Visual Odometry Pipeline

Vision Algorithms for Mobile Robotics

January 6, 2019

Authors	Christian Sprecher	mailto:cspreche@student.ethz.ch
	Kamil Ritz	mailto:kritz@student.ethz.ch
	Luca Somm	mailto:somml@student.ethz.ch
	GianAndrea Müller	mailto:muellegi@student.ethz.ch

1 Deviations from proposed Pipeline

In general the implementation follows the proposed pipeline quite closely. The data was structured accordingly and the functionality is equivalent. Still there are a few notable differences which are listed below. In addition the tuning factors for our implementation are listed and explained shortly in table 1.

1.1 Automatic Keyframe Selection

Instead of manually defining a suitable pair of frames for bootstrapping the selection of a second keyframe is done automatically, as described in algorithm 1. The motivation was to make the bootstrapping robust for different types of datasets. It now guarantees a sufficiently long initial baseline without the need to tune it by hand.

Note that keypoints are not tracked from first frame to a potential second bootstrap frame, but rather they are tracked through all intermediate frames up to the actual second bootstrap frame. Thus we end up with more and better feature matched between the two bootstrap frames.

Since Matlab's `estimateFundamentalMatrix` is non-deterministic, the bootstrapping was not always successful. Increasing the number of iterations to find an inlier set helped to make the bootstrapping more reliable.

Algorithm 1 Automatic Keyframe Selection

```
Initialize reference frame with Harris Features
while found_keyframe == 0 do
    Track Keyoints to next frame with KLT
    8-Point with Matlab's estimateFundamentalMatrix
    Triangulate Landmarks
    Landmark Sanity Check
    if keframe_distance/average_depth > 0.05 then
        found_keyframe ← 1
    end if
end while
```

1.2 Landmark Sanity Check

In order to improve the minimum baseline on landmark quality a short sanity check has been written. First only points with a positive z coordinate in the actual camera frame, thus points actually in front and visible by the camera are passed as valid. Further we impose an upper limit on the distance from the camera, since far away points tend to have a higher uncertainty. This limit is implement as a multiple of the median of the depth of the landmarks in the camera frame.

1.3 Localized Harris Keypoints with Non-maxima Supression

The feature detection is based on the Harris detection as implemented within Matlab. With the standard implementation however, we faced the issue that keypoints tend to be concentrated in in bright, high-contrast areas, neglecting part of the image completely. This problem often caused our pipeline to break down. The first step towards a solution was to use `histeq` to mitigate the overall brightness changes of the image.

Further we introduce localized harris keypoints. The idea is to search for keypoints locally. For that reason the frame is split into smaller regions with a rectangular grid and each cell is searched for the best Harris corners. Afterwards we perform a non-maxima suppression. These possible new

Figure 1: Comparison of standard Harris (above) with the improved approach (below).

point are then included as candidates if they are not too close to existing keypoints or candidates. This makes for a more even distribution of keypoints, as seen in figure 1. Also read algorithm 2.

Algorithm 2 Localized Harris Keypoints Non-Maxima Suppression

```
function LOC_HARRIS_NONMAXIMA_SUP(grid,radius,npoints,minQuality,FilterSize)
    all_points ← {}
    for all cells in grid do
        cell_points ← detectHarrisFeatures(minQuality,FilterSize)
        while number(filtered_points) < npoints do          ▷ Point by point extraction
            filtered_points ← NonMaximaSuppression(cell_points, radius)
        end while
        all_points ← filtered_points
    end for
end function
```

1.4 Pure Yaw Motion Recovery & Feature Removal

We encountered the problem in our own datasets that on a pure yaw motion of the camera many landmarks were lost, but no new features were added based on the angle threshold imposed on the promotion of candidates.This happens since the bearing vectors are only changing in inertial frame when the camera undergoes a translational motion. Without any translational motion no new landmarks are added based on the proposed bearing vector angle. To solve the problem we introduced a second possibility for candidates to be promoted, which is based on the idea that candidates that have been tracked for a sufficient amount of frames must be somehow reliable. Then among those a pixel threshold is imposed, which promotes candidates that have traveled at least a certain distance on their track. The motivation behind this is that pixel who have moved enough through the frame without being promoted by the bearing vector angle, are likely points that are far away in relation to the movement of the camera. Even if these points can not be triangulated so reliably they still can be useful to estimate the rotation of the camera.

The problem introduced with this practice was a vast amount of new landmarks when performing a prolonged rotation. We recorded peaks of up to 6000 features. The solution was to set a limit of 500-1000 features. After that threshold is passed keypoints are removed by choosing one after another and always eliminating all keypoints within a certain radius.

1.5 Pose Refinement

When localizing an new frame based on the tracked keypoints and on the current landmarks using P3P and subsequently DLT on the best inlier set, a small pose refinement is conducted. We minimize the reprojection error of all found inliers by optimizing the pose. Only very few iterations are necessary to reduce the reprojection error quiet a bit.

1.6 Checking Reprojection Error of new Landmarks

When assessing the validity of new landmarks, in addition to enforcing an angle threshold, the reprojection error of the candidate landmarks is limited as well. This is important since wrong feature tracks or moving scene objects can result in bearing vectors that are skew, and therefore do not intersect at all. In this case the triangulated landmark are not very helpful and should be omitted. Fortunately, this case can be captured very effectively by looking at the reprojection error of the landmark. Candidates with a large reprojection error are consequently removed from the state.

1.7 Tunable Parameters

Functionality	Parameter Name	Description	Range
Harris	`rows`	Number of rows of the grid.	3 - 5
	`cols`	Number of columns of the grid.	3 - 5
	`radius`	Radius used for the non maxima suppression.	4 - 10
	`npoints`	Desired number of features per grid cell.	30 - 45
	`minQuality`	Minimum allowed quality for harris features.	0.01
	`FilterSize`	Gaussian filter dimension for detection of harris features.	3 - 25
Matching	`maxBidirectionalError`	Forward-backward error threshold for KLT tracking.	2.7 - 3
	`radius_to_new_feature`	Minimum distance between existing features (keypoints and candidates) and new candidates.	2 - 5
	`KLT_score_limit`	Minimum quality of KLT tracked features.	0.997
	`block_size`	Size of neighborhood around each point being tracked for KLT tracking.	25 - 35
Localization	`RANSAC_iteration`	Number of performed RANSAC iterations.	1k - 3k
	`RANSAC_error`	Maximum reprojection error for inliers.	3.5 - 10
Triangulation	`sanity_check_factor`	Determines depth after which landmarks are removed.	7
	`angle_threshold`	Angle threshold for new landmarks.	1.5 - 5
	`age_threshold`	Minimum track duration for a new candidate to be promoted to landmark.	8 - 10
	`pixel_threshold`	Minimum pixel distance for a new candidate to be promoted to landmark.	35
BA	`periode`	Number of frames until next optimization.	5 - 10
	`window_size`	Window size for windowed BA.	12 - 20
	`max_iteration`	Maximum number of iterations for `lsqnonlin`.	15 - 20

Table 1: Tunable Parameters

The feature detector's and tracker's parameter had to be well tuned for each single dataset. The tuning depends mostly on the image resolution and the image content and structure. Depending on how good the tracking works over the whole sequence one has to adapt the number of RANSAC iterations for the localization. Since it is not required to boost the efficiency of the code, the

RANSAC iterations are chosen rather conservatively. The angle threshold for new landmarks had to be chosen rather small, such that we add new landmarks fast enough during corner turns.

2 Milestones

	Name	Task	Problems	Solutions
1	Setup	Problem identification	Complexity	Divide and conquer, functional programming, paper prototyping, skeleton main file with function prototypes
		Team organisation	Communication	Regular meetings, teamchat
		Code organisation	Collaboration	Git for code, LaTeX for documentation
		Data handling	Diversity	Establish dataset class equipped with methods for loading data, getting frames, loading tuning parameter.
2	Basics	Implement basic features	Conventions	Adhere to established conventions to improve readability and consistency.
		Bootstrapping	Feature Integration	Clarify misconceptions
3	Minimum Viable Product	Continuous Operation	Overview	Code refactoring and reorganisation of the folder structure
			Small bugs	Peer review and debugging plots
		Tuning	Poor Performance	Realizing that tuning the feature detector and the feature tracker is critical for the performance of the pipeline.
4	Extension	Produce Datasets	Calibration	Realize that calibration is specific to video format.
			Distortion	Only allow detection of features within the valid image region of the undistorted frame.
		BA	Parse state history	Carefully associate keypoints with the right landmark.
			Poor performance	Fix first two poses.

Table 2: Overview of Project Milestones

2.1 Most Interesting Problems

The most challenging problem was actually getting the pipeline to work after having established its basic functionality. We realized that debugging it as a whole is basically impossible and thus decided to split the work and reiterate on the building blocks. Mostly by improving the debugging plots we realized that we were only generating new landmarks that were extremely close the the position of the camera. Without useful new landmarks the localisation breaks down very fast even if the feature tracking works well. The problem was that checking whether the candidate keypoints have travelled a large enough angle and performing the sanity check was not a sufficient condition for valid new landmarks. Instead we decided to limit the reprojection error which was key for getting the VO pipeline to run reasonably.

3 Results

The following section will give critical information about the uploaded screencasts. The different plots included in the screencast are explained in figure 2. The relevant screencasts were run on a machine with the following specs: $8 \cdot 2.5\,\text{GHz}$ CPU, $8\,\text{G}$ Byte RAM and using a maximum of 121 threads. The videos are provided here.

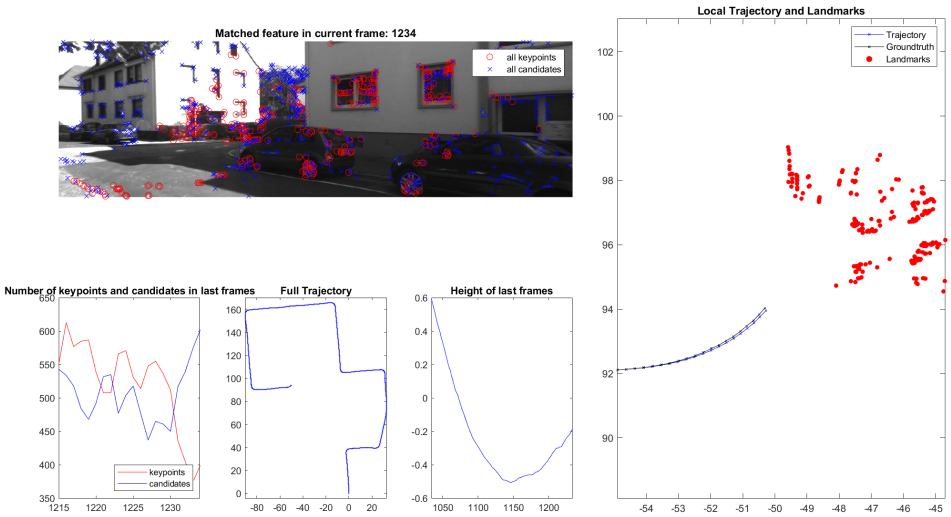

Figure 2: The elements included in the screencast are, from top left to bottom right: The current frame with all current keypoints candidates, the number of keypoints vs. the number of candidates over time, the full trajectory either in 2D or in 3D, the height of the camera pose over time and the trajectory over the last few frames compared to ground truth and showing all current landmarks.

To evaluate the pipeline, a qualitative comparison was made by adding the ground truth to the local plot. The ground truth was transformed, such that the first pose of the last twenty frames align with the one of the generated trajectory. Furthermore, the scale of the ground truth has been adapted to fit. Hence, scale drifts are not that visible in this comparison. However, the result of this comparison looks very promising. The difference is only marginal. Even when looking at the full trajectory of the kitti datset, the result seems to be plausible.

There were multiple things leading to an accurate trajectory. The biggest impact came from the pose refinement and the bundle adjustment, which both improved the localization and triangulation significantly. Next to that, the decision to split the image in multiple regions and search for key points in all of them individually helped to make the localization more robust. Since we added multiple dataset with cameras on drones, we adapted are pipeline to be able to deal with fast yaw rotations without losing landmarks.

One of the actions of the pipeline is to remove all landmarks which are behind the camera. This leads to a better localization, but can lead to a failing behavior. If the localization determines (falsely) a large rotation, many landmarks will be removed, such that the pipeline could break. To lower the risk of poor pose estimation and therby such actions, a high number of ransac iterations has been chosen.

3.1 Parking

The VO pipeline runs on the parking dataset very nicely. The feature tracking and the localization work very reliably. The performance increased quiet a lot by implementing a sliding window bundle adjustment as described in section 4.2.

3.2 Kitti

The results of the VO pipeline on the Kitti dataset are looking very promising. The overall trajectory looks quiet reasonable. A bit of slow scale drift is noticeable. The dataset contains a few video sequences with high contrast scenes. In these sequences we had sometimes problems with the feature tracking. In these situation the localization lost its robustness. As already mentioned above a poor pose estimation can lead to the removal of almost all landmarks during the sanity check of the landmarks, which can break the VO pipeline. Due to this it was not possible to run the whole dataset at once without having to restart it a few frames before the break down.

3.3 Malaga

In our opinion this is the most difficult dataset to handle. It is very hard to tune the addition of new landmarks. On one side new landmarks should be added very quickly to deal with the sharp turn, on the other side landmarks should be added very conservatively to not get landmarks of detected features in the cloud and at borders of overexposed image regions. We think that this could have caused the rather large scale drift that we have experienced on this dataset.

3.4 Bike Zurich Straight

This dataset shows a short bike ride along a straight street. It was recorded at high framerate and has a higher resolution than the provided datasets. Therefore a lot of harris corners can be detected and tracked reliably. Since we have chosen radius_to_new_feature rather small, the number of tracked keypoints increases very fast. As soon as the number of keypoints are bigger than a certain threshold we remove to close keypoints from the state. This explains why the number of landmarks is changing a lot. This behavior can also be observed in the fpv dataset, which are described in the next section. A simple solution would be to increase the radius_to_new_feature, which would help to run the VO a bit faster.

Location	Zurich
Motion	Handheld on a bike
Device	Gopro Hero 6 (no stabilization)
Resolution	1920x1080
Number of Frames	1653
Framerate	120 fps (downsampled to 60 fps)

3.5 fpv 1,2,3

Three datasets - fpv1, fpv2 and fpv3 - were recorded with a racing drone. Therefore the camera is moving in all 6 DOF. One challenge is the fast yaw motion that can occur even when the camera is staying at the same position. As explained in section 1.4, our VO pipeline also adds new landmarks in these situation. This can be observed for example in the middle of the fpv1 dataset. After a short rise, the drone performs a 180 degree yaw motion before it then dives down again. Turning this yaw motion it only can track features that are relatively far away. Even tough these potential landmark were always observed from the same point, we add them if they are tracked for a long enough duration and if they moved enough trough the image frame. Since the triangulation of these landmarks includes a rather high uncertainty, small jumps in the camera poses can be observed during this turn. Nevertheless the proposed solution manages to deal with these situation quiet well.

Location	Schüpfheim, LU
Motion	Drone flight
Device	Gopro Hero 6 (no stabilization)
Resolution	1920x1080
Number of Frames	[706,1660,1412]
Framerate	60 fps (downsampled to 30 fps)

4 Additional Features

4.1 Datasets

As an additional feature a series of datasets was recorded.

1. **Camera calibration**

   The Gopro 6 was calibrated based on a short clip of a checkerboard, using the calibration toolbox in Matlab. The size of the used checkerboard, on A4 paper, is rather small. Therefore the checkerboard has to be held rather close in front of the camera. This could decrease the quality of the calibration. Nevertheless, a very nice performance on our custom datasets could be observed. Another issue arose after undistorting the images which led to black padding around the image. In order to avoid to crop the image, harris corners at the border of the frame are discarded.

2. **Dataset** A first test dataset was recorded handheld on a bike. To create a more interesting dataset, the camera was mounted on a racing drone. This makes for various camera motions like rolling and pitching. Even though the motion was rather fast the high frame rate of 60 fps and thus short exposure made for sharp images. For reducing computational effort every second image was used, reducing the effective frame rate by half.

4.2 Sliding Window Bundle Adjustment

For refinement of the trajectory bundle adjustment was implemented. To reduce the cycle time the implementation proposed in exercise 9 was extended with a sliding window, therefore only optimising a certain number of past frames. The implementation includes the precalculation of the jacobian's non-zero elements to maximize speed.

The extraction of landmarks from the state history, the mapping to the corresponding landmarks and the restructuring for the optimization framework was challenging. First we assume that landmarks have the same coordinates throughout the considered window. By this we find correspondences between the different frames. Then we use Matlab's `unique` to find the unique set of landmarks viewed during the chosen track. Finally this set is restructured to fit the data structure given in exercise 9.

To make sliding window bundle adjustment effective, a measure for connecting the independently optimized piece of the trajectory to its past has to be found. The approach proposed is to introduce continuity constraints by explicitly excluding the first two poses of the current segment from the optimization. Continuity is enforced by still optimizing the reprojection error of the landmarks in the two poses. This solution effectively keeps the trajectory smooth. In order to not change the scale during the bundle adjustment also the second pose of the window is kept constant. The window size and periode are chosen such that the first two frames of the window are already optimized in the previous iteration.

To see how much the bundle adjustment improves the performance of the VO pipeline, a test case was set up on the parking dataset. In this test we compare the trajectory with and without bundles adjustment to the ground truth. To get the scale of the monocular VO right, the pose of the ground truth was used for the bootstrapping. Landmarks with a big reprojection error are removed to get a good bootstrapping. Five runs of the VO pipeline were performed with and without bundle adjustment with the same bootstrapping and then compared to the ground truth trajectory. The following error measurement was used to determine the performance of the trajectory

$$\sum_i t^{vo}_{i,i+1} \cdot \frac{t^{gt}_{i-1,i}}{t^{vo}_{i-1,i}}$$

, where $t^{vo}_{i,i+1}$ is the translation vector from camera pose i to camera pose $i+1$ of the VO trajectory. $t^{gt}_{i,i+1}$ is the translation vector between camera poses of the ground truth trajectory. The fraction used in the error equation helps to compensate for the scale drift that was occurred in previous frames. Since this error measurement is looking at the error from frame to frame, and not only comparing the last poses of both trajectories, it is not so sensitive to rotation errors and scale drifts in the beginning of the trajectory.

The VO pipeline without bundle adjustment showed a mean error of 5.548 and a standard deviation of 0.505, while the VO pipeline with bundle adjustment showed a mean error of 1.597 and a standard deviation of 0.008. The bundle adjustment seems not only improve the performance of the VO pipeline, but also to make it performance more consistent.