VAMR mini project report

Nicola Carrino(19-948-967)
Sizhe Tian(19-952-084)

January 2021

1 Overview

The implementation of the pipeline can be separated into initialization and continuous operation. In the initialization part, the first 3D-2D correspondences are extracted based on the bootstrap frames in order to have the initial camera poses and landmarks. During the continuous operation, the camera pose at each frame is estimated based on the 3D-2D correspondences using P3P. New keypoints and landmarks are regularly added. At each frame, a nonlinear optimization step is performed to refine the pose estimate.

As an additional feature, we also implemented sliding window bundle adjustment based on the keyframes selection.

Finally, we recorded a small video around ETH with our smartphone and we used this video to build our own dataset which was run on our vision pipeline. The camera of the smartphone is calibrated using camera calibration toolbox to obtain the K matrix.

2 Implementation

2.1 Definition

The VO pipeline is implemented in a Markovian way. In each step, it takes the state of last frame as well as the image of two subsequent frames as input and it outputs the state of the current frame.

We defined the state of frame i $\mathcal{S}^i$ to be:

$$\mathcal{S}^i = [\mathcal{P}^i, \mathcal{X}^i, \mathcal{C}^i, \mathcal{F}^i, \mathcal{T}^i, \mathcal{V}^i] \tag{1}$$

where $\mathcal{P}^i$ is the set of keypoints at frame i, $\mathcal{X}^i$ is the set of landmarks at frame i, $\mathcal{C}^i$ is the set of candidate keypoints at frame i, $\mathcal{F}^i$ is the set of candidate keypoints pixel values at their first observation, $\mathcal{T}^i$ contains the camera pose of the candidate keypoints when the candidate keypoints are observed for the first time, $\mathcal{V}^i$ contains the current state camera pose.

2.2 Initialization

We first detect Harris corner at the first bootstrap frame, then we track these corners at the second bootstrap frame(choosen differently for each dataset) using KLT. All the tracked corner points construct the initial set of keypoints. Using these keypoints, we estimate the fundamental matrix using RANSAC. Based the inlier keypoints, the relative camera pose between two camera is calculated. The outlier keypoints in the fundamental matrix estimation are discarded. The camera pose at first bootstrap frame will be set as reference. Its orientation is set to be identity matrix, and its location is set to be at $[0, 0, 0]$. The distance between the camera pose of two bootstrap frames is set to be 1.

Based on the set of matched keypoints and two camera poses, we can triangulate the 3D location of the keypoints in world frame. These 3D points in world frame are the landmarks that we will use. At this point, some landmarks could lie behind the camera, which is invalid, or some landmarks could be really far from the camera, which makes the landmark location unreliable. Thus, we applied a selection process to this landmark set. All the landmarks that are behind the camera, or too far from the camera will be eliminated. The position of the landmarks with respect to the camera can be computed transforming the landmarks in world frame to landmarks in camera frame using the homogeneous transform T_{CW}. The criteria used to decide if the landmark is too far is as follows: the median of the landmark coordinates along Z is calculated, if the landmark z coordinate exceeds the median distance times a scalar parameter, this landmark is considered too far from the camera. These selected 3D location construct our initial set of landmarks.

Note: we have used the median instead of the mean because the presence of possible outliers in the landmarks could influence substantially the mean calculation leading to non optimum results.

In order to extract the candidate keypoints, the Harris corner detector is applied again but now to the second bootstrap frame. After removing the corner points which are too close to the existing keypoints, these new detected points define the set of candidate keypoints. The set of the first observation of candidate keypoints will be exactly the same as the set of candidate keypoints since all the candidate are observed for the first time. The set of the camera pose of candidate keypoints at its first observation can be initialized according to the current camera pose so the one of the second bootstrap frame.

With all these sets constructed, the state of bootstrap frame are initialized.

2.3 Continuous operation

The continuous operation of our VO pipeline can be divided into the following steps.

1. Track keypoints in the current frame and associate to existing landmarks
2. Estimate camera pose using associated keypoints and landmarks
3. Triangulate new landmarks regularly

4. Bundle adjustment

2.3.1 Associate keypoints to existing landmarks

At frame i, all the landmarks from the previous frame are at the same location. Keypoints $\{p_k^{i-1}\}$ in the keypoints set $\mathcal{P}^{i-1}$ at last frame are tracked to the current frame using KLT. The keypoints which are successfully tracked through the current frame constitute $\mathcal{P}^i$ and get associated with the landmarks associated with the tracked keyframes from the previous frame in $\mathcal{X}^i$. The keypoints which are not be able to be tracked using KLT are discarded along with the corresponding landmarks.

2.3.2 Estimate camera pose using associated keypoints

With the tracked keypoints $\mathcal{P}^i$ and associated landmarks $\mathcal{X}^i$ in Section 2.3.1, the camera pose in the world frame can be estimated from the 3D-2D correspondence of the keypoints and landmarks using P3P with RANSAC. At this point in order to refine the camera pose, we include a pose refinement step which, using non-linear optimization, it refines only the just estimated camera pose. This pose refinement step will be further discussed in Section **??**. This step is performed using the inlier set obtained from P3P+RANSAC to select the inlier keypoints and landmarks. Again at this point we applied a selection process to this landmark set. All the landmarks that are behind the camera, or too far from the camera will be eliminated, similarly as described before.

2.3.3 Triangulate new landmarks

First, candidate keypoints $\{c_k^{i-1}\}$ in the keypoints set $\mathcal{C}^{i-1}$ at last frame are tracked to the current frame using KLT. Untracked candidate keypoints and their associated first observations and camera pose in set $\mathcal{F}^{i-1}$, $\mathcal{T}^{i-1}$ are discarded.

Then, we calculate the angle between each bearing vector of the candidate keypoint and its associated first observation. If this angle exceed a threshold, the candidate keypoint will be considered a tracked keypoint. Now we have to triangulate the new landmark associated with this keypoint. After the new landmark is triangulated, a similar selection criteria introduced in Section 2.2 will applied, so the landmark will be first transformed to the current local camera frame using current camera pose. If the triangulated landmarks meet the criteria, the candidate keypoint, as well as this landmark will be added to the keypoints set $\mathcal{P}^i$ and landmarks set $\mathcal{X}^i$. All the data related to candidate keypoint will be delete from the corresponding sets $\mathcal{C}^i$, $\mathcal{F}^i$ and $\mathcal{T}^i$. With this passage we have expanded the set of tracked keypoints and landmarks to contrast the shrinking phenomenon typical of tracking.

In order to contrast also the shrinking of the candidate keypoints we have to add new candidate keypoints. Therefore, we detect Harris corner points at the current frame I^i as potential candidate keypoints. These potential candidate keypoints will be considered a candidate keypoints only they are not close to the current keypoints and candidate keypoints. When one valid candidate keypoint is detected, its data will be added to corresponding sets $\mathcal{C}^i$, $\mathcal{F}^i$ and $\mathcal{T}^i$.

2.4 Bundle Adjustment

Bundle adjustment is applied to refine the estimated poses and landmarks in order to minimizes the reprojection error. In our VO Bundle Adjustment is implemented using keyframes selection. A keyframes is detected when the following inequality is satisfied:

$$\frac{keyframe\ distace}{average\ depth} > 0.2 \tag{2}$$

Every time a keyframe is detected, bundle adjustment will be applied between last two keyframes.

In order to avoid the an entire scene position shift during the optimization, we applied an extra contraints to the optimization problem to set the first state to be fixed to its original position before the bundle adjustment.

BA Implementation First the state has been converted into a $hidden_state$ vector, *observations* vector and *landmarks* vector. The $hidden_state$ vector contains the optimization variables

$$[q_0\ t_0\ ...\ q_n\ t_n\ X_0\ ...\ X_m]$$

where q is a $4x1$ quaternion, t is a $3x1$ translational vector and X is a $3x1$ landmark. The *observations* vector contains the number of landmarks, the keypoints (x, y) and the indeces with the corresponding landmarks for each frame. The *landmarks* vector contains all the landmarks for all the states that we are optimizing over.

After having converted the state to a proper form for Bundle Adjustment we use a non linear optimizer to reduce the error term. The error term is simply obtained by computing the reprojection error between the landmarks, projected in camera frame and the keypoints.

After having applied the non linear optimization step we simply have to convert the result from the solver into the state form used in our VO. In order to speed up the optimization a Jacobian matrix is given to the solver.

2.5 Pose refinement

Pose refinement has been implemented very similarly to Bundle Adjustment but using only one state and not considering the landmarks as optimization variables. First the state has been converted into a $hidden_state$ vector, *observations* vector and *landmarks* vector. The $hidden_state$ vector contains the optimization variables in this case are 7: 4 are from the quaternion and 3 from the translational vector. The *observations* vector contains the number of landmarks, the keypoints (x, y) and the indices with the corresponding landmarks. The *landmarks* vector contains all the landmarks.

After having converted the state to a proper form for pose refinement we use a non linear optimizer to reduce the error term on the pose refinement. The error term is obtained by computing the reprojection error between the landmarks, projected in camera frame using the optimization variables and K, and the keypoints.

After having applied the non linear optimization step we simply have to convert the result from the solver into the state form used in our VO.

3 Problems

3.1 Keypoint Shrinking

When we run our pipeline for the first time, the pipeline is not able to estimate the camera pose after few frames. There are not enough keypoints and landmarks for it to perform the estimation. The number of the keypoints quickly shrinks to almost zero, even if we are adding new landmarks regularly. The decreasing speed of keypoints is faster than the the speed of newly triangulated landmarks.

We solved this problem by adapting Harris corner detector parameters and the threshold angle used to decide if a new landmark need to be triangulated. In this way, the number of newly triangulated landmarks increases, and we can make sure that there are always enough keypoints and landmarks to estimate the camera pose.

3.2 Uneven Keypoints

Many keypoints are detected in each frame. And these keypoints have two tendency that are not beneficial.

Firstly, many keypoints are concentrated in areas with very far landmarks. For example, many detected keypoints in KITTI database are concentrated in the center of image while the camera goes straight on the road. These keypoints are valid Harris corners, and can be used in triangulation, but the results of such landmarks are noisy and not reliable. So we increased the Gaussian filter size of the Harris corner detector. In the image, the object that are very far is treated like a noise and get smoothed. Thus the unreliable landmarks are removed.

Another problem we have with keypoints is present when the image illumination is not uniform. For example, around frame 30 of the KITTI database, half of the image is shadowed. The detected keypoints are not evenly distributed. Most of the keypoints lie in the illuminated half. Sometimes all the keypoints lie in the illuminated half. By lowering the quality of Harris corners detection, we are able to detect several keypoints in the shadowed half. But the distribution of the keypoints is still biased. We solved this issue by used a higher radius threshold to decide if two points are too close to be considered as separate points, this sparsify the keypoints selection.

3.3 Bundle Adjustment instablity

We implemented a windowed bundle adjustment approach but we encountered some instability problems during the optimization. We have noted that that especially when there are thousands of landmark tracked and we have estimated more that a few hundreds frames our pose and landmark estimation becomes highly unreliable.

A possible cause could be the presence of outlier landmarks especially far away that cause massive error terms. Since we are using a 2-norm approach the presence of these points skews the optimization toward a worse estimate. We propose 2 possible solution to this issue (not implemented in our VO):

1. Instead of the 2-norm, it would be possible to use the Huber or Tukey norm to make the optimization process more robust to outliers.

2. Since most of these spurious points have low Harris score, it could be possible to weight the error term with a weighting term given by the cornerness function.

So we can say that our implementation of BA is beneficial especially in the first hundreds of frames but becomes to unstable to be used reliably on the entire dataset. For this reason we recommend to put the BA_{flag} to false if the VO is required to run for a lot of frames.

3.4 Scaling Drift

Our VO pipeline suffers from scaling drift, which is reasonable to expect since it is implemented using a monocular approach. We noted that the scale drift increases noticeably after a series of turns as can be seen in KITTI dataset.

4 Recording Dataset

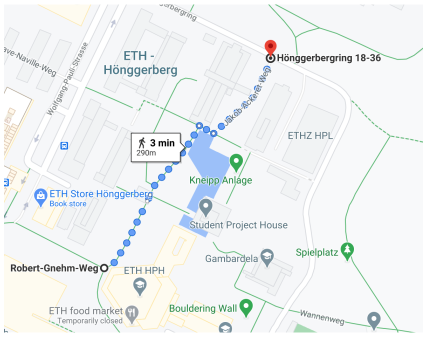

Figure 1: Path for recorded dataset

We record our own dataset called "Recording" and run our VO pipeline with it. The device used to film is Huawei P30pro. The video is filmed at ETH. The path is shown on Figure 1. The Original frame rate for the video is 30fps, in order to run the pipeline, we downsample the video to 5fps.

First we calibrated the camera using the camera calibration toolbox. The distortions are negligible and are not considered in the VO pipeline since the pre-processing of the camera sensor already compensate for this greatly.

The resulting trajectory have a similar path to the true path, yet the trajectory is not very smooth. This could partially due to the fact that the camera is held by hand and recording is filmed on foot. The video itself is not very stable. It has many extra and sudden movements than the provided datasets, make it almost impossible to get a smooth trajectory especially in corners where a car turns smoothly while a person much more abruptly.

5 Results

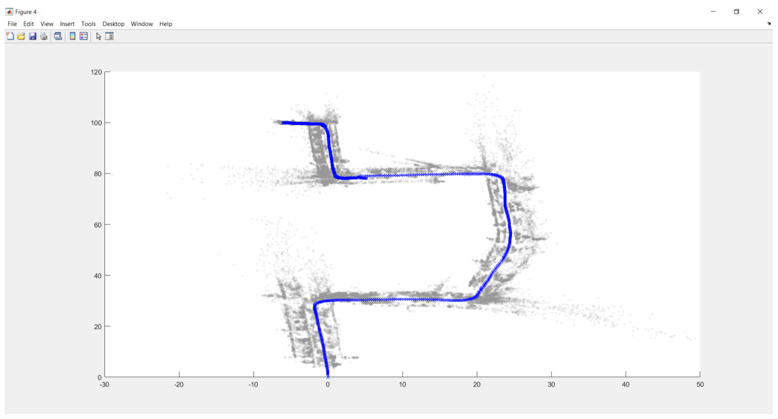

Figure 2: Trajectory and Landmarks, Dataset KITTI without BA, first 1000 frames

The results of our VO pipeline are shown in this section. Figure 2 shows the estimated trajectory for KITTI for first 1000 frames. Figure 3 shows the estimated trajectory for Malaga for first 1000 frames. Figure 4 shows the estimated trajectory for parking dataset. Figure 5 shows the estimated trajectory in our own recorded dataset for 1000 frames. Overall, our VO pipeline works reasonably. However, it does not produces perfect results, there are some jitters in the trajectory, or some position shift during the estimation.

Figure 6 shows the estimated trajectory results for parking dataset with bundle adjustment. After introducing bundle adjustment, the jitters on the trajectory can be optimized almost entirely, as shown in Figure 8 and Figure 7. Here, we use the first 50 frames to show the results of bundle adjustment. But bundle adjustment reduces the stability of pipeline, the pipeline might not

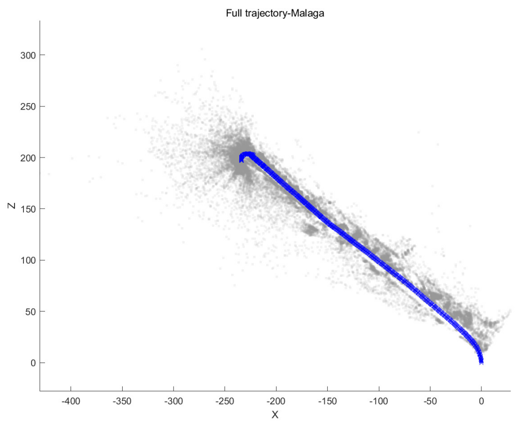

Figure 3: Trajectory and Landmarks, Dataset Malaga without BA, first 1000 frames

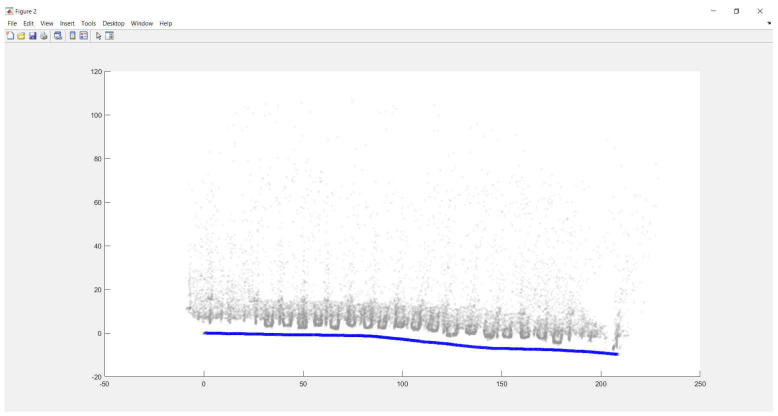

Figure 4: Trajectory and Landmarks, Dataset parking without BA

work after a while if bundle adjustment is enabled. This problem is discussed in Section 3.3.

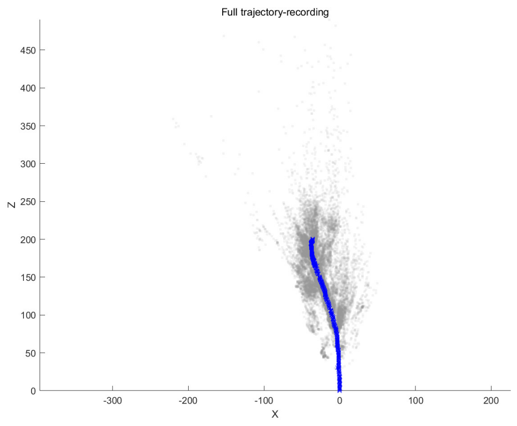

Figure 5: Trajectory and Landmarks, own recorded Dataset without BA, 1000 frames

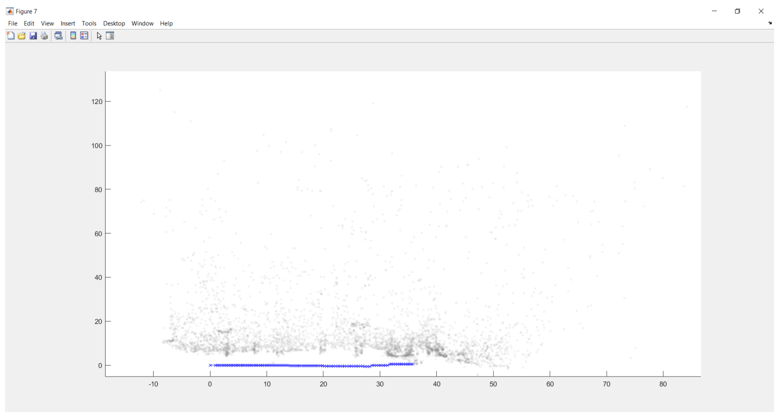

Figure 6: Trajectory and Landmarks, Dataset parking with BA

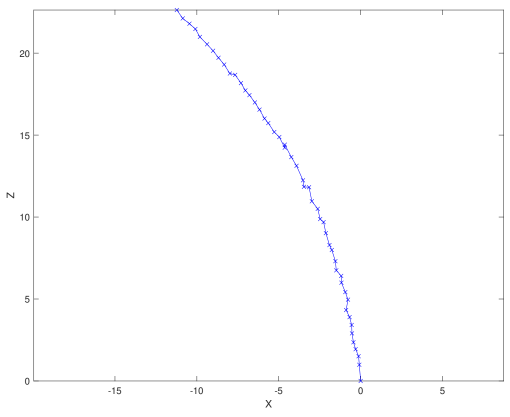

Figure 7: Trajectory and Landmarks, Dataset malaga without BA in first 50 frames

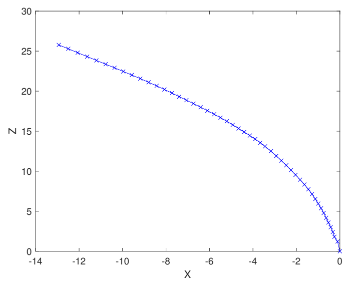

Figure 8: Trajectory and Landmarks, Dataset malaga with BA in first 50 frames