Lecture 09: Multi-View Geometry III

1 Bundle Adjustement (BA)

Bundle adjustment refers to the **nonlinear, simultaneous refinement of structure and motion (i.e. R, T, P^i).** It is usually used after linear estimation of R and T (e.g. after having used the 8-point algorithm). This, computes R, T, P^i by minimizing the Sum of Squared Reprojection Errors:

$$(R, T, P^i) = \text{argmin}_{R,T,P^i} \sum_{i=1}^{N} \|p_1^i - \pi_1(P^i, C_1)\|^2 + \|p_2^i - \pi_2(P^i, C_2)\|, \qquad (1.1)$$

where C_1, C_2 are the **poses** of the camera in the **world frame**. This can be minimized using *Lavenberg-Marquardt* (more robust than Gauss-Newton to local minima). **It is better to initialize it close to the minimum** (in order not to get stuck in local minima). In the case of multiple views, one includes each view to the error computation.

1.1 Computing Initial Structure and Motion

1.1.1 Hierarchical Structure From Motion

Hierarchical structure from motion can be summarized in the following points:

1. Extract and match features between nearby frames.
2. Identify clusters consisting of 3 nearby frames.
3. Compute SFM for the 3 frames.
   - Compute SFM between 1 and 2 and build pointcloud.
   - Merge 3rd view running 3-point RANSAC between the point cloud and the 3rd view.
4. Merge clusters pairwise and refine (BA) both structure and motion.

An example of hierarchical structure from motion is *building Rome in one day*. In this paper (`http://grail.cs.washington.edu/rome`), parts of the city were reconstructed using 150000 images from Flickr.com.

1.1.2 Sequential Structure From Motion

This works with n views (also called Visual Odometry (VO)). The process can be summarized into the following points:

1. Initialize structure and motion from 2 views (**bootstrapping**).
2. For each additional view:
   - Determine pose (localization).
   - Extend structure (i.e. extract and triangulate new features).

- Refine both pose and structure (BA).

Remark.

- Note that even if Structure From Motion is used as a synonym of Visual Odometry, VO is a particular case of SFM. In fact, VO focuses on estimating the 3D motion of the camera **sequentially** (as the new frame arrives) and in **real time**.
- VO vs. Visual Slam
  - VO: Focus on incremental estimation/**local consistency**. VO sacrifies consistency for real-time performance, without need to take into account all previous history of the camera (as SLAM does).
  - Visual SLAM (VSLAM): Simultaneous Localizazion and mapping. Focus on **globally consistent** estimation. Practically VO + loop detection + graph optimization.

1.1.3 Motion Estimation in Visual Odometry

2D to 2D Motion from Image feature correspondences:

- Both feature points f_{k-1} and f_k are specified in 2D.
- As we have seen in the previous classes, the minimal-case solution involves 5-point correspondences.
- The solution is found by minimizing the reprojection error:

$$\begin{aligned} T_{k,k-1} &= \begin{pmatrix} R_{k,k-1} & t_{k,k-1} \\ 0 & 1 \end{pmatrix} \\ &= \text{argmin}_{T_{k,k-1}} \sum_i \|p_k^i - \hat{p}_{k-1}^i\|^2. \end{aligned} \tag{1.2}$$

Popular algorithms: 8-/5-point algorithms.

3D to 2D Motion from 3D structure and Image correspondences

- f_{k-1} is given in 3D, f_k in 2D.
- This problem is known as *camera resection* or PnP (perspective from n points).
- As we have seen in previous classes, the minimal-case solution involves 3 **correspondences** (+1 for disambiguating the four solutions).
- The solution is found by minimizing the reprojection error:

$$\begin{aligned} T_{k,k-1} &= \begin{pmatrix} R_{k,k-1} & t_{k,k-1} \\ 0 & 1 \end{pmatrix} \\ &= \text{argmin}_{X^i,C_k} \sum_{i,k} \|p_k^i - g(X^i, C_k)\|^2. \end{aligned} \tag{1.3}$$

Popular algorithms: P3P.

3D to 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).
- As we have seen in the previous classes, the minimal case-solution involves **3 non collinear correspondences**.
- The solution is found by minimizing the 3D-3D euclidean distance

$$\begin{aligned} T_{k,k-1} &= \begin{pmatrix} R_{k,k-1} & t_{k,k-1} \\ 0 & 1 \end{pmatrix} \\ &= \text{argmin}_{T_{k,k-1}} \sum_i ||\tilde{X}_k^i - T_{k,k-1} \cdot \tilde{X}_{k-1}^i||. \end{aligned} \tag{1.4}$$

Popular algorithms: Arun 87, ICP, BA.

1.2 Case Study: Monocular Visual Odometry (one camera!)

1.2.1 Bootstrapping

- Initialize structure and motion from 2 views: e.g. 8-point algorithm + RANSAC.
- Refine structure and motion (BA)
- How far should the frames be? If we choose a too small baseline, we'll face large depth uncertainty. If we choose a too large baseline, we'll face small depth uncertainty. One way to avoid this consists of **skipping frames** until the average uncertainty of the 3D points decreases below a certain threshold. The selected frames are called **keyframes**. In general one can define the thumbrule:

$$\frac{\text{keyframe distance}}{\text{average-depth}} > \text{threshold } (10-20\%). \tag{1.5}$$

1.2.2 Localization

- Compute camera pose from known 3D-to-2D feature correspondence.
  - Extract correspondences by solving for R and t (K is known).

$$\lambda \cdot \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = K \cdot [R|T] \cdot \begin{pmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{pmatrix} \tag{1.6}$$

- What is the minimal number of required point correspondences? As we have seen previously:
  - 6 for linear solution (DLT algorithm).
  - 3 for a non linear solution (P3P algorithm).
  - 3 point RANSAC.

1.2.3 Extend Structure

- Extract and triangulate new features.

By denoting the relative motion between adjacent keyframes as

$$T_{k,k-1} = \begin{pmatrix} R_{k,k-1} & t_{k,k-1} \\ 0 & 1 \end{pmatrix}, \tag{1.7}$$

we can concatenate transformations to find the full trajectory of the camera as

$$C_k = T_{k,k-1} \cdot C_{k-1} \tag{1.8}$$

A non-linear refinement (BA) over the last m poses (+visible structure) can be performed to get a more accurate estimate of the local trajectory.

1.2.4 Loop Closure Detection (i.e. Place Recognition)

- Relocalization problem: during VO, tracking can be lost (due to occlusions, low tecture, quick motion, illumination change).
- Solution is to re-localize camera pose and continue.
- Loop closing problem: when go back where you already have been:
  - Loop detection: to avoid map duplication (e.g. same crossing rotated)
  - Loop correction: to compensate the accumulated drift!
- In both cases places recognition is needed (lecture 12).

1.2.5 List of Algorithms

Feature-based Methods

1. Extract and match features (+RANSAC)
2. Minimize Reprojection Error:

$$T_{k,k-1} = \text{argmin}_T \sum_i ||u_i' - \pi(p_i)||^2_\Sigma \tag{1.9}$$

Good: Large frame-to-frame motions, accuracy and efficient optimization of SFM (BA).
Bad: Slow due to costly feature extraction and matching, matching outliers (RANSAC).

Direct Methods (all pixels)

1. Minimize **photometric error**:

$$T_{k,k-1} = \text{argmin}_T \sum_i ||I_k(u_i') - I_{k-1}(u_i)||^2_\sigma, \tag{1.10}$$

where

$$u_i' = \pi(T \cdot (\pi^{-1}(u_i) \cdot d)) \tag{1.11}$$

Good: All information in the image can be exploited. Increasing camera frame-rate reduces computational cost per frame.
Bad: Limited frame to frame motion. Joint optimization of dense structures and motion too expensive.

ORB-SLAM

- Feature based:
  - Fast corner + Oriented Rotated Brief descriptor.
  - Binary descriptor.
  - Very fast to compute and compare.
  - Minimizes reprojection error.
- Includes:
  - Loop closing.
  - Relocalization.
  - Final optimization.
- Real time: 30Hz

LSD-SLAM

- Direct based + Semi-dense formulation:
  - 3D geometry represented as semi dense depth maps.
  - Minimizes **photometric error**.
  - **Separately** optimizes poses and structures.
- Includes:
  - Loop closing.
  - Relocalization.
  - Final optimization.
- Real time: 30Hz

DSO

- Direct based + sparse formulation:
  - 3D geometry represented as sparse large gradients.
  - Minimizes **photometric error**.
  - **Jointly** optimizes poses and structures (sliding window).
  - Incorporate photometric correction to compensate exposure time change
- Real time: 30Hz

SVO

- Direct based :
  - Corners and edgelets.
  - Frame to frame motion estimation.
- Feature based :
  - Frame to Keyframe pose refinement.
- Mapping:
  - Probabilistic depth estimation.
  - Multi camera system
- 400 fps on i7 laptops, 100 fps on smartphone PC

Dense, Semidense, Sparse: Dense and Semidense behave similarly. Dense is only useful if one has motion blur and defocus.

1.3 Understanding Check

Are you able to answer the following questions?

- *Are you able to define Bundle Adjustment (via mathematical expression and illustration)?*
- *Are you able to describe hierarchical and sequential SFM for monocular VO?*
- *What are keyframes? Why do we need them and how can we select them?*
- *Are you able to define loop closure detection? Why do we need loops?*
- *Are you able to provide a list of the most popular open source VO and VSLAM algorithms?*
- *Are you able to describe the differences between feature-based methods and direct methods?*