μ-Planner: A Robot Path Planning Approach Based on Language Measure of Unsupervised Automata

Jose G.N. Carvalho Filho, Lucas Molina, Eduardo O. Freire and Elyson A.N. Carvalho*

Electrical Engineering Department - UFS, Sergipe, Brazil

Abstract: This paper proposes a robot path planner based on language measure, μ-planner. Workspace is discretized in a occupancy grid map and we model the system by considering how events, associated to robot's motions, take it to different cells (discrete positions). The calculated language measure values corresponds to a gradient, which the robot can use reach its destination by choosing events that take it to states with higher measure values. Concepts of Lapace's equation and harmonic functions are used to prove that our method guarantees both the existence and monotonicity of language measure. The proposed method is simple and computationally inexpensive and guarantees existence of path from any co-accessible state to the destination. Experiments considering different scenarios have been performed to validate and compare μ-planner with similar methods.

Keywords: Path planning, Language measure, Event probability, Mobile robots and grid map.

1. INTRODUCTION

Path planning is one of the most basic tasks to be performed in mobile robot applications. Several methods have been proposed in the last decades to allow robots calculating a path from its current position to a desired destination [10-13, 16, 19].

Based on the mathematical tools used to describe the workspace and calculate a path taking the robot toward the destination, most methods can be classified in grid maps, roadmaps and potential field planners, [17]. Path planners based on both grid maps and roadmaps describe how a robot can move through the workspace using a graph, than search algorithms, such as Dijkstra and *A**, can be used to obtain a path connecting the robot's initial position to the destiny. The main difference in these methods is how they discretize workspace in nodes and define edges connecting them.

Potential field methods define a potential value for each robot configuration (position in the workspace, for instance). Potential values usually are calculated defining repulsive forces from obstacles and attractive forces from destiny.

Authors also proposed path planning methods based on Discrete Event Systems (DES) [2, 18, 24]. Most methods use automaton or Petri net structures and formal verification to generate paths, as sequences of events representing robot actions or states representing discrete positions in the workspace. Recently, methods based on language measure, [4, 5, 21, 22], have been proposed to calculate robot paths. The basic idea of language measure is to attribute a value to each state based on how close they are to marked states and how many event strings intersects on them.

Path planners based on language measure model the robot possible motions through a workspace as an automaton, marking states representing the destiny and obstacles. By attributing positive values (+1) to the destiny state and negative (-1) to the obstacles (or collision) states, the methods can generate a gradient without local maximum (or minimum), allowing the robot to reach the destination from any co-accessible state. However, the computational cost can be quite high.

This paper proposes μ-planner, a simple and computationally efficient method based on language measure. The method is able to produce paths similar to those obtained from [4 and 5] at much lower computational cost. μ-planner does not require a supervisory control to ensure the existence of numerical solution and absence of local minimum or maximum values. By defining an automaton structure that already guarantees both existence of a solution and its monotonicity, μ-planner avoids iterative processes to obtain a supervisory, being able to calculate language measure with a single matrix inversion.

2. LANGUAGE MEASURE THEORY

Let $G = (Q, \Sigma, \delta, q_{init}, Q_m)$ be a deterministic finite-state automaton (DFSA) and $\hat{\delta} : Q \times \Sigma^* \to Q$ the extended state transition function. In addition, let G_i be a version o G in which $q_{init} = q_i \in Q$.

*Address correspondence to this author at the Electrical Engineering Department - UFS, Sergipe, Brazil; E-mail: jgnunes@ufs.br

E-ISSN: 2409-9694/20

Definition 1. The generated and marked languages of an automaton G_i, $L(G_i)$ and $L_m(G_i)$, are defined as:

$$L(G_i) = \{s \in \Sigma^* \mid \hat{\delta}(q_i, s) \in Q\} \tag{1}$$

$$L_m(G_i) = \{s \in \Sigma^* \mid \hat{\delta}(q_i, s) \in Q_m\} \tag{2}$$

Definition 2. Language $L(q_i, q_j)$ (or simply $L_{i,j}$) corresponds to the set of strings starting in q_i and terminating in q_j. Formally, $L(q_i, q_j)$ is defined as:

$$L(q_i, q_j) = \{s \in \Sigma^* \mid \hat{\delta}(q_i, s) = q_j \in Q\} \tag{3}$$

Language $L(q_i)$, the set of all strings allowed in G from state q_i, is defined as:

$$L(q_i) = \bigcup_{q_j \in Q} L(q_i, q_j) \equiv L(G_i) \tag{4}$$

Language $L(G)$ corresponds to the set with all strings initiating at any state of Q. Formally, we have that:

$$L(G) = \bigcup_{q_i \in Q} L(q_i) \tag{5}$$

The set of marked states, Q_m, can be partitioned as $Q_m = Q_m^+ \cup Q_m^-$, where Q_m^+ represents the set of states we desire to reach and Q_m^- the states we have to avoid.

Definition 3. The language measure function $\mu : 2^{L(G)} \to \mathbb{R}$ associates a real value to a language $L(q_i) \in L(G)$, such that:

$$\mu(L(q_i)) \begin{cases} = 0, & q_i \notin Q_m \\ > 0, & q_i \in Q_m^+ \\ < 0, & q_i \in Q_m^- \end{cases} \tag{6}$$

In order to give a physical meaning to the measure μ of a language $L(q_i)$, [22] defined it as the probability of reach a state in Q_m^+ from a state q_i, while avoiding states in Q_m^-.

The computation of μ relies on three structures: event probability (or cost) matrix $\tilde{\prod}$; state transition probability matrix $\prod$ and the characteristic function $\chi : Q \to [-1, 1]$.

Definition 4. For each event $\sigma_k \in \Sigma$ and state $q_j \in Q$, the probability of triggering σ_k at q_j, $\tilde{\pi}(q_j, \sigma_k)$ (or simply $\tilde{\pi}_{jk}$), is defined such that:

1. $\tilde{\pi}_{jk} \in [0, 1)$ and $\sum_k \tilde{\pi}_{jk} < 1$

2. $\tilde{\pi}(q_j, \varepsilon) = 1$ and $\tilde{\pi}_{jk} = 0$ if $\delta(q_j, \sigma_k)$ is undefined;

3. $\tilde{\pi}(q_j, \sigma_k s) = \tilde{\pi}(q_j, \sigma_k)\tilde{\pi}(\delta(q_j, \sigma_k), s)$.

Item (1) provides a sufficient condition to existence of a finite value of μ [22]. Items (2) and (3) provide an iterative way to get the probability of occurring a string s from a state q_j.

Considering $|Q| = n$ and $|\Sigma| = l$, the event probability matrix is defined as:

$$\tilde{\prod} = \begin{bmatrix} \tilde{\pi}_{11} & \tilde{\pi}_{12} & \cdots & \tilde{\pi}_{1l} \\ \tilde{\pi}_{21} & \tilde{\pi}_{22} & \cdots & \tilde{\pi}_{2l} \\ \vdots & \vdots & \ddots & \vdots \\ \tilde{\pi}_{n1} & \tilde{\pi}_{n2} & \cdots & \tilde{\pi}_{nl} \end{bmatrix} \tag{7}$$

Definition 5. The probability of reaching a state q_j from q_i, with the occurrence of a single event, is defined as:

$$\pi_{ij} = \sum_{\sigma \in \Sigma | \delta(q_i, \sigma) = q_j} \tilde{\pi}(q_i, \sigma) \tag{8}$$

Based on (8), the state transition probability matrix is defined as $\prod |_{ij} = \pi_{ij}$.

Definition 6. The characteristic function $\chi : Q \to [-1, 1]$ allows the designer to set weights on the states based on its perception of the application. Formally, we have that:

$$\forall q_i \in Q, \quad \chi(q_i) \in \begin{cases} [-1, 0) & , \ q_j \in Q_m^- \\ (0, +1] & , \ q_j \in Q_m^+ \\ 0 & , \ q_j \notin Q_m \end{cases} \tag{9}$$

The state weighting vector, χ-vector, is defined by $\chi = [\chi_1 \chi_2 \cdots \chi_n]^T$, where $\chi_j \equiv \chi(q_j)$. The signed real measure of a language $L(q_i)$ is defined as:

$$\mu_i \equiv \mu(L(q_i)) = \sum_{q_j \in Q} \mu(L(q_i, q_j)) \tag{10}$$

where $\mu(L(q_i, q_j))$ is defined by (11).

$$\mu(L(q_i,q_j)) = \left(\sum_{s \in L(q_i q_j)} \tilde{\pi}(q_i,s) \right) \chi_j \quad (11)$$

In [22], the authors show that Equation (10) can be expressed as:

$$\mu_i = \sum_{q_j \in Q} \pi_{ij} \cdot \mu_j + \chi_i \quad (12)$$

Considering a matrix structure, Equation (12) can be expressed as:

$$\boldsymbol{\mu} = \prod \boldsymbol{\mu} + \boldsymbol{\chi} \quad (13)$$

The solution of (13) is given by:

$$\boldsymbol{\mu} = (\mathbf{I} - \prod)^{-1} \boldsymbol{\chi} \quad (14)$$

where $\mathbf{I}$ is the $n \times n$ identity matrix.

After obtaining the $\boldsymbol{\mu}$-vector, $\boldsymbol{\mu} = [\mu_1 \mu_2 \cdots \mu_n]$, the system gets a metric from which it can choose the next action (enabled event in the plant G). By choosing the event that leads to the state with higher μ value, the system will be executing the string with higher probability to reach a state in Q_m^+.

3. LANGUAGE MEASURE ON ROBOT PATH PLANNING

In the last decades, several works on path planning based on language measure have been proposed. Next, we present a brief overview of how language measure have been applied in robotics.

In [21 and 22], the authors define the basis of signed real language measure. Other works from the same research group address specific issues such as computational costs of the algorithms proposed to obtain the language measure, [20]; present proofs that, under some conditions, its always possible to obtain a finite measure μ, [23]; etc.

An important aspect often addressed by the authors is how guarantee that $(\mathbf{I} - \prod)$ is an inversible matrix. In [23], the author redefine $\tilde{\prod}$ based on Markov conditional probability and presents a method for estimating it that results in $\prod$ as a stochastic matrix.

As result of the use of stochastic matrices, the premise $\sum_k \tilde{\pi}_i k < 1$ is not satisfied anymore and there is no guarantee matrix $(\mathbf{I} - \prod)$ is inversible. To circumvent such problem, the authors proposed choosing a convenient value of a parameter θ, such that $0 < \theta << 1$, and calculating the language measure as:

$$\boldsymbol{\mu}(\theta) = [\mathbf{I} - (1-\theta)\prod]^{-1} \boldsymbol{\chi} \quad (15)$$

The chosen θ must be small enough to guarantee that $\boldsymbol{\mu}(\theta)$ is invariant to θ, i.e. $\forall q_i, q_j \in Q \mid \mu_i < \mu_j \rightarrow \mu(\theta)_i < \mu(\theta)_j$, and higher enough to guarantee that there will be no numerical problem to calculate the inverse.

Chattopadhyay and Ray [6] addresses this problem and proposes an extension of language measure, the *renormalized language measure*. In [7], the authors proposed an algorithm to estimate the critical lower bound of θ, namely θ^*.

In [4 and 5], authors proposed a path planning method based language measure and supervisory control theories. The workspace is discretized in a grid and it's considered the robot can move to one of its 8 neighbors, each motion represented by an event $\sigma_k \in \Sigma_C$, with $k = \{1, \cdots, 8\}$. Transitions to states representing occupied positions are allowed. However, once in such a state, there is a single uncontrollable event σ_u that can occur and taking plant G to O_b state, which represents a collision with an obstacle. Also, the boundaries of the workspace are not considered in the model, i.e. there are no states representing them.

Figure **1** illustrates how the workspace is discretized and an automaton G representing how the robot can move through the workspace grid cells. The events representing the robot motions where suppressed to simplify the figure.

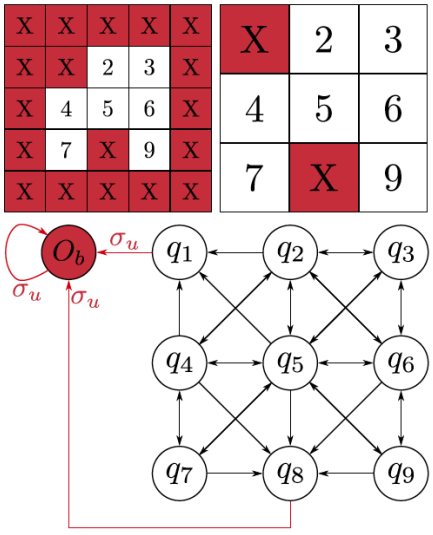

Figure 1: Illustration of the discretized workspace and an automaton modeling the robot motion.

Formally, the system is represented by an automaton $G=(Q,\Sigma,\delta,\Gamma,q_{init},Q_m)$. The set Q is formed by states representing both the free (Q_F) and occupied (Q_O) cells in the grid map and state O_b, $Q=Q_F\cup Q_O\cup\{O_b\}$. The alphabet is defined by $\Sigma=\Sigma_C\cup\{\sigma_u\}$ and $Q_m=\{q_{goal},O_b\}$. Finally, function $\Gamma:Q\rightarrow 2^{\Sigma}$ indicates which events are enabled at each state.

The event probability matrix, $\tilde{\prod}$, is defined based on the number of events enabled in each state. For states representing occupied positions, only event σ_u is enabled and the probability is set as 1. Equation (16) presents how event probabilities are defined:

$$\tilde{\prod}|_{ik}=\begin{cases}\dfrac{1}{|\Gamma(q_i)\cap\Sigma_C|} & ,if\ \sigma_k\in\Gamma(q_i)\cap\Sigma_C\\ 0 & ,if\ \sigma_k\notin\Gamma(q_i)\cap\Sigma_C\\ 1 & ,if\ \sigma_k=\sigma_u\in\Gamma(q_i)\end{cases}\tag{16}$$

Characteristic function χ is defined as follows:

$$\chi(q_i)=\begin{cases}-1 & ,if\ q_i\in Q_O\\ 1 & ,if\ q_i=q_{goal}\\ 0 & ,otherwise\end{cases}\tag{17}$$

In order to ensure global monotonicity of the language measure, [4 and 5] propose an algorithm to compute the optimal supervisor for G. Iteratively, the algorithm recalculates the language measure, defined by Equation (18), and use it to obtain the set of disabled transitions.

$$\boldsymbol{\nu}=\theta^*[\mathbf{I}-(1-\theta^*)\textstyle\prod]^{-1}\boldsymbol{\chi}\tag{18}$$

where θ^* is the critical lower bound of θ and $\prod$ is obtained according to Definition 5.

At each iteration, the algorithm disables controllable transitions $q_i\overset{\sigma}{\rightarrow}q_j$ such that $\boldsymbol{\nu}_i>\boldsymbol{\nu}_j$. When a transition is disabled, the value π_{ij} is added to π_{ii} (self-loop) and π_{ij} is set to 0. Then, θ^* and $\boldsymbol{\nu}$ are recalculated based on the updated $\prod$. The algorithm terminates when the sets of disabled transitions ($\mathscr{D}$) of two consecutive iterations coincide. Final language measure, $\boldsymbol{\nu}_{\#}$, is defined as the $\boldsymbol{\nu}$ from the last iteration.

Since the method ensures global monotonicity, the robot can navigate toward the goal by simply moving from its current position to its neighbor with the highest $\boldsymbol{\nu}_{\#}$ value.

Other works focusing in specific aspects such as localization uncertainties, smoother paths, efficient re-planning and navigation without global positioning facilities have been proposed, [3, 8, 9, 14 and 15]. However, as these works are based on the same framework to calculate $\boldsymbol{\nu}$, we do not present them in this paper.

4. PROPOSED METHOD

In this paper, the path planning problem is also addressed considering a language measure approach. However, we define event probability matrix $\tilde{\prod}$ in a way that guarantees the existence of matrix $(\mathbf{I}-\prod)$ inverse and that $\boldsymbol{\mu}$, defined by Equation (14), corresponds to a globally monotonic gradient. Specifically, we propose a definition for event probability matrix $\tilde{\prod}$ that results in a μ function that holds maximum and minimum principle. By doing so, our method does not need to calculate neither a θ value nor a supervisor, making it computationally inexpensive.

Maximum and minimum principle states that there is no local (or even global) maximum or minimum in the gradient inner region [1]. The maximum and minimum values occurs only in the boundary region and critical points. In path planning applications, the boundary region, $\partial\Omega$, corresponds to the workspace borders and obstacles. A single critical point represents the destination.

Let $G=(Q,\Sigma,\delta,\Gamma,q_{init},Q_m)$ be an automaton representing how the robot can move through a 2-D workspace and $\Sigma=\{\sigma_1,\cdots,\sigma_8\}$ the event set, as illustrated in Figure **2**. Also, let the set of states be defined as a partition $Q=Q_f\cup Q_o\cup\{q_{goal}\}$, in which Q_o is the set of states representing positions in $\partial\Omega$ and Q_f the states in the free space region. In this paper, we propose a μ function definition, presented in Equation (19), that satisfies the maximum and minimum principle.

$$\mu(q_i)=\begin{cases}\dfrac{1}{8}\displaystyle\sum_{\sigma_k\in\Gamma(q_i)}\mu(\delta(q_i,\sigma_k)) & ,if\ q_i\in Q_f\\ K_1 & if\ q_i\in Q_o\\ K_2 & if\ q_i=q_{goal}\end{cases}\tag{19}$$

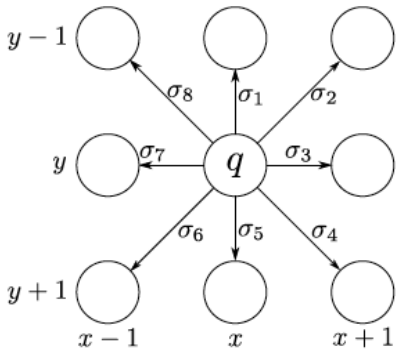

Figure 2: Events associated with the robot's motion.

where K_1 and K_2 are chosen constants that just need to be different. If $K_2 > K_1$, the language measure grows toward the destiny

Proof. The classical example of functions that holds maximum and minimum principle are the harmonic functions. Thus, the μ defined by Equation (19) being a harmonic function is a sufficient, but not necessary, condition to ensure the maximum and minimum principle.

Let $\ell : Q \to \mathbb{N}^2$ be a function that maps a state q_i to its position (x,y) in the grid cell and $\mu(q_i) \equiv \mu(x,y)$, if $\ell(q_i) = (x,y)$.

A harmonic function is a solution for Laplace's equation, defined by Equation (20) for a 2-dimensional space:

$$\nabla^2 U(q) = \frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2} = 0, \quad \forall q = (x,y) \in \Omega \tag{20}$$

where $U(q)$ is a harmonic function defined over region Ω.

To solve Laplace's equation in discretized environments, the partial differential equations are often replaced by finite difference equations, an approximation obtained by Taylor series around a point (x_0, y_0).

By considering $\mu(x,y)$ as a Talyor series limited to second-order (around x_0 and y_0), we have that:

$$\begin{aligned} \mu_{x,y} = \; & \mu_{x_0,y_0} + (x-x_0)\frac{\partial \mu}{\partial x}(x_0,y_0) + (y-y_0)\frac{\partial \mu}{\partial y}(x_0,y_0) \\ & + \frac{(x-x_0)^2}{2}\frac{\partial^2 \mu}{\partial x^2}(x_0,y_0) + \frac{(y-y_0)^2}{2}\frac{\partial^2 \mu}{\partial y^2}(x_0,y_0) \\ & + (x-x_0)(y-y_0)\frac{\partial}{\partial x}\frac{\partial \mu}{\partial y}(x_0,y_0) \end{aligned} \tag{21}$$

Evaluating (21) for $\Delta_x > 0$ and $\Delta_y > 0$ on both sides of x_0 and y_0, we can rewrite Equation (20) as a finite difference equation centered in (x_0, y_0):

$$\begin{aligned} \nabla^2 \mu(x,y) = & \sum_{\substack{i,j \in \{-1,0,1\} \\ i \neq 0 \; or \; j \neq 0}} \mu(x_0 + i\Delta_x, y_0 + j\Delta_y) \\ & -8\mu(x_0,y_0) = 0 \end{aligned} \tag{22}$$

Considering $\Delta_x = \Delta_y = 1$ (smallest displacement in the grid cell), Equation (22) can be rewritten as:

$$\begin{aligned} & \mu_{x-1,y-1} + \mu_{x,y-1} + \mu_{x+1,y-1} + \mu_{x,y-1} + \mu_{x,y+1} \\ & + \mu_{x+1,y-1} + \mu_{x+1,y} + \mu_{x+1,y+1} - 8\mu_{x,y} = 0 \end{aligned} \tag{23}$$

Thus, for each position that is not a critical point or inside the boundary region, the μ value can be calculated based on (23). Analysing Equation (23), one can notice it corresponds to the first line of our μ definition, presented in Equation (19).

Considering *Dirichlet's condition* [1], we can set constant values to the potential of points in $\partial\Omega$ and of critical points, such as:

$$U(q) = \begin{cases} K_1 & , \forall q \in \partial\Omega \\ K_2 & , q = q_{goal} \end{cases} \tag{24}$$

in which $K_1, K_2 \in \mathbb{R}$ are chosen constant values.

Notice that Equation (24) corresponds to the remaining lines of Equation (19). Harmonic functions properties also guarantees that the magnitude of $|K_2 - K_1|$ does not influence the gradient directions. Thus, for any $K_2 > K_1$, the generated path will be the same. □

Our goal, then, is to propose a formulation to $\tilde{\prod}$ such that state transition matrix $\prod$ and characteristic function χ results in the $\mu(q_i)$ definition from (19). To do so, we define automaton G transitions allowing only events taking the robot to a neighbor in Q_f. Formally:

$$\delta(q_i, \sigma_k) = \begin{cases} q_j & , if \; q_i, q_j \in Q_f \\ undefined & , otherwise \end{cases} \tag{25}$$

Based on automaton G, $\tilde{\prod}$ can be defined as:

$$\tilde{\prod}|_{ik} = \begin{cases} \frac{1}{8} & , if \; \sigma_k \in \Gamma(q_i) 0.1cm \\ 0 & , otherwise \end{cases} \tag{26}$$

The characteristic function χ is defined as follows:

$$\chi(q_i) = \begin{cases} K_1 & , if \ q_i \in Q_o \\ K_2 & , if \ q_i = q_{goal} \\ 0 & , otherwise \end{cases} \quad (27)$$

Thus, the system defined by Equation (19), for all $q \in Q$, can be represented by:

$$\boldsymbol{\mu} = \boldsymbol{\mu}\prod + \boldsymbol{\chi} \quad (28)$$

As presented in section 2, μ can be obtained as:

$$\boldsymbol{\mu} = (\mathbf{I} - \prod)^{-1}\boldsymbol{\chi} \quad (29)$$

However, our automaton G and $\tilde{\prod}$ definitions guarantees both the existence of a numerical solution for the system and monotonicity of the obtained $\boldsymbol{\mu}$.

The path taking the robot from its current position to destination can be obtained as sequence of neighbor states with highest μ value. Formally:

$$\mathbf{path} = [q_1 q_2 \cdots q_n] \quad (30)$$

where $q_1 = q_{init}$, $q_n = q_{goal}$ and $q_{j+1} = \delta(q_j, \sigma)$, such that $\sigma = \max_{\sigma_k \in \Gamma(q_j)} \mu(\delta(q_j, \sigma_k))$.

Next, we present the experiments performed in order to evaluate the similarity of the plans obtained from μ-planner with those obtained using ν^* path planning.

5. RESULTS AND DISCUSSIONS

In order to validate the proposed method and evaluate the quality of paths it generates, we perform experiments considering different workspaces and robot positions. The path planner proposed by [5], $\boldsymbol{\nu}^*$, is also implemented and used to generate paths in the same scenarios. Both methods are compared based on the time necessary to calculate the language measure vectors and the similarity of the paths. All experiments were performed using a PC with 4 GB of RAM and Processor Intel I3 running Linux Mint (18.2). Next, we present the metrics used to evaluate the paths.

5.1. Metrics

Paths are compared regarding four metrics: number of steps, length and minimum and average distance to obstacles. Number of steps, n_{steps}, corresponds to the number states in the path, while the length, p_{len}, is the real distance the robot have to move. These metrics can be defined as:

$$n_{steps} = |\mathbf{path}| - 1 \quad (31)$$

$$p_{len} = \sum_{i=1}^{n_{steps}} \left\| \ell(q_{i+1}) - \ell(q_i) \right\| \quad (32)$$

where $q_i \in \mathbf{path}$ is the i-th state in the path, $\ell(q_i)$ corresponds to q_i position in the grid and operator $\|\cdot\|$ is the Euclidean distance.

Regarding the distance to obstacles, let $d_{co}(q_i)$ be the Euclidean distance between q_i's cell (position in the grid map) and the closest obstacle. The minimum, d_{min}, and average, $\bar{d}$, distances to obstacles can be defined as:

$$d_{min} = \min_{q_i \in \mathbf{path}} d_{co}(q_i) \quad (33)$$

$$\bar{d} = \frac{1}{|\mathbf{path}|} \sum_{q_i \in \mathbf{path}} d_{co}(q_i) \quad (34)$$

Figure **3** illustrates part of a path and the distance between each cell in the path and its closest obstacle.

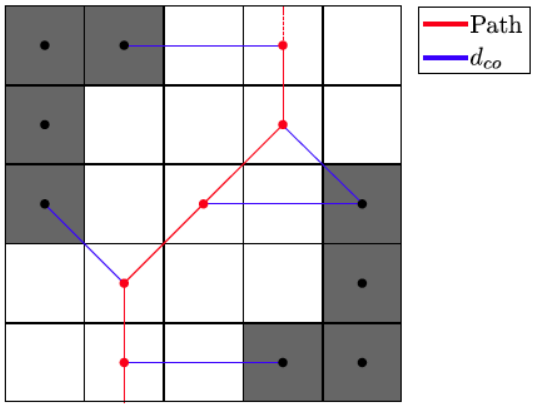

Figure 3: Path length and distances to obstacles.

5.2. Experiments

We performed experiments in three different workspaces, presented in Figure **4**. For each workspace, a destiny and 10 different initial positions were randomly chosen. Then, the proposed method and $\boldsymbol{\nu}^*$ path planner were used to calculate paths for each configuration. For simplicity, we chose $K_1 = -1$ and $K_2 = 1$ for our method.

Additionally, we calculate paths using $\boldsymbol{\nu}^*$ algorithm with constant (arbitrary) values of θ (10^{-2} and 10^{-3}). Thus, one can observe the impact of θ values in path generation and the cost of calculating θ^* (lower bound of θ) at each iteration.

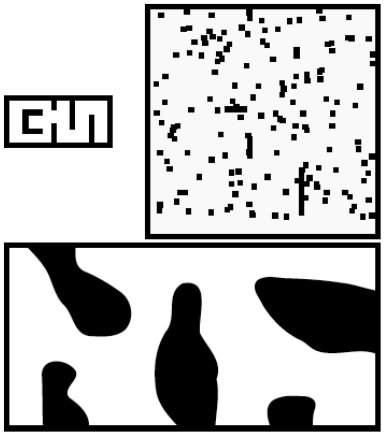

Figure 4: Workspaces considered in the experiments.

Figures **5**, **6** and **7** present the paths, for a single (q_{init}, q_{goal}) configuration, generated by the proposed method and ν^*.

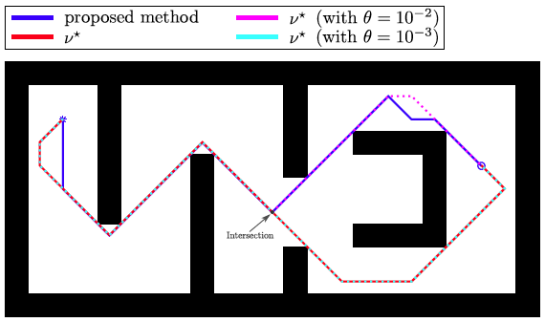

Figure 5: Experiment in workspace prone to local minimum.

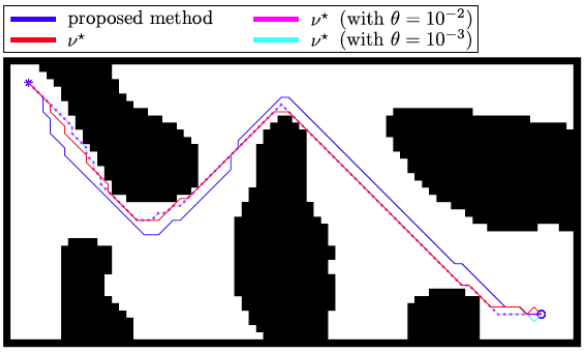

Figure 6: Experiment in cave like workspace.

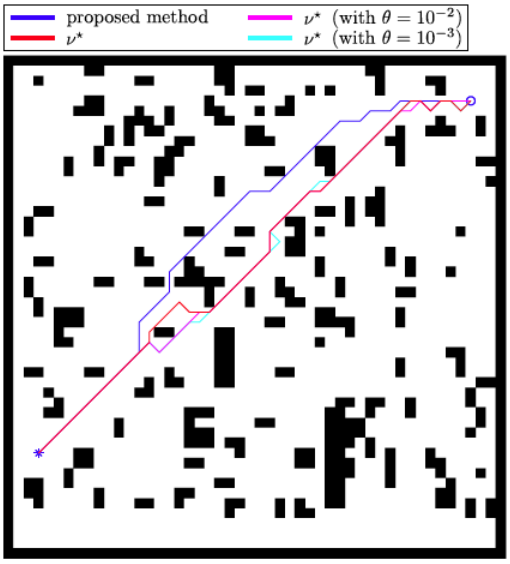

Figure 7: Experiment in workspace with scattered obstacles.

Table **1** presents the data of trials for a single (q_{init}, q_{goal}) configuration (shown in figure **5**) of the experiment with workspace prone to local minimum. The critical lower bound, at the last iteration, was $\theta^* = 9.2 \times 10^{-12}$.

Tables **2** and **3** presents the data of the trials shown in Figures **6** and **7**, respectively. The critical lower bound in these experiments were $\theta^* = 3.6 \times 10^{-14}$ and $\theta^* = 9.1 \times 10^{-15}$, respectively.

For each experiment, different initial positions are considered, so the number of steps and length of paths vary widely in trials. Thus, we compare paths obtained from ν^* path planner and the proposed method by calculating the difference of steps, length and distance to obstacle for each trial. Table **4** presents the average value and standard deviation of these differences.

5.3. Discussions

Both Figures **5**, **6** and **7** and Tables **1**, **2**, **3** and **4** show the proposed method generate paths similar to those obtained using ν^* path planner. For instance, in experiments with the workspace prone to local

Table 1: Results of Experiments in a Workspace Prone to Local Minimum

	Time (s)	n_{steps}	p_{len}	d_{min}	$\bar{d}$
Proposed	0.084 ± 0.001	21	28.04	1	1.34 ± 0.50
ν^*	1.590 ± 0.025	23	30.87	1	1.27 ± 0.47
ν^* ($\theta = 10^{-3}$)	0.995 ± 0.098	23	30.87	1	1.27 ± 0.47
ν^* ($\theta = 10^{-2}$)	0.957 ± 0.031	21	28.87	1	1.25 ± 0.46

Table 2: Results of Experiments in a Cave Like Workspace

	Time (s)	n_{steps}	p_{len}	d_{min}	$\bar{d}$
Proposed	1.10 ± 0.01	79	104.68	1.4	3.64 ± 1.2
ν^*	609.1 ± 2.7	75	99.85	1	2.99 ± 1.92
ν^* ($\theta = 10^{-3}$)	25.83 ± 0.52	75	100.68	1	2.86 ± 1.92
ν^* ($\theta = 10^{-2}$)	25.89 ± 0.093	75	99.85	1	2.88 ± 1.93

Table 3: Results of Experiments in a Workspace with Scattered Obstacles

	Time (s)	n_{steps}	p_{len}	d_{min}	$\bar{d}$
Proposed	1.00 ± 0.04	48	60.43	1	1.98 ± 0.72
ν^*	242.2 ± 4.0	46	61.74	1	1.81 ± 0.59
ν^* ($\theta = 10^{-3}$)	28.46 ± 0.59	45	61.98	1	1.81 ± 0.63
ν^* ($\theta = 10^{-2}$)	27.87 ± 0.41	45	60.33	1	1.81 ± 0.60

Table 4: Summarized Results for all (q_{init}, q_{goal}) Configurations Considered in the Experiments

Workspace	Θ_{steps}	Θ_{len}	Θ_d
Local minimum	1.00 ± 1.05	1.05 ± 1.00	0.076 ± 0.208
Cave like	1.80 ± 1.32	2.46 ± 1.08	0.71 ± 0.23
Scattered obst.	3.3 ± 3.2	3.69 ± 3.30	0.24 ± 0.18

minimum, the difference in the number of steps is 1 ± 1. Such value represents less than 0.5% of n_{steps}, for the paths presented in Figure **5**. Regarding the distance to obstacles, the difference is less than 1 cell unit. Differences increases a little for workspaces with scattered obstacles, due to the higher number of possible paths, but it still less than 1% of the values presented in Table **3**.

On the other hand, Tables **1**, **2** and **3** show that, as the workspace's size increases, ν^* path planner becomes computationally expensive due to θ^* calculation. Notice that, when θ is previously defined, the time necessary to execute ν^* decreases significantly.

Another important aspect is that, for both the proposed method and ν^* path planner, diagonal motions "cost" the same as an one direction motion (up, down, left or right). So, "unecessary" diagonal motions may occur in the generated paths, as in ν^*'s path shown in Figure **7**. However, by defining $\prod$ matrix based on Laplace's equation, the proposed method generates smoother paths (as can be better illustrated in Figure **6**), usually keeping farther away from obstacles.

6. CONCLUSIONS

This paper presents a path planning method based on language measure of unsupervised automata. By considering path planning problem from both DES and the maximum and minimum principle viewpoints, we propose a methodology to define the event probability matrix, $\tilde{\prod}$, that guarantees existence of a numerical solution for $\mu = (\mathbf{I} - \prod)^{-1} \chi$.

Additionally, $\tilde{\prod}$'s definition guarantees the monotonicity of the solution. Thus, language measure μ can be view as a gradient without local maximum (or minimum) and, from any free cell, it's possible to generate a path to destiny by iteratively choosing the neighbor with highest μ value. Also, changes in the workspace can be handled online by adding/removing transitions in automaton G and recalculating μ, since μ-planner is computationally inexpensive.

Experiments with several workspaces show μ-planner generates paths similar to those obtained using ν^* planner. However, since our $\tilde{\prod}$ definition already guarantees existence of solution, there is no need to iteratively compute a supervisor and θ values, which decreases greatly our method computational cost. Specifically, μ-planner considers a single matrix inversion to calculate μ language measure vectors.

6.1 Future Works

In future works, strategies to smooth generated paths and better cope with dynamic environments, as those proposed in [3 and 15], will be addressed. Additionally, other $\tilde{\prod}$ definitions will be studied, in order to allow considering different "costs" for diagonal motions.

REFERENCES

[1] Sheldon Axler, Paul Bourdon, and Ramey Wade. Harmonic Function Theory, volume 137. Springer Science & Business Media, 2001.
https://doi.org/10.1007/978-1-4757-8137-3

[2] José Gilmar Nunes Carvalho Filho, Jean-Marie Alexandre Farines, and José Eduardo Ribeiro Cury. Modeling and synthesis of controllers for multi-robot systems using game structures. In 2013 16th International Conference on Advanced Robotics (ICAR), pages 1-8. IEEE, 2013.
https://doi.org/10.1109/ICAR.2013.6766560

[3] Ishanu Chattopadhyay, Anthony Cascone, and Asok Ray. Formal-language-theoretic optimal path planning for accommodation of amortized uncertainties and dynamic effects. arXiv preprint arXiv:1008.3760, 2010.

[4] Ishanu Chattopadhyay, Goutham Mallapragada, and Asok Ray. L: An intelligent path planning algorithm based on renormalized measure of probabilistic regular languages. In 2008 American Control Conference, pages 1249-1254. IEEE, 2008.
https://doi.org/10.1109/ACC.2008.4586664

[5] Ishanu Chattopadhyay, Goutham Mallapragada, and Asok Ray. : a robot path planning algorithm based on renormalised measure of probabilistic regular languages. International Journal of Control, 82(5):849-867, 2009.
https://doi.org/10.1080/00207170802343196

[6] Ishanu Chattopadhyay and Asok Ray. Renormalized measure of regular languages. International Journal of Control, 79(9):1107-1117, 2006.
https://doi.org/10.1080/00207170600801429

[7] Ishanu Chattopadhyay and Asok Ray. Language-measure-theoretic optimal control of probabilistic finite-state systems. International Journal of Control, 80(8):1271-1290, 2007.
https://doi.org/10.1080/00207170701286322

[8] Ishanu Chattopadhyay and Asok Ray. Generalization of path planning for accommodation of amortized dynamic uncertainties in plan execution. In 2009 American Control Conference, pages 2415-2420. IEEE, 2009.
https://doi.org/10.1109/ACC.2009.5160367

[9] Pritthi Chattopadhyay, Devesh K Jha, Soumik Sarkar, and Asok Ray. Path planning in gps-denied environments: A collective intelligence approach. In 2015 American Control Conference (ACC), pages 3082-3087. IEEE, 2015.
https://doi.org/10.1109/ACC.2015.7171806

[10] Stuart Eiffert, He Kong, Navid Pirmarzdashti, and Salah Sukkarieh. Path planning in dynamic environments using generative rnns and monte carlo tree search. In 2020 IEEE International Conference on Robotics and Automation (ICRA), pages 10263-10269. IEEE, 2020.
https://doi.org/10.1109/ICRA40945.2020.9196631

[11] Raouf Fareh, Mohammed Baziyad, Tamer Rabie, and Maamar Bettayeb. Enhancing path quality of real-time path planning algorithms for mobile robots: A sequential linear paths approach. IEEE Access, 8:167090-167104, 2020.
https://doi.org/10.1109/ACCESS.2020.3016525

[12] Alessandro Gasparetto, Paolo Boscariol, Albano Lanzutti, and Renato Vidoni. Path planning and trajectory planning algorithms: A general overview. Motion and operation planning of robotic systems, pages 3-27, 2015.
https://doi.org/10.1007/978-3-319-14705-5_1

[13] Alejandro Hidalgo-Paniagua, Miguel A Vega-Rodrguez, Joaqun Ferruz, and Nieves Pavón. Solving the multi-objective path planning problem in mobile robotics with a firefly-based approach. Soft Computing, 21(4):949-964, 2017.
https://doi.org/10.1007/s00500-015-1825-z

[14] Devesh K Jha, Pritthi Chattopadhyay, Soumik Sarkar, and Asok Ray. Path planning in gps-denied environments via collective intelligence of distributed sensor networks. International Journal of Control, 89(5):984-999, 2016.
https://doi.org/10.1080/00207179.2015.1110754

[15] Devesh K Jha, Yue Li, Thomas A Wettergren, and Asok Ray. Robot path planning in uncertain environments: A language-measure-theoretic approach. Journal of Dynamic Systems, Measurement, and Control, 137(3):034501, 2015.
https://doi.org/10.1115/1.4027876

[16] Rahul Kala, Anupam Shukla, and Ritu Tiwari. Fusion of probabilistic a* algorithm and fuzzy inference system for robotic path planning. Artificial Intelligence Review, 33(4):307-327, 2010.
https://doi.org/10.1007/s10462-010-9157-y

[17] Jean-Claude Latombe. Robot motion planning, volume 124. Springer Science & Business Media, 2012.

[18] Brenan J McCarragher. Petri net modelling for robotic assembly and trajectory planning. IEEE Transactions on Industrial Electronics, 41(6):631-640, 1994.
https://doi.org/10.1109/41.334580

[19] Pablo Muñoz, Mara D R-Moreno, and Bonifacio Castaño. 3dana: A path planning algorithm for surface robotics. Engineering Applications of Artificial Intelligence, 60:175-192, 2017.
https://doi.org/10.1016/j.engappai.2017.02.010

[20] Asok Ray, Jinbo Fu, and Constantino Lagoa. Optimal supervisory control of finite state automata. International Journal of Control, 77(12):1083-1100, 2004.
https://doi.org/10.1080/00207170420002737 62

[21] Asok Ray and Shashi Phoha. Signed real measure of regular languages for discrete-event automata. International Journal of Control, 76(18):1800-1808, 2003.
https://doi.org/10.1080/00207170310001635392

[22] Asok Ray and Xi Wang. Signed real measure of regular languages. In Quantitative Measure for Discrete Event Supervisory Control, pages 3-37. Springer, 2005.
https://doi.org/10.1007/0-387-23903-0_1

[23] Asok Ray. Signed real measure of regular languages for discrete event supervisory control. International Journal of Control, 78(12):949-967, 2005.
https://doi.org/10.1080/00207170500202447

[24] Faranak Rahimi Soofiyani, Amir Masoud Rahmani, and Mehran Mohsenzadeh. A straight moving path planner for mobile robots in static environments using cellular automata. In 2010 2nd International Conference on Computational Intelligence, Communication Systems and Networks, pages 67-71. IEEE, 2010.
https://doi.org/10.1109/CICSyN.2010.28

Received on 20-11-2020 Accepted on 28-12-2020 Published on 31-12-2020

DOI: https://doi.org/10.31875/2409-9694.2020.07.5