Proceedings of the International Conference on
Information Technologies (InfoTech-2017)
20-21 September 2017, Bulgaria

EVOLUTIONARY GRAPH-BASED SLAM

Ventseslav Shopov, Vanya Markova

Institute of Robotics – BAS
e-mails: vkshopov@yahoo.com
Bulgaria

Abstract: The simultaneous localization and mapping (SLAM) with evolutionary graph based approach is studied. The impact of re-sampling population on performance of evolutionary SLAM is experimentally studied. Two algorithms are compared: one with random re-sampling and one with greedy population re-sampling. With augmentation of number of nodes the greedy re-sampling crossing-over approach demonstrated significantly better performance.
Key words: SLAM, graph-based SLAM, evolutionary approach

1. INTRODUCTION

The simultaneous localization and mapping (SLAM) is a necessary for several robotic approaches in which the robot should move autonomously. The mobile robot needs a map of its environment for planning the appropriate paths to the its targets. In addition the robot should localizes itself on its map. Many modern SLAM methods follow the graph based approach [1, 2, 3, 4]. In this approach, each pose of the robot or each orientation position is represented as a node on the graph. The restriction between two nodes, which is obtained from observations, is represented by the edge on the graph. The first part of the common problem is to create a graph based on sensor data, and such a system is often referred to as an interface.

The second phase is devoted to the search for a node configuration, which best explains the limitations modelled by the edges. This step corresponds to the calculation of the most probable map (or distribution on possible maps), and the system that solves it is usually called the back-end.

In the community of geometric mapping, one of the main goals was to build massive maps, some of which covered even continents. It was assumed that such

kind of maps will be used either directly by people or to study the properties of the Earth.

The most popular method for solving this problem was EKM – Extended Kalman Filter. At each step, EKM has a set of previously received features (landmark) and the newly received data (ranging and RGB). Based on the new and previous frames, we can determine the displacement of the robot (using the methods of visual odometry) and predict the new position of the robot. On the other hand, from a new frame, we can highlight the location of the features and calculate the robot's position relative to them. Based on the difference between these two robot position estimates, the probabilities for all features are updated and the robotic position-trajectory is corrected.

Loop closure - loop detection. Separated from the tasks of SLAM is the question of how to monitor situations when a robot returns to where it has already visited (in the literature this is called a loop closure). One solution is the so-called basket of "words" (English Bags of Binary Words). Each frame is assigned a descriptor (BRIEF descriptor) [5], calculated on the basis of visual features of the image. To store information about images, based on the data for training, a dictionary is created in the form of a tree, containing "words" for representing descriptors and their weight (reflecting how often they met in the set of images for training). .

As mentioned above, at the present time the most popular representation of the SLAM problem is in the form of a graph, where the vertices and edges represent the position of the robot and the location of the features of the scene. ISam - one of the open implementations built on this principle [6]. ISam uses a bipartite graph consisting of node-poses containing the results of computing the robot coordinates and factor nodes containing the results of visual odometry estimates reflecting the shift between two consecutive poses. In addition, a number of feature nodes are used, containing the calculated coordinates of the features presented on the scene. Knot-poses and knots-features cannot be connected directly to each other, but only through knots-factors. As an extension, anchor nodes can be used-features observed by different robots (at the same time) or features that the same robot sees during several independent journeys.

We will study the impact of cross-breeding re-sampling on general performance. Our hypothesis is based on assumption that for random distributed the random cross-breading re-sampling should be good enough. On the other hand if there exist pattern in the environment we should expect that biased (greedy) re-sampling will benefit from such irregularity. In second part of the paper we outline the main issues of evolutionary graph-based SLAM approach. In the third part of the paper we will study the impact of greedy re-sampling on overall SLAM performance. As a case study we will use data from iSAM1 study for MIT Killian Court Dataset.

2. EVOLUTIONARY GRAPH BASED SLAM

To apply evolutionary approach to our problem, we introduces a graph using the coordinates of all vertices as chromosome. A layout is a list of two-dimensional Euclidean points. The length of the list is equal to the number of vertices n. Each of the points is composed of two genes that are represented as real numbers. Genes are initially generated randomly in a range depending on the weight of the column.

We used hybrid mutation to all results presented here. We call this mutation "hybrid" in connection with the fact that it is using a method based on the force in combination with classical genetic mutation.

Two cross-breeding approaches are used: a direct approach in which only randomly selected genes are crossed between children. The second method is a one point crossing-over in which from a pair of parents two children are constructed so that from randomly chosen gene the children swap all remaining genes.

Let G be an undirected connected graph, V the set of its vertices, E the set of edges; Of all vertices of n, and edges of m. We will store edges as adjacency lists. Let L_v be the set of vertices connected to v by an edge, L the set of all L_v. For each vertex v, we also introduce the set M_v, which contains unordered pairs of vertices from L_v. We denote by M the set of all M_v. Thus, if for all vertices v the vertices of L_v are divided into pairs in M_v, then the order of traversal is given on G up to the first edge: the pair (u, w) in L_v means that after coming from u, one must go to w (or On the contrary).

Let's consider an example.

Fitness function

The fitness function for the evolutionary algorithm for searching the Eulerian cycle in the graph looks like this:

F (M) = m - | M | + K, where

M is the number of edges in the graph;

| M | Is the size of the set M;

The mutation operation is introduced for two vertices u and w from L_v. How to choose them is described in the next section. It happens this way:

If u = w, then we do nothing;

If for u and for w there is no pair, then add a pair (u, w) to M_v;

If u and w are already contained in M_v as a pair, then delete it;

If u is already added in pair with some vertex p, and w does not have a pair, then remove (u, p) from M_v and add (u, w);

If w is already added in pair with some vertex p, and u does not have a pair, then remove (w, p) from M_v and add (u, w);

If u is already added in pair with some vertex p, and w is already added in pair with some k, then remove (u, p) and (w, k) from M_v and add (u, w) and (p, k);

Choice of vertices for mutation

Let d (v) be the degree of the vertex v (the number of edges that come out of it), d (G) the middle degree among the vertices of G the maximum degree among the vertices of G, and d (G). There are two ways to select the two vertices for the mutation:

EGA1 approach when we first, we randomly choose v from V. Then, randomly and independently choose u and w from L_v. after that we select randomly the vertex u from all 2m vertices in all lists L. Let it be in L_v. Then we randomly choose w from L_v. We choose randomly the pair (u, w) from all pairs for all 2m vertices in all lists in L. Let both vertices be present in L_v.

In **EGA2** approach we start with random selection of the v from V as in AGA1 approach. The second phase of EGA2 however differs from the second phase of EGA1: we use greedy approach to select the vertex u, in such way that E-total is minimal. This should guarantee faster conversion to the optimal solution. To avoid sticking in local minima on the final step we choose randomly the pair (u, w) from all pairs in the same manner as in EGA1 approach. These three cases are equivalent in the case of a sparse. In the general case the best result should be achieved by greedy method that is oriented to vertices.

3. EXPERIMERNT AND RESULTS

We used data from iSAM1 Results for MIT Killian Court Dataset [6] which contains 1941 poses, 2190 constraints, 14.2s or 7.3ms/step. The result of mapping is illustrated on fig 1.

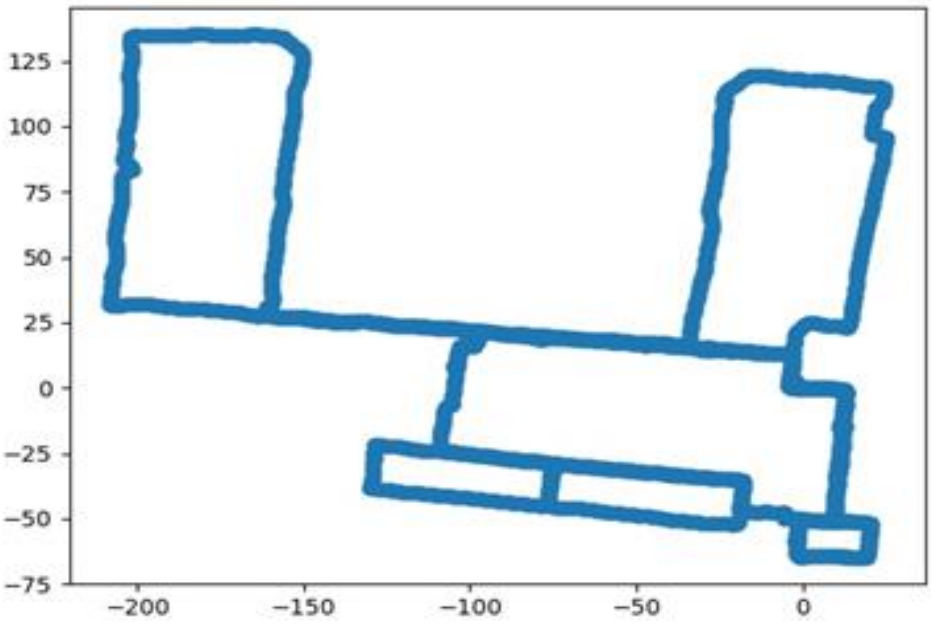

Fig. 1. SLAM mapping for MIT Killian Court Dataset

3.1. Impact of number of epochs on total error

We perform optimist ion of layouts for the data from iSAM1 data set. And apply two evolutionary approaches in order to study the impact of greedy initial strategy on overall performance.

The greedy approach is slightly better in the beginning but in time it outperform the random approach significantly.

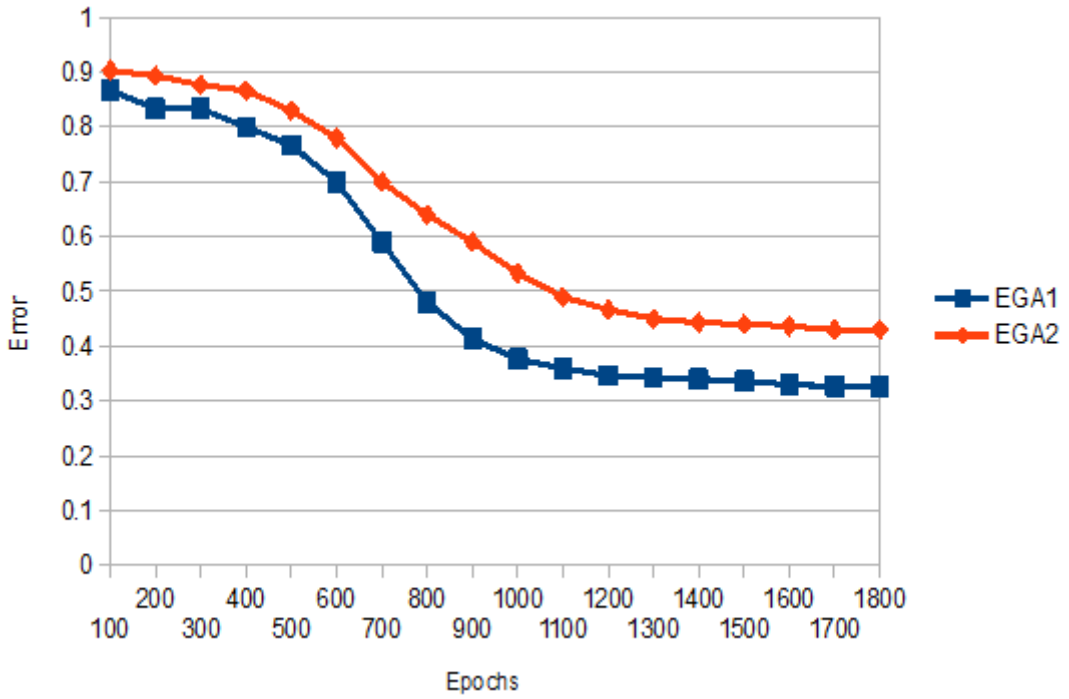

Fig. 2. Impact of number of epochs on total error

In case of very big minimal connected graphs the shorter learning period could be significant faster, so for big data sets should achieve better performance and converge much faster.

From the results we could conclude that greedy approach have significant advantage in regard of total error. However for small data sets the difference is small enough. In this case there are not clear winner. For smaller population sizes the greedy approach give more accurate layouts but the difference is insignificant..

4. CONCLUSION

We study the impact of number of epochs, number of nodes, and population size on total error E_total. We use two approaches: one classic evolutionary approach which is random generated and one greedy generated population.

When number of epochs and population size the choice of method is not straightforward. With augmentation of number of nodes the greedy crossing-over approach demonstrated significantly better performance.

In light of fact that some social network graphs have thousands of nodes the choice of proper method could significantly improve the performance of SLAM algorithms.

REFERENCES

[1 Lu F.and E. Milios, "Globally consistent range scan alignment for environment mapping," Autonomous Robots, vol. 4, 1997.

[2] Folkesson J, and H. I. Christensen, "Graphical SLAM - a self-correcting map," in Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), 2004.

[3]. Thrun S., and M. Montemerlo, "The GraphSLAM algorithm with applications to large-scale mapping of urban structures," International Journal of Robotics Research, vol. 25, no. 5-6, 2006.

[4] Grisetti G., R. K¨ummerle, C. Stachniss, and W. Burgard, "A tutorial on graph-based SLAM," IEEE Transactions on Intelligent Transportation Systems Magazine, vol. 2, 2010.

[5] Doerr B., C. Klein, and T. Storch. Faster evolutionary algorithms by superior graph representation. In Proc. of the 2007 IEEE Symposium on Foundations of Computational Intelligence (FOCI), 245–250 (2007).

[6] iSAM1 open source (LGPL) at: http://people.csail.mit.edu/kaess/isam