Building A Typical Visual SLAM Pipeline

YoungWoo Seo

May 22, 2018

Overview

- What is SLAM (Simultaneous Localization and Mapping)?
  - Problem of building a globally consistent representation of the environment by leveraging both ego-motion compensation and loop-closure [Cadena et al., 2016].
  - Mathematically, it requires to solve a nonlinear least square problem.
  - Estimating the camera trajectory while reconstructing the environment.
  - Nonlinear filtering vs. optimization using (global) bundle adjustment [Strasdat et al., 2010]
  - An activity you do, occasionally, when you go to a department store to pick up something for your significant other.

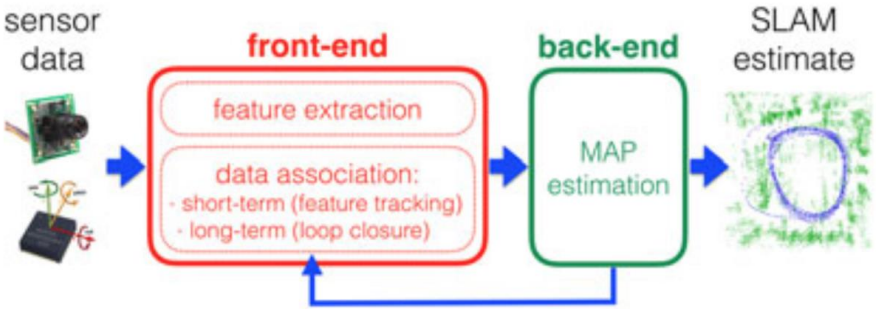
Overview

$$\underset{x}{\text{argmax}}\ p(x|x_0)\prod_i p(z_i|x) =$$

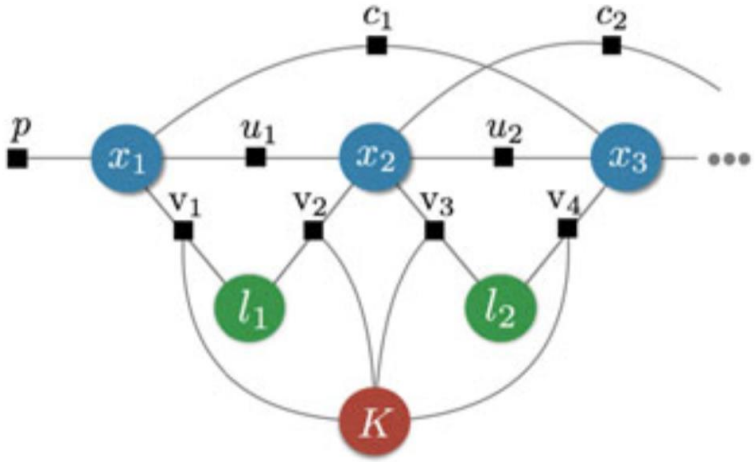
$$\underset{x}{\text{argmax}}\ exp(-\|x-x_0\|^2_{\Sigma_{x_0}})\prod_i exp(-\|z_i-h_i(x)\|^2_{\Sigma_{z,i}}) =$$

$$\underset{x}{\text{argmin}}\ \|x-x_0\|^2_{\Sigma_{x_0}} + \sum_i \|z_i-h_i(x)\|^2_{\Sigma_{z,i}}$$

SLAM as a nonlinear least square.



[Cadena et al., 2016]



"SLAM is formulated as a factor graph"
[Cadena et al., 2016]

Overview

- Why do we care?
  - Anywhere motion needs to be properly handled, e.g., virtual/augmented reality, wearable computing, mapping, localization, robotics, automotive etc.
- What does this presentation cover?
  - A very short crash course on a typical Visual SLAM pipeline: We'll go over an example of feature-based (or sparse feature-based) SLAM, ORB-SLAM.

SLAM is indeed one of the activities you do once in a while -- you go by a department store to buy something for your significant other – at the store you have never visited before.

A Typical Visual SLAM Pipeline

Typical SLAM Pipeline

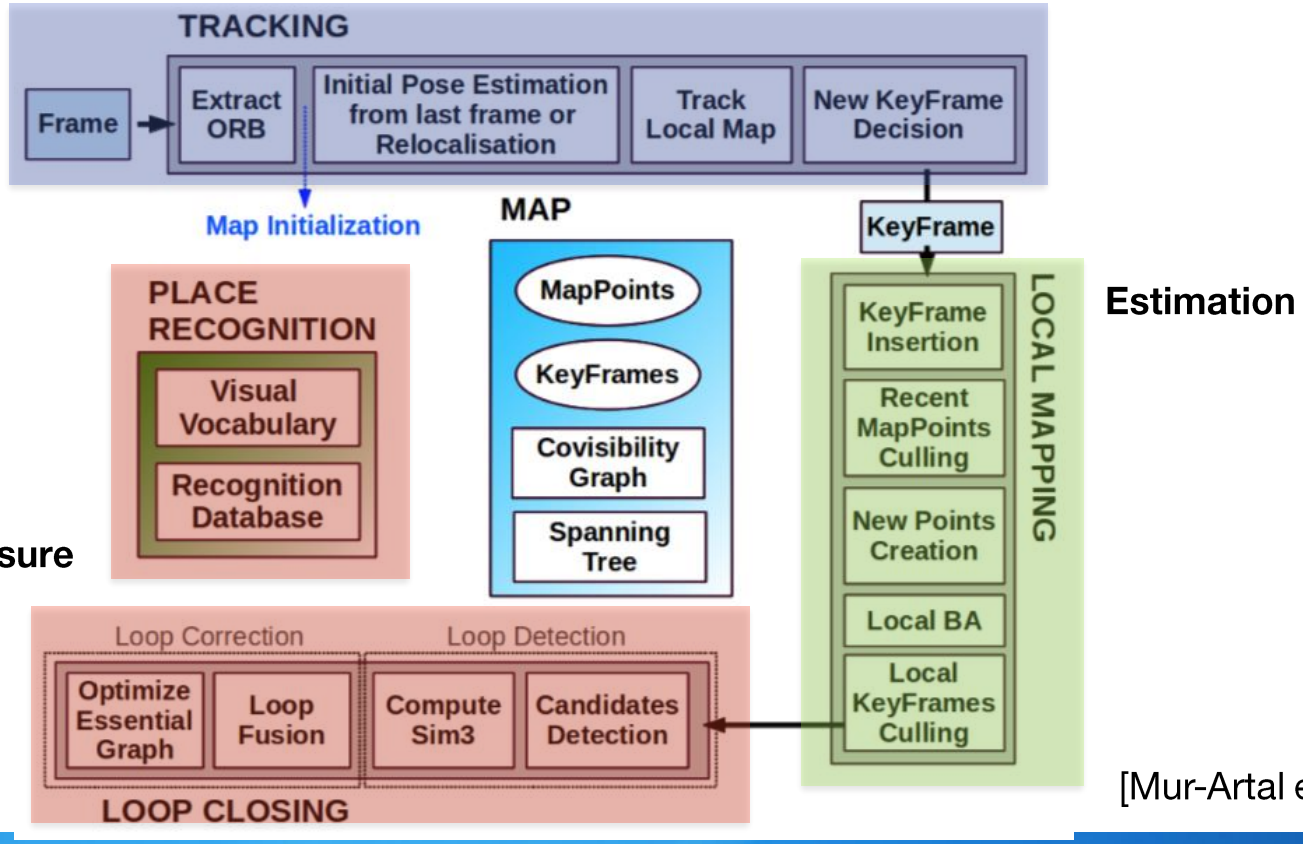
- Feature Exaction
- Feature Matching
- Estimation
- Loop Closure

ORB-SLAM

- Extract ORB features
- Feature Matching/Tracking
- Camera Motion Estimation using PnP
  Local Mapping with Local Bundle Adjustment
- Loop Closure using BoW
  Global Bundle Adjustment

Overview of ORB-SLAM Pipeline

Feature extraction and tracking



Estimation

Loop Closure

[Mur-Artal et al., 2015]

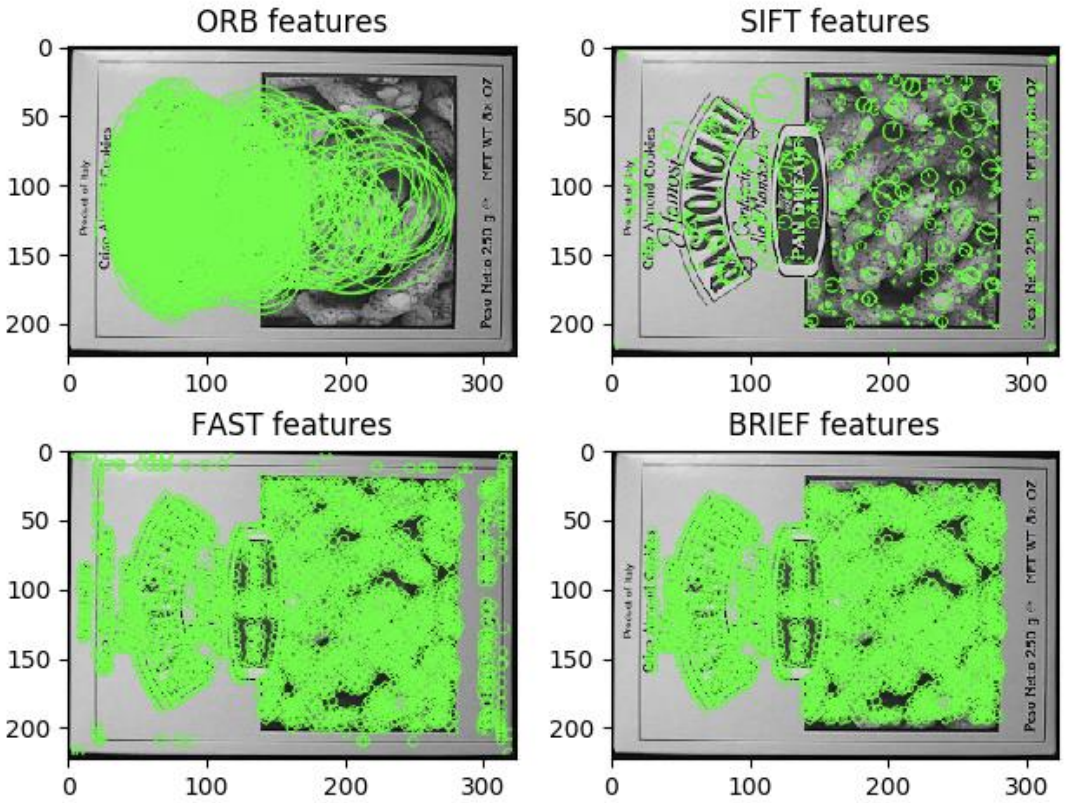
Feature Extraction

- What is a feature?
  - In computer vision, a feature is a (multi-dimensional) representation of an object or part of an object that is relevant for solving a computational task (e.g., feature matching) related to a certain application (e.g., visual SLAM).
  - A feature is typically about a point in an image [Huang and Netravalli, 1994] and its descriptor is about neighboring pixels.
  - Invariance on detection and description: We'd like to have a feature repeatedly detected from different images of the same scene and keeping its description consistent, even under the appearance variation, i.e., changes in scale, illumination, pose.

Feature Extraction

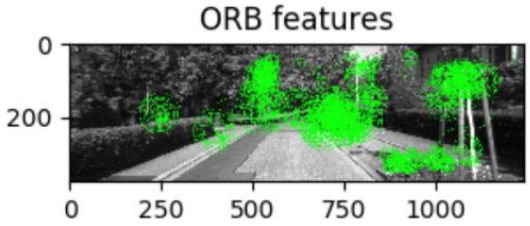
- How can we extract features? Analyze the appearance of a given image to generate a set of features. Use feature extraction methods.
- What methods are available? SIFT, BRIEF, FAST, ORB, SURF, LDB,…many others…
- Read two seminal papers [Lindeberg, 1998] and [Lowe, 2004] of visual feature extraction for more details.
- What makes one method better than another? Invariance in appearance (e.g., scale, orientation, illumination, etc.) change, fast computation
- What do you recommend? ORB (Oriented FAST and Rotated BRIEF) [Rublee et al., 2011]

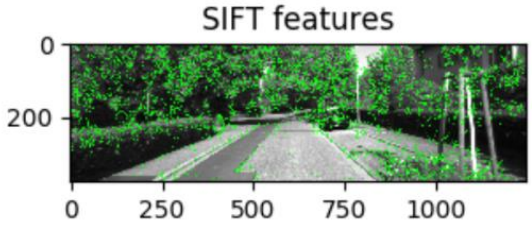
Feature Extraction

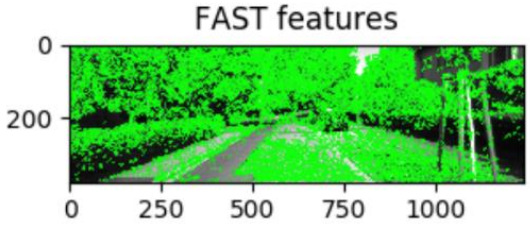


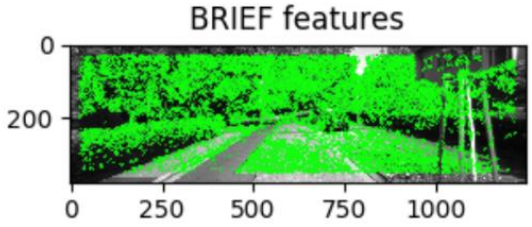
Examples of feature extractions

Feature Extraction

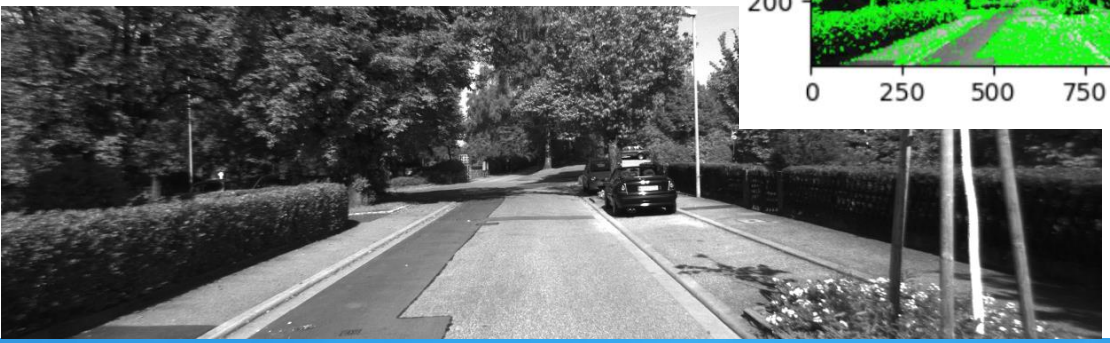








Examples of feature extractions



Feature Matching/Tracking

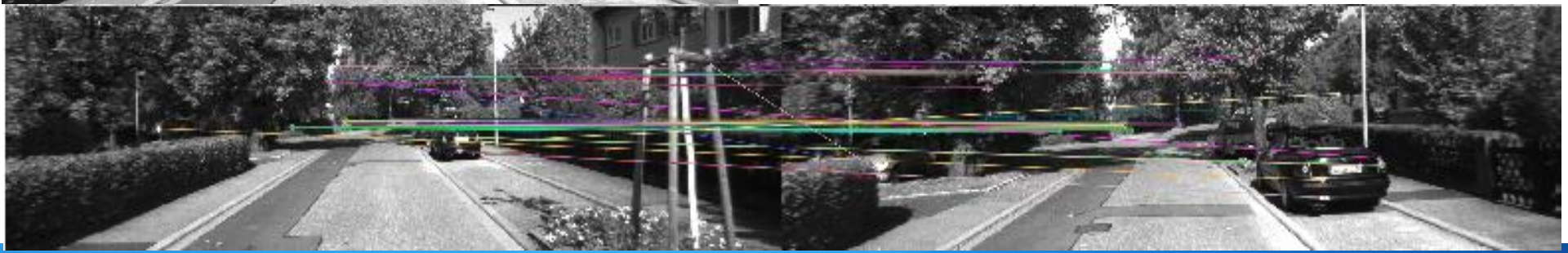
- What do you mean by matching features?
  - Matching features extracted from the previous frames to ones from current frame
  - Often called feature tracking—finding features in one image and then tracking them in the next images using a local search techniques like correlation.
- Why do we do this? To compile a correspondence between features to figure out how the motion of camera has changed
- How we do this? Unless you're inventing a new method for this, use the OpenCV built-in functions

```
import cv2
…

img1 = cv2.imread('./images/000013.png',0) # queryImage
img2 = cv2.imread('./images/000023.png',0) # trainImage

#img1 = cv2.imread('./images/box.png',0) # queryImage
#img2 = cv2.imread('./images/box_in_scene.png',0) # trainImage

# Initiate ORB detector
orb = cv2.ORB_create()

# find the keypoints and descriptors with SIFT
kp1, des1 = orb.detectAndCompute(img1, None)
kp2, des2 = orb.detectAndCompute(img2, None)

# SIFT
sift = cv2.xfeatures2d.SIFT_create()
kp1_sift, des1_sift = sift.detectAndCompute(img1, None)

# FAST
fast = cv2.FastFeatureDetector_create()
kp1_fast = fast.detect(img1, None)

# BRIEF
brief = cv2.xfeatures2d.BriefDescriptorExtractor_create()
kp1_brief, des1_brief = brief.compute(img1, kp1_fast)#None)
…

# create BFMatcher object
bf = cv2.BFMatcher(cv2.NORM_HAMMING, crossCheck=True)

# Match descriptors.
matches = bf.match(des1,des2)

# Sort them in the order of their distance.
matches = sorted(matches, key = lambda x:x.distance)

# Draw first k matches.
img3 = cv2.drawMatches(img1,kp1,img2,kp2,matches[:30], None, flags=2)
```

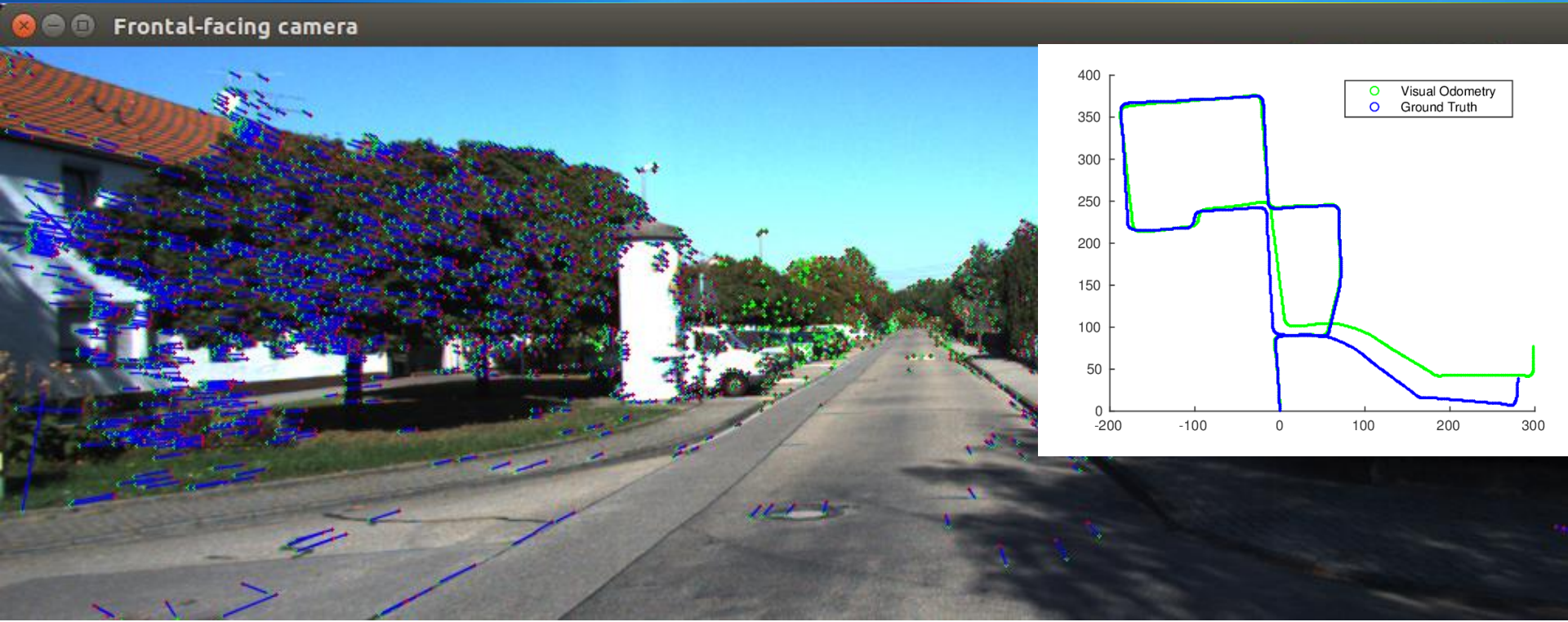
```
void featureDetection(Mat img_1, vector<Point2f>& points1)       {
  vector<KeyPoint> keypoints_1;
  int fast_threshold = 20;
  bool nonmaxSuppression = true;

  FAST(img_1,
       keypoints_1,
       fast_threshold,
       nonmaxSuppression);

  KeyPoint::convert(keypoints_1, points1, vector<int>());
}

  void featureTracking(Mat img_1, Mat img_2, vector<Point2f>& points1,
  vector<Point2f>& points2, vector<uchar>& status)  {

    vector<float> err;
    Size winSize=Size(21,21);
    TermCriteria
  termcrit=TermCriteria(TermCriteria::COUNT+TermCriteria::EPS, 30,
  0.01);

    calcOpticalFlowPyrLK(img_1, img_2, points1, points2, status, err,
  winSize, 3, termcrit, 0, 0.001);

    // remove points if the KLT tracking is failed or if they are outside of
  the image frame
    int indexCorrection = 0;
    for( int i=0; i<status.size(); i++) {
      Point2f pt = points2.at(i- indexCorrection);

      if ( (status.at(i) == 0) ||
            (pt.x<0) ||
            (pt.y<0))       {
        if((pt.x<0)||(pt.y<0))         {
          status.at(i) = 0;
        }

        points1.erase (points1.begin() + (i - indexCorrection));
        points2.erase (points2.begin() + (i - indexCorrection));
        indexCorrection++;
      }
    }
  }
```

KITTI/../sequences/03/image_0/000013.png

KITTI/../sequences/03/image_0/000023.png



Feature Tracking



KLT (Kanade-Lucas-Tomasi) tracker is used to track FAST features for Visual Odometry on KITTI sequence05.

Estimation

- Camera Motion Estimation: Estimate the motion of a camera, rotation and translation, using feature correspondences
  - 2D-to-2D, 3D-to-3D, 3D-to-2D [Scaramuzza and Fraundorfer, 2011]
- How can we do this?
  - Solve the PnP (Perspective n-Point, aka Camera 3D registration) problem: estimate the pose of a (calibrated) camera given a set of *n* 3D points in the world and their corresponding 2D projections in the images by minimizing re-projection error.
    - [Hartley and Zisserman, 2004], [Lepetit et al, 2009], [Moreno-Noguer et al., 2007]
- ORB-SLAM uses g2o (General Graph Optimization) for camera pose estimation.

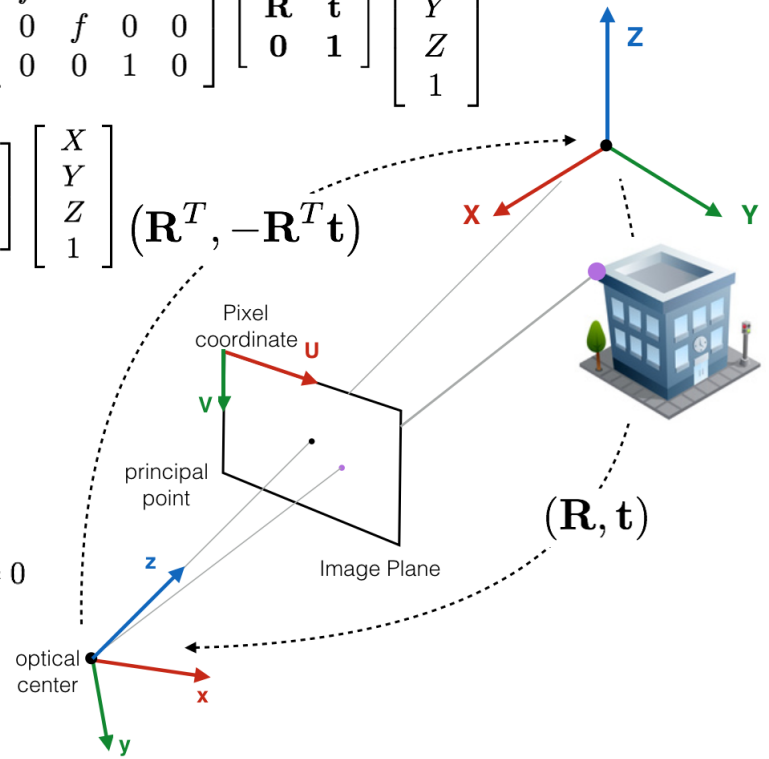
Camera Motion Estimation

$$s\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{1}{p_u} & 0 & u_0 \\ 0 & \frac{1}{p_v} & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & \mathbf{1} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

$$s\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \left(\mathbf{R}^T, -\mathbf{R}^T\mathbf{t}\right)$$

$$\arg\min_{T_k} \sum_i ||p_k^i - \hat{p}_{k-1}^i||^2$$

$$\begin{bmatrix} 0 & 0 & 0 & 0 & -X & -Y & -Z & -1 & x\tilde{v} & y\tilde{v} & z\tilde{v} & \tilde{v} \\ X & Y & Z & 1 & 0 & 0 & 0 & 0 & -X\tilde{u} & -Y\tilde{u} & -Z\tilde{u} & -\tilde{u} \end{bmatrix} \begin{bmatrix} p^1 \\ p^2 \\ p^3 \end{bmatrix} = 0$$



Bundle Adjustment

- What is bundle adjustment? Problem of refining a visual reconstruction to produce jointly optimal 3D structure and viewing parameter (camera pose and/or calibration) estimates.
  - "Bundle": Bundle of light rays leaving each (3d) feature and converging on the camera center of each view

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

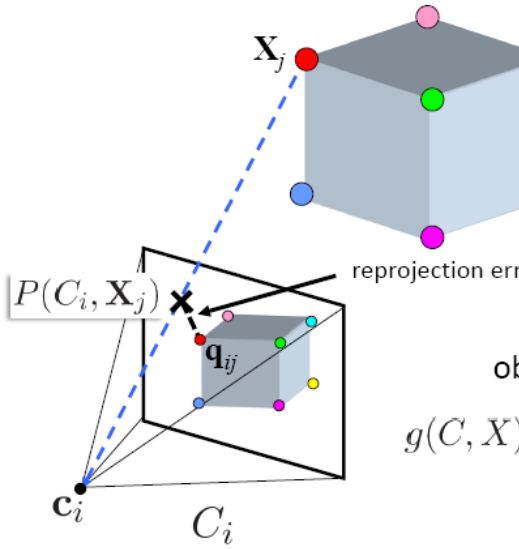
- Wait, didn't we already do this when estimating camera poses?
  - Frame-by-frame, camera pose estimation will quickly "drift" due to accumulated error.

Bundle Adjustment

The objective (or error) function for the bundle adjustment for SLAM is typically a nonlinear least square.

$$E = \min_{\mathbf{x}} ||\mathbf{f}(\mathbf{x}) - \mathbf{b}||^2$$
$$= \min_{\mathbf{x}} \left(\mathbf{f}(\mathbf{x})^T \mathbf{f}(\mathbf{x}) - 2\mathbf{b}^T \mathbf{f}(\mathbf{x}) \right)$$
$$\mathbf{x}_{t+1} = \mathbf{x}_t + \Delta\mathbf{x}$$

$$\frac{\partial E}{\partial \mathbf{x}} = 2\frac{\partial \mathbf{f}(\mathbf{x})^T}{\partial \mathbf{x}}\mathbf{f}(\mathbf{x}) - 2\frac{\partial \mathbf{f}(\mathbf{x})^T}{\partial \mathbf{x}}\mathbf{b} = 0$$
$$\mathbf{f}(\mathbf{x} + \boldsymbol{\Delta}\mathbf{x}) \approx \mathbf{f}(\mathbf{x}) + \frac{\partial \mathbf{f}(\mathbf{x})}{\partial \mathbf{x}}\Delta\mathbf{x}$$
$$\boldsymbol{\Delta}\mathbf{x} = (\mathbf{J}^T\mathbf{J})^{-1}\mathbf{J}^T(\mathbf{b} - \mathbf{f}(\mathbf{x}))$$



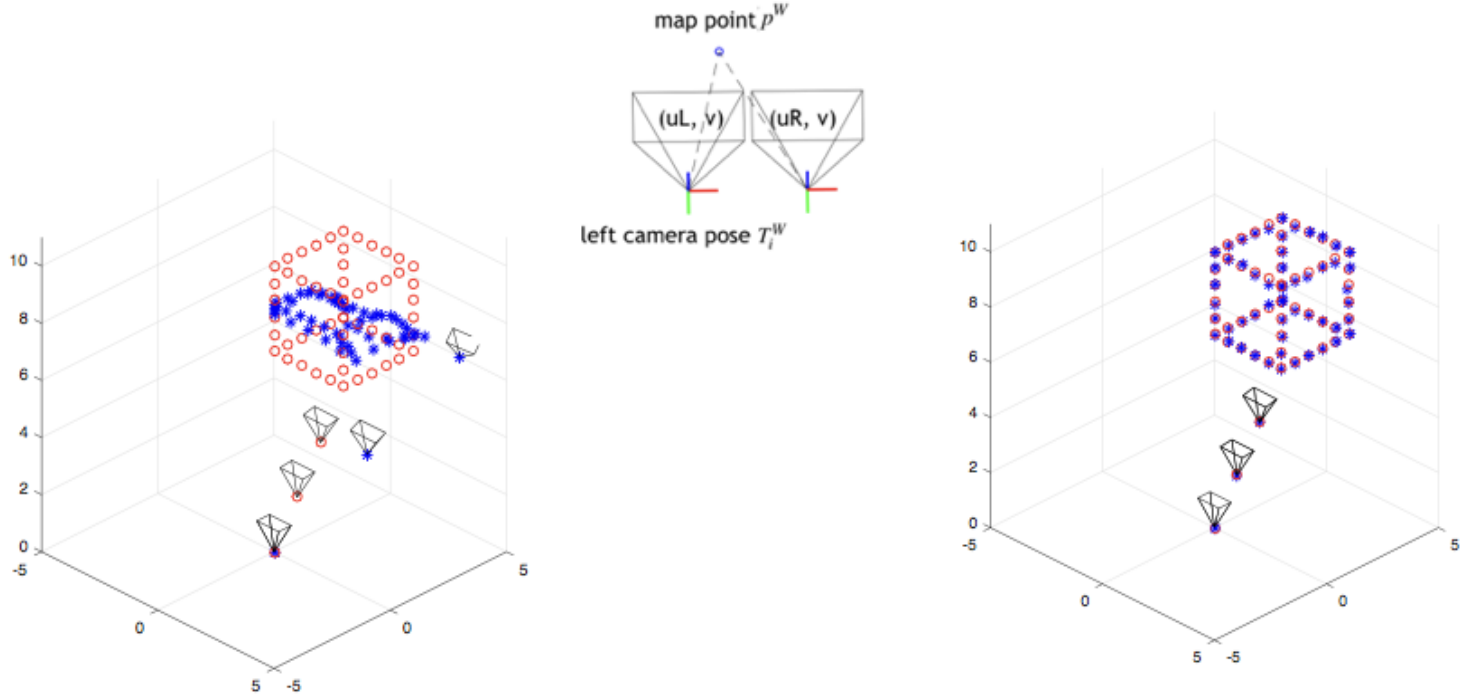
objective function:

$$g(C, X) = \sum_{i=1}^{n}\sum_{j=1}^{m} w_{ij}||\mathbf{q}_{ij} - P(C_i, \mathbf{X}_j)||^2$$

indicator variable:
1 if point *j* is visible in camera *i*
0 otherwise

http://ncsu-geoforall-lab.github.io/uav-lidar-analytics-course/lectures/04_Imagery_Processing.html#/

Bundle Adjustment



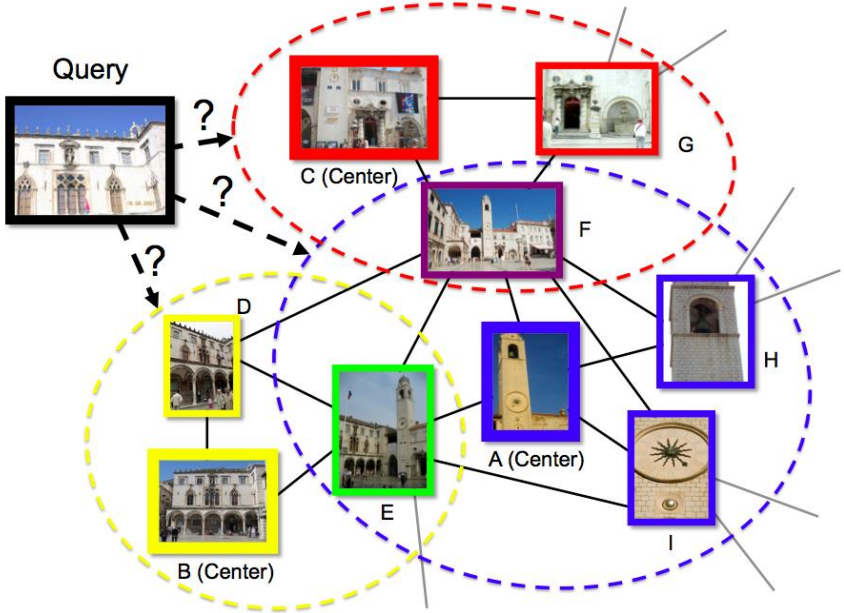
Example of bundle adjustment [Chou et al., 2018]

Loop Closure

- What is a loop closure? Problem of recognizing a previously-visited location (Loop Detection) and jointly optimizing poses of camera and locations of map-points (Global Bundle Adjustment) to complete a map-building task [Bailey and Durrant-Whyte, 2006].
- What technique does ORB-SLAM use for this?
  - Loop Detection: Bags of words representation, DBoW2
    - [Galvez-Lopez and Tardos, 2012], [Mur-Artal and Tardos, 2014]
  - Global Bundle Adjustment

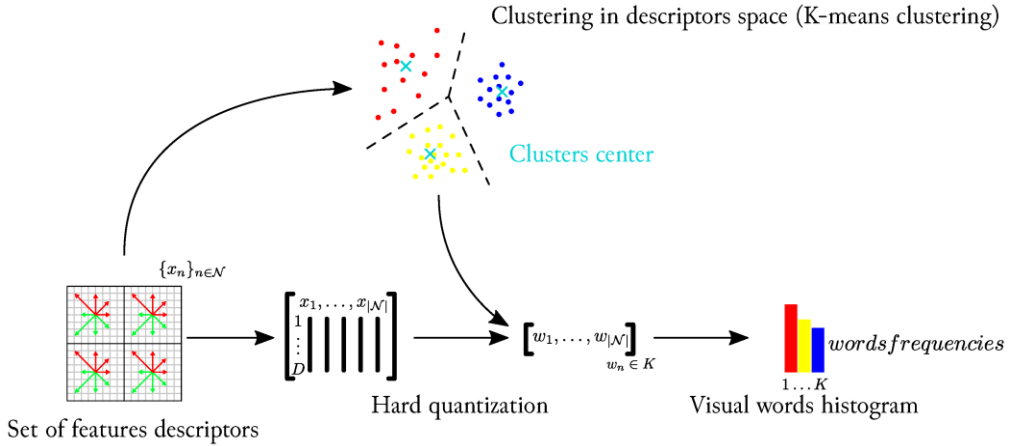
Loop Detection

Q: Where am I if I see the image in the “Query”?



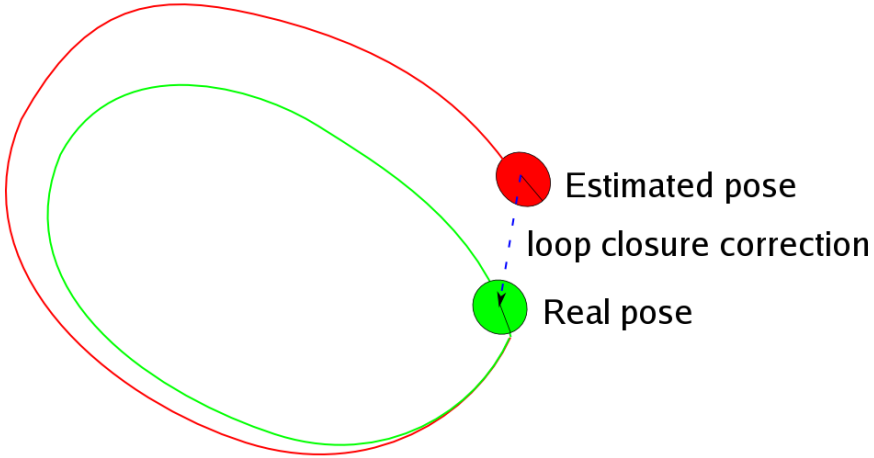
[Graph-based discriminative learning for location recognition.]

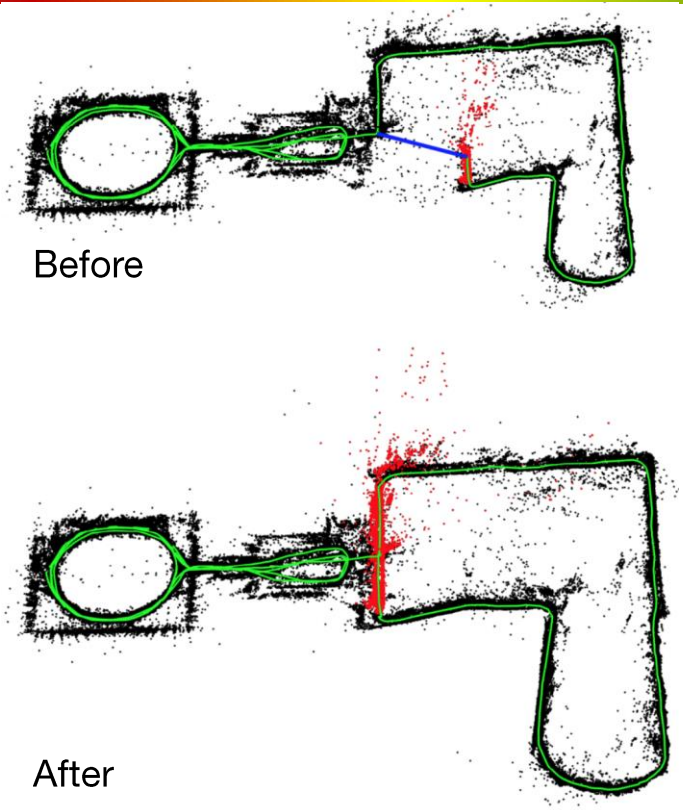
Typical procedure of building Bag-of-Words/Visual Descriptor Vocabulary.



Similarity comparison is usually executed by 1) map-to-map, 2) image-to-image, and 3) image-to-map

Loop Closure





Performance Metrics

- How do we know how good the output of a visual SLAM pipeline is? Compare the outputs with the ground-truth with respect to known performance metrics
  - Ground-truth? (Presumably) most accurate outputs for given problem
    - For outdoor: Can be built by (differential) GPS+IMU measurements
    - For indoor: IMU, odometry, lidar measurements
- Metrics: Measuring the error of trajectories by outputs and ground-truth [Kummerle et al., 2009].

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{k=1}^{n}\left(\mathbf{x}_k^{est} - \mathbf{x}_k^{true}\right)^2}$$

Open Source Packages for Visual SLAM

- ORB-SLAM, http://webdiis.unizar.es/~raulmur/orbslam/
- RT-SLAM (Real Time): SLAM library based on EKF, C++, https://www.openrobots.org/wiki/rtslam/
- MRPT (Mobile Robot Programming Toolkit): Not just for SLAM, but for developing mobile robot S/W deck, C++, https://www.mrpt.org/
- openMVG (MultiView Geometry), https://github.com/openMVG/openMVG
- SLAM++, fast nonlinear least square solver, http://lukas-polok.cz/proj_slam++.htm
- OpenSLAM, https://github.com/OpenSLAM
- SLAMBench 2.0, http://apt.cs.manchester.ac.uk/projects/PAMELA/tools/SLAMBench/

Data set for Visual SLAM

- KITTI, vision benchmark suite (IMU, Velodyne HDL-64E, 2 Gray, 2 Color, …)
  - http://www.cvlibs.net/datasets/kitti/
- RAWSEEDS, multisensor benchmark dataset (Omnidirectional vision, sonar, lidar SLAM)
  - http://www.rawseeds.org/home/category/benchmarking-toolkit/benchmark-problems/
- Datasets for VSLAM comparison, https://afrl.cse.sc.edu/afrl/resources/datasets/
- TUM CV group's dataset
  - Monocular Visual Odometry/SLAM data, https://vision.in.tum.de/data/datasets/mono-dataset
  - RGB-D SLAM dataset and benchmark, https://vision.in.tum.de/data/datasets/rgbd-dataset
  - Data collection by MRPT package, https://www.mrpt.org/robotics_datasets

Challenges

- Monotone Appearance: Difficult to extract a sufficient number of features to represent and track
- Presence of Outliers: Nature of incremental estimation is susceptible to mismatch by outliers
- Map Representation: How to effectively and efficiently represent a resulting map
- Time Varying and Deformable Objects: How to deal with dynamic and non-rigid nature of environments
- Map Update: How to keep a map up-to-date
- HW Failures: How to respond to sensor degradation/failure/inconsistent measurements

Summary

- A crash course for an introduction of a typical visual SLAM pipeline
- (Very briefly) walk-through the ORB-SLAM
- Open source packages, publicly available data sets, performance metrics, challenges

- Send me email, youngwoo.blank.seo@gmail.com, for any further questions/comments!

Resource

- A tutorial on bundle adjustment, https://www.cc.gatech.edu/~choudhar/presentations/BA_Tutorial.pptx
- Bundle Adjustment, http://frc.ri.cmu.edu/~kaess/vslam_cvpr14/media/VSLAM-Tutorial-CVPR14-A13-BundleAdjustment.pdf
- Bundler: Structure from Motion for Unordered Image Collections, http://www.cs.cornell.edu/~snavely/bundler/
- Ceres Solver, ceres-solver.org/index.html
- Datasets for pose graph optimization, https://lucacarlone.mit.edu/datasets/
- DBoW2, https://github.com/dorian3d/DBoW2
- FLANN, https://www.cs.ubc.ca/research/flann/
- g2o, General Graph Optimization, https://github.com/RainerKuemmerle/g2o
- Graph based discriminative learning for location recognition, http://www.cs.cornell.edu/projects/graphlocation/
- ORB, https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_feature2d/py_orb/py_orb.html
- ORB-SLAM, http://webdiis.unizar.es/~raulmur/orbslam/
- Multiple View Geometry in Computer Vision, http://www.robots.ox.ac.uk/~vgg/hzbook/

Resource

- Visual SLAM Tutorial: Bundle Adjustment, https://pdfs.semanticscholar.org/6081/417b95bec070fb842a704044def427f8ef69.pdf
- Real time pose estimation of a textured object, https://docs.opencv.org/3.3.0/dc/d2c/tutorial_real_time_pose.html
- Perspective n Point, https://en.wikipedia.org/wiki/Perspective-n-Point
- Steve Lavalle's lecture on Perspective n Points problem, https://www.youtube.com/watch?v=0JGC5hZYCVE
- Understanding features, https://docs.opencv.org/3.1.0/df/d54/tutorial_py_features_meaning.html
- Frank Dellaert's Visual SLAM Tutorial, http://frc.ri.cmu.edu/~kaess/vslam_cvpr14/
- Frank Dellaert's Visual SLAM Tutorial: Bundle Adjustment, https://pdfs.semanticscholar.org/6081/417b95bec070fb842a704044def427f8ef69.pdf

Reference

- [Bailey and Durrant-Whyte, 2006] Simultaneous localization and mapping: part II. state of the art, *Robotics Automation Magazine*, 13(3): 108-117, 2006.
- [Cadena et al., 2016] Past, present, and future of simultaneous localization and mapping: toward the robust-perception age, *IEEE Trans. On Robotics,* 32(6): 1309-1332, 2016.
- [Chou et al., 2018] Chih-Chung Chou, Chun-Kai Chang and YoungWoo Seo, A structureless approach for visual odometry, In *Proc of the IEEE Intelligent Vehicles Symposium* (IV-18), to appear, 2018.
- [Durrant-Whyte and Bailey, 2006] Simultaneous localization and mapping: part I. the essential algorithms, *Robotics Automation Magazine*, 13(2): 99-110, 2006.
- [Fuentes-Pacheco et al., 2012] J. Fuentes-Pacheco, J. Ruiz-Ascencio, and J. M. Rendon-Mancha, Visual simultaneous localization and mapping: a survey, *Artificial Intelligence Review*, 2012.
- [Hartley and Zisserman, 2003] *Multiple View Geometry in Computer Vision*, Cambridge Univ Press, 2003.
- [Ho and Newman, 2006] Loop closure detection in SLAM by combining visual and spatial appearance, *Journal of Robotics and Autonomous Systems*, 54: 740-749, 2006.
- [Huang and Netravalli, 1994] Motion and structure from feature correspondence: a review, *Proc IEEE*, 82(2): 252-268, 1994.
- [Galvez-Lopez and Tardos, 2012] Bags of binary words for fast place recognition in image sequences, *IEEE Trans on Robotics*, 28(5): 1188-1197, 2012.
- [Lindeberg, 1998] Tony Lindeberg, Feature detection with automatic scale selection, *Int'l Journal of Computer Vision*, 30(2): 79-116, 1998.

Reference

- [Lowe, 2004] David G. Lowe, Distinctive image features from scale-invariant keypoints, *Int'l Journal of Computer Vision*, 60(2): 91-110, 2004.
- [Kummerle et al., 2009] R. Kummerle, B. Steder, C. Dornhege, M. Ruhnke, G. Grisetti, C. Stachniss, On measuring the accuracy of SLAM algorithms, *Autonomous Robots*, 27: 387, 2009.
- [Lepetit et al, 2009] EPnP: an accurate O(n) solution to the PnP problem, *Int'l Journal of Computer Vision*, 2009. (http://icwww.epfl.ch/~lepetit/papers/lepetit_ijcv08.pdf)
- [Lowry et al., 2016] Visual place recognition: a survey, *IEEE Trans on Robotics*, 32(1): 1-19, 2016.
- [Moreno-Noguer et al., 2007] Accurate non-iterative O(n) solution to the PnP problem, In *Proc of IEEE Int'l Conf on Computer Vision* (ICCV-07), pp. 1-8, 2007.
- [Mur-Artal and Tardos, 2014] Fast relocalisation and loop closing in keyframe-based SLAM, In *Proc of Int'l Conf on Robotics and Automation* (ICRA-14), pp. 846-853, 2014.
- [Mur-Artal et al., 2015] ORB-SLAM: A versatile and accurate monocular slam system, *IEEE Trans. On Robotics,* 31(5): 1147-1163, 2015.
- [Nister, 2004] An efficient solution to the five-point relative pose problem, *IEEE Trans on Pattern Analysis and Machine Intelligence*, 26(6): 756-770, 2004.
- [Rublee et al., 2011] ORB: an efficient alternative to SIFT or SURF, In *Proc of IEEE Int'l Conf on Computer Vision* (ICCV-11), pp. 2564-2571, 2011.
- [Scaramuzza and Fraundorfer, 2011] D. Scaramuzza and F. Fraundorfer, Visual odometery, part I: the first 30 years and fundamentals, *IEEE Robotics and Automation Magazine,* 18(4): 80-92, 2011.

Reference

- [Strasdat et al., 2010] Real time monocular slam: why filter?, In *Proc IEEE Int'l Conf Robotics and Automation* (ICRA-10), pp. 2657-2664, 2010.
- [Umeyama, 1991] Least-squares estimation of transformation parameters between two point patterns, *IEEE Trans on Pattern Analysis and Machine Intelligence*, 13(4): 376-380, 1991. (http://www.graphics.stanford.edu/courses/cs164-09-spring/Handouts/paper_Umeyama.pdf)