Estimation of Absolute Scale in Monocular SLAM Using Synthetic Data

Danila Rukhovich drk@zurich.ibm.com
Daniel Mouritzen dmo@zurich.ibm.com
Ralf Kaestner alf@zurich.ibm.com
Martin Rufli mru@zurich.ibm.com
Alexander Velizhev ave@zurich.ibm.com
IBM Research - Zurich, Switzerland

Abstract

This paper addresses the problem of scale estimation in monocular SLAM by estimating absolute distances between camera centers of consecutive image frames. These estimates would improve the overall performance of classical (not deep) SLAM systems and allow metric feature locations to be recovered from a single monocular camera. We propose several network architectures that lead to an improvement of scale estimation accuracy over the state of the art. In addition, we exploit a possibility to train the neural network only with synthetic data derived from a computer graphics simulator. Our key insight is that, using only synthetic training inputs, we can achieve similar scale estimation accuracy as that obtained from real data. This fact indicates that fully annotated simulated data is a viable alternative to existing deep-learning-based SLAM systems trained on real (unlabeled) data. Our experiments with unsupervised domain adaptation also show that the difference in visual appearance between simulated and real data does not affect scale estimation results. Our method operates with low-resolution images (0.03 MP), which makes it practical for real-time SLAM applications with a monocular camera.

1. Introduction

Monocular visual SLAM allows the translation and rotation of a moving camera and a sparse map representation to be determined, but only up to scale. This paper targets the problem of absolute scale estimation using sequences of images captured by a monocular camera. We consider the most general formulation of this problem but make no assumptions about the observed scene and its objects.

Estimation of scale is important for several reasons. First, it helps recover metric feature locations, which is valuable for numerous real-world applications. Other applications that do not require absolute scale also benefit from its estimation because negative effects caused by scale drift are reduced. For example, scale drift makes it more difficult to detect loops, and selecting key frames might not take into account actual camera displacements. In addition, scale drift does not allow scale correction by optimizing just one global scale parameter.

Additional sensors might be used to overcome the scale estimation problem, including IMU, GPS, LiDAR, stereo or depth cameras. However, they raise the cost, complexity, power consumption and weight of the entire system and thus reduce the number of possible applications. Therefore, the ability to estimate scale using but a single camera would be very beneficial.

The success of deep learning has opened new options for SLAM systems. For example, single image depth prediction makes it possible to retrieve absolute depth maps from color images. This might solve the scale estimation problem as well and improves SLAM systems in general [31, 33]. However, this approach solves a much more complex problem, requires a lot of training data and therefore might be considered too computationally expensive for a given scale estimation problem. End-to-end deep learning approaches for SLAM were also recently introduced [32, 22]. In this case, absolute scale might be estimated as well if relative camera poses with absolute scale are used for training. For the time being, complete end-to-end SLAM approaches require a modern GPU unit, which limits applications of these methods.

In this paper, we focus only on estimating the scale, which might then be integrated into classical SLAM systems. Comparison of classical and deep learning-based SLAM systems or solving the full six-degrees-of-freedom (DOF) pose estimation problem is beyond the scope of this work.

An elegant and lightweight approach for scale estimation using CNNs is described in [10]. The absolute distance (or speed) between two consecutive images is estimated independently for each pair. This helps reduce the scale drift ef-

fect significantly. The approach is very general (no explicit assumptions are made about the observed scene or objects inside it), is applicable to monocular SLAM and can be easily integrated into existing SLAM systems. This paper considers the method established in [10] as a baseline.

Our detailed analysis of results [10] on the KITTI dataset [12] highlighted several issues. First, we observed considerable systematic errors as the camera turned, see Figure 4, which might be attributed to a strong domination of pure forward camera movements in the training set. Another issue is related to the sensor configuration. Specifically, a camera placed at an offset to the vehicle's center of rotation constrains the possible combinations of rotation and translation values. For example, a front camera would never experience a pure rotation around the camera center because the global rotational center of a conventional vehicle is located on the rear wheel axis [26]. This means that actual camera rotation will always be combined with a certain translational movement. Adapting to new sensor configurations will require new data collection efforts. The last issue pertains to absolute scale estimation accuracy. There are often significant changes in the estimated distances between consecutive image pairs, even when the speed is relatively constant. Based on these arguments, the following goals of this paper can be formulated as follows:

1. Improve the state-of-the-art accuracy of absolute scale estimation
2. Improve estimates for camera turning movements
3. Improve robustness against variations of the sensor configuration and environmental conditions.

Our solution to the first and the second problems is to modify the neural network architecture with respect to the baseline approach [10] by increasing complexity of the network and adding a recurrent neural layer. These network modifications lead to significant reduction of the training loss to the very low values. This fact makes unnecessary further investigation of the network architecture, so we do not focus on this topic in the paper. The third issue is solved by using autonomous driving simulators, with which we can easily model any type of sensor configuration and generate a large amount of training data with wide variations of lighting and weather conditions. Recent works in similar domains (depth prediction [3] or optical flow generation [6]) show promising results when synthetic data is used. Although our simulated training environment differs significantly from the KITTI dataset [12] (for example, there are no parked cars), the trained network is still able to generalize to new and unseen environments. We will show that training on synthetic data yields comparable results to models trained on real data. In addition, we evaluate the influence of image photorealism for the problem at hand.

The paper is organized as follows. Section 2 presents an overview of existing methods, and Section 3 describes proposed network architectures, datasets and domain adaptation strategies. Section 4 presents evaluation results and compares different methods. Concluding remarks and discussion are available in Section 5.

2. Related work

This section describes previous work on the scale estimation problem for monocular SLAM. Our discussion covers the following approaches: (i) using explicit knowledge about the scene, (ii) using explicit soft assumptions about the scene, (iii) general, assumption-free approaches.

The first approach explicitly uses information about the scene, *e.g*. a 3D model with absolute scale. The scale of image-based reconstruction is estimated using correspondences between images and the 3D model. A known 3D model is a very strong assumption, which strongly limits the applicability of these approaches. These methods were introduced more than 30 years ago [25] and are mentioned primarily for completeness.

The second approach uses soft assumptions about the observed scene. This might be a known height of the camera above the ground plane [15, 14, 13, 35] or information about absolute sizes of objects presented in a scene [9, 21, 27, 29, 4]. In the first case, a position of the ground plane is estimated from space 3D points of the reconstructed scene, and the distance between camera center and this plane is constrained. In the second case, a pretrained object detector for a defined set of object classes (*e.g*. cars) is employed to incorporate general knowledge of object sizes into the optimization problem.

The main disadvantage of both these approaches are the assumptions themselves. For instance, the constraint on the camera height is applicable only for cameras mounted on vehicles and assumes this height is known beforehand and is constant during image recording. Analogously, relying on having certain classes of objects in a scene fails when none of these objects are present. Moreover, observed objects might be only partially visible, which makes their size estimates inaccurate. Objects might also have intra-class variations in size (*e.g*. different types of cars), which additionally decreases the accuracy of the scale estimates. In order to mitigate those issues, different flexible schemes using object detection have been introduced. For example, [9] introduces so-called object bundle adjustment, which optimises 3D landmark positions associated with objects of known size. The work in [30] fuses single detections from a generic object detector within a Bayesian framework. Using the nonholonomic constraints of wheeled vehicles (*e.g*. cars, bikes or differential drive robots) was proposed in [26] to estimate the absolute scale from a single vehicle-mounted camera. This approach uses no as-

sumptions about the scene, but it works only for cameras mounted on wheeled vehicles and only when the vehicle is turning. This limitation makes it difficult to recover absolute scale for long, linear trajectories.

The last type of methods is more generic and does not use explicit assumptions about the scene. The most common idea is to recover scale using absolute depth maps constructed from single images using deep-learning methods [19, 18]. These approaches are similar to 3D reconstruction using RGB-D sensors, which directly output metric depth maps. The scale is recovered natively by integrating the absolute depth maps into the 3D reconstruction. Recent progress in depth prediction from single images [8, 11, 3] has made it possible to apply these methods to monocular scenes [31].

Another approach, introduced by Frost *et al*. [10], trains the network to predict the absolute distance between camera centers from a pair of images with significant visual overlap. This approach is fairly generic because no explicit assumptions about the scene are made. Intuitively this approach learns how similar image regions are shifted between two frames. Given intrinsic camera parameters, these shifts would be proportional to the camera displacement. The distance predictions are directly included in the bundle adjustment, which improves scale accuracy significantly. The method is applied to images with a relatively low resolution of 240×120 pixels and can be executed in real time. Frost *et al*. [10] also present benefits of integrating the scale estimator into a full monocular SLAM system. From our point of view this integration is straightforward and we are focusing only on improving the scale estimator itself and leaving the integration for future work.

End-to-end SLAM approaches such as [32, 22] provide an alternative to the classical feature-based SLAM approach. These methods also provide full camera poses in absolute scale. We consider these approaches to be overcomplicated (in terms of the number of parameters) for dealing with the scale estimation problem alone. For example, the DeepVO method [32] introduces a fully connected layer with 122M trainable parameters and thus significantly increases computational complexity with strong implications to real-time applications.

The idea to use synthetic data for scale estimation raises the following question: How important is the visual similarity between simulated and real environments? To answer this question, one could apply domain adaptation methods, which help improve the visual realism of synthetic data and train the scale estimator with more realistic synthetic data. Numerous unsupervised image domain adaptation methods have been introduced recently (*e.g*. [36, 17]) and show impressive results with regard to changing the visual similarity between different image domains (*e.g*. synthetic and real). The scale estimation problem takes as input a pair of images and implicitly operates on image changes so the role of photorealism itself is unclear. To evaluate this influence, we test two modern domain adaptation methods ([1, 34]), which are trained in an end-to-end fashion including the target problem (in our case: scale estimation). This approach allows us to optimize the visual image appearance and scale estimation within the same network. Theoretically, this allows us to apply only those image transformations that are relevant for the target problem instead of targeting visually appealing image photorealism for humans. Our implementation of both domain adaptation methods is similar to that proposed in the original papers, see [1, 34] for technical details.

3. Proposed method

We will first introduce the synthetic data collection pipeline, then propose network architectures and finally describe the domain adaptation methods we used.

3.1. Synthetic data collection

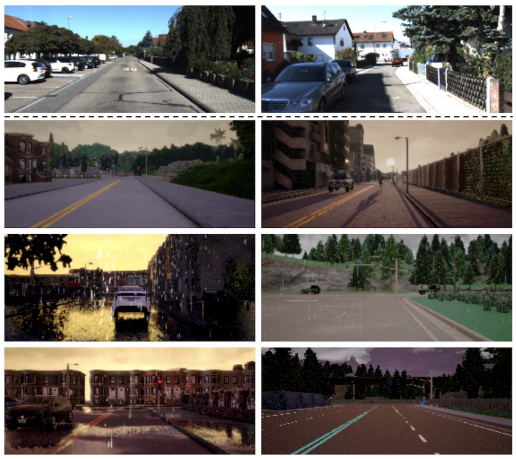

Figure 1. Example of real images from KITTI [12] (top row) and CARLA simulator [7] (bottom three rows) with different daytime and weather conditions.

Using real data (such as the KITTI dataset [12]) for training imposes certain limitations: the variation of weather, illumination and time of day might be limited, diversity of scene appearance (*e.g*. urban, rural) requires significant data collection efforts. In addition, sensor configuration and vehicle movement type determine the variability of observed data in the parameter space. A change of the camera position in the vehicle could lead to the significant systematic errors because certain combinations of rotation and translation are not presented in the training set. One way to overcome these issues is the use of synthetic data collected from autonomous driving simulators like [7, 28, 24]. A simulator allows to attach one or multiple cameras at specific locations on the vehicle, vary weather, time and lighting con-

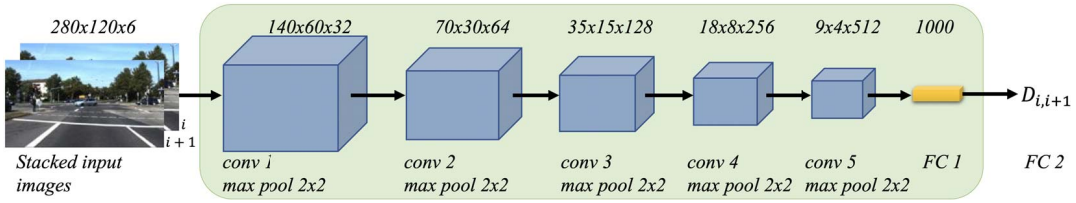

Figure 2. Baseline CNN architecture used in all experiments of this paper.

ditions, and define custom camera intrinsic parameters. An autopilot mode enables automatic driving in the scene combined with training data collection. In our case the collected training data includes color images and ground truth camera trajectories. The synthetic data collection pipeline and a network training procedure form one complete framework which is executed fully automatically. Basically this solves the problem of sensor configuration and, as we show later, allows to reach scale estimation accuracy comparable to accuracy produced by the model trained on real data.

3.2. Network architectures

To start, we selected the method from [10] as a baseline implementation. The key idea is to concatenate a pair of RGB images into one 6-channel image, pass it through three convolutional layers with max-pooling (window size 2x2, stride 2), followed by two fully connected layers. As mentioned above, our goal is to improve the absolute distance estimation accuracy, so our first step is to increase the complexity of the network. We added two more convolutional layers, which improves the accuracy remarkably, see Figure 2 for details. In contrast to [11], we use exponential linear units (ELUs) [5] as activation functions instead of $\tanh(\cdot)$ because it leads to a faster convergence of the training. Dropout layers are used between all convolutional and fully connected layers. A detailed configuration of the network is presented in Table 1. We use the Adam optimizer [20] and the mean squared error as our loss function. For simplicity, the following sections refer to this as a "CNN" architecture.

Layer	Kernel Size	Padding	Number of Filters
conv 1	11x11	5	32
conv 2	9x9	4	64
conv 3	7x7	3	128
conv 4	5x5	2	256
conv 5	3x3	1	512

Table 1. Configuration of CNN layers.

As described in Section 1, there are often significant changes in the estimated distances between consecutive image pairs. This is not surprising because distances are estimated independently for all image pairs, which means that vehicle dymanics are ignored. Instead of adding explicit constraints to possible vehicle movements, we propose that the actual dynamics be learned from data. For this purpose, we use a recurrent neural network (RNN), in particular a many-to-one LSTM [16]. This allows the network to learn how previously observed image pairs influence the current distance estimate. The key concept of our architecture is inspired by [5], which uses an LSTM for action recognition in videos. A similar approach was also used in [32] to predict full 6DOF camera pose. In contrast to the previous approaches, we also evaluate a bidirectional LSTM version, see Figure 3). In this case, we propagate information from N past and N future frames. Our algorithm outputs distance estimates with a small delay, but this limitation is not critical because this step can be done in parallel with other SLAM steps such as feature detection and descriptor computation. In addition, a convolutional part of the network (the green block) is executed only once per image pair, so we need to evaluate only relatively lightweight LSTM layers. Given the low resolution of images in our experiments (280×120 pixels) this delay has a minor impact on computational performance. For comparison, we also evaluated a similar unidirectional LSTM version. We will refer to this as an "LSTM" architecture.

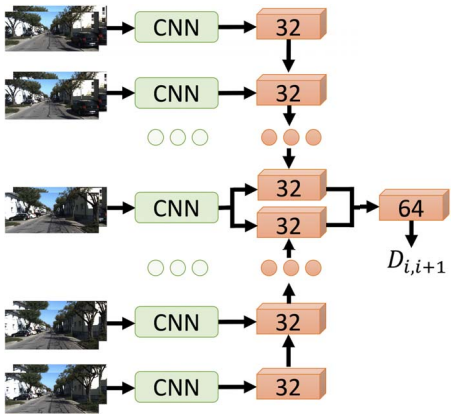

Figure 3. Proposed bidirectional LSTM architecture. The CNN block corresponds to the green block of Figure 2.

Our changes of the baseline architecture [10] lead to the

convergence of the training loss to the values comparable to the accuracy of training data itself. So we conclude that complexity of the network is enough for the considered problem.

We use the same image normalization procedure as in [10], which helps make the trained model invariant to different intrinsic camera parameters. For both synthetic and real data, we use the following image augmentation steps:

- Random image contrast and brightness adjustment
- Random image horizontal flip
- Constrained random image rotation and translation
- Using all consecutive or non-consecutive image pairs as long as the distance between camera poses is not greater than D_{max}
- Duplication of image pairs recorded while vehicle is turning.

3.3. Domain adaptation

Images collected from autonomous driving simulators have quite a different appearance compared with real data, see Figure 1. However, for the task at hand, the network takes a pair of images and, in principle, basically has to look at the geometric scene changes between the images. This raises the question of whether the difference in appearance has a significant impact on the quality of the distance estimation. To answer this question, we reimplemented and applied two state-of-the-art frameworks for unsupervised domain adaptation [1, 34]. Both approaches show promising results for semantic segmentation and single-image depth prediction. We trained these frameworks using a full training set of synthetic data with ground-truth camera locations and unlabeled images from the real training sequences. We tested these approaches out-of-the-box without fine-tuning them, which is beyond the scope of this paper.

4. Experiments

4.1. Experiment overview

This section is organized as follows. First we describe real and synthetic datasets as well as training details for all proposed network architectures. Then, in order to evaluate the proposed methodology, we run the following series of experiments:

- Comparison of absolute scale accuracy of the baseline [10] with our proposed CNN and LSTM architectures trained on real or/and synthetic data
- Evaluation of LSTM length
- Evaluation of synthetic data diversity
- Evaluation of domain adaptation.

These results are followed by a detailed analysis of errors and their distribution for our best model.

4.2. Training details

We observed that a large part of the image is cropped when the original intrinsic camera parameters are used [10]. To use the full image, we take a focal length 250 px and a principal point $(140, 60)$ for the target camera within the image normalization procedure. This change enlarges image resolution slightly from $(240, 120)$ to $(280, 120)$.

Contrast, brightness, rotation and translation are randomized with the same parameters for both images within a pair. Images are augmented and models are trained and evaluated using TensorFlow [2], version 1.12. Random image rotation is applied with an angle in the range $\pm 10^{\circ}$, and random image translation is applied with a shift in the range $\pm 10\%$ of image size. We choose $D_{\text{max}} = 1.7m$ to constrain the distance between nonconsecutive frames within a single training image pair, accounting for the maximum distance between frames in the testing set with an additional margin of 20cm.

We follow the strategy of [10] and use the KITTI outdoor dataset [12] to train and evaluate our approach using real data. Specifically, we use image sequences 01, 03, 04, 05, 06, 07, 09 and 10 for training and sequences 00, 02 and 08 for testing. Ground-truth distances between camera centers are estimated separately for the left and right-hand cameras while taking into account the different camera offsets. For the KITTI dataset [12], the total number of input real training image pairs before random augmentation is approximately 100 K. During evaluation, we use a total of approximately 26 K consecutive testing image pairs both the left and right-hand cameras.

We chose the CARLA simulator [7] to generate synthetic training data out of convenience as it facilitates an autopilot mode. Other simulators ([28, 24] might be applicable instead or in addition to achieve an ever larger variability of the data. The CARLA simulator allows the time of day and weather conditions to be changed—in particular, we can add puddles and vary rain intensity—and provides six different maps. These maps have different visual appearances and road networks. To collect data, we randomly initialize the vehicle's position and the weather conditions. We save all images and their respective camera poses while driving for a short distance (*e.g.* 100 meters) in autopilot mode, then resume driving with a new vehicle position and weather settings. For performance reasons, we change maps only after 100 K image pairs have been collected. We model both the left and right RGB camera using the same extrinsic parameters, i.e., the same offsets to the vehicle's center of rotation, as those provided for the KITTI dataset. The full synthetic training set includes 800 K image pairs from all six virtual maps available in the CARLA simulator [7].

For the CNN architecture, the learning rate of the Adam Optimizer [20] is set to 0.0001 and decreased by a factor of 2 after every 10 K iterations. We add a dropout layer with a

#	Method	Training Data	Seq #00		Seq #02		Seq #08	
			μ	σ	μ	σ	μ	σ
1	Fixed height [14]	KITTI only	0.072	0.252	−0.012	0.160	0.154	0.349
2	Frost *et al*. [10], CNN alone	KITTI only	−0.014	0.177	−0.018	0.203	−0.004	0.165
3	Frost *et al*. (our impl., $240x120$)	KITTI only	−0.009	0.177	0.013	0.180	−0.061	0.152
4	Frost *et al*. (our impl., $280x120$)	KITTI only	−0.015	0.175	−0.010	0.178	−0.057	0.149
5	CNN	KITTI only	0.009	0.107	0.023	0.113	−0.017	0.092
6	CNN	CARLA only	−0.017	0.111	0.023	0.105	−0.029	0.121
7	CNN	CARLA and KITTI	0.036	0.079	0.029	0.084	0.013	0.081
8	CNN, LSTM (B, 19)	KITTI only	0.019	**0.069**	0.033	**0.083**	−0.017	**0.064**
9	CNN, LSTM (B, 19)	CARLA only	−0.004	0.102	−0.003	0.132	−0.049	0.128
10	CNN, LSTM (B, 19)	CARLA and KITTI	0.039	0.076	0.030	0.084	0.002	0.084

Table 2. Comparison of absolute scale estimation results for the baseline method [10] and proposed architectures. Means and standard deviations of the difference between ground truth and predicted values are provided separately for each testing sequence. LSTM (B, 19) stands for bidirectional LSTM with a sequence length 19. Models are trained on the KITTI dataset (KITTI only), synthetic data from the CARLA simulator [7] (CARLA only) or both datasets CARLA and KITTI. All values are in meters.

probability rate of 15%. The batch size is 75 and number of iterations is 100 K.

For the network with an LSTM part, we use slightly different parameters: initial learning rate is 0.00002 with a decay factor of 2 after every 2500 iterations. The batch size is set to 16. The augmentation scheme is as described above. The convolutional part of the network is initialized from the model trained without LSTM layers, which allows for limiting the total number of training iterations to 15 K.

4.3. Evaluation of scale estimators

Table 2 contains the main results of this paper, including a comparison with previous results. For each method we compute standard deviations σ of the difference between ground truth and predicted values. For most of the experiments means are close to zero which indicates an absence of systematic errors. We do not use more advanced metrics (like Absolute Trajectory Error (ATE)) because we estimate only distances between consecutive pairs and not full camera poses.

The first two results come from the literature. We use slightly different settings for dropout and image augmentation than those used in [10], so we evaluate the influence of these changes by reimplementing that architecture. As reported within the 3rd row of Table 2, our results are slightly better. The change of intrinsic parameters has a minor effect on the evaluation results (see rows 3 and 4), which indicates that high accuracy may be reached even with a smaller field of view.

Result 5 corresponds to our findings regarding the proposed CNN network trained using only the KITTI data, for which we observe a significant improvement of σ compared to the results of [10]. Row 6 of Table 2 presents results of the proposed CNN network trained only on synthetic data. An important conclusion comes from comparing the numbers provided in rows 5 and 6, which are quite similar. This proves the ability of the model trained on synthetic data to generalize on a completely unseen set of real images. We also trained the model using a combined dataset from KITTI and CARLA images (row 7). Clearly, these results outperform those trained using a single input modality, which means KITTI and CARLA datasets are complementary.

Our best results are obtained using a bidirectional LSTM with a length of 19, see row 8 in Table 2. This confirms the importance of taking vehicle dynamics into account. However, in the case of CARLA-generated images, our LSTM does not improve accuracy, see rows 9–10. We interpret these results as a special property of synthetic trajectories: they are very smooth and regular, while real trajectories are more noisy and less linear. This gives rise to the assumption that our virtual vehicle dynamics do not generalize well to the real dynamics. Updating the autopilot mode of the CARLA simulator [7] might help solving this problem.

Method	Training data	μ	σ
CNN	CARLA only	−0.009	0.109
CNN, LSTM (B, 19)	CARLA only	0.006	0.118

Table 3. Evaluation of the models trained exclusively on CARLA data on **all available** KITTI sequences (20K image pairs). All σ values are in meters.

In addition we evaluated models trained exclusively on synthetic CARLA data on all KITTI sequences as none of them is used for training. Results are presented in the Table 3 and show that the trained models are able to generalize very well to the full corpus of KITTI data.

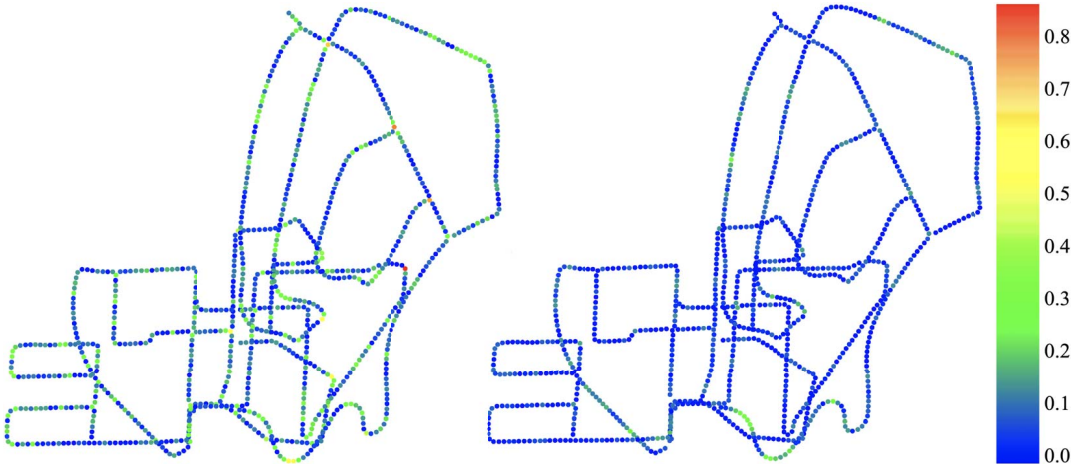

Figure 4. Trajectories colored by the absolute distance estimation errors (KITTI sequences 00, 02, 08). Left: State-of-the-art results of [10] (reimplementation). Right: Best results of this paper. Color bar units are meters.

4.4. Evaluation of LSTM length

LSTM Version	LSTM Length	KITTI Seq #00, #02, #08	
		μ	σ
U	5	0.003	0.087
U	11	0.002	0.085
U	19	0.001	0.088
B	5	0.019	0.084
B	11	0.005	0.077
B	19	0.012	**0.076**

Table 4. Evaluation of different sequence lengths within the unidirectional (U) vs bidirectional (B) LSTM arhitectures. Training and evaluation are performed on the KITTI dataset. All σ values are in meters.

Evaluation of different lengths of LSTM sequences and comparison of unidirectional and bidirectional LSTM versions are shown in Table 4. This experiment shows that the length of sequences plays a minor role, both for unidirectional LSTM (U) and bidirectional LSTM (B).

4.5. Evaluation of synthetic data diversity

Table 5 presents evaluation results illustrating the importance of diversity in the synthetic training data. The numbers show a clear improvement as soon as a virtual scene (or map) with appearances very different from those of the previously considered maps is added to the training set. Town 1 and 2 are quite similar to each other, so adding the data from town 2 does not yield a noticeable improvement. Town 3 contains roads with multiple lanes and thus helps to reduce σ values by about 1 cm. Town 6 contains two highways and improves results even further. All maps available in CARLA are not very large, and the amount of effort involved in adding more maps is small compared to the cost of launching a campaign to collect real data. This is the benefit of using synthetic data for training.

Training CARLA Maps	KITTI Seq #00, #02, #08	
	μ	σ
1	0.004	0.139
1 and 2	-0.025	0.137
1, 2 and 3	-0.008	0.128
1, 2, 3 and 4	-0.012	0.129
1, 2, 3, 4 and 5	-0.011	0.129
1, 2, 3, 4, 5 and 6	-0.007	**0.115**

Table 5. Scale estimation results for the CNN architecture trained using synthetic data with different numbers of virtual maps. Evaluation is performed on the testing part of the KITTI dataset. All σ values are in meters.

4.6. Results of domain adaptation

This subsection addresses the question regarding how photorealism of synthetic data influences absolute scale estimation. Table 6 contains results of distance estimations for three use cases: (i) no domain adaptation, (ii) domain adaptation using approaches T2Net [34] and (iii) CyCADA [1]. Our experiments indicate almost no improvement or even minor degradation of the results if domain adaptation is applied. The difficulties may be related to the problem of changing both input images in a similar way. It seems challenging to constrain the generator of an adversarial network such that realism is equally improved on multiple images, whilst those features important for the task remain preserved. Hence, our insights are aligned with the conclusions drawn in [23], i.e., that photorealism of synthetic

data is less important than diversity.

Method	KITTI Seq #00, #02, #08	
	μ	σ
No domain adaptation	-0.007	**0.115**
T2Net [34]	-0.011	0.117
CyCADA [1]	0.011	0.120

Table 6. Results of domain adaptation for the CNN architecture. The training set consists of a fully annotated synthetic part and unlabeled training images from KITTI. Evaluations are performed on the testing part of the KITTI dataset. All σ values are in meters.

4.7. Error analysis

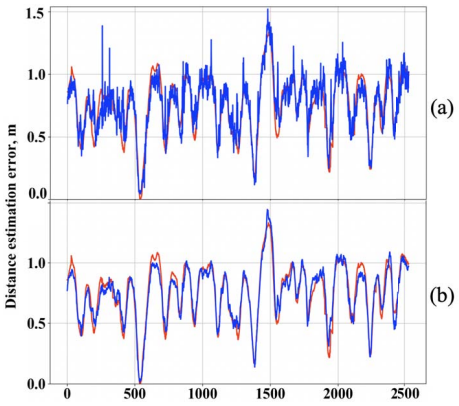

Figure 5. Comparison of recovered distances (blue) and ground truth (red) per frame for the part of the KITTI sequence 00 from the (a) CNN alone; (b) LSTM (B), length 19. Training data: KITTI only. X-axis: frame index.

An overview of all testing trajectories for the baseline method [10] and our best model (LSTM, length 19) is presented in Figure 4. The overview gives a clear understanding of our improvements: (a) absolute scale estimations are more accurate (b) random noise is reduced, and (c) large errors during vehicle turns are nearly eliminated.

Figure 5 compares the proposed LSTM architecture with the proposed CNN architecture. Clearly, the LSTM results are smoother, which confirms the value of adding LSTM layers. However, vehicle rotations remain the most difficult challenge for the proposed method.

Figure 6 shows the cumulated error distribution for all testing sequences. From the histogram, we observe a minor tendency of overestimating the "true" distances.

Figure 7 provides example images where relatively large errors are still observed (LSTM (B), length 19). The areas with large amounts of vegetation are the most difficult ones.

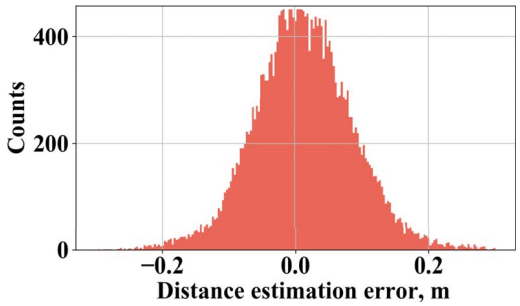

Figure 6. Histogram of distance estimation errors for LSTM (B, length 19) on all testing KITTI sequences. Training data: KITTI only.

Figure 7. Images with largest errors for LSTM (B, length 19). Training data: KITTI only.

5. Conclusion

This paper addresses the problem of scale estimation in monocular SLAM by estimating the distance between camera centers of consecutive image frames. These estimates would improve the overall performance of classical (not deep) SLAM systems and cast the entire 3D reconstruction from a monocular camera in metric values. The proposed solution estimates scale for each pair independently (or with soft-constrained LSTM network), which makes it insensitive to long-term drift effects. Our work introduced several network architectures, which lead to an improvement of scale estimation accuracy over the state of the art. With respect to the baseline method, our results show significant improvements of the estimates for camera rotations. In addition, we exploit the possibility to train the neural network only with synthetic data derived from a computer graphics simulator. Our experiments indicate that, using only synthetic training inputs, we can achieve similar scale estimation accuracy as that obtained from real data. This provides a practical solution to the sensor reconfiguration problem. Our experiments with unsupervised domain adaptation also demonstrate that differences in visual appearance (photorealism) between simulated and real data does not affect scale estimation results. The proposed methods operate with low-resolution images (0.03 MP), which makes them practical for real-time SLAM applications with a monocular camera.

6. Acknowledgment

This work has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731993 and No 688652.

References

[1] Cycada: Cycle consistent adversarial domain adaptation. In *International Conference on Machine Learning (ICML)*, pages 1994–2003, 2018.

[2] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015.

[3] A. Atapour-Abarghouei and T.P. Breckon. Real-time monocular depth estimation using synthetic data with domain adaptation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, 2018.

[4] R. O. Castle, D. J. Gawley, G. Klein, and D. W. Murray. Towards simultaneous recognition, localization and mapping for hand-held and wearable cameras. In *"International Conference on Robotics and Automation (ICRA)"*, pages 4102–4107, 2007.

[5] Jeff Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2625–2634, 2015.

[6] A. Dosovitskiy, P. Fischer, E. Ilg, P. Häusser, C. Hazırbaş, V. Golkov, P. v.d. Smagt, D. Cremers, and T. Brox. Flownet: Learning optical flow with convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 2758–2766, 2015.

[7] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16, 2017.

[8] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. In *Advances in Neural Information Processing Systems*, pages 2366–2374. 2014.

[9] Duncan P. Frost, Olaf Kähler, and David W. Murray. Object-aware bundle adjustment for correcting monocular scale drift. In *"International Conference on Robotics and Automation (ICRA)"*, pages 4770–4776, 2016.

[10] Duncan P. Frost, David W. Murray, and Victor Adrian Prisacariu. Using learning of speed to stabilize scale in monocular localization and mapping. In *International Conference on 3D Vision (3DV)*, pages 527–536, 2017.

[11] Huan Fu, Mingming Gong, Chaohui Wang, Kayhan Batmanghelich, and Dacheng Tao. Deep ordinal regression network for monocular depth estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2002–2011, 2018.

[12] Andreas Geiger. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3354–3361, 2012.

[13] Andreas Geiger, Martin Lauer, Christian Wojek, Christoph Stiller, and Raquel Urtasun. 3d traffic scene understanding from movable platforms. *IEEE transactions on pattern analysis and machine intelligence*, 36(5):1012–1025, 2013.

[14] Andreas Geiger, Julius Ziegler, and Christoph Stiller. Stereoscan: Dense 3d reconstruction in real-time. In *IEEE Intelligent Vehicles Symposium (IV)*, pages 963–968, 2011.

[15] Johannes Gräter, Tobias Schwarze, and Martin Lauer. Robust scale estimation for monocular visual odometry using structure from motion and vanishing points. In *IEEE Intelligent Vehicles Symposium (IV)*, pages 475–480, 2015.

[16] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[17] Xun Huang, Ming-Yu Liu, Serge Belongie, and Jan Kautz. Multimodal unsupervised image-to-image translation. In *European Conference on Computer Vision (ECCV)*, pages 172–189, 2018.

[18] Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison, and Andrew Fitzgibbon. Kinectfusion: Real-time 3d reconstruction and interaction using a moving depth camera. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*, pages 559–568, 2011.

[19] C. Kerl, J. Sturm, and D. Cremers. Robust odometry estimation for rgb-d cameras. In *"International Conference on Robotics and Automation (ICRA)"*, pages 3748–3754, 2013.

[20] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[21] Bernd Kitt, Andreas Geiger, and Henning Lategahn. Visual odometry based on stereo image sequences with ransac-based outlier rejection scheme. In *IEEE Intelligent Vehicles Symposium (IV)*, pages 486–492, 2010.

[22] Qing Li, Jiasong Zhu, Rui Cao, Ke Sun, Jonathan M Garibaldi, Qingquan Li, Bozhi Liu, and Guoping Qiu. Relative geometry-aware siamese neural network for 6dof camera relocalization. *arXiv preprint arXiv:1901.01049*, 2019.

[23] Nikolaus Mayer, Eddy Ilg, Philipp Fischer, Caner Hazirbas, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox. What makes good synthetic training data for learning disparity and optical flow estimation? *International Journal of Computer Vision*, 126(9):942–960, 2018.

[24] Craig Quiter and Maik Ernst. deepdrive/deepdrive: 2.0, 2018.

[25] Dan Rosenholm and Kenner! Torlegard. Three-dimensional absolute orientation of stereo models using digital elevation models. *Photogrammetric Engineering and Remote Sensing*, 54(10):4102–4107, 1988.

[26] Davide Scaramuzza, Friedrich Fraundorfer, Marc Pollefeys, and Roland Siegwart. Absolute scale in structure from motion from a single vehicle mounted camera by exploiting

nonholonomic constraints. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 1413–1419, 2009.

[27] Davide Scaramuzza, Friedrich Fraundorfer, Marc Pollefeys, and Roland Siegwart. Absolute scale in structure from motion from a single vehicle mounted camera by exploiting nonholonomic constraints. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 1413 – 1419, 2009.

[28] Shital Shah, Debadeepta Dey, Chris Lovett, and Ashish Kapoor. Airsim: High-fidelity visual and physical simulation for autonomous vehicles. In *Field and Service Robotics*, pages 621–635, 2017.

[29] Shiyu Song and Manmohan Chandraker. Robust scale estimation in real-time monocular SFM for autonomous driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1566–1573, 2014.

[30] Edgar Sucar and Jean-Bernard Hayet. Bayesian scale estimation for monocular SLAM based on generic object detection for correcting scale drift. In *"International Conference on Robotics and Automation (ICRA)"*, pages 1–7, 2018.

[31] K. Tateno, F. Tombari, I. Laina, and N. Navab. Cnn-slam: Real-time dense monocular slam with learned depth prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6243–6252, 2017.

[32] S. Wang, R. Clark, H. Wen, and N. Trigoni. Deepvo: Towards end-to-end visual odometry with deep recurrent convolutional neural networks. pages 2043–2050, 2017.

[33] N. Yang, R. Wang, J. Stueckler, and D. Cremers. Deep virtual stereo odometry: Leveraging deep depth prediction for monocular direct sparse odometry. In *Proceedings of the IEEE European Conference on Computer Vision (ECCV)*, pages 817–833, 2018.

[34] Chuanxia Zheng, Tat-Jen Cham, and Jianfei Cai. T2net: Synthetic-to-realistic translation for solving single-image depth estimation tasks. In *Proceedings of the IEEE European Conference on Computer Vision (ECCV)*, pages 767–783, 2018.

[35] Dingfu Zhou, Yuchao Dai, and Hongdong Li. Ground plane based absolute scale estimation for monocular visual odometry. *arXiv preprint arXiv:1903.00912*, 2019.

[36] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2223–2232, 2017.