Structure from Motion with known Correspondences

Hoang Tran
Linköping University
hoatr725@student.liu.se

Matheus Bernat
Linköping University
matvi959@student.liu.se

Viktor Ivarsson
Linköping University
vikiv480@student.liu.se

Yunhee Kim
Linköping University
yunki172@student.liu.se

Abstract

A structure from motion system was developed to reconstruct a 3D model from a data set with known correspondences and a known camera intrinsic matrix. Using the known information, a reference camera was chosen and other camera poses were iteratively estimated in relation to the reference camera using the known correspondences. New point correspondences were also triangulated in each iteration to create a sparse 3D point cloud. Additionally, bundle adjustment was performed during each iteration to minimize the sum of mean reprojection error for each view. Finally, a 3D mesh was made to visualize the results and the camera poses were compared to ground truth data to evaluate the system.

1. Introduction

This report describes a system that performs 3D reconstruction of a static scene given images of this scene taken from different angles. This task is thoroughly studied in the field of computer vision and known as *Structure from Motion* (SfM).

In the specific solution to SfM that this report describes, some assumptions are made. Firstly, the 3D points depicted in the images are fixed relative to the world coordinate system, i.e., the scene is static. Secondly, likely correspondences between views are known. Thirdly, the internal camera calibration is also known. The 3D positions of the interest points and the camera poses of each view are sought.

The approach used to solve SfM in this report is known as an *incremental* approach. Initially, a pair of views is chosen and the essential matrix between the cameras is found, which then gives the camera pose of the second camera. Then, for each new image there is, correspondences between the new image and the old ones are found, from which one can find the camera pose of the new camera. In order to make the reconstruction pipeline more robust, an optimization problem is solved to refine the camera poses and the position of the 3D points, what is called bundle adjustment. See the described pipeline in figure 1.

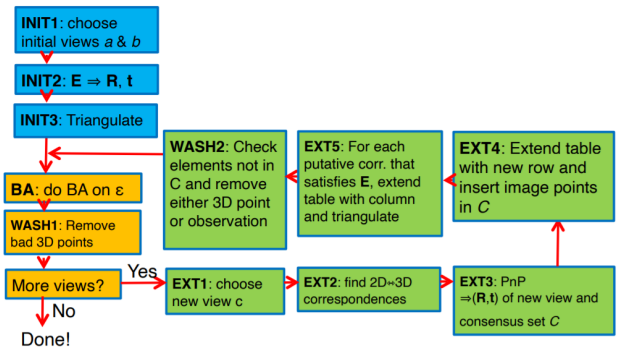

Figure 1: Incremental SfM pipeline.

The responsibilities in the group were divided in the following manner: Yunhee Kim was responsible for the initialization steps, Matheus Bernat and Viktor Ivarsson for the extension steps, and Hoang Tran for the bundle adjustment.

The data set used by the authors throughout the solution of this task was the noise-free *dinosaur sequence* by the Visual Geometry Group at Oxford University. It consists of 36 images of a dinosaur toy standing on a turn-table, where all images are taken with the same camera. The ground truth is provided as well as 676 interest points and point correspondences between images. See an image from the data set in figure 2.

2. Initialization

The two initial views have to be chosen in a way that meets the following criteria: they have to form a long enough baseline and have enough corresponding points. After trying some different pairs from 36 given cameras and

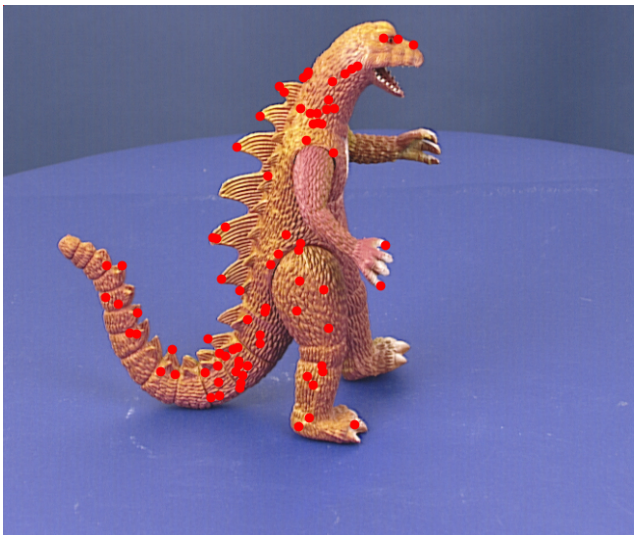

Figure 2: Dinosaur toy and interest points.

comparing the result of the triangulation, we decided to pick camera 1 and 3, for future reference denoted as view_A and view_B.

The first camera, which is view_A, is set to be the origin of the world coordinate system. The camera extrinsic of view_B, which is $[R_B \ \ t_B]$, can be estimated by using the essential matrix E. The essential matrix can be achieved by using the five-point algorithm [1], however, due to its complexity in implementation, a function *findEssentialMat* from the OpenCV library [2] is used. We set the function's parameters to use the random sample consensus algorithm (RANSAC) [3] with the threshold equal to 1 in order to find the best E.

Now that E has been estimated, the rotation R_B and the translation t_B can be calculated with the internal constraint.

$$E = R_B^T [t_B]_\times \tag{1}$$

t_B is the right null vector of E. Therefore, it can be determined from the singular value decomposition (SVD) of E by taking the right most column in the V matrix. However, t_B is determined only up to an undetermined scaling. Therefore, we assume two possible translation vectors, t_1 and t_2 as follows:

$$\|t_1\| = \|t_2\| = 1, \ t_2 = -t_1, \ where \ t_1, t_2 \in t_B \tag{2}$$

To guarantee that $R_B \in SO(3)$, we used the special singular value decomposition (SSVD). There are two possible rotation matrices of R_B, which are R_1 and R_2 and they are defined as follows:

$$R_1 = VW^TU^T, R_2 = VWU^T, W = \begin{pmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{3}$$

Where V and U^T are achieved by SSVD of E and W is a rotation matrix.

The two possible rotation matrices R_1 and R_2 and two possible translation vectors t_1 and t_2 satisfy the internal constraint described in equation 1. As a consequence, they form four configurations. This ambiguity is resolved by checking if the 3D points are in front of both cameras view_A and view_B. See algorithm 10.3 in IREG [4].

3. Adding Views

In this section we describe the extension of new camera views to the reconstruction pipeline. In summary, this step adds a hitherto unused image to the reconstruction pipeline by finding the correspondences between it and the already processed cameras, finds the camera pose of the new camera and triangulates the correspondences to find the sought 3D position of the interest positions.

In the first step, a new camera is chosen. This is done by a simply choosing a camera that lied close (i.e. had a significant number of correspondences) to one of the already processed cameras. This step was simplified in this assignment as the id numbers of the cameras corresponded to the time in which the picture was taken, so the next camera is chosen as the camera with the closest id to one of the processed cameras. The 2D to 3D correspondences between the cameras were also found in this step by simply picking the already triangulated 3D points that were visible by both cameras.

Given the set of 3D points and its believed 2D projections, the problem called *Perspective-n-Point* [5] (PnP) to estimate the pose of the new camera was employed. A RANSAC scheme was used when estimating the camera pose and the associated consensus set, where the consensus set is defined as 3D points whose reprojection errors are smaller than the threshold of 3 pixels. In order for the PnP-RANSAC to perform fast, the amount of points used by RANSAC when estimating the camera pose is the minimal amount of 4 points required by the OpenCV function *solvePnPRansac*.

Now that the camera matrix of the new view is found through PnP, the fundamental matrix between the previous camera and the newly added camera is calculated. Then, the correspondences of the cameras are triangulated in an optimal maximum likelihood sense and added to the set of triangulated 3D points.

Note that the triangulated points can still be outliers since the epipolar constraint is only necessary and not sufficient.

A reasonable criterion to decide if a 3D point is an outlier or not is to check the *age* of the triangulated points (i.e. for how many iterations of the reconstruction pipeline the 3D point has existed in the set of already triangulated points). Therefore, for each 2D to 3D correspondence that is not in the consensus set, either the 3D point is removed if it is visible for only few cameras, or the 2D projection of the 3D point is set as invisible from the new camera if the 3D point is enough aged.

4. Bundle Adjustment

Bundle adjustment is a step performed every time a new view has been added to the system. What it essentially does is adjust the camera poses and the positions of the 3D-points to minimize the reprojection error of these points to the views. This is done to reduce drift from small, continuous errors. To perform this, the following error function, sum of mean squared reprojection errors, is minimized:

$$\epsilon(R,t,x) = \sum_{j \in Q} \frac{\sum_{i \in P} v_{ij} d_{PP}^2(y_{ij}, K[R_j|t_j]x_i)}{N_j}, \quad (4)$$

where P is the set of 3D points, Q is the set of views, v_{ij} is the visibility of point i in view j, y_{ij} is the point i in pixel coordinates in view j, $K[R_j|t_j]$ is the camera matrix for view j, x_i are the 3D coordinates for point i and N_j is the amount of visible correspondences in view j. This was implemented in *PyTorch* using *Adaptive Moment Estimation* [6] as an optimizer. The learning rate was set to 0.001 and we did not optimize over the first view as the reference view was chosen to be set at the origin. A maximum amount of iterations was set at 100 and a threshold of 0.001 in loss between iterations was set to speed up the pipeline. After summing up the mean square reprojection error for all 36 views, the loss ϵ ended up at around 3.6 to 3.7, meaning each view in average had a mean reprojection error around 0.1 pixels.

An additional step correlated to the bundle adjustment is the WASH1 step. Here the mean reprojection error for each 3D point is calculated and if it reaches a certain threshold it is removed. Additionally, if a 3D point has moved far during bundle adjustment, it is also removed. In our case we found that a good threshold for the mean reprojection error was around 2 pixels. However, for the movement in the bundle adjustment step we did not see much point in lowering it further than 0.2, where it still would not remove any points when running. Therefore, in our case, it exists more as a fail safe to catch bad points that the washing steps might eventually miss.

5. Evaluation

The estimated 3D points and the camera poses are related to the ground-truth 3D points and camera poses by a similarity transformation T, which consists of a rotation, a translation and a scaling:

$$T \sim \begin{pmatrix} s\mathbf{R} & \bar{\mathbf{t}} \\ \mathbf{0} & 1 \end{pmatrix}. \quad (5)$$

The rotation is estimated using the algorithm for solving the *special Orthogonal Procrustes Problem* (SOPP) in order to guarantee that the estimated R is in $SO(3)$ even for significant levels of noise. See algorithm 15.2 in IREG [4]. The scaling and the translation are then calculated according to the closed formulas in the paper by Horn [7]. The camera poses are then mapped using the transformation T.

The individual errors of the mapped estimations of the camera positions relative to the ground truth are defined as

$$e_k = \left\| t_{gt,k} - t_{map,k} \right\|. \quad (6)$$

The individual angular errors of the camera orientations using 3D angular error are

$$a_k = 2 \arcsin\left(\frac{\left\| R_{gt,k} - R_{map,k} \right\|}{\sqrt{8}}\right). \quad (7)$$

The errors are summarized for the entire set of poses using the mean-absolute errors (MAE) for the two errors above, where MAE is defined as

$$MAE = \frac{\sum_{i=0}^{N} \|e_k\|}{N}. \quad (8)$$

6. Meshing

The 3D mesh was constructed with Meshlab using Screened Poisson surface reconstruction [8]. In order to generate the mesh with Meshlab we structured the point cloud according to the *.ply* file standard [9]. An oriented point cloud as well as an estimated color for each point was calculated. To get an oriented point cloud we estimate the normal at all 3D points with the Python library Open3D [10]. In order to decide the signs of the estimated normal vectors we added a refinement step which checked the constraint:

$$\bar{\mathbf{v}}^T \bar{\mathbf{n}} < 0, \quad (9)$$

where $\bar{\mathbf{v}}$ is the optical axis for a camera and $\bar{\mathbf{n}}$ is the estimated normal vector for a point that is visible by the camera. Normal vectors which conformed with less than 50 percent of the cameras that could see the corresponding point were multiplied by -1 to flip the direction.

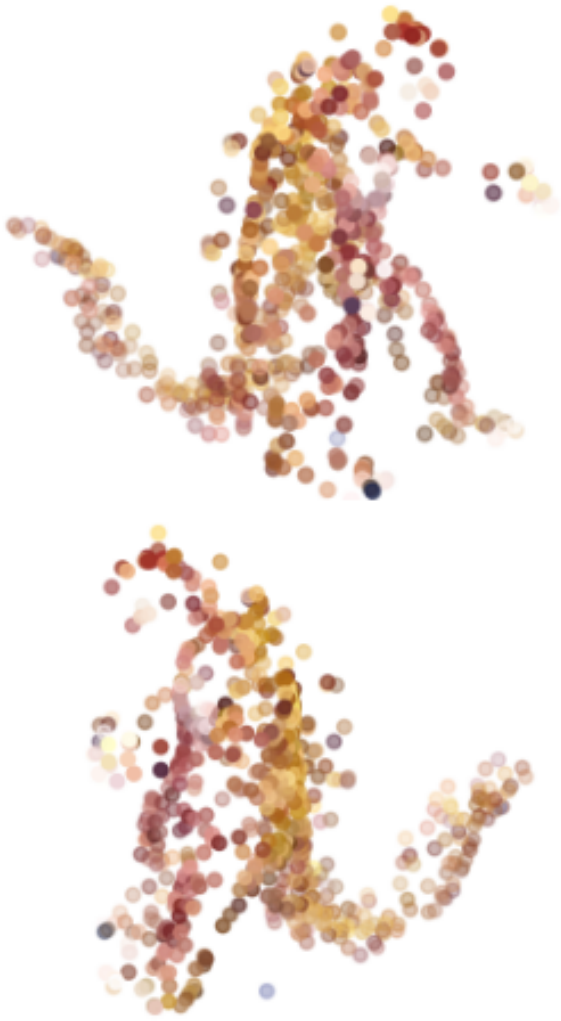

Figure 3: Triangulated interest points.

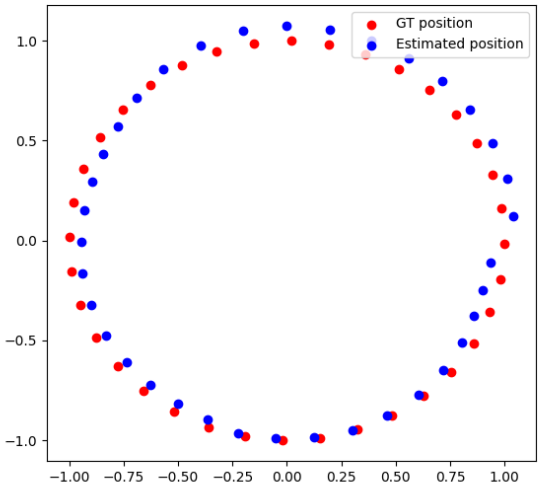

Figure 4: Ground-truth camera poses in red, estimated in blue.

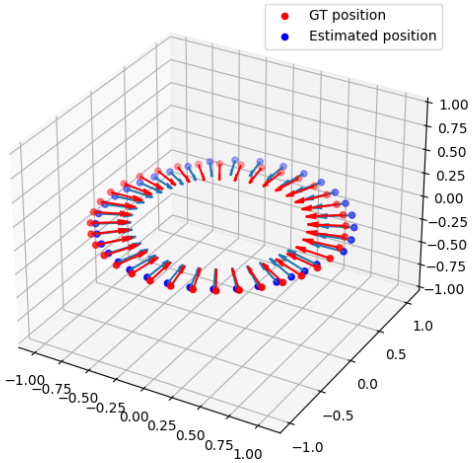

Figure 5: 3D plot of ground-truth camera poses in red, estimated in blue, as well as their respective viewing directions.

7. Results

As a result from the pipeline, the camera poses and the 3D positions of the interest points are found. See in Figure 3 the triangulation of the sparse 3D points, and in Figure 4 and Figure 5 the estimated camera poses compared to the ground-truth camera poses. In Figure 6 the final result of screened Poisson surface reconstruction can be seen. The general shape and color of the dinosaur can be distinguished but due to it being a sparse point cloud, it lacks some distinct features of the original model.

The translation error according to equation 6 was 0.06297, and the rotation error according to equation 7 was 0.06546. However, these values should be taken with a grain of salt as the evaluation was not done entirely by the correct procedure. There was a problem encountered where the ground truth rotational matrices were flipped 180 degrees along the relative x-axis for each camera, hence they were all flipped before evaluating the rotational MAE. This might be due to a bug in our implementation of camera resectioning or a misinterpretation of the camera positions somewhere in the pipeline. Finally, the whole reconstruc-

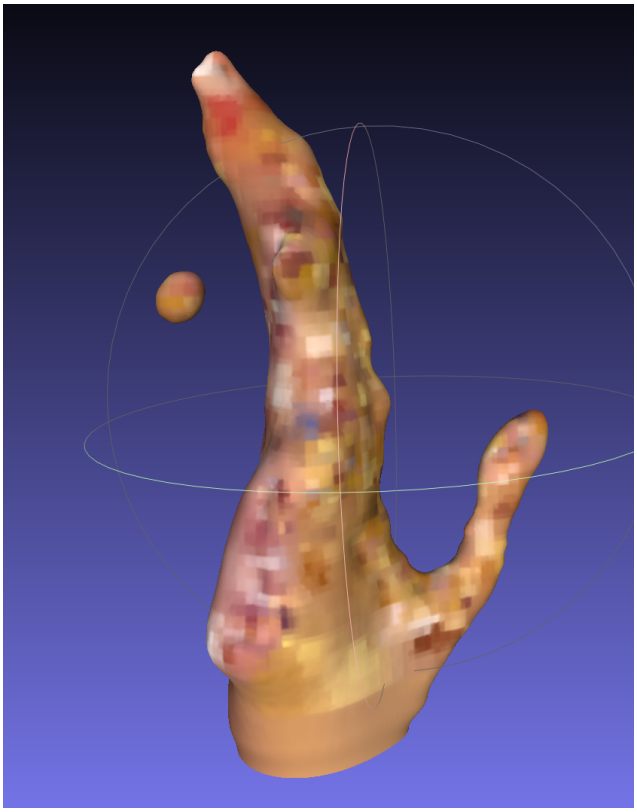

Figure 6: Resulting mesh viewed in Meshlab.

tion pipeline takes consistently one minute to run, which is considered as relatively fast by the authors.

8. Conclusions

The SfM problem can be solved robustly with the suggested structure. Slight errors turn up due to estimations and perhaps the resolution of the data set. Implementing bundle adjustment through PyTorch proved to be simple and fast. Different optimizers were tested but Adaptive Moment Estimation was deemed as a suitable choice. To further improve this one could look into applying an adaptive learning rate to lower the final reprojection error. A further improvement to the 3D reconstruction is to densify the achieved point cloud in order to obtain a better mesh.

References

[1] David Nistér. An efficient solution to the five-point relative pose problem. *IEEE transactions on pattern analysis and machine intelligence*, 26(6):756–770, 2004.

[2] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[3] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[4] Nordberg K. *Introduction to Representations and Estimation in Geometry*. 2018.

[5] Perspective-n-point. `https://en.wikipedia.org/wiki/Perspective-n-Point`. Accessed: 2021-05-20.

[6] Jimmy Lei Ba Diederik P. Kingma. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 2015.

[7] Berthold K. P. Horn. Closed-form solution of absolute orientation using unit quaternions. *Pattern Recognition*, 4(4), 1987.

[8] Hugues Hoppe Michael Kazhdan. Screened poisson surface reconstruction. 2012.

[9] Ply - polygon file format. `http://paulbourke.net/dataformats/ply/`. Accessed: 2021-05-18.

[10] Open3d, a modern library for 3d data processing. `http://www.open3d.org/`. Accessed: 2021-05-18.