Calibrating Depth Sensors with a Genetic Algorithm

Jonas Haeling[1,2], Marc Necker[1], Andreas Schilling[2]
[1]Daimler AG R&D, Sindelfingen, Germany;
[2]University of Tübingen, Tübingen, Germany;

ABSTRACT

In this report, we deal with the optimization of the transformation estimate between the coordinate systems of depth sensors, i.e. sensors that produce 3D measurements. For that, we present a novel method using a genetic algorithm to refine the six degrees of freedom (6 DoF) transformation via three rotational and three translational offsets. First, we demonstrate the necessity for an accurate depth sensor calibration using a depth error model of stereo cameras. The fusion of stereo disparity assumes a Gaussian disparity error distribution, which we examine with different stereo matching algorithms on the widely-used KITTI visual odometry dataset. Our analysis shows that the existing calibration is not adequate for accurate disparity fusion. As a consequence, we employ our genetic algorithm on this particular dataset, which results in a greatly improved calibration between the mounted stereo camera and the Lidar. Thus, stereo disparity estimates show improved results in quantitative evaluations.

1. INTRODUCTION

An accurate calibration between depth sensors is important to ensure that data between sensors can be merged or evaluated jointly. The evaluation of a cheap sensor is an additional important task, e.g. comparing a stereo camera against an accurate but prohibitively expensive sensor like a Lidar or a structured light setup. Even though our proposed technique is general enough to handle any comparisons between depth maps, we are particularly interested in Lidar-to-stereo calibration in this report. This calibration is needed in vehicle setups like the popular KITTI visual odometry dataset [1] in which the Lidar delivers accurate but sparse measurements. The stereo camera produces dense depth maps with additional image intensity or color information. The stereo depth error increases quadratically with the depth itself (see Subsection 2.1), depending on the intrinsics of the stereo camera and the particular stereo algorithm. To correctly evaluate the achieved accuracy of the dense stereo matching against an accurate Lidar, an exact transformation estimate, i.e. a calibration, between the two coordinate systems is required, which is the main focus of this report.

Genetic algorithms are optimization techniques, which are bio-inspired by evolution. The term genetic algorithm has a diffuse meaning [2], but the general idea is to use an iterative process with evolutionary elements like natural selection, crossover and mutation to find solutions to hard problems, e.g. the traveling salesman problem. The solution may not be optimal and might only be a local optimum in a niche, but a direct solution for some problems is not always feasible.

A generation of individuals constitutes the current population in a genetic algorithm. An individual could simply be an encoded solution to the problem, e.g. a set of real numbers or indices. The genetic algorithm iteratively produces new generations using the immediate previous one to optimize solutions. To mimic natural selection, one has to model the fitness of an individual, which corresponds to a measure of how good the solution is. The fittest individuals have a higher chance to reproduce. For the reproduction, there is a crossover phase in which new offspring for the next generation are created by the recombination of genes. The parents are sampled from the population with weights according to the fitness. Random mutations occur during reproduction, which adds variance to the gene pool, thus potentially better or worse fitness values. That concludes one iteration. This process leads to an optimization of the species to the specific environment over time, i.e. an optimized solution to the problem.

Our genetic algorithm optimizes the six degrees of freedom between the coordinate systems of both depth sensors. We assume an existing approximate estimate for the transformation and refine it with a rotational and translational offset. An individual of a population has six offsets, which can be altered. The genetic algorithm iteratively recombines and mutates new offspring, which is then evaluated over several frames to compute an

individual fitness using the histogram of the resulting error distribution. We demonstrate our algorithm on a real dataset and show greatly improved results concerning the disparity error distributions. This new "recalibration" may also improve quantitative results for stereo reconstruction approaches, especially those that rely on disparity fusions.

A similar algorithm to our proposed solution is the widely researched iterative closest point (ICP) algorithm to register two 3D point clouds iteratively [3]. Here, we match the points in image space like some ICP variants over multiple frames and explore and refine the existing solution via a genetic algorithm. Even though we recalibrate a Lidar to a stereo camera, it would still be possible to calibrate two separate stereo cameras. This will result in higher run times due to the increased point cloud density.

Section 2 describes the motivation behind calibrating depth sensors using a stereo camera depth error model, including an examination of the disparity error on a real dataset. Section 3 explains the basic sequence of our genetic algorithm and the GPU implementation. We test our algorithm and show the corresponding results in Section 4. Finally, we give a summary and an outlook for further research in Section 5.

2. MOTIVATION

2.1 Stereo Disparity Fusion

Kalman filters have been widely used in stereo vision for various aspects including the fusion of disparities between frames [4–9]. For most Kalman filters used in the stereo vision context, one key assumption is Gaussian stereo matching noise, i.e. Gaussian disparity errors. If this assumption is not given, e.g. the distribution is actually skewed or multimodal, then formally correct fusion algorithms might help measurements to be more precise, but the accuracy of the measurement does not improve because of an inherent bias. Popular stereo matching benchmarks like the "Middlebury Stereo Evaluation" [10] or "Stereo Evaluation 2015" from the KITTI Vision Benchmark Suite [11] do not report the error distribution directly, unfortunately. Assuming that a certain stereo matching algorithm exhibits Gaussian noise as disparity errors, an inexact transformation to ground truth depth sensor measurements would yield an artificial error overestimation in a quantitative error evaluation. Thus, refining this calibration would improve quantitative results without changing the actual stereo depth estimation process.

Geyer et al. [12] proposed to model the depth error e_z as follows:

$$e_z = \frac{Bf}{d} - \frac{Bf}{d + e_d} = \frac{z^2 e_d}{Bf + z e_d} \tag{1}$$

B is the stereo baseline, f the focal length, d the disparity and e_d the disparity or matching error. Thus, to decrease the depth error e_z we have to either increase B or f, decrease e_d or simply move closer to the object for a smaller z. Gallup [13] varied the baseline and focal length by choosing the input cameras for their plane sweep algorithm or downsampling the images to achieve a desired fixed depth accuracy. Geyer et al. [12] increased the stereo baseline and fused measurements from a moving airborne vehicle observing the ground using overlapping images of a mono-camera to reduce the error variance of range measurements.

Alternatively, we can fuse 3D points temporally between frames with a static stereo baseline, which enables us to decrease the stereo matching error e_d. Reformulating Equation 1, we can approximate the needed disparity error e_d to achieve a certain desired and fixed depth error e_z:

$$e_d = \frac{e_z B f}{z(z - e_z)} \tag{2}$$

e_z is not distributed as a Gaussian because the disparities are transformed non-linearly from image space to 3D camera space (cf. Sibley et al. [14]). Therefore we can only approximately handle e_z as Gaussian.

Assuming Gaussian noise for the disparity error (with e_d being the standard deviation), the fusion of two uncorrelated observations of the same depth will result in a lower variance of the error. If the error is perfectly correlated, e.g. when running a deterministic stereo matching with unchanged parameters again on the same exact images, then the fusion will not decrease the error.

Table 1: Examples for the needed uncorrelated observations (n) at a certain fixed depth to achieve a desired accuracy $e_{z'} = 0.10$ m with $e_d = 0.5$ px, $B = 0.54$ m and $f = 700$ px.

Depth	n	Fused e_z	Unfused e_z
8.7 m	1	0.10 m	0.10 m
10 m	1.7	0.10 m	0.13 m
20 m	27.7	0.10 m	0.53 m
30 m	140.8	0.10 m	1.19 m
40 m	445.7	0.10 m	2.12 m
50 m	1089.2	0.10 m	3.32 m

Geyer et al. [12] found in simulated experiments that the error of stereo measurements was correlated up to 60%, which suggests that we underestimate the error with disparity fusion in practice. For the time-being, we will not address this issue. Assuming uncorrelated errors in the stereo matching, the fused disparity estimate $e_{d'}$ is computed from the normalized product of two 1D Gaussian probability density functions with standard deviations e_{d_1} and e_{d_2} (see Barfoot [15]):

$$\frac{1}{e_{d'}^2} = \frac{1}{e_{d_1}^2} + \frac{1}{e_{d_2}^2} \tag{3}$$

This is essentially the summation of both information terms to a new information term. To generalize, the fusion of n uncorrelated observations with the same disparity variance results in a new error $e_{d'}$, which allows us to compute the needed number of fusions n for the disparity error with a fixed depth error:

$$\begin{aligned} &\frac{1}{e_{d'}^2} = n \cdot \frac{1}{e_d^2} \\ &\Rightarrow n = \frac{e_d^2}{e_{d'}^2} = \frac{e_d^2}{\left(\frac{e_{z'} B f}{z(z-e_{z'})}\right)^2} = \frac{e_d^2 z^4 + e_d^2 e_{z'}^2 z^2 - 2 e_d^2 e_{z'} z^3}{e_{z'}^2 B^2 f^2} = \mathcal{O}(z^4) \end{aligned} \tag{4}$$

We can see that the number of frames n is bound by $\mathcal{O}(z^4)$, which means to achieve the desired and fixed depth error $e_{z'}$ from measurements of a particular depth, we need quartically more uncorrelated measurements of the same depth (assuming $e_{d'} < e_d$).

One concrete fusion example could be a house facade seen from far away by a sidewards looking camera in a vehicle, so that we always measure the same depth. Table 1 gives examples for the needed independent observations to get a desired accuracy. The intrinsic parameters for this were chosen similar to the actual parameters of the KITTI visual odometry dataset [1]. Without fusion, the desired accuracy is achieved by points below a distance of ca. 8.7 m. We also compare the fused vs. unfused depth error in e_z, which is only an approximation since the resulting distribution is skewed for far depth values (cf. Sibley et al. [7]).

To summarize, temporal disparity fusion is able to improve reconstruction results if the Gaussian noise assumption for disparity errors is valid.

2.2 Stereo Matching Error as Gaussian Noise Assumption in Literature

In the literature, Sibley et al. [14] used the assumption that the disparity error is distributed as a Gaussian and provided evidence using corner features and a synthetic checkerboard scene. Similarly, Matthies and Grandjean [16] found the disparity distribution to be Gaussian-like when computing the disparity by minimizing the sum of squared difference (SSD) in a search window. Sebe and Lew [17] found that in their case that a better fit is a Cauchy function when using the sum of absolute differences or SSD with real world ground truth data. Wedel and Cremers [8] on the other hand propose to approximate the disparity distribution with a Laplacian distribution.

Nevertheless, this noise attribute is highly dependent on the actual stereo matching algorithm itself and the transformation to the ground truth, i.e. the calibration between the depth sensors. On the one hand, the fusion may still be unbiased when the true distribution is unimodal and symmetrical, like the Cauchy or Laplacian distribution. On the other hand, the variance estimate will be off and hence the weighting of the fusion, which may introduce errors or limit the accuracy increase. In practice, the true variance is hard or impossible to estimate anyway due to the aforementioned unknown error correlations.

2.3 Verification of Distribution

To verify whether the stereo matching error distribution is indeed Gaussian, we test how the disparity error of actual stereo matching results is distributed. We tested the disparity distribution of the publicly available and widely used stereo matcher ELAS ("Efficient Large-scale Stereo Matching" [18]) on the KITTI odometry training dataset. This dataset is widely used to evaluate 3D reconstructions from multi-view stereo quantitatively (see [19–26]) as it provides synchronized laser-scanned data as ground-truth and ground-truth poses. It is a de facto benchmark for stereo scene reconstructions from a moving vehicle. For disparity error distributions, we compute the disparity on the rectified color camera images via stereo matching, then project the 3D laser points onto the rectified left color camera image and compare the laser projections to the stereo matching disparities. We only evaluate pixels for which both stereo and laser measurements are present. We do this for every frame and every sequence except sequence 03, as there is no raw calibration data available and the measurements were made on a different day than all the others. Note that we can omit the ground-truth poses for this evaluation as the transformation between stereo camera and Lidar should remain constant. The error is computed via $d_{Stereo} - d_{Laser}$ and in Figure 1 all disparity error histograms of ELAS are plotted. The sequences 00 to 02 were recorded on a different day than 04 to 10, which is why the calibration also differs, as well as the resulting shape of the distribution. For sequences 04 to 10 we clearly see a local maximum at ca. +0.6 px and for sequences 00 to 02 closer to ca. +1.4 px. However, testing ELAS on the KITTI 2015 stereo dataset [11], we see very different results, as shown in Figure 2. Unimodal distributions with only a small bias are visible. Sources of this bias may be camera calibration errors, biased stereo matching, relative small ground truth transformation errors or Lidar uncertainty.

Figure 3 shows one sample frame in sequence 08 and 01, consisting of RGB image, depth image and disparity comparisons. The disparity error images were enhanced to make the visual comparison easier. Note that the vertical field of view of the velodyne laser is inherently tilted down, resulting in a cutoff of higher stereo matching

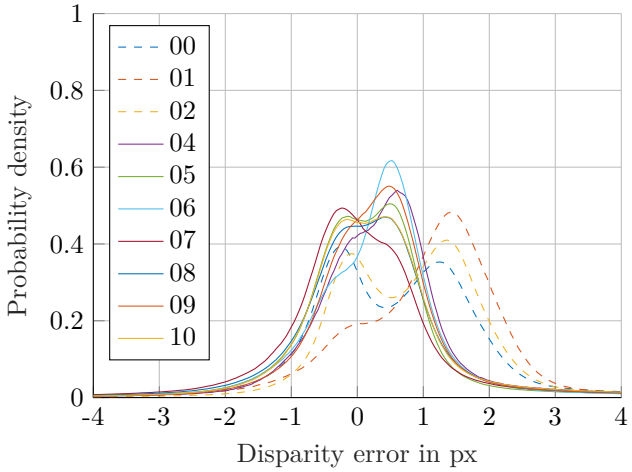


Figure 1: The disparity error distribution of ELAS compared to velodyne laser measurements on all KITTI odometry training datasets (except 03) with the original velodyne transformation. The different line styles indicate different calibrations.

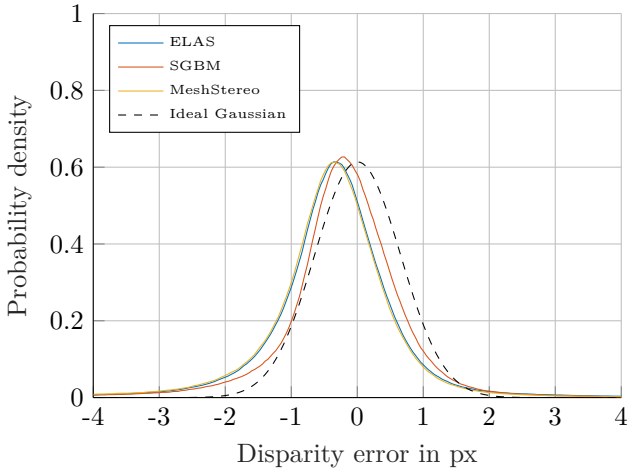
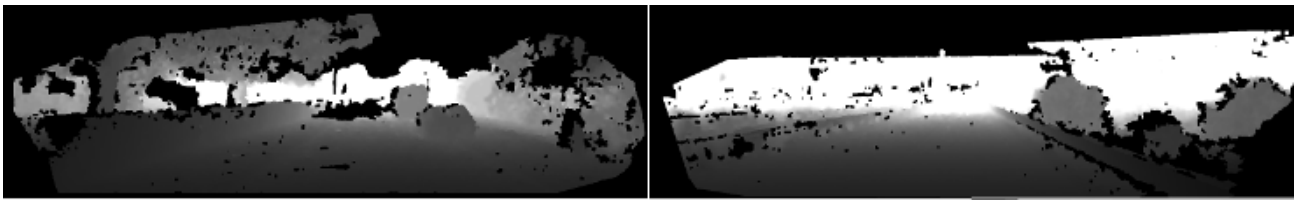
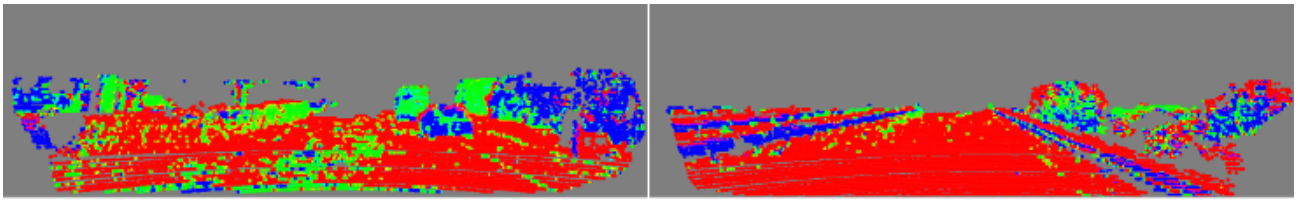


Figure 2: Disparity error distribution of ELAS, SGBM and MeshStereo of 200 ground truth frames of the multi-view sequences of the SceneFlow 2015 KITTI training dataset [11]. A normal distribution with $\mu = 0$ and $\sigma = 0.65$ is drawn for comparison.

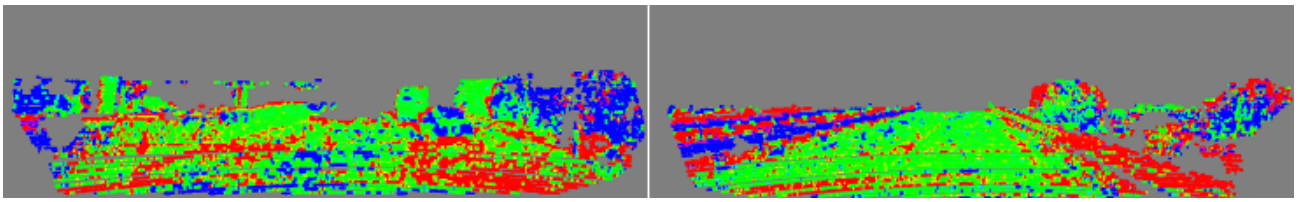
(a) Left color camera RGB image.



(b) Left color camera ELAS depth image.



(c) Disparity comparison of stereo to Lidar measurements with original calibration.



(d) Disparity comparison after applying our new calibration.

Figure 3: Comparison of disparities between ELAS and the ground-truth on frame 08-430 (left) and frame 01-1100 (right). Red corresponds to positive disparity errors and blue to negative ones. Green represents good matches (< 0.5 px) between the laser and stereo measurement.

points. We see a strong disparity bias, represented by red pixels on the ground, denoting too high disparity estimates of the stereo.

We test OpenCV's StereoSGBM algorithm (SGBM) [27] as well, which is an altered implementation of semi-global matching (SGM) [28]. Additionally, we employ the approach "MeshStereo" by Zhang et al. [29]. We use three different stereo algorithm to show that they produce similar distributions, i.e. it is not the fault of a single stereo matcher. After testing ELAS on synthetic images and the KITTI 2015 stereo dataset, we did not expect such a bias with the ground surface on the odometry dataset. We suspected that the velodyne to camera transformation was not sufficiently calibrated, which was confirmed in early tests with small transformation changes (also cf. Guindet et al. [30]). Some of those changes made the bimodal distribution more unimodal and moved the center of mass of the error probability distribution towards zero.

Fusing measurements with the original bimodal error distributions would still increase the precision, but not the accuracy. We would still be off the mark due to the bias of the distributions, i.e. not being centered around zero error. A better calibration remedies this problem, so a recalibration of the Lidar to camera transformation is required for improved quantitative results.

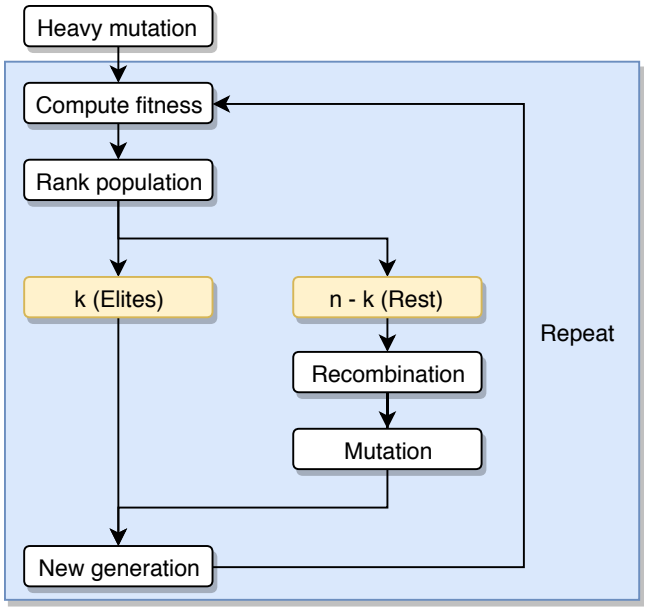


Figure 4: Schematic sequence of the genetic algorithm.

3. GENETIC ALGORITHM

Here, we are interested only in the refinement of a depth sensor to depth sensor transformation, i.e. we already have a rough estimate and want to hone in on an improved solution via evolution. We do this by adding three translational offsets and three Euler angles as rotational offsets to the existing calibration.

To find suitable offsets, we employ a standard genetic algorithm, which explores and optimizes the 6 DoF solution space. Per generation, we have a population of $n = 100$ individuals. An overview of the general sequence of the algorithm is shown in Figure 4. We heavily mutate the first generation before the first iteration. We compute the fitness of the population and rank the individuals according to the fitness score. Here, we actually use a fitness cost instead of score, so we minimize the cost instead of maximizing the score. For the fitness cost, we calculate the ratio of pixels with a resulting absolute disparity error of below 0.5 px to the total amount of valid pixels. This corresponds to the green pixels seen in Figure 3 and empirically resulted in satisfactory results fast. It is also one of the metrics used in the Middlebury stereo evaluation [10]. We also add one additional term to the fitness cost, which is almost identical but uses a threshold of 0.2 px to hone in on excellent stereo matches. These fitness function terms reward unbiased error distributions with low variance. We also strictly punish outlier distributions, which do not have enough projected Lidar points. After the ranking, we use elitism with $k = 3$ to transfer the three best individuals to the next generation directly. For the rest, we use rank selection to select parents and produce the offspring by randomly choosing the parameter value from one parent, which corresponds to a crossover rate of 1. The offspring then finally mutates, in which a big and a tiny mutation can occur. A big mutation consists of choosing a random value from an interval which spans the given search space (”wiggle room”), e.g. 1° for the angles and 5 cm for the translational components. A tiny mutation is a very small variation around the current value, given a downscaled random value from the aforementioned interval. These mutations ensure that the solution space is explored but also that local minima are found faster. We use a mutation rate of 0.1 for both kinds of mutation. Then we begin computing the fitness again, which marks a completed generation iteration.

3.1 GPU Implementation of the Fitness Evaluation

The fitness evaluation of a whole generation is the bottleneck of the evolutionary algorithm. All Lidar points over multiple frames have to be projected with the new transformation and compared to the corresponding depth image for each individual. On a single-core CPU implementation, this has to be done sequentially: For each individual, for each frame and for each velodyne point in the frame. Almost all of the input data used in the

evolution remains static, except the new calibration T_{velo}^{cam}. For these reasons, a simple GPU implementation of the fitness evaluation was developed with CUDA [31].

Almost all the input data remain static, so we only have to upload all precomputed depth images, Lidar points and the projection matrix once to the GPU. Since the Lidar points have an irregular size each frame, we take the maximum Lidar point count of one frame as the array length. We pad the rest with points that will not be projected onto the camera. The GPU kernel takes as input an array of n calibrations and outputs n average fitnesses over all frames. We launch a kernel for each individual and each frame, which means there is potential to break down the work load even further by partitioning the Lidar point cloud per frame. We allocate n floats for the fitness of each individual beforehand. For that, we employ the CUDA function atomicAdd() on the specific individual fitness index to add the computed fitness per frame, which prevents race conditions during the write operation. After the kernels have finished, we compute the individual mean of the fitness values. Then these results are downloaded to the CPU, which logs the results and starts the next generation iteration.

4. RESULTS ON KITTI VISUAL ODOMETRY DATASET

4.1 Main Results

We are looking for a new calibration between the laser and the cameras. This calibration does not need to be optimal, but should show improved error distribution results. The transformation chain for transforming a homogeneous 4D velodyne point $X = [x, y, z, 1]^T$ into a camera image point $p = (u, v, 1)^T$ in the rectified left color camera coordinate system (camera 2) is the following (see Geiger et al. [32]):

$$p = P_{rect}^{(2)} R_{rect}^{(0)} T_{velo}^{cam} X \tag{5}$$

with

$$T_{velo}^{cam} = \begin{pmatrix} R_{velo}^{cam} & t_{velo}^{cam} \\ 0 & 1 \end{pmatrix} \tag{6}$$

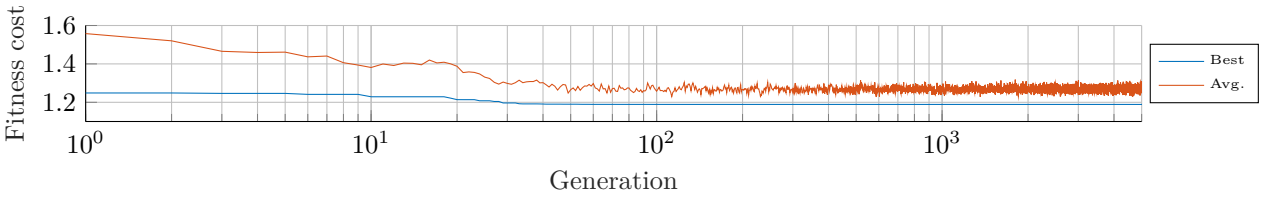
$P_{rect}^{(2)}$ is the 3x4 projection matrix for the left rectified color camera ("camera 2") and $R_{rect}^{(0)}$ the rotation from the left gray camera ("camera 0") to the rectified camera coordinate system of all rectified cameras. Note that $P_{rect}^{(2)}$ incorporates a translational component from camera 0. T_{velo}^{cam} denotes the 4x4 transformation from the velodyne laser coordinate system to camera 0, which incorporates a 3x3 rotation matrix R_{velo}^{cam} and a 3x1 translation vector t_{velo}^{cam}. The intrinsic stereo camera calibration was done separately from the velodyne to camera calibration (see [1]), which is why we only try to optimize T_{velo}^{cam}. Since we assume that the original calibration is already close to the truth, we try to find an offset to a new calibration T'^{cam}_{velo} consisting of R'^{cam}_{velo} and t'^{cam}_{velo}. Thus, we are looking for three Euler angle offsets and three translation offsets to improve the calibration. The Euler angles are global rotations around X, Y and Z in this order (pitch-yaw-roll), which corresponds to the following multiplication sequence of local 3D rotation matrices:

$R_X R_Y R_Z = R$

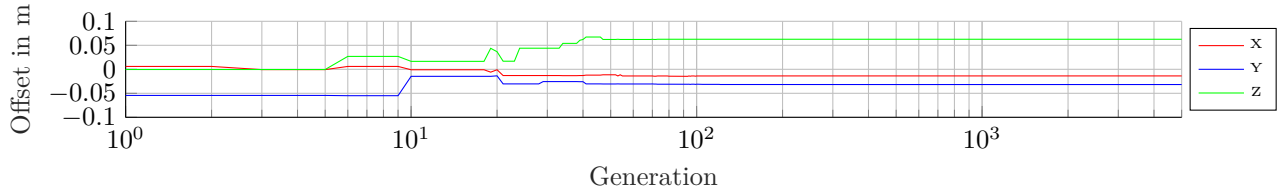
One observation is that the sequences 00 to 02 were recorded on the same day and the original calibration did not change. Similarly, sequences 04 to 10 were also recorded on the same day. This means that we only have to recalibrate twice, finding one calibration for 00-02 and one for 04-10. The raw calibration data for the sequence 03 is not available.

As input to our genetic algorithm, we use every 50th KITTI frame, corresponding to a frame every five seconds to offer variation in the scenes and still maintain acceptable run times. We employed a wiggle room of $\pm 2.5°$ and $\pm$ 7.5 cm, which delivered good results quickly.

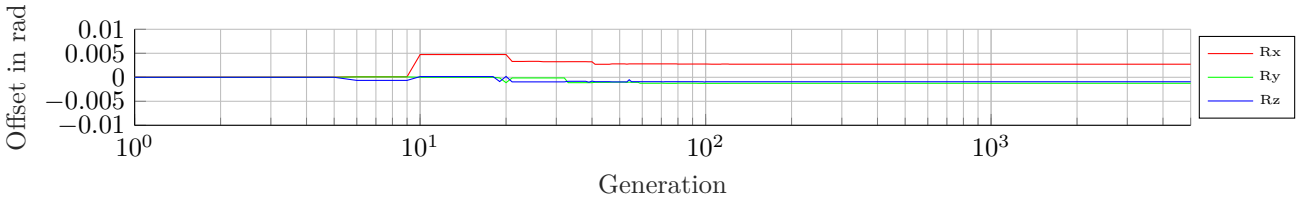
The evolution process for sequence 00 is shown in Figure 5 and for sequence 08 in Figure 6. The horizontal axis depicts the generations on a logarithmic scale. One can see that after ca. 100 generations the fitness improvements are largely marginal. Improvements still occur, but they get less probable and significant. The resulting offsets after 5000 generations represent our final recalibration for the sequences 00-02 and 04-10. One observation is that the computed calibration offset for the translation is up multiple centimeters, which means that either we may be overfitting and evolving to an unlikely offset or that the original calibration is less accurate than expected.



(a) Fitness cost of best individual and average value of the generation.

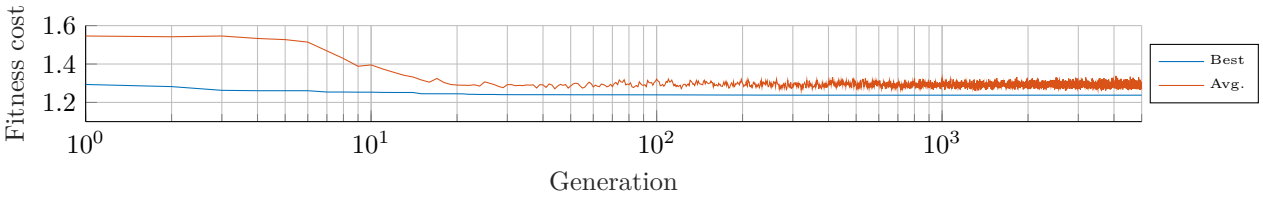


(b) Translational offset of best individual.

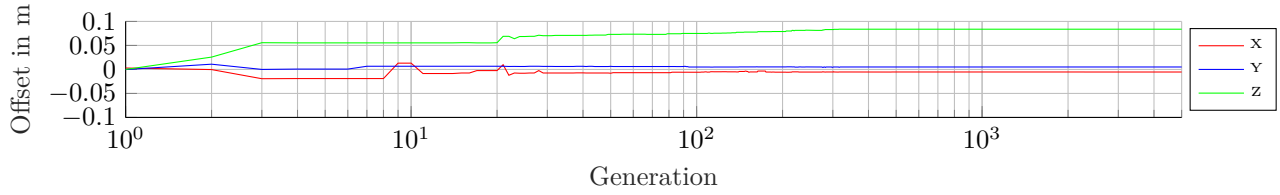


(c) Rotational offset of best individual.

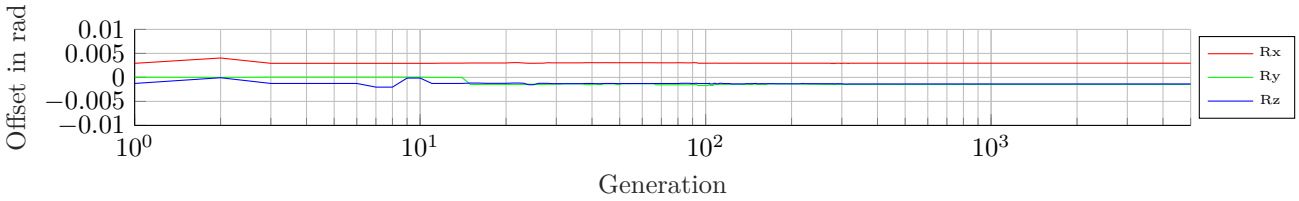
Figure 5: Evolution results over 5000 generations for sequence 00. Top to bottom: fitness scores, translational and rotational offset.



(a) Fitness cost of best individual and average value of the generation.



(b) Translational offset of best individual.



(c) Rotational offset of best individual.

Figure 6: Evolution results over 5000 generations for sequence 08. Top to bottom: fitness scores, translational and rotational offset. The average fitness values appear fuzzier in the end due the logarithmic scale.

Table 2: Final calibration offset parameters for the KITTI VO dataset.

Calibration	X	Y	Z	Rx	Ry	Rz
00 - 02	-0.0150154	-0.030895	0.0644574	0.00277647	-0.00124684	-0.00100826
04 - 10	-0.00486741	0.00384346	0.0912176	0.00283365	-0.00150622	-0.00124315

Table 3: Average run time per generation of both algorithm implementations on sequence 08.

CPU	GPU	GPU speedup
48.677 s	0.092 s	ca. 529 x

In Figure 7, the two final calibration offsets for sequences 00-02 and 04-10 are used to compute the new disparity error distribution for ELAS, SGBM and MeshStereo. The new resulting distributions are clearly more unimodal, sharper and show less bias than the original distributions. Again, we did not alter the stereo measurements, merely the relative transformation between the ground truth and the stereo for the evaluation. Although we only use ELAS for the evolution of these offsets, we still improve the results of SGBM and MeshStereo. This is evidence that we are indeed not overfitting, but that the offsets are plausible since they improve the error distributions of other stereo algorithms in a very similar way. In Table 2 we showcase our final optimized offsets for reference.

Figure 3 shows two KITTI VO frames without and with the new calibration. The original calibration shows a clear bias on the ground surface, while more vertical structures like the hedge and vehicles show less. Our recalibration result removes this bias and keeps the rest of the scene intact or improves it. The aforementioned ground bias is largely eliminated, which is also apparent from the new distributions shown in Figure 7.

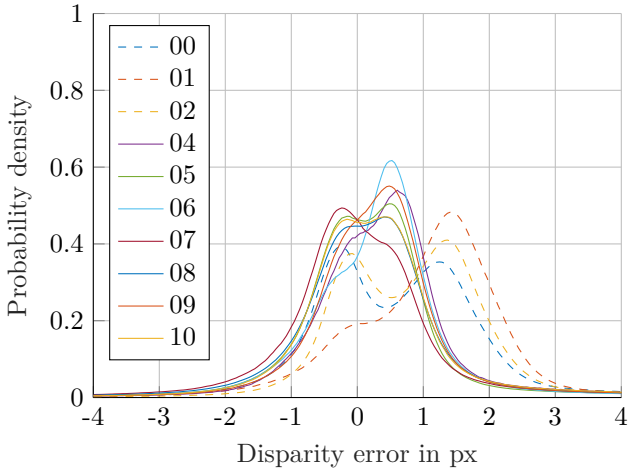
Table 3 compares the CPU single core implementation without SIMD instructions against the implementation with our CUDA fitness evaluation with a frame skip of 50. We use an Intel i7-6700T 2.8 GHz CPU with a nVidia GTX 1080 GPU. Our GPU implementation accomplishes a speedup of 529 x with ca. 11 Hz, which makes it feasible for online recalibration using a fixed generation limit.

4.2 Comparison of Variations and Alternative Approaches

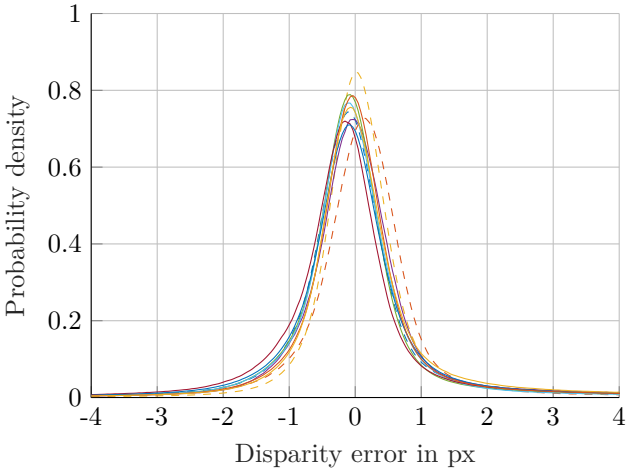
For our final fitness cost function we chose to punish bad pixels instead of minimizing the mean absolute error (MAE) or the root mean square error (RMSE). To compare these variants, we computed the mean MAE and mean RMSE over all considered frames of sequence 00 and tested it on the whole sequence. In Figure 8 we show the disparity error distribution of the calibration trained with our bad pixel approach, an MAE approach and an an RMSE approach. The bad pixels approach delivers a less biased error distribution with less variance. Even with perfect calibration, the error distribution would still possess non-empty tails due to stereo matching outliers, which would dramatically influence the RMSE value and also MAE values. Our function represents a robust alternative.

Additionally, we tested training on every 25th frame or 100th frame instead of every 50th frame, which is also depicted in Figure 8. This did not significantly improve the error distribution, which means that skipping even more frames than 50 might speed up the calibration process even further without introducing significant errors.

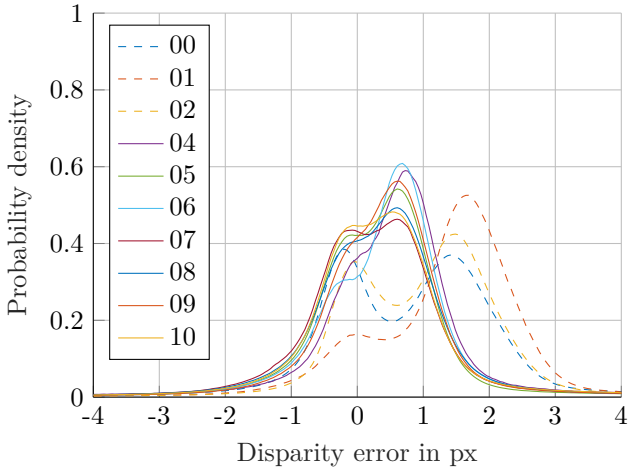
Furthermore, the optimization was evaluated with a standard downhill simplex method and a standard gradient descent method. We computed the cost of a parameter set with our GPU implementation of the fitness cost function. The downhill simplex version in our implementation was not suitable, because it quickly got stuck in local minima. The gradient descent approach behaves similarly, but depends strongly on the choice of the step size. A gradual decrease of the step size for example proved to produce comparable results to our evolution algorithm. Still, our evolution method was faster in delivering good results due to the sheer amout of parallelization. Also, our fitness cost function compares the ratio of bad pixels over the chosen frames. This means that it is a non-differentiable function due to the discrete pixel steps, which might prove to be problematic for the convergence of gradient descent methods.



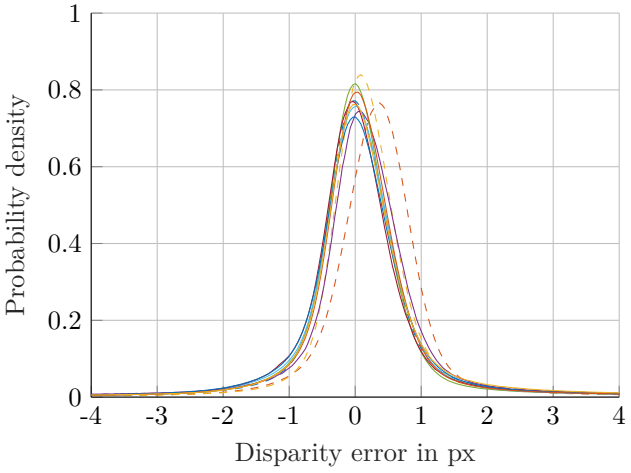
(a) ELAS Original



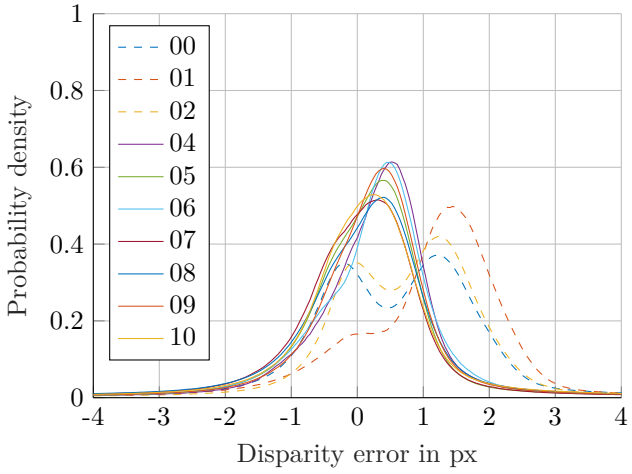
(b) ELAS Recalibration



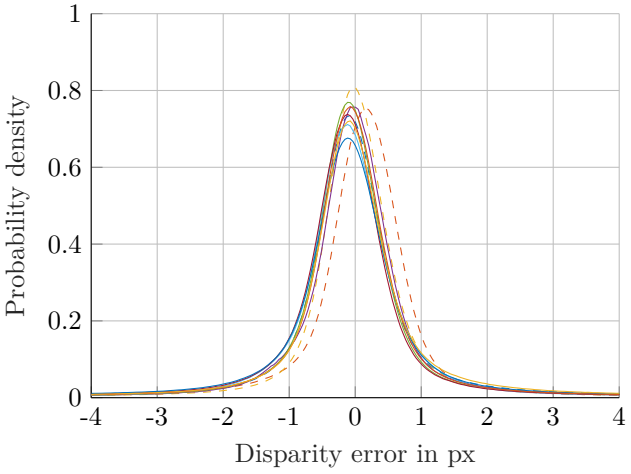
(c) SGBM Original



(d) SGBM Recalibration



(e) MeshStereo Original



(f) MeshStereo Recalibration

Figure 7: Left column: Original disparity error distributions of different stereo matching algorithms over the KITTI visual odometry training sequences. Right column: Applying the resulting offsets of the genetic algorithm.

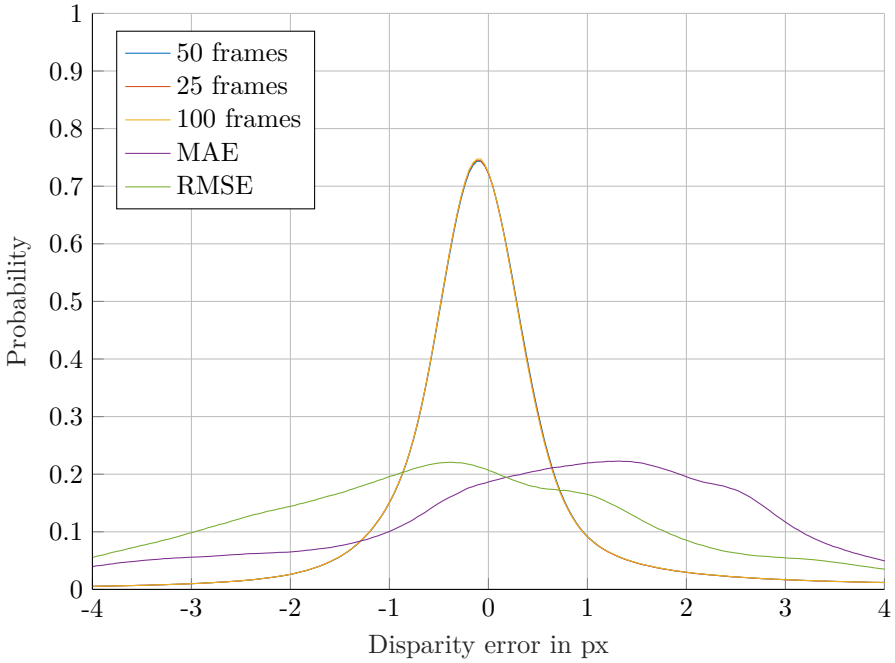


Figure 8: Comparison of the disparity error distributions on sequence 00 of a calibration trained in the same sequence with different frame skips (50, 25, 100) and other fitness cost functions (mean MAE, mean RMSE) than ours. Note that the frame skip graphs are overlapping.

5. CONCLUSION AND FUTURE WORK

We presented a novel recalibration method for existing depth sensors calibrations using a genetic algorithm. We showed the potential of disparity accuracy improvements with fusion if the disparity error distribution is indeed Gaussian. To demonstrate the effectiveness of our algorithm, we showed that the Lidar to stereo camera transformation in the KITTI visual odometry dataset is not sufficiently calibrated, i.e. the disparity error distribution is not Gaussian. The resulting new calibration shows significant improvements in the evaluation, which was even true for other stereo matching algorithms than the input.

For future work, it might be interesting to see if an even faster implementation or better fitness function would make stereo to stereo calibration feasible for the automatic online calibration of depth sensors even during transit. It might even be possible to refine stereo camera pose movements between frames, i.e. as a pose movement optimizer in a visual odometry framework. Furthermore, a comparison to an analogical ICP variant may yield more insight on the true performance of the proposed method.

REFERENCES

[1] A. Geiger, P. Lenz, and R. Urtasun, “Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite,” in Conference on Computer Vision and Pattern Recognition (CVPR), (2012).

[2] D. Whitley, “A genetic algorithm tutorial,” Statistics and computing 4(2), 65–85 (1994).

[3] P. J. Besl and N. D. McKay, “Method for registration of 3-d shapes,” in Sensor Fusion IV: Control Paradigms and Data Structures, 1611, 586–607, International Society for Optics and Photonics (1992).

[4] L. Matthies, T. Kanade, and R. Szeliski, “Kalman Filter-based Algorithms for Estimating Depth from Image Sequences,” International Journal of Computer Vision 3(3), 209–238 (1989).

[5] S. Lee and Y. Kay, “A Kalman Filter Approach for Accurate 3-D Motion Estimation from a Sequence of Stereo Images,” in 10th International Conference on Pattern Recognition, 1990. Proceedings., 1, 104–108, IEEE (1990).

[6] U. Franke, C. Rabe, H. Badino, and S. Gehrig, "6D-Vision: Fusion of Stereo and Motion for Robust Environment Perception," in Joint Pattern Recognition Symposium, 216–223, Springer (2005).

[7] G. Sibley, G. S. Sukhatme, and L. H. Matthies, "The Iterated Sigma Point Kalman Filter with Applications to Long Range Stereo.," in Robotics: Science and Systems, 8(1), 235–244 (2006).

[8] A. Wedel and D. Cremers, Stereo Scene Flow for 3D Motion Analysis, Springer Science & Business Media (2011).

[9] S. Morales and R. Klette, "Kalman-filter Based Spatio-temporal Disparity Integration," Pattern Recognition Letters 34(8), 873–883 (2013).

[10] D. Scharstein and R. Szeliski, "Middlebury Stereo Vision Page," Online at http://vision.middlebury.edu/stereo/eval3/ Version 3 (2002).

[11] M. Menze and A. Geiger, "Object Scene Flow for Autonomous Vehicles," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 3061–3070 (2015).

[12] C. Geyer, T. Templeton, M. Meingast, and S. S. Sastry, "The Recursive Multi-Frame Planar Parallax Algorithm," in Third International Symposium on 3D Data Processing, Visualization, and Transmission, 17–24, IEEE (2006).

[13] D. Gallup, Efficient 3D Reconstruction of Large-scale Urban Environments from Street-level Video, PhD thesis, The University of North Carolina at Chapel Hill (2011).

[14] G. Sibley, L. Matthies, and G. Sukhatme, "Bias Reduction and Filter Convergence for Long Range Stereo," Robotics Research , 285–294 (2007).

[15] T. D. Barfoot, State Estimation for Robotics, Cambridge University Press (2017).

[16] L. Matthies and P. Grandjean, "Stochastic Performance, Modeling and Evaluation of Obstacle Detectability with Imaging Range Sensors," IEEE Transactions on Robotics and Automation 10(6), 783–792 (1994).

[17] N. Sebe and M. S. Lew, "Maximum Likelihood Stereo Matching," in Proceedings of the 15th International Conference on Pattern Recognition, 1, 900–903, IEEE (2000).

[18] A. Geiger, M. Roser, and R. Urtasun, "Efficient Large-Scale Stereo Matching," in Asian Conference on Computer Vision, 25–38, Springer (2010).

[19] A. Geiger, J. Ziegler, and C. Stiller, "StereoScan: Dense 3D Reconstruction in Real-time," in Intelligent Vehicles Symposium (IV), 2011 IEEE, 963–968, IEEE (2011).

[20] S. Sengupta, E. Greveson, A. Shahrokni, and P. H. Torr, "Urban 3D Semantic Modelling Using Stereo Vision," in 2013 IEEE International Conference on Robotics and Automation (ICRA), 580–585, IEEE (2013).

[21] P. F. Alcantarilla, C. Beall, and F. Dellaert, "Large-scale Dense 3D Reconstruction From Stereo Imagery," Georgia Institute of Technology (2013).

[22] V. Usenko, J. Engel, J. Stückler, and D. Cremers, "Reconstructing Street-Scenes in Real-Time from a Driving car," in Proceedings of the 2015 International Conference on 3D Vision, 607–614, IEEE (2015).

[23] S. Pillai, S. Ramalingam, and J. J. Leonard, "High-Performance and Tunable Stereo Reconstruction," in 2016 IEEE International Conference on Robotics and Automation (ICRA), 3188–3195, IEEE (2016).

[24] M. Tanner, P. Pinies, L. M. Paz, and P. Newman, "DENSER Cities: A System for Dense Efficient Reconstructions of Cities," arXiv preprint arXiv:1604.03734 (2016).

[25] C. Cigla, R. Brockers, and L. Matthies, "Gaussian Mixture Models for Temporal Depth Fusion," in 2017 IEEE Winter Conference on Applications of Computer Vision (WACV), 889–897, IEEE (2017).

[26] I. A. Bârsan, P. Liu, M. Pollefeys, and A. Geiger, "Robust Dense Mapping for Large-Scale Dynamic Environments," (2018). Submitted to ICRA 2018.

[27] OpenCV, "Open Source Computer Vision Library." https://github.com/opencv/opencv (2018).

[28] H. Hirschmüller, "Stereo Processing by Semiglobal Matching and Mutual Information," IEEE Transactions on Pattern Analysis and Machine Intelligence 30(2), 328–341 (2008).

[29] C. Zhang, Z. Li, Y. Cheng, R. Cai, H. Chao, and Y. Rui, "MeshStereo: A Global Stereo Model With Mesh Alignment Regularization for View Interpolation," in The IEEE International Conference on Computer Vision (ICCV), 2057–2065 (12 2015).

[30] C. Guindel, J. Beltrán, D. Martín, and F. García, "Automatic Extrinsic Calibration for Lidar-Stereo Vehicle Sensor Setups," arXiv preprint arXiv:1705.04085 (2017).

[31] J. Nickolls, I. Buck, M. Garland, and K. Skadron, "Scalable Parallel Programming with CUDA," in ACM SIGGRAPH 2008 classes, 16, ACM (2008).

[32] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets Robotics: The KITTI Dataset," The International Journal of Robotics Research 32(11), 1231–1237 (2013).