GODZILA: A Low-Resource Algorithm for Path Planning in Unknown Environments

P. Krishnamurthy and F. Khorrami

Abstract— **In this paper, we propose a novel path-planning and obstacle avoidance algorithm GODZILA for navigation in unknown environments. No prior knowledge of the environment is required. The path-planning algorithm follows a purely local approach using only the current range sensor measurements at each sampling instant and requiring only a small number of stored variables in memory. No map of the environment is built during navigation. This minimizes the memory and computational requirements for implementation of the algorithm, a feature that is especially attractive for small autonomous vehicles. The algorithm utilizes three components: an optimization algorithm, a local straight-line path planner to visible targets, and random navigation. It is proved, for navigation in any finite-dimensional space, that the path-planning algorithm converges in finite time with probability 1. The performance of the algorithm is demonstrated through simulations for path-planning in two-dimensional and three-dimensional spaces. It is seen that a relatively small number of range sensor measurements is sufficient even in complex unknown environments.**

I. Introduction

Several problems encountered in a variety of applications including trajectory planning for vehicles and robotic arms subject to constraints can be formulated as path-planning and obstacle avoidance problems. Several approaches have been proposed in the literature to address this problem in a variety of scenarios. These approaches can be broadly classified into two categories based on whether they require building a map of the environment. In two-dimensional space, the path planning problem has been solved in quite general cases using the "Bug" algorithm and its variants [1–3]. This class of algorithms is essentially based on attempting a straight-line path to the target and tracking the contour of any encountered obstacle. They require only a few points from the path to be stored in memory and do not build any map of the environment. However, since they depend critically on the fact that any two-dimensional obstacle has a one-dimensional boundary and hence, that it is always possible to "go around" an obstacle, these algorithms cannot be generalized to higher-dimensional spaces. In higher-dimensional spaces, the available algorithms construct some form of a map of the environment and utilize it for navigation. Approaches along this direction include potential-field based methods [4–6], graph theoretic methods [7–9], control-theory based methods [10], and various heuristic methods. Of these, the potential-field approach which considers a physical analogy with electromagnetic field patterns generated by a negatively charged target and positively charged obstacles appears to be the oldest and the most popular. While this method was first developed for known environments [5,6], numerous techniques have been proposed for navigation in unknown environments [11] by incrementally building an obstacle map during the operation of the algorithm. Harmonic potential fields [12–14] have been considered for elimination of local minima. Variants of the potential-field approach which utilize a similar philosophy include analogies with fluid dynamics [15] and current flow in resistive grids [16]. However, these approaches are based on the building of an obstacle map which could impose formidable memory and computational requirements. While randomized algorithms [17,18] using random walks and Brownian motions have been proposed to reduce the computational complexity, these algorithms still require building and processing of an obstacle map, a task that is particularly not feasible in the context of small autonomous vehicles which have tight payload and power constraints.

In this paper, we propose an algorithm GODZILA (Game-Theoretic Optimal Deformable Zone with Inertia and Local Approach) which provides a solution to the navigation problem in completely unknown environments without requiring the building of an obstacle map. The algorithm follows a purely local approach using only the sensor measurements at the current time and requiring only a small number of stored variables in memory. This minimizes the memory and computational requirements for implemention of the algorithm, a feature that is especially attractive for small autonomous vehicles. The trajectory is generated through the online solution of an optimization cost at each sampling instant. It is shown that the optimization cost can be chosen so that the minimizer can be obtained in closed form. The optimization cost has three terms which penalize, respectively, motion in directions other than the direction to the target, motion towards obstacles, and back-tracking. In addition to the optimization algorithm, GODZILA includes two components, a local straight-line planner utilized if the target is visible and navigation towards a random target. Since the algorithm follows a local approach, it is possible to be caught in a local minimum. When a local minimum or a "trap" is detected, navigation towards a random target is initiated to escape the trap. It is shown that GODZILA provides convergence to the target in finite time with probability 1 in any finite-dimensional space. The path planning and obstacle avoidance problem that we consider is outlined in Section II. The three components of the GODZILA algorithm, i.e., optimization cost-based navigation, local straight-line motion towards a visible target, and random navigation when a local trap is detected are explained in Sections III, IV, and V, respectively. The complete algorithm is summarized in Section VI and the proof of

This work is supported in part by the ARO under contract no. W911NF-04-C-002 and by IntelliTech Microsystems, Inc.

The authors are with the Control/Robotics Research Laboratory(CRRL), Dept. of Electrical and Computer Engineering, Polytechnic University, Six Metrotech Center, Brooklyn, New York 11201.

Emails: pk@crrl.poly.edu, khorrami@smart.poly.edu

convergence is provided in Section VII. Simulations are presented in Section VIII.

II. Problem Formulation

A path-planning algorithm can be viewed as a dynamic system operating on sensor measurements (*inputs*) and generating heading commands (*outputs*). It might incorporate memory (*state*), the simplest model being one that "remembers" the current position and heading. Thus, a path-planning algorithm can be considered as an ordered quadruplet (U, Y, X, T) where U is the input space, Y is the output space, X is the state space, and T is the time set. The underlying set over which the trajectory-generation problem is posed is taken to be a Hilbert space[1] H. In this case, $U = \mathcal{R}^n, Y = H, X = H \times H$, and $T \subset \mathcal{R}$. Here, n is the number of measurement sensors available.

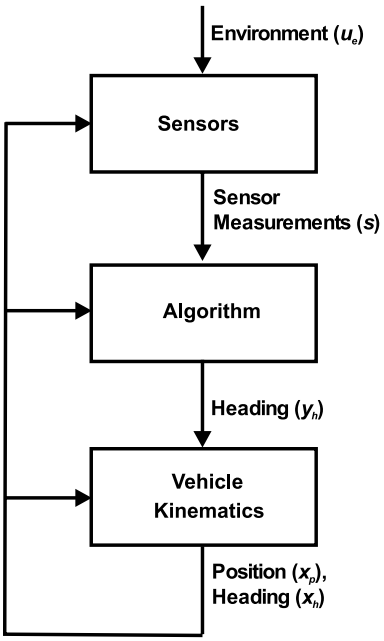

Fig. 1. The system.

Figure 1 illustrates the interaction of the environment, the sensors, and the algorithm in the path-planning system. The sensors provide inputs $s \in U$ to the path-planning algorithm. The sensors themselves can be interpreted as a system with input u_e being a (possibly time-dependent) subset of H and output $s \in U$. Variable u_e defines the obstacle set, i.e., the set of points in H occupied by obstacles. The sensors return the distance from the current position to the obstacle set in the directions of the sensors. Each sensor direction is defined by a unit vector, i.e., by an element of H. The sensor geometry provides $q \in H^n$, with the i^{th} element being the unit vector defining the direction of the i^{th} sensor. The sensor outputs are described as

$$s_i = \inf\{d \in [0, \infty) : x_p + d(x_h + q_i) \in u_e\}, \quad (1)$$

$$i = 1, 2, \ldots, n \; ; \; s = (s_1, s_2, \ldots, s_n) \quad (2)$$

where x_p and x_h are the current position and heading unit-vectors, respectively. If there is no $d > 0$ such that $x_p + d(x_h + q_i) \in u_e$, then s_i is defined to be ∞ . Variables s_i are the distances from the current position to the obstacle set in the n sensor directions (see Figure 2). The sensor measurements can be modified to enforce a clearance zone by exaggerating the physical dimensions of the vehicle. Since this exaggeration is only relevant when the obstacles are close to the vehicle, introducing, for instance, $s_i' = s_i - p_i e^{-s_i}$ with p_i being positive constants, we have $s_i' \approx s_i$ in the far zone and $s_i' \approx s_i - p_i$ in the near zone. This weighting induces large penalties when obstacles are near and small penalties when obstacles are distant.

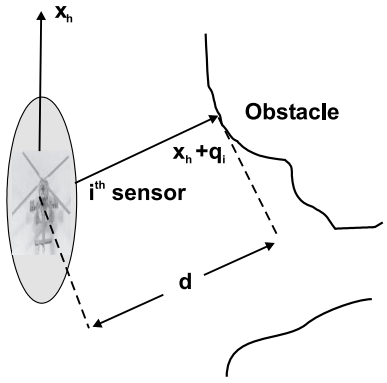

Fig. 2. Sensor Measurements.

We now formulate the path-planning objective. Roughly stated, the objective is to generate a trajectory that tracks a target trajectory $x_d(t)$ while avoiding the obstacle set $u_e(t)$. This objective is stated more explicitly as follows.

Path-planning Objective: Given time-signals $x_d(t)$ and $u_e(t)$, a final time t_f, and a positive constant ϵ_c , generate a time-signal $x_h(t)$ such that the ensuing position signal has the properties[2]

$$\text{a) } \inf_t \rho(x_p(t), u_e(t)) \geq \epsilon_c$$

$$\text{b) } \lim_{t \to t_f} \rho(x_p(t), x_d(t)) = 0. \quad (3)$$

Property (a) characterizes the obstacle-avoidance requirement with ϵ_c being a specified clearance to be maintained whereas property (b) is the target-reaching requirement. If only the final position is specified, $x_d(t)$ is simply a constant x_d. The GODZILA path-planning and obstacle avoidance algorithm achieves this objective using a combination of an optimization procedure, a random walk, and approach to a visible target. At any time, the optimal heading is computed by an optimization procedure using as effective target one of the following:

1) The actual target x_d
2) A fictitious target location on an intermediate straight-line trajectory to the target x_d
3) A random target

[1]A Hilbert space is a vector space with an inner product $< ., . >$ such that the metric space endowed with the metric ρ induced by the inner product is complete. In several applications, H is simply $\mathcal{R}^n$, the n-dimensional Euclidean space.

[2]Here, we use the customary definition of the distance between an element x of a metric space H and a subset A of the metric space as $\rho(x, A) = \inf_{a \in A} \rho(x, a)$, with the distance being defined as infinity if A is the empty set.

III. Optimization Cost Based Navigation

The optimization is performed with respect to an objective function given by

$$J(y) \quad = \quad J_1(y) + J_2(y) + J_3(y) \tag{4}$$

which has the following three components:

1) $J_1(y)$ - a component that penalizes motion in directions other than the target direction,
2) $J_2(y)$ - a component that penalizes motion towards obstacles, and
3) $J_3(y)$ - a component that penalizes back-tracking.

The first component $J_1(y)$ which attempts to make the vehicle move in the direction of the target is of the form

$$J_1(y) \; = \; f_1\left(\left|\left|y - \frac{(x_d - x_p)}{||x_d - x_p||}\right|\right|\right) g_1(||x_d - x_p||) \tag{5}$$

where f_1 is a class-$\mathcal{K}$ function[3] and g_1 is a class-$\mathcal{L}$ function[4]. Let $g_1(0)$ be large so that, in the immediate vicinity of the target, $J_1(y)$ is the dominant term in $J(y)$.

The second component $J_2(y)$ attempts to prevent motion in directions that would bring the vehicle in the proximity of obstacles. The effect of this component is more important in directions that are either along the current heading or along the straight-line heading to the target. This component should also encourage motion in directions along which obstacles are far away, especially if such a direction is along the straight-line heading to the target. Such a behavior is attained by using a function $J_2(y)$ of the general form

$$\begin{aligned} J_2(y) = & \sum_{i \in \mathcal{I}_1} g_{21}(s_i') g_{22}\left(\left|\left|y - \frac{(x_h + q_i)}{||x_h + q_i||}\right|\right|\right) \times \\ & \times \left[g_{23}\left(\left|\left| \frac{(x_d - x_p)}{||x_d - x_p||} - \frac{(x_h + q_i)}{||x_h + q_i||} \right|\right|\right) \right. \\ & \left. + g_{24}\left(\left|\left| x_h - \frac{(x_h + q_i)}{||x_h + q_i||} \right|\right|\right) \right] \\ & - \sum_{i \in \mathcal{I}_2} f_{21}(s_i') g_{25}\left(\left|\left|y - \frac{(x_h + q_i)}{||x_h + q_i||}\right|\right|\right) \times \\ & \times g_{26}\left(\left|\left| \frac{(x_d - x_p)}{||x_d - x_p||} - \frac{(x_h + q_i)}{||x_h + q_i||} \right|\right|\right) \end{aligned} \tag{6}$$

where $g_{21}, \ldots, g_{26}$ are class-$\mathcal{L}$ functions and f_{21} is a positive monotonically increasing bounded function. $\mathcal{I}_1 \cup \mathcal{I}_2$ is a partition of the set $\{1, 2, \ldots, n\}$ where $i \in \mathcal{I}_1$ if $s_i' \leq C$ and $i \in \mathcal{I}_2$ if $s_i' > C$ with C being a threshold clearance such that if the nearest obstacle in a direction is farther than C, then it can be considered a relatively *obstacle-free* direction. g_{21} is picked such that $g_{21}(0)$ is large. This implies that in the close vicinity of obstacles, the effect of obstacles predominates in the objective function. By picking an appropriate value of $g_{21}(0)$ relative to the sizes of the other functions in the optimization cost, a prescribed clearance to obstacles can be imposed.

[3]A continuous function $f : [0, \infty) \mapsto [0, \infty)$ is said to be a class-$\mathcal{K}$ function if it is strictly increasing and $f(0) = 0$.

[4]A function $f : [0, \infty) \mapsto [0, \infty)$ is said to be a class-$\mathcal{L}$ function if it is continuous and strictly decreasing.

The third component $J_3(y)$ of the objective function penalizes change in heading and is of the form

$$J_3(y) \quad = \quad f_3(||y - x_h||) \tag{7}$$

with f_3 being a class-$\mathcal{K}$ function.

$J_1(y)$ penalizes the deviation of y from the "straight-line" heading to the target position. $J_2(y)$ penalizes approach to obstacles and $J_3(y)$ penalizes change in heading. $J_1(y)$ and $J_2(y)$ address the requirements (b) and (a), respectively, of the path-planning objective. $J_3(y)$ which is inspired by the physical notion of inertia is instrumental in preventing limit-cycle oscillations. Without $J_3(y)$, the vehicle could be caught in an endless cycle of back-tracking in a situation wherein the obstacle avoidance incentive is exactly counterbalanced by the target-reaching incentive. $J_3(y)$ essentially makes a potential limit cycle spatially larger and hence provides better chance of finding a way around a blocking obstacle. If $J(y)$ has a unique minimum over the unit ball $\mathcal{B} = \{y : ||y|| = 1\}$, the output of the algorithm is the unique minimizer. If the minimizer is not unique, the output is defined as one which gives the smallest $J_3(y)$. More explicitly, define $P_{cm} = \{y \in \mathcal{B} : J(y) \leq J(y') \forall y' \in \mathcal{B}\}$. P_{cm} is not empty since $J(y)$ is a continuous function defined on the compact set $\mathcal{B}$. The output y_h of the algorithm is any $y \in P_{cm}$ such that $J_3(y) \leq J_3(y') \forall y' \in P_{cm}$. Alternatively, the minimization can be performed over P_c, the projection of the convex hull of the set $(q_1, q_2, \ldots, q_n)$ onto the unit ball. A mechanism similar to the one outlined above can be used for choosing one of a set of minima in the case that the optimization problem does not have a unique minimizer.

The optimization cost $J(y)$ includes the design functions f_1, f_{21}, f_3, g_{21}, g_{22}, g_{23}, g_{24}, g_{25}, and g_{26} which are free to be picked by the designer. Of particular interest is the case when f_1, f_3, g_{22}, and g_{25} are quadratic. The class-$\mathcal{K}$ functions f_1 and f_3 can be picked as $f_1(r) = \overline{f}_1 r^2$ and $f_3(r) = \overline{f}_3 r^2$ where $\overline{f}_1$ and $\overline{f}_3$ are positive constants. The class-$\mathcal{L}$ functions g_{22} and g_{25} can be chosen to be $g_{22}(r) = \overline{g}_{22}(4 - r^2)$ and $g_{25}(r) = \overline{g}_{25}(4 - r^2)$ where $\overline{g}_{22}$ and $\overline{g}_{25}$ are positive constants. Note that the arguments of g_{22} and g_{25} in (6) being the norms of the differences between two unit vectors are smaller than 2. Choosing f_1, f_3, g_{22}, and g_{25} to be the specified quadratic functions, the optimization problem can be solved in closed form to obtain the optimal heading. Using the Lagrange multiplier method to model the fact that the optimization is to be performed over the unit ball, the augmented optimization cost is $J_{aug}(y) = J(y) + \zeta(< y, y > -1)$. The optimal heading is given by $y_h = Y/||Y||$ where

$$\begin{aligned} Y = & \overline{f}_1 g_1(||x_d - x_p||) \frac{(x_d - x_p)}{||x_d - x_p||} \\ & - \overline{g}_{22} \sum_{i \in \mathcal{I}_1} g_{21}(s_i') \left[g_{23}\left(\left|\left| \frac{(x_d - x_p)}{||x_d - x_p||} - \frac{(x_h + q_i)}{||x_h + q_i||} \right|\right|\right) \right. \\ & \left. + g_{24}\left(\left|\left| x_h - \frac{(x_h + q_i)}{||x_h + q_i||} \right|\right|\right) \right] \frac{(x_h + q_i)}{||x_h + q_i||} \\ & + \overline{g}_{25} \sum_{i \in \mathcal{I}_2} f_{21}(s_i') g_{26}\left(\left|\left| \frac{(x_d - x_p)}{||x_d - x_p||} - \frac{(x_h + q_i)}{||x_h + q_i||} \right|\right|\right) \times \end{aligned}$$

$$\times \frac{(x_h+q_i)}{||x_h+q_i||} + \overline{f}_3 x_h. \quad (8)$$

In practice, the calculated heading is translated into a heading rate that attains the required heading in a time dictated by the capabilities of the vehicle. The required heading rate is $\alpha_{max}\frac{y_h-x_h}{||y_h-x_h||}$ where α_{max} is the maximum heading rate achievable by the vehicle.

The above optimization scheme generates a trajectory to solve the path-planning problem. In a practical application of the algorithm, the linear velocity of the vehicle is also free to be chosen. This freedom in choosing the linear velocity is particularly useful when the maximum heading rate achievable by the vehicle imposes a constraint since in that case, the ideal heading generated by the algorithm will only be attained after a delay so that it is important that the vehicle not move too far in that time interval. Thus, the linear velocity v is designed to be such that it is small in the close vicinity of obstacles, i.e., $v = \lambda(s_{ds})$ where λ is a positive monotonically increasing bounded function and s_{ds} given by

$$\begin{aligned} s_{ds} &= \min_{i\in\{1,\ldots,n\}} \gamma_{v1}(s_i)\Bigg[\gamma_{v1}^o \\ &\quad +\gamma_{v2}\left(\left|\left|x_h - \frac{(x_h+q_i)}{||x_h+q_i||}\right|\right|\right)\Bigg] \end{aligned} \quad (9)$$

indicates proximity of obstacles. γ_{v1} and γ_{v2} are class-$\mathcal{K}$ functions and γ_{v1}^o is a positive constant. This ensures that the speed is smaller when obstacles are close in the heading direction. When x_d is constant, it is also desirable that the linear velocity be small when the target is near. The linear velocity can be defined as $v = \lambda(s_{ds})\phi(||x_p - x_d||)$ where ϕ is a bounded class-$\mathcal{K}$ function.

IV. Approaching A Visible Target

At any time, if the target is visible, i.e., the nearest obstacle in the direction of the target is farther than the target, then the vehicle can proceed in a straight-line towards the target. This is particularly true in the case that $x_d(t)$ is a constant. However, a straight-line path to the target may bring the vehicle in close proximity with obstacles. An alternative is to prescribe the straight-line path to the target as a desired trajectory and navigate along this path using the optimization-based approach described in Section III. If a clear straight-line path to the target is detected at time t_0 when the vehicle is at position x_{p0}, then a straight-line trajectory to the target is given by

$$\hat{x}_d(t) = x_{p0} + (x_d(t) - x_{p0})\mu(t - t_0) \quad (10)$$

where $\mu(r)$ is an increasing function that takes the value 1 for $r \geq T_{sl}$. Hence, for times later than $t = t_0 + T_{sl}$, $\hat{x}_d(t)$ coincides with $x_d(t)$. The optimal heading at each time instant is computed using the optimization algorithm described in Section III with x_d replaced by $\hat{x}_d$. This ensures that the vehicle stays away from obstacles while tracking the straight-line trajectory to the target as closely as possible. This feature is of particular use when the vehicle needs to pass through a narrow corridor to approach the target. The introduction of an intermediate trajectory makes the effective target location closer to the vehicle so that the $g_1(||\hat{x}_d - x_p||)$ term is larger. This, indeed, is the motivation for introducing the class-$\mathcal{L}$ function g_1.

If due to detours caused by the presence of obstacles, the vehicle does not reach the target location x_d by time t_0+T_{sl}, then at the next time instant at which a clear straight-line path to the target is detected, another intermediate trajectory can be mapped.

To implement this feature, one of the sensors can be mounted on a pan-and-tilt stage which keeps that sensor constantly in the direction of the target to detect a possible clear straight-line path. Alternatively, the vehicle can periodically be rotated to align one of the sensors in the direction of the target.

V. Random Navigation

It is possible for the vehicle to be caught in a local limit cycle oscillation. The introduction of the inertial term in the optimization cost has the effect of increasing the spatial extent of possible limit cycles. However, if the obstacle set is spatially large, this feature by itself cannot prevent potential limit cycle situations. This is a consequence of the fact that the navigation is based on local sensor data and no map of the environment is built. A local limit cycle or a *trap* can be detected using, for instance, the variance of the position variable over some number of successive sampling instants. Alternatively, the distance to the target can be used as a metric and a trap situation can be inferred if this distance does not change significantly over some period of time. If a trap is detected at time t_0, a random target location $\hat{x}_d$ is chosen. This process is analogous to *simulated annealing*. The random location $\hat{x}_d$ can be picked to be a certain (fixed or random) distance d_{roam} away from the current position in a random direction in which the distance to the nearest obstacle is larger than d_{roam}. Navigation is continued for a time T_{roam} using the optimization procedure described in Section III with x_d replaced by $\hat{x}_d$. If a trap situation has been detected i times till time t, $\alpha(i)$ successive random moves using randomly chosen targets are performed where $\alpha(i)$ is a nondecreasing positive integer function. At time $t = t_0 + \alpha(i)T_{roam}$, the target location is restored as x_d.

VI. Summary of the GODZILA Algorithm

The algorithm is initialized at time $t = 0$. The target trajectory $x_d(t)$ is provided. The algorithm "knows" its position x_p and its heading x_h.

The variables targetvisible, trap, and n_{trap} are initialized to be zero. At each time t, do the following:

- if (!targetvisible)&(!trap)&(the distance to the nearest obstacle in the direct heading to the target is more than the distance to the target) then targetvisible=1, $x_{p0} = x_p$, $t_0 = t$, $\hat{x}_d = x_d$
- if (targetvisible)&($t < t_0 + T_{sl}$) then $\hat{x}_d = x_{p0} + (x_d(t) - x_{p0})\mu(t - t_0)$
- if (targetvisible)&($t \geq t_0 + T_{sl}$) then targetvisible=0
- if (!targetvisible)&(!trap)&(a trap situation is detected) then trap=1, $t_0 = t$, $n_{trap} = n_{trap} + 1$, $\hat{x}_d$=randomly chosen target, $n_t = 1$
- if (trap)&($t \geq t_0 + n_t T_{roam}$) $\hat{x}_d$=randomly chosen target, $n_t = n_t + 1$
- if (trap)&($t \geq t_0 + \alpha(n_{trap})T_{roam}$) trap=0

- if (!targetvisible)&(!trap) then compute optimal heading using optimization algorithm described in Section III
- if (targetvisible)||(trap) then compute optimal heading using optimization algorithm described in Section III with x_d replaced by $\hat{x}_d$

The operators !, &, and || stand for logical NOT, AND, and OR, respectively.

VII. Convergence of the GODZILA Algorithm

The optimization algorithm ensures that a specified clearance to the obstacles is maintained. This is achieved by choosing a large enough value of $g_{21}(0)$ so that the term in the optimization cost that attempts to move the vehicle away from obstacles is dominant. The target reaching objective is achieved in the sense that it is guaranteed that the probability of reaching the target within time t goes to 1 as $t \to \infty$. To demonstrate this property, assume, for simplicity, that the target location and the obstacle set are constant in time. This assumption can be relaxed. Assume, furthermore, that the underlying space H is finite-dimensional. Hence, H is topologically equivalent to $\mathcal{R}^n$ for some n, and therefore can be considered to be $\mathcal{R}^n$. We can also assume[5] that a path to the target exists with the prescribed clearance to the obstacles, i.e., assume that a continuous function $\omega : [0,1] \mapsto H$ exists such that $\omega(0) = x_{pi}$, $\omega(1) = x_d$, and $\inf_{\theta\in[0,1]} \rho(\omega(\theta), u_e) \geq \epsilon_c$. Hence, an open set $\mathcal{O} \subset H$ containing x_d exists such that the target is visible from all points in $\mathcal{O}$, i.e., $\forall a \in \mathcal{O} \backslash \{x_d\}$

$$\inf\left\{d \in [0,\infty) : \left(a + d\frac{(x_d - a)}{||x_d - a||}\right) \in u_e\right\} > \rho(a, x_d). \quad (11)$$

The presence of the class-$\mathcal{L}$ function g_1 in $J_1(y)$ with $g_1(0)$ large coupled with the mechanism to generate intermediate trajectories from points from which the target is visible implies that convergence to x_d is assured from a nonempty open subset $\mathcal{O}_1$ of $\mathcal{O}$. $\mathcal{O}_1$ being an open set contains a ball of radius $r_1 > 0$ centered at x_d. Define $r = \frac{1}{2\sqrt{n}} \min(r_1, \epsilon_c)$. Consider a gridding of $\mathcal{R}^n$ using a hypercube (or *box*) of side r in any orientation centered at x_d and its translates. Assuming that an upper bound on $\sup_{\theta\in[0,1]} ||\omega(\theta)||$ is known, i.e., an upper bound on the excursion required to obtain a path to the target is known, a bounded subset of $\mathcal{R}^n$ and hence a finite number of cells in the grid need to be considered. Such a bound is always available in practice. Number the finite number of cells in this bounded subset of $\mathcal{R}^n$ as $C_1, \ldots, C_N$. A nonempty subset $\mathcal{N}_t$ of $\{1, \ldots, N\}$ exists such that $C_i \subset \mathcal{O}_1$ for all $i \in \mathcal{N}_t$. Define the subset $\mathcal{N}_o$ of $\{1, \ldots, N\}$ to be the indices of those cells which atleast partially contain an obstacle, i.e., $\mathcal{N}_o = \{i | 1 \leq i \leq N, C_i \cap u_e \neq \phi\}$. Then, the set of cells which are completely free of obstacles have indices

$$\mathcal{N}_f = \{i | 1 \leq i \leq n, i \notin \mathcal{N}_o\}. \quad (12)$$

Since $\inf_{\theta\in[0,1]} \rho(\omega(\theta), u_e) \geq \epsilon_c$, the trajectory ω is completely contained in obstacle-free cells, i.e.,

$$\Omega = \{\omega(\theta) | \theta \in [0,1]\} \subset \bigcup_{i\in\mathcal{N}_f} C_i. \quad (13)$$

The optimization procedure in the GODZILA algorithm enforces the ϵ_c clearance and ensures that the cells with indices in $\mathcal{N}_o$ are not entered. At any time, if a cell with index in $\mathcal{N}_t$ is entered, then convergence to the target is guaranteed. An adjacency graph can be constructed with the cells $\{C_i | i \in \mathcal{N}_f\}$ as vertices with adjacency between two cells defined by the cells having a common face. While the resulting graph may not be a connected graph, a path does exist between the cell containing the initial position and the cell containing the target. This is ensured by the fact that ω is a continuous function and is contained in the cells corresponding to indices in $\mathcal{N}_f$. The random navigation constitutes a random walk on this graph. The detection of a local limit cycle is guaranteed by the strategies outlined in Section V. The length of the random walk initiated on detection of a trap increases with time t if the target is not reached. Being a random walk on a finite graph, the probability of entering a cell with index in $\mathcal{N}_t$ goes to 1 as $t \to \infty$. Hence, the probability of reaching the target goes to 1 as $t \to \infty$, i.e., the path-planning algorithm converges in some finite time with probability 1.

VIII. Simulation Studies

In this section, the performance of the GODZILA algorithm is demonstrated using simulations in various two-dimensional and three-dimensional environments. The number of range sensors used in the two-dimensional simulations is three with the orientations being at angles of 0^o, 45^o and -45^o with respect to the vehicle heading. The three-dimensional simulations use five range sensors oriented at yaw and pitch of $(0^o, 0^o)$, $(-45^o, 0^o)$, $(45^o, 0^o)$, $(0^o, 45^o)$, and $(0^o, -45^o)$ with respect to the current heading.

Simulations with two simple two-dimensional environments are shown in Figure 3. In Figure 4, a more complicated obstacle map is shown in which the path to the target lies through a narrow corridor which is visible only from a small region in the space. Note that the GODZILA path-planning algorithm uses only the current sensor measurements and avoids building a map of the environment. Hence, a period of wandering is seen since no way out of the enclosed space is visible. However, when a local trap is detected, random navigations are initiated which successfully bring the vehicle into the region from which the opening is visible.

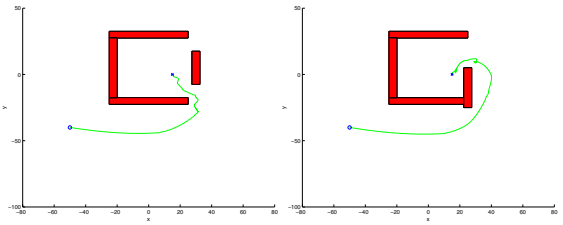

Fig. 3. Two simple two-dimensional obstacle maps.

Figure 5 shows a three-dimensional simulation in which the vehicle starts from an enclosed region to escape from

[5]If such a path does not exist, then the posed path-planning problem is not solvable.

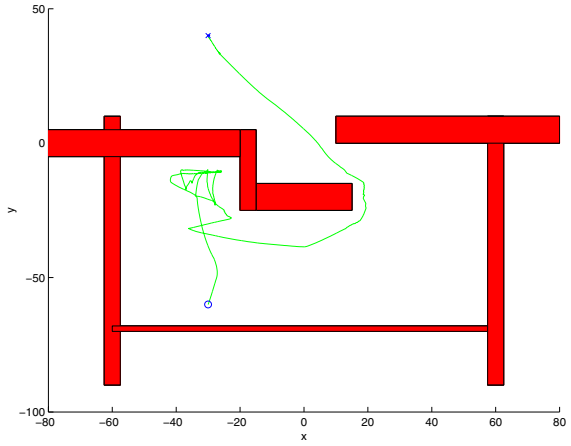

Fig. 4. A more complicated two-dimensional maze.

which it needs to move either above or below the wall. In Figure 6, a small window is provided in the enclosing wall. It is seen that the algorithm elects to fly through the window leading to a shorter path.

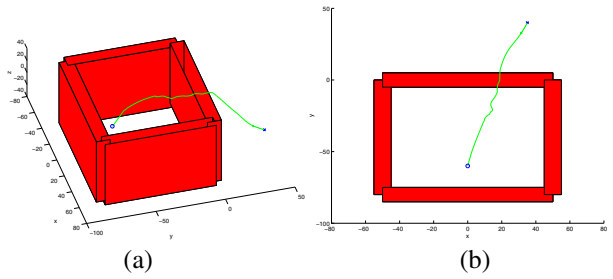

Fig. 5. Three-dimensional simulation: The vehicle flies over the wall to reach the target. (a) 3D view; (b) Top view.

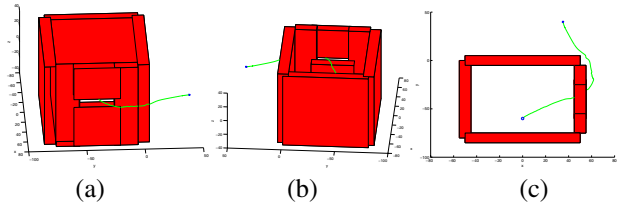

Fig. 6. Three-dimensional simulation: The vehicle flies through a window. (a) 3D view; (b) Another 3D view; (c) Top view.

IX. Conclusion

In this paper, we proposed a novel path-planning and obstacle avoidance algorithm GODZILA for navigation in unknown environments. The GODZILA algorithm does not require any prior knowledge of the environment. Furthermore, the algorithm follows a purely local approach using only the current range sensor measurements at each sampling instant and requiring only a small number of stored variables in memory. No map of the environment is built during navigation. This minimizes the memory and computational requirements for implementation of the algorithm, a feature that is especially attractive for small autonomous vehicles. The algorithm utilizes three components: an optimization algorithm, a local straight-line path planner to visible targets, and random navigation. It was proved, for navigation in any finite-dimensional space, that the path-planning algorithm converges in finite time with probability 1. The performance of the algorithm was demonstrated through simulations. The algorithm features a number of design functions that are free to be picked. It is a topic of further research to characterize the precise effect of these design functions on the algorithm behavior and to investigate the use of these functions to satisfy additional performance requirements.

References

[1] V. J. Lumelsky and A. A. Stepanov, "Dynamic path planning for a mobile automaton moving with limited information on the environment," *IEEE Transactions on Automatic Control*, vol. 31, no. 11, pp. 1058–1063, Nov. 1986.

[2] A. Sankaranarayanan and M. Vidyasagar, "Path planning for moving a point object amidst unknown obstacles in a plane: a new algorithm and a general theory for algorithm development," in *Proceedings of the IEEE Conference on Decision and Control*, Honolulu, HI, Dec. 1990, pp. 1111–1119.

[3] I. Kamon and E. Rivlin, "Sensory-based motion planning with global proofs," *IEEE Transactions on Robotics and Automation*, vol. 13, no. 6, pp. 814–822, Dec. 1997.

[4] C. Hull, "The goal-gradient hypothesis applied to some "field-force" problems in the behavior of young children," *The Psychological Review*, vol. 45, no. 4, pp. 271–299, July 1938.

[5] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *Proceedings of the IEEE International Conference on Robotics and Automation*, March 1985, pp. 500–505.

[6] E. Rimon and D. E. Koditschek, "Exact robot navigation using artificial potential functions," *IEEE Transactions on Robotics and Automation*, vol. 8, no. 5, pp. 501–518, Oct. 1992.

[7] V. Lumelsky, "On the connection between maze-searching and robot motion planning algorithms," in *Proceedings of the IEEE Conference on Decision and Control*, Austin, TX, Dec. 1988, pp. 2270–2275.

[8] K. I. Trovato and L. Dorst, "Differential A^*," *IEEE Transactions on Knowledge and Data Engineering*, vol. 14, no. 6, pp. 1218–1229, Dec. 2002.

[9] J. Y. Hwang, J. S. Kim, S. S. Lim, and K. H. Park, "A fast path planning by path graph optimization," *IEEE Transactions on Systems, man, and cybernetics*, vol. 33, no. 1, pp. 121–129, Jan. 2003.

[10] R. Zapata and P. Lepinay, "Flying among obstacles," in *Proceedings of the European Workshop on Advanced Mobile Robots*, Sept. 1999, pp. 81–88.

[11] J. Borenstein and Y. Koren, "Real-time obstacle avoidance for fast mobile robots," *IEEE Transactions on Systems, man, and cybernetics*, vol. 19, no. 5, pp. 1179–1187, Sep./Oct. 1989.

[12] J.-O. Kim and P. Khosla, "Real-time obstacle avoidance using harmonic potential functions," in *Proceedings of the IEEE International Conference on Robotics and Automation*, Sacramento, CA, Apr. 1991, pp. 790–796.

[13] H. J. S. Feder and J. Slotine, "Real-time path planning using harmonic potentials in dynamic environment," in *Proceedings of the IEEE International Conference on Robotics and Automation*, Albuquerque, NM, Apr. 1997, pp. 874–881.

[14] S. A. Masoud and A. A. Masoud, "Motion planning in the presence of directional and regional avoidance constraints using nonlinear, anisotropic, harmonic potential fields: a physical metaphor," *IEEE Transactions on Systems, man, and cybernetics*, vol. 32, no. 6, pp. 705–723, Nov. 2002.

[15] S. Akishita, S. Kawamura, and K. Hayashi, "New navigation function utilizing hydrodynamic potential for mobile robot," in *Proceedings of the IEEE International Workshop on Intelligent Motion Control*, Istanbul, Turkey, Aug. 1990, pp. 413–417.

[16] G. F. Marshall and L. Tarassenko, "Robot path planning using VLSI resistive grids," in *Proceedings of the International Conference on Artificial Neural Networks*, Brighton, UK, May 1993, pp. 163–167.

[17] L. Kavraki and J. C. Latombe, "Randomized preprocessing of configuration space for fast path planning," in *Proceedings of the IEEE International Conference on Robotics and Automation*, San Diego, CA, May 1994, pp. 1764–1771.

[18] F. Lamiraux and J. P. Laumond, "On the expected complexity of random path planning," in *Proceedings of the IEEE International Conference on Robotics and Automation*, Minneapolis, MN, April 1996, pp. 3014–3019.