Practical Course: Vision-based Navigation
SS 2018

Lecture 4. Visual Odometry

Dr. Jörg Stückler, Dr. Xiang Gao
Vladyslav Usenko, Prof. Dr. Daniel Cremers

Contents

- Feature extraction and matching
- Optical Flow
- Pose estimation approaches
- Direct method

Contents

- Feature extraction and matching
- Optical Flow
- Pose estimation approaches
- Direct method

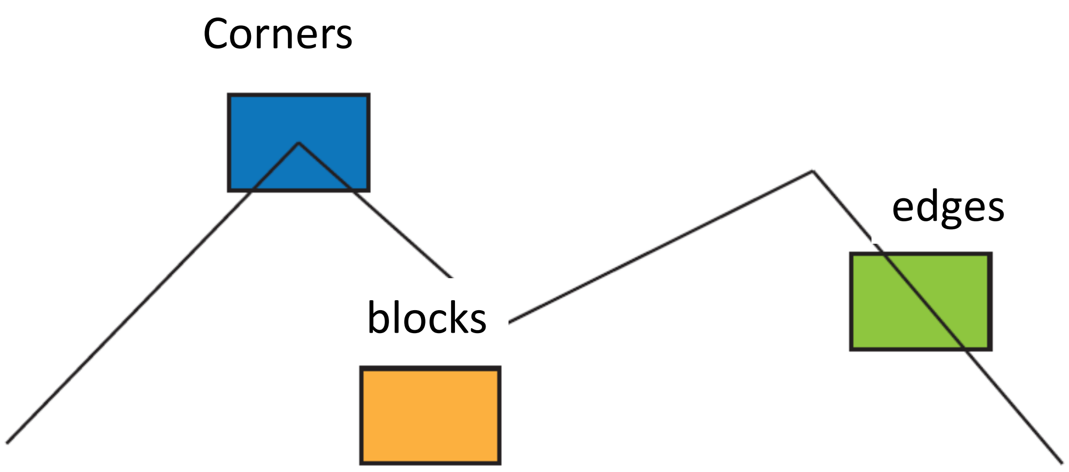
1. Feature Extraction and Matching

- Visual odometry steps
  - Find the corresponding points in the images
  - Estimate camera motion
  - Expand the map if needed

1. Feature Extraction and Matching

- We estimation camera pose by the observed landmarks
  - Landmarks: fixed in 3D space and can be observed in the image
  - Distinctive: landmarks should be easy to distinguish
- Image features are used as landmarks in visual SLAM

- Image features
  - Repeatable
  - Distinctive
  - Efficient
  - Local



1. Feature Extraction and Matching

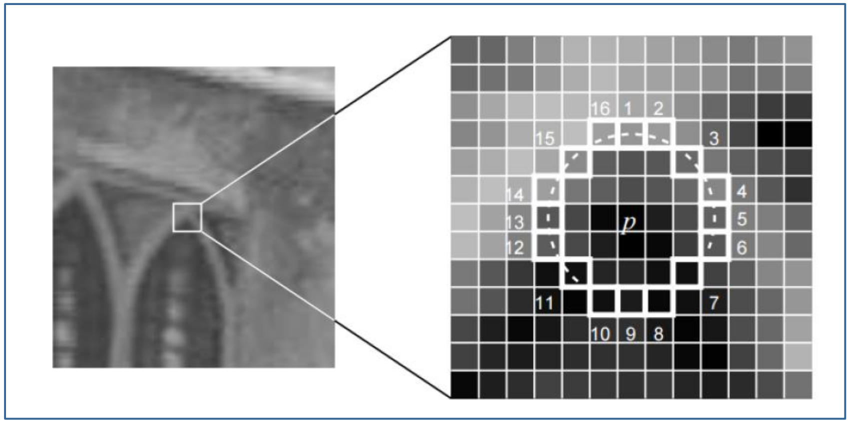
- Feature
  - Keypoint: position, size, angle, score, etc.
  - Descriptor: encode the surrounding image information

- Examples:
  - SIFT
  - SURF
  - ORB
  - etc (see OpenCV's features2d module)

features2d. 2D Features Framev

- Feature Detection and Description
  - FAST
  - MSER
  - MSER::MSER
  - MSER::operator()
  - ORB
  - ORB::ORB
  - ORB::operator()
  - BRISK
  - BRISK::BRISK
  - BRISK::BRISK
  - BRISK::operator()
  - FREAK
  - FREAK::FREAK
  - FREAK::selectPairs
- Common Interfaces of Feature Detectors
  - KeyPoint
  - KeyPoint::KeyPoint
  - FeatureDetector
  - FeatureDetector::detect
  - FeatureDetector::create
  - FastFeatureDetector
  - GoodFeaturesToTrackDetector
  - MserFeatureDetector
  - StarFeatureDetector
  - DenseFeatureDetector
  - SimpleBlobDetector
  - GridAdaptedFeatureDetector
  - PyramidAdaptedFeatureDetector
  - DynamicAdaptedFeatureDetector
  - DynamicAdaptedFeatureDetector::DynamicAdap

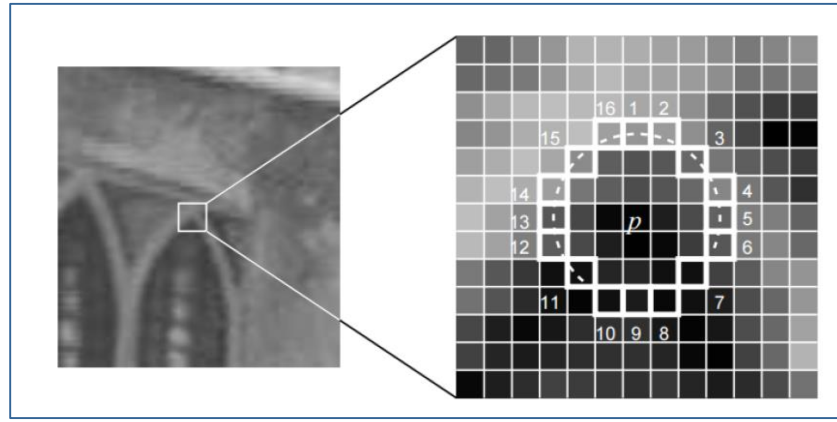
1. Feature Extraction and Matching



- Take ORB as an example
- ORB
  - Keypoint: Oriented FAST
  - Descriptor: Steer BRIEF

- FAST keypoint
  - P is a corner if we have continuous n points whose image intensity is larger/smaller than p over a threshold
  - Called FAST-n, e.g., FAST-12, FAST-10, FAST-9 …

1. Feature Extraction and Matching



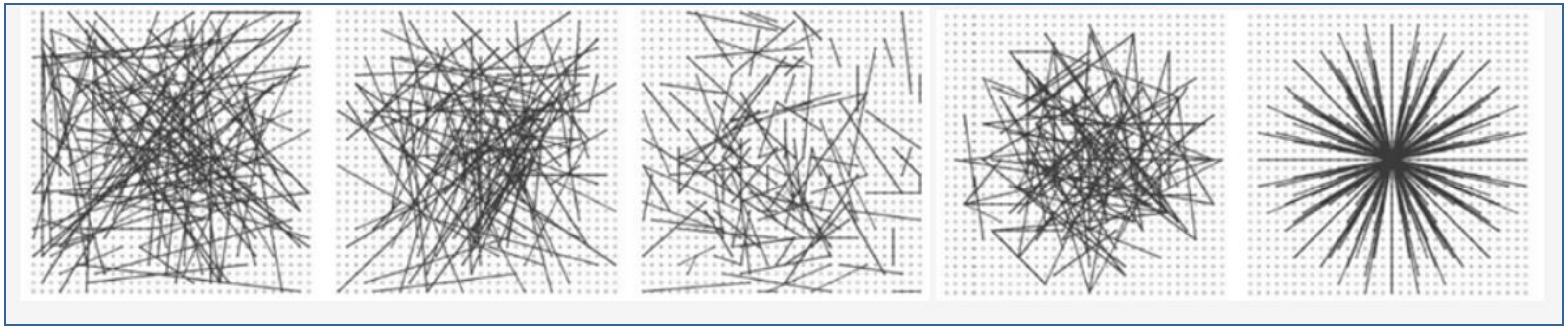
- Oriented FAST
  - Compute an angle in the FAST
- In a image patch B, define its moment as:

$$m_{pq} = \sum_{x,y \in B} x^p y^q I(x,y), \{p,q\} \in \{0,1\}$$

- Find the centroid of the patch: $\left(\frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}}\right)$
- And compute the angle: $\theta = \arctan(m_{01}/m_{10})$

1. Feature Extraction and Matching

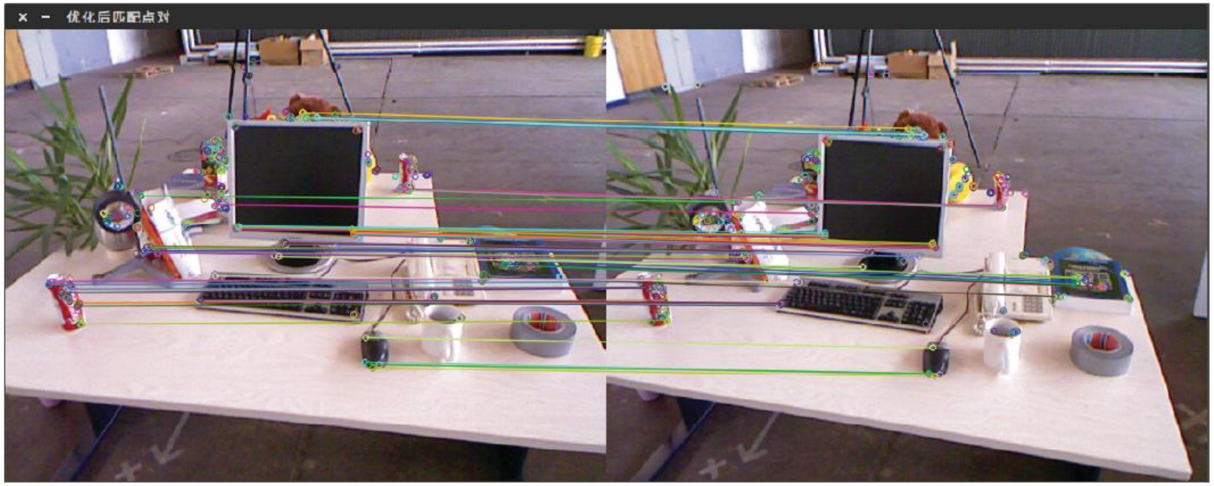
- BRIEF descriptor
  - Binary Robust Independent Elementary Features (BRIEF)
  - BRIEF-n: Compare n pairs of pixels around the keypoint
    - The pairs are chosen randomly or with a certain pattern



- ORB use steer (rotated) BRIEF
  - Rotate the BRIEF pattern according to the precomputed angle

1. Feature Extraction and Matching

- Feature matching
  - Compute the data association according the descriptor distance
  - Brute-force matching: compare each pairs of descriptors
  - FLANN: Fast approximate nearest neighbor
  - For binary descriptors like BRIEF and ORB, use Hamming distance:
    - Hamming(x,y) = number of different coefficients

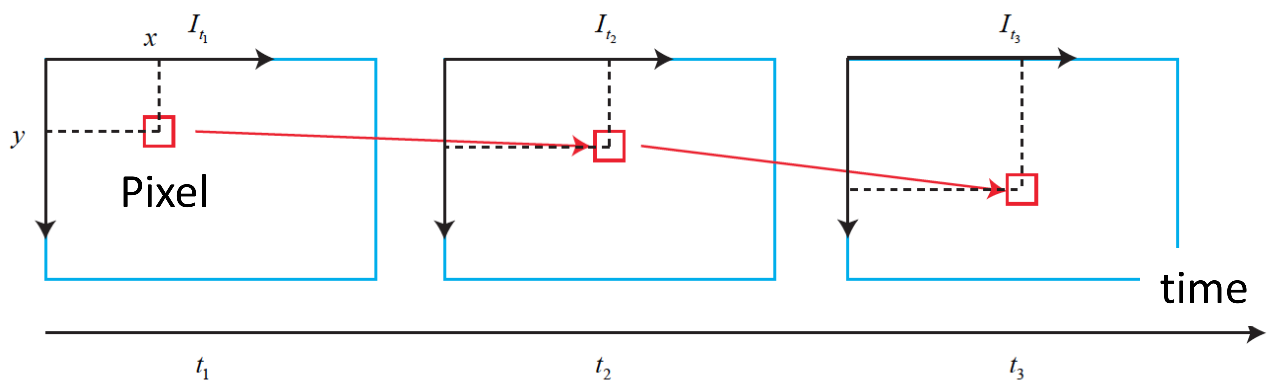


Contents

- Feature extraction and matching
- Optical Flow
- Pose estimation approaches
- Direct method

2. Optical Flow

- Optical Flow
  - Estimate the motion of pixels in continuous images
  - Sparse vs Dense Flow
    - Sparse: Lucas-Kanade (LK) flow
    - Dense: Horn-Schunck (HS) flow
  - Can be used to find corresponding pixels in images



2. Optical Flow

- How to estimate optical flow?
- Assume at time t we have a pixel at x,y, then its intensity is: $\boldsymbol{I}(x, y, t)$.
- At t+dt, it moves to x+dx, y+dy, and the intensity is: $\boldsymbol{I}(x + \mathrm{d}x, y + \mathrm{d}y, t + \mathrm{d}t)$

- Brightness constancy assumption:

$$\boldsymbol{I}(x + \mathrm{d}x, y + \mathrm{d}y, t + \mathrm{d}t) = \boldsymbol{I}(x, y, t).$$

- Note this is really a strong and ideal assumption since brightness can be changed by highlight/shadow/occlusion/material/exposure and will not hold any more

2. Optical Flow

- With brightness constancy, we expand the assumption:

$$\boxed{\boldsymbol{I}(x + \mathrm{d}x, y + \mathrm{d}y, t + \mathrm{d}t) = \boldsymbol{I}(x, y, t).}$$

$$\boldsymbol{I}\left(x + \mathrm{d}x, y + \mathrm{d}y, t + \mathrm{d}t\right) \approx \boldsymbol{I}\left(x, y, t\right) + \frac{\partial \boldsymbol{I}}{\partial x}\mathrm{d}x + \frac{\partial \boldsymbol{I}}{\partial y}\mathrm{d}y + \frac{\partial \boldsymbol{I}}{\partial t}\mathrm{d}t.$$

- And obtain:

Gradient with time

$$\frac{\partial \boldsymbol{I}}{\partial x}\mathrm{d}x + \frac{\partial \boldsymbol{I}}{\partial y}\mathrm{d}y + \frac{\partial \boldsymbol{I}}{\partial t}\mathrm{d}t = 0.$$

$$\frac{\partial \boldsymbol{I}}{\partial x}\frac{\mathrm{d}x}{\mathrm{d}t} + \frac{\partial \boldsymbol{I}}{\partial y}\frac{\mathrm{d}y}{\mathrm{d}t} = -\frac{\partial \boldsymbol{I}}{\partial t}.$$

Gradient on x-axis

Gradient on y-axis

- Our object: compute dx/dt and dy/dt

2. Optical Flow

$$\frac{\partial \boldsymbol{I}}{\partial x}\frac{\mathrm{d}x}{\mathrm{d}t} + \frac{\partial \boldsymbol{I}}{\partial y}\frac{\mathrm{d}y}{\mathrm{d}t} = -\frac{\partial \boldsymbol{I}}{\partial t}.$$

- However it is a underdetermined linear equation
  - 2 unknowns and 1 equation
  - We need extra constraints: assume brightness constancy in a small window of $w \times w$ patch

$$\begin{bmatrix} \boldsymbol{I}_x & \boldsymbol{I}_y \end{bmatrix}_k \begin{bmatrix} u \\ v \end{bmatrix} = -\boldsymbol{I}_{tk}, \quad k = 1, \ldots, w^2.$$

  - Then we get an overdetermined equation and solve it by linear least square (or Moore-Penrose inverse of the coefficient matrix:

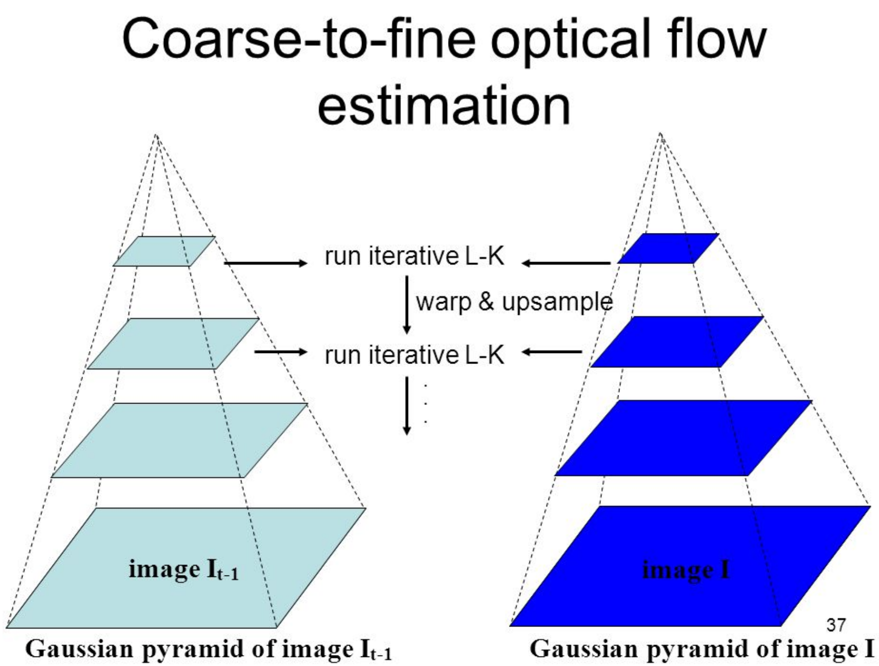
$$\boldsymbol{A} = \begin{bmatrix} [\boldsymbol{I}_x, \boldsymbol{I}_y]_1 \\ \vdots \\ [\boldsymbol{I}_x, \boldsymbol{I}_y]_k \end{bmatrix}, \boldsymbol{b} = \begin{bmatrix} \boldsymbol{I}_{t1} \\ \vdots \\ \boldsymbol{I}_{tk} \end{bmatrix}.$$

$$\begin{bmatrix} u \\ v \end{bmatrix}^* = -\left(\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}\right)^{-1}\boldsymbol{A}^{\mathrm{T}}\boldsymbol{b}.$$

2. Optical Flow

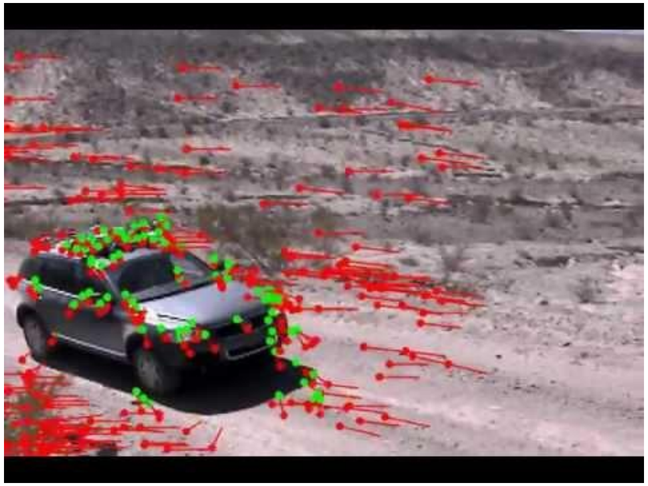
$$\frac{\partial \boldsymbol{I}}{\partial x}\frac{\mathrm{d}x}{\mathrm{d}t} + \frac{\partial \boldsymbol{I}}{\partial y}\frac{\mathrm{d}y}{\mathrm{d}t} = -\frac{\partial \boldsymbol{I}}{\partial t}.$$

- The solution of LK flow depends on the image gradient
  - Which is not smooth and can have dramatic changes
  - Multi-level optical flow: from coarse to fine



2. Optical Flow

- Some notes on LK-flow
  - We can also use non-linear optimization tools (Gauss-Newton, L-M, etc.) to solve the optical flow iteratively
  - Optical flow can be used to track the motion of the corners in videos
  - After obtaining the points, motion estimation step will be same as feature methods
  - Need to wrap the patches if the motion is not pure translation



Contents

- Feature extraction and matching
- Optical Flow
- Pose estimation approaches
- Direct method

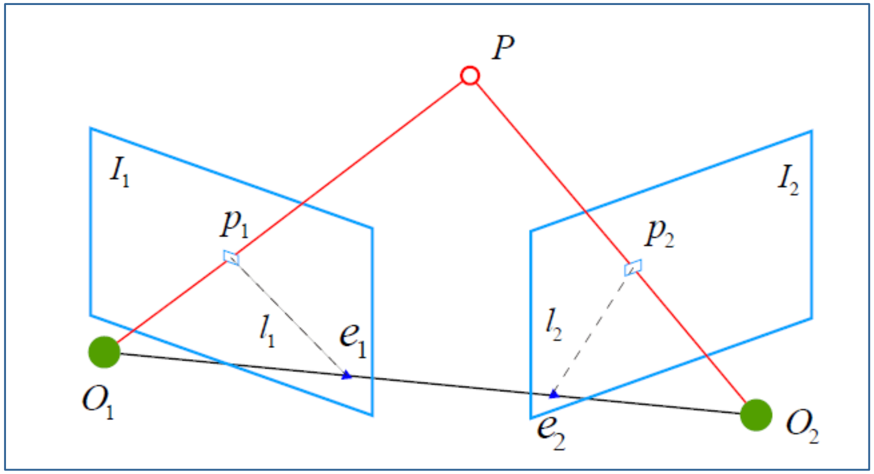
3. Pose estimation approaches

- After obtaining the corresponding feature points we can estimate the camera motion
- In practice we have several cases:
  - 2D-2D: if we only have two images
  - 3D-2D: if we have a pre-built scene and an image
  - 3D-3D: if we have two RGB-D image pairs or if we want to align a model to a scene

- 2D-2D: epipolar geometry
- 3D-2D: Perspective-n-Points (PnP)
- 3D-3D: iterative closest points (ICP)

3. Pose estimation approaches

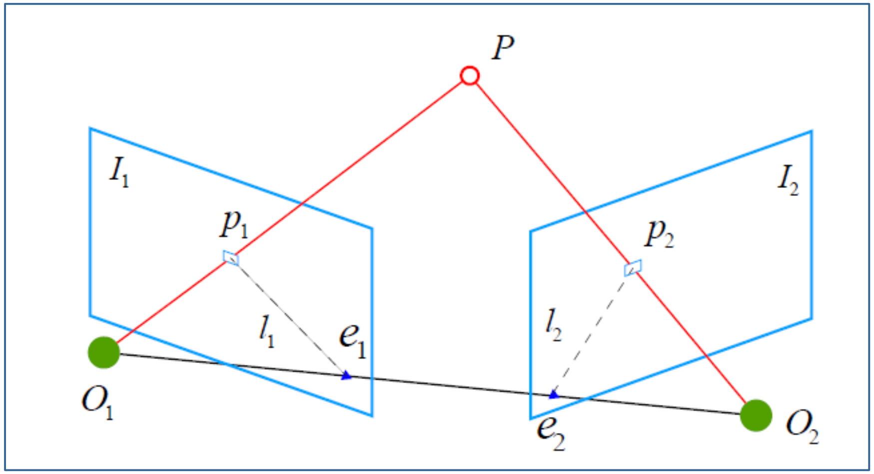
- 2D-2D
  - 3D point P (unknown)
  - Projections: p1, p2
  - Images I1, I2
  - Transform: T12



- Epipoles: projection of O_1, O_2 -> e_1, e_2
- Epipolar line: O_1P projected in image 2 e_2p_2 and vise vesa

- Our purpose: estimate T12(T21) given p1 and p2

3. Pose estimation approaches



- Epipolar geometry
- Assume the point P at $P = [X, Y, Z]^T$.
- Projection model:

$$s_1 p_1 = KP, \quad s_2 p_2 = K(RP + t).$$

- Use unit plane homogenous coordinates (to remove the intrinsics):

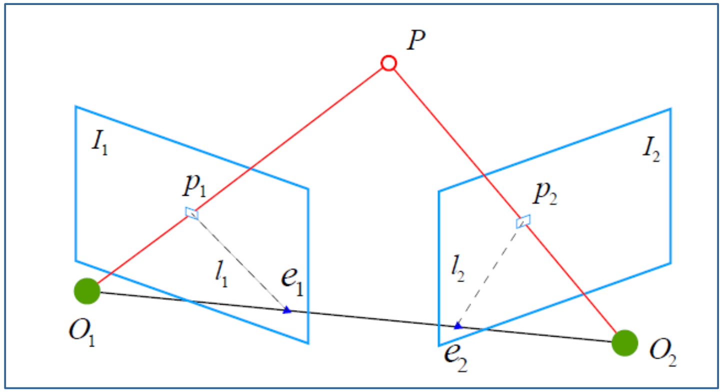
$$x_1 = K^{-1} p_1, \quad x_2 = K^{-1} p_2. \quad \Rightarrow \quad x_2 = Rx_1 + t.$$

- Left multiplied by $t^\wedge$: $t^\wedge x_2 = t^\wedge R x_1.$
- Then left multiplied by x_2^T : $x_2^T t^\wedge x_2 = x_2^T t^\wedge R x_1.$

- This should be zero: $x_2^T t^\wedge R x_1 = 0.$

3. Pose estimation approaches

- Epipolar constraints: $x_2^T t^\wedge R x_1 = 0.$ or $p_2^T K^{-T} t^\wedge R K^{-1} p_1 = 0.$
- Which meas O1, O2 and P are in the same plane
- Define: $E = t^\wedge R, \quad F = K^{-T} E K^{-1}, \quad x_2^T E x_1 = p_2^T F p_1 = 0.$
  - Essential/Fundamental matrix



- Estimate the pose in two steps:
  - Estimate Essential/Fundamental matrix
  - Decompose the E/F to get R,t

3. Pose estimation approaches

- Estimate essential matrix (eight-point algorithm)
  - Treat E as an ordinary matrix, then one point gives us:

$$\begin{pmatrix} u_1, v_1, 1 \end{pmatrix} \begin{pmatrix} e_1 & e_2 & e_3 \\ e_4 & e_5 & e_6 \\ e_7 & e_8 & e_9 \end{pmatrix} \begin{pmatrix} u_2 \\ v_2 \\ 1 \end{pmatrix} = 0.$$

  - Rewrite it as: $[u_1 u_2, u_1 v_2, u_1, v_1 u_2, v_1 v_2, u_2, v_2, 1] \cdot e = 0.$
  - Then we need at least eight points to solve this linear equation (because E is homogenous and can be multiplied by any non-zero factor)

$$\begin{pmatrix} u_1^1 u_2^1 & u_1^1 v_2^1 & u_1^1 & v_1^1 u_2^1 & v_1^1 v_2^1 & v_1^1 & u_2^1 & v_2^1 & 1 \\ u_1^2 u_2^2 & u_1^2 v_2^2 & u_1^2 & v_1^2 u_2^2 & v_1^2 v_2^2 & v_1^2 & u_2^2 & v_2^2 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \\ u_1^8 u_2^8 & u_1^8 v_2^8 & u_1^8 & v_1^8 u_2^8 & v_1^8 v_2^8 & v_1^8 & u_2^8 & v_2^8 & 1 \end{pmatrix} \begin{pmatrix} e_1 \\ e_2 \\ e_3 \\ e_4 \\ e_5 \\ e_6 \\ e_7 \\ e_8 \\ e_9 \end{pmatrix} = 0.$$

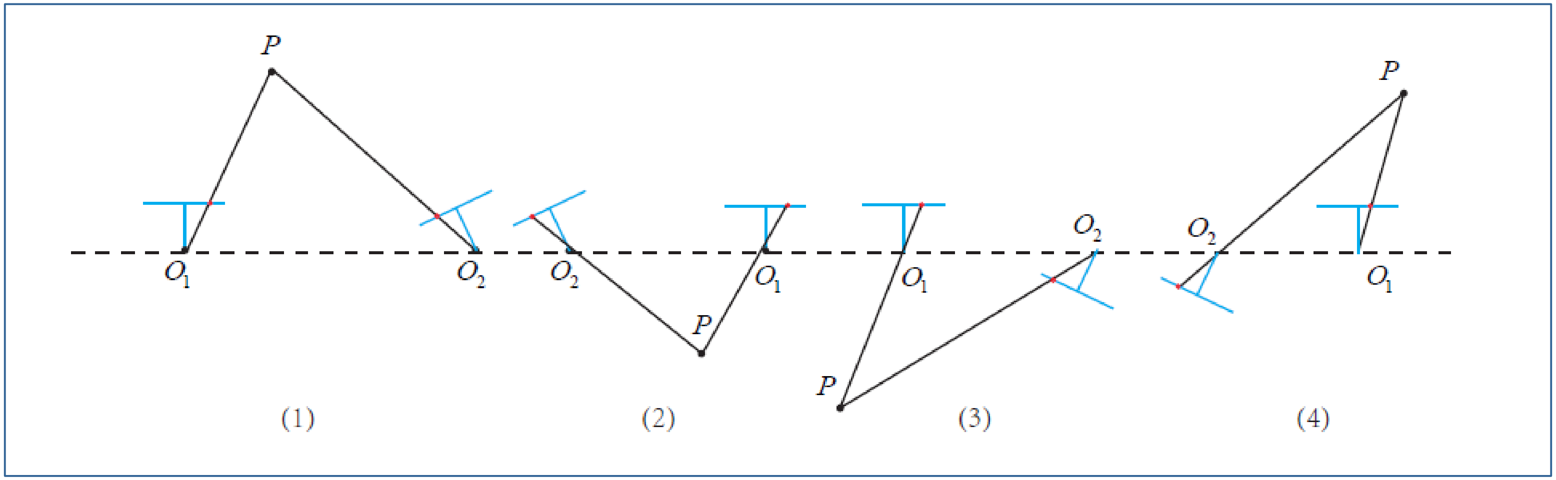
3. Pose estimation approaches

- From Essential to R,t: use the SVD method

$$E = U\Sigma V^T,$$

$$t_1^{\wedge} = UR_Z(\tfrac{\pi}{2})\Sigma U^T, \quad R_1 = UR_Z^T(\tfrac{\pi}{2})V^T$$
$$t_2^{\wedge} = UR_Z(-\tfrac{\pi}{2})\Sigma U^T, \quad R_2 = UR_Z^T(-\tfrac{\pi}{2})V^T.$$

- Four possible solutions but only one of them has positive depth values



3. Pose estimation approaches

- During the SVD we can also:
  - Take $E = U\mathrm{diag}(\frac{\sigma_1+\sigma_2}{2}, \frac{\sigma_1+\sigma_2}{2}, 0)V^T.$ since essential matrix requires its singular value as $\sigma, \sigma, 0$
- And because DoF of E is only five (3 rot+3 trans -1 scale), we can also solve it using only 5 points
  - Called five point algorithm [1]
- More than eight points:
  - RANSAC or least square

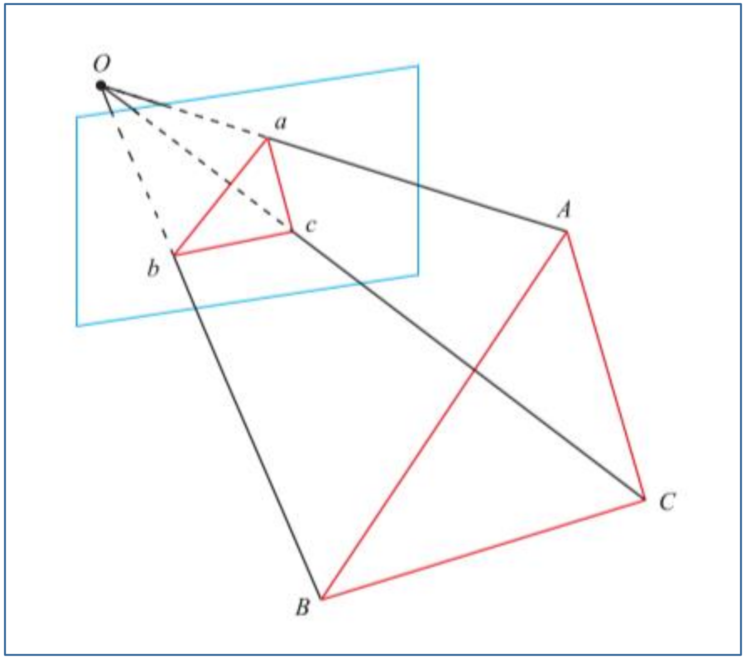
[1] Nistér D. An efficient solution to the five-point relative pose problem[J]. IEEE transactions on pattern analysis and machine intelligence, 2004, 26(6): 756-770.

3. Pose estimation approaches

- Eight-point algorithm can be used in the initialization step of monocular SLAM
- Some notes on Essential matrix
  - Scale is undetermined: we can normalize t or the mean depth of the scene
  - Pure rotation problem: when t=0 and t^R should be zero, and E cannot be decomposed
  - When the eight points are on the same plane, the problem will be degenerated and we will use Homography matrix to solve the initialization problem

3. Pose estimation approaches

- 3D-2D: Perspective-n-Points
- Given n 3D points and their projections, estimate the camera pose
- Methods: linear algebra or nonlinear optimization
- Linear algebra:
  - DLT
  - P3P
  - EPnP/UPnP/etc.
  - You can just call cv::SolvePnP
  - The methods can be chosen by giving different parameters



- Nonlinear:
  - Minimizing the reprojection error
  - Shown in ex3.3

3. Pose estimation approaches

- 3D-3D: ICP
- Given two pairs of 3D points, estimate the rotation and translation
- The points can be pre-matched or not matched
- If we have the matches, then the problem has analytical solution, otherwise we don’t
- Assume we have two point sets: $\boldsymbol{P} = \{\boldsymbol{p}_1, \cdots, \boldsymbol{p}_n\}, \quad \boldsymbol{P}' = \{\boldsymbol{p}'_1, \cdots, \boldsymbol{p}'_n\},$
- And the motion is: $\forall i, \boldsymbol{p}_i = \boldsymbol{R}\boldsymbol{p}'_i + \boldsymbol{t}.$
- Define the error: $\boldsymbol{e}_i = \boldsymbol{p}_i - (\boldsymbol{R}\boldsymbol{p}'_i + \boldsymbol{t}).$
- And the least square will be:

$$\min_{\boldsymbol{R},\boldsymbol{t}} J = \frac{1}{2} \sum_{i=1}^{n} \| (\boldsymbol{p}_i - (\boldsymbol{R}\boldsymbol{p}_i' + \boldsymbol{t})) \|_2^2.$$

3. Pose estimation approaches

- Some derivation
- Define the centroids: $\boldsymbol{p} = \frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{p}_i), \quad \boldsymbol{p}' = \frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{p}'_i).$
- And rewrite the objective function:

$$\begin{aligned}
\frac{1}{2}\sum_{i=1}^{n}\left\|\boldsymbol{p}_i - (\boldsymbol{R}\boldsymbol{p}_i{}' + \boldsymbol{t})\right\|^2 &= \frac{1}{2}\sum_{i=1}^{n}\left\|\boldsymbol{p}_i - \boldsymbol{R}\boldsymbol{p}_i{}' - \boldsymbol{t} - \boldsymbol{p} + \boldsymbol{R}\boldsymbol{p}' + \boldsymbol{p} - \boldsymbol{R}\boldsymbol{p}'\right\|^2 \\
&= \frac{1}{2}\sum_{i=1}^{n}\left\|(\boldsymbol{p}_i - \boldsymbol{p} - \boldsymbol{R}(\boldsymbol{p}_i{}' - \boldsymbol{p}')) + (\boldsymbol{p} - \boldsymbol{R}\boldsymbol{p}' - \boldsymbol{t})\right\|^2 \\
&= \frac{1}{2}\sum_{i=1}^{n}\left(\left\|\boldsymbol{p}_i - \boldsymbol{p} - \boldsymbol{R}(\boldsymbol{p}_i{}' - \boldsymbol{p}')\right\|^2 + \left\|\boldsymbol{p} - \boldsymbol{R}\boldsymbol{p}' - \boldsymbol{t}\right\|^2 + \right. \\
&\qquad \boxed{2(\boldsymbol{p}_i - \boldsymbol{p} - \boldsymbol{R}(\boldsymbol{p}_i{}' - \boldsymbol{p}'))^T(\boldsymbol{p} - \boldsymbol{R}\boldsymbol{p}' - \boldsymbol{t})).}
\end{aligned}$$

This part is zero if you write it out

3. Pose estimation approaches

- So the objective function is simplified as:

$$\min_{\boldsymbol{R},\boldsymbol{t}} J = \frac{1}{2}\sum_{i=1}^{n} \left\|\boldsymbol{p}_i - \boldsymbol{p} - \boldsymbol{R}\left(\boldsymbol{p}_i{}' - \boldsymbol{p}'\right)\right\|^2 + \left\|\boldsymbol{p} - \boldsymbol{R}\boldsymbol{p}' - \boldsymbol{t}\right\|^2.$$

- We can just minimize the first part, and choose a t to set the second part to zero
- How to solve the first part?
- Remove the centroid: $\boldsymbol{q}_i = \boldsymbol{p}_i - \boldsymbol{p}, \quad \boldsymbol{q}_i' = \boldsymbol{p}_i' - \boldsymbol{p}'.$
- The problem becomes:

$$\boldsymbol{R}^* = \arg\min_{\boldsymbol{R}} \frac{1}{2}\sum_{i=1}^{n} \left\|\boldsymbol{q}_i - \boldsymbol{R}\boldsymbol{q}_i'\right\|^2.$$

3. Pose estimation approaches

- Some derivation:

$$\frac{1}{2}\sum_{i=1}^{n}\|\boldsymbol{q}_i - \boldsymbol{R}\boldsymbol{q}'_i\|^2 = \frac{1}{2}\sum_{i=1}^{n}\boldsymbol{q}_i^{\mathrm{T}}\boldsymbol{q}_i + \boldsymbol{q}_i'^{\mathrm{T}}\boldsymbol{R}^{\mathrm{T}}\boldsymbol{R}\boldsymbol{q}'_i - 2\boldsymbol{q}_i^{\mathrm{T}}\boldsymbol{R}\boldsymbol{q}'_i.$$

- It’s only related to the last part:

$$\sum_{i=1}^{n} -\boldsymbol{q}_i^{\mathrm{T}}\boldsymbol{R}\boldsymbol{q}'_i = \sum_{i=1}^{n} -\mathrm{tr}\left(\boldsymbol{R}\boldsymbol{q}'_i\boldsymbol{q}_i^{\mathrm{T}}\right) = -\mathrm{tr}\left(\boldsymbol{R}\sum_{i=1}^{n}\boldsymbol{q}'_i\boldsymbol{q}_i^{\mathrm{T}}\right).$$

- Solve it by SVD:

$$\boldsymbol{W} = \sum_{i=1}^{n}\boldsymbol{q}_i\boldsymbol{q}_i'^{\mathrm{T}}. \qquad \boldsymbol{W} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\mathrm{T}}. \qquad \boldsymbol{R} = \boldsymbol{U}\boldsymbol{V}^{\mathrm{T}}.$$

3. Pose estimation approaches

- It can be proven that if we find a solution, then this solution is the global minimal
- Otherwise, in some special degenerated cases, we cannot find a solution

- If we don’t have matches, then assume the closes points are matches and solve this ICP iteratively
- Also, note in the RGB-D case, you can use ICP and PnP separately or put them together into a Bundle Adjustment

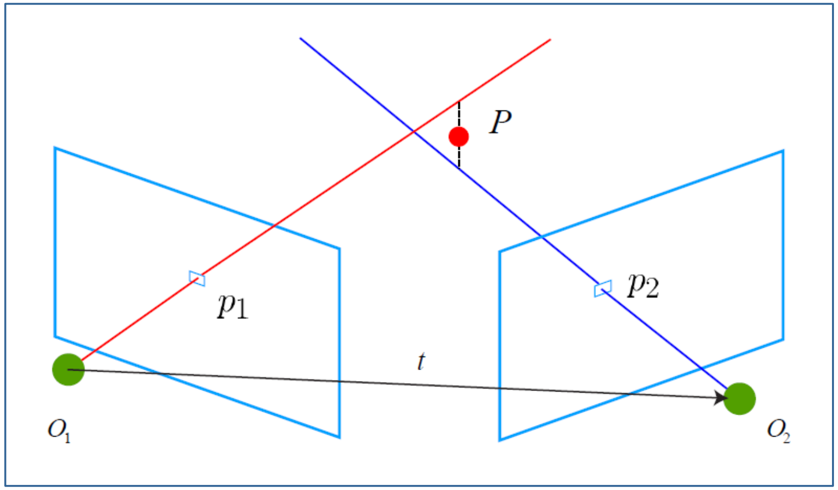
3. Pose estimation approaches

- Triangulation
- Given the motion and the pixels, estimate the 3D point position
- From geometry we know: $s_1 x_1 = s_2 R x_2 + t.$
- We can either solve s1 or s2, take s2 as an example
- Left multiply by $x_1^\wedge$:

$$s_1 x_1^\wedge x_1 = 0 = s_2 x_1^\wedge R x_2 + x_1^\wedge t.$$

  - Overdetermined linear equation
- We can also solve s1, s2 together:

$$[-Rx_2, x_1] \begin{bmatrix} s_1 \\ s_2 \end{bmatrix} = t$$



3. Pose estimation approaches

- Summary
  - We can get point pairs by feature matching or optical flow
  - Estimate the pose in some cases:
    - 2D-2D: estimate the essential/fundamental/Homography matrix and then solve R,t from them
    - 3D-2D: PnP, linear and non-linear method
    - 3D-3D: ICP, also linear (SVD) method or nonlinear method

Contents

- Feature extraction and matching
- Optical Flow
- Pose estimation approaches
- Direct method

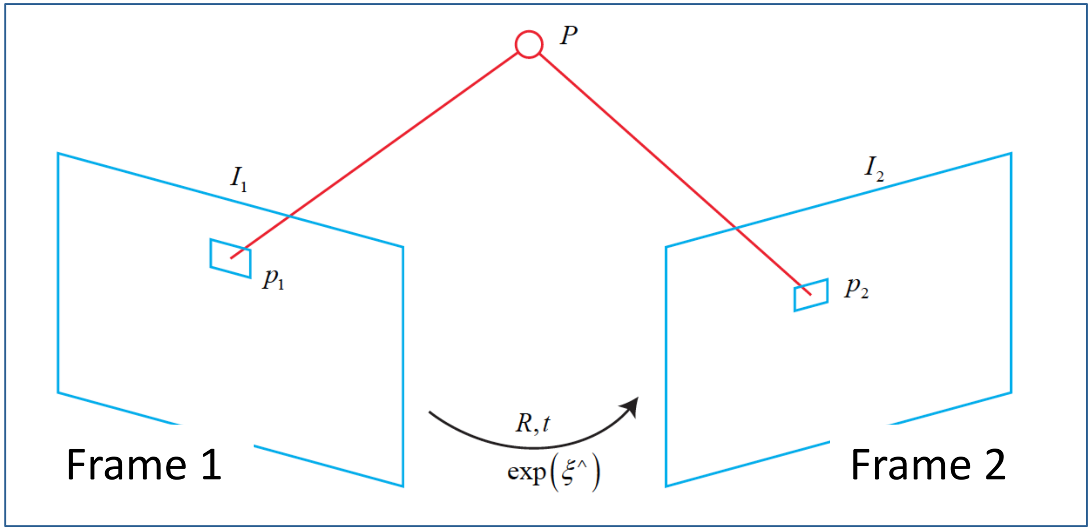
4. Direct Method

- Optical flow can estimate pixel's motion, but
  - Without considering the camera's projection model
- Direct method: put them together in an optimization problem

4. Direct Method

- Derivation of direct method
  - Assume we have two image and the motion is unknown (but with a initial guess)
  - We have a pixel at image 1, and know its depth
  - Then we can compute the projection using the initial guess of transform

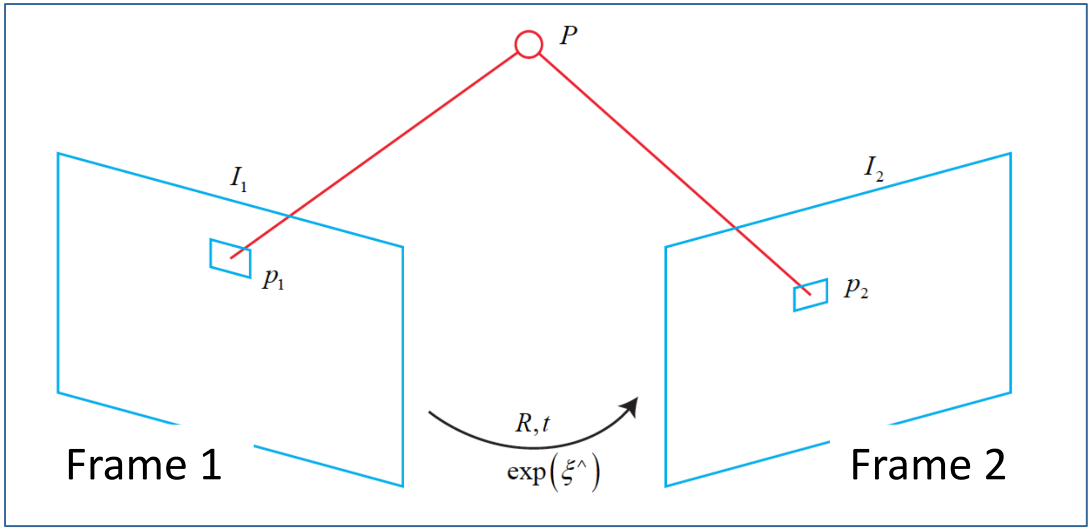
$$s_1 p_1 = KP$$
$$s_2 p_2 = K(RP + t)$$



4. Direct Method

- Brightness constancy assumption: $I_1(p_1) = I_2(p_2)$
- Brightness error: $e = I_1(p_1) - I_2(p_2)$
- Motion estimation by least square:

$$\min_{T} J(T) = \sum_{i=1}^{N} e_i^T e_i$$



4. Direct Method

- Jacobians by disturb model in SE(3)

$$
\begin{aligned}
e(\delta T \oplus T) &= I_1(p_1) - I_2\left(\frac{1}{Z_2} K \exp(\delta\xi^\wedge) T P\right) \\
&\approx I_1(p_1) - I_2\left(\frac{1}{Z_2} K (1 + \delta\xi^\wedge) T P\right) \\
&= I_1(p_1) - I_2\left(p_2 + \frac{1}{Z_2} K \delta\xi^\wedge T P\right)
\end{aligned}
$$

Define:

$$
q = \delta\xi^\wedge T P, \qquad u = \frac{1}{Z_2} K q
$$

Then:

$$
\begin{aligned}
e(\delta T \oplus T) &= I_1(p_1) - I_2(p_2 + u) \\
&\approx I_1(p_1) - I_2(p_2) - \frac{\partial I_2}{\partial u}\frac{\partial u}{\partial q}\frac{\partial q}{\partial \delta\xi}\delta\xi \\
&= e - \frac{\partial I_2}{\partial u}\frac{\partial u}{\partial q}\frac{\partial q}{\partial \delta\xi}\delta\xi
\end{aligned}
$$

4. Direct Method

- So Jacobian has three parts:
  - First $\frac{\partial I_2}{\partial u}$ is the image gradients in the second image
  - And the second the third part is same as geometric reprojection error:

$$\frac{\partial \boldsymbol{u}}{\partial \delta \boldsymbol{\xi}} = \begin{bmatrix} \frac{f_x}{Z} & 0 & -\frac{f_x X}{Z^2} & -\frac{f_x XY}{Z^2} & f_x + \frac{f_x X^2}{Z^2} & -\frac{f_x Y}{Z} \\ 0 & \frac{f_y}{Z} & -\frac{f_y Y}{Z^2} & -f_y - \frac{f_y Y^2}{Z^2} & \frac{f_y XY}{Z^2} & \frac{f_y X}{Z} \end{bmatrix}.$$

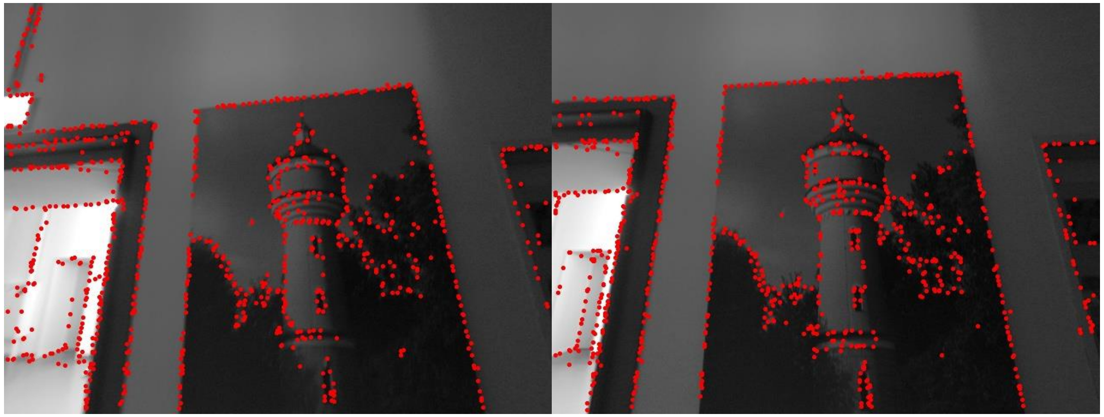
  - So the overall Jacobian will be:

$$\boldsymbol{J} = -\frac{\partial \boldsymbol{I}_2}{\partial \boldsymbol{u}} \frac{\partial \boldsymbol{u}}{\partial \delta \boldsymbol{\xi}}.$$

- So we need to choose points that have non-zero gradients, otherwise they won’t contribute to the pose estimation

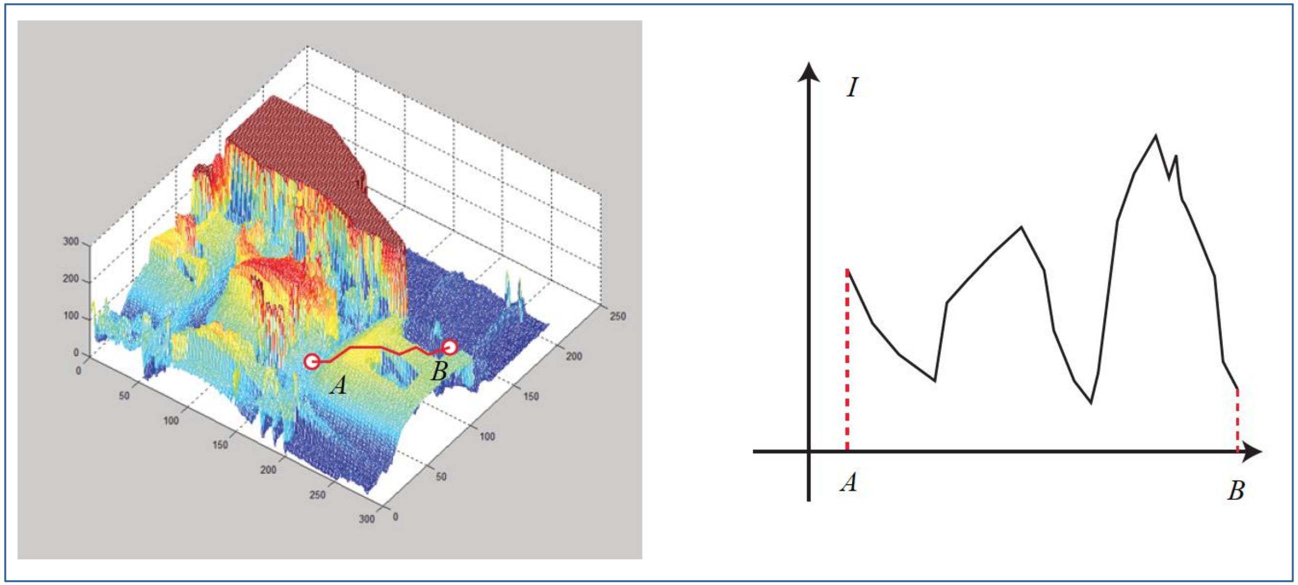
4. Direct Method

- Some notes on direct method
  - We need to know the depth in the first image (or the reference image), which can be obtained from an RGB-D camera or pre-built structures
  - Don’t need explicit matched points, what we only need is image gradients (thus we can choose edges and smooth areas)
  - Are able to build dense or semi-dense maps
  - Can be also extended to multi-level and lead to a coarse-to-fine direct method



4. Direct Method

- Direct method is also affected by image gradients
  - We usually can not control or predict the image data
  - So if the motion is too large, we can not guarantee the cost function is always decreasing during the path to the correct point
  - So direct method is only suitable for smooth motion (or high speed camera)



Any Questions?