Final Report for "Online" RGB Bundle Adjustment

Benedict Ruppel
03780391
benedict.ruppel@tum.de

Jan Giedl
03728034
jan.giedl@tum.de

Matteo Wohlrapp
03727947
matteo.wohlrapp@tum.de

Damla Tuana Karasozen
03780712
damla.tuana.karasoezen@tum.de

1. Motivation and Idea

Bundle Adjustment (BA) is an essential optimization technique in computer vision, especially in Structure-from-Motion (SfM) problems, where it is utilized to accurately reconstruct the 3D data of a scene and determine camera motion from a series of images. The process involves refining estimates of camera parameters and 3D features by minimizing the errors between their observed and projected image positions. BA is a foundational element in the field, providing the necessary accuracy and robustness for complex vision tasks [10].
Our goal with this project was to approach BA similarly to an online SfM setting working on RGB-only input data. To allow for this, the image data was considered as real-time input and processed frame-by-frame to simulate accumulating information. We took inspiration from SLAM examples to approach BA as an online problem.
Since we knew from the beginning that this would be an ambitious objective, we tried structuring the project in a way that we still could provide some reasonable results in the end. Despite we ultimately failed to reach our set goal due to the inherent complexity of this task, we implemented multiple initializations and pose estimation approaches and tested different optimizers as our own contributions. In our results, we compare the performance of these methodologies and analyze the accuracy of the results.

2. Related Work

Bundle adjustment is a well-studied problem in 3D reconstruction. Various aspects like camera projection, parametrization, and choice of cost function and optimization have been studied in depth, as described in [10]. With modern hardware, bundle adjustment can be used as the main camera tracking system component for real-time updates, as shown in [10]. Additionally, bundle adjustment can play a prominent role in modern SLAM systems [1].
As BA was considered solved [2], and we propose a solution to utilize BA online, we will focus more on SLAM to draw inspiration from the proposed real-time tracking and mapping systems. In SLAM, the camera trajectory is estimated while reconstructing the environment, introducing a so-called chicken-and-egg problem.
While various sensors can be employed in SLAM, e.g., inertial, optical, or GPS, in visual SLAM (vSLAM), only images from cameras or other image sensors are utilized. This includes monocular, stereo, or RGB-D cameras.
Advantages of vSLAM include inexpensive implementation and wide availability of the sensors.
In monocular vSLAM, the number of sensors is limited to one camera, so the localization and mapping are performed on a single stream of input frames over time.
In early works [3, 4], all input frames were processed to simultaneously estimate the camera pose and reconstruct the environment. However, the keyframe-based approach for mapping introduced in [5, 6] led to increased performance by considering only a selected subset of the input frames for mapping. This detaches the tracking, which benefits from a high frame rate to most accurately reproduce the camera trajectory, from the mapping, which benefits from less frequent but higher quality information.
In addition to using keyframes, PTAM [5] was the first to propose splitting the tracking and mapping into two parallel threads, further improving tracking and mapping.
Since the pose estimation is not perfectly accurate, minor errors in each iteration accumulate over time into a drift. To solve this issue, in more recent solutions [7–9], an additional thread is employed to detect loops in the camera trajectory and perform loop closure.
vSLAM approaches can be further categorized into indirect and direct methods. We do not cover direct SLAM here, because we did not include any dense feature terms in our optimizer.

2.1. Indirect SLAM

Indirect SLAM is based on extracting features in input frames and matching them with observations in previous keyframes. For each 2D image feature match, a 3D point can be created by triangulation with known keyframe poses, generating a sparse point cloud. An initial pose estimate of a new frame is optimized by finding already observed features and minimizing the reprojection error between the projected 3D correspondences of this feature and the observed 2D feature position.

An example of an indirect monocular vSLAM method is PTAM [5], which showcased real-time performance in a small room for applications in augmented reality. While it provided major advances in tracking robustness and accuracy at its time, it suffered from the selected feature descriptor (FAST corners that are matched by patch coherence) and failed to implement proper loop closure.

A more recent method, ORB-SLAM, was introduced in [7]. It is robust to severe motion clutter and supports wide baseline loop closing and relocalization. ORB-SLAM uses the same features for all SLAM tasks: tracking, mapping, relocalization, and loop closing. Its system includes three main threads: tracking, local mapping, and loop closing, operating in parallel. This approach results in an efficient, simple, and reliable system, offering significant advancements in tracking robustness and map quality.

3. Methodology

This section will discuss the details of our bundle adjustment system.

3.1. Data Structures

The foundation of the system lies in the Frame, MapPoint, and SceneMap data structures. Frames represent individual images with associated features, MapPoints are 3D points in the Scene, and the SceneMap organizes these elements into a coherent structure. 2D-3D correspondences are parametrized with shared indices in the list of key points and associated 3D MapPoints of a frame.

- **Frame**: Stores image data, key points, and pose information. Each Frame is linked to observed MapPoints.
- **MapPoint**: Represents a 3D point in the world. It holds positional data, descriptor information, and references to Frames observing it.
- **SceneMap**: Aggregates Frames and MapPoints, facilitating global access and manipulation

3.2. Bundle Adjustment Loop

The core of the code structure is contained in the main bundle adjustment loop, described in 1. In the following sections, we will explain each part in more detail.

Listing 1. Main Bundle Adjustment Process

```
for each Frame in the Scene:
    Detect and Match Features
    if not initialized:
        Initialize SceneMap
    else:
        Associate initial MapPoints
        Estimate Camera Pose
        if #associated < 0.01
            break;
        end if
        if #associated < 0.1
            Cull MapPoints
            Triangulate new MapPoints
            Add Frame to Map
            Update Covisibility Graph
            Search Points in Neighbors
            Optimize Poses and MapPoints
            Cull Frames from Map
        end if
    end if
end for
```

3.2.1 Feature Detection and Matching

Each frame undergoes feature detection and matching. This step is crucial for establishing correspondences between consecutive frames. We implemented SURF with the help of the OpenCV library to detect key points between previous and current frames.

3.2.2 Scene Initialization

The initialization is crucial for setting up the initial geometry of the scene. In our implementation, we leverage the 2D-2D correspondences between two frames to estimate the initial camera pose utilizing the Essential or Homography Matrix. Based on a scoring system measuring the reprojection error of matched key points in both frames, one of the two solutions is selected, or a new pair of frames is considered. Matched key points that do not agree with the estimated pose are marked as outliers. Additionally, initialization using the sensor's depth information from the dataset was implemented for comparison. To ensure the initialization is meaningful, we disregard the initial frames when there are insufficient matches between frames.

3.2.3 MapPoint Association and Creation

Before the pose can be estimated using motion-only BA, initial 2D-3D correspondences must be established. For each match identified between a consecutive frame pair, the algorithm checks if the key point in the last frame is already

associated with a 3D MapPoint and is not marked as an outlier. If the descriptor distance is below a defined threshold, it indicates a good match, and the current Frame is added as a new observation to the MapPoint.

The remaining matches with no MapPoint associated are triangulated using the projection matrices of both frames. The triangulation process involves several sanity checks:

- The baseline between both frames must be larger than 1% of the median depth of 3D points in both frames
- The new 3D point must be in front of both cameras.
- The reprojection error must be within a chi-square threshold to ensure accurate triangulation.
- Scale consistency is checked between frames, ensuring that the ratio of distances aligns with the inverse ratio of the key points' octave scales.

These checks are crucial to filter out bad triangulations and maintain the integrity of the 3D reconstruction. Points passing these checks are added as new MapPoints to the Scene, linked to the current and last frames.

3.2.4 Pose Estimation

We provide several techniques for pose estimation, each with a different approach.

- Perspective-n-Point (PnP): PnP methods are used when correspondences between 3D world points and their 2D projections in the image are known. This approach solves for the camera pose that best aligns these correspondences.
- Motion-only Bundle Adjustment: This is a specialized form of BA that focuses solely on optimizing the camera extrinsics while keeping the map points fixed.
- Essential Matrix and Homography: Similar to the initialization, this approach used 2D-2D correspondences between frames. Our idea was that we could minimize drift because we don't rely on previously calculated 3D points but rather the 2D matches directly.

The constant speed assumption is utilized to initialize the camera pose of each frame, which involves predicting the next pose based on the movement observed in previous frames. This method assumes that the pose change between consecutive frames remains relatively constant.

3.2.5 Covisibility Graph and Search in Neighbors

A covisibility graph is implemented to represent the visibility relationships among different frames in the scene. It is constructed by analyzing map points shared across frames. Each Frame is a node in the graph, and edges are formed based on the shared map points. The weight of an edge represents the number of shared points between two frames. This graph assists in identifying frames with substantial overlap in observed features, which is critical for reliable feature matching and maintaining coherence in the map. Frames with high visibility are more likely to contain consistent and accurate feature matches, facilitating robust 3D reconstruction and enhancing the system's overall performance. Frames with many common map points saved higher weights, and highly correlated frames are checked for matching points. Any points that were found twice in both frames are merged.

3.2.6 Optimization Process

During the optimization phase, the system aims to iteratively refine camera poses and MapPoints to minimize the overall reprojection error. Mathematically, the optimization problem is formulated as follows:

$$\min \sum_{i=1}^{n} \sum_{j=1}^{m} \left(u_{ij} - \pi(C_j, X_i)\right)^2 \tag{1}$$

where u_{ij} is the observed point coordinate in pixel level, X_i represents the i-th 3D point, C_j denotes the camera parameters for the j-th camera, and $\pi(C_j, X_i)$ is the nonlinear projection function that projects the 3D point X_i onto the j-th camera's image plane [2].

The system supports both global and local BA. Global adjustment optimizes all camera poses and MapPoints in the SceneMap, whereas local adjustment only optimizes the poses of the newest frame in the SceneMap, its best ten covisible frames, and the positions of MapPoints perceived by them. As an additional constraint, 2D-3D correspondences of all other frames observing the selected MapPoints are added to the problem while not optimizing their poses. Local BA converges faster since fewer points and frames are optimized, but also reduces the accuracy and does not correct for the drift in scale and pose.

While we tried different rotation parametrizations within our optimization, e.g. a whole rotation matrix or Rodrigues rotations, we opted for the Rodrigues rotations optimization in the final version, as we observed the best results. After processing each frame, global or local bundle adjustment is applied to refine the overall structure.

3.2.7 Culling of Map Points and Frames

Post-optimization, the system performs culling to remove outliers and redundant data. MapPoints with high reprojection errors or insufficient observations are discarded. Similarly, frames that do not contribute significantly to the scene understanding are culled. The criteria for culling MapPoints include the reprojection error, number of observations, and consistency across multiple Frames. For frame culling, the system evaluates the redundancy of each Frame based on its contributions to the overall scene structure.

3.3. Visualization

After the BA process, the generated point clouds can be saved. The system employs the Point Cloud Library (PCL) for visualization. Although we included face visualization capabilities through greedy triangulation and Poisson surface reconstruction, the detail and accuracy of the faces are limited due to the sparse nature of the point clouds.

4. Results

In the following, we show reconstructions for the FreiburgXYZ scene from the TUMRGB dataset and the room0 scene from the Replica dataset.

4.1. Matching

In 1, you can see the key points found in two consecutive frames. The key points matched together are connected by a line to show that they are found again in another frame. We recognized that almost no key points were detected on the black monitor. This area is also empty in 2, with no MapPoints showing up in the final point cloud.

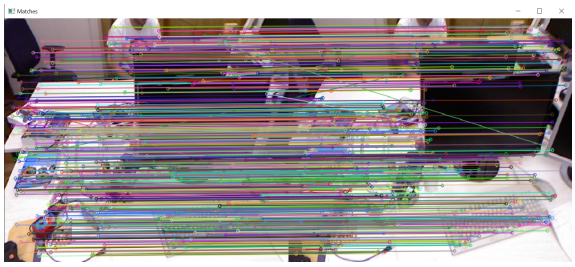

Figure 1. Matches found in the FreiburgXYZ scene

4.2. FreiburgXYZ reconstruction

In the FreiburgXYZ scene 2, you can make out the general structure of the white desk. You could also see where the black monitors should stand and where the keyboard is.

4.3. Replica reconstruction

In 3, the room0 from the Replica dataset is reconstructed. The red dots are part of the camera trajectory from inside the room. You can see the room's boundaries and make out other details, like couches and the table.

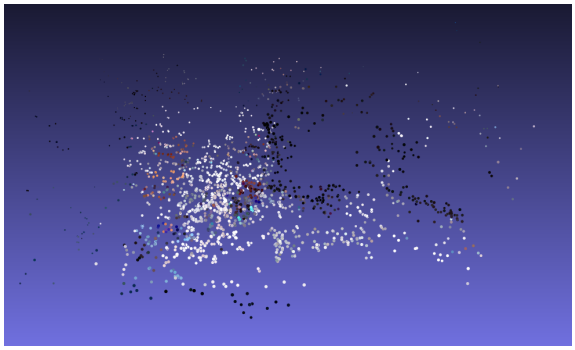

Figure 2. Final point cloud of the FreiburgXYZ scene

Figure 3. Final point cloud of the room0 scene from Replica

5. Analysis

The Absolute Trajectory Error (ATE) calculation is used to measure the accuracy of the estimated trajectory from our results compared to the ground truth trajectory of the dataset. Additionally, we can visualize the reconstruction error by comparing our implementation to a dataset that has ground truth point cloud data. We scale the point clouds to the same size, remove outliers, and then apply Iterative Closest Point (ICP) to align them.

5.1. Reconstruction Error

To visualize the reconstruction error, we compared our algorithm against a dataset with the ground truth data. For this, we used the Replica room0 dataset. The estimation is obtained based on all 2000 RGB images in the scene. In 4, you can see the estimated point cloud in green and the target cloud in red. The first image shows the alignment inside of the room. While our point cloud is a lot sparser, you can see that we fit the ground truth data reasonably well, especially in areas allowing for good keypoint matches, like the image on the wall or the furniture. The approximation of the couches and the table work best in the scene. The second image in 4 shows the alignment with the overall room dimensions. Here, we did remove outliers before aligning the two clouds. However, the core of the key points seems to have a similar dimension ratio compared to the original dataset.

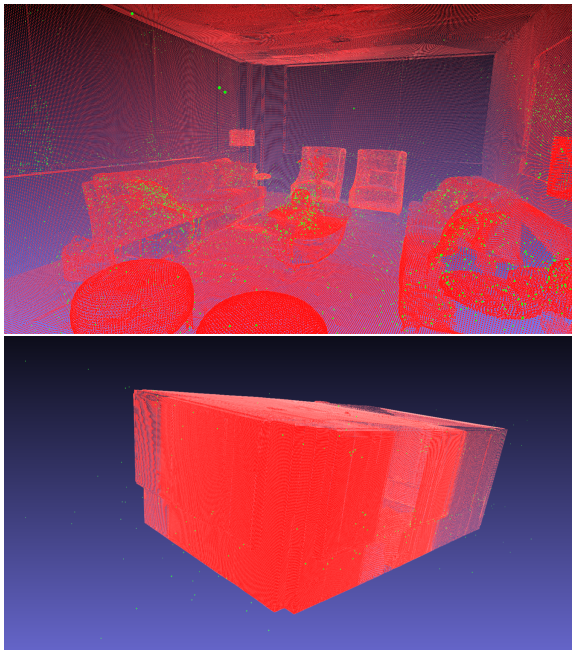

Figure 4. Image of the aligned target and estimation point cloud

5.2. Absolute Trajectory Error (ATE)

Error rates are calculated and scaled by using a modified version of the TUM-RGBD benchmark tool [7].

Table 1 shows that ground truth initialization gives slightly better results than 2D-2D initialization for all pose estimation techniques except localBA. While motion-only BA and PnP seem to provide similar good results, localBA and 2D pose estimation techniques perform worse, with 2D pose estimation being the worst performant. Additionally, you can see that, especially for only 2D-2D estimation and the PnP method, frame culling doesn't increase accuracy even though it does remove frames from the scene and, therefore, speeds up computation.

	GT Init	**2D2D Init**	**2D2D Init + Frame Culling**
MoBA	0.030112	0.043800	0.037608
2D2D	0.304378	0.324296	0.361948
PnP	0.046074	0.051962	0.062798
localBA	0.185375	0.174784	0.145844

Table 1. ATE-RMSE results on FreiburgXYZ dataset in meters

6. Conclusion

We presented our project with the ambitious goal of implementing a monocular visual SLAM system. While our method is able to estimate the general outline of target point clouds, it still fails to produce adequate and precise reconstructions and does not achieve real-time performance.

From initial issues, like a high number of inaccurate map points resulting in bad reconstruction or too few map points, which led to a failure of tracking, we proceeded to more involved problems, like checking optimizer functionality or finding ways to avoid the drift in scale and position. Many of the issues were solved by introducing various features, each improving the results slightly but also adding more and more complexity to our system. After all, this complexity makes it difficult to fine-tune the system to produce reasonable results since there are many adjustable parameters. The problems that remain are mainly related to accumulated scale and drift and also to too few observations per map point and can be most certainly fixed by further refining the process of how map points are initially associated with frames and by introducing measures to perform loop closure. Additionally, real-time performance would be feasible through parallelization and running mapping and tracking in their own threads. Another improvement could be achieved by adding more post-processing to the produced point clouds.

In the end, various methodologies to improve our system's performance were explored, including multiple initializers and estimation algorithms. This provided a valuable opportunity to compare the performances of different techniques in terms of the robustness and accuracy of our system.

In summary, while our project has limitations, it has provided valuable insights regarding a bundle adjustment implementation.

References

[1] Alvaro Parra Bustos, Tat-Jun Chin, Anders Eriksson, and Ian Reid. Visual slam: Why bundle adjust? In *2019 In-*

ternational Conference on Robotics and Automation (ICRA). IEEE, May 2019. 1

[2] Yu Chen, Yisong Chen, and Guoping Wang. Bundle adjustment revisited. *CoRR*, abs/1912.03858, 2019. 1, 3

[3] Andrew J. Davison, Ian D. Reid, Nicholas D. Molton, and Olivier Stasse. Monoslam: Real-time single camera slam. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(6):1052–1067, 2007. 1

[4] E. Eade and T. Drummond. Scalable monocular slam. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 1, pages 469–476, 2006. 1

[5] Georg Klein and David Murray. Parallel tracking and mapping for small ar workspaces. In *2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 225–234, 2007. 1, 2

[6] E. Mouragnon, M. Lhuillier, M. Dhome, F. Dekeyser, and P. Sayd. Real time localization and 3d reconstruction. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 1, pages 363–370, 2006. 1

[7] Raúl Mur-Artal, J. M. M. Montiel, and Juan D. Tardós. Orb-slam: A versatile and accurate monocular slam system. *IEEE Transactions on Robotics*, 31(5):1147–1163, 2015. 1, 2, 5

[8] Katrin Pirker, Matthias Rüther, and Horst Bischof. Cd slam - continuous localization and mapping in a dynamic world. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3990–3997, 2011. 1

[9] Hauke Strasdat, J. Montiel, and Andrew Davison. Scale drift-aware large scale monocular slam. volume 2, 06 2010. 1

[10] Bill Triggs, Philip F. McLauchlan, Richard I. Hartley, and Andrew W. Fitzgibbon. Author manuscript, published in "n/p" doi: 10.1007/3-540-44480-7_21 bundle adjustment — a modern synthesis. 2010. 1