A Hybrid Control Strategy for Autonomous Navigation while Avoiding Multiple Obstacles at Unknown Locations

Vijay Muthukumaran[1], Ricardo G. Sanfelice[2] and Gabriel H. Elkaim[3]

Abstract— **We consider the problem of steering an autonomous robot to a target position in the presence of multiple obstacles. We propose a modification of the potential field method where the robot is repulsed from an obstacle. We use a Lyapunov-based hybrid state-feedback controller that, using logic, executes an algorithm suitable for steering a point-mass robot to the target while avoiding multiple unknown obstacles. Global convergence and stability properties induced by the closed-loop hybrid controller are verified in simulations.**

I. INTRODUCTION

Autonomous navigation that involves control of unmanned ground vehicles, robot manipulators, unmanned aerial vehicles, among others has gained popularity in the last decade due to its applicability in rescue, surveillance, and discovery. Autonomous control requires trajectory generation with obstacle detection and avoidance.

Collision avoidance of obstacles has been a common area of research with extensive work done using model predictive control, neural networks and reinforcement learning as seen in [14], [15] and [16]. The problem with the above methods is the requirement for heavy computations to provide efficient solutions. One method that requires low computational power is the artificial potential field [7], [8], [17], which generates a collision-free trajectory from any initial position. This method has been shown to lead to a desirable solution with collision avoidance as seen in cases of manipulators [7], autonomous cars [9], mobile robots [3], swarm robots [5], aircraft [8], and underwater vehicles [10]. In the potential field method, a repulsive potential is created when the robot is closer to the obstacle leading to its velocity vector being pushed away from the obstacle. A common problem with this method is the presence of a saddle point at which the controller applies a control signal that prevents leaving that point, or in the presence of noise, a neighborhood of it; see [3]. Existing solutions to this problem include using random walks [12], rotational velocity vectors [8], random potential fields [13], and hysteresis switching [11]. While these solutions solve the saddle point problem, they require the robot to have a full understanding of its environment and cannot deal with dynamic changes such as the introduction of new obstacles.

We propose a new method which involves logic and the inherent property of the obstacle detection region to generate a solution which is easily implementable and does not require the robot to have a pre-existing knowledge of the location of the static obstacles. We exploit the ideas behind the Lyapunov-based hybrid state feedback controllers in [3] and sample-and-hold hybrid feedback in [4]. Note that these ideas have not been verified for obstacles with dynamic motion. In the first part of the paper, we present the model of the robot, the obstacle, and the environment. Then, provide a hybrid feedback controller that uses the repulsive nature of the potential field to construct a barrier that ensures that the robot avoids the detected obstacle and does not get stuck at saddle points. More precisely, we show using the ideas in this paper that the robot can successfully evade multiple obstacles of unknown locations. By simulation, we validate the algorithm and show that using alternative collision avoidance methods, such as modified rotational velocity fields around obstacles in [8].

II. PRELIMINARIES

A. Notation

The n-dimensional Euclidean space is denoted as $\mathbb{R}^n$. The set of natural numbers and 0 is denoted as $\mathbb{N}$. For any two vectors x, y, we equivalently write (x, y) and $[x^\top y^\top]^\top$. The set with two elements $\{1, 2\}$ is denoted as Q. $\mathcal{C}$ denotes the set $[\underline{c}, \overline{c}]$ where $0 < \underline{c} \le \overline{c} \in \mathbb{R}$. $\mathcal{Z}$ denotes the set $\mathbb{R}^2 \times \mathbb{R}^2 \times \mathbb{N} \times \mathcal{C} \times Q$. $\mathcal{A}$ denotes the set to render asymptotically stable and is the target set. $\mathbb{B}$ denotes the unit ball in Euclidean space centered at the origin. The i^{th} obstacle is denoted as $i \in \mathbb{N}$. $\mathcal{N}$ denotes the set $\{1, 2, \ldots, N\}$ where $N \in \mathbb{N}$. For $a, b \in \mathbb{R}^2$, $a = (a_1, a_2)$ and $b = (b_1, b_2)$, the distance function $d(a, b)$ is defined as the Euclidean distance from a to b given as $d(a, b) = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2}$. The parameter denoting the minimum distance between obstacles is $\rho > 0$. The hybrid system is denoted as $\mathcal{H}$ and its flow set by C, jump set by D, flow map by F, jump map by G.

B. Hybrid Systems Framework

We employ the hybrid system framework introduced in [1], and [6], where solutions are defined on *hybrid time domains*[1]. In general, a hybrid system $\mathcal{H}$ has data (C, F, D, G). F

[1]Vijay Muthukumaran is a graduate student from the Computer Engineering department at University of California, Santa Cruz. vmuthuku@ucsc.edu

[2]Ricardo G. Sanfelice is the head of Hybrid Systems Lab at University of California, Santa Cruz. Research by R. G. Sanfelice has been partially supported by the National Science Foundation under Grant no. ECS-1710621 and Grant no. CNS-1544396, by the Air Force Office of Scientific Research under Grant no. FA9550-16-1-0015, Grant no. FA9550-19-1-0053, and Grant no. FA9550-19-1-0169, and by CITRIS and the Banatao Institute at the University of California. ricardo@ucsc.edu

[3]Gabriel H. Elkaim is the head of Autonomous Systems Lab at University of California, Santa Cruz. elkaim@soe.ucsc.edu

[1]Solutions to a hybrid system are parameterized by flow time t and the number of jumps j, which are treated as independent variables. A hybrid arc ξ is a solution to the hybrid system $\mathcal{H}$ if for all $j \in \mathbb{N}$ and almost all t such that $\xi(t, j)$ is the value of the solution at $(t, j) \in \operatorname{dom} \xi$.

represents the set-valued flow map which defines the continuous dynamics on the flow set C, and the set-valued map G defines the discrete dynamics on the jump set D. A hybrid system $\mathcal{H}$ with state χ is written by

$$\mathcal{H}: \begin{cases} \dot{\chi} \in F(\chi) & \chi \in C \\ \chi^+ \in G(\chi) & \chi \in D \end{cases}$$

See [1], and [6] for more details.

III. Problem Description and Models

We consider the problem of steering an autonomous robot to the desired location in the presence of multiple obstacles using a Lyapunov-based state feedback hybrid controller. We assume the existence of a function $V : \mathbb{R}^2 \to \mathbb{R}_{\geq 0}$ such that the gradient of V has a global minimum at the desired target location and creates large potential around each obstacle. The location of the obstacles are unknown and is detected online, during navigation using onboard sensors.

The robot model is given by a point-mass model (see [2], [3]). Obstacles are represented as a set of points in $\mathbb{R}^2$ that can only be sensed locally and are separated from each other by a finite distance. The location of the obstacles is not known to the robot a priori. Measurements from the sensor that locally detects obstacle are instantaneous. Multiple obstacles can be detected locally at any given instant. Obstacles close to each other are treated as a single obstacle. The target location is a static point in $\mathbb{R}^2$ and is far enough from obstacles.

A. Robot Model

The state of the robot is given by $x = (x_1, x_2) \in \mathbb{R}^2$, where (x_1, x_2) is the planar position of the robot in Cartesian coordinates. The dynamics of the robot are given by

$$\dot{x} = \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} \tag{1}$$

where $u = (u_1, u_2) \in \mathbb{R}^2$ is the input to the robot. This simplified point-mass model allows us to focus on obstacle avoidance and is considered reasonable for navigation as observed in [2], [3].

B. Obstacle Model, Obstacle Detection, and Target Set

For simplicity in explaining the algorithm, the obstacles are assumed to be set of points and the location of the i^{th} obstacle in Cartesian coordinates is defined as $o_i = (o_{i1}, o_{i2}) \in \mathbb{R}^2$ The obstacles are assumed to be contained within the set $o_i + b_o\mathbb{B}$, where b_o defines the radius of the disk containing the obstacles, as depicted in Fig. 1. The region around the robot defining where obstacles can be detected is $x + c\mathbb{B}$, with $c \in \mathcal{C}$ is a tunable obstacle detection radius around the robot and is initialized to $c = c_o \in \mathcal{C}$. See also Fig. 1. The obstacles are located at fixed locations which are unknown until the robot is capable of detecting them, which is when

$$(o_i + b_o\mathbb{B}) \cap (x + c\mathbb{B}) \neq \emptyset \tag{2}$$

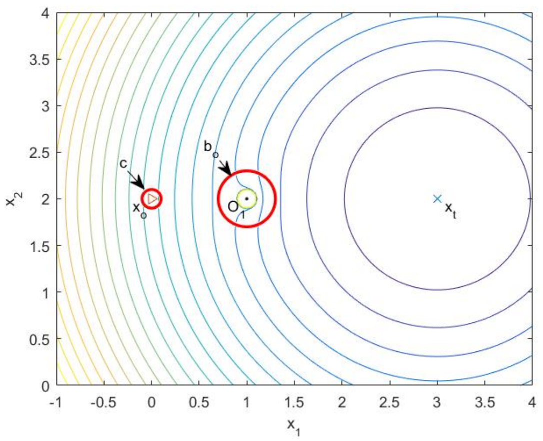

Fig. 1. The robot is at x_o with obstacle detection radius c. The target position is at x_t. The obstacle position is at o_1 with obstacle ball radius b_o. The repulsive potential function seen is given by [7] and [8] and is only used for illustrative purposes.

The total number of obstacles in the system is $N \in \mathbb{N},\ N > 0$ which is also unknown to the robot. For all $i, k \in \mathcal{N}$, the obstacle locations o_i and b_o satisfy

$$d(o_i, o_k) \geq \rho \tag{3}$$

$$(o_i + b_o\mathbb{B}) \cap (o_k + b_o\mathbb{B}) = \emptyset \tag{4}$$

where $\rho > 0$. The number of obstacles detected is given by the cardinality of the set $P(x, c)$, which is defined as

$$\{i \in \mathcal{N} : (x + c\mathbb{B}) \cap (o_i + b_o\mathbb{B}) \neq \emptyset\} \tag{5}$$

where x is the state of the robot and c is the obstacle detection radius.

The target desired location is defined as $x_t = (x_{t1}, x_{t2}) \in \mathbb{R}^2$. We denote the target by the set $\mathcal{A} = \{x_t\}$, which is such that $(o_i + b_o\mathbb{B}) \cap \mathcal{A} = \emptyset$ for all $i \in \mathcal{N}$.

IV. Hybrid Control Strategy for Avoidance of Obstacles with Unknown Locations

In this section, we introduce a general hybrid control algorithm that is capable of steering the vehicle to the desired target location while reactively avoiding the unknown obstacles in its path. The tasks performed by the algorithm are summarized as follows:

- Steer robot towards the target set $\mathcal{A}$ by applying an input u to the robot that guarantees a decrease of a properly defined potential function that vanishes at $\mathcal{A}$.
- When obstacles are detected, evade them by creating barriers around the obstacle locations through the modification of the potential function.
- After the obstacle is avoided, remove the barriers and steer the robot towards the target set $\mathcal{A}$ using the law that makes the potential function decrease.
- If multiple obstacles are detected, the obstacle detection radius c is reduced until at most a single obstacle is detected.

A. Steering Law

Given a target location x_t, let us define $V_o : \mathbb{R}^2 \to \mathbb{R}$ as

$$V_o(x) = \frac{1}{2}(x_1 - x_{t1})^2 + \frac{1}{2}(x_2 - x_{t2})^2 \quad \forall x \in \mathbb{R}^2 \tag{6}$$

The feedback law given by minus the gradient of V_o steers the model of the robot in $\mathbb{R}^2$ to the target set $\mathcal{A}$. When no obstacles are detected, i.e., when the robot is in position 1 as seen in Fig. 2, such steering law is given by

$$u = -\nabla V_o(x) = -\begin{pmatrix} x_1 - x_{t1} \\ x_2 - x_{t2} \end{pmatrix} \quad \forall x \in \mathbb{R}^2 \tag{7}$$

B. Steering plus Barrier Law

When an obstacle is detected, i.e., (2) is satisfied for some i, we modify the steering law in (7) so as to avoid the detected obstacle. We exploit the ideas in [3] and when the i^{th} obstacle is detected we construct regions of the space around it and modify V_o so that minus its gradient guarantees avoidance when used as a feedback law. Specifically, when the i^{th} obstacle is detected we construct closed sets $W_1^{o_i}, W_2^{o_i} \subset \mathbb{R}^2$ such that $W_1^{o_i} \cup W_2^{o_i} = \mathbb{R}^2$. These are the wedge-looking sets observed in Fig. 2 and given by $W_s^{o_i}$ for each $s \in Q := \{1, 2\}$. These two sets are defined such that they mirror each other horizontally at opposite ends of the obstacle o_i. A minimum distance $\delta > 0$ exists from the tip of $W_s^{o_i}$ to any point on the ball $o_i + b_o\mathbb{B}$. In order to define the wedge set $W_s^{o_i}$, it is necessary to define the lines that form its boundaries, which, for each $s \in \{1, 2\}$ are given as $x_2 = x_1 + b_{si},\ x_2 = -x_1 + b_{si}$, where $x_1, x_2 \in \mathbb{R}$. The wedge set $W_1^{o_i}$ can then be defined as

$$W_1^{o_i} = \{(x_1, x_2) : x_2 \leq x_1 + b_{1i},\ x_2 \leq -x_1 + b_{1i}\}$$

and for wedge set $W_2^{o_i}$ as

$$W_2^{o_i} = \{(x_1, x_2) : x_2 \geq x_1 + b_{2i},\ x_2 \geq -x_1 + b_{2i}\}$$

To implement the above defined strategy, certain variables are introduced:

- The number of obstacles detected by the robot is given by the value of the variable $p \in \mathcal{N}$.
- The location of the current detected obstacle o_i is stored in the variable $\ell \in \mathbb{R}^2$. When multiple obstacles are detected, the value stored in ℓ remains unchanged.
- The appropriate wedge set W_1^ℓ or W_2^ℓ to be used is tracked by the variable $q \in Q$.

When a single obstacle is detected, i.e., when the robot is in position 2 as seen in Fig. 2. The wedge set W_s^ℓ is then used to create a barrier around the obstacle which is used to define the feedback law. When multiple obstacles are detected simultaneously as seen in Fig. 3, i.e., $p > 1$, the value of c is multiplied by the parameter $\eta \in (0, 1)$ so as to reduce the radius of the detection region. This is done to ensure that the robot avoids at most one obstacle at a time. With the above definition, the proposed control law is given by minus the gradient with respect to x

$$V_q(x, \ell, p, c) = \begin{cases} V_o(x) & \text{if } |P(x, c)| = 0 \\ V_o(x) + B(\tilde{d}_q(x, \ell)) & \text{if } |P(x, c)| = 1 \end{cases} \tag{8}$$

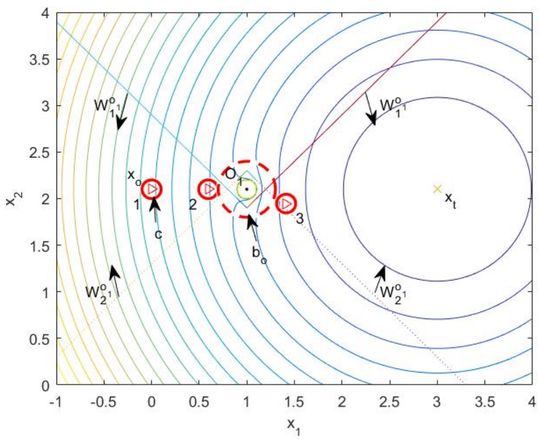

Fig. 2. On intersection with the obstacle set $o_1 + b_o\mathbb{B}$. The sets $W_1^{o_1}$, $W_2^{o_1}$ and x_t are depicted.

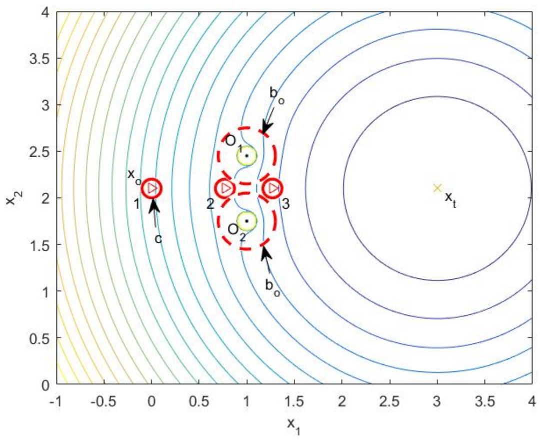

Fig. 3. Interaction with two obstacles sets $o_1 + b_o\mathbb{B}$, $o_2 + b_o\mathbb{B}$, and the subsequent avoidance of the obstacles are depicted.

where V_q and $B \circ \tilde{d}_q$ are continuously differentiable. The barrier function $B : \mathbb{R}_{\geq 0} \to \mathbb{R}$ gives large potential close to the obstacle and is defined as

$$B(y) = \begin{cases} (y - 1)^2 \ln \frac{1}{y} & \text{if } y \in [0, 1] \\ 0 & \text{if } y > 1 \end{cases} \tag{9}$$

The barrier function is evaluated at $\tilde{d}_q(x, \ell)$ along flows, which are only allowed for x on the current wedge and for each $\ell = o_i, i \in \mathcal{N}$, where $x \mapsto \tilde{d}_q(x, \ell)$ is a continuously differentiable function that measures the distance from any point in $\mathbb{R}^2$ to the boundary of the wedge sets $\overline{W_q^\ell}$.

As introduced in Section III-B, the set of points surrounding the robot $x + c\mathbb{B}$ is the obstacle detection area. When the robot has successfully evaded the obstacle as seen in position 3 in Fig. 3, the robot returns to the steering without the barrier with $c = c_o$.

C. Hybrid Feedback Closed-Loop Algorithm

The hybrid system representing the robot under the effect of the hybrid controller $\mathcal{H}$ seen in Fig. 4 is denoted as

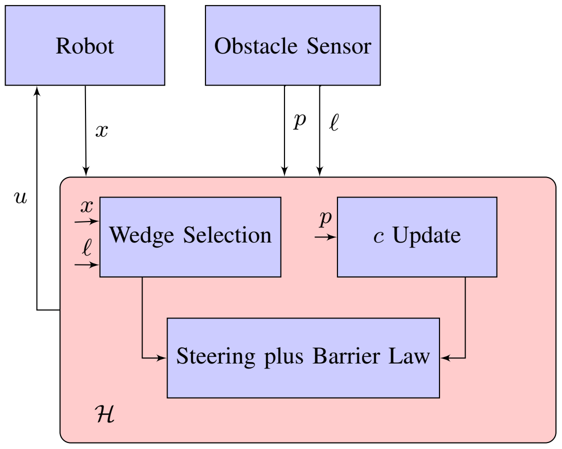

Fig. 4. Block diagram of the proposed controller.

$\mathcal{H} = (C, D, F, G)$. The controller has three inputs obtained from sensors and one control output to steer the robot. The state of the hybrid closed-loop system is denoted as $\chi = (x, \ell, p, c, q) \in \mathcal{Z}$.

1) Jumps Triggered by the Algorithm: At every jump, the logic embedded in the Wedge Selection and c Update modules are executed. The first jump set D_{1a} is defined for regions in $\mathbb{R}^2$ where the robot has detected an obstacle when previously no obstacles were detected. The jump set D_{1b} is defined for regions where multiple obstacles are detected simultaneously. Both jump sets D_{1a} and D_{1b}, contain points where specifically the detection region intersects the obstacle ball regions. Based on the logic as defined in Section V-B, the appropriate set-valued mapping G_{1a} and G_{1b} update the state χ. Then, to trigger jumps, when a single obstacle is detected, we define the set

$$D_{1a} := \{\chi \in \mathcal{Z} : |P(x, c)| = 1, p = 0\}$$

and the jump map G_{1a} to reset[2] ℓ and p as $\ell^+ = o_i,\ p^+ = |P(x, c)|$. To trigger jumps, when multiple obstacles are detected, we define the set

$$D_{1b} := \{\chi \in \mathcal{Z} : |P(x, c)| \geq 2\}$$

and the jump map G_{1b} to reset p and c as $p^+ = |P(x, \eta c)|,\ c^+ = \eta c$.

For updating the logic state q and choosing the appropriate wedge to be used for the continuous dynamics, we follow the definitions used in Section IV-A of [3]. Employing two design parameters $\mu > 1$ and $\lambda > 0$, to trigger jumps that update q we define the jump set D_2 as

$$\{\chi \in \mathcal{Z} : V_q(x, \ell, p, c) \geq (\mu - \lambda) V_{3-q}(x, \ell, p, c), p \geq 1\}$$

and to update q, we define the update law $q^+ = 3 - q$.

The jump set D_3 represents the regions where the robot has successfully evaded the obstacles following the use of the steering plus barrier law or the detection region reduction strategy where $c < c_o$. We define set-valued mappings G_3 such that jumps are triggered after avoidance of the obstacles when χ is in

$$D_3 := \{\chi \in \mathcal{Z} : |P(x, c_o)| = 0, p \geq 1\}$$

at which events, we define G_3 to update p and c according to $p^+ = |P(x, c)|,\ c^+ = c_o$. The jump set is defined as $D = D_1 \cup D_2 \cup D_3$, where $D_1 := D_{1a} \cup D_{1b}$, collecting the above definitions, the jump map is

$$G(\chi) := \begin{cases} G_{1a}(\chi) & \text{if } \chi \in D_{1a} \setminus D_2 \\ G_{1b}(\chi) & \text{if } \chi \in D_{1b} \setminus D_2 \\ G_2(\chi) & \text{if } \chi \in D_2 \setminus (D_{1a} \cup D_{1b} \cup D_3) \\ G_3(\chi) & \text{if } \chi \in D_3 \setminus D_2 \\ \{G_{1a}(\chi), G_2(\chi)\} & \text{if } \chi \in D_{1a} \cap D_2 \\ \{G_{1b}(\chi), G_2(\chi)\} & \text{if } \chi \in D_{1b} \cap D_2 \\ \{G_3(\chi), G_2(\chi)\} & \text{if } \chi \in D_3 \cap D_2 \end{cases} \tag{10}$$

2) Continuous Dynamics of the Algorithm: The flow set is given by $C = C_1 \cup C_2 \cup C_3$. The flow set C_1 is defined as the region in $\mathbb{R}^2$ with no obstacles present, i.e., $|P(x, c)| = 0$, $p = 0$, and the radius of the detection region $c = c_o$. More precisely

$$C_1 := \{\chi \in \mathcal{Z} : x \in \mathbb{R}^2, |P(x, c)| = 0, p = 0, c = c_o\}$$

When a single obstacle is detected, and after G_{1a} is executed, we follow the construction used in Section IV-B of this paper and Section IV-A of [3] for the flow set C_2, which is given as

$$\{\chi \in \mathcal{Z} : V_q(x, \ell, p, c) \leq \mu V_{3-q}(x, \ell, p, c), \\ |P(x, c)| = 1, p \geq 1\}$$

When multiple obstacles are detected, after G_{1b} is executed, the strategy used in Section V-C is implemented resulting in the reduction of the radius of detection region such that $c \leq c_o + \epsilon$ for some constant $\epsilon > 0$. The flow following this strategy is captured by the flow set C_3 defined as

$$\{\chi \in \mathcal{Z} : x \in \mathbb{R}^2, |P(x, c_o)| \geq 1, |P(x, c)| = 0, p \geq 1\}$$

The continuous dynamics are defined by the flow map

$$F(\chi, \kappa(x, \ell, p, q)) = \begin{pmatrix} \kappa(x, \ell, p, q) \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \tag{11}$$

where κ is the output of the hybrid controller which, is given by

$$\kappa(x, \ell, p, q) = -\overline{k} \nabla V_q(x, \ell, p, c) = -\overline{k} \begin{pmatrix} \frac{\partial V_q(x, \ell, p, c)}{\partial x_1} \\ \frac{\partial V_q(x, \ell, p, c)}{\partial x_2} \end{pmatrix} \tag{12}$$

[2]The update law of variables that remain constant at jumps are omitted.

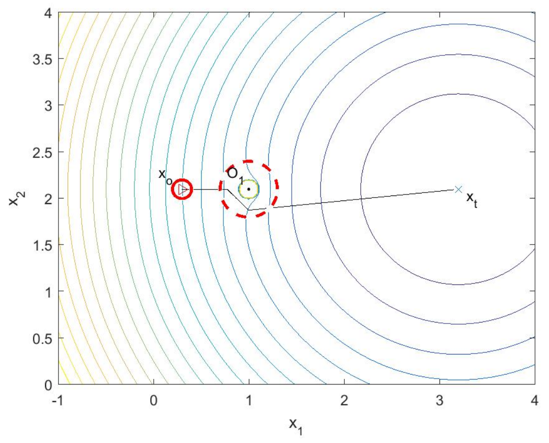

Fig. 5. The convergence of the robot to the target position with avoidance of one obstacle.

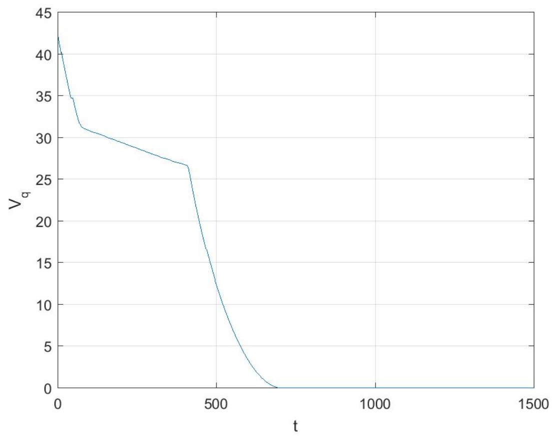

Fig. 6. The value of the Lyapunov function V_q gradually decreases to 0 as $x \to x_t$.

where, $\overline{k} > 0$, and for each $k \in \{1, 2\}$

$$\frac{\partial V_q(x,\ell,p,c)}{\partial x_k} = (x_k - x_{tk}) + 2(\tilde{d}_q(x,\ell) - 1)\frac{\partial \tilde{d}_q(x,\ell)}{\partial x_k}\ln\frac{1}{\tilde{d}_q(x,\ell)} - \frac{1}{\tilde{d}_q(x,\ell)}(\tilde{d}_q(x,\ell)-1)^2\frac{\partial \tilde{d}_q(x,\ell)}{\partial x_k} \quad (13)$$

3) Closed-Loop System: It can be shown that the controller globally steers the robot to the target set $\mathcal{A}$. Consider the function V_q in (8). When the robot position is in the subsets of the flow set C_1 or C_3, the gradient of V_q is given as

$$\nabla_x V_q(x,\ell,p,c) = \begin{pmatrix} x_1 - x_{t1} \\ x_2 - x_{t2} \end{pmatrix} \quad \forall x \in \mathbb{R}^2 \quad (14)$$

V. Numerical Verification

In this section, we show the successful convergence of the robot to the target position x_t for four different scenarios[3].

A. Scenario 1: Single Obstacle

This simulation is conducted for a single obstacle o_1 with robot starting position at x_o. The robot converges to x_t when the obstacle ball set $o_1 + b_o\mathbb{B}$ is detected. The robot position is then determined by the steering plus barrier law such that the trajectory of the robot is pushed below the obstacle o_1 as defined by the wedge $W_1^{o_1}$, the distance to which, is used as the input to the barrier function B. Once, the obstacle is evaded, the robot position converges to the target position x_t with gradually decreasing V_q and can be observed in Fig. 5, 6.

B. Scenario 2: Two Obstacles

There are two obstacles present in Fig. 7 at positions o_1 and o_2. As the robot converges to x_t, the first obstacle is detected first and wedge set $W_1^{o_1}$ is used as the input to the steering plus barrier law to avoid the obstacle. After avoiding the first obstacle, the steering causes the detection of the second obstacle, wherein wedge set $W_2^{o_2}$ is used as input to the steering plus barrier law for avoidance. After both obstacles are avoided, the robot converges to x_t as observed in Fig. 7.

[3]Source Code- github.com/HybridSystemsLab/HybridAvoidanceControl

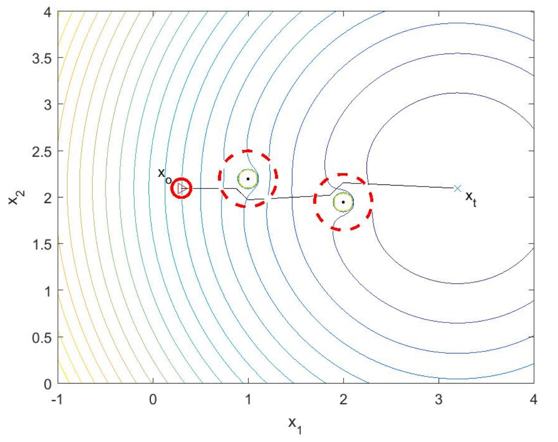

Fig. 7. The robot starts converging to the target position when obstacle o_1 is detected and the robot is pushed downwards due to the barrier formed by wedge set $W_1^{o_1}$. Similarly, for obstacle o_2, the robot is pushed upwards due to wedge set $W_2^{o_2}$.

C. Scenario 3: Four Obstacles

In this scenario, four obstacles are placed close to each other. Obstacle o_1 is detected first, and the wedge set $W_1^{o_1}$ is used for the steering plus barrier law. The steering that follows allows for detection of the subsequent obstacles o_2, o_3, o_4. Each obstacle is evaded successfully following the steering plus barrier law which uses the wedge set $W_1^{o_1}$ as the input to the barrier function B and can be observed in Fig. 8.

D. Scenario 4: Detection Region Reduction

In Fig. 9, the location of the obstacles o_1, o_2 are detected simultaneously, i.e., robot position is in D_{1b}. The detection

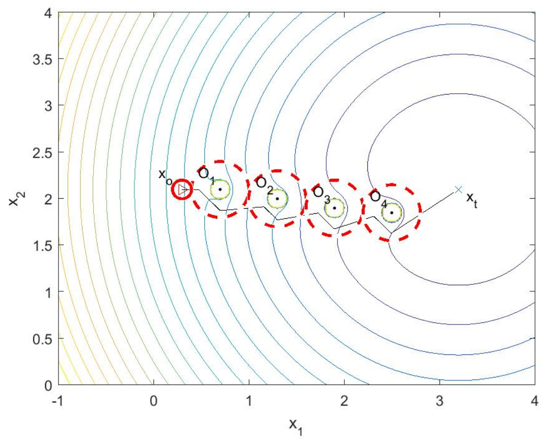

Fig. 8. The robot detects obstacle o_1, o_2, o_3, and o_4 with $q = 1$ thus using the wedge $W_1^{o_i}$ for the steering plus barrier law.

region $x + c\mathbb{B}$ is reduced by η in order to allow for flow. In this scenario, the detection region is reduced and the robot moves forwards until the obstacles are detected again which causes the detection region to be reduced further. This process keeps repeating until no obstacle is detected.

VI. CONCLUSIONS

We have shown and numerically verified that a hybrid controller can be used for autonomous navigation with avoidance of multiple obstacles where locations of the obstacle are not known priorly. This work can be expanded to three dimensions and the motion of the robot determined using Dubin's model.

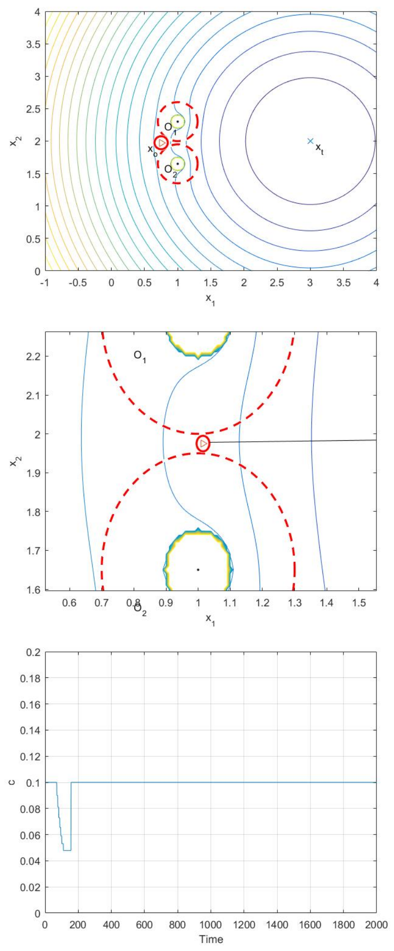

Fig. 9. The figure on the top shows the detection of obstacles o_1, o_2 simultaneously and the middle figure shows the reduction in obstacle detection region $x + c\mathbb{B}$ around the robot until the existence of a solution. The bottom figure shows the reduction in c when the obstacles are detected, c returns to its original value when the obstacles are evaded.

REFERENCES

[1] R. Goebel, R.G. Sanfelice, and A.R. Teel, Hybrid Dynamical Systems: Modeling, Stability, and Robustness, 2012, Princeton University Press.

[2] C. G. Mayhew, R. G. Sanfelice and A. R. Teel, Robust Source-Seeking Hybrid Controllers for Autonomous Vehicles, 2007 American Control Conference, New York, NY, 2007, pp. 1185-1190.

[3] R. G. Sanfelice, M. J. Messina, S. Emre Tuna and A. R. Teel, "Robust hybrid controllers for continuous-time systems with applications to obstacle avoidance and regulation to disconnected set of points," 2006 American Control Conference, Minneapolis, MN, 2006, pp. 6 pp.-.

[4] F.H. Clarke, Yu. S. Ledyaev, L. Rifford, and R.J. Stern, Feedback stabilization and Lyapunov functions, SIAM J. Control Optimization, 39(1):25 to 48, 2000.

[5] G. H. Elkaim and R. J. Kelbley, "A lightweight formation control methodology for a swarm of non-holonomic vehicles," 2006 IEEE Aerospace Conference, Big Sky, MT, 2006, pp. 8 pp.-.

[6] R. Goebel and A.R. Teel, Solutions to hybrid inclusions via set and graphical convergence with stability theory applications, Automatica, 42(4):573-587, 2006

[7] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," Proceedings. 1985 IEEE International Conference on Robotics and Automation, St. Louis, MO, USA, 1985, pp. 500-505.

[8] H. Rezaee and F. Abdollahi, "Adaptive artificial potential field approach for obstacle avoidance of unmanned aircrafts," 2012 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM), Kachsiung, 2012, pp. 1-6.

[9] D. Reichardt and J. Shick, "Collision avoidance in dynamic environments applied to autonomous vehicle guidance on the motorway," Proceedings of the Intelligent Vehicles '94 Symposium, Paris, France, 1994, pp. 74-78.

[10] C. Petres, Y. Pailhas, P. Patron, Y. Petillot, J. Evans and D. Lane, "Path Planning for Autonomous Underwater Vehicles," in IEEE Transactions on Robotics, vol. 23, no. 2, pp. 331-341, April 2007.

[11] M. S. Branicky, "Multiple Lyapunov functions and other analysis tools for switched and hybrid systems," in IEEE Transactions on Automatic Control, vol. 43, no. 4, pp. 475-482, April 1998.

[12] M. W. Spong, S. Hutchinson, and M. Vidyasagar, Robot Modeling and Control, 1st ed. John Wiley and Sons, 2005

[13] Michael A. Goodrich, 2000, Potential fields tutorial

[14] X. Wang, V. Yadav and S. N. Balakrishnan, "Cooperative UAV Formation Flying With Obstacle/Collision Avoidance," in IEEE Transactions on Control Systems Technology, vol. 15, no. 4, pp. 672-679, July 2007.

[15] Il-Kyun Jung, Ki-Bum Hong, Suk-Kyo Hong and Soon Chan Hong, "Path planning of mobile robot using neural network," ISIE '99. Proceedings of the IEEE International Symposium on Industrial Electronics (Cat. No.99TH8465), Bled, Slovenia, 1999, pp. 979-983 vol.3.

[16] Aranibar, D. B. and Alsina, P. J. (2004). Reinforcement learning-based path planning for autonomous robots. In EnRI - XXIV Congresso da Sociedade Brasileira de Computaçaõ , pp. 10, Salvador

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