



Lidar-Monocular Visual Odometry with Genetic Algorithm for Parameter Optimization

Adarsh Sehgal[✉], Ashutosh Singandhupe, Hung Manh La[✉],
Alireza Tavakkoli, and Sushil J. Louis

University of Nevada, Reno, NV 89557, USA
hla@unr.edu
<https://ara.cse.unr.edu>

Abstract. Lidar-Monocular Visual Odometry (LIMO), an odometry estimation algorithm, combines camera and Light Detection And Ranging sensor (LIDAR) for visual localization by tracking features from camera images and LIDAR measurements. LIMO then estimates the motion using Bundle Adjustment based on robust key frames. LIMO uses semantic labelling and weights of the vegetation landmarks for rejecting outliers. A drawback of LIMO as well as many other odometry estimation algorithms is that it has many parameters that need to be manually adjusted according to dynamic changes in the environment in order to decrease translational errors. In this paper, we present and argue the use of Genetic Algorithm (GA) to optimize parameters with reference to LIMO and maximize LIMO's localization and motion estimation performance. We evaluate our approach on the well-known KITTI odometry dataset and show that the GA helps LIMO to reduce translation error in different datasets.

Keywords: LIMO · Genetic Algorithm · Sensor odometry

1 Introduction

Motion estimation has long been a popular subject of research in which many techniques have been developed over the years. Much work has been done related to Visual Simultaneous Localization and Mapping (VSLAM), also referred to as Visual Odometry, which simultaneously estimates the motion of the camera and the 3D structure of the observed environment. A recent review of SLAM techniques for autonomous car driving can be found in [18]. Bundle Adjustment

This material is based upon work supported by the National Aeronautics and Space Administration (NASA) Grant No. NNX15AI02H issued through the NVSGC-RI program under sub-award No. 19–21, and the RID program under sub-award No. 19–29, and the NVSGC-CD program under sub-award No. 18–54. This work is also partially supported by the Office of Naval Research under Grant N00014-17-1-2558.

is the most popular method for VSLAM. Bundle Adjustment is a procedure of minimizing the re-projection error between the observed point (landmarks in reference to LIMO) and the predicted points. Recent developments make use of offline VSLAM for mapping and localization [19].

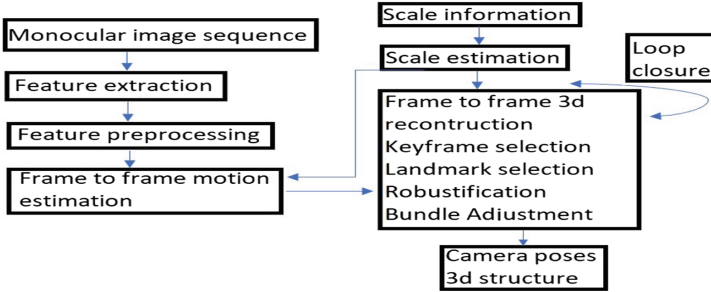


Fig. 1. VSLAM pipeline. The input is temporal sequence of images, and the system outputs a sparse reconstruction of the observed environment and the camera poses [13]. In this work, LIMO does not perform loop closure.

Figure 1 illustrates the structure of the VSLAM pipeline [13]. Algorithms such as ROCC [2] and SOFT [4] rely on pre-processing and feature extraction, which is in very contrast to most of the methods that obtain scale information from a camera placed at a different viewpoint. The former mentioned algorithms (SOFT and ROCC) extract robust and precise features and select them using special techniques, and have managed to attain high performance on the KITTI Benchmark [6], even without Bundle Adjustment.

Light Detection And Ranging (LIDAR)-camera calibration is an expanding topic of research [8, 14] with accuracy reaching a few pixels. Previous work has been done with VSLAM and LIDAR [24, 25]. Lidar-Monocular Visual Odometry (LIMO) [13] uses feature tracking capability of the camera and combines it with depth measurements from a LIDAR but suffers from translation and rotation errors. Later on, we describe our approach for increasing LIMO’s robustness to translation errors.

LIMO takes advantage of the depth information from LIDAR, which is to be used for feature detection in the image. Outliers are rejected if do not meet the local plane assumptions, and points on the ground plane are treated for robustness. As illustrated in Fig. 1, in the VSLAM pipeline, the depth information is fused with monocular feature detection techniques. Another approach is taken for prior estimation, landmark selection and key frame selection to fulfill real time constraints. Unlike the approach in [25], LIMO does not use any LIDAR-SLAM algorithms such as Iterative Closest Point (ICP). The major drawback of LIMO is that it has many parameters, which needs to be manually tuned. LIMO suffers from translation and rotation errors even more than existing algorithms such as Lidar Odometry and Mapping (LOAM) [24] and Vision-Lidar Odometry

and Mapping (V-LOAM) [1]. Typically, researchers tune parameters (in LIMO as well) in order to minimize these errors but there always exists the possibility of finding better parameter sets that may be optimized for specific camera and LIDAR hardware or for specific scenarios. Hence, there is a need to use optimization algorithms to increase LIMO’s performance. In this paper, we propose using a genetic algorithm (GA) to efficiently search the space of possible LIMO parameter values to find precise parameters that maximize performance. Our experiments with this new GA-LIMO algorithm show that GA-LIMO performs better than stock LIMO.

Much empirical evidence shows that evolutionary computing techniques such as GA work well as function optimizers in poorly-understood, non-linear, discontinuous spaces [10, 22]. GA and the GA operators of crossover and mutation have been tested on numerous problems. Closer to our research, GA has been applied to early SLAM optimization [5], multi-robot flocking [16], and deep reinforcement learning [17]. This provides good evidence for GA efficacy on localization problems, and our main contribution in this paper is a demonstration of smaller translation error when using a GA to tune LIMO parameter values compared to the stock LIMO Algorithm [13]. Our experiments show that translation error is non-linearly related to LIMO parameters, that is, translation error can vary non-linearly based on the values of the LIMO’s parameters and thus suitable for GA search. The following sections describe the LIMO, the GA and GA-LIMO algorithms. We then show results from running LIMO with GA tuned parameters on the KITTI odometry data sequences [7].

2 Background

In this section, we present prior work related to our GA-LIMO algorithm. We first describe the VSLAM pipeline and then the LIMO algorithm.

2.1 Feature Extraction and Pre-processing

Figure 1 shows feature extraction’s procedure in the pipeline. Feature extraction consists of tracking the features and associating the features using the Viso2 library [9]. It is further used to implement feature tracking, which comprises of non-maximum suppression, sub-pixel refinement and outlier rejection by flow. Deep learning is used to reject landmarks that are moving objects. The neighborhood of the feature point in a semantic image [3] is scanned, and if the majority of neighboring pixels categorize to a dynamic class, like vehicle or pedestrian, the landmark is excluded.

2.2 Scale Estimation

For scale estimation, the detected feature points from camera is mapped to the depth extracted from LIDAR. LIMO uses a one shot depth estimation approach. Initially LIDAR point cloud is transformed into the camera frame, and then it

is projected onto the image plane. In detail, the following steps are executed for every feature point f :

1. A region of interest is selected around f , which is a set F consisting of projected LIDAR points.
2. A new set called foreground set F_{seg} is created by segmenting the elements of F .
3. The elements of F_{seg} are fitted with a plane p . A special fitting algorithm is used in case f belongs to the ground plane.
4. To estimate the depth, p is intersected with the line of sight corresponding to f .
5. For the previous estimated depth, a test is performed. Depth estimates that are more than 30 m are rejected since they could be uncertain. In addition, the angle between the line of sight of the feature point and the normal of the plane must be smaller than a threshold.

From the point clouds, neighborhoods for ordered point clouds can be selected directly. However, projections of the LIDAR points on the image are used, and the points within a rectangle in the image plane around f are selected in case the point clouds are unordered. Before the plane estimation is performed, the foreground F_{seg} is segmented. In the next step, a histogram of depth having a fixed bin width of $h = 0.3$ m is created and interpolated with elements in F . LIDAR points of the nearest bin is used to perform segmentation using all detected feature points. For estimating the local surface around f precisely, fitting the plane to F_{seg} can help. Three points are chosen from the points in F_{seg} , which traverse the triangle F_{Δ} with maximum area. Depth estimation is avoided if the area of F_{Δ} is too small, to evade incorrectly estimated depth.

However, the above technique cannot be used to estimate the depth of points on the ground plane because LIDAR has a lower resolution in the perpendicular direction than in a level direction. A different approach is followed to enable depth estimation for a relevant ground plane. For solving this, RANSAC with refinement is used on the LIDAR point cloud to extract the ground plane [21]. In order to estimate feature points on the road, points that corresponds to the ground plane are segmented. Outliers are extracted by allowing only local planes that lie close to the ground plane.

2.3 Frame to Frame Odometry

Perspective-n-Point-Problem [21] serves as the starting point of the frame to frame motion estimation.

$$\underset{x,y,z,\alpha,\beta,\gamma}{\operatorname{argmin}} \sum_i \|\varphi_{i,3d \rightarrow 2d}\|_2^2 \quad (1)$$

$$\varphi_{3d \rightarrow 2d} = \bar{p}_i - \pi(p_i, P(x, y, z, \alpha, \beta, \gamma)), \quad (2)$$

where \bar{p}_i is the observed feature point in the current frame, p_i is the 3D-point corresponding to \bar{p}_i , the transform from the previous to the current frame is

denoted by freedom $P(x, y, z, \alpha, \beta, \gamma)$, which has three translation and three rotation degrees of freedom. While $\pi(\cdot)$ is the projection function from the 3D to 2D domain. The extracted features with valid estimated depth from the environments that has low structure and large optical flow may be very small. LIMO introduces epipolar error as $\varphi_{2d \rightarrow 2d}$ [15].

$$\varphi_{2d \rightarrow 2d} = \bar{p}_i F \left(\frac{x}{z}, \frac{y}{z}, \alpha, \beta, \gamma \right) \bar{p}_i, \quad (3)$$

where fundamental matrix F can be calculated from the intrinsic calibration of the camera and from the frame to frame motion of the camera. LIMO suggests the loss function to be Cauchy function [15]: $\rho_s(x) = a(s)^2 \cdot \log(1 + \frac{x}{a(s)^2})$, where $a(s)$ is the fix outlier threshold. For frame to frame motion estimation, the optimization problem can be denoted as:

$$\underset{x, y, z, \alpha, \beta, \gamma}{\operatorname{argmin}} \sum_i \rho_{3d \rightarrow 2d}(\|\varphi_{i, 3d \rightarrow 2d}\|_2^2) + \rho_{2d \rightarrow 2d}(\|\varphi_{i, 2d \rightarrow 2d}\|_2^2). \quad (4)$$

2.4 Backend

LIMO proposes a Bundle Adjustment framework based on keyframes, with key components as selection of keyframes, landmark selection, cost functions and robustification measures. The advantage with this approach is that it retains the set that carries information, which is required for accurate pose estimation as well as excludes the unnecessary measurements. Keyframes are classified as required, rejected, and sparsified keyframes. Required frames are crucial measurements. Frame rejection is done when the vehicle does not move. The remaining frames are collected, and the technique selects frames every 0.3s. Finally in keyframe selection, length of optimization window is chosen.

An optimal set of landmarks should be well observable, small, free of outliers and evenly distributed. Landmark selection divides all landmarks into three bins, near, middle and far, each of which have fixed number of landmarks selected for the Bundle Adjustment. Weights of the landmarks are then determined based on the semantic information. The estimated landmark depth is taken into consideration by an additional cost function,

$$\xi_{i,j}(l_i, P_j) = \begin{cases} 0, & \text{if } l_i \text{ has no depth estimate} \\ \hat{d}_{i,j} - \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \tau(l_i, P_j), & \text{else,} \end{cases} \quad (5)$$

where l_i denotes the landmark, τ mapping from world frame to camera frame and \hat{d} denotes the depth estimate. The indices i, j denote combination of landmark-pose. A cost function ν punishes deviations from the length of translation vector,

$$\nu(P_1, P_0) = \hat{s}(P_1, P_0) - s, \quad (6)$$

where P_0, P_1 are the last two poses in the optimization window and $\hat{s}(P_0, P_1) = \|\operatorname{translation}(P^{-1}P_1)\|_2^2$, where s is a constant with value of $\hat{s}(P_1, P_0)$ before optimization.

While outliers need to be removed because they do not let Least-Square methods to converge [23], semantics and cheirality only does preliminary outlier rejection. The LIMO optimization problem can now be formulated as:

$$\begin{aligned} & \underset{P_j \in P, l_i \in L, d_i \in D}{\operatorname{argmin}} \quad w_0 \|\nu(P_1, P_0)\|_2^2 + \\ & \sum_i \sum_j w_1 \rho_\phi(\|\phi_{i,j}(l_i, P_i)\|_2^2) + w_2 \rho_\xi(\|\xi_{i,j}(l_i, P_j)\|_2^2), \end{aligned} \quad (7)$$

where $\phi_{i,j}(l_i, P_j) = \bar{l}_{i,j} - \pi(l_i, P_j)$ is the re-projection error, and weights w_0 , w_1 and w_2 are used to scale the cost functions to the same order of magnitude.

3 GA-LIMO Algorithm

In this section, we present the main contribution of our paper. The specific GA searches through the space of parameter values used in LIMO for parameter values that maximize performance and minimize translation error as a result of pose estimation. We are targeting the following parameters: outlier rejection quantile δ ; maximum number of landmarks for near bin ϵ_{near} ; maximum number of landmarks for middle bin ϵ_{middle} ; maximum number of landmarks for far bin ϵ_{far} and weight for the vegetation landmarks μ . As described in the background section on rejecting outliers, δ , plays an important role in converging to minimum. Thus the weight of outlier rejection has notable impact on translation error. Landmarks are categorized into three bins, which also have great significance in translation error calculation. Trees that have a rich structure result in feature points that are good to track, but they can move. So, finding an optimal weight for vegetation can significantly reduce translation error. δ and μ range from 0 to 1, while ϵ_{near} , ϵ_{middle} and ϵ_{far} range from 0 to 999. We have set these ranges based on early experimental results. Our experiments show that adjusting the values of parameters did not decrease or increase the translation error in a linear or easily appreciable pattern. So, a simple hill climber will not do well in finding optimized parameters. We thus use a GA to optimize these parameters.

Algorithm 1 explains the combination of LIMO with the GA, which uses a population size of 50 runs for 50 generations. We used ranking selection [11] to select the parents for crossover and mutation. Rank selection probabilistically selects higher ranked (higher fitness) individuals. Unlike fitness proportional selection, ranking selection pays attention to the existence of a fitness difference rather than to the magnitude of fitness difference. Children are generated using uniform crossover, which are then mutated using flip mutation [12]. Chromosomes are binary encoded with concatenated parameters. δ and μ are considered up to three decimal places, which means a step size of 0.001, because changes in values of parameters cause considerable change in translation error. All the parameters require 11 bits to represent their range of values, so we have a chromosome length of 55 bits, with parameters arranged in the order: δ , ϵ_{near} , ϵ_{middle} , ϵ_{far} , μ .

Algorithm 1. GA-LIMO

-
- 1: Choose population of n chromosomes
 - 2: Set the values of parameters into the chromosome
 - 3: Run LIMO with the GA selected parameter values
 - 4: **for** all chromosome values **do**
 - 5: Run LIMO on KITTI odometry data set sequence 01
 - 6: Compare LIMO estimated poses with ground truth
 - 7: Translation error σ_1 is found
 - 8: Run LIMO on KITTI odometry data set sequence 04
 - 9: Compare LIMO estimated poses with ground truth
 - 10: Translation error σ_4 is found
 - 11: Average error $\sigma_{avg} = \frac{\sigma_1 + \sigma_4}{2}$
 - 12: **return** $1/\sigma_{avg}$
 - 13: **end for**
 - 14: Perform Uniform Crossover
 - 15: Perform Flip Mutation at rate 0.1
 - 16: Repeat for required number of generations for optimal solution
-

The algorithm starts with randomly generating a population of n individuals. Each chromosome is sent to LIMO to evaluate. LIMO evaluates the parameter set represented by this individual by using those parameters to run on the KITTI dataset [6]. The KITTI benchmarks are well known and provide the most popular benchmark for Visual Odometry and VSLAM. This dataset has rural and urban scenes along with highway sequences and provides gray scale images, color images, LIDAR point clouds and their calibration. Most LIMO configurations are as in [13]. In our work, we focus on two sequences in particular: sequence 01 and 04. Sequence 01 is a highway scenario, which is challenging because only a road can be used for depth estimation. Sequence 04 is an urban scenario, which has a large number of landmarks for depth estimation. We consider both sequences for each GA evaluation because in this initial research we want a common set of parameters that works well with multiple scenes.

The fitness of each chromosome is defined as the inverse of translation error. This translates the minimization of translation error into a maximization of fitness as required for GA optimization. Since each fitness evaluation takes significant time, an exhaustive search of the 2^{55} size search space is not possible, hence we are using the GA. During a fitness evaluation, the GA first runs the LIMO with sequence 01. It then compares the LIMO estimated poses with ground truth (also found in [6]) and finds the translation error using the official KITTI metric [6]. The same steps are followed for sequence 04. The fitness value of each chromosome is the average of the inverse translation errors from the two sequences.

$$\sigma_{avg} = \frac{\sigma_1 + \sigma_4}{2}. \quad (8)$$

Selected chromosomes (ranked selection) are then crossed over and mutated to create new chromosomes to form the next population. This starts the next GA iteration of evaluation, selection, crossover, and mutation. The whole system

takes significant time since we are running $50 \times 50 = 2500$ LIMO evaluations to determine the best parameters. The next section shows our experiments with individual and combined sequences, with and without the GA. Our results show that the GA-LIMO performs better than the results of LIMO [13].

4 Experimental Results

In this section we show our experiments with individual KITTI sequences, a combination of sequences, and overall results. First, we run the GA-LIMO with sequences 01 and 04 separately. We show the translation error and the error mapped onto the trajectory compared to the ground truth (reference) [7]. We then, show our results when GA-LIMO runs with evaluations on both sequences 01 and 04. Finally, we compare the values of parameters found by GA-LIMO versus LIMO. Figure 2 shows camera data while GA-LIMO is estimating the pose from that data in Fig. 3.



Fig. 2. Camera data while GA-LIMO is in action.

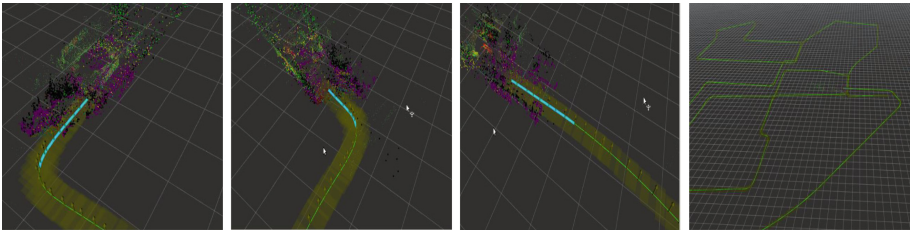


Fig. 3. GA-LIMO estimating the pose.

Figure 4 compares LIMO performance with GA-LIMO on sequence 04. Here the GA was run on this sequence individually to find the optimal parameters, as in Algorithm 2. Absolute Pose Error (APE) and Root Mean Squared Error (RMSE) are one of the important measures [20]. The translation error for each

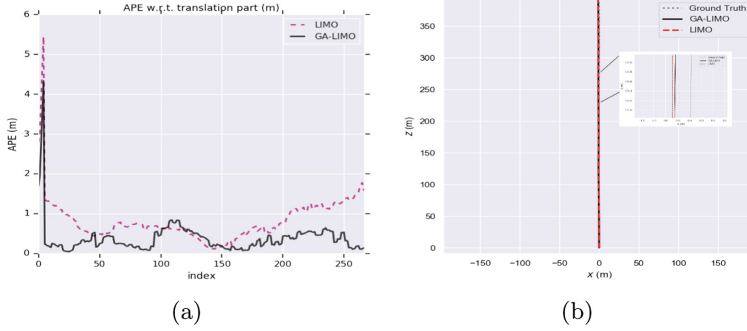


Fig. 4. Results comparison for sequence 04 (Algorithm 2). LIMO has 1.01% translation error, while GA-LIMO has about half this error with 0.56%. (a) Translation error comparison over the poses. (b) Trajectory comparison for sequence 04, when GA-LIMO was run on this sequence individually (Algorithm 2).

sequence is the RMSE calculated with respect to ground truth. Figure 4(a) compares the translation error over the poses, while Fig. 4(b) compares the error mapped onto the trajectory with the zoomed in trajectory. Our results show that the GA-LIMO trajectory is closer to ground truth compared to LIMO. We found that the translation error was 0.56% with GA-LIMO, in contrast to 1.01% with LIMO.

Algorithm 2. GA-LIMO individual

- 1: Choose population of n chromosomes
 - 2: Set the values of parameters into the chromosome
 - 3: Run LIMO with the GA selected parameter values
 - 4: **for** all chromosome values **do**
 - 5: Run LIMO on individual KITTI odometry dataset sequence
 - 6: Compare LIMO estimated poses with ground truth
 - 7: Translation error σ is found
 - 8: **return** $1/\sigma$
 - 9: **end for**
 - 10: Perform Uniform Crossover
 - 11: Perform Flip Mutation at rate 0.1
 - 12: Repeat for required number of generations to find optimal solution
-

Figure 5 compares the performance of LIMO with GA-LIMO when the system is run on just sequence 01. Here first, the GA was run on sequence 01 (Algorithm 2) and the optimal parameters were used to test the same sequence. Table 1 compares the original and GA found parameter values. Figure 5(a) compares translation error, while Fig. 5(b) shows the error mapped onto the trajectory for LIMO and GA-LIMO. As shown in the zoomed in-Fig. 5(b), GA-LIMO is closer to the ground truth. The translation error for LIMO is found to be around

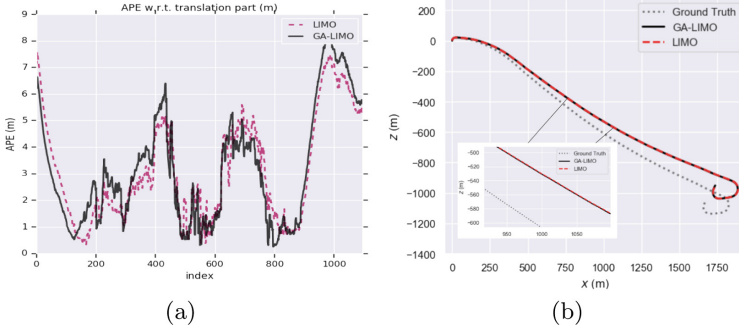


Fig. 5. Results comparison for sequence 01 (Algorithm 2). LIMO has 3.71% translation error, while GA-LIMO has 3.8%. (a) Translation error comparison over the poses. (b) Trajectory comparison.

3.71% and 3.8% in case of GA-LIMO, with sequence 01. GA found parameters that did not outperform the original parameters, when GA-LIMO was run on just sequence 01.

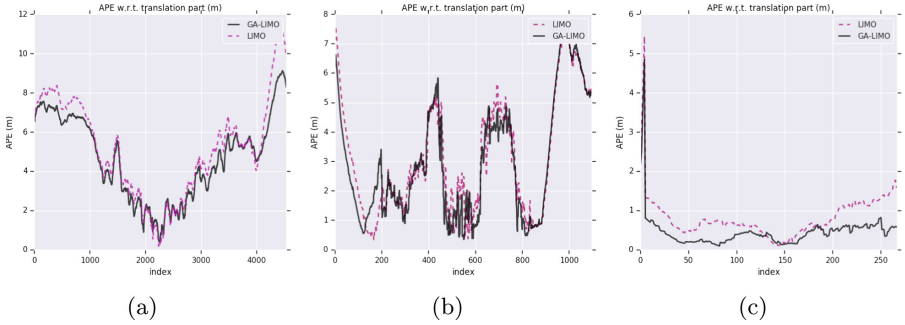


Fig. 6. (a) Translation error comparison over the poses for sequence 01. LIMO has 3.71% translation error while GA-LIMO has 3.59%. (b) Translation error comparison over the poses for sequence 00. LIMO has 5.77% translation error while GA-LIMO has 5.13%. (c) Translation error comparison over the poses for sequence 04. LIMO has 1.01% translation error while GA-LIMO has 0.65%. The parameters are found using GA-LIMO using combination of sequences 01 and 04 (Algorithm 1). These parameters are then tested on three sequences. In all three sequences GA-LIMO performs better than LIMO.

We finally ran the system with both sequences 01 and 04 as described in Algorithm 1. The fitness of each evaluation is the average of translation errors of the sequences when run using the input parameters. The parameters found in GA-LIMO as shown in Table 1, were then tested on sequences 00, 01 and 04, as shown in Figs. 6 and 7. It is evident that GA-LIMO performed better than

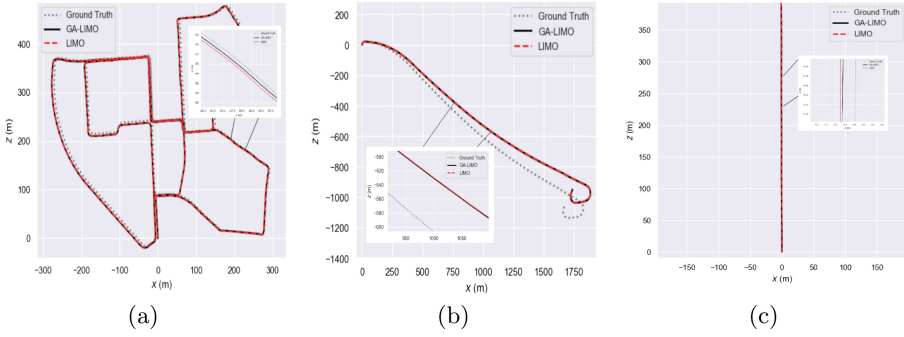


Fig. 7. Trajectory comparison when GA-LIMO was run as in Algorithm 1. In all three sequences GA-LIMO performs better than LIMO.

Table 1. LIMO vs. GA-LIMO parameters when GA ran on LIMO: (Left table)-with sequence 01 individually; (Middle table)-with combined sequence 01 and 04; (Right table)-with sequence 04 individually.

Param	LIMO	GA-LIMO	Param	LIMO	GA-LIMO	Param	LIMO	GA-LIMO
δ	0.95	0.958	δ	0.95	0.963	δ	0.95	0.986
ϵ_{near}	400	999	ϵ_{near}	400	999	ϵ_{near}	400	999
ϵ_{middle}	400	593	ϵ_{middle}	400	554	ϵ_{middle}	400	960
ϵ_{far}	400	877	ϵ_{far}	400	992	ϵ_{far}	400	859
μ	0.9	0.813	μ	0.9	0.971	μ	0.9	0.128

LIMO in all three sequences. The zoomed-in figures show a closer view on one part of the trajectories. GA-LIMO trajectories are closer to the ground truth and have lesser translation errors. GA-LIMO has a translation error of 5.13% with sequence 00, 3.59% with sequence 01 and 0.65% with sequence 04, in contrast with 5.77% with sequence 00, 3.71% with sequence 01 and 1.01% with sequence 04 using original parameters.

Our method helped to find common set of optimal parameters, which works better and hence lead to better performance in different kinds of environments.

5 Discussion and Future Work

This paper shows results that demonstrated that the GA can tune LIMO parameters to achieve better performance, reduced translation error, across a range of scenarios. We discussed existing work on VSLAM, presented an algorithm to integrate LIMO with GA to find LIMO parameters that robustly minimize translation error, and explained why a GA might be suitable for such optimization. Initial results had the assumption that GAs are a good fit for such parameter optimization, and our results show that the GA can find parameter values that

lead to faster learning and better (or equal) performance. We thus provide further evidence that heuristic search as performed by genetic and other similar evolutionary computing algorithms are a viable computational tool for optimizing LIMO's performance. Open source code for this paper is available on our ARA Lab's github: <https://github.com/arialab-unr/LIMOWithGA>.

References

1. Balazadegan Sarvrood, Y., Hosseinyalamdary, S., Gao, Y.: Visual-lidar odometry aided by reduced IMU. *ISPRS Int. J. Geo-Inf.* **5**(1), 3 (2016)
2. Buczko, M., Willert, V.: Flow-decoupled normalized reprojection error for visual odometry. In: 19th IEEE International Conference on Intelligent Transportation Systems, pp. 1161–1167 (2016)
3. Cordts, M., et al.: The cityscapes dataset for semantic urban scene understanding. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3213–3223 (2016)
4. Cvišić, I., Petrović, I.: Stereo odometry based on careful feature selection and tracking. In: European Conference on Mobile Robots, pp. 1–6 (2015)
5. Duckett, T.: A genetic algorithm for simultaneous localization and mapping. In: IEEE International Conference on Robotics and Automation, pp. 434–439, September 2003
6. Geiger, A., Lenz, P., Stiller, C., Urtasun, R.: Vision meets robotics: the kitti dataset. *Int. J. Robot. Res.* **32**(11), 1231–1237 (2013)
7. Geiger, A., Lenz, P., Urtasun, R.: Are we ready for autonomous driving? the kitti vision benchmark suite. In: Conference on Computer Vision and Pattern Recognition (2012)
8. Geiger, A., Moosmann, F., Car, Ö., Schuster, B.: Automatic camera and range sensor calibration using a single shot. In: IEEE International Conference on Robotics and Automation, pp. 3936–3943 (2012)
9. Geiger, A., Ziegler, J., Stiller, C.: Stereoscan: dense 3D reconstruction in real-time. In: IEEE Intelligent Vehicles Symposium, pp. 963–968 (2011)
10. Gibb, S., La, H.M., Louis, S.: A genetic algorithm for convolutional network structure optimization for concrete crack detection. In: 2018 IEEE Congress on Evolutionary Computation (CEC), pp. 1–8 (2018)
11. Goldberg, D.E., Deb, K.: A comparative analysis of selection schemes used in genetic algorithms. In: Foundations of Genetic Algorithms, vol. 1, pp. 69–93 (1991)
12. Goldberg, D.E., Holland, J.H.: Genetic algorithms and machine learning. *Mach. Learn.* **3**(2), 95–99 (1988)
13. Graeter, J., Wilczynski, A., Lauer, M.: LIMO: lidar-monocular visual odometry. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 7872–7879 (2018)
14. Gräter, J., Strauss, T., Lauer, M.: Photometric laser scanner to camera calibration for low resolution sensors. In: 19th IEEE International Conference on Intelligent Transportation Systems (ITSC), pp. 1552–1557 (2016)
15. Hartley, R., Zisserman, A.: Multiple View Geometry in Computer Vision. Cambridge university press, Cambridge (2003)
16. La, H.M., Nguyen, T.H., Nguyen, C.H., Nguyen, H.N.: Optimal flocking control for a mobile sensor network based a moving target tracking. In: IEEE International Conference on Systems, Man and Cybernetics, pp. 4801–4806, October 2009

17. Sehgal, A., La, H., Louis, S., Nguyen, H.: Deep reinforcement learning using genetic algorithm for parameter optimization. In: IEEE International Conference on Robotic Computing, pp. 596–601 (2019)
18. Singandhupe, A., La, H.: A review of SLAM techniques and security in autonomous driving. In: IEEE International Conference on Robotic Computing, pp. 602–607 (2019)
19. Sons, M., Lategahn, H., Keller, C.G., Stiller, C.: Multi trajectory pose adjustment for life-long mapping. In: 2015 IEEE Intelligent Vehicles Symposium (IV), pp. 901–906 (2015)
20. Sturm, J., Engelhard, N., Endres, F., Burgard, W., Cremers, D.: A benchmark for the evaluation of RGB-D SLAM systems. In: 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 573–580 (2012)
21. Szeliski, R.: Computer Vision: Algorithms and Applications. Springer, London (2010). <https://doi.org/10.1007/978-1-84882-935-0>
22. Tavakkoli, A., Ambardekar, A., Nicolescu, M., Louis, S.: A genetic approach to training support vector data descriptors for background modeling in video data. In: International Symposium on Visual Computing, pp. 318–327 (2007)
23. Torr, P.H., Fitzgibbon, A.W.: Invariant fitting of two view geometry. IEEE Trans. Pattern Anal. Mach. Intell. **26**(5), 648–650 (2004)
24. Zhang, J., Singh, S.: LOAM: lidar odometry and mapping in real-time. In: Robotics: Science and Systems, vol. 2, p. 9 (2014)
25. Zhang, J., Singh, S.: Visual-lidar odometry and mapping: low-drift, robust, and fast. In: IEEE International Conference on Robotics and Automation, pp. 2174–2181 (2015)