ISSN 2278-3091

Volume 8, No.1.6, 2019

International Journal of Advanced Trends in Computer Science and Engineering

Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse2581.62019.pdf

https://doi.org/10.30534/ijatcse/2019/2581.62019

A Genetic Algorithm with Online Learning Approach for Improving Loop Closure Detection of a Visual SLAM

Arif Haikal Ahmad Hassan Ayoppan[1], **Mohd Faisal Ibrahim***[1,2], **Mohd Hairi Mohd Zaman**[1,2]
[1]Department of Electrical, Electronic and Systems Engineering,
[2]Centre for Integrated Systems Engineering and Advanced Technologies (Integra),
Universiti Kebangsaan Malaysia, UKM Bangi, 43600, Selangor, Malaysia.
*Corresponding author: faisal.ibrahim@ukm.edu.my

ABSTRACT

This paper presents a genetic algorithm with online learning approach for improving loop closure detection of a visual simultaneous localisation and mapping (SLAM) technique. The reality gap issue in evolutionary robotics field is known as the main factor that degrades the quality of simulated solutions when transferring to a real robot. The proposed method can optimise the parameter of a visual SLAM in real-time. The aim is to evolve and search for the best Bayes filter parameters of the loop closure detection using online data gathered directly from a connected robot. A fitness function calculation utilising real-time images and robot motion is proposed to evaluate the performance of candidate solutions throughout the learning session. The experimental results show that SLAM with the optimised loop closure detection parameters outperforms SLAM with the default parameters for about 90% improvement.

Key words: Evolutionary robotics, genetic algorithm, loop closure detection, visual SLAM.

1. INTRODUCTION

Constructing a learning method for an evolutionary robotics application such as simultaneous localisation and mapping (SLAM) is a non-trivial task. This is due to the fact that the application of evolutionary computation algorithms in the robotics field are vast and there is no standard learning method for most applications.

This work aims to adopt an evolutionary computation (EC) algorithm known as genetic algorithm (GA) to optimise the performance of a visual SLAM by improving the loop closure detection component of the SLAM. This aim can be achieved by choosing either a virtual or physical based learning method. The former method, also known as offline method utilises simulation environment to find and test solutions. However, a common issue known as the reality gap cannot be simply ignored 1. The reality gap is a problem of maintaining the exact performance of solution found in the learning phase based on a simulated environment into the application phase of a real robot environment 1.

Comparatively, physical based or online learning method uses real robots to perform evolution. It has an advantage that a discovered solution can be applied directly in the application phase without comprising the quality of the solution. Taking into account this advantage, this paper presents a GA with online learning approach for improving loop closure detection of a visual SLAM known as RTAB-Map or real-time appearance based mapping 2.

Our proposed approach utilises online data gathered from a connected Turtlebot 2 robot to learn near-optimal parameters of RTAB-Map loop closure detection. Fitness evaluation is done via real-time loop closure count given depth images taken from a Kinect sensor while the robot performing motion actions.

The contribution of this work is two-fold. First, the utilization of a GA technique with real-time real-data fitness calculation for SLAM optimisation. Second, the generation of GA chromosome for RTAB-Map Bayes filter parameters configuration.

The next section discusses related works on evolutionary computation on SLAM. Section 3 describes the methodology of our proposed approach. Then, the experimental results and discussion are presented in Section 4 before concluded with a conclusion section in Section 5.

2. RELATED WORKS

The argument to choose either a virtual or physical learning method for evolving a robotic system has been debated since the early implementation of EC techniques in robotic area in

Research supported by The Ministry of Education Malaysia under the grant number FRGS/1/2017/TK04/UKM/02/10.

90's [4], [5]. EC is a family of algorithms in machine learning inspired by the concept of biological evolution process [6], [7]. The preference can be based on a proper system architecture selection considering three attributes that are type of learning method, evolution phase and components to evolve 7.
Some works choose offline evolution over online evolution because evaluation time (fitness calculation) can be sped up using a simulation model of an evolved robotic system 8. This is useful when the robotic system can be modelled using either first principle or sampled-data procedures. For an example, a work in 9 has solved a bi-objective nonlinear nonconvex and multimodal problem of a robotic gripper movement. Offline evolution is used to find seven parameters of link lengths and joint angles of the gripper based on force analysis and link geometry analysis.

Another reason for using offline evolution is to cater a complex environment. A deep belief network (DBN) controller has been proposed in 10 for mapless robot navigation. The DBN controller was trained using a hybrid EC and support vector machine (SVM) technique where datasets for model development were gathered from robot's sensors.

Although the abovementioned examples of virtual-based evolution work well in a simulated environment, this approach suffers from the reality gap problem. The performance of an optimal solution in a simulated environment degrades significantly when the solution is transferred to the real environment 11. Such condition happens due to imprecise or simplified simulation models such as neglecting sensory noise or coarse discretization of data. Experimental results have shown that the degradation of performance due to transferring process can be as high as 30% reduction 12.

On another hand, online evolution is chosen on some other works. Online evolution is not prone to the reality gap problem, however has difficulty in controlling evaluation time. Few attempts to control the evaluation time were proposed. First approach is to use time-sharing mechanism 13. The method cuts short evaluation time of bad individual while giving longer time for good individual. Another approach is based on parallel computing mechanism. A work in 14 presented four processors configuration to decompose the main evolution into four small series of co-evolution.

Online evolution can be extended for SLAM tasks in many ways. Evolution of SLAM can be considered as a complex optimisation problem containing few local minima and huge search space 15. Differential Evolution (DE) technique is used in 16 to develop a localization filter of a grid-based SLAM. Online evolution is applied to re-estimate the position of robot in two levels of processes i.e. local level and loop closure detection. Evolution Strategy (ES) is used in 17 to solve the problem of localisation for multiple robots and multiresolution maps settings. Experimental results show that the robots successfully updated their position while performing map generation in 50ms sampling interval. In contrast, virtual-based evolution of SLAM is proposed in 18 for miniature autonomous sensory agents that have limited onboard computational power. Non-dominated sorting GA (NSGA) is used to estimate the accuracy of a proposed map and the total energy consumption by the robots. This work differs from the above works on SLAM in that online evolution is chosen to improve the loop closure detection efficiency to ensure better SLAM performance.

3. METHODOLOGY

In this section, the detailed description of our proposed online GA to optimise the loop closure detection of RTAB-Map SLAM is presented. The next sub-section first explains Bayes filter used in detecting loop closure and parameters in the filter that will be tuned by GA. Then, the following sub-section describes our proposed GA method in detail.

3.1 Bayes Filter

RTAB-Map applies a Bayesian filter to estimate the probability of a new location matches any visited locations. The filter estimates the full posterior probability $\rho(S_t|L^t)$. S_t is a variable describing the state of all loop closure hypotheses at time t. Meanwhile L^t is the sequence of locations $L_{-1},\ldots,L_t$. in working memory (WM) and short-term memory (STM) of RTAB-Map used for real-time processing 19. The full posterior probability is calculated using (1) as follows:

$$\boldsymbol{p}(S_t|L^t) = \eta \underbrace{\boldsymbol{p}(L_t|S_t)}_{\text{Observation}} \underbrace{\sum_{i=-1}^{t_n} \underbrace{\boldsymbol{p}(S_t|S_{t-1}=i)}_{\text{Transition}} p(S_{t-1}=i|L^{t-1})}_{\text{Belief}} \tag{1}$$

where η is a normalisation term, L_t is the current location and $S_{t-1} = i$ is the probability that location L_{t-1} closes a loop with a past location L_i.

The attention of the process of evolution made by GA is the transition model $\boldsymbol{\rho}(S_t|\, S_{t-1} = i)$ in (1). The transition model is a part of prior probability calculation that is used to estimate the probability of L_t as a new location or a past location, given the state of loop closure hypotheses at t-1. While other parts in (1) can be calculated straightforward with other auxiliary equations, the transition model is defined heuristically based on a discretised Gaussian curve. The Gaussian curve is used to predict $\boldsymbol{\rho}(S_t=i|\, S_{t-1}=j)$ with $i,j\in[0;t_n]$, the probability that a loop closure occurred at t, given there is a loop closure on a neighbour location at t-1 for a maximum of 16 neighbour locations. Identifying the best Gaussian curve is typically based on trial-and-error approach. Thus, finding the best Gaussian curve's shape by using GA is promising to get near-optimal configuration. A default Gaussian curve set in RTAB-Map package as in Figure 1. In this figure, the highest probability is set at location 0 or j. Meanwhile, the probability of neighbour locations (j-16, …, j+16) decreases gradually approaching zero probability for the farthest neighbors. Getting the right shape for the above Gaussian curve can improve the prediction of loop closure detection.

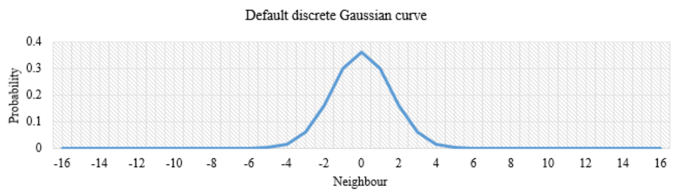


Figure 1: A default Gaussian curve used for transition model calculation

3.2 Proposed Online GA

GA is a subset of Evolutionary Algorithms (EA) which is based on Darwinian theory of natural selection 20 that increases an individual's ability to compete, survive and reproduce. Figure 2 shows an algorithm of the proposed GA with online or real-time fitness evaluation for each possible solution. For a GA, a possible solution is considered as an individual represented in the form of a chromosome.

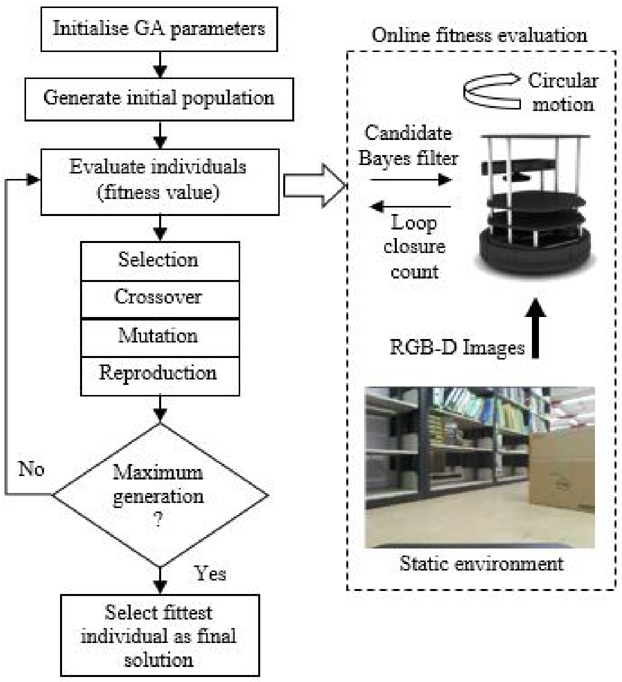


Figure 2: An algorithm of the proposed online GA to optimise loop closure parameters of RTAB-Map SLAM

The algorithm begins with the initialisation of GA parameters. These parameters determine the type and configuration of GA to be used. Table 1 describes the GA parameters.

Table 1: GA Parameters

GA Parameter	Value
Type of GA	Binary GA
Number of generations	10
Population size	3
Type of selection	Tournament
Type of crossover	2-Point Crossover
Probability of crossover, ρ_c	0.9
Type of mutation	Point mutation
Probability of mutation, ρ_m	0.01

After the initialisation, the GA processes start with the generation of the first population of chromosomes. In this work, a chromosome constitutes an array of variables that can be used to re-construct the shape of Gaussian curve explained in section 3.1. Note that, to construct the discretised Gaussian curve, the following array of numerical values, **N** in (2) are required to be set up in RTAB-Map.

$$\mathbf{N} = [VP, LC, N_1, N_2, N_3, ..., N_{16}] \tag{2}$$

where VP is the probability L_t is a new location given a loop closure was detected on the last iteration $\rho(S_t\!=\!-1|S_{t-1}\!=\!j)$. LC is the probability L_t closes a loop at j and N_1, N_2, N_3 *until* N_{16} are the 16 probability of loop closure at neighbours $j+1$. $j+2$, $j+3$ until $j+16$, respectively. Thus, a GA chromosome is configured such that the best numerical values in (2) except VP can be searched. VP is an exception since the graph concerns on $\rho(S_t\!=\!i|\,S_{t-1}\!=\!j)$ only. Here, VP is fixed with 0.1.

N_1 value should be lower than LC and N_2 value should be lower than N_1. This pattern continues and persists until N_{16}, thus encoding those probabilities directly into the chromosome is not practical. To ensure that this probability values decreases gradually and forms a valid discretised Gaussian curve, an array of intermediate variables **D** as in (3) is introduced. A variable in array **D** represents a relative distance value of probabilities between two adjacent neighbours. LC and array **D** are concatenated to form a chromosome structure as in Figure 3.

$$\mathbf{D} = [D_1, D_2, D_3, \ldots, D_{16}] \tag{3}$$

The relationship between a chromosome and Gaussian curve parameters can be linked by using (4) and (5).

$$N_1 = LC - D_1 \tag{4}$$
$$N_K = N_{K-1} - D_K \quad K\in[2;16] \tag{5}$$

Since Binary GA is chosen in this work, all variables in the chromosome were converted into their corresponding binary value before performing GA operations. Table 2 presents the value range and number of bits required for each variable.

After the conversion, each bit is considered as gene of the chromosome, thus a total of concatenated genes is 272 genes (16 bits x 17 variables).

The third process in the algorithm of Figure 2 is the evaluation of generated individuals or chromosomes. This is a crucial process in this work where an online fitness evaluation is proposed. A real robot situated in a static environment is connected to the GA learning algorithm. A Turtlebot2 robot with a Kinect camera and a Kobuki differential-drive plaform is used. Kinect camera captures RGB-D images with depth information as input to RTAB-Map. Meanwhile, Kobuki platform allows the robot to turn in-place to provide circular motion needed for the evaluation process.

Table 2: Configuration of Chromosome's Variables

Variable	Value Range		No. of Bits
	Min.	Max.	
LC	0.3	1.0	16 bit
D_1, D_2	0.0	0.3	16 bit
D_3, D_4	0.0	0.1	16 bit
D_5, D_6	0.0	0.05	16 bit
D_7, D_8	0.0	0.005	16 bit
D_9, D_{10}	0.0	0.0005	16 bit
D_{11}, D_{12}	0.0	0.00005	16 bit
D_{13}, D_{14}	0.0	0.000005	16 bit
D_{15}, D_{16}	0.0	0.0000005	16 it

LC	D_1	D_2	D_3	………………	D_{16}

Figure 3: Chromosome representation to construct a discretised Gaussian curve.

Figure 4 describes the detailed flowchart of online fitness evaluation to calculate fitness value for each chromosome. First, chromosome's variables were converted into Gaussian curve's parameters. RTAB-Map with the given setting of Bayes filter was executed to perform SLAM operation. Robot is pre-programmed to perform in-place rotation of 360^O for three times once RTAB-Map was established. Simultaneously, another program to monitor the number of loop closure count from RTAB-Map is executed. The time taken to complete the task was approximately 1 minute. After the task is completed, robot will stop and the final value of loop closure count is returned to the GA algorithm.

After all chromosomes have been evaluated, GA operations were performed in sequence – selection, crossover, mutation and reproduction. There are a few techniques of GA selection such as roulette wheel selection and tournament selection. Tournament selection is chosen in this work because it provides stronger selection pressure over the entire search procedure and prone to noise especially for small population size 21. Parents are selected based on comparison of fitness values of randomly selected chromosomes.

For GA crossover operation, a 2-Point crossover method with crossover probability, ρ_c set to 0.9 was selected. This method interchanges the head and tail of binary string from two selected parents and produces two new offspring as illustrated in Figure 5 22.

Next, GA mutation operation was performed to all produced offspring. Point mutation was used in which all gene's values are changed from 0 to 1 or vice versa with mutation probability, ρ_m of 0.01.

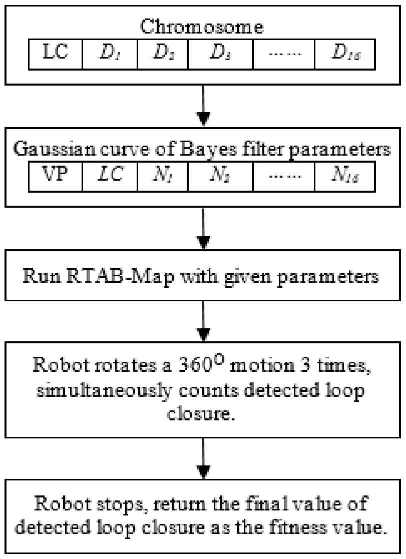

Chromosome
LC
D_1
D_2
D_3
……
D_{16}
Gaussian curve of Bayes filter parameters
VP
LC
N_1
N_2
……
N_{16}
Run RTAB-Map with given parameters
Robot rotates a 360^O motion 3 times, simultaneously counts detected loop closure.
Robot stops, return the final value of detected loop closure as the fitness value.

Figure 4: Flowchart of online fitness evaluation

Finally, a simple reproduction procedure was adopted where all produced offspring will form a new population and replace the current population to be fed into the next generation. Then, the online fitness evaluation function and GA operations were repeated until the maximum number of generations. In this work, the experiment was conducted using a personal computer with the specification of Intel i5 processor, 1.6 GHz clock and 8 GB RAM. ROS software 23 was used to establish the control platform of the robot as well as executing the RTAB-Map package. GAlib 24 was integrated to the ROS system to provide the proposed online GA algorithm. A complete cycle to run all generations took approximately 30 minutes.

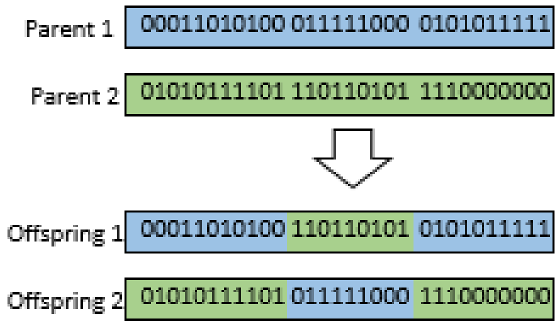


Figure 5: Example of 2-point crossover

4. RESULTS AND DISCUSSION

This section is divided into two sub-sections. The first sub-section discusses the results obtained during the learning phase and the later sub-section presents the results in the testing phase.

4.1 Learning Phase

RTAB-Map and online GA programs with parameters setting as in Table 1 were run simultaneously. Using ROS as the platform to integrate all functional programs, it provided robust data transfer via messages exchange within nodes with *publish-and-subscribe* method. Turtlebot2 was connected to the system to allow real data being captured by the programs. A static indoor environment of a laboratory at the Faculty of Engineering and Built Environment, Universiti Kebangsaan Malaysia as in Figure 6 was chosen for running the experiment.

Figure 7 shows the learning results after executing the abovementioned set up. At the initial generation, the best fitness found was only 17 loop closure detection events. The number increases gradually until generation 4 with 21 loop closure events. Then, no increment of the best fitness at generation 5. After that, the convergence continues until the final generation with the final best fitness is at 29 loop closure detection events.

Using RTAB-Map in ROS package, loop closure detection event is gathered by establishing the GA program to echo a topic called /rtabmap/info/LoopClosureId. Non-zero value returned by the topic will be summed up and stored after the robot has finished rotating. This summed value is the number of loop closure detection events that is directly converted as the fitness value of a chromosome.

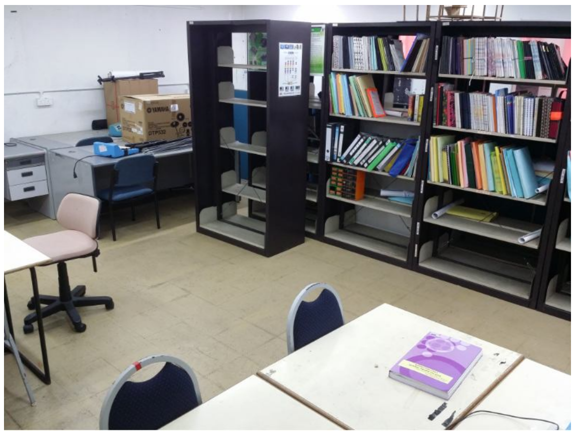


Figure 6: A static environment for GA learning phase

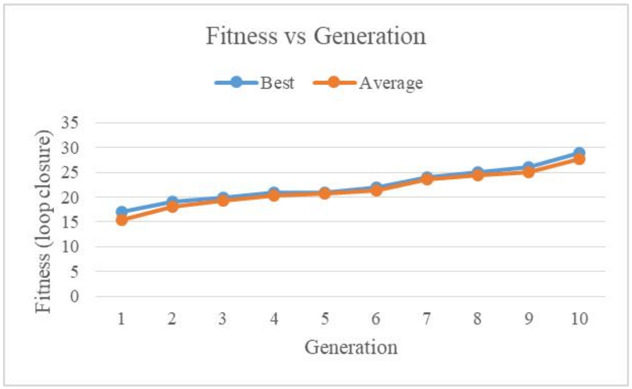


Figure 7: GA learning results presented the best and average fitness in each generation.

Figure 8 shows an example of loop closure detection event from the experiment. Frame 7 was the first image encountered by the robot. RTAB-Map uses bag-of-words approach to detect loop closure, thus visual features are extracted from every processed image. When Frame 18 comes in to the system, RTAB-Map detects there were few similar visual features between Frame 7 and Frame 18 (indicated by green lines in the figure). The threshold number of visual features is set at 8 for accepting a loop closure hypothesis. Based on this requirement, Frame 7 and Frame 18 has exceeded the minimum number of similar visual features, therefore the hypothesis is accepted.

4.2 Testing Phase

Next, we conducted an experiment to test and validate the evolved Bayes filter in another environment within the same laboratory. The setup is mostly similar in the learning phase except in a new environment. The robot was allowed to turn in-place (circular motion) within 1 minute.

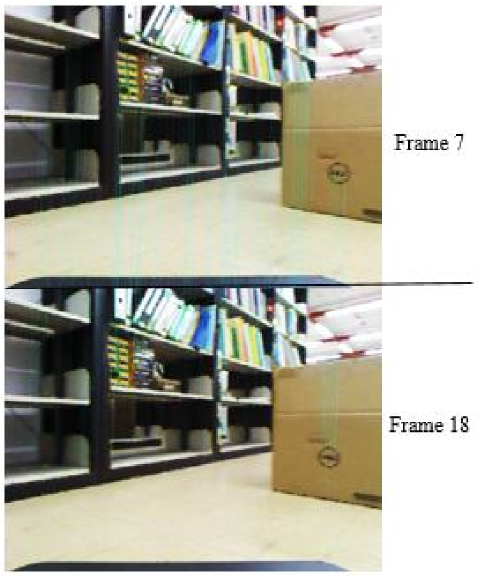


Figure 8: An example of loop closure detected on Frame 18 that matches Frame 7. The threshold number of visual features is set at 8 for accepting a loop closure hypothesis. The green lines show matched features and are more than 8, thus the hypothesis is accepted.

In this phase, the best evolved Bayes filter obtained in the learning phase was evaluated and compared with the default Bayes filter. The performance of both filters were measured in terms of the true number of loop closure detection within the same testing environment. A filter with higher number of true loop closure detection was considered as having better performance.

The parameters of Gaussian curve of the default Bayes filter is shown in Figure 1. Figure 9 presents the parameters of Gaussian curve of the evolved Bayes filter. Note that, the evolved curve suggests the probability value are given only to LC and N_1, meanwhile the probability for other neighbours are set to zero.

Each filter was evaluated to detect loop closure events by running three repeated experiment runs to get average performance. Figure 10 tabulates the results of the experiment runs for both filters. Numbers of loop closure were measured at the end of each run. Based on the graphs, the evolved Bayes filter outperformed the default Bayes filter in three test runs. Improvement of loop closure detection is consistent where the evolved Bayes filter found 26 to 28 loop closure events compared to the default Bayes filter that found only between 12 to 16 events on the same environment. Table 3 records the average loop closure performance for both filters. Note that, the performance improvement made by the evolved Bayes filter is approximately 90.5% from the default Bayes filter.

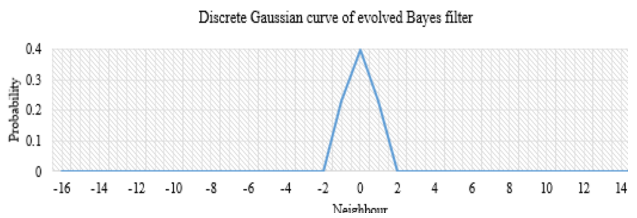


Figure 9: Evolved Bayes filter

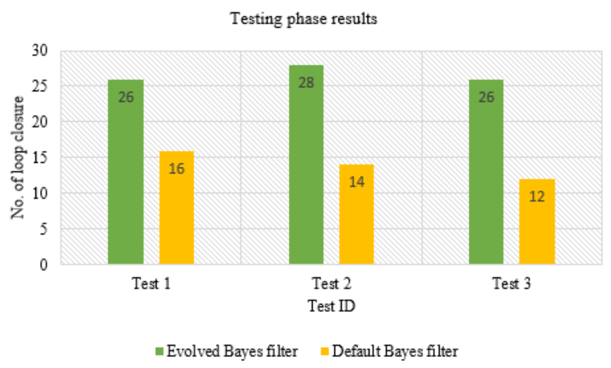


Figure 10: Testing phase results comparing the loop closure detection performance between the evolved Bayes filter and the default Bayes filter.

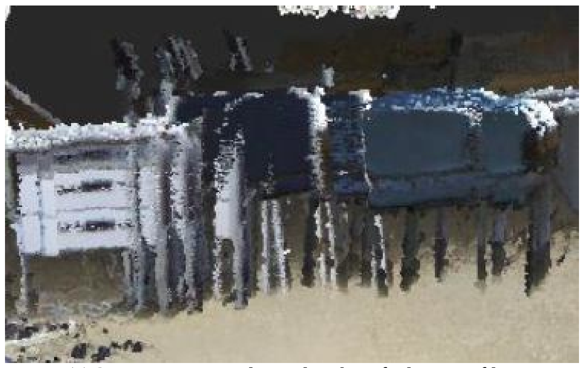
Table 3: Average Performance of the Evolved Bayes Filter and the Default Bayes Filter

Average no. of loop closure detection	Evolved Bayes filter	Default Bayes filter
	26.67	14.00

Validation of the performance can be observed qualitatively from one of RTAB-Map outputs that is from a three-dimensional (3D) occupancy grid map. The example of this visual output can be seen in Figure 11(a) for the default Bayes filter and Figure 11(b) for the evolved Bayes filter.

Using these maps for comparison, the most significant area was focused on three objects namely two chairs and one cabinet. It can be seen that the map generated by the default Bayes filter shows objects' duplication effect where those objects are blurred and appear to have more than three objects. This is because the loop closing process is not successful in the area so the robot recognized the place as a new area each time it passes through the same area. When a robot recognized the place as a different area, it re-engraved the image on the map, causing the entire map to see the object overlap between two or three images.

In contrast, the evolved Bayes filter produces a map with chairs and cabinet images as solid objects. This is an evidence that the loop closure process is successfully detected that allows map graphs to be updated correctly each time the robot passes the same area.


(a) 3D occupancy grid visual with Default Bayes filter

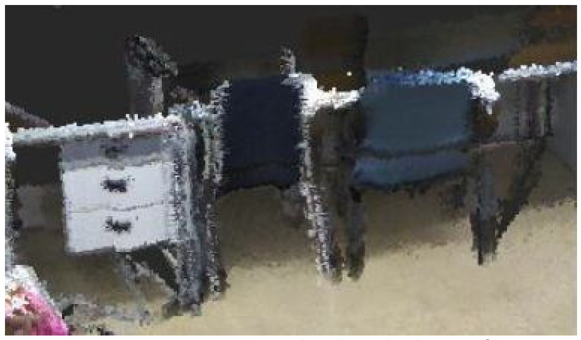

(b) 3D occupancy grid visual with Evolved Bayes filter

Figure 11: Visual observation of RTAB-Map 3D occupancy grid maps generated from (a) the default Bayes filter, and (b) the evolved Bayes filter.

5. CONCLUSION

Based on the results, it is concluded that changing the Bayes filter parameters, specifically the discrete Gaussian curve, changes the performance of RTAB-Map to detect the loop closure count. The proposed GA with online learning approach allows the parameters to be evolved in real-time using a real robot to minimize the reality gap problem. The experimental results shows that the evolved Bayes filter outperforms the default Bayes filter in terms of the number of loop closure count. Thus, RTAB-Map with the evolved Bayes filter produces accurate outputs such as a 3D occupancy grid map for the robot to utilise it for autonomous navigation or other related tasks.

ACKNOWLEDGEMENT

We would like to thank JKEES and Integra, FKAB, UKM for the facility and supports especially for the usage of Project laboratory and Control laboratory.

REFERENCES

1. J.C. Bongard. **Evolutionary Robotics**, *Communications of the ACM*, vol. 56(8), pp. 74-83, 2013. https://doi.org/10.1145/2492007.2493883
2. D.N. Hofstadler, M. Wahby, M. K. Heinrich, H. Hamann, P. Zahadat, P. Ayres and T. Schmickl. **Evolved Control of Natural Plants: Crossing the Reality Gap for User-Defined Steering of Growth and Motion**, *ACM Transactions on Autonomous and Adaptive Systems (TAAS) - Special Issue on SASO*, vol. 12(3), pp. 15:1 – 15:24, 2017.
3. M. Labbé and F. Michaud. **RTAB-Map as an Open-Source Lidar and Visual SLAM Library for Large-Scale and Long-Term Online Operation**, *Journal of Field Robotics*, vol. 36(2), pp. 416-446, 2018.
4. F. Silva, M. Duarte, S.M. Oliveira and A.L. Christensen. **Open Issues in Evolutionary Robotics**, *Journal of Evolutionary Computation*, vol. 24(2), pp. 205-236, 2016. https://doi.org/10.1162/EVCO_a_00172
5. P.A. Vargas,E.A. Di-Paolo, I. Harvey and P. Husbands. **Context and Challenges for Evolutionary Robotics**. in *The Horizons of Evolutionary Robotics*, MIT Press, 2014, ch. 1, pp. 1-16.
6. D.L.C. Guillermo, F.R.C. Dim, S.M.L. Fernando and L.Y.C. Young. **A study on how evolution simulator utilizes the windows operating system to demonstrate artificial intelligence learning**, *International Journal of Advanced Trends in Computer Science and Engineering*, vol.8(3): 764-771, 2019. https://doi.org/10.30534/ijatcse/2019/67832019
7. R.J. Alattas, S.Patel, T.M. Sobh. **Evolutionary Modular Robotics: Survey and Analysis**, *Journal of Intelligent & Robotic Systems*, vol. 95(3-4): 815–828, 2018. https://doi.org/10.1007/s10846-018-0902-9
8. M. Jelisavcic, M. de-Carlo, E. Hupkes, P. Eustratiadis, J. Orlowski, E. Haasdijk, J.E. Auerbach, A.E. Eiben. **Real-World Evolution of Robot Morphologies: A Proof of Concept**, *Artificial Life*, vol. 23(2), pp. 206-235, 2017.
9. J. Gomes, P. Mariano and A.L. Christensen. **Challenges in Cooperative Coevolution of Physically Heterogeneous Robot Teams**, Natural Computing, vol. 18(1), pp. 29-46, 2019. https://doi.org/10.1007/s11047-016-9582-1
10. R. Datta, S. Pradhan and B. Bhattacharya. **Analysis and Design Optimization of a Robotic Gripper Using Multiobjective Genetic Algorithm**, *IEEE Transactions on Systems, Man and Cybernetics*, vol. 46(1), pp. 16-26, 2016.
11. P. Jiang, C. Chen and X. Liu. **Time Series Prediction for Evolutions of Complex Systems: A Deep Learning Approach**, *in Proc. 2016 IEEE International Conference on Control and Robotics Engineering (ICCRE)*, 2016.
12. V. Trianni and M. López-Ibáñez. **Advantages of Task-Specific Multi-Objective Optimisation in Evolutionary Robotics**, PLoS ONE, vol. 10(8), pp. 2015.
13. K.Y.W. Scheper, S. Tijmons, C.C. de Visser and G.C.H.E de Croon. **Behavior Trees for Evolutionary Robotics**, *Journal of Artificial Life* vol. 22(1), pp. 23-48, 2016.
14. A.Q. Arif, D.G. Nedev and E. Haasdijk. **Controlling Evaluation Duration in On-Line, On-Board Evolutionary Robotics**, in *Proc. 2013 IEEE Conference on Evolving and Adaptive Intelligent Systems (EAIS)*, 2013, pp. 84-90. https://doi.org/10.1109/EAIS.2013.6604109
15. K.B. Lee, H. Myung and J.H. Kim. **Online Multiobjective Evolutionary Approach for Navigation of Humanoid Robots**, *Journal of IEEE Transactions on Industrial Electronics*, vol. 62(9), pp. 5586-5597, 2015.
16. M. Zemzami, N. Elhami, M. Itmi and N. Hmina. **An evolutionary hybrid algorithm for complex optimization problems**, *International Journal of Advanced Trends in Computer Science and Engineering*, vol. 8(2): 126-133, 2019. https://doi.org/10.30534/ijatcse/2019/05822019
17. L. Moreno, S. Garrido, F. Martín and M.L. Muñoz. **Differential evolution approach to the grid-based localization and mapping problem**, in *Proc. IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2007, pp. 3479-3484.
18. Y. Toda and N. Kubota. **Self-localization based on multiresolution map for remote control of multiple**

mobile robots, *IEEE Transactions on Industrial Informatics*, vol. 9(3), pp. 1772-1781, 2013.

19. S. Schlupkothen, A. Hallawa and G. Ascheid. **Evolutionary Algorithm Optimized Centralized Offline Localization and Mapping**, in *Proc. 2018 International Conference on Computing, Networking and Communications (ICNC)*, 2018, pp. 625-631.
20. M. Labbé and F. Michaud. **Appearance-Based Loop Closure Detection for Online Large-Scale and Long-Term Operation**, *IEEE Transactions on Robotics*, vol. 29(3), 2013, pp. 734-745.
21. A.M. Aibinu, H.B. Salau, N. Arthur, M.N. Nwohu and C.M. Akachukwu. **A Novel Clustering based Genetic Algorithm for Route Optimization**. *International Journal of Engineering Science and Technology*, vol. 19(4), pp. 2022–2034, 2016. https://doi.org/10.1016/j.jestch.2016.08.003
22. B. L. Miller and D. E. Goldberg. **Genetic Algorithms, Tournament Selection, and the Effects of Noise**, *Complex Systems*, vol.9, pp. 193–212, 1995.
23. L. Haldurai, T. Madhubala and R. Rajalakshmi. **A Study on Genetic Algorithm and its Applications**, *International Journal of Computer Sciences and Engineerin*, vol. 4(10), pp. 1-5. 2016.
24. Y. Pyo, H. Cho, R. Jung and T. Lim. ***ROS Robot Programming***. ROBOTIS Co. Ltd., Seoul, 2017.
25. Lancet.mit.edu. **GAlib: A C++ Library of Genetic Algorithm**. [Online]. Available: http://lancet.mit.edu/ga. [Accessed: 29 August 2019].