Lidar-Monocular Visual Odometry with Genetic Algorithm for Parameter Optimization

Adarsh Sehgal, Ashutosh Singandhupe, Hung Manh La, Alireza Tavakkoli, Sushil J. Louis

Abstract— **Lidar-Monocular Visual Odometry (LIMO), a depth extraction algorithm, combines camera and LIDAR for visual localization by tracking camera features from LIDAR measurements and for motion estimation by Bundle Adjustment based on robust keyframes. LIMO further uses semantic labelling for outlier rejection and weighting of vegetation landmarks. A drawback of LIMO is its many parameters that need to be manually adjusted to decrease translation errors. In this paper, we use a Genetic Algorithm to optimize LIMO parameters and maximize LIMO's localization and motion estimation performance. Our experimental evaluation on the well known KITTI odometry dataset shows that the genetic algorithm helps LIMO to reduce translation error in different scenes.**

I. INTRODUCTION AND RELATED WORK

Motion estimation has long been a subject for research for which many techniques have been developed over the years [1]. Much work has been done related to Visual Simultaneous Localization and Mapping (VSLAM) or Visual Odometry [2], which simultaneously estimates the 3d structure and motion of camera by observing the environment using a camera. A recent review of SLAM techniques for autonomous car driving can be found in [3]. Bundle Adjustment is the most popular method for VSLAM. It aligns bundles of lines of sight focus in observed points (landmarks). Recent developments make use of offline VSLAM for mapping and localization [4]–[6].

Figure 1 illustrates the structure of the VSLAM pipeline [7]. While a majority of the methods obtain scale information from a second camera [6], [8]–[10], algorithms such as ROCC [11] and SOFT [12] emphasize feature extraction and preprocessing. These algorithms extract precise and robust feature tracks and select them using special techniques, which attain high performance on the KITTI Benchmark [13], even without Bundle Adjustment.

The strong dependency on precise extrinsic camera calibration poses a major drawback to stereo vision. It was later found that learning a compensation of the calibration bias through a deformation field enhances performance [14]. LIDAR to camera calibration is also an expanding topic of research [15], [16] with accuracy reaching a few pixels. Previous work has been done with VSLAM and LIDAR [17]–[20]. Lidar-Monocular Visual Odometry (LIMO) [7], uses depth measurements from a LIght Detection And Ranging sensor (LIDAR) combined with the feature tracking capability of the camera but suffers from translational and rotational errors. Later in this section, we describe our approach for increasing LIMO's robustness to translational and rotational errors.

LIMO extracts depth information from LIDAR for feature detection in an image. Outliers not satisfying the local plane assumption are rejected and points on the ground plane are treated for robustness. As illustrated in figure 1, depth information is fused with monocular techniques in the VSLAM pipeline. Special care is taken for prior estimation, landmark selection and keyframe selection, to fulfill real time constraints. Unlike the approach in [18], 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 rotational errors even more than existing algorithms such as Lidar Odometry and Mapping (LOAM) [20] and Vision-Lidar Odometry and Mapping (V-LOAM) [21]. Typically, researchers tune LIMO parameters 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 to efficiently search the space of possible LIMO parameter values to find 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 Genetic Algorithms (GAs) work well as function optimizers in poorly-understood, non-linear, discontinuous spaces [22]–[26]. GAs [27], [28] and the GA operators of crossover and mutation [29] and have been applied to a variety of problems. Closer to our research, GAs have been applied to early SLAM optimization problems [30], mobile localization using ultrasonic sensors [31], and to deep reinforcement learning [32]. This provides good evidence for genetic algorithm efficacy on localization problems and our main contribution in this paper is a demonstration of better translation error when using a GA to tune LIMO parameter values compared to the stock LIMO algorithm [7]. Our experiments show that translation error is non-linearly

Adarsh Sehgal, Ashutosh Singandhupe and Dr. Hung La are with the Advanced Robotics and Automation (ARA) Laboratory. Dr. Alireza Tavakkoli is with the Computer Vision Laboratory (CVL). Dr. Sushil J. Louis is with the Evolutionary Computing Systems Lab, University of Nevada, Reno, NV 89557, USA. Corresponding author: Hung La, email: hla@unr.edu

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, 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.

related to LIMO parameters, that is, translation error can vary non-linearly based on the values of the LIMO's parameters. 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 benchmark data sequences [33].

This paper is organized as follows: Section 2 describes the LIMO and GA algorithms. Section 3 describes the combined GA-LIMO algorithm. Section 4 then specifies our experiments and provides results. The last section delivers conclusions and possible future work.

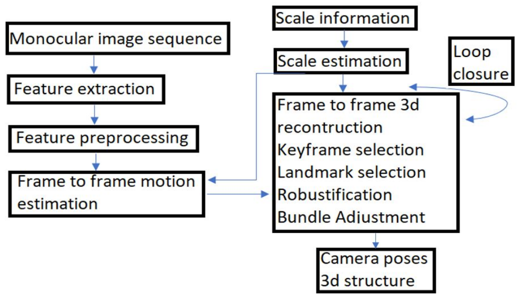


Fig. 1: Structure of the VSLAM pipeline. The input is a temporal image sequence, outputs are camera poses and a sparse reconstruction of the environment. Scale information to be obtained from another calibrated camera with known baseline [8], structure inherent information, such as the distance to the ground plane [34], IMU [35] or depth measurements from LIDAR [7]. In this work, LIMO does not perform loop closure.

II. BACKGROUND

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

A. Feature extraction and preprocessing

Figure 1 shows feature extraction's position in the pipeline. Feature extraction consists of feature tracking and feature association and the Viso2 library [9] is used to implement feature tracking which comprises of non-maximum suppression, outlier rejection by flow, and subpixel refinement. Deep learning is used to reject landmarks on or from moving objects. The neighborhood of the feature point in a semantic image [36] is scanned, and if the majority of neighboring pixels categorize to a dynamic class, like vehicle or pedestrian, the landmark is excluded.

B. Scale Estimation

For scale estimation, a depth corresponding to the detected feature points is extracted from LIDAR. LIMO uses a one shot depth estimation approach. It starts with LIDAR point cloud transformation into the camera frame and projected onto the image plane. The following steps are executed for every feature point f:

1) In a given region of interest around f, a set F of projected LIDAR points is chosen.
2) A new set F_{seg} is created by segmenting the elements of F that are in the foreground.
3) The elements of F_{seg} are fitted with a plane p. In case f belongs to the ground plane, a special fitting algorithm is used.
4) p is intersected with the line of sight corresponding to f to get its depth.
5) A test is performed for the estimated depth. Depth estimates of more than 30m are considered uncertain and rejected. 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.

Neighborhoods for ordered point clouds can be selected directly from the point cloud. However, for unordered point clouds, projections of the LIDAR points on the image are used and the points within a rectangle in the image plane around f are selected. The foreground F_{seg} is segmented before the execution of the plane estimation. Elements in F are then interpolated into a histogram by depth having a fixed bin width of $h = 0.3$m. Segmentation is exercised for all detected feature points by choosing the LIDAR points of the nearest notable bin. Fitting the plane to F_{seg} can precisely estimate the local surface around f. 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 depth of points on the ground plane cannot be estimated with this technique 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. A ground plane is first extracted from the LIDAR point cloud by RANSAC with refinement [5]. In order to estimate feature points on the road, points corresponding to the ground plane are segmented. Outliers are extracted by allowing only local planes that lie close to the ground plane.

C. Frame to Frame Odometry

Perspective-n-Point-Problem [5] serves as the starting point of the frame to frame motion estimation.

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

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

where $\bar{p_i}$ is the measured 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 $\pi(...)$ is the projection function from the 3d world domain to the 2d image domain. The extracted number of features with valid depth from the environments with low structure and large optical flow may be very small. LIMO introduces epipolar error as $\varphi_{2d\to 2d}$ [4].

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

where fundamental matrix F can be found from the frame to frame motion and the intrinsic calibration of the camera. LIMO suggests the loss function to be Cauchy function [4]: $\rho_s(x) = a(s)^2.log(1 + \frac{x}{a(s)^2})$, where $a(s)$ is the fix outlier threshold. The optimization problem for the frame to frame motion estimate is denoted as:

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

D. Backend

LIMO proposes a keyframe based Bundle Adjustment framework with key components as keyframe selection, landmark selection, cost functions and robustification measures. This approach excludes the unnecessary measurements while retaining the set that carries the information needed for an accurate pose estimation. 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 gathered and the technique chooses frames in the intervals of 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,

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

where l_i denotes the landmark, τ mapping from world coordinates to camera coordinates and $\hat{d}$ the depth estimate. The indices i, j denote landmark-pose combinations. 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 $\hat{s}(P_0, P_1) = \|translation(P^{-1}P_1)\|_2^2$, 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 [37], [38], semantics and cheirality only does preliminary outlier rejection. The LIMO optimization problem can now be formulated as:

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

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

E. Genetic Algorithm (GA)

Genetic Algorithms (GAs) [22], [23], [27], [39] were designed to search poorly-understood spaces, where exhaustive search may not be feasible (because of search space and time space complexity), and where other search techniques perform poorly. A genetic algorithm as a function optimizer tries to maximize a fitness tied to the optimization objective. In general, evolutionary computing algorithms and specifically GAs have had verifiable success on a diversity of difficulty, non-linear, design and optimization problems. GAs usually start with a randomly initializes population of candidate solution encoded in a string. Selection operators then focus search on higher fitness areas of search space whereas crossover and mutation operators generate new candidate solutions. We next explain the specific GA used in this paper.

III. GA-LIMO ALGORITHM

In this section, we present the main contribution of our paper. The specific genetic algorithm searches through the space of parameter values used in LIMO for the values that maximizes the performance and minimizes the 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, rejecting outliers, δ, plays an important role in converging to minimum, the weight of outlier rejection thus has notable impact on the translation error. The landmarks are categorized into three bins, which also have great significance in translation error calculation. The impact of landmarks on vegetation for visual odometry is also important. Trees have a rich structure resulting 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 probably 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 run for 50 generations. We used ranking selection [40] 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 also to the magnitude of fitness difference. Children are generated using uniform crossover [41], which are then mutated using flip mutation [23]. Chromosomes are binary encoded with

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 $\sigma_1$ is found
8:     Run LIMO on KITTI odometry data set sequence 04
9:     Compare LIMO estimated poses with ground truth
10:    Translation error $\sigma_4$ 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 No. of generations for optimal solution
```

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.

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 [13]. The KITTI benchmarks are well known and provide the most popular benchmark for Visual Odometry and VSLAM. This dataset has rural, 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 [7]. 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 genetic algorithm evaluation because we want a common set of parameters that work 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 genetic algorithm optimization. Since each fitness evaluation takes significant time, an exhaustive search of the 2^{55} size search space in not possible, hence our using the genetic algorithm. 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 [13]) and finds the translation error using the official KITTI metric [13]. 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.

$$\sigma_{avg} = \frac{\sigma_1 + \sigma_4}{2} \tag{8}$$

Selected chromosomes (ranked selection) are then crossed over and mutated to create new chromosomes to form the next population. This starts the next genetic algorithm iteration of evaluation, selection, crossover, and mutation. The whole system takes significant time since we are running $50 * 50 = 2500$ 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 in [7].

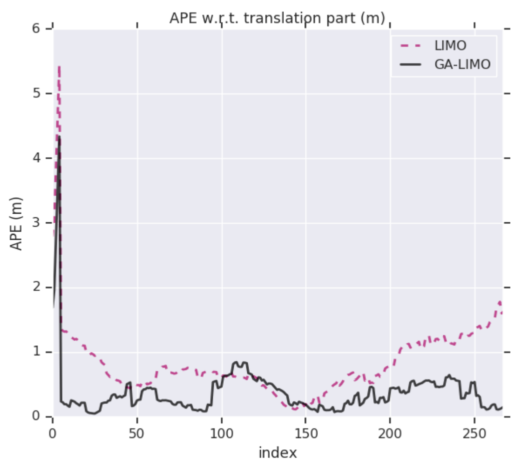
IV. EXPERIMENT AND RESULTS

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 trajectory, compared to the ground truth (reference) [33]. 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.

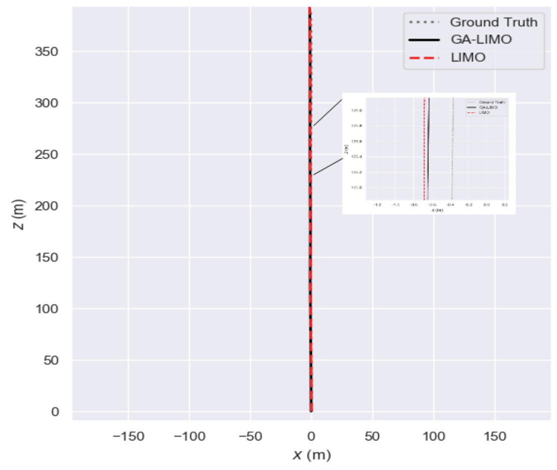
Algorithm 2 GA-LIMO individual

```
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 $\sigma$ 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 6 shows camera data while GA-LIMO is estimating the pose from that data in figure 7. Figure 2 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 [42]. The translation error for each sequence is the RMSE calculated with respect to ground truth. Figure 2a compares the translation error over the poses, while figure 2b compares the error mapped onto the trajectory with the zoomed in trajectory. Table I compares the values of parameters for LIMO and GA-LIMO. 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.

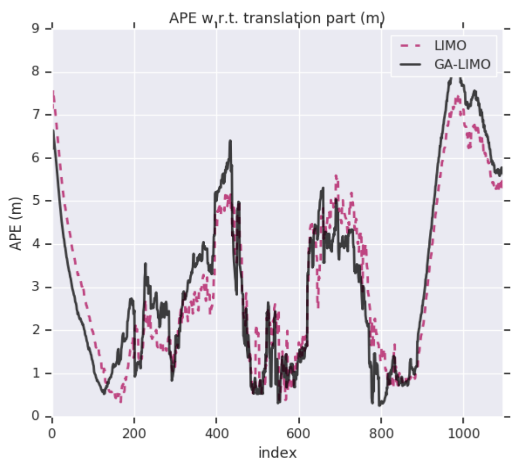


(a) Translation error comparison over the poses.

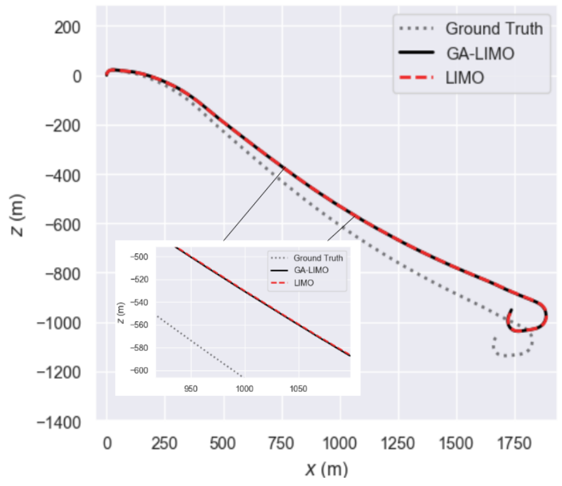


(b) Trajectory comparison for sequence 04, when GA-LIMO was run on this sequence individually (algorithm 2).

Fig. 2: 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.

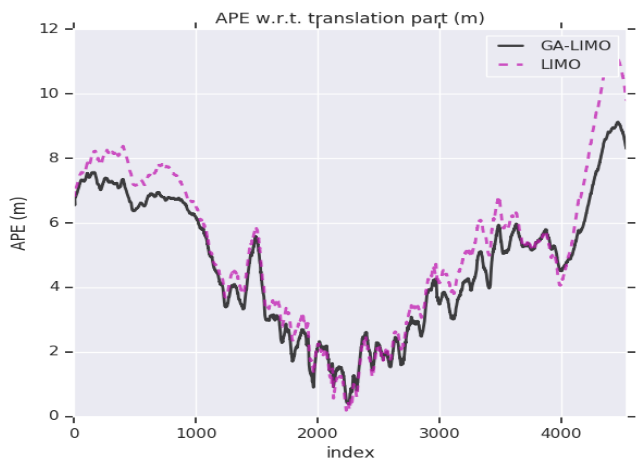
Fig. 3: Results comparison for sequence 01 (algorithm 2). LIMO has 3.71% translation error, while GA-LIMO has 3.8%.

Parameters	LIMO	GA-LIMO
δ	0.95	0.986
ϵ_{near}	400	999
ϵ_{middle}	400	960
ϵ_{far}	400	859
μ	0.9	0.128

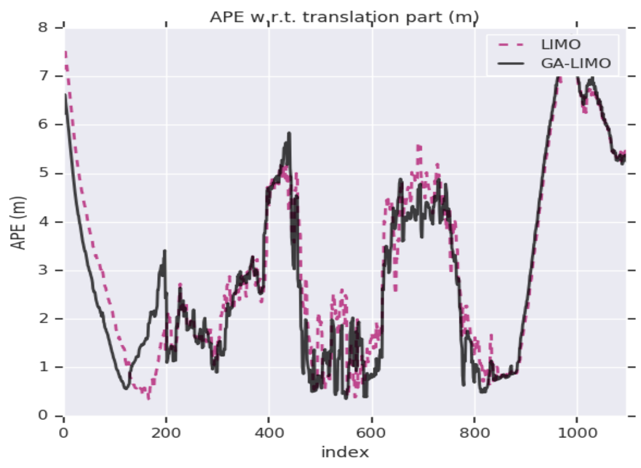
TABLE I: LIMO vs GA-LIMO values of parameters when GA was run on LIMO with sequence 04 individually.

Figure 3 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 II compares the original and GA found parameter values. Figure 3a compares translation error, while figure 3b shows the error mapped onto the trajectory for LIMO and GA-LIMO. As shown in the zoomed in figure 3b, GA-LIMO is closer to the ground truth. The translation error for LIMO is found to be around 3.71% and 3.8 % in case of GA-LIMO, with sequence 01. GA found parameters did not out perform the original parameters, when GA-LIMO was run on just sequence 01.

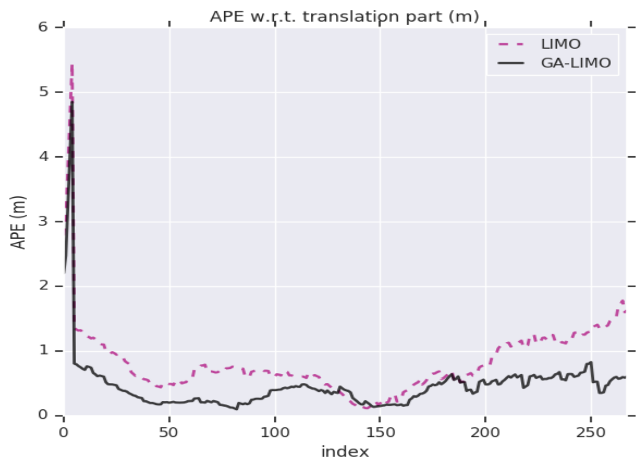
We finally ran the system with both sequences 01 and 04 as described in Algorithm 1. The fitness of each evaluation is the average of translation errors of the sequences when run using the input parameters. The parameters found in GA-LIMO as shown in table III, were then tested on sequences sequences 00, 01 and 04, as shown in figure 4 and 5. It is evident that GA-LIMO performed better than 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



(a) Translation error comparison over the poses for sequence 00. LIMO has 5.77% translation error while GA-LIMO has 5.13%.

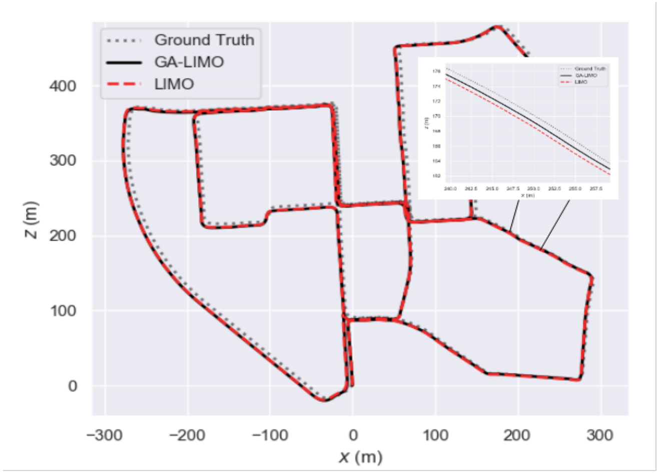


(b) Translation error comparison over the poses for sequence 01. LIMO has 3.71% translation error while GA-LIMO has 3.59%.

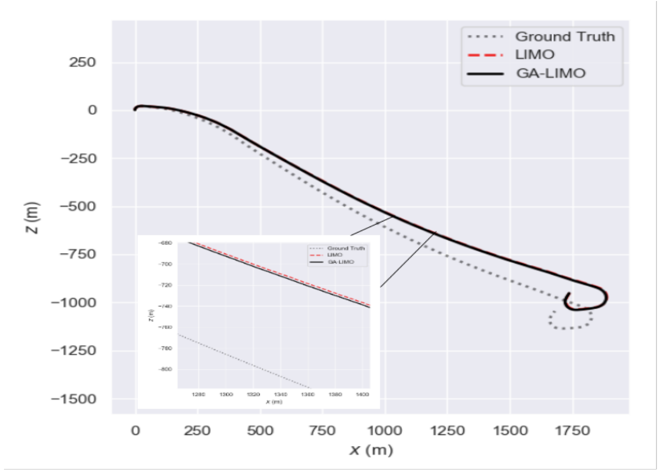


(c) Translation error comparison over the poses for sequence 04. LIMO has 1.01% translation error while GA-LIMO has 0.65%.

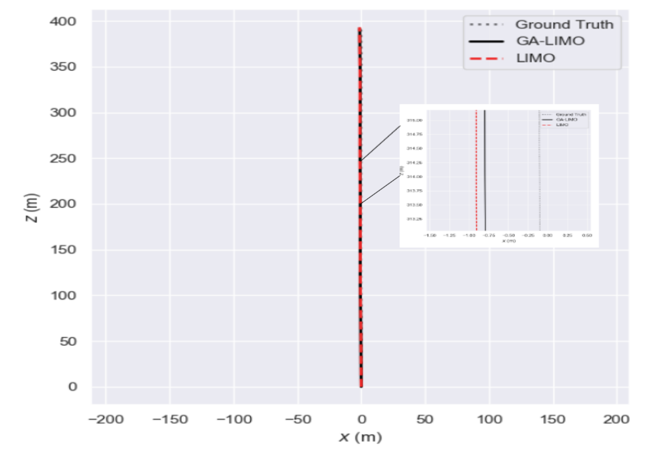
Fig. 4: The parameters are found using GA-LIMO using combination of sequences 01 and 04 (Algorithm 1). These parameters are then tested on three sequences. In all three sequences GA-LIMO perform better than LIMO.



(a) Sequence 00 trajectories showing GA-LIMO closer to ground truth.



(b) Sequence 01 trajectories showing GA-LIMO closer to ground truth.



(c) Sequence 04 trajectories showing GA-LIMO closer to ground truth.

Fig. 5: Trajectory comparison when GA-LIMO was run as in Algorithm 1. In all three sequences GA-LIMO perform better than LIMO.

Manuscript 527 submitted to 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). Received February 28, 2019.

Fig. 6: Camera data while GA-LIMO is in action.

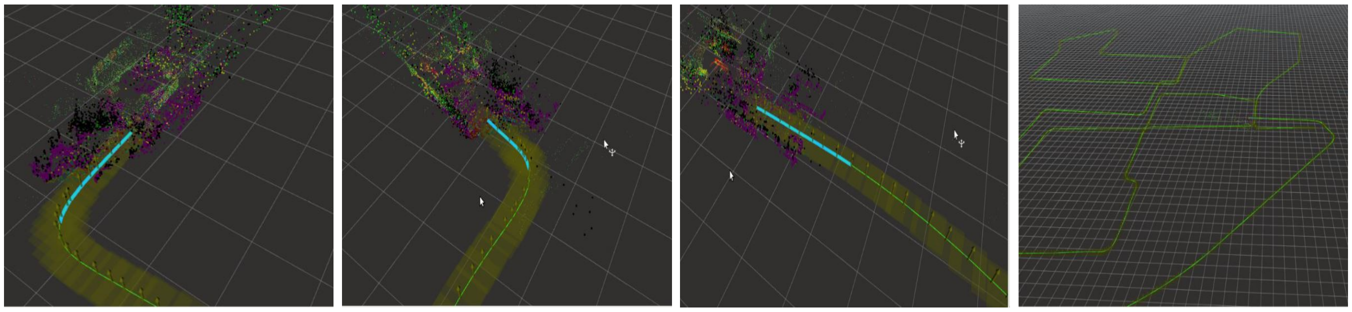


Fig. 7: GA-LIMO estimating the pose.

Parameters	LIMO	GA-LIMO
δ	0.95	0.958
ϵ_{near}	400	999
ϵ_{middle}	400	593
ϵ_{far}	400	877
μ	0.9	0.813

TABLE II: LIMO vs GA-LIMO values of parameters when GA was run on LIMO with sequence 01 individually.

Parameters	LIMO	GA-LIMO
δ	0.95	0.963
ϵ_{near}	400	999
ϵ_{middle}	400	554
ϵ_{far}	400	992
μ	0.9	0.971

TABLE III: LIMO vs GA-LIMO values of parameters when GA is run on LIMO with combined sequence 01 and 04.

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.

V. DISCUSSION AND FUTURE WORK

This paper shows results that demonstrated that the genetic algorithm 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.

APPENDIX

Open source code for this paper is available on github: *https://github.com/aralab-unr/LIMOWithGA*. The parameters used in this paper: 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, corresponds to *outlier_rejection_quantile*, *max_number_landmarks_near_bin*, *max_number_landmarks_middle_bin*, *max_number_landmarks_far_bin*, *shrubbery_weight*, respectively in the code.

REFERENCES

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

[2] T. Taketomi, H. Uchiyama, and S. Ikeda, "Visual slam algorithms: A survey from 2010 to 2016," *IPSJ Transactions on Computer Vision and Applications*, vol. 9, no. 1, p. 16, 2017.

[3] A. Singandhupe and H. La, "A review of slam techniques and security in autonomous driving," in *IEEE International Conference on Robotic Computing (IRC)*. Italy, 2019.

[4] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[5] R. Szeliski, *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.

[6] M. Sons, H. Lategahn, C. G. Keller, and C. Stiller, "Multi trajectory pose adjustment for life-long mapping," in *2015 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2015, pp. 901–906.

[7] J. Graeter, A. Wilczynski, and M. Lauer, "Limo: Lidar-monocular visual odometry," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 7872–7879.

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

[9] A. Geiger, J. Ziegler, and C. Stiller, "Stereoscan: Dense 3d reconstruction in real-time," in *2011 IEEE Intelligent Vehicles Symposium (IV)*. Ieee, 2011, pp. 963–968.

[10] I. Cvišic, J. Cesic, I. Markovic, and I. Petrovic, "Soft-slam: Computationally efficient stereo visual slam for autonomous uavs," *Journal of field robotics*, 2017.

[11] M. Buczko and V. Willert, "Flow-decoupled normalized reprojection error for visual odometry," in *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2016, pp. 1161–1167.

[12] I. Cvišić and I. Petrović, "Stereo odometry based on careful feature selection and tracking," in *2015 European Conference on Mobile Robots (ECMR)*. IEEE, 2015, pp. 1–6.

[13] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, 2013.

[14] I. Krešo and S. Šegvic, "Improving the egomotion estimation by correcting the calibration bias," in *10th International Conference on Computer Vision Theory and Applications*, 2015.

[15] A. Geiger, F. Moosmann, Ö. Car, and B. Schuster, "Automatic camera and range sensor calibration using a single shot," in *2012 IEEE International Conference on Robotics and Automation*. IEEE, 2012, pp. 3936–3943.

[16] J. Gräter, T. Strauss, and M. Lauer, "Photometric laser scanner to camera calibration for low resolution sensors," in *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2016, pp. 1552–1557.

[17] Y. Xu, P. Dong, J. Dong, and L. Qi, "Combining slam with muti-spectral photometric stereo for real-time dense 3d reconstruction," *arXiv preprint arXiv:1807.02294*, 2018.

[18] J. Zhang and S. Singh, "Visual-lidar odometry and mapping: Low-drift, robust, and fast," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 2174–2181.

[19] T. Caselitz, B. Steder, M. Ruhnke, and W. Burgard, "Monocular camera localization in 3d lidar maps," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 1926–1931.

[20] J. Zhang and S. Singh, "Loam: Lidar odometry and mapping in real-time." in *Robotics: Science and Systems*, vol. 2, 2014, p. 9.

[21] Y. Balazadegan Sarvrood, S. Hosseinyalamdary, and Y. Gao, "Visual-lidar odometry aided by reduced imu," *ISPRS International Journal of Geo-Information*, vol. 5, no. 1, p. 3, 2016.

[22] J. H. Holland *et al.*, *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press, 1992.

[23] D. E. Goldberg and J. H. Holland, "Genetic algorithms and machine learning," *Machine learning*, vol. 3, no. 2, pp. 95–99, 1988.

[24] M. Mitchell, *An introduction to genetic algorithms*. MIT press, 1998.

[25] S. Gibb, H. M. La, and S. Louis, "A genetic algorithm for convolutional network structure optimization for concrete crack detection," in *2018 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2018, pp. 1–8.

[26] A. Tavakkoli, A. Ambardekar, M. Nicolescu, and S. Louis, "A genetic approach to training support vector data descriptors for background modeling in video data," in *International Symposium on Visual Computing*. Springer, 2007, pp. 318–327.

[27] L. Davis, "Handbook of genetic algorithms," 1991.

[28] D. Kalyanmoy, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *IEEE Transactions on evolutionary computation*, vol. 6, no. 2, pp. 182–197, 2002.

[29] P. W. Poon and J. N. Carter, "Genetic algorithm crossover operators for ordering applications," *Computers & Operations Research*, vol. 22, no. 1, pp. 135–147, 1995.

[30] T. Duckett *et al.*, "A genetic algorithm for simultaneous localization and mapping," 2003.

[31] L. Moreno, J. M. Armingol, S. Garrido, A. De La Escalera, and M. A. Salichs, "A genetic algorithm for mobile robot localization using ultrasonic sensors," *Journal of Intelligent and Robotic Systems*, vol. 34, no. 2, pp. 135–154, 2002.

[32] A. Sehgal, H. La, S. Louis, and H. Nguyen, "Deep reinforcement learning using genetic algorithm for parameter optimization," in *IEEE International Conference on Robotic Computing (IRC)*. Italy, 2019.

[33] 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.

[34] J. Gräter, T. Schwarze, and M. Lauer, "Robust scale estimation for monocular visual odometry using structure from motion and vanishing points," in *2015 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2015, pp. 475–480.

[35] G. Nützi, S. Weiss, D. Scaramuzza, and R. Siegwart, "Fusion of imu and vision for absolute scale estimation in monocular slam," *Journal of intelligent & robotic systems*, vol. 61, no. 1-4, pp. 287–299, 2011.

[36] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 3213–3223.

[37] P. H. Torr and A. Zisserman, "Mlesac: A new robust estimator with application to estimating image geometry," *Computer vision and image understanding*, vol. 78, no. 1, pp. 138–156, 2000.

[38] P. H. Torr and A. W. Fitzgibbon, "Invariant fitting of two view geometry," *IEEE transactions on pattern analysis and machine intelligence*, vol. 26, no. 5, pp. 648–650, 2004.

[39] B. T. Polyak and A. B. Juditsky, "Acceleration of stochastic approximation by averaging," *SIAM Journal on Control and Optimization*, vol. 30, no. 4, pp. 838–855, 1992.

[40] D. E. Goldberg and K. Deb, "A comparative analysis of selection schemes used in genetic algorithms," in *Foundations of genetic algorithms*. Elsevier, 1991, vol. 1, pp. 69–93.

[41] G. Syswerda, "Uniform crossover in genetic algorithms," in *Proceedings of the third international conference on Genetic algorithms*. Morgan Kaufmann Publishers, 1989, pp. 2–9.

[42] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of rgb-d slam systems," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 573–580.

Manuscript 527 submitted to 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). Received February 28, 2019.