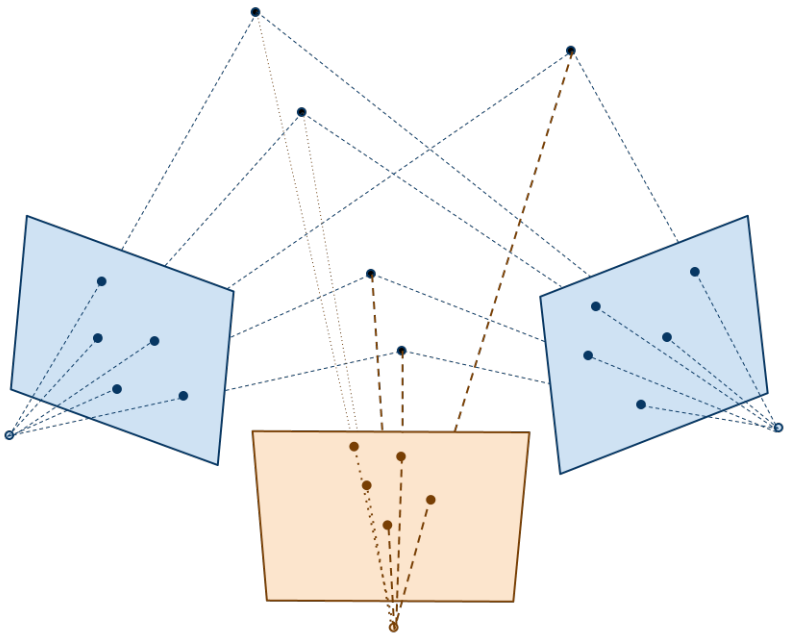
https://www.youtube.com/watch?v=Y9HMn6bd-v8

Initialization with 3 Kframes

- Only once at beginning
- Track and detect markers
  - Get 3 Kframes
  - Solve the 3 kframe geometry
  - 5 points algorithm
  - 3D reconstruction
- Then start tracking
- Drawback (inevitable):
  - No 3D information during initialization

Initialization with 3 Kframes

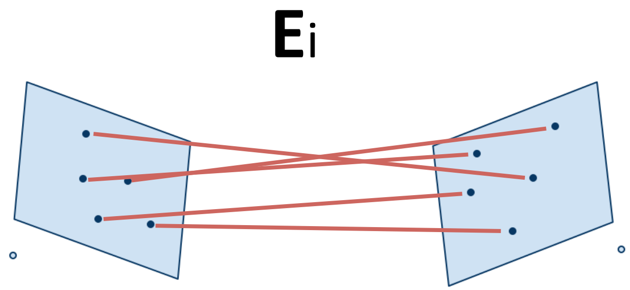
- 5 Points algorithm
  - Robust algorithm to solve 3 view problem
  - Based on solution of a minimal problem
  - Multiple solutions possible, other points to choose the correct one



Nistér, D. 2004. *An Efficient Solution to the Five-Point Relative Pose Problem*. IEEE Trans. Pattern Anal. Mach. Intell. 26, 6 (Jun. 2004), 756-777. doi:http://dx.doi.org/10.1109/TPAMI.2004.17

5 Points algorithm

- Estimate **E** from 2 views (up to 10 solutions)

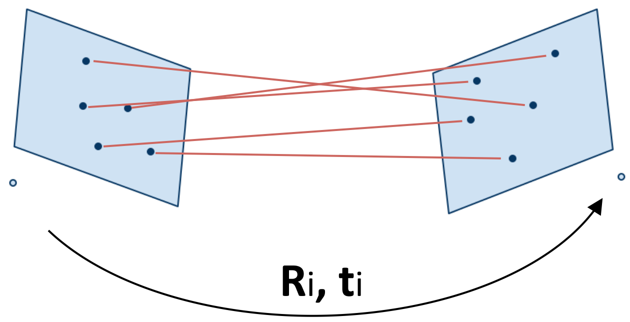


5 Points algorithm

- estimate the E (up to 10 solutions)
- for each solution $\mathbf{E}_i$
  - decompose $\mathbf{E}_i$ in **R** and **t**
  - up to 8 possible solutions for R and t

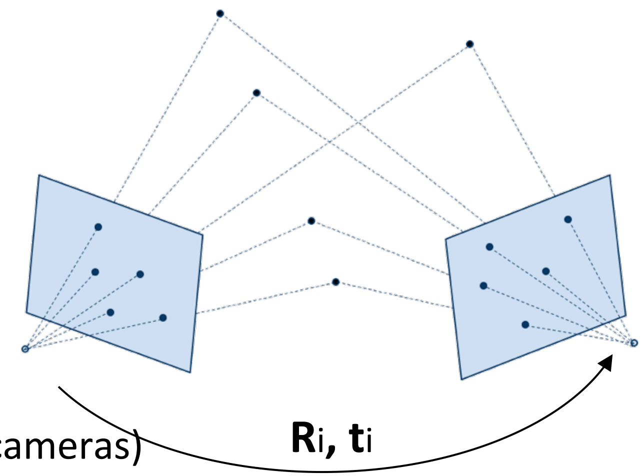
8x

$\mathbf{E}_i \rightarrow \mathbf{R}, \mathbf{t}$



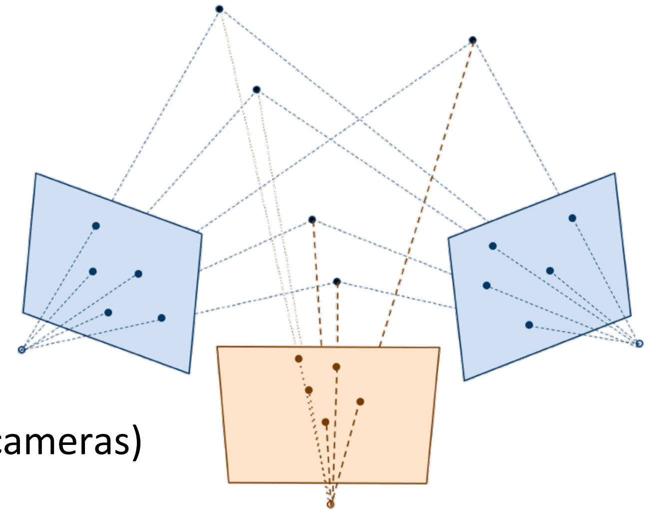
5 Points algorithm

- estimate the E (up to 10 solutions)
- for each solution $\mathbf{E}_i$
  - decompose $\mathbf{E}_i$ in $\mathbf{R}$ and $\mathbf{t}$
  - up to 8 possible solutions for R and t
  - for each possible $\mathbf{R}_i$ and $\mathbf{t}_i$
    - reconstruct the 5 3D points
    - cheirality test (points must be in front of the cameras)



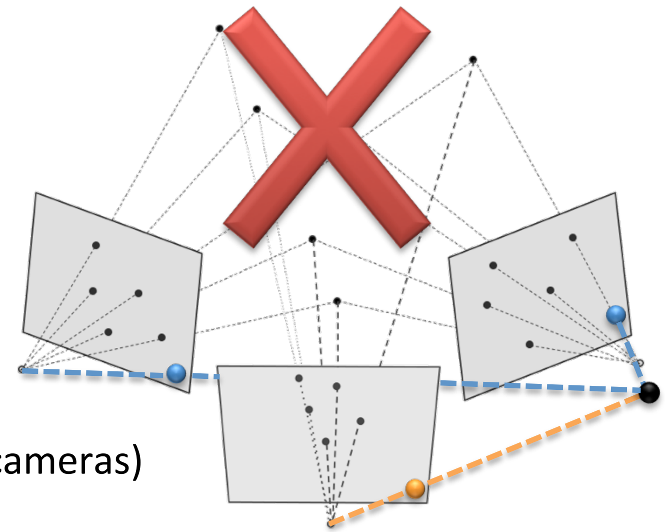
5 Points algorithm

- estimate the E (up to 10 solutions)
- for each solution $\mathbf{E}_i$
  - decompose $\mathbf{E}_i$ in $\mathbf{R}$ and $\mathbf{t}$
  - up to 8 possible solutions for R and t
  - for each possible $\mathbf{R}_i$ and $\mathbf{t}_i$
    - reconstruct the 5 3D points
    - cheirality test (points must be in front of the cameras)
  - consider the feasible $\mathbf{R}_i$ and $\mathbf{t}_i$
  - compute the pose of the 3rd camera (resection problem)

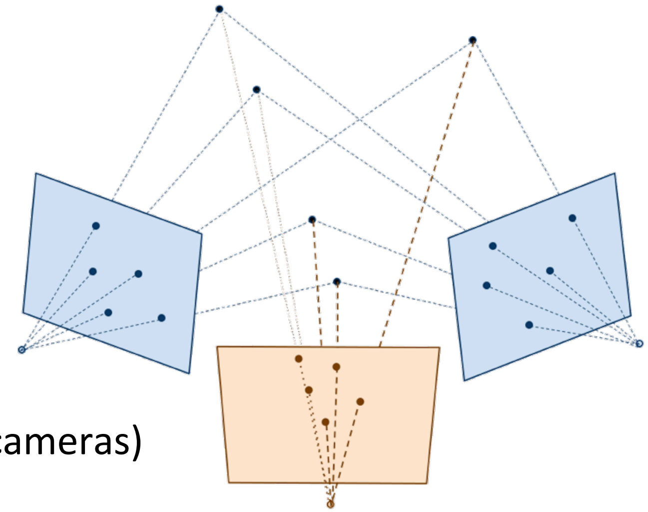


5 Points algorithm

- estimate the E (up to 10 solutions)
- for each solution $\mathbf{E}_i$
  - decompose $\mathbf{E}_i$ in $\mathbf{R}$ and $\mathbf{t}$
  - up to 8 possible solutions for R and t
  - for each possible $\mathbf{R}_i$ and $\mathbf{t}_i$
    - reconstruct the 5 3D points
    - cheirality test (points must be in front of the cameras)
  - consider the feasible $\mathbf{R}_i$ and $\mathbf{t}_i$
  - compute the pose of the 3rd camera (resection problem)
  - reconstruct all the other points
  - cheirality test to validate the solution

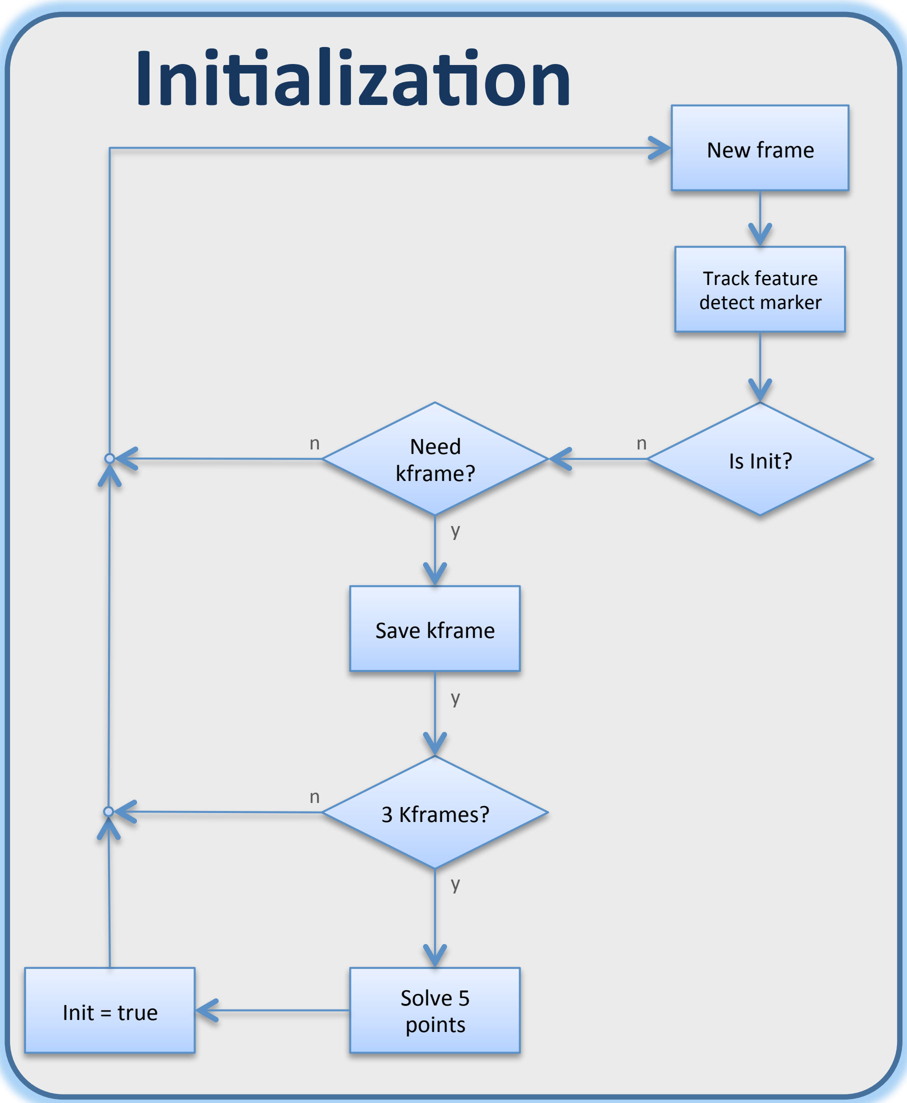


5 Points algorithm



- estimate the E (up to 10 solutions)
- for each solution $\mathbf{E}_i$
  - decompose $\mathbf{E}_i$ in $\mathbf{R}$ and $\mathbf{t}$
  - up to 8 possible solutions for R and t
  - for each possible $\mathbf{R}_i$ and $\mathbf{t}_i$
    - reconstruct the 5 3D points
    - cheirality test (points must be in front of the cameras)
  - consider the feasible $\mathbf{R}_i$ and $\mathbf{t}_i$
  - compute the pose of the 3rd camera (resection problem)
  - reconstruct all the other points
  - cheirality test to validate the solution
- bundle adjustment to refine the poses and the 3D structure
  - if more than one solution choose the one with best (lowest) reprojection error.

An initialization algorithm



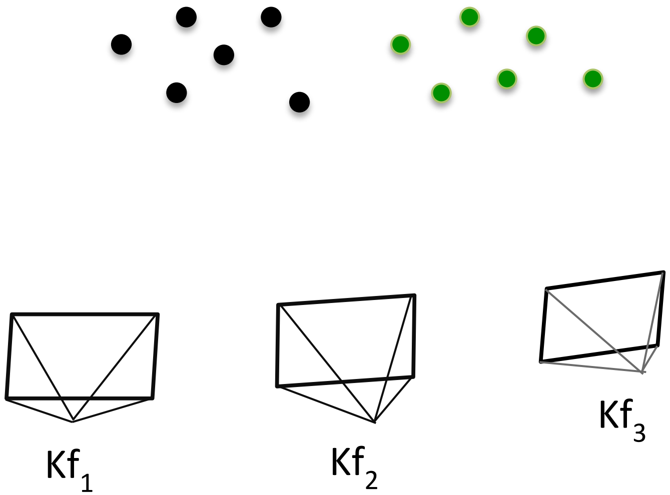
- Main Idea
  - Only once at beginning
  - No 3D information during initialization
  - Track and detect markers
  - Get 3 Kframes
  - Solve the 3 kframe geometry
    - 5 points algorithm
  - 3D reconstruction
  - Then start tracking

And after the initialization?

- After initialization step we have:
  - 3 kframes
  - Set of 3D points from reconstruction
  - Set of new features detected in Kf_3 without 3D

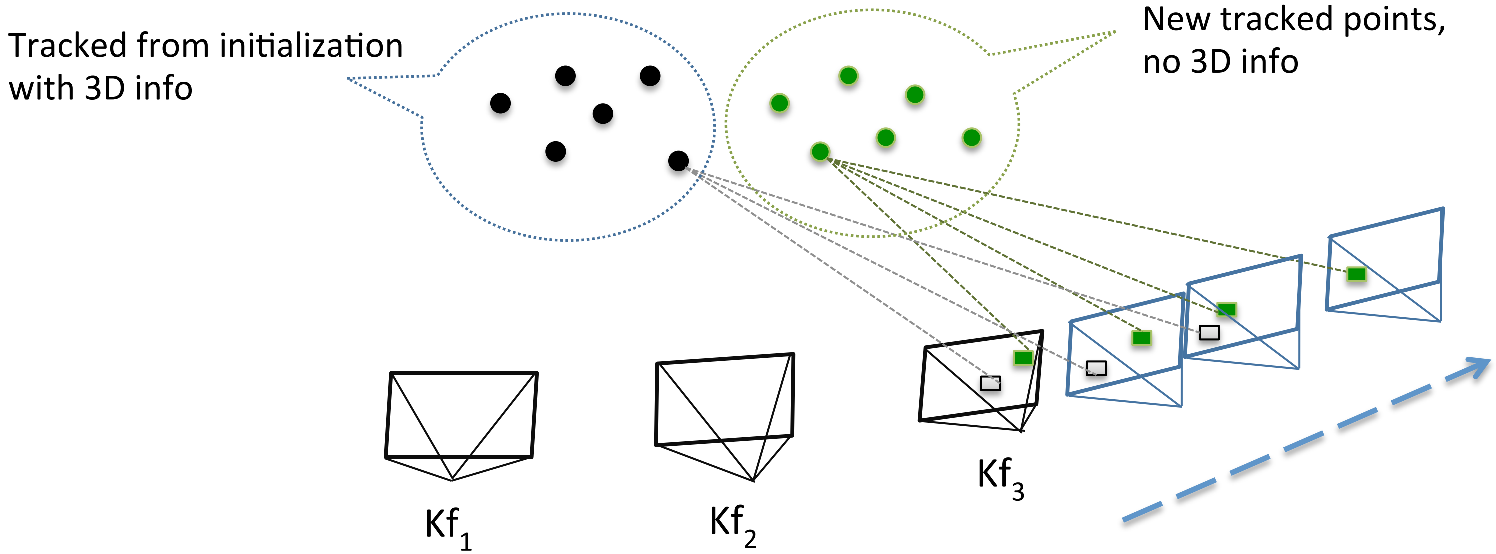
3D points visible by all the
3 kframes

Features detected in the 3rd
Kframe not yet reconstructed



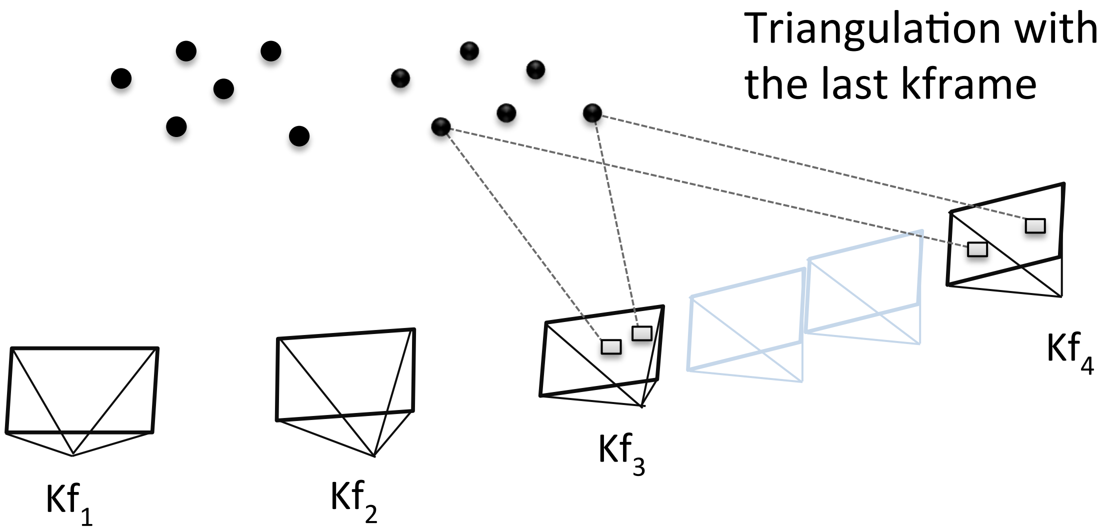
Tracking with kframes

- For each new frame
  - The 3D points are tracked and used to estimate the pose (PnP)
  - The new features are tracked
  - Kframe selection as in the initialization step

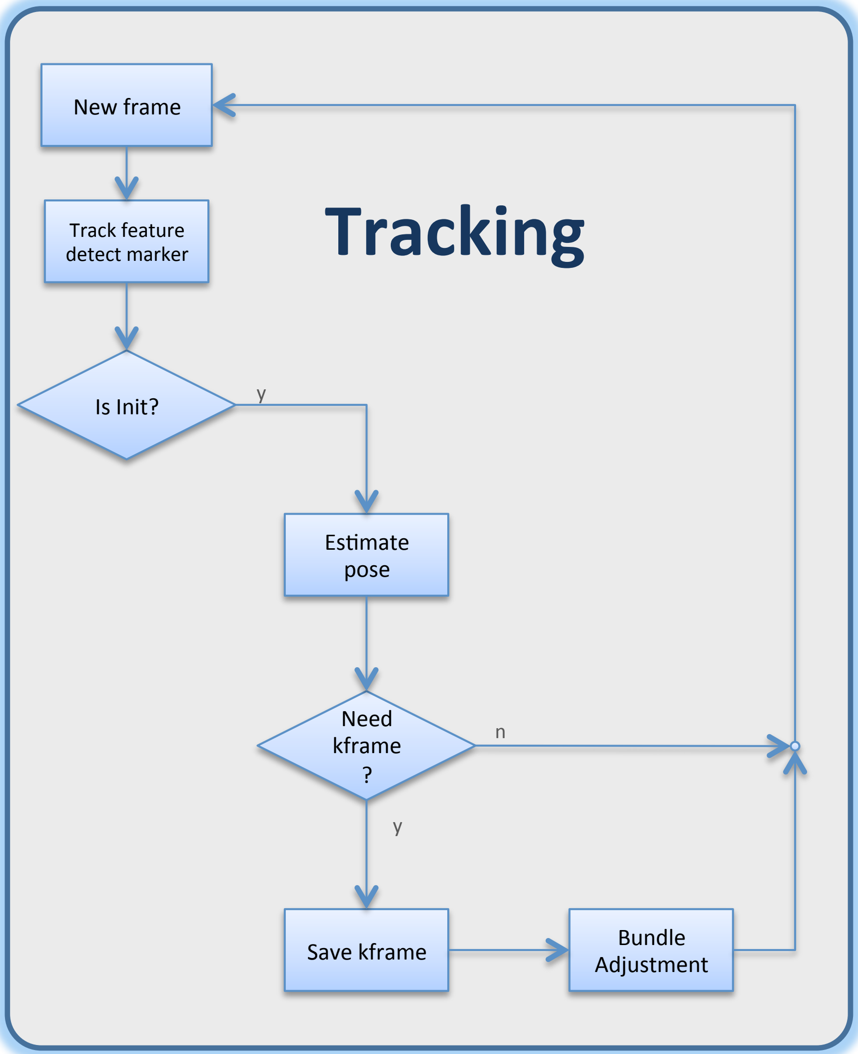


Tracking with kframes

- When a new key frame is needed
  - New tracked features reconstructed by triangulation with the last kframe
  - Optimization of all the 3D points and camera poses (bundle adjustment)

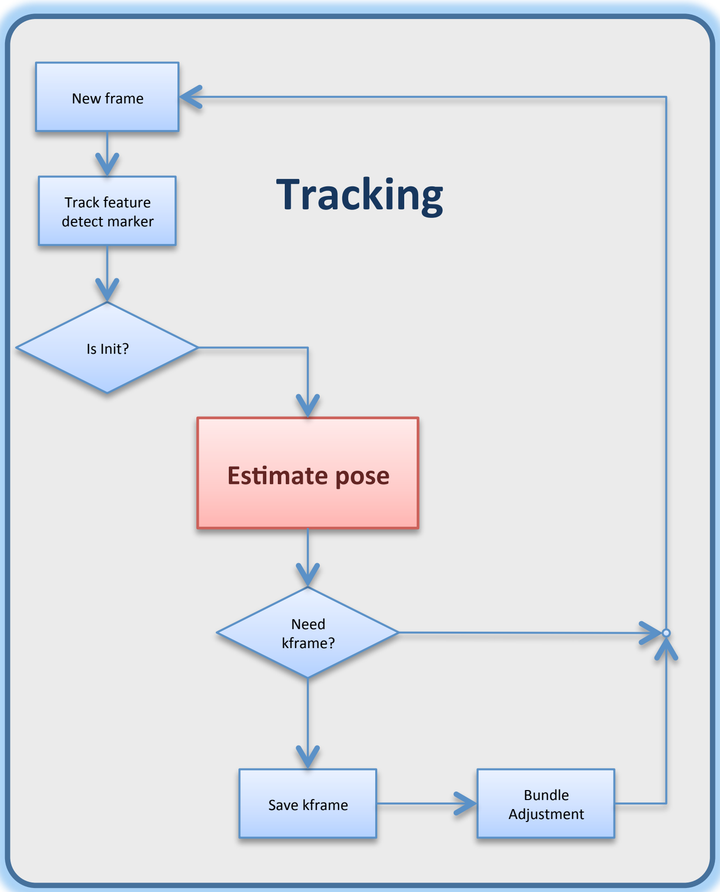


A tracking algorithm



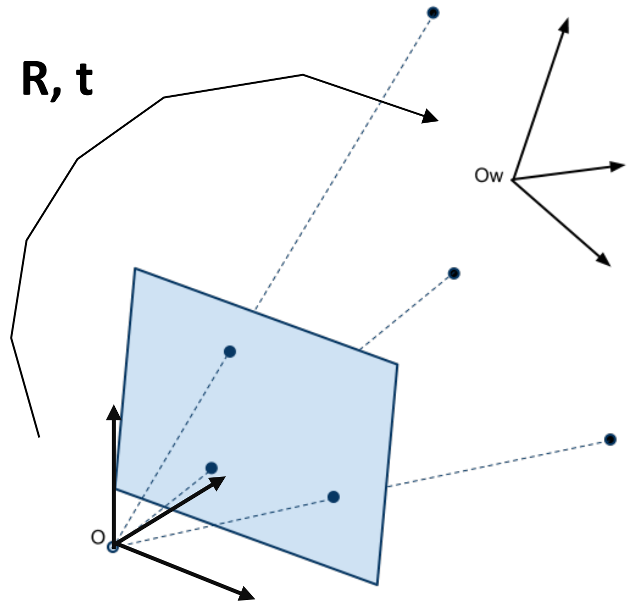
- At every frame
- Track features and markers
- Estimate pose
- If kframe needed
  - 3D reconstruction of newly tracked points
  - Bundle adjustment
  - Detect new points

Pose estimation



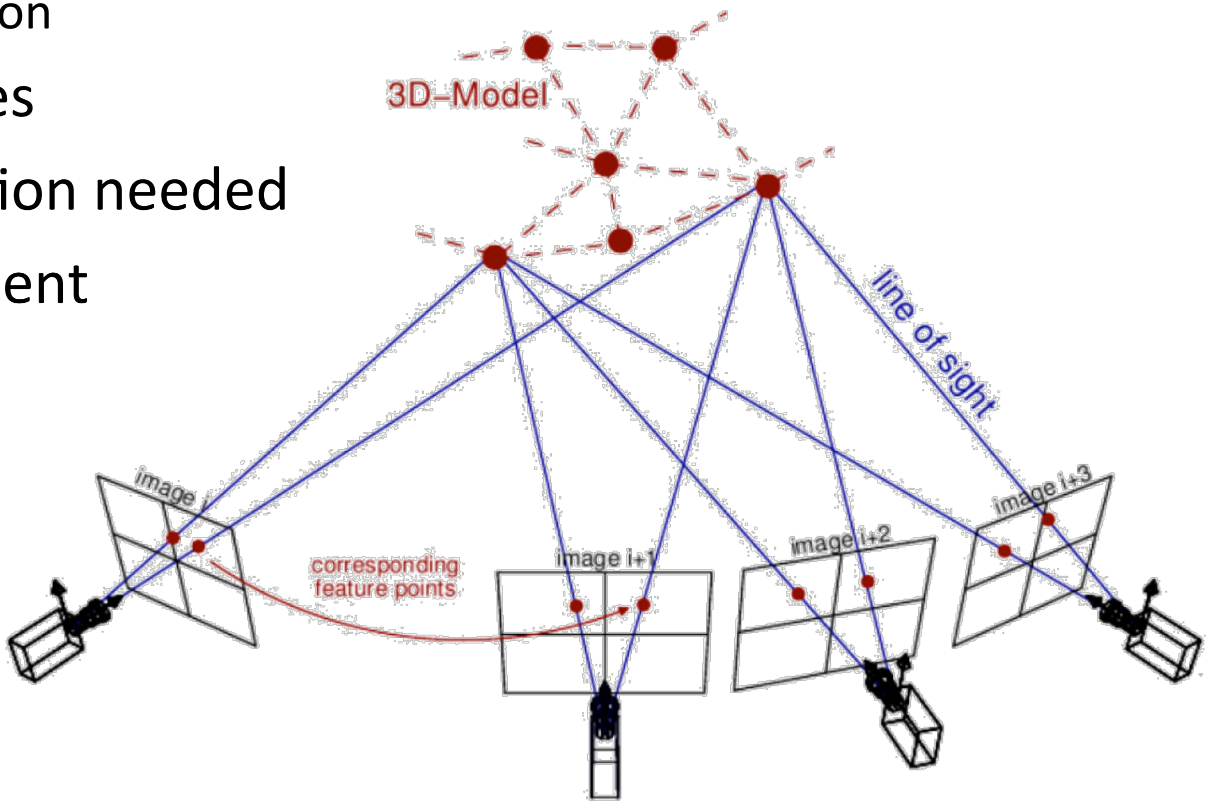
- 3D – 2D correspondences
- Resection (PnP) problem:
  – Estimate the rotation **R** and the translation **t**

$$\mathbf{q}_i \sim \mathbf{K}\left[\mathbf{R} \mid \mathbf{t}\right]\mathbf{Q}_i$$

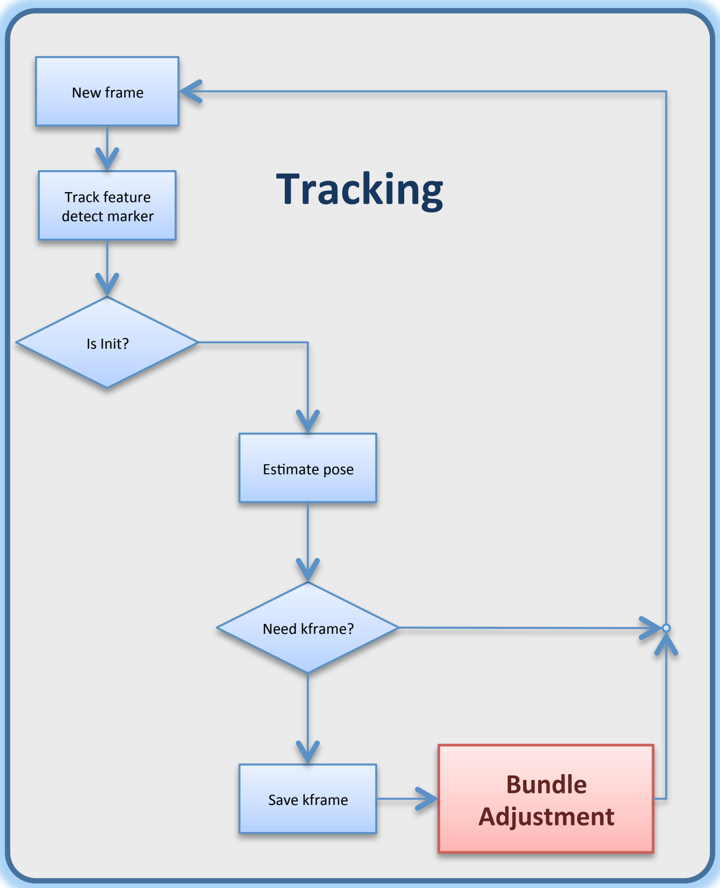


Challenges (2)

- **Errors accumulation**
- Errors are inevitable due to noise etc…:
  - 3D reconstruction
  - Pose estimation
- Errors cumulates
- Global registration needed
- Bundle adjustment



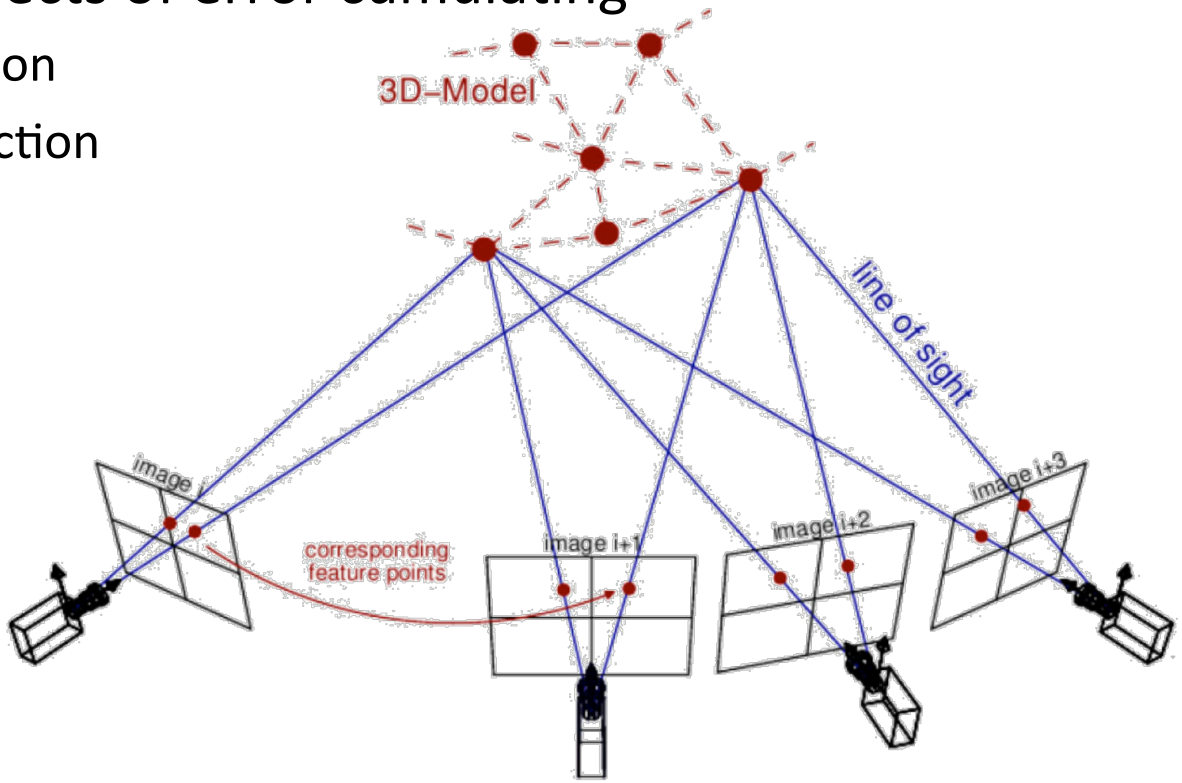
Bundle Adjustment



- **Local bundle adjustment**
  - Optimization over a subset of kframes
    - 3D structure and camera pose
- Optimization over:
  - The (kframe) camera poses
  - The 3D points
- Mitigate error cumulating
  - Pose estimation
  - 3D reconstruction

Bundle Adjustment

- Global optimization over:
  - The (kframe) camera poses
  - The 3D points
- Mitigate the effects of error cumulating
  - Pose estimation
  - 3D reconstruction



Bundle Adjustment

- Refines a visual reconstruction to produce jointly optimal 3D structure and viewing parameters
- *'bundle'* → bundle of light rays leaving each 3D feature and converging on each camera center.
- Non linear Least-squares fitting
  - maximum likelihood estimation of the fitted parameters if the measurement errors are independent and normally distributed with constant standard deviation
  - The probability distribution of the sum of a very large number of very small random deviations almost always converges to a normal distribution

Bundle Adjustment

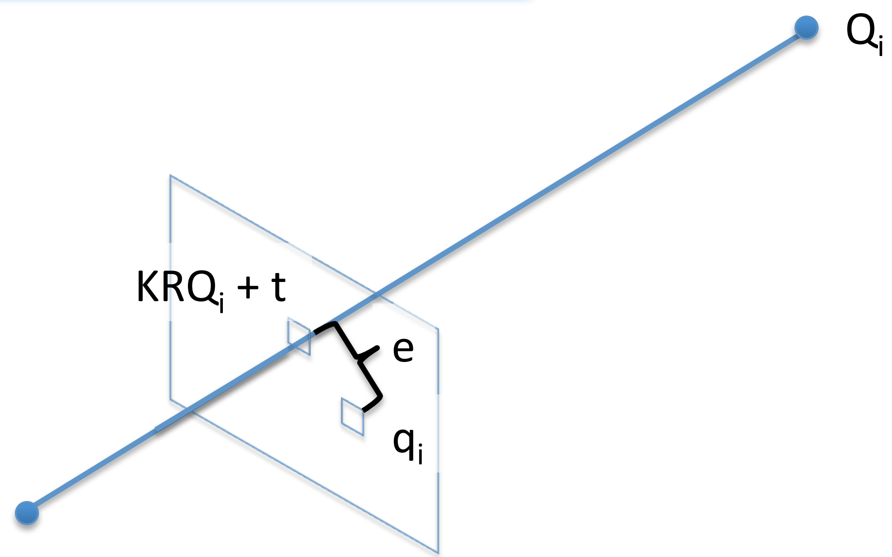
- Reprojection error for a 3D point wrt its image point (measure)

For a single point in one camera

$$e\left(\mathbf{R},\mathbf{t},\mathbf{Q}_i\right)=\left\|\mathbf{q}_i-\left(\mathbf{KRQ}_i+\mathbf{t}\right)\right\|$$

3 parameters for the 3D point
6 parameters for the camera (R and t)

Warning: abuse of notation
Scale factor λ missing



Bundle Adjustment

- Reprojection error for a 3D point wrt its image point (measure)

For a single point in one camera

$$e(\mathbf{R},\mathbf{t},\mathbf{Q}_i) = \left\| \mathbf{q}_i - (\mathbf{K}\mathbf{R}\mathbf{Q}_i + \mathbf{t}) \right\|$$

3 parameters for the 3D point
6 parameters for the camera (R and t)

For M points in one camera

$$e(\mathbf{R},\mathbf{t},\mathbf{Q}_i) = \sum_i^M w_i \left\| \mathbf{q}_i - (\mathbf{K}\mathbf{R}\mathbf{Q}_i + \mathbf{t}) \right\|$$

3*M parameters the 3D points
6 parameters for the camera (R and t)

Indicator variable:
1 if point i is visible from the camera
0 otherwise

Bundle Adjustment

- Reprojection error for a 3D point wrt its image point (measure)

For a single point in one camera

$$e\left(\mathbf{R},\mathbf{t},\mathbf{Q}_i\right)=\left\|\mathbf{q}_i-\left(\mathbf{KRQ}_i+\mathbf{t}\right)\right\|$$

3 parameters for the 3D point Q_i
6 parameters for the camera (R and t)

For M points in one camera

$$e\left(\mathbf{R},\mathbf{t},\mathbf{Q}_i\right)=\sum_i^M w_i\left\|\mathbf{q}_i-\left(\mathbf{KRQ}_i+\mathbf{t}\right)\right\|$$

3*M parameters the 3D points Q_i
6 parameters for the camera (R and t)

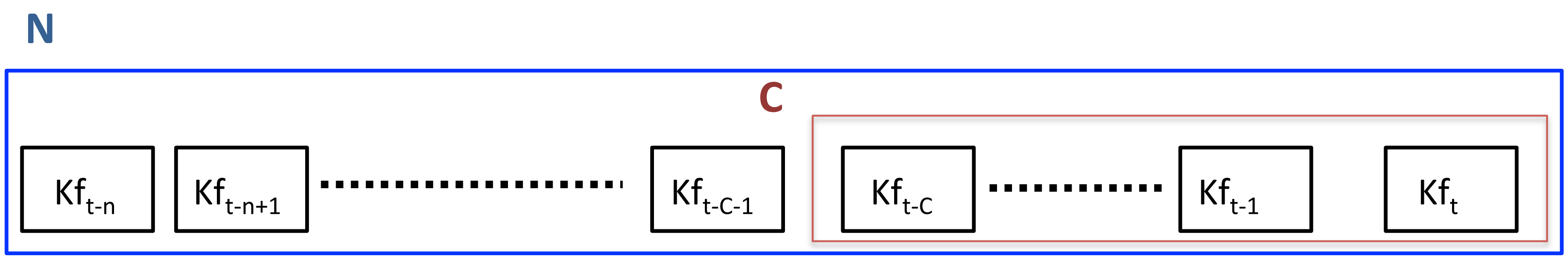
For M points in N cameras

$$e\left(\mathbf{R},\mathbf{t},\mathbf{Q}_i\right)=\sum_j^N\sum_i^M w_{ij}\left\|\mathbf{q}_{ij}-\left(\mathbf{K}_j\mathbf{R}_j\mathbf{Q}_i+\mathbf{t}_j\right)\right\|$$

3*M parameters the 3D points Q_i
6*N parameters for each camera
(R_j and t_j)

Bundle Adjustment

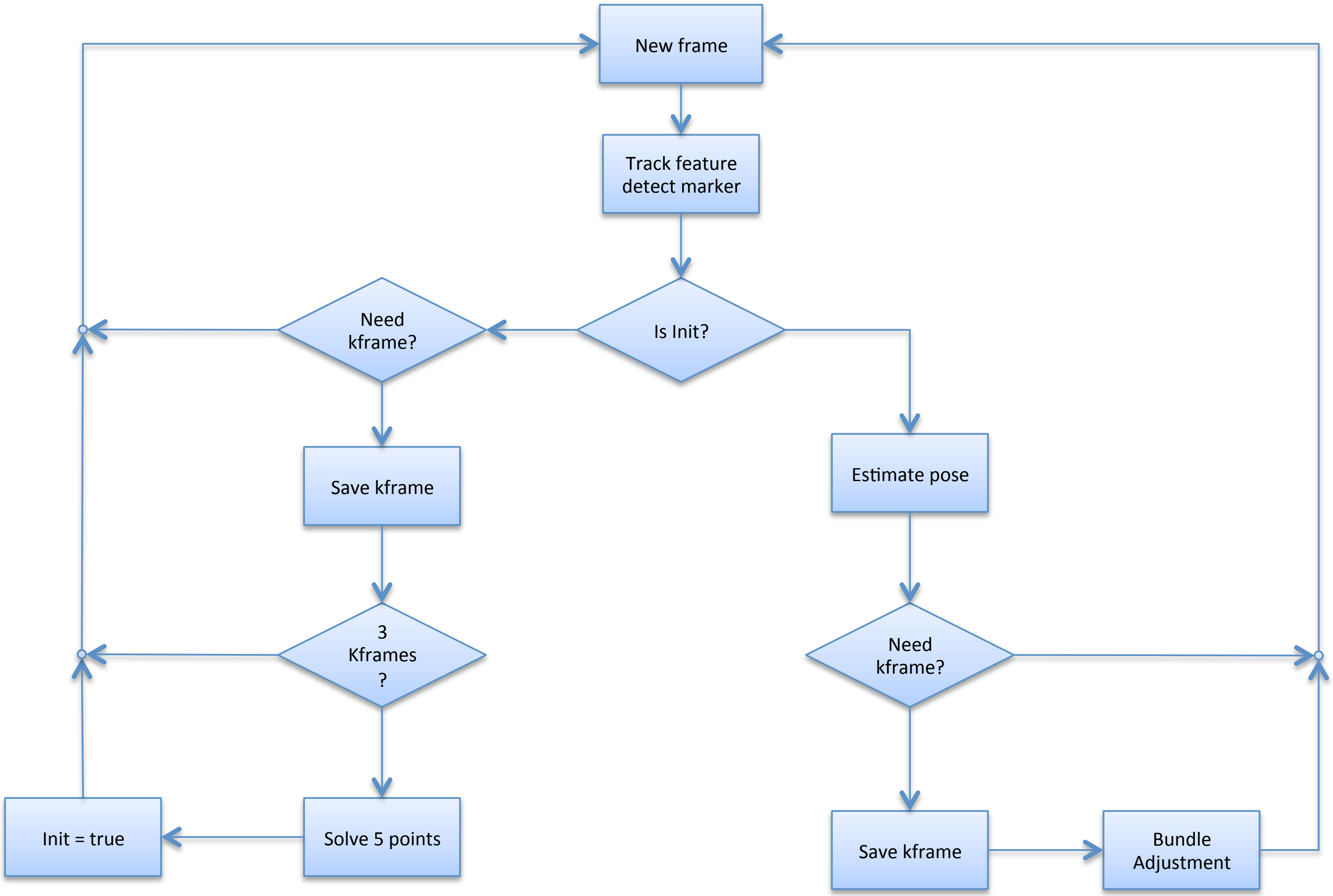
- **Local bundle adjustment**
  - Optimization over a subset of kframes rather than all
    - 3D structure and camera pose
  - Reduce computation and memory
  - only the last **C**<**N** <u>cameras are optimized</u>
  - reprojection error accounted for last **N** key frames.



3 parameters for each 3D point
6 parameters for each camera (R and t)

Only 6*C + 3*P variables

The overall algorithm



The overall algorithm