Indirect Visual Inertial Odometry with Optical Flow

Kerem Yildirir
Technical University Munich
kerem.yildirir@tum.de

Poyraz Kivanc Karacam
Technical University Munich
poyraz.karacam@tum.de

Yigit Aras Tunali
Technical University Munich
yigitaras.tunali@tum.de

August 9, 2021

Contents

1 Introduction & Project Goals

As the processors shrink in size and grow in compute power, many robots are becoming more and more integrated with businesses. Most of the tasks in these cases, require the robot to be autonomous and unmanned. This requirement brings out many additional challenges. One of which is the need for localization and mapping, for the robot to interact with its environment. To perform their tasks well and navigate their environment without any problems, the algorithms in use have to be quick, accurate, and robust. Camera-based motion estimation is one such technique. In this project, we extend a stereo visual odometry implementation with a sparse optical flow tracking system into the visual odometry pipeline as an alternative to feature descriptor matching, aiming to improve the quality of the generated trajectory. To achieve this goal, we adopted the optical flow approach described in [8], which we henceforth refer to as the BASALT approach. Additionally, we also throw out the stereo descriptor matching in the original baseline and replace it with stereo matching using optical flow from the left camera to the right camera. We then evaluate the resulting maps and predicted trajectories of the base and the extended visual odometry implementations both visually and quantitatively by plotting the trajectories and also computing relative and absolute pose errors for both versions. We also compare the impact of BASALT optical flow with OpenCV's Lukas-Kanade approach [4] and include these results in our evaluation. Finally, we identify the shortcomings of the method and discuss further improvements.

2 Visual Odometry

With the rise of automation and the popularity of autonomous robots, visual navigation is becoming more and more sought after. Visual odometry is one of the approaches used widely in Robotics to determine the position and orientation of a robot. A visual odometry system can be implemented using a mono, stereo, or RGB-D camera, depending on the needs of the project. While the monocular camera approaches are inexpensive and can have a small form, they suffer from scale ambiguity. On the other hand, stereo approaches mimic the human visual system and can easily estimate the scale [5, 2]. The stereo approach has its drawbacks such as a bulkier set-up, requirement of calibration, and a good synchronization between cameras.

In this project, we have an initial visual odometry system that uses a stereo camera set-up. The whole pipeline and the operations done in each step can be seen in the figure 1. To achieve real-time performance, only a subset of the frames

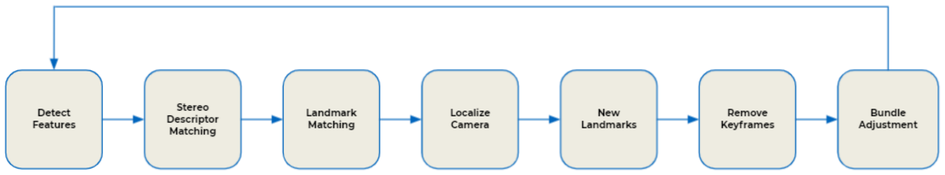

Figure 1: Complete stereo Visual Odometry Pipeline.

is used in the optimization step. The first step of the pipeline consists of keypoint detection for images of both cameras for that frame. Here, we detect the features using Shi-Tomasi corner detector [7] and compute ORB descriptors [6] for each point. Afterward, these keypoints are matched using their descriptors over the stereo set-up and inliers are picked via the epipolar constraint. Then the descriptors are matched between the map and the current frame by projecting the landmarks onto the image plane. Using these landmark-to-keypoint matches and PnP (Perspective-n-Point) the camera is localized and the pose is found. This is followed by the addition of new landmarks and removal of old keyframes according to some thresholds and finally, a bundle adjustment optimization is done over the keyframes that are still not removed.

3 Indirect Visual Odometry with Optical Flow

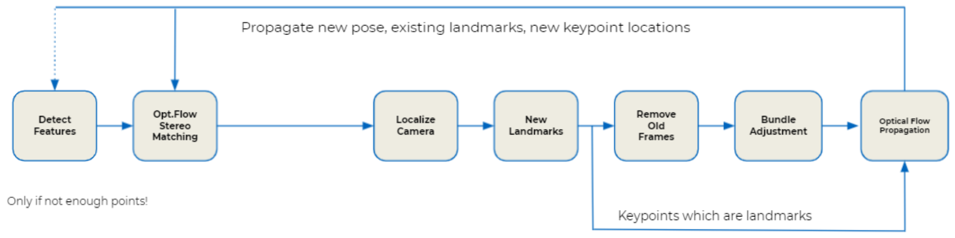

Figure 2: Modified stereo VO Pipeline with optical flow.

In figure 2, the extended pipeline can be observed. The most substantial change is the removal of descriptor matching and the approach of optimization over keyframes. For the optical flow to work, a frame-to-frame approach was necessary where the optimization was over a window of a certain number of frames in each step. The keypoints detected in a frame are propagated to the next ones. As long as the tracking of keypoints is not lost we also know to which landmark they correspond. This approach also allows us to circumvent the landmark matching step used in the baseline visual odometry implementation. If the keypoint is no longer tracked, the landmark is marked as ”inactive”. If the same point is found again during the algorithm it will be registered as a new landmark and the effects of drift will be clearly visible in the constructed map, as it can be observed in 3. This snapshot has been taken after visiting the same spot in the world for the third time, the gray points are ”inactive” landmarks while light blue ones are the currently active ones. It is clear that points that were supposed to correspond to the same landmark are registered as different ones and this is further enhanced by the effect of drift.

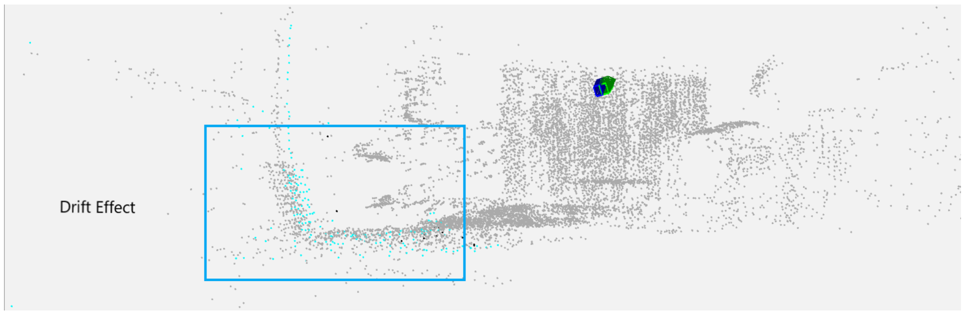

Figure 3: Effect of drift on the constructed map.

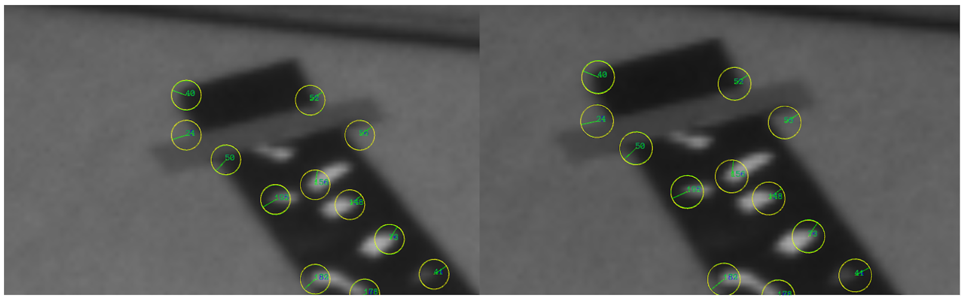

Figure 4: Stereo matching with BASALT.

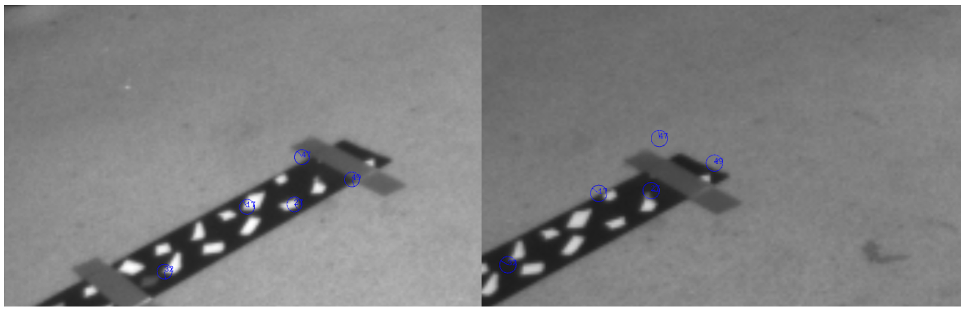

Figure 5: Stereo matching with Lukas-Kanade.

The propagated points are matched between cameras again using optical flow. The remaining parts of the pipeline stay the same as the original implementation. As for the optical flow algorithm, two approaches were taken into consideration with one being the basic Lukas-Kanade algorithm and the other an advanced version, namely the BASALT approach [8]. While both optical flow algorithms perform well for the overall tracking of keypoints through frames, we have observed that the OpenCV's optical flow approach is less robust to situations where the assumptions of the optical flow are slightly violated. The more problematic part is the usage of optical flow for stereo matching. In our experiments, we have observed that the OpenCV's [4] Lukas-Kanade algorithm is not able to find proper stereo matches for the keypoints and fails almost all the time, which can be seen in 5. This is most likely because the Lukas-Kanade approach can't handle the baseline between the cameras. An example for this case can be seen for the BASALT case in 4 where the stereo matched points are robust and correspond to each other properly.

4 Optical Flow

Our main contribution in this project is the integration of a KLT-based optical flow tracking, which we adopted from [8], as a point tracking system. We use optical flow in both frame to frame and left to right stereo frame matching.

4.1 Level Pyramids

Given a frame, we construct a level pyramid. The pyramid consists of 3 levels, where the i^{th} level is the scaled-down frame with a scaling factor of 2^{i-1}. Note that level 1 corresponds to the given frame.

When tracking points across two frames, we track them on each level of their respective pyramids. If a point is considered valid on each level, then it is accepted as a potential inlier. Using such a scheme enables us to track points robustly against large displacements.
To find valid points between i^{th} levels P_i and P_i' of two pyramids, we first track the keypoints from P_i to find their correspondences on P_i'. Then, we track the found correspondences on P_i' back to P_i. if a correspondence from P_i' does not lead back to its origin point on P_i, it is considered an outlier and discarded.

4.2 Tracking a point

We use a patch-based system for tracking. Given a point, we use pre-defined offsets to construct an octagonal patch centered around it. The patches help us incorporate the local information around a point. This local information is used to formulate a dissimilarity term that is invariant to intensity scaling.
To find a warp $T \in SE(2)$ that characterizes the motion of between two patches across two images t and t', we use the same residual term defined in [8] as

$$r_i(\xi) = \frac{I_{t'}(Tx_i)}{\overline{I_{t'}}} - \frac{I_t(x_i)}{\overline{I_t}}, \quad \forall x_i \in \Omega \tag{1}$$

where $I_t(x_i)$ is the intensity value of the point x_i in image t, Ω is the set of points in a patch, and $\overline{I_t}$ is the mean intensity of the patch in image t.

4.3 Warp Optimization

Optimizing the warp comprises two main parts:

- Computation of the point gradients
- Updating the warp until convergence

When computing the point gradients, we utilize floating point representations. Given a floating point coordinate (x, y), we bilinearly interpolate its value using the four surrounding pixel values (u, v), $(u + 1, v)$, $(u, v + 1)$, and $(u + 1, v + 1)$, where $u = \lfloor x \rfloor$ and $v = \lfloor y \rfloor$. Then, we also interpolate the points in one unit distance to the point (x, y) along the vertical and horizontal axes. Finally, using the additionally interpolated points, we apply the central difference method to estimate the gradients of the point (x,y). Figure 6 demonstrates the described method.

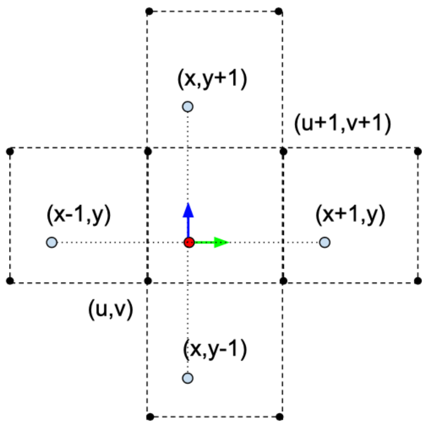

Figure 6: Interpolation scheme for the point (x,y).

Note that this method would not work without utilizing the floating-point representation, since for a discrete point the gradient would be 0.
Once the point gradients are calculated, the Jacobian is computed using these values. Then, we estimate the inverse Hessian using Cholesky Decomposition and apply the Gauss-Newton method to the residual term from (1) to find the increment $\xi \in R^3$. Finally, updates on the warp are calculated as:

$$R^{j+1} = R \cdot exp(\xi), \quad t^{j+1} = t^j + \xi \tag{2}$$

where R^j and t^j are the rotational and translational parts of the warp T^j during the j^{th} iteration, respectively.

5 Experiments and Results

We compare the performance of our implementation with the baseline visual odometry. Furthermore, we also add the results of our method by using first the Lukas-Kanade optical flow from OpenCV and the BASALT approach. For quantitative analysis of our results, we use Relative Pose Error (RPE) and Absolute Pose Error (APE) metrics [3].

Method	**rmse**	**mean**	**median**	**std**	**min**	**max**
OpenCV	0.09	0.03	0.02	0.09	0.00	4.29
Ours	**0.03**	**0.03**	**0.02**	**0.02**	**0.00**	**0.07**
Original Odometry	0.04	0.03	0.03	0.02	0.00	0.17

Table 1: Relative Pose Error

Method	**rmse**	**mean**	**median**	**std**	**min**	**max**
OpenCV	4.24	3.57	2.64	2.28	1.28	10.55
Ours	**0.12**	**0.11**	**0.11**	**0.04**	**0.01**	**0.21**
Original Odometry	0.16	0.14	0.14	0.07	0.01	0.37

Table 2: Absolute Pose Error

In tables 1 and 2, the RPE and APE results can be seen. For all the results Vicon Room 1 01 from EuRoC MAV [1] dataset have been used. While testing the performance of the OpenCV method, Lukas-Kanade optical flow was used only to track points in consecutive frames and not for stereo matching. For stereo matching again the BASALT approach is used since Lukas-Kanade failes to find a sufficient amount of matches. The results of the BASALT approach and the original odometry are close, but the BASALT approach performs slightly better with having a lower max error. As for the OpenCV version, it fails more often while tracking keypoints, thus the trajectory estimation error grows very large at the failure sequences. The figures 7 and 8 are also helpful to visually analyze how well the tracking ensues. As expected, the BASALT approach outperforms other versions in all metrics, however, it still suffers from the accumulated drift.

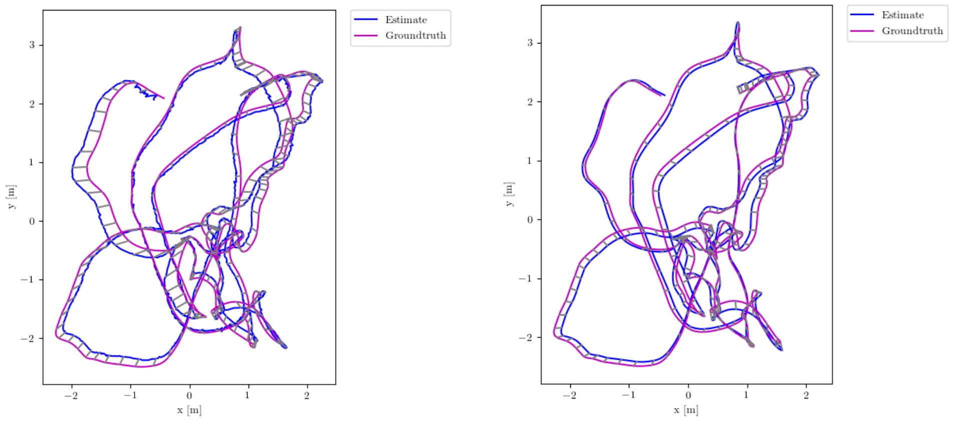

Figure 7: Original vs our trajectory comparison with ground truth.

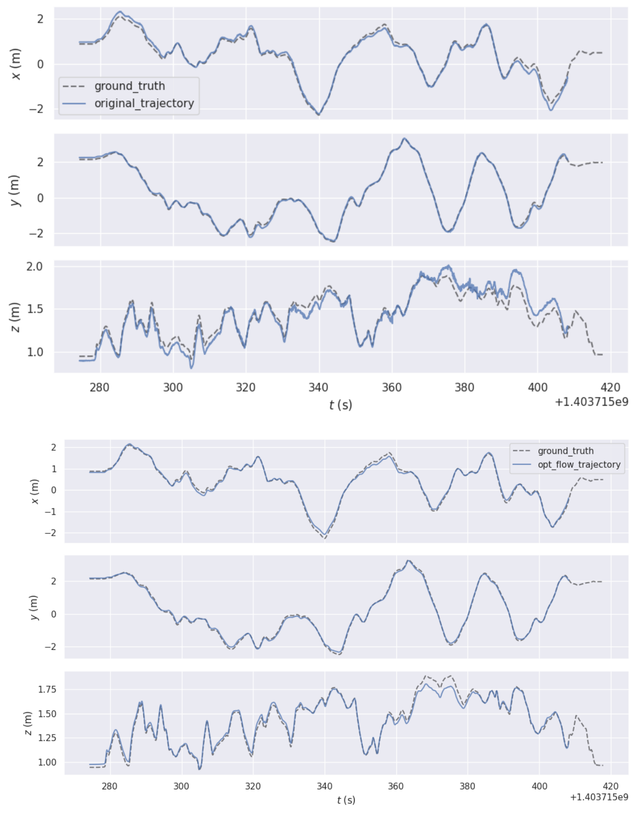

Figure 8: Translation errors, original vs ours

6 Conclusion & Possible Extensions

In conclusion, we have observed that using optical flow as a feature tracker allows us to track the keypoints more robustly than descriptor matching. However, it also introduces new complexities, such as having an extra parameter as a threshold for the number of currently tracked points, which will be used to decide if new keypoints should be detected.

The sudden increase in number of keypoints at each detection step also lowers the performance of the system for a couple of frames. A possible workaround for this is dividing the image into a grid and detecting points only in grid cells where the number is below a threshold as illustrated in 9. Unfortunately, this approach doesn't solve every problem. There might be times when a grid cell will only have a white wall or a textureless object, which will violate the brightness consistency assumption of optical flow algorithms. Movements with relatively large baselines also cause the tracking to fail, due to the violation of assumptions made by the optical flow. While testing the performance of the OpenCV method, Lukas-Kanade optical flow was used only to track points in consecutive frames and not for stereo matching. For stereo matching again the BASALT approach is used since Lukas-Kanade failes to find a sufficient amount of matches. Since we are following a frame-to-frame approach, the failure of tracking for a few sequences affects the overall results.

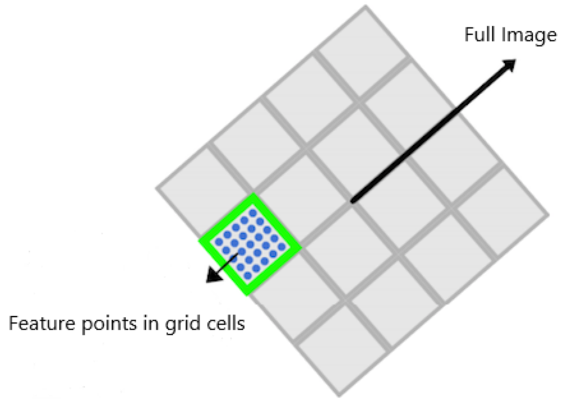

Figure 9: Grid-wise feature detecting approach.

Another shortcoming of the system is that as tracking progresses the error accumulates and the results start to drift. Since we only use a set number of previous frames, the same landmarks that were discarded before will be re-observed again and will be registered in the map again with a slight drift. This is a well-known problem in systems such as ours and can be solved with a loop closure algorithm, which detects the situations when a landmark is re-observed and corrects the camera pose accordingly. Overall, loop closure will increase the robustness of the algorithm against drift and make the reconstructed map more consistent.

Finally, our current frame-to-frame approach achieves a relatively satisfying run-time speed even without multi-threading. We haven't had the chance to work on parallelizing it due to time constraints. Admittedly, the run-time performance can be considerably increased by some structural changes and enabling multi-threading. If the grid-wise feature points approach is implemented, it can also be easily parallelized since the grids will be independent of each other.

References

[1] Michael Burri, Janosch Nikolic, Pascal Gohl, Thomas Schneider, Joern Rehder, Sammy Omari, Markus W Achtelik, and Roland Siegwart. The euroc micro aerial vehicle datasets. *The International Journal of Robotics Research*, 2016. 9

[2] Cesar Cadena, Luca Carlone, Henry Carrillo, Yasir Latif, Davide Scaramuzza, Jose Neira, Ian Reid, and John J. Leonard. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on Robotics*, 32(6):13091332, Dec 2016. 3

[3] Michael Grupp. evo: Python package for the evaluation of odometry and slam. https://github.com/MichaelGrupp/evo, 2017. 8

[4] Itseez. Open source computer vision library. https://github.com/itseez/opencv, 2015. 3, 6

[5] Shashi Poddar, Rahul Kottath, and Vinod Karar. Evolution of visual odometry techniques, 2018. 3

[6] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In *2011 International Conference on Computer Vision*, pages 2564–2571, 2011. 4

[7] Jianbo Shi and Tomasi. Good features to track. In *1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 593–600, 1994. 4

[8] V. Usenko, N. Demmel, D. Schubert, J. Stueckler, and D. Cremers. Visual-inertial mapping with non-linear factor recovery. *IEEE Robotics and Automation Letters (RA-L) & Int. Conference on Intelligent Robotics and Automation (ICRA)*, 5(2):422–429, 2020. 3, 6, 7