The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Volume XLII-2/W8, 2017
5th International Workshop LowCost 3D – Sensors, Algorithms, Applications, 28–29 November 2017, Hamburg, Germany

A VARIANT OF LSD-SLAM CAPABLE OF PROCESSING HIGH-SPEED LOW-FRAMERATE MONOCULAR DATASETS

S. Schmid[a], D. Fritsch[b]

[a] Daimler AG - stephan.s.schmid@daimler.com

[b] Institute for Photogrammetry, University of Stuttgart - dieter.fritsch@ifp.uni-stuttgart.de

Commission II

KEY WORDS: monocular visual odometry, filter-based visual odometry, Augmented Reality

ABSTRACT:

We develop a new variant of LSD-SLAM, called C-LSD-SLAM, which is capable of performing monocular tracking and mapping in high-speed low-framerate situations such as those of the KITTI datasets. The methods used here are robust against the influence of erronously triangulated points near the epipolar direction, which otherwise causes tracking divergence.

1. INTRODUCTION

Our work is motivated by automotive Augmented Reality applications, in particular by aiming to augment a live camera feed of a forward-looking vehicle-mounted camera. Augmented Reality applications typically need to solve two major problems: The first problem is to provide localization within the real world in order to correctly place virtual content, which often comes from an external source (e.g. a digital map or a content provider). The second problem is to obtain a notion of the surrounding real world in order to facilitate integration of the virtual content with real objects. A cost-effective solution to the second problem for video-based Augmented Reality applications is to triangulate the objects of the real world from the (monocular) camera images. This omits both the cost of an additional sensor such as LIDAR or the second camera necessary for a stereo setup and the complexity of having to calibrate the various sensor relative to each other, which is a significant factor for mass-produced systems. Since one needs to know the camera movement to perform object triangulation, one quickly arrives at visual odometry as well as visual SLAM, the latter of which actually solves both of the above problems at the same time. In our case, we use a modified version of LSD-SLAM (Engel et al., 2014), which is a semi-dense visual SLAM method, to obtain depth information for each frame of the camera image sequence.

Since there are highly specialized methods for localization relative to a global map, we will mostly ignore the global mapping and tracking capabilities of LSD-SLAM (in fact, we removed the global mapping portions of LSD-SLAM in our variant to simplify development). Instead, we will focus on the frame-to-frame tracking, which is necessary for obtaining good disparity maps, as well as the disparity maps themselves.

In the sequel, we give a short overview of the related work. In the literature, two main branches of approachs to visual odometry have evolved (cf. e.g. (Fuentes-Pacheco et al., 2015, Younes et al., 2016)): Methods following the filter-based approach are applied in some of the oldest works on visual odometry. The general concept of these methods is to use the depth estimates of the scene points as the state vector of a Kalman filter or similar. A main challange of filter-based methods is the fact that the size of the covariance matrix of the Kalman filter is quadratic in the number of scene points, which makes scaling filter-based methods to large numbers of scene points computationally expensive (Fuentes-Pacheco et al., 2015). Due to this obstacle and in particular after an influental analysis in 2010 (Strasdat et al., 2010), the local bundle adjustment approach has been established in various methods cf. e.g. (Younes et al., 2016). Here, batches of several consecutive frames are processed at once via bundle adjustment (often keyframes are inserted at the boundaries of such batches). Thus, incremental processing is combined with bundle adjustment, which is by itself a well-established and highly efficient approach (Triggs et al., 1999).

The rest of the article is structured as follows. In section 2, we give an introduction to our method, C-LSD-SLAM, and how it originated from its precedessor LSD-SLAM. In section 3, we describe a stabilization method for C-LSD-SLAM for high-speed forward movement. Section 4 describes the parallelization method used in C-LSD-SLAM and why it differs from the well-established parallel and tracking paradigm (Klein and Murray, 2007). Section 5 gives some qualitative and quantitative evaluation.

2. C-LSD-SLAM

LSD-SLAM is a method for performing visual SLAM on a monocular image sequence. It is a direct method thus tracking and mapping operate directly on the image intensities instead extracting features as in feature-based methods. LSD-SLAM is a keyframe-based method, so it distinguishes keyframes from "normal" image frames in the sense that "normal" frames have an pose estimate, whereas keyframes additionally feature an estimated depth map and can thus serve as a reference for tracking frames. LSD-SLAM natively uses several different threads for different functions:

- tracking
- depth estimation (keyframe updating) and generation of new keyframes
- global map optimization, loop closure

Tracking is performed for each new frame, using the newest keyframe with its estimated depth map as tracking reference. If the tracked frame's pose differs more than a certain threshold from the keyframe's pose, the depth estimation thread will generate a new keyframe in

its next iteration, which is performed by propagating (i.e. reprojecting) the depth map from the currently used keyframe to the new keyframe. Thus LSD-SLAM generates a new keyframe every several frames.

While this works well in situations where the content changes only gradually between frames (slow movement, high framerate), it causes problems if there is a large change of perspective in comparatively few frames. E.g. in the KITTI datasets (Geiger et al., 2013), there frequently are 90°-turns in only 20-30 frames and the camera moves past a large amount of occluding objects such as parking cars. These situations need a large number of intermediate steps (i.e. frames suitable as tracking reference, which are keyframes in our case) to properly establish a connection between these different perspectives. If the keyframes are far apart, the depth map of a new keyframe is usually much less complete than the depth map of the preceding keyframe. Also, significant areas of the frames between two keyframes do not overlap with the older keyframe, preventing these image parts from being used in depth estimation. Overall, the algorithm is forced to discard the information of certain parts of the camera images due to insufficient overlap of the selected frames. If the keyframes are too far apart, this can cause the algoritm to loose more information to keyframe reprojection than gaining from depth estimation, with the result ultimately being tracking failure.

In our variant of LSD-SLAM, which we call continuous LSD-SLAM or C-LSD-SLAM, we solve this problem by using each frame as keyframe. While we have also obtained good results by using e.g. every second frame as keyframe, this yields the best results with the large changes of perspective mentioned above. Additionally, this format is directly suitable for Augmented Reality use-cases which require depth information for each camera frame, e.g. occlusion computation between real and virtual objects.

Instead of the multithreaded approach used by LSD-SLAM, C-LSD-SLAM uses a sequential approach for tracking and depth estimation (instead each step is parallelized, cf. section 4): At the beginning of a frame cycle, the most recent frame is a keyframe, i.e. it has a depth map. Let us call this the "old" keyframe. At the arrival of a new frame, the following steps are performed:

1. Preprocessing of the new frame (format conversion, rectification etc.)
2. Tracking of the new frame relative to the old keyframe.
3. Updating the depth map of the old keyframe by triangulation between the old keyframe and the new frame
4. Regularization of the depth map of the old keyframe
5. Reprojection of the depth map of the old keyframe to the new frame. This way, the new frame becomes the next keyframe.
6. Regularization of the depth map of the new frame and hole filling

C-LSD-SLAM is robustified againgst the system instability caused driving towards erroneous distance estimates near the epipolar line (cf. fig. 2) as described in the following section 3. This feature is crucial for stable system operation in situations with high-speed forward motion.

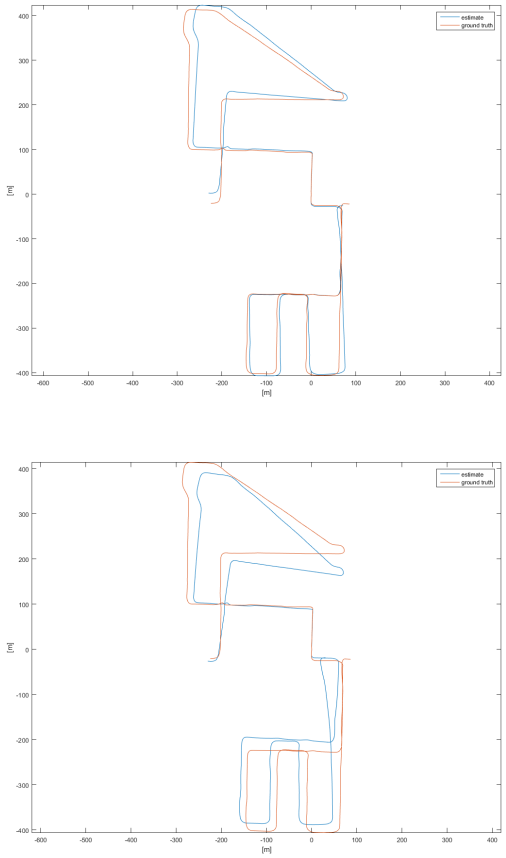

Figure 1. Tracking result for sequence 08 of the KITTI odometry challange (Geiger et al., 2012). Upper plot: full resolution input (1226×370). Lower plot: half resolution input (584×184). The trajectories are aligned at the middle of the sequence. We use a simple variant of ground plane estimation (cf. e.g. (Song and Chandraker, 2014)) to determine the absolute scale.

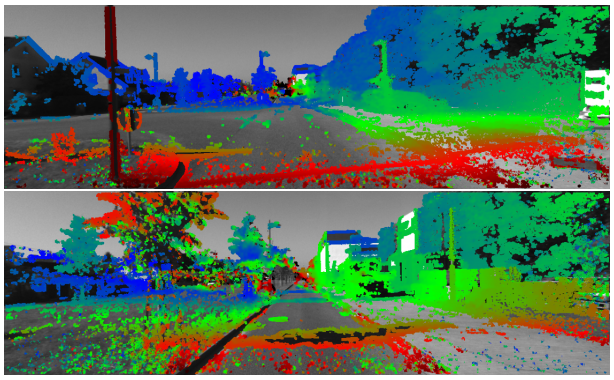

Figure 2. Instablitiy of non-robustified C-LSD-SLAM on sequence 06 of the KITTI odometry challange. Distance estimates of the depth maps are colored from blue (far) over green to red (near). The upper image shows invalid distance estimates appearing near the epipolar direction, which then propagate mainly to the left. The lower image shows the situation several frames later. Here, invalid distance estimates have overtaken a large part of the image, causing tracking divergence several frames later.

3. C-LSD-SLAM AS A FILTER-BASED VISUAL ODOMETRY METHOD, PREVENTING TRACKING INSTABILITY IN FORWARD MOTION

Note that C-LSD-SLAM only maintains the state of the last frame for processing the succeeding frame, so it is effectively a filter-based visual odometry method.

The original LSD-SLAM follows the local bundle-adjustment / keyframe approach. The return to the filter-based approach was originally a side-effect of the demands given by the high speeds of vehicle-mounted cameras and by the necessity of obtaining a depth map for each frame for usage in Augmented-Reality applications. Note however, that C-LSD-SLAM (as LSD-SLAM) maintaints confidence estimates of the depth estimates *per pixel* instead of using a (full) coveriance matrix as in traditional filter-based approachs. That is C-LSD-SLAM effectively ignores the cross-correlation between different pixel estimates. Hence, the computational cost of maintaining uncertainities is linear in the number of pixel for C-LSD-SLAM instead of quadratic for traditional filter-based approachs. This way, C-LSD-SLAM can be easily scaled up to large image resolutions. Furthermore, this way of maintaining uncertainities can be motivated by the fact that if the camera movement is well known, the positions estimates of the scene points are approximatively uncorrelated, and if the scene points are well known, the camera movement can be determined well. The challenge, however, is to manage the feedback between the depth estimation stage and the camera pose estimation stage in order to prevent errors in each stage from boosting each other in a runaway fashion. In the following, we investigate and mitigate a particular instability mode of C-LSD-SLAM, although we do not know how much of it is caused by the above simplification of the covariance matrix and how much is caused by the general high nonlinearity of fast forward motion.

For sufficiently fast forward motion, C-LSD-SLAM suffers from a type of instability where erroneous depth estimates propagate over the image, eventually causing tracking divergence or failure (cf. fig. 2): Near the epipolar direction, distance estimation is difficult due to small relative object movement as well as the epipolar direction itself being subject to estimation errors. Due to the fact that the camera moves in the epipolar direction, these erroneous estimates have a tendency to cover larger and larger parts of the image, which can cause tracking divergence.

The concept underlying our method for preventing this behaviour is to prevent the noisy distance estimates near the epipolar direction from gaining influence in larger image parts by suitable weighting and depth estimation strategies.

Erroneous distance estimates near the epipolar direction can be seperated into two classes, distinguished by whether they are further away or closer than the real distance. The estimates which are too far away tend to stay close to the epipolar direction, i.e. they stay within the region of high uncertainity surrounding the epipolar direction. Our only measure targeting these estimates is disabling the usage of negative inverse depth results in the depth estimation component, since negative inverse depth estimates can be caused by incorrect estimation of the epipolar direction.

On the other hand, distance estimates which are too close tend to move rapidly away from the epipolar direction, i.e. to a region where depth estimation uncertainities are much lower than near the epipolar direction. If it happens to be that the bogous estimate is confirmed in the depth estimation step (by a mismatch), we have an incorrect depth estimate on which the system places high confidence. Due to their closeness and high confidences, incorrect depth estimates of this type have the capability of strongly affecting the tracking component, in particular the estimation of the epipolar direction. Misestimation of the epipolar direction in turn cases systematic underestimation of pixel distances on one side of the epipolar direction (on the other side, distances are overestimated), which can cause error runaway.
To prevent this, we penalize all approaching pixels when reprojecting them from a keyframe to a new keyframe. In particular, we multiply the variance estimate for their inverse depth by the factor

$$F(dist_{old}, dist_{new}) := \left(\frac{dist_{old}}{dist_{new}}\right)^c,$$

where the exponent c is a positive number. For distance estimates which are too small, the fraction $\left(\frac{dist_{old}}{dist_{new}}\right)$ is larger than the "true" value, so these estimates are penalized more heavily than correct estimates or estimates which are too large.

Experimentally, we have found values from $c = 8$ up to $c = 13$ to give good results. We use $c = 13$ for evaluation. Higher values for c tend to make the whole system more stable when faced with high movement speeds or scenes with moving objects near the epipolar direction. However, it should also be noted that decreasing depth estimate confidences means increasing the confidence region of the depth estimate. This causes the depth estimation module to perform stereo matching for longer intervals of the epipolar line, thereby increasing the required computational effort.

4. PARALLELIZATION

Many visual odometry methods follow the parallel tracking and mapping paradigm for parallelization (e.g. (Forster et al., 2014, Engel et al., 2014, Mur-Artal et al., 2015)), which was popularized by the method aptly called parallel trackand mapping (PTAM) (Klein and Murray, 2007) in 2007. A major reason given for this is the ability to decouple tracking and mapping in the context of real-time processing. In particular, this allows different timing constraints for tracking and mapping. E.g. while tracking usually runs in a real-time fashion, the mapping component can perform more complex processing on select keyframes or can process batches of consecutive frames in the manner of local bundle adjustment. Running tracking and mapping in seperate threads which can run concurrently firstly allows the tracking thread to preempt other computations, which may be used to meet real-time constraints, and secondly improves usage of available ressources in multi-core systems. We argue that the efficient usage of multi-core systems alone does not justify the high amount of complexity required by this style of concurrent processing any more. Indeed, since typically tracking uses a map as reference and mapping uses tracking results for generating and updating maps, tracking and mapping operate on and modify the very same data, which requires fairly complex synchronization methods to prevent race conditions without overly harming efficiency. On the other hand, processing units with multiple cores, each featuring simultaneous multithreading and/or SIMD vectorization, have as of 2017 become ubiquitous. E.g. in the desktop segment, current high-end systems have dozens of logical cores and the newest SIMD variant of the x86 architecture (AVX-512) features 512-bit vector registers. Multi-core systems are standard in mobile phones, with SIMD provided e.g. by ARM's NEON instruction set. GPUs have been using massively parallel processing for decades, with GPUs featuring high numbers of wide and highly multithreaded SIMD units.

Thus, in order to effectively make use of these kind of parallelized computing units, each significant processing step of a vi-

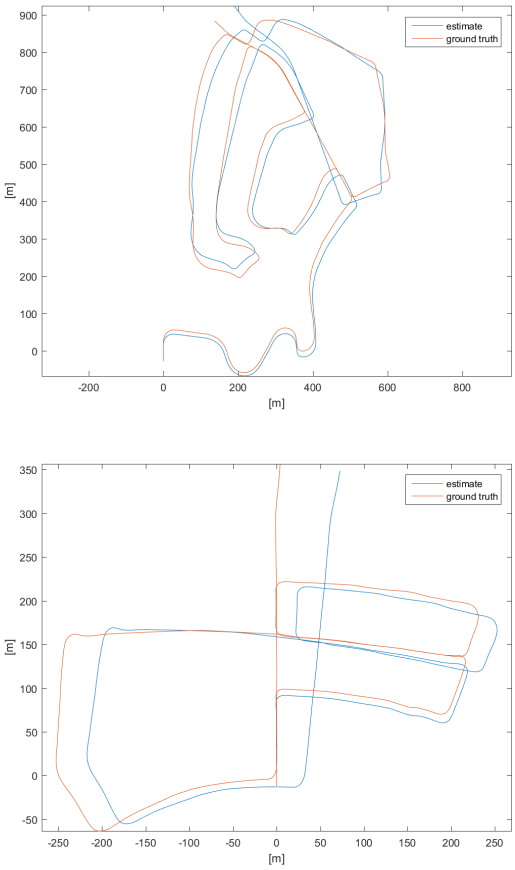

Figure 3. Tracking result for sequences 02 (upper plot) and 05 (lower plot) of the KITTI odometry challange.

sual odometry method should be fully parallelized and vectorized, otherwise that processing step will become a bottleneck. This way, each processing step can make usage of all of the available processing power. The choice whether different processing steps run concurrently or sequentially is then mainly a choice of latency and of thread synchronization complexity, not of total processing time.

Parallelization is a major part where C-LSD-SLAM has been streamlined over LSD-SLAM. Parallelization is facilitated by the fact that most processing steps consist of "do x for all pixels independently, possibly followed by a parallel reduction step". All processing steps of C-LSD-SLAM can employ an arbitrary number of threads and most are vectorized. This way, the different processing steps can be performed sequentially without performance penalty, which greatly reduces the thread synchronization complexity compared to LSD-SLAM.

5. EVALUATION

We use the sequences of the KITTI odometry benchmark(Geiger et al., 2012) for evaluation. The main reason for this choice is that these datasets feature very long sequences, hence the stability of C-LSD-SLAM can be investigated which was a major issue in the development of C-LSD-SLAM.

5.1 Initialization

For visual odometry systems, initialization is a delicate step as misestimation of the initial state may cause the system to diverge

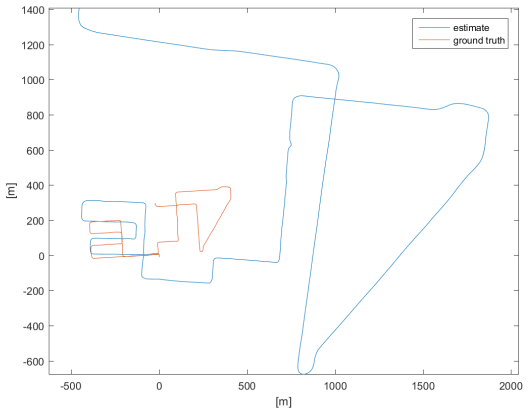

Figure 4. Tracking result for the sequence 08 of the KITTI odometry challange, with absolute scale estimation enabled only at the beginning of the sequence. The scale drifts towards larger scales.

due to the nonlinearity of the problem. For initialization, LSD-SLAM initializes the depth map of the first frame randomly. C-LSD-SLAM basically follows the same approach, but chooses a different distribution for the initial candidates. In particular, LSD-SLAM chooses the candidates for the inverse depths uniformly between a positive minimum value (maximum distance) and a maximum value (minimum distance). C-LSD-SLAM sets the minimum inverse depth to zero (i.e. infinite distance). This way, candidates for all distances are provided by the initialization (the minimum distance has no practical meaning since the scale factor is chosen arbitrarily by the system) and thus the initial state fits a much wider range of scenes. Notably for vehicle mounted cameras facing forwards, the region the camera is moving towards is the road in front of the vehicle, so it typically features large distances. The LSD-SLAM variant does not have viable candidate depths for this region, so the system needs to first reject all of these candidates before it can create valid depth estimates in this region, which is a process that often takes many frames. In the C-LSD-SLAM variant, convergence is much quicker, which also reduces the likelihood of the system diverging at initialization.

C-LSD-SLAM additionally features an initialization technique specifically targeted at car-mounted cameras: A car cannot turn without driving a curve, so a car-mounted camera (approximatively) does not rotate while moving in a straight line. Moving in straight line is the most difficult situation for initialization. C-LSD-SLAM assumes that the camera orientation does not change in the first few frames. If the camera orientation indeed stays approximatively the same (i.e. the vehicle is driving in a straight line), this greatly helps the initialization process. If the camera turns, the vehicle does not drive in a straight line, which is a much easier initial situation where the system can usually initialize successfully despite having made incorrect assumptions.

5.2 Tracking stability

The tracking of a new frame relative to the last frame proceeds by direct image alignment via hierarchical optimization. In particular for scenes with repetitive structures, the success of this optimization scheme depends on availability of a reasonable initial guess. To improve the performance in situations with high turnrates, C-LSD-SLAM uses the pose change between the last frame and its predecessor as the initial guess for the pose change between the last frame and the new frame.

Generally, C-LSD-SLAM is highly stable. We tested C-LSD-SLAM on all 21 sequences of the KITTI odometry challange. There are three sequences (01, 12 and 21) on which C-LSD-SLAM shows signs of instablity, that is significant areas near the epipolar direction with incorrect depth estimates. All of these are on highway scenes, with regions of instability typically initiating near moving vehicles near the epipolar direction. Only in sequence 21, the system becomes actually unstable. In sequence 21, the system switches several times between periods of correct operation and divergent behaviour. This behaviour may be attributed to the fact that highway scenes have very low scene structure, hence it is difficult for the system to eliminate the influence of moving objects. C-LSD-SLAM performs very well in urban scenes. In particular situations where the vehicle turns around corners, which are difficult for the original LSD-SLAM due to the fast change of perpective, are handled easily by C-LSD-SLAM. In fact, during turns C-LSD-SLAM reliably recovers even from a completely diverged tracking state (e.g. due to initialization failure).

5.3 Accuracy

Without a method to determine the absolute scale, C-LSD-SLAM shows a consistent scale drift towards increasing scales (cf. fig. 4). In order to make reasonable comparisons with ground truth trajectories, we use a simple variant of ground plane estimation (cf. e.g. (Song and Chandraker, 2014)) to determine the absolute scale. It is performed by fitting a plane to the depth estimates in a small image area corresponding to the road patch in front of the vehicle and then using this plane to estimate the height above the ground. This method works only if the the road in front of the vehicle has sufficient texture to allow depth estimates, which fails e.g. for very smooth road surfaces or high speeds (motion blur).

In figures 1 and 3, we compare tracking results with the ground truth data provided by KITTI. The rotational errors are typically around $0.004\mathrm{deg}/\mathrm{m}$ in the training datasets of the KITTI odometry benchmark at full image resolution, which is neither particularly good nor particularly bad for a monocular slam method. Translational errors can be quite high, which is caused by the fact that there are situations for which the basic absolute scale estimation described above simply does not work.

A reason for the low accuracy is probably that while C-LSD-SLAM is capable of extracting almost all information from the images due to processing each frame in a direct fashion, the stabilization method described above discards much information, which decreases the filter gain and the achievable accuracy.

5.4 Performance

At full resolution (ca. 1200×370px) of the KITTI datasets, C-LSD-SLAM runs at about 7fps on a (rather old) Intel Core i5 M540 (dual core, 2.53GHz) and at about 30 fps at half resolution (ca. 600×180px). On a more recent Intel Core i7-6820HQ (quad core, 2.7GHz), we obtain about 20fps at full resolution.

6. CONCLUSION

We have modified the keyframe based LSD-SLAM into a filter based monocular visual odometry method which is capable of functioning in situations with high forward motion and low framerates. For preventing instability in forward motion, we have developed a method for modifying depth confidences in such a way that invalid depth estimates are prevented from gaining too much influence. The method performs well in urban scenes and can specifically deal very well with fast changes in perspective, but struggles with scenes with little structure, in particular if there are moving objects present.

REFERENCES

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

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

Fuentes-Pacheco, J., Ruiz-Ascencio, J. and Rendón-Mancha, J. M., 2015. Visual simultaneous localization and mapping: a survey. *Artificial Intelligence Review* 43(1), pp. 55–81.

Geiger, A., Lenz, P. and Urtasun, R., 2012. Are we ready for autonomous driving? the kitti vision benchmark suite. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*.

Geiger, A., Lenz, P., Stiller, C. and Urtasun, R., 2013. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*.

Klein, G. and Murray, D., 2007. Parallel tracking and mapping for small ar workspaces. In: *Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on*, IEEE, pp. 225–234.

Mur-Artal, R., Montiel, J. M. M. and Tardos, J. D., 2015. Orb-slam: a versatile and accurate monocular slam system. *IEEE Transactions on Robotics* 31(5), pp. 1147–1163.

Song, S. and Chandraker, M., 2014. Robust scale estimation in real-time monocular sfm for autonomous driving. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1566–1573.

Strasdat, H., Montiel, J. and Davison, A. J., 2010. Real-time monocular slam: Why filter? In: *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, IEEE, pp. 2657–2664.

Triggs, B., McLauchlan, P. F., Hartley, R. I. and Fitzgibbon, A. W., 1999. Bundle adjustment-a modern synthesis. In: *International workshop on vision algorithms*, Springer, pp. 298–372.

Younes, G., Asmar, D. and Shammas, E., 2016. A survey on non-filter-based monocular visual slam systems. *arXiv preprint arXiv:1607.00470*.