Article

Generative Cloud of Points SLAM (G-SLAM)

Nikos Zikos [1,*] and Vassilios Petridis [2]

[1] Department of Electrical and Computer Engineering, Aristotle University of Thessaloniki, Greece; nzikos@auth.gr
[2] Department of Electrical and Computer Engineering, Aristotle University of Thessaloniki, Greece; petridis@eng.auth.gr
* Correspondence: nzikos@auth.gr

Version February 1, 2019 submitted to Preprints

Abstract: Mobile robots need an environmental perception ability in order to interact with the surrounding environment. In this paper we present the G-SLAM method, where the map is consisted of a cloud of scattered points in the continuous space and each point is accompanied by an obstacle existence probability. On the other hand the robot's pose (and trajectory) is estimated by an particle filters while the cloud of points is estimated by an adaptive recursive algorithm which is presented. Main feature of the presented method is the adaptive repositioning of the scattered points and their convergence around obstacles. In addition to our previous publications, In this paper a mathematical derivation of the method is presented and real case scenarios are discussed, presented and compared with other methods.

Keywords: SLAM ; G-SLAM ; Particle filters, Monte Carlo

1. Introduction

In this paper it is presented the G-SLAM [1] method, a Simultaneous Localization and Mapping (SLAM) method, where the map is cloud of points in the continuous space followed by a probability which describes the existence of an obstacle in the subsequent point in space. In addition to our previous publication [1], in this paper it is presented the mathematical formulation and derivation of the problem and the experiments and the results are enhanced with more SLAM methods.

EKF-SLAM was extensively used in the past [2], but since computational cost increases significantly with the number of features new methods proposed to overcome this problem. Many probabilistic approaches have been proposed [3] including the Montemerlo's et al. solution to stochastic SLAM, FastSLAM 1.0 and 2.0 [4–7], Grid-based SLAM [8,9], Dual-FastSLAM [10], DP-SLAM [11], L-SLAM [12,13], etc.

Here we use a probabilistic approach which is based on particle filters. Particle filters are used for the robot's pose estimation. Usually an occupancy grid map with high resolution grid defined over the space but in this paper a cloud of scattered points in the continuous space is used instead. The map update procedure uses a recursive algorithm and produces the map's probability distribution . G-SLAM method generates new hypothetical points of features in space which are subsequently tested whether they correspond to real obstacles or not.

The G-SLAM's main advantage is the adaptive repositioning of the scattered map's points that "drives" them in a convergence around the obstacles. The final map resulted from the G-SLAM method exhibits high density of weighted points around the obstacle and a subsequent high sparsity in the free space. The resulted weighted points represent the probability distribution of the obstacles in the continuous space. This method fits best on problems where detailed maps are necessary but with low computational resources.

In section 2 we discuss the probabilistic analysis of the combined SLAM problem in terms of recursively computed probability distributions which estimates the probabilistic map and the robot's trajectory along with the G-SLAM method are presented. In section 3.1 it is shown the model of the robot used which consist of the robot's kinematic model and the laser sensor's measurement model is

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 1 February 2019 doi:10.20944/preprints201902.0003.v1

described. Section 3.3 exhibits the experimental results from real case scenario in a variety of methods including G-SLAM.

2. SLAM problem definition

2.1. Notations

- s^t: is a time series of the robot's pose, while s_t is only the pose at a time instance.
- Θ: represents the map and is a set of points θ^k in space.
- z^t: is a time series of the measurements, while z_t represents only the measure at time t.
- u^t is the time series of the robot's control inputs, while u_t refers only at time t control input.
- $f(.)$: is the robot's kinematic model.
- $g(.)$: represents the sensor's measurement model.
- $d_z(.)$: represents the probability density function of the sensor's measure noise.
- $d_f(.)$: represents the probability density function of the transition's model noise.

2.2. SLAM Posterior

While most SLAM methods are trying to estimate the robot's pose s_t and the map Θ at timestamp t, in this paper our goal is to estimate the whole time series s^t and the map Θ using the observation time series z^t and the control time series u^t. In probabilistic terms this posterior is expressed as:

$$Prob(s^t, \Theta | u^t, z^t) \tag{1}$$

Using the definition of the conditional probability, the posterior in equation 1 can be expressed as:

$$Prob(s^t, \Theta | u^t, z^t) = \underbrace{Prob(s^t | z^t, u^t)}_{\text{trajectory posterior}} \underbrace{Prob(\Theta | s^t, z^t, u^t)}_{\text{map posterior}} \tag{2}$$

The two factors of the equation 2 correspond to the robot's trajectory posterior and the map's posterior respectively. The calculation of these two factors is discussed bellow.

2.3. Pose prediction

The left posterior of the equation 2 refers to the estimation of the pose time series given the map and the time series of the observation and the controls.

The calculation of this posterior is done using the technique of particle filtering. The proposal distribution for the particles will be the posterior $p(s^t | z^{t-1}, u^t)$, thus the drawing process for each particle i evolves only the previous state s^i_{t-1} and the current control input u_t.

$$s^i_t \sim p(s_t | s^i_{t-1}, u_t) \tag{3}$$

The proposal distribution is generated from the posterior $Prob(s_t | s_{t-1}, u_t)$ using the robot's kinematic model f and of course a random sample of the control's input noise ϵ^i_t.

$$s^i_t = f(s^i_{t-1}, u_t, \epsilon^i_t) \tag{4}$$

This procedure creates a cloud of N particles, all representing a possible pose of the robot. It is noteworthy that each particle carries out its own estimation of the map which is independent from the other particles' maps. The estimation of the pose timeseries is not done yet since particle filter's importance factors have to be calculated. The calculation of the importance factors is discussed below in the section 2.5.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 1 February 2019 doi:10.20944/preprints201902.0003.v1

2.4. Map Update

The rightmost factor of the equation 2 refers to the estimation of the map given the time series of the observation and the controls. The map consists of a set of scattered points in space θ and each one is associated with a probability that the point θ is an obstacle. The distribution of this probabilistic map can be represented by the following posterior.

$$p_t^k = Prob(\theta_k | s^t, u^t, z^t) \tag{5}$$

The equation 5 gives the probability that the feature θ_k is an obstacle given the observations z^t and the controls u^t. By the definition of the conditional probability, the posterior 5 is expressed as:

$$p_t^k = \frac{Prob(z_t, \theta_k | s^t, u^t, z^{t-1})}{Prob(z_t | s^t, u^t, z^{t-1})} \tag{6}$$

Using the law of total probability for all θ_j the denominator becomes

$$p_t^k = \frac{Prob(z_t, \theta_k | s^t, u^t, z^{t-1})}{\sum_j Prob(z_t, \theta_j | s^t, u^t, z^{t-1})} \tag{7}$$

The posteriors of the numerator and the denominator have the same form

$$Prob(z_t, \theta_k | s^t, u^t, z^{t-1}) = Prob(z_t | s^t, u^t, z^{t-1}, \theta_k) Prob(\theta_k | s^t, u^t, z^{t-1})$$

In the rightmost posterior we note that θ_k is independent of the control input u_t and the pose s_t due to the absence of the measurement z_t. Thus the above expression becomes

$$Prob(z_t | s^t, u^t, z^{t-1}, \theta_k) Prob(\theta_k | s^{t-1}, u^{t-1}, z^{t-1}) = Prob(z_t | s^t, u^t, z^{t-1}, \theta_k) p_{t-1}^k \tag{8}$$

Equations (7) and (8) imply the recursion:

$$p_t^k = \frac{Prob(z_t | s^t, u^t, z^{t-1}, \theta_k) p_{t-1}^k}{\sum_j Prob(z_t | s^t, u^t, z^{t-1}, \theta_j) p_{t-1}^j} \tag{9}$$

Since z_t is independent from previous observations, control inputs and previous robot's positions the equation (9) is simplified as:

$$p_t^k = \frac{Prob(z_t | s_t, \theta_k) p_{t-1}^k}{\sum_j Prob(z_t | s_t, \theta_j) p_{t-1}^j} \tag{10}$$

In order to compute the recursion (10) we need to calculate the quantity $q_t^k = Prob(z_t | s_t, \theta_k)$. In case that the probability distribution function of the measurement noise is given by function $d_z(z)$, then this quantity can be calculated by:

$$q_t^k = d_z \left(g(s_t, \theta_k) \right) \tag{11}$$

Combining equations (10) and (11) the probability of every point k is calculated using equation (12).

$$p_t^k = \frac{q_t^k p_{t-1}^k}{\sum_j q_t^j p_{t-1}^j} \tag{12}$$

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 1 February 2019 doi:10.20944/preprints201902.0003.v1

2.5. Importance factor calculation

The distribution as proposed in equation (3) is only the proposal distribution. Using the simulation technique of particle filtering the target distribution is calculated as:

$$\text{target distribution} = \text{proposal distribution} * \text{importance factor}$$

Through the target distribution the best estimation for the robot's pose s_t is calculated.

$$w_t^i = \frac{\text{target distribution}}{\text{proposal distribution}} = \frac{p(s^{t,i}|z^t, u^t)}{p(s^{t,i}|z^{t-1}, u^t)} \tag{13}$$

Using the Bayes Theorem the equation (13) is simplified as:

$$w_t^i \propto \frac{p(s^{t,i}|z^{t-1}, u^t)p(z_t|s^{t,i}, u^t, z^{t-1})}{p(s^{t,i}|z^{t-1}, u^t)} \tag{14}$$

$$w_t^i \propto p(z_t|s^{t,i}, u^t, z^{t-1}) \tag{15}$$

So the importance factor is proportional to the posterior $p(z_t|s^{t,i}, u^t, z^{t-1})$ which is already calculated in the map update section 2.4 as the denominator of equations (6) or (10).

$$w_t^i \propto \sum_j Prob(z_t|s_{t,i}, \theta_{j,i})p_{t-1}^{j,i} \tag{16}$$

Since the set of particles $S^t = \{s_1^t, ..., s_M^t\}$ is finite, the "cloud" of particles is growing as the time increases, which can lead to the degeneracy of the algorithm. Thus a resampling technique is necessary. In this paper and on the experiments that took place, the technique of Residual Systematic Resampling (RSR) is used [14].

2.6. The G-SLAM Method

The proposed method, G-SLAM is based on a technique that generates stochastically new scattered points that are added into the map. This stochastic generation is based on the current particle and the current observation. Then the update procedure updates the map by updating each point's probability, while afterwards the "meaningless" features are removed from the map set. The sensor's noise is responsible for the stochastic nature of this procedure. This addition of scattered point into the map set is achieved using a drawing procedure which is described bellow. Afterwards the extended map is updated and the updated points with low probabilities are removed. The small probability in a map point, states that this point in space is unlikely to be an obstacle. Algorithms of this type converges as are discussed in [15]. In the context of this paper, map update procedure converges to high probabilities in map points near obstacles. Removing all low probability map points, the parts of space which are free of obstacles are also free of map points while on the other hand the parts of space with obstacles gathers all the scattered points around them.

The G-SLAM method can be described abstractly in six steps:

1. Draw pose s_t^i for every particle i using the subsequent pose s_{t-1}^i and the control u_t
2. Generate and add new map points θ into the particle's i map set Θ^i using drawing process based on the observation z_t and pose s_t^i
3. Update map by calculating the probabilities of all map points
4. Remove the map points with low probabilities
5. Calculate Importance factors for every particle
6. Resample particles if necessary

Steps 1,3,5,6 are already discussed in sections 2.3,2.4 and 2.5, while generation and removal of map points are discussed bellow.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 1 February 2019 doi:10.20944/preprints201902.0003.v1

2.6.1. Updating existing map points

The existing map can be easily updated using equation (12). For every map point θ_i it is calculated the probability of the measure z_t to correspond to this point in map using the equation (11) and then it is multiplied to the previous map's point probability p^i_{t-1}. It is noteworthy that the denominator of the equation (12) is just a normalization factor.

2.6.2. Adding new map points

The stochastic addition of new points into the map is achieved based on the observation z_t and the current pose of each particle. For every observation z_t a drawing procedure takes place in order to generate a set of M new map points that represents the sensor's measurement probability distribution.

$$\hat{z}^m_t \sim \mathcal{N}(z_t; z_t, R_t) \tag{17}$$

where R_t is the covariance matrix of the sensor's noise.

Every element $\hat{z}^m_t$ is given a probability:

$$\hat{q}^m = d_z(\hat{z}^m_t) \tag{18}$$

where $d_z(.)$ represents the probability density function of the measurement's noise.

These elements $\hat{z}^m_t$ are unlikely to correspond to a map point $\theta \in \Theta$. Thus in order to update the map it is necessary to create new map points $\hat{\theta}_m = g(s_t, \hat{z}^m_t)$ in the map Θ. But also, in order to proceed with the update, it is necessary to calculate each point's probability $\hat{p}^m_{t-1}$. In the G-SLAM method the calculation of the probability $\hat{p}^m_{t-1}$ is done numerically using interpolation methods. The pre-updated map contains points and their subsequent probabilities at time $t-1$. These points in space are interpolated with $\hat{\theta}_m$ in order to estimate the subsequent $\hat{p}^m_{t-1}$ probability. Afterwards and using the equation (12) the new map points are updated and added to the map set Θ. This procedure augments the probabilistic map with M new points and their probabilities.

2.6.3. Removing meaningless map points

As already discussed, update procedure returns an augmented map with more map points than previously had. Some of those points might have near zero probability meaning that the probability of an obstacle existence in this point in space is highly unlikely.

In G-SLAM a map point removal procedure takes place after map update procedure in order to remove all the meaningless map points. So all points θ_i which their probabilities p^i_t are less than a predefined thresshold $p^i_t < p_{thr}$ are removed from the map set. Using this technique it is prevented the overpopulation of the set Θ and the features θ tends to gather near obstacles.

2.7. Agorithm

A pseudo code of the G-SLAM algorithm is given below.

function G-SLAM($s_{t-1}, z_t, u_t, \Theta$)
  for $i = 1 : N$ particles **do**
    draw $s^i_t \sim f(s^i_{t-1}, u_t)$ ▷ Drawing proccess
    for k=1:all $\theta \in \Theta$ **do** ▷ Existing map update
      $q^k_t = d_z(g(s^i_t, \theta_k))$
      $p^k_t = \frac{q^k_t p^k_{t-1}}{\sum_j q^j_t p^j_{t-1}}$
    end for
    for m=1:M **do** ▷ Adding new map points
      $\hat{z}^m_t \sim \mathcal{N}(z_t; z_t, R_t)$
      $\hat{q}^m = d_z(\hat{z}^m_t)$
      $\hat{\theta}_m = g(s^i_t, \hat{z}^m_t)$
      p^m_{t-1} interpolate using Θ and $\hat{\theta}_m$
      $p^m_t = \frac{q^m_t p^m_{t-1}}{\sum_j q^j_t p^j_{t-1}}$

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 1 February 2019 doi:10.20944/preprints201902.0003.v1

Add $\hat{\theta}_m$ and p_t^m in Θ
end for
$w_i = \sum_j q_t^j p_{t-1}^j$ ▷ Importance Factor Caclulation
Remove θ_k where $p_t^k < p_{thr}$ ▷ Meaninless point removal
end for
Resample particles
end function

3. Experiments & Results

To confirm our method's integrity we use the dataset performed by Nieto, Nebot et al. from the University of Sydney [16,17]. The experiments were performed with a car performing a full loop (fig. 2). A four-wheel rear-drive car was used, which was equipped with a horizontal scanning laser sensor with 180 degrees field of view and 80 meters of maximum observing radius. The control of the car consists of the linear velocity of the rear left wheel and the heading angle of the front wheels. GPS measurements comes with the dataset in order to validate the car's position.

3.1. System description

This dataset is a two dimensional planar dataset and the map is considered as a set of map point $\Theta = [\theta_1, \theta_2, ..., \theta_N]$ each one corresponding to a point in space (fig. 1). The probability that a point θ_k is an obstacle is denoted by p^k. The set of features along with their probabilities is a probabilistic map of the space. The robot's path is represented by a time-series of it's pose $s^t = [s_1, s_2, ...s_t]$. In a planar problem each feature θ_k is a vector with entries (x, y) coordinates of the point. Robot's pose s_t is also a vector with entries (x, y, φ) at time t where φ represents the angle of the robot's orientation corresponding to a global axis system.

The measurement timeseries $z^t = [z_1, z_2, ...z_t]$ consist of distance bearing measures (d, ϑ) acquired from the laser sensor and corresponds to the sensor's coordinate system. The distance-bearing laser sensor's feedback consists of a 361 distance measurements with half a degree angular distance between them.

3.2. Model

The car used for this experiments was a rear drive car-like four-wheel. The kinematic model of a vehicle like this is described by equation (19).

$$\begin{bmatrix} p_{x,t+1} \\ p_{y,t+1} \\ \varphi_{t+1} \end{bmatrix} = \begin{bmatrix} p_{x,t} + (v_c \cos(\varphi_t) - (a \sin(\varphi_t) + b \cos(\varphi_t)) \frac{v_c}{L} \tan(\omega)) \Delta t \\ p_{y,t} + (v_c \sin(\varphi_t) + (a \cos(\varphi_t) - b \sin(\varphi_t)) \frac{v_c}{L} \tan(\omega)) \Delta t \\ \varphi_t + \frac{v_t \Delta t}{L} \tan(\omega) \end{bmatrix} \tag{19}$$

where v_c is the robot's linear velocity at time t, ω is the steering angle at time t, L is the distance between the car's axles and a, b are the coordinates of the laser sensor according to the car's coordinate system. The velocity v_c is a function of the rear wheel's linear velocity and depends on the steering angle.

$$v_c = \frac{v_e}{1 - \tan(\omega) \frac{H}{L}}$$

where H is the distance between the center point of the rear axle and the rear wheel.

The measurement model of a distance-bearing sensor is given by the nonlinear equations:

$$\begin{aligned} z_t =& g(s_t, \theta_n) = \\ =& \begin{bmatrix} \sqrt{(p_{x,t} - \vartheta_{x,n})^2 + (p_{y,t} - \vartheta_{y,n})^2} \\ \arctan(\frac{p_{y,t} - \vartheta_{y,n}}{p_{x,t} - \vartheta_{x,n}}) - \varphi_t \end{bmatrix} \end{aligned} \tag{20}$$

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 1 February 2019 doi:10.20944/preprints201902.0003.v1

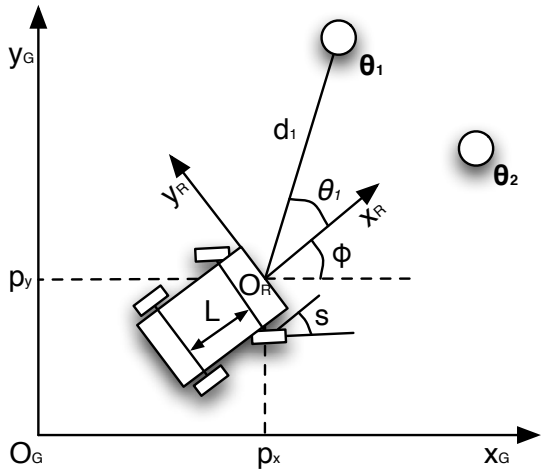

Figure 1. The robot coordinate system and the position of θ_i landmark with respect to the robot coordinate system O_R. θ is the angle between the vectors X_R and $O_R\theta$

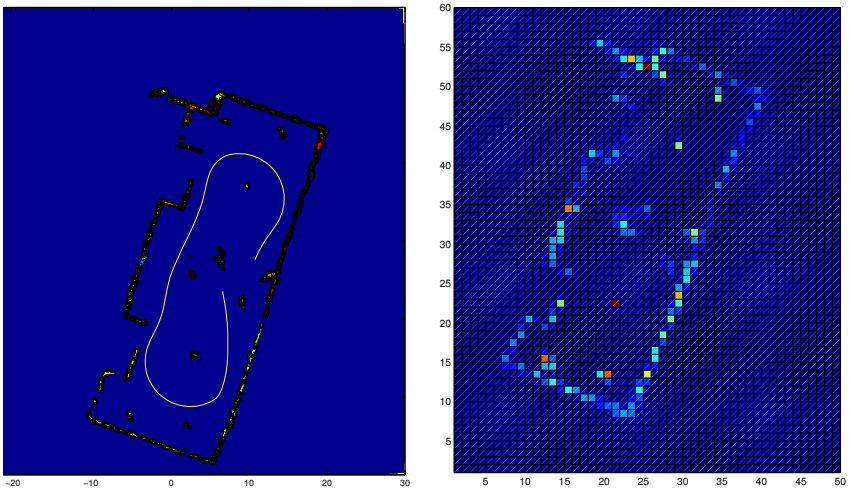

Figure 2. a. Contour graph of the map's probability distribution using G-SLAM. The yellow line represents the car's path estimation. b. The map represented using Grid based SLAM. The resolution of the grid (3000 cells) was chosen to match the G-SLAM's population (3280 map points)

It is assumed that the distance-bearing sensor's measurements and the control measurements are noisy with noise functions of a known probability distributions.

3.3. Results

The vehicle's kinematic model is described by equation 19 with parameters: $L = 2.75$, $H = 0.74$, $a = L + 0.5$ and $b = 0.5$.

As it is presented in the figures the algorithm results a probabilistic map that consists of a cloud of points and their probabilities. In figure 2 it is shown a contour graph with the map's probability distribution in contrast with the Grid Occupancy SLAM with almost the same number of map points. The yellow line represents the best estimation for the car's path. G-SLAM method resulted 3280 map points all of them gathered around obstacles, while Grid occupancy SLAM with 3000 cells resulted a lower resolution map since most of the cells covers a free area.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 1 February 2019 doi:10.20944/preprints201902.0003.v1

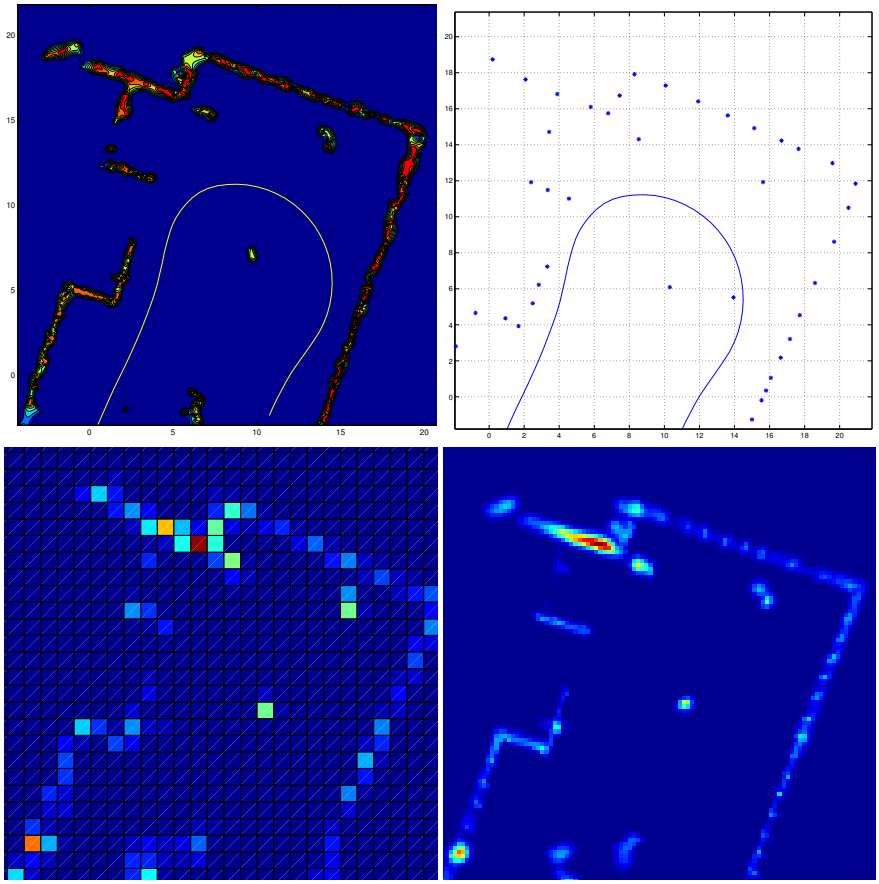

Figure 3. A detailed contour graph of the map's probability distribution in contrast with the FastSLAM 2.0 and Grid based SLAM with low and high resolution. Colors blue to red corresponds to low to high probability. The yellow line represents the car's path estimation. a. G-SLAM with 3280 map points, b. FastSLAM 2.0, c. Grid SLAM with 3000 cells, d. High resolution Grid SLAM with 48000 cells.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 1 February 2019 doi:10.20944/preprints201902.0003.v1

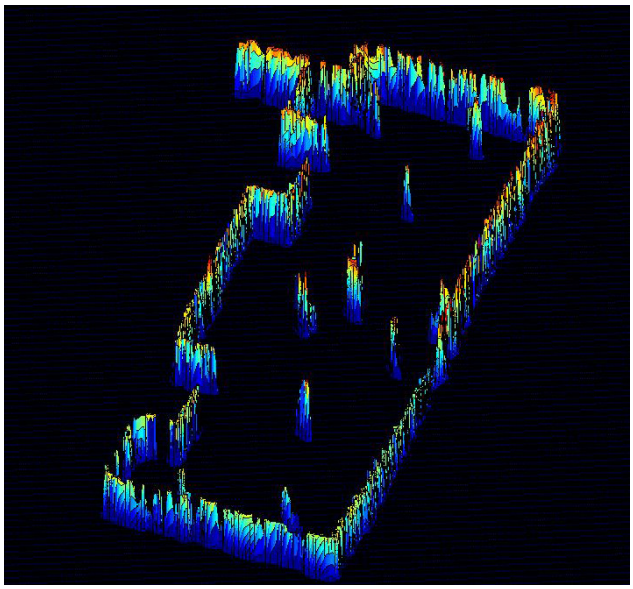

Figure 4. Surface of the map's probability distribution as resulted from the G-SLAM method

Figures 2,3,4 demonstrates the map resulted from the G-SLAM method with $N = 8$ particles and $M = 10$ additional features for every observation, while the resulted map consisted of about 3280 map points (a mean density of 1.1 map points per square meter).

In figure 3 we see a detailed view of the G-SLAM map probability distribution in comparison to the FastSLAM 2.0 map with 8 particles and the Grid SLAM with 3000 cells and a high resolution Grid SLAM with 48000 cells. It is noteworthy that the G-SLAM's map exhibits more detailed characteristics than the FastSLAM's and the low resolution Grid SLAM even if the Grid SLAM's cells are almost the same in number with the G-SLAM's map points. In order to achieve the same resolution with Grid SLAM we need to use around 15 times more dense grid with almost 48000 cells as it is shown in the fourth image on figures 3 and 4. Table 1 shows that the G-SLAM method is slower and inaccurate with small amount of feature particles than FastSLAM, but on the other hand G-SLAM is more accurate and faster when is used with higher number of features M. Also the area which is free of obstacle (blue area) is also free of features θ and all of the features are gathered around the obstacles. The red area represents the highest possibility of the existence of obstacles.

More experiments with a variety in the number of particles N were performed. Also the method tested in different number of additional map points M. Table 2 presents the resulted mean distance error of the car and the mean process time using 2, 8 and 30 particles in the pose estimation procedure and 4, 10 and 20 additionally generated map points for every observation.

Table 2 shows that the G-SLAM algorithm results in a relatively high mean distance error when runs with 2 particles, due to its incapability to be consistent with few particles. In this case the map and the car's path acquire a cumulative error high enough to lead the algorithm into inconsistency. On the other hand the algorithm seems to converge relative fast with respect to the number of particles, since with 8 particle results in the minimum error.

4. Conclusions

In this paper it is presented the G-SLAM method and it's mathematical derivation. The G-SLAM usesn the simulation technique on both the kinematic and measurement models in order to combine probabilities resulted from recursive forms. It results a detailed probability distribution of the map especially arount the objects while it estimates of the robot's trajectory.

Future work will be the examine the G-SLAM method compatibility in to dynamic environments. We intent to use the proposed method to a robotic platform and we will investigate the accuracy of the results and the consistency in dynamic environments.

 doi:10.20944/preprints201902.0003.v1

Table 1. Comparison results between G-SLAM, FastSLAM 1.0 (FS 1), FastSLAM 2.0 (FS 2), Grid Occupancy SLAM (GOSLAM) and Grid Occupancy SLAM with high resolution (GOSLAM HR) on the Car Park dataset.

Method	Number of particles N	Number of features M	Time/step (sec)	Position error (m)
GSLAM	2	4	22 ms	1.55 m
FS 1	2	–	12 ms	0.84 m
FS 2	2	–	31 ms	0.47 m
GOSLAM	2	–	14 ms	1.04 m
GOSLAM HR	2	–	180 ms	1.10 m
GSLAM	8	10	81 ms	0.41 m
FS 1	8	–	51 ms	0.62 m
FS 2	8	–	101 ms	0.42 m
GOSLAM	8	–	46 ms	0.57 m
GOSLAM HR	8	–	648 ms	0.41 m
GSLAM	30	10	294 ms	0.40 m
FS 1	30	–	148 ms	0.43 m
FS 2	30	–	281 ms	0.40 m
GOSLAM	30	–	221 ms	0.40 m
GOSLAM HR	30	–	1943 ms	0.41 m

Table 2. Comparison results on the Car Park dataset with different number of particles and different number of per step additional points M.

Number of particles N	Number of features M	Time/step (ms)	Position error (m)
2	4	22 ms	1.55 m
2	10	28 ms	1.15 m
2	20	33 ms	1.19 m
8	4	74 ms	0.44 m
8	10	81 ms	0.41 m
8	20	92 ms	0.40 m
30	4	278 ms	0.42 m
30	10	294 ms	0.40 m
30	20	314 ms	0.40 m

 doi:10.20944/preprints201902.0003.v1

References

1. Zikos, N.; Petridis, V. G-SLAM: A novel SLAM method. Control Automation (MED), 2012 20th Mediterranean Conference on, 2012, pp. 530–535. doi:10.1109/MED.2012.6265692.
2. Smith, R.; Cheeseman, P. On the Representation and Estimation of Spatial Uncertainty **1986**.
3. Thrun, S.; Burgard, W.; Fox, D. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents series)*; The MIT Press, 2005.
4. Montemerlo, M.; Thrun, S. Simultaneous localization and mapping with unknown data association using FastSLAM. Robotics and Automation, 2003. Proceedings. ICRA '03. IEEE International Conference on, 2003, Vol. 2, pp. 1985–1991 vol.2. doi:10.1109/ROBOT.2003.1241885.
5. Montemerlo, M.; Thrun, S.; Koller, D.; Wegbreit, B. FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem. AAAI/IAAI, 2002, pp. 593–598.
6. Montemerlo, M.; Thrun, S.; Koller, D.; Wegbreit, B. FastSLAM 2.0: An Improved Particle Filtering Algorithm for Simultaneous Localization and Mapping that Provably Converges. IJCAI, 2003, pp. 1151–1156.
7. Montemerlo, M.; Thrun, S.; Siciliano, B. *FastSLAM: a scalable method for the simultaneous localization and mapping problem in robotics*; Vol. v. 27, Springer: Berlin, 2007.
8. Dissanayake, M.; Newman, P.; Clark, S.; Durrant-Whyte, H.; Csorba, M. A solution to the simultaneous localization and map building (SLAM) problem. *Ieee Transactions On Robotics and Automation* **2001**, *17*, 229–241.
9. Grisetti, G.; Stachniss, C.; Burgard, W. Improved Techniques for Grid Mapping With Rao-Blackwellized Particle Filters. *IEEE Transactions on Robotics* **2007**, *23*, 34–46. doi:10.1109/TRO.2006.889486.
10. Rodriguez-Losada, D.; San Segundo, P.; Matia, F.; Pedraza, L. Dual FastSLAM: Dual Factorization of the Particle Filter Based Solution of the Simultaneous Localization and Mapping Problem. *Journal of Intelligent and Robotic Systems*. doi:10.1007/s10846-008-9296-4.
11. Austin Eliazar, R.P. DP-SLAM: Fast, robust simultaneous localization and mapping without predetermined landmarks. International Joint Conference on Artificial Intelligence, 2003.
12. Zikos, N.; Petridis, V. 6-DoF Low Dimensionality SLAM (L-SLAM). *Journal of Intelligent & Robotic Systems* **2014**, pp. 1–18.
13. Petridis, V.; Zikos, N. L-SLAM: reduced dimensionality FastSLAM algorithms. WCCI, 2010, pp. 2510–2516.
14. Kwak, N.; Kim, G.w.; Lee, B.h. A new compensation technique based on analysis of resampling process in fastslam. *Robotica* **2008**, *26*, 205–217. doi:http://dx.doi.org/10.1017/S0263574707003773.
15. Kehagias, A. Convergence Properties of the Lainiotis Partition Algorithm. *Control and Computers* **1991**, *1*, 6.
16. Nieto, J.; Guivant, J.; Nebot, E.; Thrun, S. Real time data association for FastSLAM. Robotics and Automation, 2003. Proceedings. ICRA '03. IEEE International Conference on, 2003, Vol. 1, pp. 412–418 vol.1.
17. Nieto, J.I.; Guivant, J.E.; Nebot, E.M. The HYbrid Metric Maps (HYMMS): A novel map representation for denseSLAM. In IEEE International Conference on Robotics and Automation (ICRA, 2004, pp. 391–396.