Systematic Analysis of Direct Sparse Odometry

Florian Particke*, Adam Kalisz*, Christian Hofmann*, Markus Hiller*, Henrik Bey* and Jörn Thielecke*
*Am Wolfsmantel 33, 91058 Erlangen

***Abstract*—In the field of robotics and autonomous driving, the camera as a sensor gets more and more important, as the camera is cheap and robust against environmental influences. One challenging task is the localization of the robot on an unknown map. This leads to the so-called Simultaneous Localization and Mapping (SLAM) problem. For the Visual SLAM problem, a plethora of algorithms was proposed in the last years, but the algorithms were rarely evaluated regarding the robustness of the approaches. This contribution motivates the systematic analysis of Visual SLAMs in simulation by using heterogeneous environments in Blender. For this purpose, three different environments are used for evaluation ranging from very low detailed to high detailed worlds. In this contribution, the Direct Sparse Odometry (DSO) is evaluated as an exemplary Visual SLAM. It is shown that the DSO is very sensitive to rotations of the camera. In addition, it is presented that if the scene does not provide sufficient clues about the depth, an estimation of the trajectory is not possible. The results are complemented by real-world experiments.**

I. Introduction

Artificial intelligence and machine learning are the main building blocks for the recent advances in robotics. The famous DARPA Grand Challenge was a competition starting in 2004 and already featured a self driving car which is now developed by Google [1]. The participating cars were equipped with a rich set of sensors. Examples are 3D laser scanners, which include a single laser beam that can rotate horizontally and vertically at a varying resolution quality. However, as more autonomous vehicles are introduced, other types of sensors has to be found which are cheaper, more robust to crashes and have less mechanical moving parts. A camera sensor is a cheaper solution to perform localization and map the environment, where the so-called Simultaneous Localization and Mapping (SLAM) problem has to be solved. The goal is to localize the robot and to create a map of the environment. It is not an easy task, as in order to localize a robot on the map, we first need the map and in turn, to map the environment, we need to know the position of the robot. Visual SLAM is a method that works on images, usually taken by one or more cameras, to perform both localization and mapping of a robot in a given reference frame. As the acronym of SLAM already tells, any algorithm which performs a localization of the camera and a mapping of the environment is considered a SLAM algorithm in this paper. Many other names for those algorithms exist, e.g., Visual Odometry, Match-Moving, Photogrammetry, Multi-View Reconstruction, Shape from X or Structure from Motion (SfM).

Although the basic ideas are similar in all algorithms, they can be distinguished by the following properties. The most obvious difference is the idea of Indirect (feature-based) against Direct methods. The Indirect methods transform image data into a feature space representation and then proceed with the camera pose and 3D structure information extraction, whereas the Direct methods directly use image data to perform the SLAM. In addition to that, recently, machine learning methods [2] gain increasing popularity. Although these could be a game-changer in this research field, up to now all of them either require large data sets specific to a certain scenario used for the mandatory training phase [3] or they are only able to reconstruct isolated objects in a scene [4]. Still, those approaches look very promising [5] and will soon provide mature alternatives to traditional feature extraction and tracking techniques.

Another property useful for classification is whether the algorithms work on one image (monocular) or two images (stereo) as their inputs per time step. While the monocular VSLAM requires algorithms to solely rely on the given image data, stereo cameras may use the distance between camera lenses (the baseline) to provide a metric reconstruction of the scene. Finally, the approaches can be separated into sparse or dense reconstruction algorithms. While sparse methods only reconstruct a subset of all available pixels in an image, dense methods try to reconstruct as many pixels as possible in an image, creating highly detailed 3D point clouds of the environment.

VSLAM approaches that use indirect methods include Blender's [6] integrated multi view library LibMV [7], the state-of-the-art of feature-based methods ORB-SLAM2 [8], Visual SfM [9], COLMAP [10], [11] and a recent framework called AliceVision, which is funded by the European Unions Horizon 2020 research and innovation programme [12].

Direct methods using VSLAM approaches are Direct Sparse Odometry (DSO) [13], which is analyzed in this contribution and its predecessor Large Scale Direct SLAM (LSD-SLAM) [14]. Hybrid implementations of direct and indirect methods exist, for example Fast Semi-Direct Monocular Visual Odometry (SVO) [15].

Most of the VSLAM algorithms are not evaluated regarding the robustness of the approaches. Often the results are only published with one available data set instead of considering different environments with sparse or dense environments. In addition, different error models are not analyzed, e.g., sensor noise, motion blur or vignette. The following contribution tries to close the gap by using Blender to simulate different environments. As an exemplary analysis, the Direct Sparse Odometry (DSO) is analyzed for three different environments, where especially the errors regarding rotation are evaluated in more detail.

This paper is organized as follows. In Section II, the DSO

is summarized, in Section III, the evaluation set-up is described. In Section IV, the evaluation is performed. The DSO is analyzed for three different environments "Calib City", "Blender City" and "Damaged Downtown". As the DSO is very sensitive to rotations, the analysis is extended to a scenario, where the camera is rotated in a circular path around a center point with increasing radii. The results are concluded in Section V and further research is pointed out.

II. Direct Sparse Odometry

In contrast to feature-based methods, direct algorithms do not transform the image to feature space but instead operate on raw image data. While feature-based methods optimize an error function which is based on a comparison between the reprojection of estimated 3D points in the image and their respective 2D feature positions called reprojection error, direct methods compare the difference of pixel intensities called photometric error. The differences between feature-based and direct methods are visualized in Figure 1.

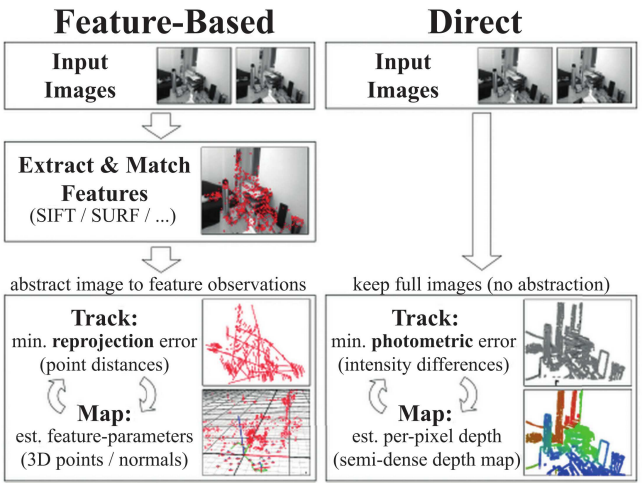

Figure 1. Comparison of indirect (feature-based) and direct methods for 3D reconstruction (image credit [16])

Recently, a surge of such direct methods can be observed [17]. Lately, DSO [13] has been published which is considered state-of-the-art and is analyzed in the following contribution. In this section, the core ideas are only introduced. The interested reader can find good in-depth explanations in [18] and [19].

A. Geometric Camera Calibration

The DSO relies on a proper camera calibration, where the typical intrinsic geometric parameters focal length and principal point have to be estimated. Consequently, only undistorted (rectified) images are used in the algorithm. Hence, these need to be estimated during the camera calibration procedure as well. Initially, every image is undistorted by the algorithm before the actual tracking is performed. The projection matrix from 3D to 2D space is defined as $\Pi_{\mathbf{c}} : \mathbf{R}^3 \rightarrow \mathbf{R}$ and the inverse back-projection is defined as $\Pi_{\mathbf{c}}^{-1} : \Omega \times \mathbf{R} \rightarrow \mathbf{R}^3$.

B. Photometric Camera Calibration

In addition to the geometric calibration, the DSO needs to calibrate the camera photometrically. This is necessary because direct methods assume a overall constancy in image brightness. A photometric calibration is used in the algorithm to adjust the brightness of consecutive image pairs in order to improve the robustness of the optimization. While this is beneficial for direct methods, it may cause feature-based VSLAM algorithms to suffer from a vanishing image gradient, for instance when images get too dark [20]. For further information regarding photometric calibration, the reader is referred to [21].

C. Point and Keyframe Management

In order to proceed, the DSO has to take care of some management tasks. For instance, it needs to determine what points $\mathcal{P}_i$ need to be selected in an image for reconstruction. Additionally, the DSO has to keep track of their visibility $\text{obs}(\mathbf{p})$ in consecutive images. Furthermore, the DSO needs to choose a subset from all images $\mathcal{F}$ to mark it as keyframes, which contain reliable depth information and are used as reference frames for tracking newly added images as illustrated in Figure 2.

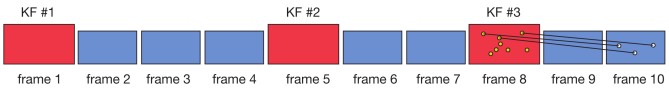

Figure 2. Keyframe management (red rectangles) chosen from all frames $\mathcal{F}$ (blue and red rectangles), points $\mathcal{P}_i$ (yellow dots) and observed points $\text{obs}(\mathbf{p})$ (white dots)

In the start of the algorithm, generally more keyframes are generated, approximately $5 \ldots 10$ per second. A few criteria for generating keyframes are used, such as the change of content in the field of view using optical flow, occlusions, dis-occlusions and strong changes in the relative brightness between frames. A standard setting is to keep 7 of those keyframes active for optimization. This is illustrated in Figure 3. A large number of point candidates (around 2000) is being identified in each keyframe, although only a subset of them is being tracked across frames. The reasons for that is to ensure that if for instance other active points leave the field of view or get occluded, new ones can be activated when necessary. Only this subset is used in the optimization process. In order to select evenly distributed points across the whole image, the image is divided into a 32x32 grid. In each grid, its mean gradient is added to a standard threshold. In each block, only pixels with a gradient larger than the mean of the region are selected. The region is divided further and pixels with a weaker gradient are added. These gradient thresholds are adapted throughout the sequence always selecting a predefined number of points in each keyframe. No photometric correction has been applied to the images at this point, yet. The points are tracked by searching for correspondences along the epipolar line. Furthermore, a depth and associated variance

is computed in order to further constrain the search interval in the subsequent frame.

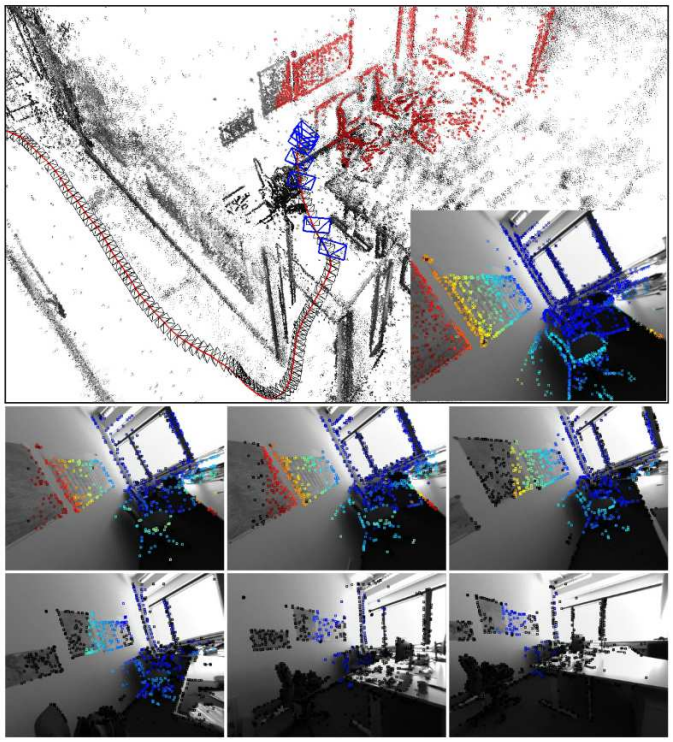

Figure 3. DSO: Keyframe management. Blue cameras in 3D view depict the seven keyframes below where a significant change in the field of view can be noticed (image credit [13])

D. Error function

At its core, Direct Sparse Odometry formulates a cost function that needs to be minimized. This is actually a windowed sparse bundle adjustment of direct image alignment, over the recently discussed frames $\mathcal{F}$, points $\mathcal{P}_i$ and observed points obs($\mathbf{p}$) in each of those frames. Although the cost function is constrained to a window spanned by only a few keyframes, the combination of the involved parameter set makes it a hard problem which can only be solved with efficient optimization techniques. One way to improve efficiency is to perform a coarse-to-fine tracking using image pyramids of varying scale. This way, a rough estimate of camera pose and 3D structure can be calculated quickly at the coarsest level and then refined using finer levels, making it also more robust to stronger changes in motion. The relative transformation from a reference camera pose $\mathbf{T}_i$ to a target camera pose $\mathbf{T}_j$ is expressed by a rotation $\mathbf{R}$ and a translation $\mathbf{t}$

$$\begin{bmatrix} \mathbf{R} & \mathbf{t} \\ 0 & 1 \end{bmatrix} := \mathbf{T}_j \mathbf{T}_i^{-1}. \tag{1}$$

This relative transformation can be used to express a reference point **p'** in one image in terms of a target point $\mathbf{p}$ in another image and vice versa. To do so, $\mathbf{p}$ is assigned an inverse depth value $d_\mathbf{p}$, reprojected to 3D space using the inverse projection matrix $\Pi_\mathbf{c}^{-1}$ with camera calibration parameters $\mathbf{c}$, rotated by $\mathbf{R}$ and translated by $\mathbf{t}$ and finally projected into the new view using the projection matrix $\Pi_\mathbf{c}$

$$\mathbf{p'} = \Pi_\mathbf{c}(\mathbf{R}\Pi_\mathbf{c}^{-1}(\mathbf{p}, d_\mathbf{p}) + \mathbf{t}). \tag{2}$$

Firstly, a cost function $E_{\mathbf{p}j}$ over a set of neighboring pixels $\mathcal{N}_\mathbf{p}$ around point $\mathbf{p}$ is formulated. The goal is to minimize the pixel intensities of a reference image I_j at position **p'** and a target image I_i at location $\mathbf{p}$ using the Huber norm $||\cdot||_\gamma$ [22] which is weighted by a gradient dependent factor $w_\mathbf{p}$

$$E_{\mathbf{p}j} := \sum_{p \in \mathcal{N}_\mathbf{p}} w_\mathbf{p} ||(I_j[\mathbf{p'}] - b_j) - \frac{t_j e^{a_j}}{t_i e^{a_i}}(I_i[\mathbf{p}] - b_i)||_\gamma, \tag{3}$$

where t_j and t_i are the exposure times of reference and target frames, respectively, and a_j, b_j, a_i and b_i are parameters of their respective brightness transfer functions.
The two-view loss function above can be extended to bundle adjustment over a window of frames to formulate the final photometric error function E_photo over every point $\mathbf{p}$ visible in a reference frame obs($\mathbf{p}$) from all points $\mathcal{P}_i$ in the current frame $\mathcal{F}$ of the recent window

$$E_\text{photo} := \sum_{i \in \mathcal{F}} \sum_{\mathbf{p} \in \mathcal{P}_i} \sum_{j \in \text{obs}(\mathbf{p})} E_{\mathbf{p}j}. \tag{4}$$

To summarize, the error function requires optimization over each points inverse depth $d_\mathbf{p}$, the camera intrinsics $\mathbf{c}$ from which the projection matrices are constructed, camera poses of the image pair from a reference $\mathbf{T}_i$ to a target camera pose $\mathbf{T}_j$, and their brightness transfer function parameters a_j, b_j, a_i b_i. This problem can only be solved iteratively, for example by a nonlinear least squares solver.
This photometric error is a highly non-convex and nonlinear cost function. Therefore a suitable algorithm for solving these kinds of least squares problems is required. DSO computes the set of parameters by using a Gauss-Newton algorithm. The DSO implements a dynamic adjustment of a scalar parameter that controls whether update steps are equal to Gauss-Newton or converge to gradient descent steps. This is a property of Levenberg-Marquardt algorithms.

III. Evaluation Set-Up

Three virtual environments are used for evaluation in Blender in which the camera trajectory is animated as a straight path with a 90 degree turn to the right. The first one being a simple scene consists of a floor and cubes, textured using a checkerboard pattern similar to that when calibrating cameras, hence, called "Calib City" (Figure 4). The second environment is a model of a city which is downloaded from BlendSwap[1] with the same camera trajectory (Figure 6). It is called "Blender City" in the following. The scene consists of many areas with no gradients. As third environment, a

[1]https://www.blendswap.com/blends/view/80580,accessed:2018-07-04.

highly detailed free city model[2] with a lot of changing image gradients and depth information at almost every pixel is used. The environment is called “Damaged Downtown” in the following. In this environment, a trajectory with a loop closure is evaluated. The evaluation is complemented with real-world analysis.

IV. Evaluation

The evaluation is performed for the environments “Calib City”, “Blender City” and “Damaged Downtown”, which are introduced in the preceding section. In addition, a systematic analysis of the sensitivity to rotations is performed. The evaluation is complemented by a real-world analysis.

A. Calib City

For the Calib City, almost no characteristic gradients are present, as all cubes look equal. Hence, it is one of the most difficult environments for the DSO, which is dependent on characteristic gradients in each frame. An exemplary view is depicted in Figure 4. It is obvious, how difficult this environment is for navigation. As a consequence, the DSO is neither able to follow a straight line nor to resolve the 90 degree turn. The results are depicted in Figure 5. Following the straight line leads to wave like movements, as a drift orthogonal to the movement direction is always present which results in a wrong estimation of the trajectory. The rotation by 90 degrees leads only to a 70 degree turn.

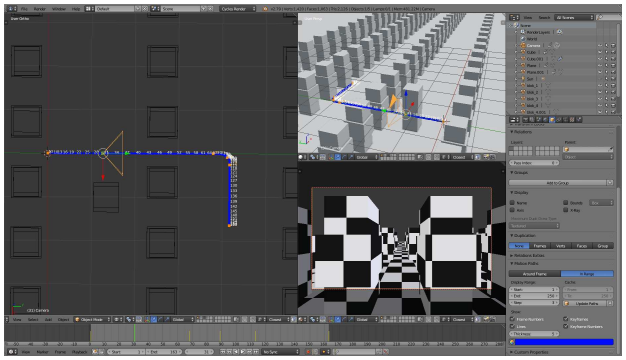

Figure 4. Exemplary view of Calib City: Virtual environment built from cubes with a checkerboard texture. A straight movement is followed by a 90 degree turn.

B. Blender City

The Blender City has more gradients in the image present than the Calib City, cf. Figure 6. As a consequence, the DSO is able to estimate the straight line in a good way, cf. Figure 7. There are enough characteristic features present to estimate a straight movement. But the rotation is still a big problem, as the 90 degree turn is overestimated. Especially, the drift to the left at the beginning of the right turn is not desirable for a good reconstruction of the trajectory. To summarize, it turns out that ideal scenes, where almost no gradients are visible, create a big challenge for the DSO algorithm. Using frames with characteristic gradients improves the ability to estimate movements with no rotation. Furthermore, a rotating camera creates notable errors in the final trajectory resulting in strongly overestimated or underestimated turns.

[2]http://open3dmodel.com/download/damaged-downtown-free-3d-model_7058.html, accessed: 2018-07-14.

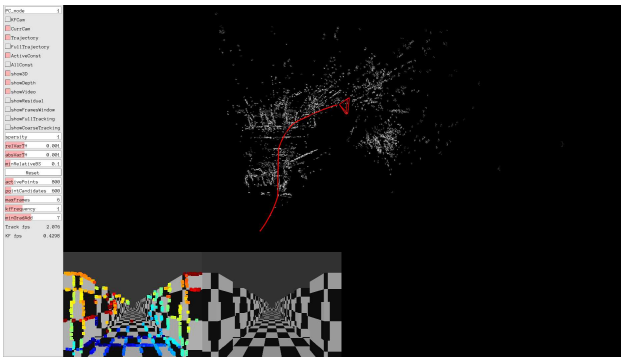

Figure 5. Estimated trajectory for a straight movement followed by a 90 degree turn in Calib City.

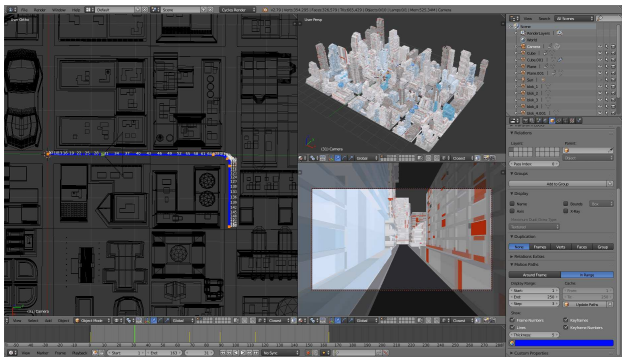

Figure 6. Exemplary view of Blender City: Virtual environment of a sci-fi-like city. A straight movement is followed by a 90 degree turn.

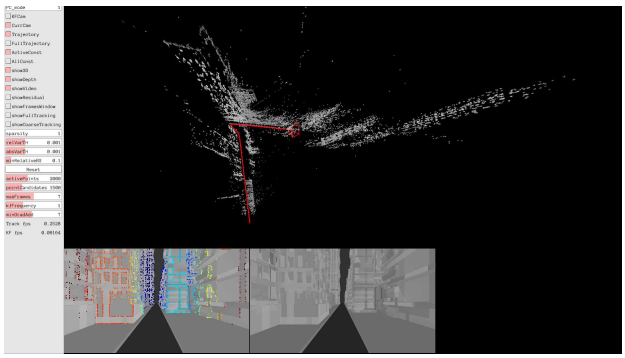

Figure 7. Estimated trajectory for a straight movement followed by a 90 degree turn in Blender City

C. Damaged Downtown

The estimation of the DSO is improved by using a highly detailed free city model with a lot of changing image gradients and depth information at almost every pixel. In Damaged

Downtown, a loop closure is tried to be achieved, cf. Figure 8. In this environment, the DSO is able to estimate the trajectory in parts of straight movement in a good way. These results are consistent with the results of the scenario Blender City. However, for this loop scenario a strong drift can be noticed when the camera is rotated. In opposite to Blender City, the rotation seems to be underestimated in most of the cases. This is visualized in Figure 8, where the camera is rotated with the movement. For the case of only moving in translational direction, the DSO is able to close the loop almost perfectly, cf. Figure 9, which emphasizes that the DSO is very sensitive to rotation. To summarize, even in highly detailed environments, the DSO is very sensitive to rotations. Without rotation the DSO is able to achieve loop closure. Therefore, the analysis regarding rotations is further extended in the next subsection.

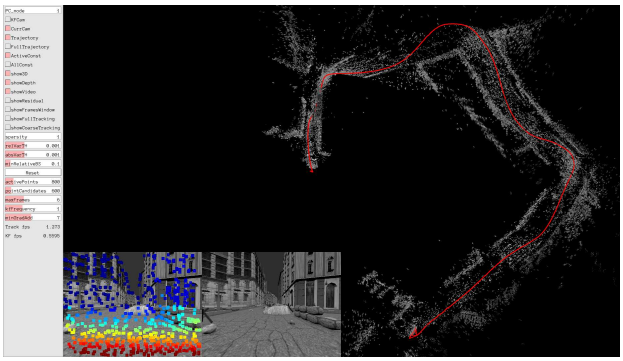

Figure 8. Results of the DSO for Loop Closure **with** rotation of the camera in Damaged Downtown

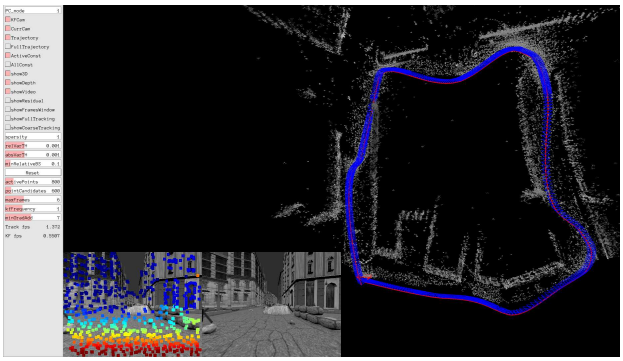

Figure 9. Results of the DSO for Loop Closure **without** rotation of the camera in Damaged Downtown

D. Further Analysis Regarding Rotation

In order to investigate the sensitivity to rotations without straight movement, the camera is rotated in a circular path around a center point with increasing radius in Damaged Downtown. The experiments are performed for the radii of 0 m and 3 m. The results are depicted in Figure 10 for a radius of 0 m and in Figure 11 for a radius of 3 m. The DSO is not able to close the loop in any of these cases due to drift independent of the rotation of the camera. To summarize, the drift increases with decreasing radii. It is also shown that the drift increases with ongoing rotation.

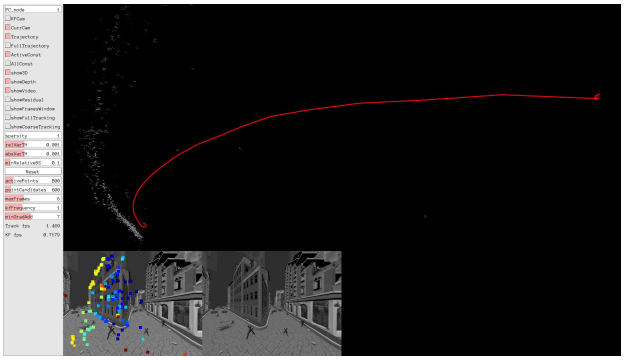

Figure 10. Virtual environment Damaged Downtown where the camera rotates around a point with a radius of 0 m

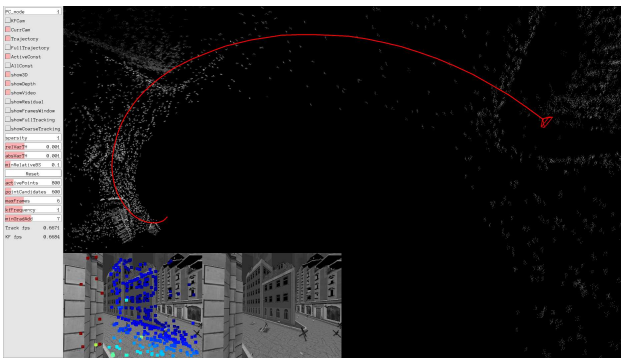

Figure 11. Virtual environment Damaged Downtown where the camera rotates around a point with a radius of 3 m

E. Real-world analysis and Conclusion

Further recordings using a GoPro Hero 4 Black rolling shutter camera show that even a suboptimal camera sensor without a prior photometric calibration can deliver good reconstruction results in certain situations. If the camera is moved slowly and carefully, the experienced drift is acceptable. Conversely, in situations with rapidly changing lighting conditions (i.e. in tunnels), accumulating motion blur or mostly planar structures the DSO either experiences a huge scale drift or fails entirely. In general, a complete failing of the algorithm is experienced many times, as there is no means to relocate the camera or use other sensors to support the camera pose estimation process. To conclude, the DSO is very sensitive in a real world setting and easily breaks when the camera movement is too quick, exposure changes too fast, the radius of a turn is too narrow or the scene does not provide sufficient clues about depth.

V. Conclusion and Future Work

Goal of this contribution is to motivate the systematic analysis of existing Visual SLAM algorithms. By way of example, the Direct Sparse Odometry (DSO) is analyzed by using Blender in simulation and in real-world. In simulation,

three different environments were used for evaluation ranging from very low detailed to high detailed worlds. It was shown that the DSO is very sensitive if the radius of a turn is too narrow or the scene does not provide sufficient clues about depth. In a real world setting, it was shown that the DSO easily breaks when the camera movement is too quick or the exposure changes too fast.

In the future, the work shall be extended to consider different error models like motion blur or vignette. In addition, more VSLAM algorithms shall be used to compare them with the existing DSO.

REFERENCES

[1] Sebastian Thrun, Udacity, Inc., "Artificial intelligence for robotics," 2018, accessed: 2018-07-04. [Online]. Available: https://in.udacity.com/course/artificial-intelligence-for-robotics--cs373

[2] V. Mohanty, S. Agrawal, S. Datta, A. Ghosh, V. D. Sharma, and D. Chakravarty, "Deepvo: A deep learning approach for monocular visual odometry," *CoRR*, vol. abs/1611.06069, 2016. [Online]. Available: http://arxiv.org/abs/1611.06069

[3] S. Vijayanarasimhan, S. Ricco, C. Schmid, R. Sukthankar, and K. Fragkiadaki, "Sfm-net: Learning of structure and motion from video," *CoRR*, vol. abs/1704.07804, 2017. [Online]. Available: http://arxiv.org/abs/1704.07804

[4] C. B. Choy, D. Xu, J. Gwak, K. Chen, and S. Savarese, "3d-r2n2: A unified approach for single and multi-view 3d object reconstruction," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2016.

[5] S. A. Eslami, D. J. Rezende, F. Besse, F. Viola, A. S. Morcos, M. Garnelo, A. Ruderman, A. A. Rusu, I. Danihelka, K. Gregor *et al.*, "Neural scene representation and rendering," *Science*, vol. 360, no. 6394, pp. 1204–1210, 2018.

[6] Blender Foundation, "blender.org - home of the blender project - free and open 3d creation software," 2018, accessed: 2018-07-04. [Online]. Available: http://blender.org

[7] Blender Contributors, "Blender foundation: Libmv," 2018, accessed: 2018-07-04. [Online]. Available: https://developer.blender.org/tag/libmv/

[8] R. Mur-Artal and J. D. Tardós, "Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255–1262, 2017.

[9] Changchang Wu, "Visualsfm : A visual structure from motion system," 2018, accessed: 2018-07-04. [Online]. Available: http://ccwu.me/vsfm/

[10] J. L. Schönberger and J.-M. Frahm, "Structure-from-motion revisited," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[11] J. L. Schönberger, E. Zheng, M. Pollefeys, and J.-M. Frahm, "Pixelwise view selection for unstructured multi-view stereo," in *European Conference on Computer Vision (ECCV)*, 2016.

[12] Czech Technical University (CTU) et. al., "Alice vision: Photogrammetric computer vision framework," 2018, accessed: 2018-07-04. [Online]. Available: https://alicevision.github.io/

[13] J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry," *IEEE transactions on pattern analysis and machine intelligence*, vol. 4, 2017.

[14] J. Engel, T. Schöps, and D. Cremers, "Lsd-slam: Large-scale direct monocular slam," in *European Conference on Computer Vision*. Springer, 2014, pp. 834–849.

[15] C. Forster, M. Pizzoli, and D. Scaramuzza, "Svo: Fast semi-direct monocular visual odometry," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE, 2014, pp. 15–22.

[16] T. Schöps, J. Engel, and D. Cremers, "Semi-dense visual odometry for ar on a smartphone," in *Mixed and Augmented Reality (ISMAR), 2014 IEEE International Symposium on*. IEEE, 2014, pp. 145–150.

[17] D. Cremers, "Direct methods for 3d reconstruction and visual slam," in *Machine Vision Applications (MVA), 2017 Fifteenth IAPR International Conference on*. IEEE, 2017, pp. 34–38.

[18] T. Schöps, "Semi-dense visual slam on mobile devices," 2014.

[19] J.-J. Engel, "Large-scale direct slam and 3d reconstruction in real-time," Ph.D. dissertation, Technische Universität München, 2017.

[20] N. Yang, R. Wang, X. Gao, and D. Cremers, "Challenges in monocular visual odometry: Photometric calibration, motion bias, and rolling shutter effect," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, 2018.

[21] P. Bergmann, R. Wang, and D. Cremers, "Online photometric calibration of auto exposure video for realtime visual odometry and slam," *IEEE Robotics and Automation Letters (RA-L)*, vol. 3, pp. 627–634, April 2018.

[22] P. J. Huber, "Robust statistics," in *International Encyclopedia of Statistical Science*. Springer, 2011, pp. 1248–1251.