Perception for Autonomous Systems

Lecture 5 - Visual Odometry (12/04/2021)

Outline/Content:

- Orientation (Attitude) Representations
- Relative Pose Estimation
  - 3D registration
  - PnP
  - Least Squares - SVD (Singular Value Decomposition)
- Visual Odometry
  - 3D-3D
  - 3D-2D
  - 2D-2D
- Local Bundle Adjustment
- Visual Inertial Odometry (VIO)
  - Loosely Coupled EKF (Extended Kalman Filter)
  - Tightly Coupled EKF

> Reading Material:
>
> - Scaramuzza, D. and Fraundorfer, F., 2011. Visual odometry [tutorial]. IEEE robotics & automation magazine, 18(4), pp.80-92. Fraundorfer, F. and Scaramuzza, D., 2012. Visual odometry: Part ii: Matching, robustness, optimization, and applications. IEEE Robotics & Automation Magazine, 19(2), pp.78-90. Solution of P3P Problem, by Tomas Wernere on Intelligent Robots and Systems (IROS), Nice, France, September 22-26, 2008.

What is Visual Odometry and what it is not? Lecture Slides - University of Toronto

Definition 1: Visual Odometry (VO) concerns the use of cameras to estimate the Pose (position and orientation) of a mobile system, by observing the apparent motion of the "static" world.

Definition 2: Visual Odometry is the process of estimating the motion of a camera in real-time using successive images.

Definition 3: VO is the process of incrementally estimating the pose (position and orientation) of the vehicle by examining the changes that motion induces on the images of its onboard cameras.

VO Facts/Assumptions

Facts:

- VO does not provide a map of the environment neither it uses previous states of the world to improve it's accuracy (SLAM).
- It has been proposed as an alternative to wheel odometry
- VO estimations are usually combined with other sensors
  - GPS, IMU, Laser, Wheel odometry

    - VINS (Visual Search for Mobile Interface Design), VIO stands for Visual Inertial Odometry
- VO has had some extraordinary usages

Assumptions:

- Sufficient illumination is in the environment
- Enough texture to allow apparent motion to be extracted
- Sufficient scene overlap between consecutive frames
- VO assumes a static world where the only moving object is the mobile system
- It's accuracy is - assuming reasonable trajectories ~1%

Illustration of the VO Pipline

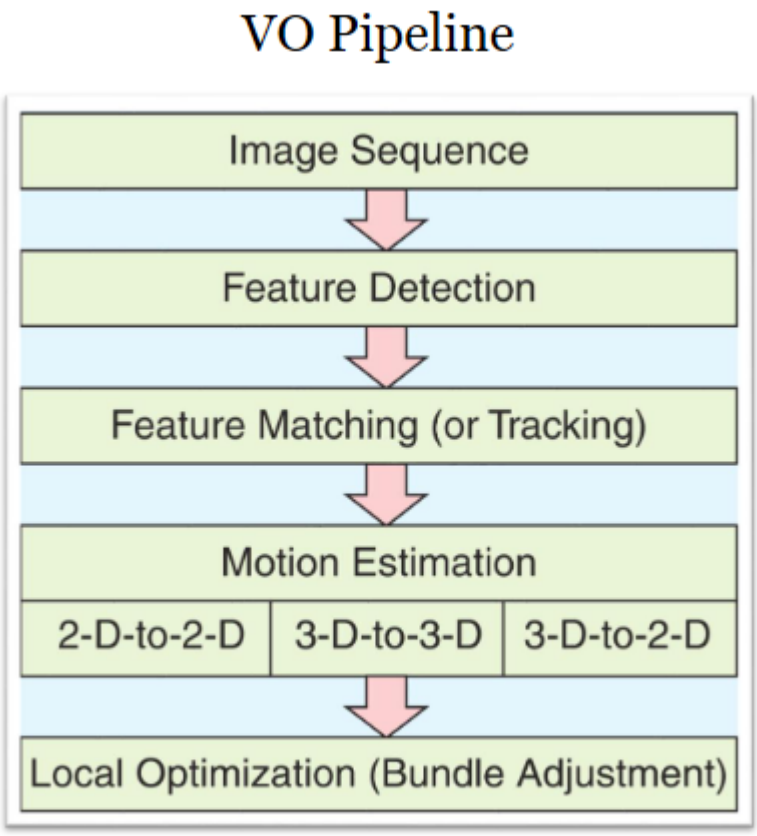
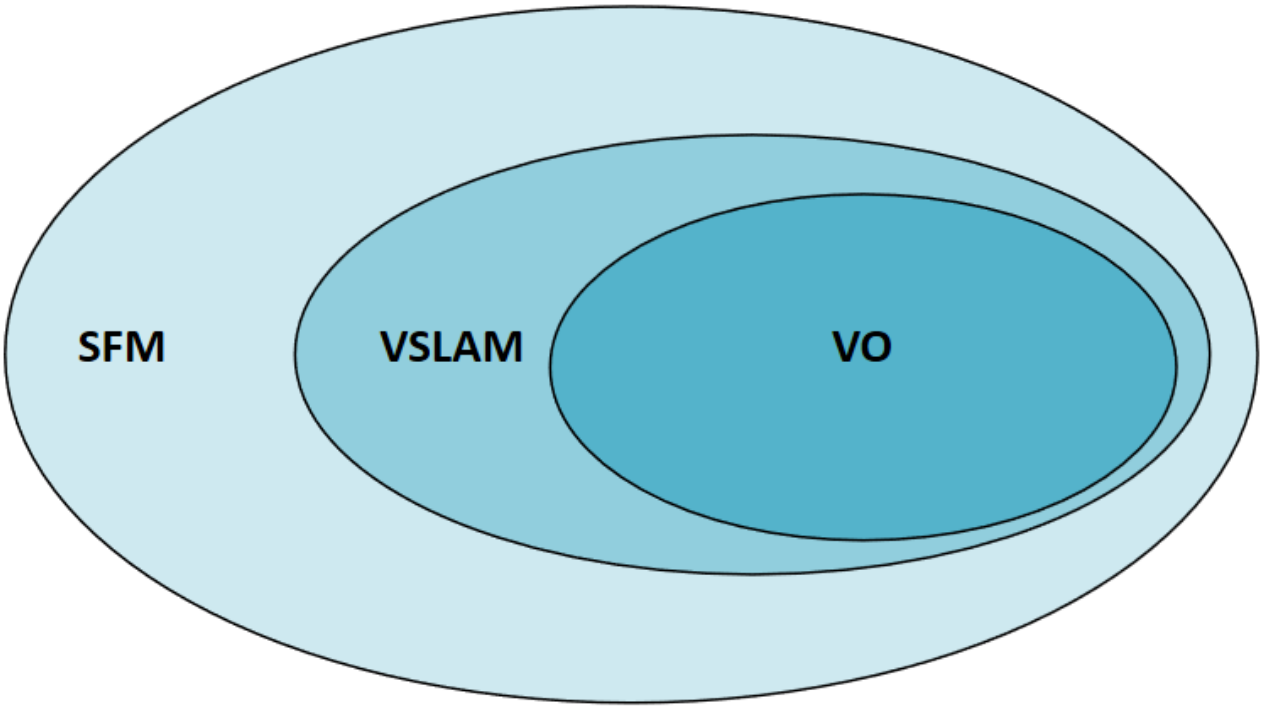


Image from Scaramuzza and Fraundorfer, 2011

**VO is neither SFM **(Structure from Motion) nor VSLAM (Visual SLAM) Lecture Slides - University of Zurich



SFM (Structure from Motion)

SFM is more general than VO and tackles the problem of 3D reconstruction and 6DOF pose estimation from unordered image sets

Structure from motion tries to solve also for the feature point as well:

- "Given Calibrated point projections of p=1...N points in camera(of frame) f=1... F(x_p^fm y_p^f)
- Find the rigid transformation R^Tt and the point's 3D position F X_p = (X_p, Y,_p, Z_p) which satisfies the projection equations

VO vs SFM

1. VO is a particular case of SFM
2. VO focuses on estimation the 3D motion of the camera sequentially (as a new frame arrives) and in real time
3. Terminology: sometimes SFM is used as a synonym of VO

V-Slam (Visual Slam)

Visual SLAM uses state estimation to exploit additional constraints and re-observations of the same areas(Loop-closures) to optimize the localization

VO vs Visual Slam

1. VO only aims to the local consistency of the trajectory
2. SLAM aims to the global consistency of the trajectory and of the map

3. VO can be used as a building block of SLAM
4. VO is SLAM before closing the loop
5. The choice between VO and V-SLAM depends on the tradeoff between performance and consistency, and simplicity in implementation
6. VO trades off consistency for real-time performance, without the need to keep track of all the previous history of the camera.

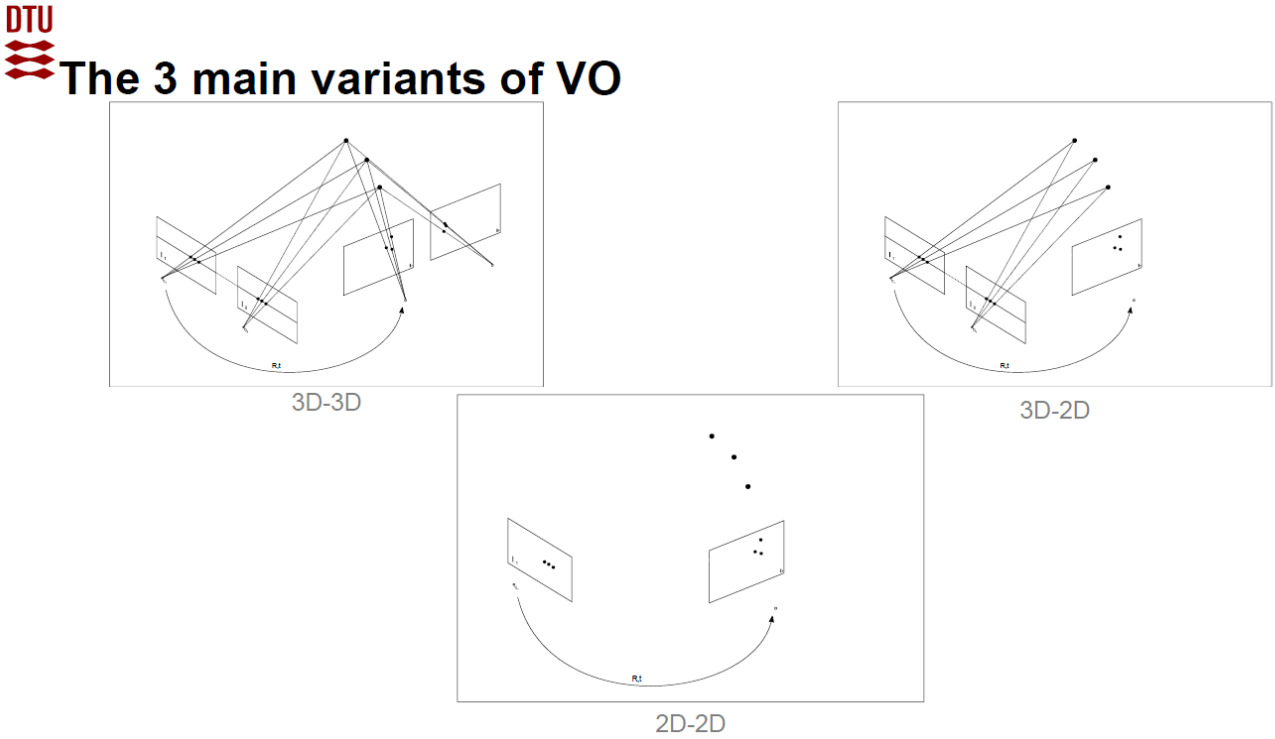
Applications of VO:

- Motion estimation for vehicles
- Driver assistance
- Autonomy
- Point-Cloud Mapping
- VO to estimate the motion of a lidar during the collection of a single scan
- Reduce rolling shutter effect

Challenges of VO:

- Robustness to lightning conditions
- Lack of features / non-overlapping images
- Without loop closure the estimate still drifts

The 3 main variants of VO

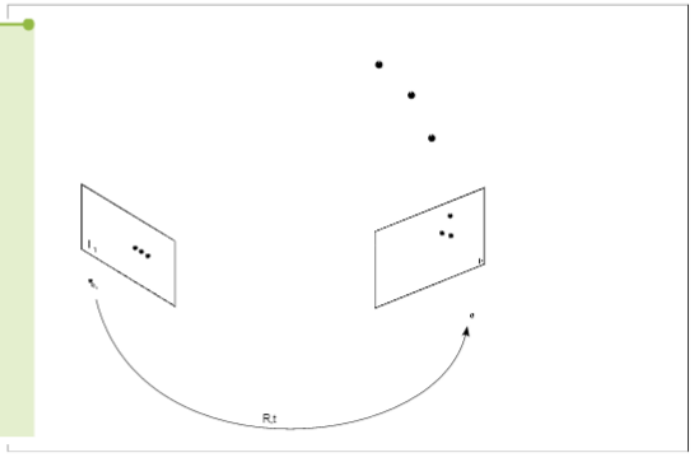


Visual Odometry 2D - to 2D

Visual Odometry 2D – to 2D

Algorithm 1. VO from 2-D-to-2-D correspondences.

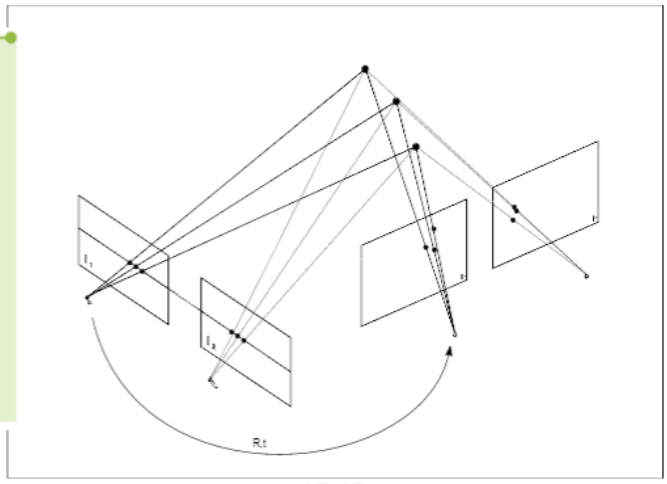
1) Capture new frame I_k
2) Extract and match features between I_{k-1} and I_k
3) Compute essential matrix for image pair I_{k-1}, I_k
4) Decompose essential matrix into R_k and t_k, and form T_k
5) Compute relative scale and rescale t_k accordingly
6) Concatenate transformation by computing $C_k = C_{k-1}T_k$
7) Repeat from 1).



Visual Odometry 3D - to 3D

Algorithm 2. VO from 3-D-to-3-D correspondences.

1) Capture two stereo image pairs $I_{l,k-1}, I_{r,k-1}$ and $I_{l,k}, I_{r,k}$
2) Extract and match features between $I_{l,k-1}$ and $I_{l,k}$
3) Triangulate matched features for each stereo pair
4) Compute T_k from 3-D features X_{k-1} and X_k
5) Concatenate transformation by computing $C_k = C_{k-1}T_k$
6) Repeat from 1).

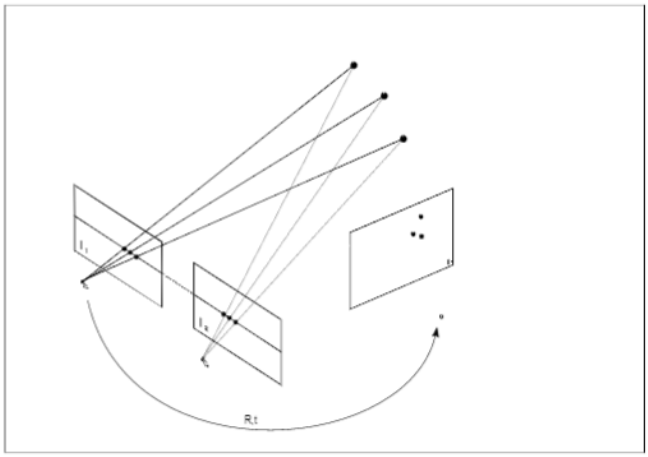


3D-3D

Visual Odometry 3D - to 2D

Algorithm 3. VO from 3-D-to-2-D Correspondences.

1) Do only once:
1.1) Capture two frames I_{k-2}, I_{k-1}
1.2) Extract and match features between them
1.3) Triangulate features from I_{k-2}, I_{k-1}
2) Do at each iteration:
2.1) Capture new frame I_k
2.2) Extract features and match with previous frame I_{k-1}
2.3) Compute camera pose (PnP) from 3-D-to-2-D matches
2.4) Triangulate all new feature matches between I_k and I_{k-1}
2.5) Iterate from 2.1).



3D-2D

Type of correspondences	Monocular	Stereo
2D-2D	X	X
3D-3D		X
3D-2D	X	X

	Structure (scene geometry)	Motion (camera geometry)	Measurements
Pose Estimation	known	estimate	3D to 2D correspondences
Triangulation	estimate	known	2D to 2D coorespondences
Reconstruction	estimate	estimate	2D to 2D coorespondences

> 3D-3D is inferior to the other approaches due to the fact that the ICP algorithm is being used, which doesn't guarantee to find optimal solutions.

Orientation

Rotation Matrix:

- Positives (The king of Orientation)
  - Unique, no gimbal lock
- Negatives
  - No pertubation, interpolation, unintuitive

Euler angles:

- Positives
  - Minimal representation, intuitive

- Negatives
    - Gimbal lock, non commutative

Axis Angles:

- Positives
    - No gimbal lock, minimal representation, nice for pertubation, linear mapping to rotation matrix
- Negatives
    - Non linear "scaling" w.r.t. magnitude
    - exponential coordinates

Quaternions

- Positives
    - all the axis angle ones, smooth trajectory
- Negatives
    - No direct geometric representation

Motion tracking

- Two main approaches
    - Feature based
    - Optical flow
    - Feature based:
        - Calculate feature on both images
        - Match among the features or
        - Calculate features
        - Do block Matching around our initial point (for small motion)

Feature-based methods

1. Extract & match features (+RANSAC)
2. Minimize **Reprojection error** minimization

$$T_{k,k-1} = \arg\min_T \sum_i \| \boldsymbol{u}'_i - \pi(\boldsymbol{p}_i) \|^2_\Sigma$$

✓ Large frame-to-frame motions

✓ Accuracy: Efficient optimization of structure and motion (Bundle Adjustment)

× Slow due to costly feature extraction and matching

× Matching Outliers (RANSAC)

Direct methods

1. Minimize **photometric error**

$$T_{k,k-1} = \arg\min_T \sum_i \| I_k(\boldsymbol{u}'_i) - I_{k-1}(\boldsymbol{u}_i) \|^2_\sigma$$

where $\boldsymbol{u}'_i = \pi\left(T \cdot (\pi^{-1}(\boldsymbol{u}_i) \cdot d)\right)$

[Jin,Favaro,Soatto'03] [Silveira, Malis, Rives, TRO'08], [Newcom
[Engel et al., ECCV'14], [Forster et al., ICRA'14]

✓ All information in the image can be exploited (precision, robustness)

✓ Increasing camera frame-rate reduces computational cost per frame

× Limited frame-to-frame motion

× Joint optimization of dense structure and motion too expensive

17

Motion tracking - Optical Flow Youtube Lecture Slides - Carnegie Mellon University

Definition: The most general (and challenging) version of motion estimation is to compute an independent estimate of motion at each pixel, which is generally known as optical (or optic) flow. This generally involves minimizing the brightness or color difference between corresponding pixels summed over the image.

Aim: Simply put, the goal of optical flow is basically to find a motion vector for each pixel within an image so that you then know in which direction each pixel is moving.

> Different surfaces can move in different directions

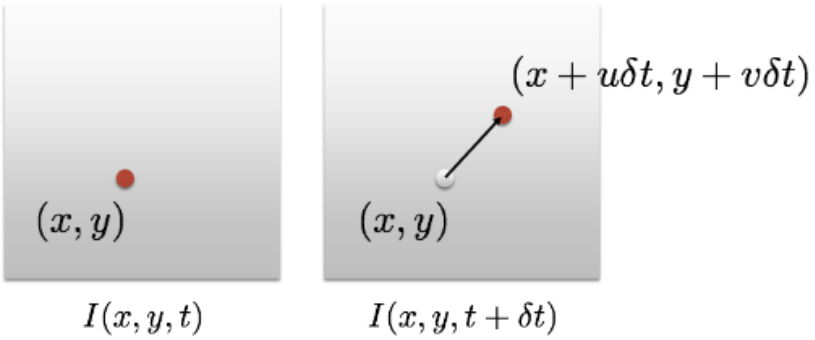
Assumption that need to be made:

- color/brightness constancy
  - otherwise, the optic flow algorithms would assume that the change of lightning is due to motion within the image.
- Small Motion, we analyze the motions for only very short changes in time (e.g. only 2 frames)
- enough features are needed (a glass sphere would not work (textureless, reflects a lot))

Application in which optical flow is used for:

- Image stabilization
- To see what global motion is happening

Small motion



Optical flow (velocities): (u, v) Displacement: $(\delta x, \delta y) = (u\delta t, v\delta t)$

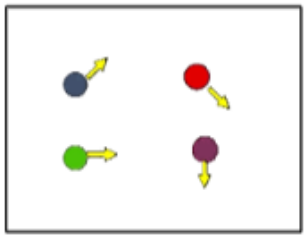
For a *really small space-time step*…

$$I(x + u\delta t, y + v\delta t, t + \delta t) = I(x, y, t)$$

… the brightness between two consecutive image frames is the same

> Terminology: I(x,y,t) is the image irradiance at time t at the image point (x,y). u(x,y) and v(x,y) are the optical flow functions.

- Estimate the apparent motion
- Given a pixel in location I(x,y,t), find the "nearby pixels with the same color"
  - Same intensity (in the local window)
  - Limited displacement



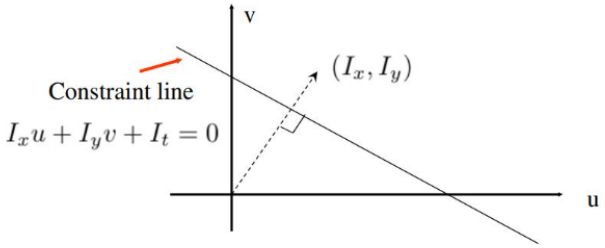
$I(x,y,t)$

$I(x,y,t+1)$

- $I(x + y, t) = I(x + u, y + v, t + 1)$
- $I(x + u, y + v) \approx I(x, y) + \frac{\partial I}{dx}u + \frac{\partial I}{dy}v \rightarrow I_t + \nabla I \cdot (u, v) = 0 \rightarrow I_x u + I_y v + I_t = 0$
  - This problem is the problem of "global Optical Flow" and is hard to solve due to *under definition*

Motion tracking – Optical Flow – Lukas Kanade

- The previous function is a line in the u,v space
- $I_x u + I_y v + I_t = 0$



- We can try to impose more constrains so the line becomes a point
- By assuming that for an image neighborhood we have constant “velocity”: We want to minimize:

$$E(u,v) = \sum_{x,y \in \Omega} \left(I_x(x,y)u + I_y(x,y)v + I_t \right)^2$$

What do the term of the brightness constancy equation represent?

flow velocities

$$I_x u + I_y v + I_t = 0$$

Image gradients
(at a point p)

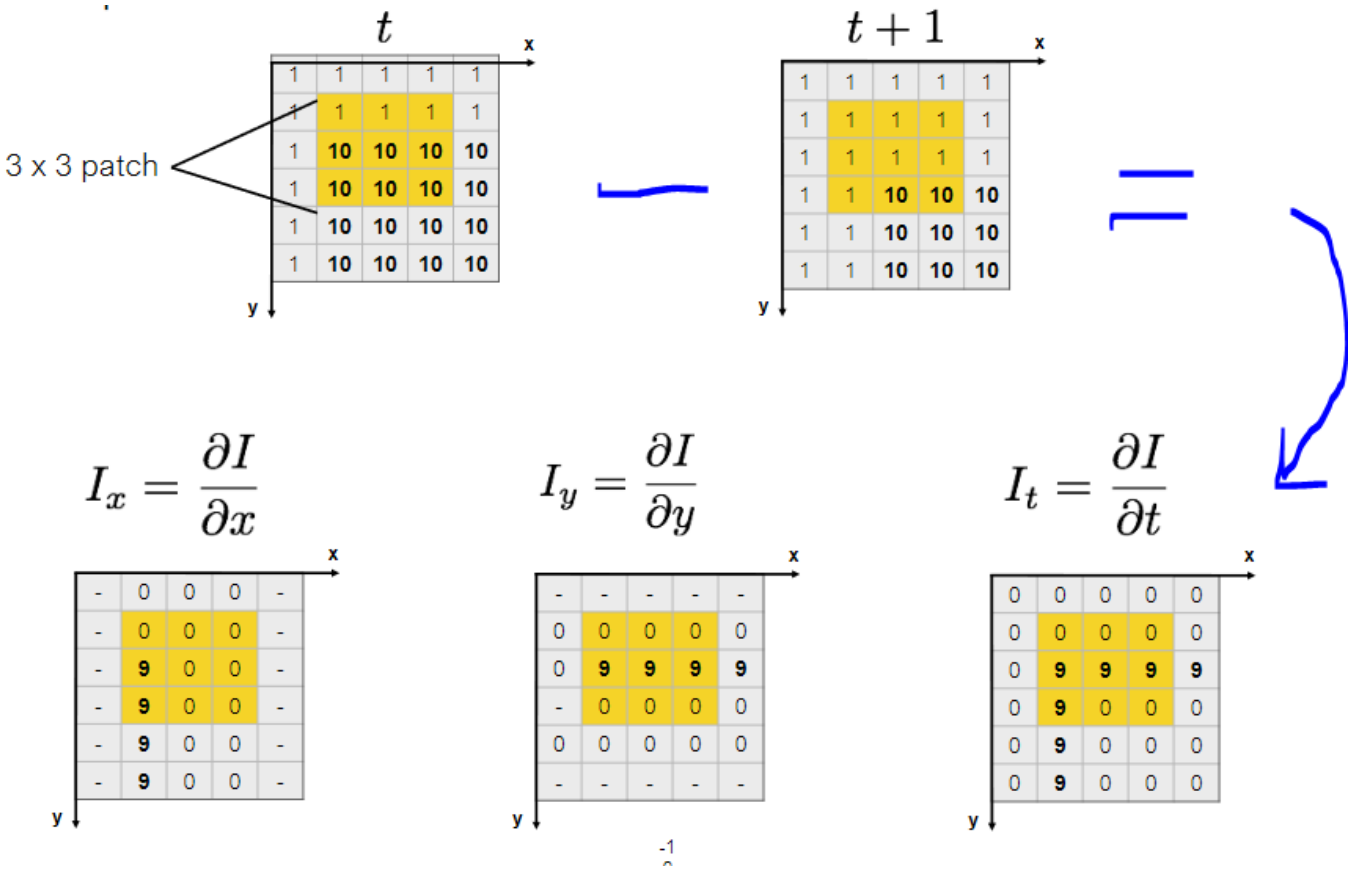
temporal gradient

How do you compute these terms?

$$I_x u + I_y v + I_t = 0$$

How do you compute …

$I_x = \frac{\partial I}{\partial x}$ $I_y = \frac{\partial I}{\partial y}$	$u = \frac{dx}{dt}$ $v = \frac{dy}{dt}$	$I_t = \frac{\partial I}{\partial t}$
spatial derivative	**optical flow**	**temporal derivative**
Forward difference Sobel filter Derivative-of-Gaussian filter …	**We need to solve for this!** (this is the unknown in the optical flow problem)	frame differencing



Relative pose estimation Lecture Slides - University of Toronto

- Depends on the available data
  - 3D or 2D
  - (Eg: Do we work on a monocular camera or a stereo one?)
  - 3D registration - case where f_k and f_k-1 are specified in 3D points
    - PCA
    - SVD, RANSAC

- ICP, combination of above
- What if we have correspondences?
- Rigid Transformation using RANSAC

3D-to-2D

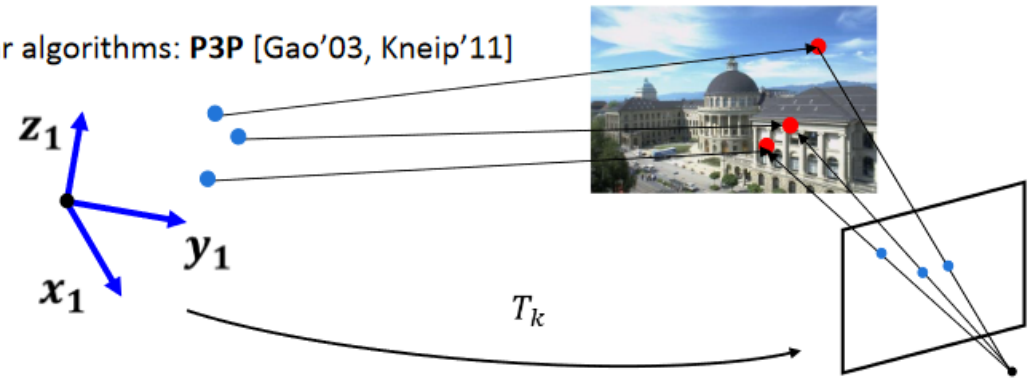
Motion estimation		
2D-2D	3D-2D	3D-3D

Motion from 3D Structure and Image Correspondences

- f_{k-1} is specified in 3D and f_k in **2D**
- This problem is known as *camera resection* or PnP (perspective from *n* points)
- The minimal-case solution involves **3 correspondences** (+1 for disambiguating the 4 solutions)
- The solution is found by minimizing the reprojection error:

$$T_k = \begin{bmatrix} R_{k,k-1} & t_{k,k-1} \\ 0 & 1 \end{bmatrix} = \arg\min_{T_k} \sum_i \| p_k^i - \hat{p}_{k-1}^i \|^2$$

- Popular algorithms: **P3P** [Gao'03, Kneip'11]



2D-to-2D

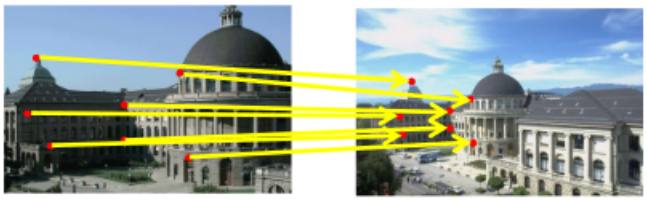
Motion estimation		
2D-2D	3D-2D	3D-3D

Motion from Image Feature Correspondences

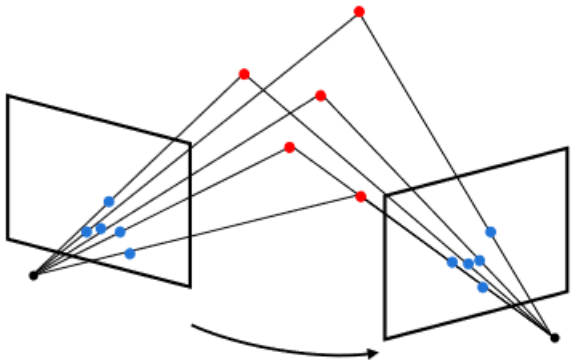
- Both feature points f_{k-1} and f_k are specified **in 2D**
- The minimal-case solution involves **5-point** correspondences
- The solution is found by minimizing the reprojection error:

$$T_k = \begin{bmatrix} R_{k,k-1} & t_{k,k-1} \\ 0 & 1 \end{bmatrix} = \arg\min_{X^i, C_k} \sum_{i,k} \|p_k^i - g(X^i, C_k)\|^2$$

- Popular algorithms: 8- and 5-point algorithms [Hartley'97, Nister'06]



I_{k-1} I_k



DTU

Relative Pose estimation - 2D Image Points

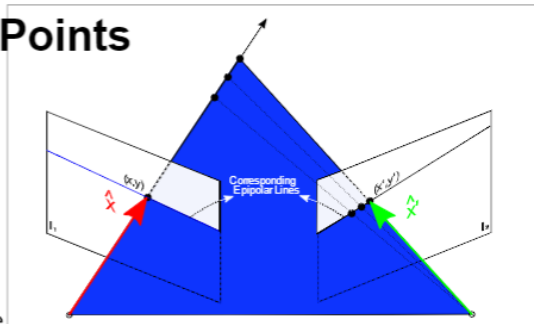
The Essential Matrix can be computed directly from the image coordinates (using SVD).
At least 5 points needed! The more points, the better!
The Essential Matrix can be decomposed into $\boldsymbol{R}$ and $\boldsymbol{t}$ (again using SVD)

Let $p_1 = \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix}$, $p_2 = \begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix}$ be the coordinates one feature correspondence

$$E = \begin{bmatrix} e_{11} & e_{12} & e_{13} \\ e_{21} & e_{22} & e_{23} \\ e_{31} & e_{32} & e_{33} \end{bmatrix} = \begin{bmatrix} e_{11} \\ \vdots \\ e_{33} \end{bmatrix}$$

$$p_2{}^T E p_1 = 0 \Rightarrow [x_1x_2 \;\; y_1x_2 \;\; z_1x_2 \;\; x_1y_2 \;\; y_1y_2 \;\; z_1y_2 \;\; x_1z_2 \;\; y_1z_2 \;\; z_1z_2]E = 0$$

which can be solved with SVD



$p_2^T E p_1 = 0$ ***Epipolar constraint***

$\mathrm{E} = [t]_\times R$ ***essential matrix***

$$[\boldsymbol{t}]_\times = \begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix}$$

3D-to-3D

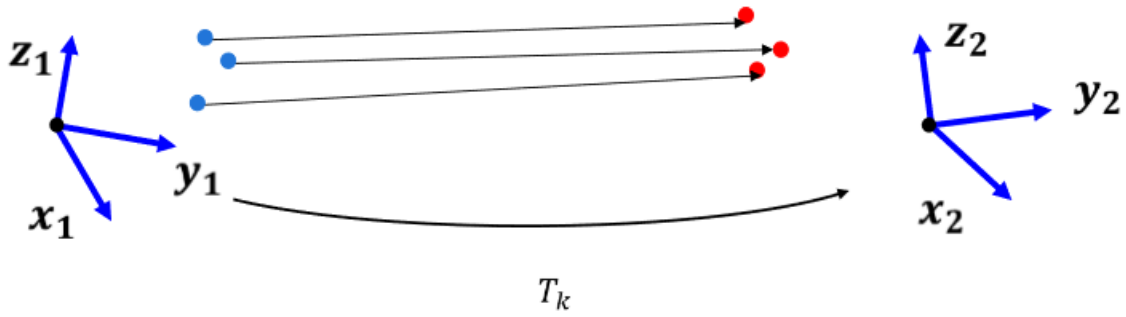
Motion estimation		
2D-2D	3D-2D	3D-3D

Motion from 3D-3D Point Correspondences (point cloud registration)

- Both f_{k-1} and f_k are specified **in 3D**. To do this, it is necessary to triangulate 3D points (e.g. use a stereo camera)
- The minimal-case solution involves **3 non-collinear correspondences**
- The solution is found by minimizing the 3D-3D Euclidean distance:

$$T_k = \begin{bmatrix} R_{k,k-1} & t_{k,k-1} \\ 0 & 1 \end{bmatrix} = \arg\min_{X^i, C_k} \sum_{i,k} \| p_k^i - g(X^i, C_k) \|^2$$

- Popular algorithm: [Arun'87] for global registration, ICP for local refinement or Bundle Adjustment (BA)



What happens over time? Lecture Slides - University of Toronto

VO Working Principle

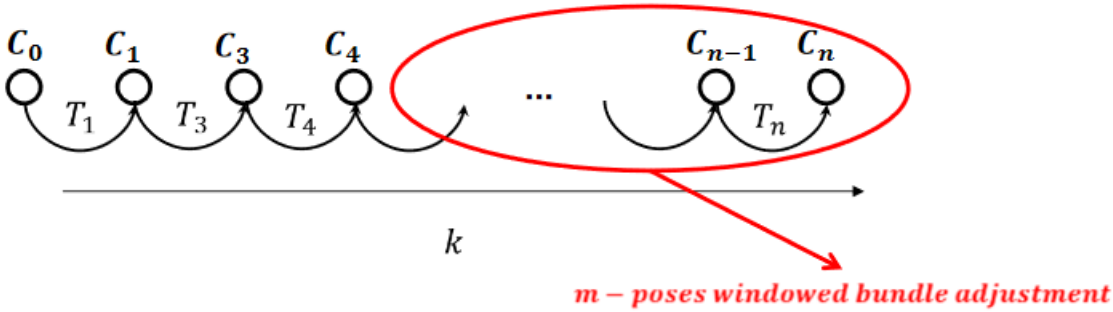
1. Compute the relative motion T_k from images I_{k-1} to image I_k

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

2. Concatenate them to recover the full trajectory

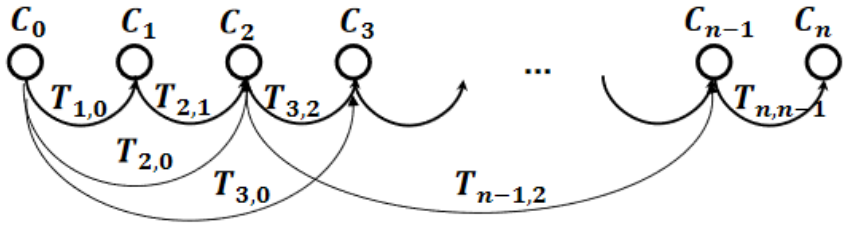
$$C_n = C_{n-1} T_n$$

3. An optimization over the last m poses can be done to refine locally the trajectory (Pose-Graph or Bundle Adjustment)



Pose-Graph Optimization

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

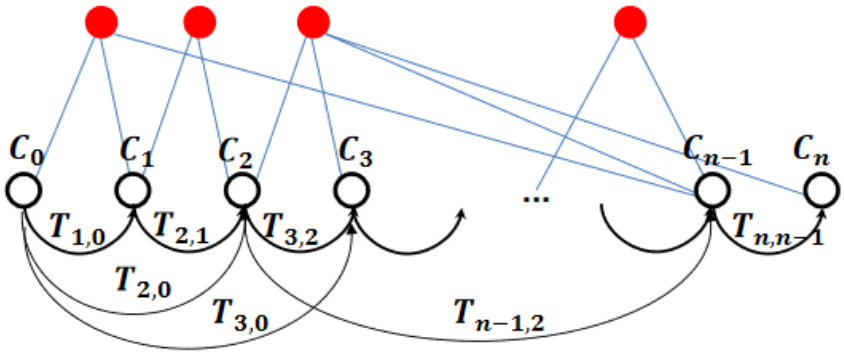


- Transformations can be computed also between non-adjacent frames T_{ij} (e.g., when features from previous keyframes are still observed). They can be used as additional constraints to improve cameras poses by minimizing the following:

$$\sum_i \sum_j \left\| C_i - T_{ij} C_j \right\|^2$$

- For efficiency, only the last m keyframes are used
- Gauss-Newton or Levenberg-Marquadt are typically used to minimize it. For large graphs, efficient open-source tools: g2o, GTSAM, Google Ceres

Bundle Adjustment (BA)



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

$$\arg\min_{X^i, C_k} \sum_{i,k} \|p_k^i - g(X^i, C_k)\|^2$$

- In order to not get stuck in local minima, the initialization should be close to the minimum
- Gauss-Newton or Levenberg-Marquadt can be used. For large graphs, efficient open-source software exists: GTSAM, g2o, Google Ceres can be used.

Bundle Adjustment vs Pose-graph Optimization

- BA is **more precise** than pose-graph optimization because it adds additional constraints (*landmark constraints*)
- But **more costly**: $O\big((qM + lN)^3\big)$ with M and N being the number of points and cameras poses and q and l the number of parameters for points and camera poses. Workarounds:
  - A **small window size** limits the number of parameters for the optimization and thus makes real-time bundle adjustment possible.
  - It is possible to reduce the computational complexity by just optimizing over the camera parameters and keeping the 3-D landmarks fixed, e.g., (**motion-only BA**)

Improving the Accuracy of VO

- IMU
- Compass
- GPS
- Laser