Self-Driving Cars

Lecture 7 – Odometry, SLAM and Localization

Prof. Dr.-Ing. Andreas Geiger

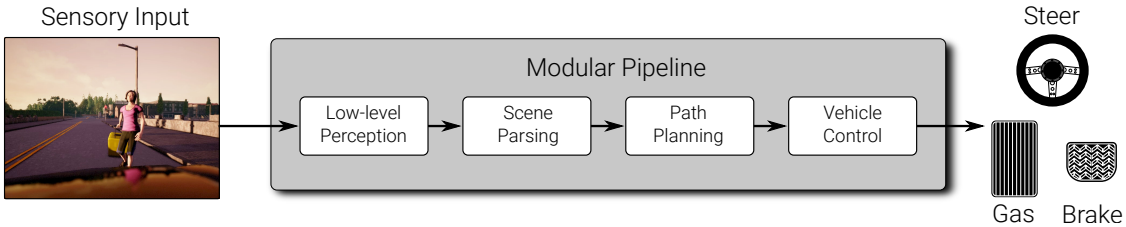
Autonomous Vision Group

University of Tübingen / MPI-IS

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

ellis
European Laboratory for Learning and Intelligent Systems

Modular Pipeline

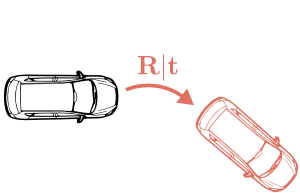


Course Content:

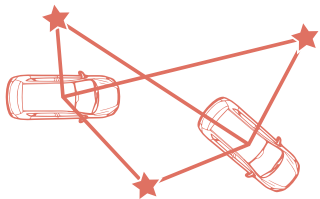
- Lectures 5+6: Vehicle Dynamics and Control
- Lectures 7-11: Low-level Perception and Scene Parsing
- Lecture 12: Decision Making and Planning

Ego-motion Estimation and Localization

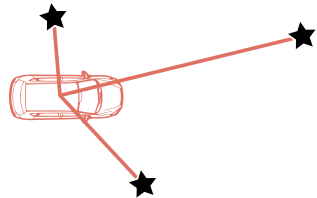
Self-driving cars operate in a **dynamic environments** and hence must be aware of their **ego-motion** (own motion) and the motion of other traffic participants.



Visual Odometry



SLAM



Localization

- **Visual odometry** refers to the estimation of relative ego-motion from images
- **SLAM algorithms** build a map and simultaneously localize in that map
- **Localization methods** find the global pose of a vehicle in a given map

Agenda

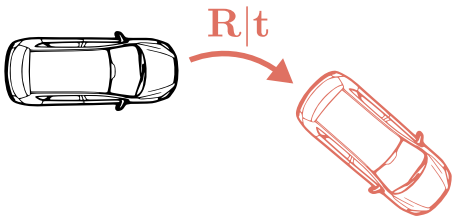
7.1 Visual Odometry

7.2 Simultaneous Localization and Mapping

7.3 Localization

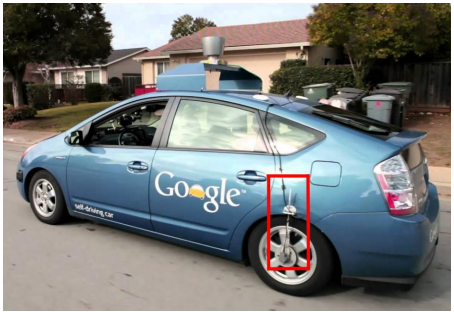
7.1
Visual Odometry

Odometry



- **Odometry** is the use of sensors to estimate change in ego-position over time
- Greek origin: odos (=route) and metron (=measure)
- In the example above, the car undergoes a **rigid body motion** between 2 frames
- Odometry yields **relative motion estimates** (not global position wrt. a map)
- It is hence sensitive to **error accumulation** over time (only precise locally)
- Odometry information can be obtained using a variety of sensors …

Odometry

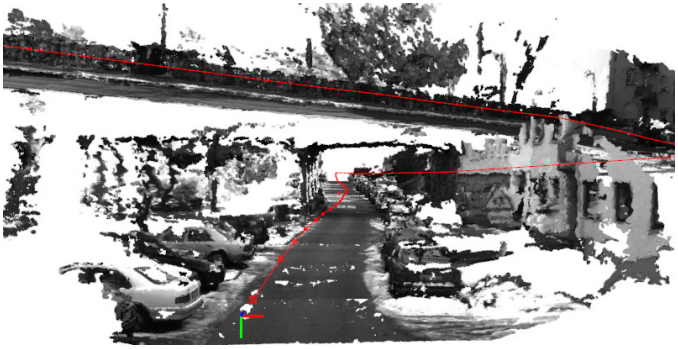


Wheel Odometry

Inertial Measurement Unit (IMU)

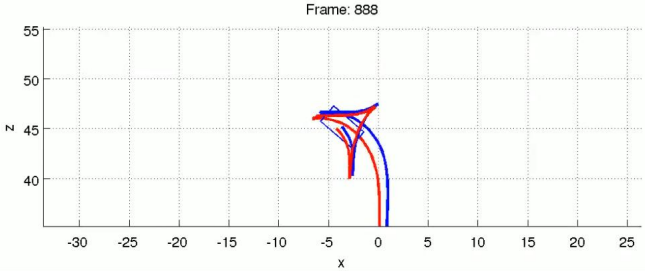
- **Wheel odometry systems** use wheel encoders to measure wheel rotation
- **Inertial measurement units** measure a body's forces ($\Rightarrow$ acceleration)
- **Visual odometry algorithms** use camera images ($\Rightarrow$ focus of this unit)
- Multiple systems often combined to complement each other, e.g., VIO

Visual Odometry



- **Visual odometry** is concerned with **tracking the pose,** i.e., position and orientation, of the camera with respect to the environment from its images
- VO considers a limited set of recent images for **real-time** $\Rightarrow$ error accumulation
- A (sparse) **local map** is often built as a by-product, but mapping is not the focus

Visual Odometry



Kitt, Geiger and Lategahn: Visual odometry based on stereo image sequences with RANSAC-based outlier rejection scheme. IV, 2010.

Visual Odometry

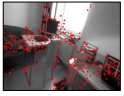
Geiger, Ziegler and Stiller: StereoScan: Dense 3d reconstruction in real-time. Intelligent Vehicles Symposium, 2011.

Indirect vs. Direct Methods

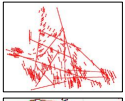
Indirect

Input images

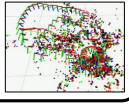
Extract and match keypoints (SIFT,BRIEF,...)



Track: min. reprojection error (point distances)



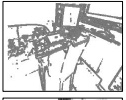
Map: estimate keypoint parameters (f.e. 3D coordinates)



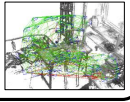
Direct

Input images

Track: min. photometric/ geometric error pixel-wise



Map: estimate per-pixel depth from photoconsistency



Slide Credits: Jörg Stückler

Indirect Visual Odometry

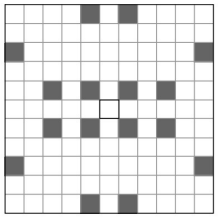
Extract and Match Keypoints

-1	-1	-1	-1	-1
-1	+1	+1	+1	-1
-1	+1	+8	+1	-1
-1	+1	+1	+1	-1
-1	-1	-1	-1	-1

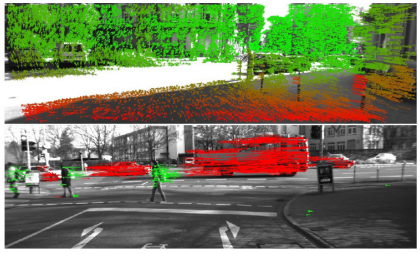
(a) Blob detector

-1	-1	0	+1	+1
-1	-1	0	+1	+1
0	0	0	0	0
+1	+1	0	-1	-1
+1	+1	0	-1	-1

(b) Corner detector



(c) Feature descriptor



- Detect **salient points** in the image (blobs, corners) and extract **local features**
- Features should be invariant to perspective and illumination changes
- Many options (some faster than others): SIFT, SURF, U-SURF, BRISK, ORB, FAST
- **Match features** between two images by their similarity $\Rightarrow$ correspondences

Geiger, Ziegler and Stiller: StereoScan: Dense 3d reconstruction in real-time. Intelligent Vehicles Symposium, 2011.

Image Formation

2D Points

2D points can be written in **inhomogeneous coordinates** as

$$\mathbf{x} = \begin{pmatrix} x \\ y \end{pmatrix} \in \mathbb{R}^2$$

or in **homogeneous coordinates** as

$$\tilde{\mathbf{x}} = \begin{pmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{w} \end{pmatrix} \in \mathbb{P}^2$$

where $\mathbb{P}^2 = \mathbb{R}^3 \setminus \{(0,0,0)\}$ is called **projective space.**

Remark: Homogeneous vectors that differ only by scale are considered equivalent and define an equivalence class. $\Rightarrow$ Homogeneous vectors are defined only up to scale.

2D Points

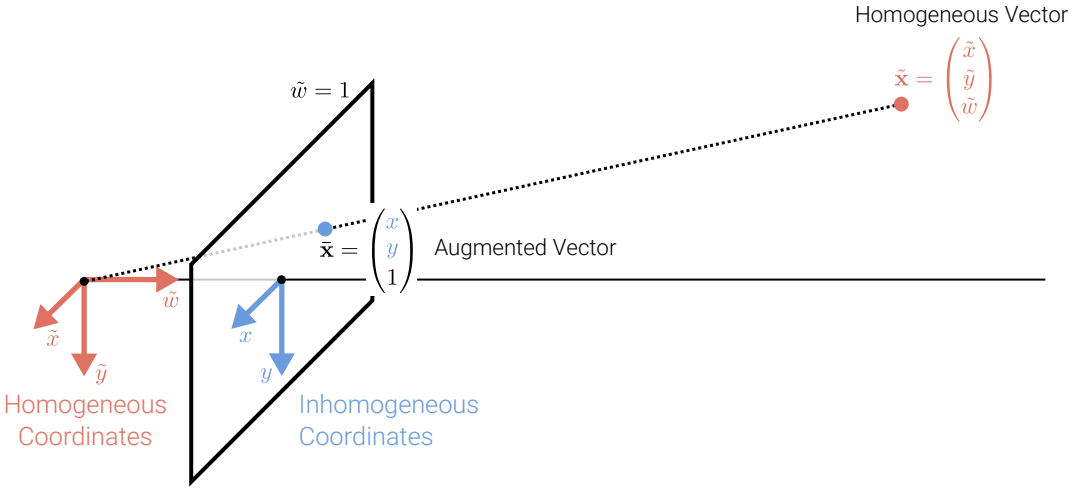
An **inhomogeneous vector** $\mathbf{x}$ is converted to a **homogeneous vector** $\tilde{\mathbf{x}}$ as follows

$$\tilde{\mathbf{x}} = \begin{pmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{w} \end{pmatrix} = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} \mathbf{x} \\ 1 \end{pmatrix} = \bar{\mathbf{x}}$$

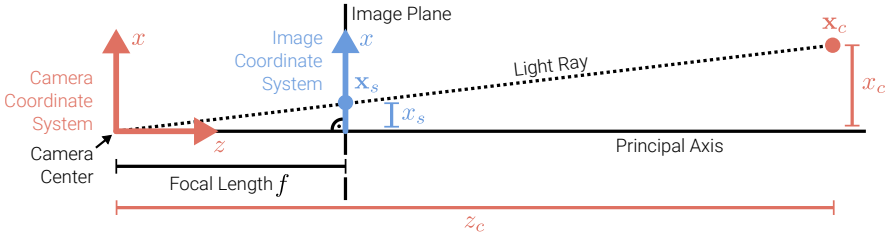
with **augmented vector** $\bar{\mathbf{x}}$.To convert in the opposite direction we divide by $\tilde{w}$:

$$\bar{\mathbf{x}} = \begin{pmatrix} \mathbf{x} \\ 1 \end{pmatrix} = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \frac{1}{\tilde{w}} \tilde{\mathbf{x}} = \frac{1}{\tilde{w}} \begin{pmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{w} \end{pmatrix} = \begin{pmatrix} \tilde{x}/\tilde{w} \\ \tilde{y}/\tilde{w} \\ 1 \end{pmatrix}$$

2D Points


Homogeneous Vector
$\tilde{\mathbf{x}} = \begin{pmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{w} \end{pmatrix}$
$\tilde{w} = 1$
$\bar{\mathbf{x}} = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$
Augmented Vector
$\tilde{w}$
$\tilde{x}$
$\tilde{y}$
x
y
Homogeneous Coordinates
Inhomogeneous Coordinates

Perspective Projection



$$\frac{x_s}{f} = \frac{x_c}{z_c}$$

Perspective projection of a 3D point $\mathbf{x}_c \in \mathbb{R}^3$ to pixel coordinates $\mathbf{x}_s \in \mathbb{R}^2$:

- The light ray passes through the camera center, the pixel $\mathbf{x}_s$ and the point $\mathbf{x}_c$
- Convention: the principal axis (orthogonal to image plane) aligns with the z-axis
- Remark: the y coordinate is not shown here for clarity, but behaves similarly

Perspective Projection

In **perspective projection,** 3D points in camera coordinates are mapped to the image plane by **dividing** them **by their z component** and multiplying with the focal length:

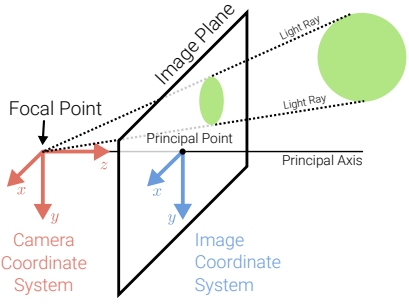
$$\begin{pmatrix} x_s \\ y_s \end{pmatrix} = \begin{pmatrix} f x_c / z_c \\ f y_c / z_c \end{pmatrix} \quad \Leftrightarrow \quad \tilde{\mathbf{x}}_s = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \bar{\mathbf{x}}_c$$

Note that this projection is **linear** when using **homogeneous coordinates.**

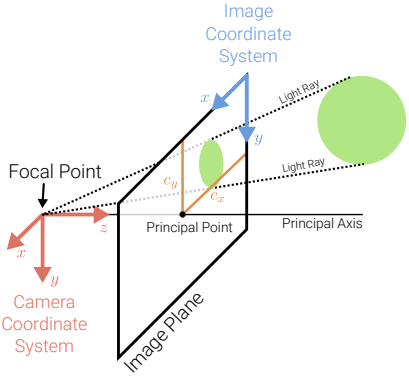
Remark: The unit for f is px (=pixels) to convert metric 3D points into pixels.

Perspective Projection

Without Principal Point Offset



With Principal Point Offset



- To ensure positive pixel coordinates, a **principal point offset** $\mathbf{c}$ is usually added
- This moves the image coordinate system to the corner of the image plane

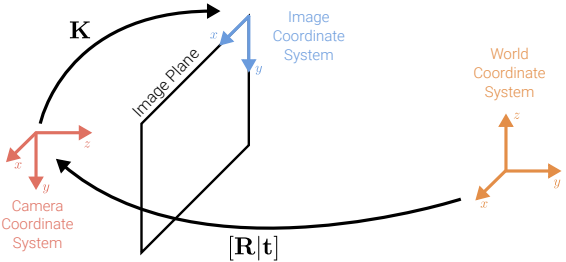
Perspective Projection

The **complete perspective projection model** is given by:

$$\begin{pmatrix} x_s \\ y_s \end{pmatrix} = \begin{pmatrix} f_x\, x_c/z_c + s\, y_c/z_c + c_x \\ f_y\, y_c/z_c + c_y \end{pmatrix} \quad \Leftrightarrow \quad \tilde{\mathbf{x}}_s = \begin{bmatrix} f_x & s & c_x & 0 \\ 0 & f_y & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \bar{\mathbf{x}}_c$$

- The left 3×3 submatrix of the projection matrix is called **calibration matrix** $\mathbf{K}$
- The parameters of $\mathbf{K}$ are called **camera intrinsics** (as opposed to extrinsic pose)
- Here, f_x and f_y are independent, allowing for different pixel aspect ratios
- The skew s arises due to the sensor not mounted perpendicular to the optical axis
- In practice, we often set $f_x = f_y$ and $s = 0$, but model $\mathbf{c} = (c_x, c_y)^\top$

Chaining Transformations



Let $\mathbf{K}$ be the calibration matrix (intrinsics) and $[\mathbf{R}|\mathbf{t}]$ the camera pose (extrinsics).

We **chain both transformations** to project a point in world coordinates to the image:

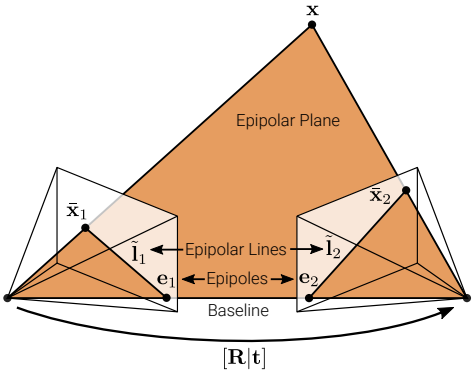
$$\tilde{\mathbf{x}}_s = \begin{bmatrix} \mathbf{K} & \mathbf{0} \end{bmatrix} \bar{\mathbf{x}}_c = \begin{bmatrix} \mathbf{K} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix} \bar{\mathbf{x}}_w = \mathbf{K} \begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix} \bar{\mathbf{x}}_w = \mathbf{P}\, \bar{\mathbf{x}}_w$$

Epipolar Geometry

Epipolar Geometry

Goal: Recovery of camera pose (and 3D structure) from image correspondences.
The required relationships are described by the two-view **epipolar geometry.**

- Let $\mathbf{R}$ and $\mathbf{t}$ denote the relative pose between **two perspective cameras**
- A 3D point $\mathbf{x}$ is projected to pixel $\bar{\mathbf{x}}_1$ in image 1 and to pixel $\bar{\mathbf{x}}_2$ in image 2
- The 3D point $\mathbf{x}$ and the two camera centers span the **epipolar plane**
- The correspondence of pixel $\bar{\mathbf{x}}_1$ in image 2 must lie on the **epipolar line** $\tilde{\mathbf{l}}_2$ in image 2
- All epipolar lines pass through the **epipole**



Epipolar Geometry

The **epipolar constraint** is given by

$$\bar{\mathbf{x}}_2^\top \tilde{\mathbf{E}}\, \bar{\mathbf{x}}_1 = 0$$

with **essential matrix:**

$$\tilde{\mathbf{E}} = [\mathbf{t}]_\times \mathbf{R}$$

Let $\bar{\mathbf{x}}_1 = (x_1\ y_1\ 1)^\top$ and $\bar{\mathbf{x}}_2 = (x_2\ y_2\ 1)^\top$. This allows us to rewrite the epipolar constraint in terms of the elements of the essential matrix as follows ..

Estimating the Epipolar Geometry

We can **recover the essential matrix** $\tilde{\mathbf{E}}$ from N **image correspondences** forming N homogeneous equations in the nine elements of $\tilde{\mathbf{E}}$:

$$\begin{array}{ccccccc} x_1\,x_2\,e_{11} & + & y_1\,x_2\,e_{12} & + & x_2\,e_{13} & & \\ x_1\,y_2\,e_{21} & + & y_1\,y_2\,e_{22} & + & y_2\,e_{23} & & \\ x_1\,e_{31} & + & y_1\,e_{32} & + & e_{33} & = & 0 \end{array}$$

As $\tilde{\mathbf{E}}$ is homogeneous we use **singular value decomposition** to constrain the scale.

Note that some terms are products of two image measurements and hence amplify measurement noise asymmetrically. Thus, the **normalized 8-point algorithm** whitens the observations to have zero-mean and unit variance before the calculation and back-transforms the matrix recovered by SVD accordingly.

Hartley: In Defense of the Eight-Point Algorithm. TPAMI, 1997.

Estimating the Epipolar Geometry

From $\tilde{\mathbf{E}}$, we can recover the **direction** $\hat{\mathbf{t}}$ of the **translation** vector $\mathbf{t}$. We have:

$$\hat{\mathbf{t}}^\top \tilde{\mathbf{E}} = \hat{\mathbf{t}}^\top [\mathbf{t}]_\times \mathbf{R} = \mathbf{0}$$

Thus, $\tilde{\mathbf{E}}$ is singular and we obtain $\hat{\mathbf{t}}$ as the left singular vector associated with singular value 0. In practice the singular value will not be exactly 0 due to measurement noise, and we choose the smallest one. The other two singular values are roughly equal.

The rotation matrix $\mathbf{R}$ can also be calculated, see Szeliski, Section 11.3 (p. 683).

Remark: The essential matrix has 5 DoF (3 for rotation $\mathbf{R}$, 2 for translation direction $\hat{\mathbf{t}}$). In other words, the **global scale cannot be determined** using monocular VO.

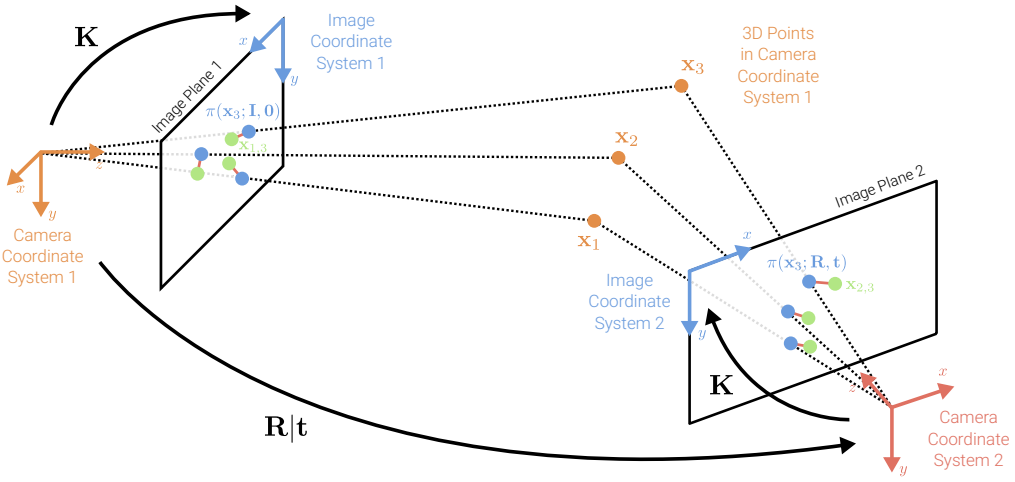
Non-linear Optimization

The rigid body motion $(\mathbf{R}, \mathbf{t})$ can be further refined via **non-linear optimization.**
Let $\pi(\mathbf{x}_i, \mathbf{R}, \mathbf{t})$ denote the projection of the i'th 3D point onto the 2D image plane and let $(\mathbf{x}_{1,i}, \mathbf{x}_{2,i})$ denote the 2D feature correspondences in the first and second frame.
Minimize the **reprojection error** of all correspondences wrt. motion and 3D location:

$$\mathbf{R}^*, \mathbf{t}^*(, \mathbf{X}^*) = \underset{\mathbf{R},\mathbf{t}(,\mathbf{X})}{\operatorname{argmin}} \sum_{i=1}^{N} \|\mathbf{x}_{1,i} - \pi(\mathbf{x}_i; \mathbf{I}, \mathbf{0})\|_2^2 + \|\mathbf{x}_{2,i} - \pi(\mathbf{x}_i; \mathbf{R}, \mathbf{t})\|_2^2$$

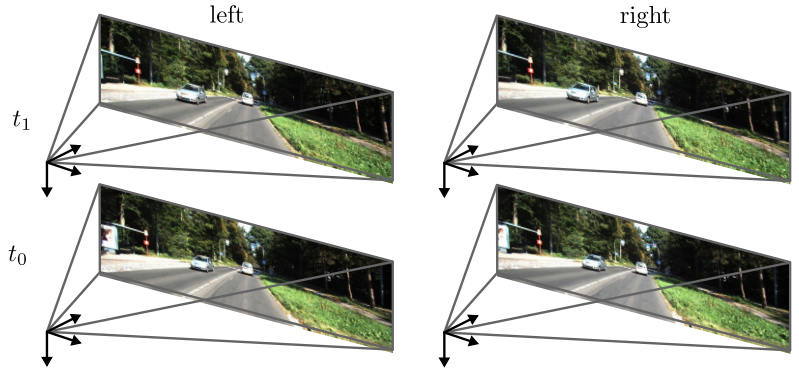
- Optimizing reprojection errors is leads to better results (better noise model)
- Triangulation can be used to initialize $\mathbf{X} = \{\mathbf{x}_i\}$; speed-up by not optimizing $\mathbf{X}$
- Outliers must be removed to not affect the estimate (e.g., using RANSAC)
- For monocular VO, the length of the translation vector cannot be determined
- This ambiguity can be resolved by using stereo images or wheel odometry

Non-linear Optimization



- Use off-the-shelf **non-linear least squares** solver, e.g., Levenberg-Marquardt

Stereo Visual Odometry

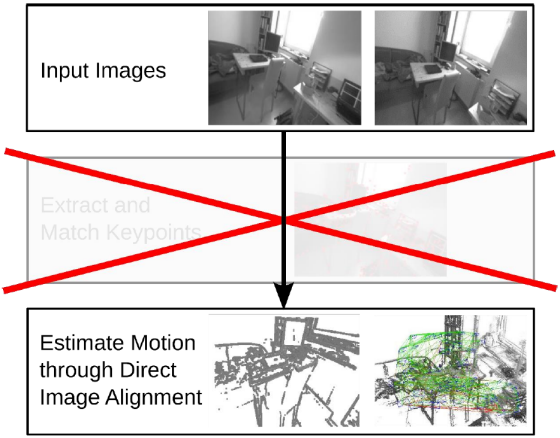


- If a **stereo camera** is available we can optimize with respect to all 4 images
- If the stereo camera is calibrated, we can estimate **global scale** (i.e., $\|\mathbf{t}\|$)
- Furthermore, stereo measurements allow for instantaneous triangulation

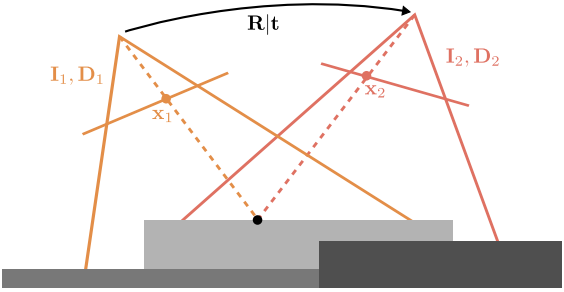
Direct Visual Odometry

Direct Visual Odometry

- Avoid hand-crafted keypoint detection, description and matching
- Instead: **direct image alignment** based on all image pixels
- Depth from sensor (RGB-D, Lidar) or through multi-view stereo



Direct Visual Odometry

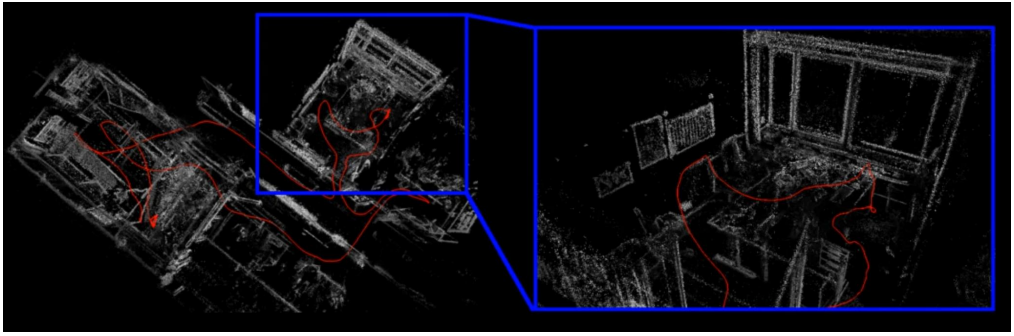


- If we know per-pixel depth, we can **simulate** an image from a different view point
- Ideally, image $\mathbf{I}_2$ **warped** into the first camera's image plane is equal to image $\mathbf{I}_1$:

$$\mathbf{R}^*, \mathbf{t}^*, \mathbf{D}_1^* = \underset{\mathbf{R},\mathbf{t},\mathbf{D}_1}{\operatorname{argmin}} \sum_{\mathbf{x}\in\Omega} \|\mathbf{I}_1(\mathbf{x}) - \mathbf{I}_2(\pi(\mathbf{x}; \mathbf{R}, \mathbf{t}, \mathbf{D}_1))\|_2^2$$

Kerl, Sturm and Cremers: Dense visual SLAM for RGB-D cameras. IROS, 2013.

Direct Visual Odometry



- Direct VO methods often consider only parts of the image (e.g., non-saturated)
- Direct methods can also be combined with feature-based (indirect) methods

Engel, Koltun and Cremers: Direct Sparse Odometry. TPAMI, 2018.

Indirect vs. Direct Methods

Some Remarks:

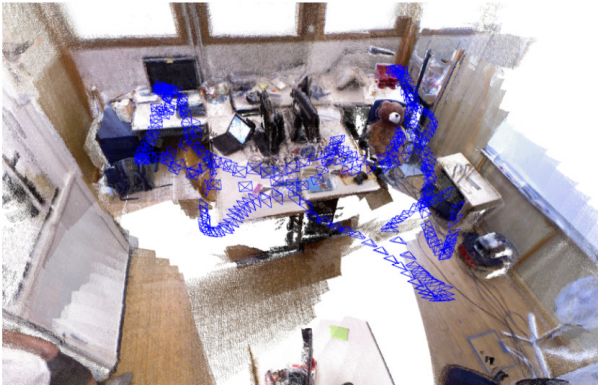
- Direct methods often lead to more accurate results as they exploit the full image
- However, satisfying real-time requirements requires efficient implementations
- Furthermore, they suffer from local minima and require accurate initialization
- Feature-based methods are faster but in general less accurate
- This is because often only few reliable feature correspondences can be found
- However, they are more robust to initialization and less prone to local minima
- Thus, often both techniques are combined (e.g., using features to initialize pose)

7.2
SLAM

Simultaneous Localization and Mapping (SLAM)

- So far: Optimization of 2 adjacent frames, no focus on map
- Now: optimize over larger windows (ideally: entire history)
- We optimize both poses and map (e.g., 3D feature locations)
- SLAM is a chicken-egg problem (localization requires mapping and vice-versa) $\Rightarrow$ Joint optimization of poses and map is necessary
- Key feature of SLAM: Correct accumulation errors via loop-closure detection
- The resulting map can be used for localization
- There exist indirect (feature-based) and direct SLAM methods
- Many flavors: EKF SLAM, Bundle adjustment, Windowed BA, …
- We will cover feature-based SLAM via bundle adjustment

Feature-based SLAM



- **Goal: Optimize reprojection errors** (distance between observed feature and projected 3D point in image plane) **wrt. camera parameters and 3D point cloud**

Bundle Adjustment

Let $\Pi = \{\pi_i\}$ denote the N cameras including their intrinsic and extrinsic parameters.
Let $\mathcal{X}_w = \{\mathbf{x}_p^w\}$ with $\mathbf{x}_p^w \in \mathbb{R}^3$ denote the set of P 3D points in world coordinates.
Let $\mathcal{X}_s = \{\mathbf{x}_{ip}^s\}$ with $\mathbf{x}_{ip}^s \in \mathbb{R}^2$ denote the image (screen) observations in all i cameras.

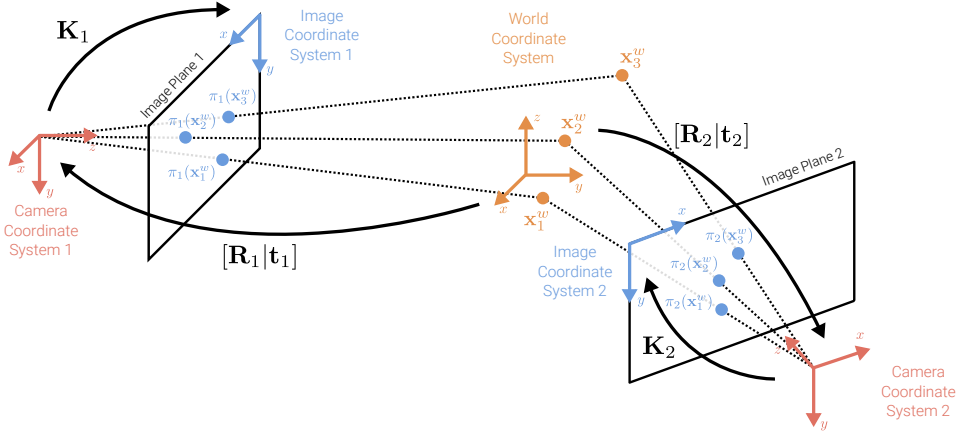
Bundle adjustment minimizes the reprojection error of all observations:

$$\Pi^*, \mathcal{X}_w^* = \underset{\Pi, \mathcal{X}_w}{\operatorname{argmin}} \sum_{i=1}^{N} \sum_{p=1}^{P} w_{ip} \left\| \mathbf{x}_{ip}^s - \pi_i(\mathbf{x}_p^w) \right\|_2^2$$

Here, w_{ip} indicates if point p is observed in image i and $\pi_i(\mathbf{x}_p^w)$ is the 3D-to-2D projection of 3D world point $\mathbf{x}_p^w$ onto the 2D image plane of the i'th camera, i.e.:

$$\pi_i(\mathbf{x}_p^w) = \begin{pmatrix} \tilde{x}_p^s / \tilde{w}_p^s \\ \tilde{y}_p^s / \tilde{w}_p^s \end{pmatrix} \quad \text{with} \quad \tilde{\mathbf{x}}_p^s = \mathbf{K}_i (\mathbf{R}_i \, \mathbf{x}_p^w + \mathbf{t}_i)$$

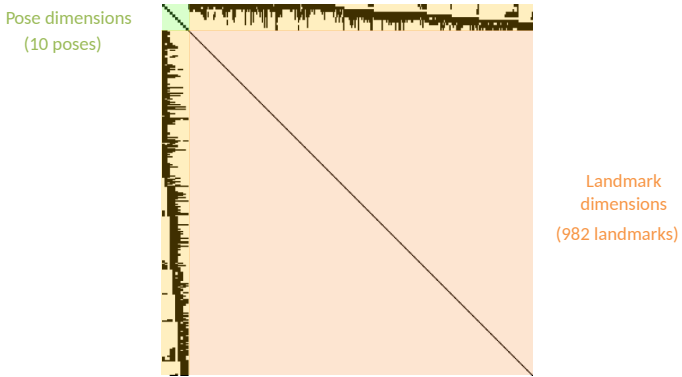
Bundle Adjustment



$\mathbf{K}_i$ and $[\mathbf{R}_i|\mathbf{t}_i]$ are the intrinsic and extrinsic parameters of π_i, respectively.

During bundle adjustment, we optimize $\{(\mathbf{K}_i, \mathbf{R}_i, \mathbf{t}_i)\}$ and $\{\mathbf{x}_p^w\}$ jointly.

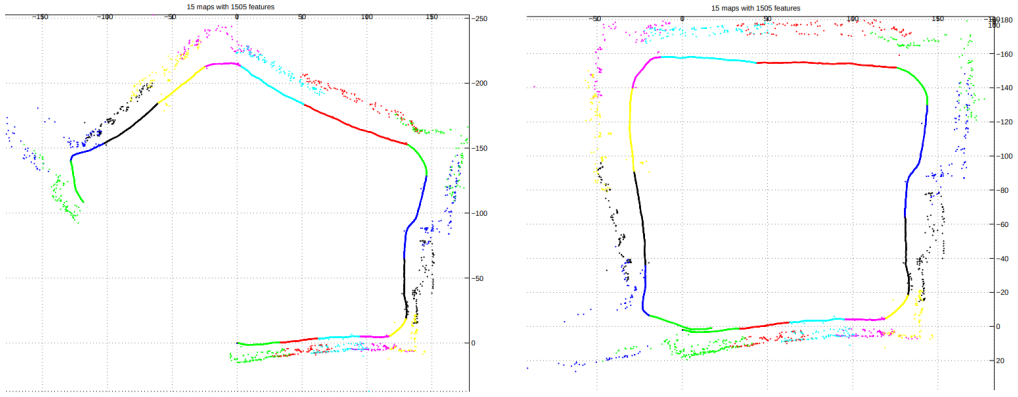
Exploiting Sparsity



- The Hessian is typically **sparse** as landmarks don't interact with each other
- This leads to tremendous computational benefits when using sparse optimizers

Slide Credits: Jörg Stückler

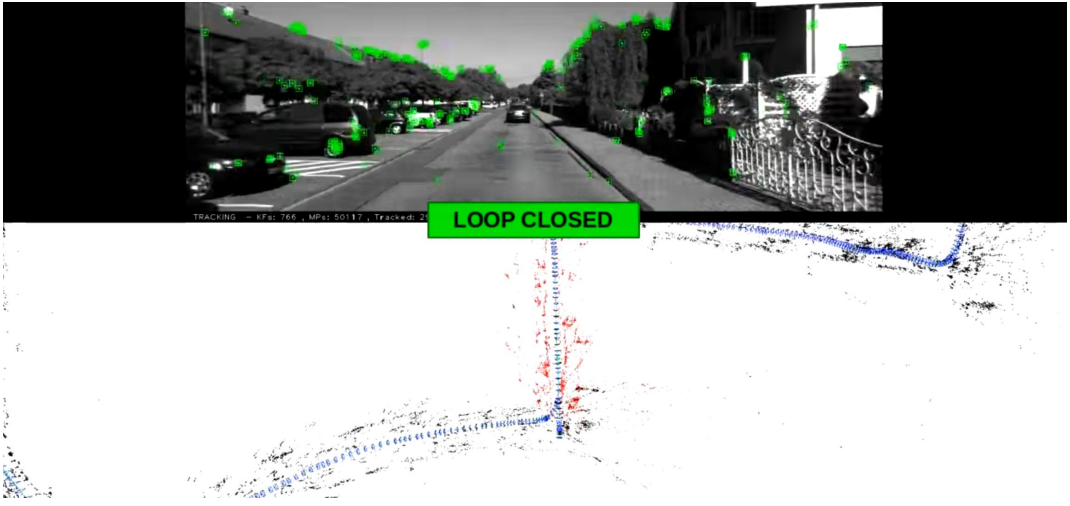
Loop Closure Detection



- Find **correspondences** between the current frame and all previous frames
- Use these correspondences as **additional constraints** during optimization

Clemente, Davison, Reid, Neira and Tardós: Mapping Large Loops with a Single Hand-Held Camera. RSS, 2007.

ORB SLAM



Artal, Montiel and Tardós. ORB-SLAM: A Versatile and Accurate Monocular SLAM System. Transactions on Robotics, 2015.

Indirect SLAM Methods

MonoSLAM [1]	PTAM [2]	ORB-SLAM 2 [3]
+ monocular cameras	+ monocular cameras	+ monocular cameras + stereo cameras + RGB-D cameras
- no global consistency	- no global consistency	+ global consistency
Extended Kalman filtering of camera pose and point coordinates. Includes a motion model	Mapping as BA over keyframes, real-time tracking towards keyframe	Mapping as BA over keyframes, real-time tracking towards keyframe, loop-closing via place recognition
Filtering	Bundle adjustment	Bundle adjustment
- local accuracy -- global accuracy	+ local accuracy - global accuracy	+ local accuracy ++ global accuracy

Slide Credits: Jörg Stückler

Direct SLAM Methods

DVO-SLAM [4]	LSD-SLAM [5]	DSO [6]
+ RGB-D cameras	+ monocular cameras + stereo cameras	+ monocular cameras + stereo cameras
+ global consistency	+ global consistency	- no global consistency
camera pose tracking towards keyframe	camera pose tracking towards keyframe	camera pose tracking towards keyframe, camera pose optimization in local keyframe window
+ depth from sensor	+ depth from stereo comparisons & filtering	++ depth optimization in local keyframe window
tracking-only & pose graph optimization	tracking-and-mapping & pose graph optimization	tracking-and-mapping & direct sparse bundle adjustment in local keyframe window with marginalization
+ local accuracy	+ local accuracy	++ local accuracy

Slide Credits: Jörg Stückler

7.3
Localization

Vehicle Localization

Motivation:

- Control wrt. a path requires knowledge about the position of the vehicle
- Sometimes local knowledge is sufficient (lateral position on highway)
- But often the global location is required (path planning, determining relevant elements that have been marked in a map such as street signs or lane markings)

Localization Approaches:

- Satellite Localization (uses infrastructure)
- Visual Localization (uses visual map)
  - Topometric Localization
  - Learning-based Localization
  - Feature-based Localization
- Map-based Localization (uses road map)

Satellite Localization

Satellite Localization

Satellite Systems:

- GPS: 24 satellites (United States)
  - First satellite launch: 1978
  - Full constellation: 1994
  - Today: 30 satellites
  - Since 2000 full accuracy for all users (not only military users)
- GLONASS: 24 satellites (Russia)
- Galileo: 30 satellites (Europe)
- BeiDou-2: 30 satellites (China)

GPS: 4-5 satellites per orbital plane

Satellite Localization

Trilateration:

- Determine location by measuring distances to satellites at known locations using the geometry of circles and spheres
- Distance to each satellite is measured as time delay between sent signal/code and received signal/code (receiver knows code!)
- Satellites equipped with onboard atomic clock
- Given one satellite, we know which sphere we are on
- Given two satellites, we know which circle we are on
- Given three satellites, we know the exact location
- In practice, a fourth satellite is used for time synchronization
- Differential GPS: Improve measurements using ground stations at fixed location

Satellite Localization

Problems with Satellite Localization:

- Availability
  - Satellites not visible in tunnels, narrow city streets
  - Dependency on national organizations / interests
- Accuracy
  - 5m - 15m for GPS and 0.5m-5m for DGPS
  - Only location, no rotation (not full pose)
- Frequency: max. 5-10 Hz
- Atmospheric variations (lead to errors)
  - Can be partially corrected by DGPS using signal from nearby base station
- DGPS requires communication and initialization
- Multipath effects (wrong signals due to reflections, e.g., at facades)

Satellite Localization

[Danny Iland, Uber]

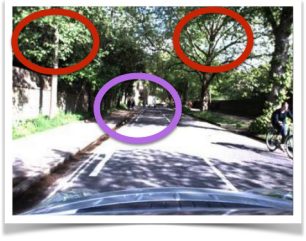
Visual Localization

Visual Localization

Main Idea:

- Record map/database of known locations with associated features
- Features can be extracted from images or laser scans
- Mapping should be conducted under similar conditions as during localization (minimize domain shift wrt. sensor setup and environmental conditions)
- At localization time, try to retrieve features extracted from the current input image/scan in the map/database
- Either localize only at the image level or refine using triangulation or geometric pose estimation

Challenges



Geometry changes

Appearance changes

Maddern, Pascoe, Linegar and Newman: 1 year, 1000 km: The Oxford RobotCar dataset. IJRR, 2017.

Visual Localization:
Topometric Localization

Topometric Localization

- Setup: Record sequence with cameras, Lidar and GPS as ground truth
- Goal: Localize new image (different day/season) wrt. recorded sequence
- Combination of metric and topological localization:
  - Metric: Optimize pose wrt. feature correspondences (discussed later in the lecture)
  - Topological: Estimate observer location qualitatively from finite set of locations
  - Here: Describe image with global feature vector; align query with recorded sequence
  - However, only localize wrt. node (=frame) in graph (no full 6D pose)

Badino, Huber and Kanade: Real-Time Topometric Localization. ICRA, 2012.

Topometric Localization

Map Creation:

- Directed graph, nodes (=frames) created based on distance threshold
- Features: SURF applied to whole image (64D) + average/std.dev. of range scan

Bayesian Localization:

- Predict: $P(x_t|z_{1:t-1}) = \sum_{x_{t-1}} P(x_t|x_{t-1})P(x_{t-1}|z_{1:t-1})$
- Update: $P(x_t|z_{1:t}) \propto P(z_t|x_t)P(x_t|z_{1:t-1})$
- With:
  - Motion model $P(x_t|x_{t-1}) = \mathcal{N}(x_t|x_{t-1} + v_t\Delta t, \sigma^2)$
  - Measurement model $P(z_t|x_t) = \chi(\|z_t - d(x_t)\|, k)$
  - Location $x_t \in \mathbb{R}$, velocity $v_t \in \mathbb{R}$, observation z_t, map feature $d(x_t)$ at location x_t
  - Note: strictly speaking this formulation handles tracking, not localization!

Badino, Huber and Kanade: Real-Time Topometric Localization. ICRA, 2012.

Topometric Localization

Badino, Huber and Kanade: Real-Time Topometric Localization. ICRA, 2012.

Visual Localization:
Learning-based Localization

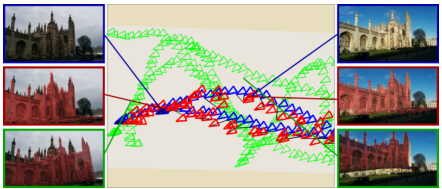
Learning-based Localization

PoseNet:

- Input: Single RGB image (224^2 Px)
- Output: 6 DOF camera pose $\mathbf{p} = (\mathbf{x}, \mathbf{q})$
  - $\mathbf{x}$: 3D camera position
  - $\mathbf{q}$: 3D camera rotation (quaternion)
- 23 Layer deep convolutional network based on GoogLeNet
- More robust than feature-based methods
- Less accurate than feature-based methods

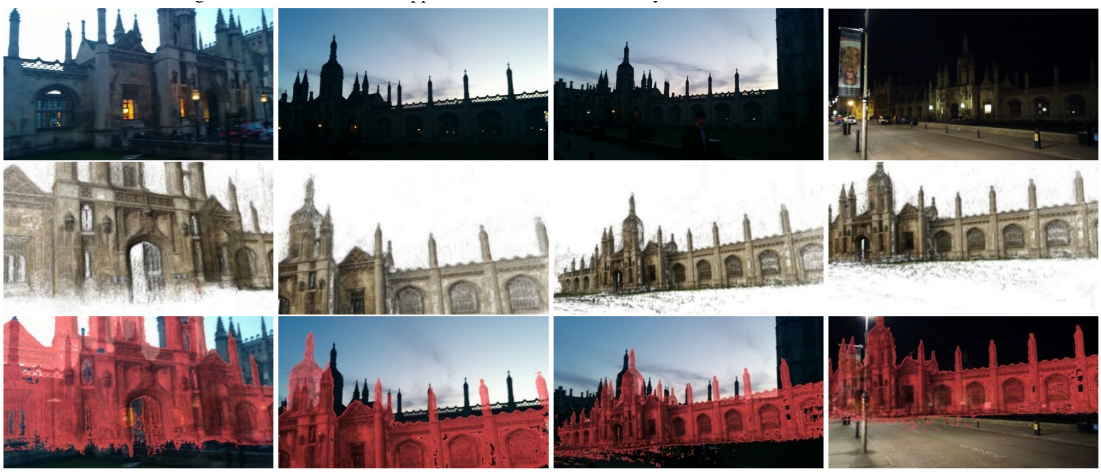
Loss:

$$\mathcal{L} = \|\hat{\mathbf{x}} - \mathbf{x}\|_2 + \beta \left\| \hat{\mathbf{q}} - \frac{\mathbf{q}}{\|\mathbf{q}\|} \right\|_2$$



(green = train, red = pred, blue = GT)

Learning-based Localization: Results



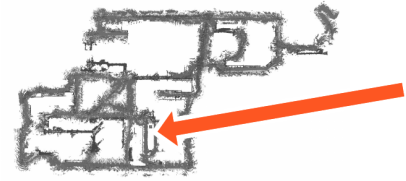
(b) Relocalization under difficult dusk and night lighting conditions. In the dusk sequences, the landmark is silhouetted against the backdrop however again the convnet seems to recognize the contours and estimate pose.

Visual Localization
Feature-based Localization

Feature-based Localization Overview

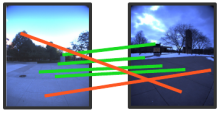
Database Images

Database Map



Geometric Verification

- Image-to-image
- Image-to-3D
- 3D-to-3D

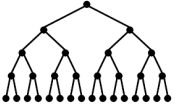


Query Image
(+Depth Map)

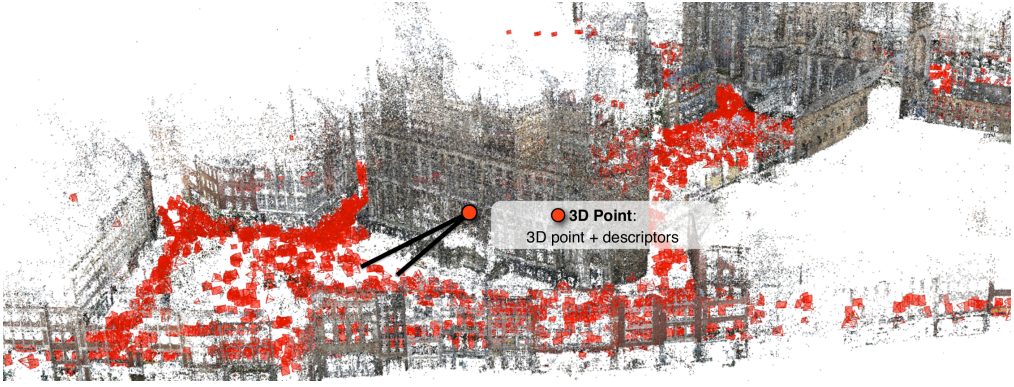
Descriptor Extraction

- Image Descriptors (SIFT, DeepDesc, …)
- Shape Descriptors (FPFH, 3DMatch, CGF, ...)

Descriptor Matching



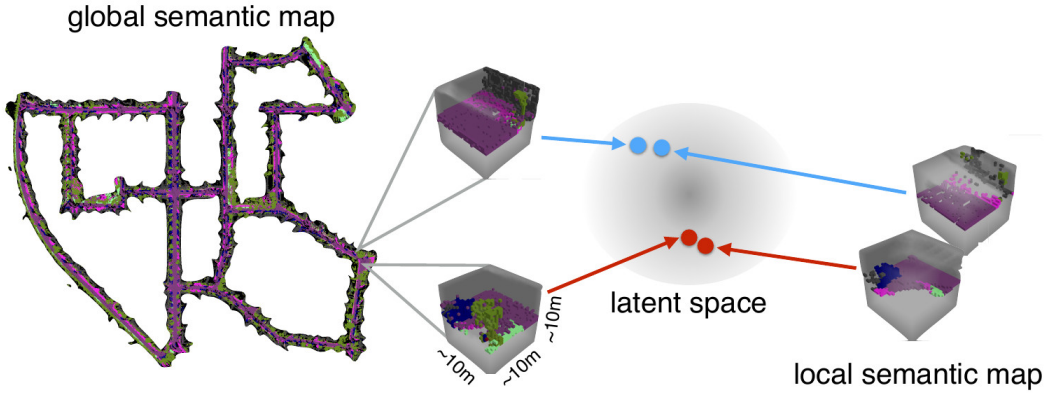
Mapping



- Sparse 3D reconstruction based on sparse feature correspondences (SLAM)
- Associate each 3D point with a local image descriptor (e.g., SIFT, SURF)

Slide Credits: Torsten Sattler

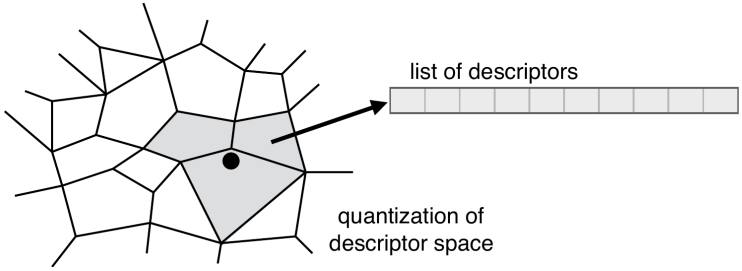
Mapping



- Semantic viewpoint invariant feature representations can facilitate matching

Schönberger, Pollefeys, Geiger and Sattler: Semantic Localization. CVPR, 2018.

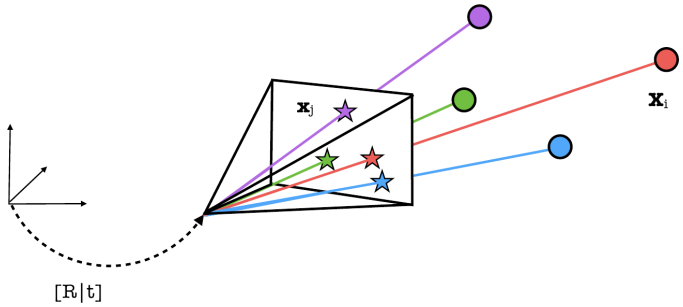
Descriptor Matching



- Search nearest neighbors of features from query image (in descriptor space)
- Approximate search due to curse of dimensionality
- Popular approaches: k-d trees, inverted index, …
- Good software packages available: FLANN, FAISS, …

Slide Credits: Torsten Sattler

Pose Estimation



- Given: n 2D-3D correspondences $(\mathbf{x}_i, \mathbf{X}_i)$
- Goal: compute pose $\mathbf{R}, \mathbf{t}$ such that $\mathbf{x}_i = \mathbf{K}[\mathbf{R}|\mathbf{t}]\mathbf{X}_i$
  Here, $\mathbf{K}, \mathbf{R}, \mathbf{t}$ denote calibration matrix, rotation matrix, and translation vector
- Very efficient methods ($< 2\mu s$) exist if camera calibration is known ($\Rightarrow$ PnP)

Geometric Verification

While probability of missing correct model >η
  Estimate model from *n* random data points
  Estimate support (= **#inliers**) of model [Chum, Matas, Optimal Randomized RANSAC. PAMI 2008]
  If new best model
    Perform Local Optimization (LO) [Lebeda, Matas, Chum, Fixing the Locally Optimized RANSAC. BMVC 2012] [code]
  update best model, η
Return: Model with most inliers

RANdom SAmple Consensus (RANSAC):

- Problem: wrong feature matches corrupt pose estimation result
- Solution: solve minimal problem many times & return best solution

Slide Credits: Torsten Sattler

Map-based Localization

Map-based Localization

- Update location probability distribution on map using VO measurements

Brubaker, Geiger and Urtasun: Map-Based Probabilistic Visual Self-Localization. PAMI, 2016.

Summary

- Visual odometry estimates relative ego-motion from images
- SLAM algorithms build a map and simultaneously localize in that map
- SLAM algorithms use loop closure detection to close loops
- Localization methods find the global pose in a given map
- Indirect and direct VO/SLAM methods exist
- Indirect methods are faster and converge better, but are less accurate
- Direct methods are slower but lead to more accurate results
- Direct and indirect methods can be advantageously combined
- Except for offline mapping, real-time computation is required