2013 European Control Conference (ECC)
July 17-19, 2013, Zürich, Switzerland.

Waypoint Trajectory Planning in the Presence of Obstacles with a Tunnel-MILP approach*

Rubens Junqueira Magalhães Afonso, Roberto Kawakami Harrop Galvão and Karl Heinz Kienitz[1]

Abstract— **This work presents a waypoint trajectory planning technique for an autonomous vehicle in the presence of obstacles using a tunnel-MILP formulation for the avoidance constraints. Predictive Control is used to address the issues of dynamic constraint satisfaction and obstacle avoidance. However, the complexity of the optimization problem to be solved may escalate as the number of obstacles increases. To circumvent this issue, a tunnel-MILP approach is employed. Even so, the optimization problem may still be too complex, i. e., involve a large number of decision variables, to be computationally tractable within the relatively small sample time required by vehicle guidance applications. The number of decision variables is reduced via the pre-computation of waypoints during an off-line trajectory planning phase. In this manner, during the on-line control phase, the optimization problem to be solved needs only to compute a control solution to reach the next waypoint in the sequence, instead of the whole control solution to reach the target set from the current position. Simulation results are presented to show that the employment of the waypoint trajectory planning technique brings about benefits regarding the computational burden associated to the solution of the on-line optimal control problem.**

Index Terms— **Predictive control, trajectory planning, waypoint, tunnel-MILP.**

I. Introduction

Typical vehicle guidance missions usually require that the state-vector of the vehicle reaches a given terminal set in finite time while avoiding collisions with obstacles. Moreover, fuel expense minimization and dynamical constraints enforcement are other issues that should be faced by a successful trajectory planning/control framework candidate, particularly for Unmanned Aerial Vehicles (UAVs) guidance missions [1].

The classical approach to guidance of vehicles has been to divide the task in three layers [1]: path planning, which encompasses the search for a set of positions that the vehicle must occupy in order to go from the initial position to the goal (possibly including avoidance of collision with obstacles), the search for a feasible trajectory that respects this path and the dynamic constraints of the vehicle and, finally, a feedback control strategy to make the vehicle follow the planned trajectory and account for modeling errors, external disturbances and possible faults.

However, different approaches have been developed which merge the first two phases, incorporating the dynamic constraint satisfaction and obstacle avoidance eliminating the path planning phase and performing trajectory planning accounting for these issues. In [2] a navigation function is developed to cope with the nonlinear dynamics of the vehicles and the obstacle avoidance constraints, allowing the use of a gradient descent method for optimization. Other methods have used *Rapidly-exploring Random Trees* (RRTs) to search the state-space respecting the dynamic and obstacle avoidance constraints [3], [4], [5]. In [6], a randomized, incremental motion-planning algorithm was proposed to deal with dynamic constraints and avoidance of collision with obstacles, both fixed and moving. An interpolation algorithm is used to generate a continuous family of vehicle maneuvers from classes of user-provided motion examples while enforcing nonlinear system equations of motion as well as nonlinear equality and inequality constraints in [7].

Model-based Predictive Control (MPC) approaches inherently address some of these issues due to their nature of obtaining a control signal by solving a constrained optimal control problem at every sample instant [8], [9]. For instance, a cost function which penalizes the variables associated to fuel expense may be used to produce a control solution minimizing the fuel consumption to carry out a particular maneuver. As for the dynamical constraints, they can be imposed as constraints over the state and/or control variables. Therefore, MPC arises as one potential candidate for the control and trajectory planning of UAVs.

A number of approaches employing MPC for trajectory planning in the presence of constraints have been developed for autonomous vehicles. In [10] nonlinear MPC was used for trajectory generation for an Unmanned Aerial Vehicle (UAV). Employing truncated Taylor series, a closed-form solution for the optimal control problem with nonlinear MPC was allowed in [11] for UAVs. In [12], the trajectory planning was posed as a constrained optimization problem in the output space using the differential flatness property of the motion equations of the vehicle. More recently, a counter-hijack system was developed in [13] which avoids collisions with buildings and other relevant structures (such as power plants) by modeling these as no-fly zones in a trajectory planning problem and a dualization method was used to permit the employment of a gradient descent technique to solve the nonlinear optimization problem.

A variable-horizon MPC formulation with terminal set and obstacle avoidance constraints was proposed in [14]. The resulting optimization problem involves a linear cost function, subject to linear inequalities. Nevertheless, the employment of a variable horizon includes discrete variables

*This work was supported by FAPESP (grants 2011/18638-8 and 2011/17610-0) and CNPq (research fellowships)

[1]R. Afonso, R. Galvão and K. Kienitz are with Instituto Tecnológico de Aeronáutica, Divisão de Engenharia Eletrônica, 12228-900 São José dos Campos, SP, Brasil Emails: rubensjm@ita.br, kawakami@ita.br, kienitz@ita.br

in the optimization problem. Thus, the resulting problem is a *Mixed Integer Linear Programming* (MILP) one, involving both continuous (control values) variables and discrete (the horizon) ones. The avoidance constraints, for their part, make the optimization problem non-convex. This issue is circumvented through the employment of some additional binary variables and can also be encoded as a MILP problem, in a method called "big-M".

The ability to cope with all these issues makes the MPC formulation for trajectory planning and control in [14] an interesting candidate for a guidance framework. On the other hand, the computational burden associated with the solution of a MILP problem involving a large number of decision variables may make it intractable within the relatively small sample time required by UAV applications.

For instance, the number of binary variables scales very quickly with the number of obstacles. Typically, one binary variable is required to account for each side of one obstacle at every step along the prediction horizon. Therefore, with a horizon N, the addition of an obstacle with N_f sides requires $N \times N_f$ additional binary variables to avoid this obstacle. In [16], the number of binary variables necessary to avoid obstacles is reduced by using a pre-planned path to identify a sequence of convex polytopes forming a tunnel that connects the initial position of the vehicle to the terminal set. Afterwards, the MPC formulation of [14] is employed, but the avoidance constraints are replaced by others that constrain the position of the vehicle to remain within one of the convex polytopes forming the tunnel. The technique proposed in [16] is termed tunnel-MILP. Recent papers describe ways of implementing the obstacle avoidance constraints with less binary variables, reducing the number of binary variables from N_f to $\log_2 N_f$ to implement the avoidance constraints for a single obstacle [17] [18]. However, this can also be explored with the tunnel-MILP approach, reducing the number N_T of variables to represent the avoidance constraints to $\log_2 N_T$. Therefore, the reduction of binary variables with employment of the tunnel-MILP is still meaningful.

Despite this effort to reduce the number of binary variables associated to the avoidance constraints, the main source of a large number of decision variables is still not addressed: if the horizon to reach the terminal set is large, a great number of decision variables results and the optimization problem to be solved in order to produce the control signals may be computationally intractable within the sample time. This is due to the fact that at every sample instant the whole trajectory from the current state to the terminal set is recomputed. This issue is addressed in [15] by using a smaller horizon for the planning phase and a terminal cost obtained by estimating the time to reach the terminal point via a graph search performed on a Visibility Graph. Another approach to circumvent this difficulty was proposed in [19] through the division of the mission in tasks that can be performed within smaller horizons. The division makes use of a pre-planned path composed of a sequence of straight-line segments connecting the initial position of the vehicle to the terminal set while avoiding obstacles. Such a path is used to constrain the position of a pre-defined number of waypoints, which are determined during a trajectory planning phase. They are calculated so that the vehicle can reach the first waypoint from the initial position within a pre-determined horizon, then the second waypoint from the first and so on, until reaching the last waypoint, from which it is possible to reach the target within the same small horizon. These waypoints are subsequently passed to a decision component which selects which of them is the active target and passes it to the control layer. Therefore, since it is possible to reach this target within a small horizon, this horizon is the one employed for the optimization problem solved to obtain the control signals.

In this paper a combination of the tunnel-MILP [16] and waypoint trajectory planning [19] techniques is proposed. The tunnel-MILP algorithm requires a pre-planned path, as does the waypoint trajectory planning. Moreover, the tunnel formed of convex polytopes can be used to replace the obstacle avoidance constraints and is also ideal to be used with the waypoint trajectory planning, since the trajectory will remain close to the pre-planned path because the waypoints are constrained to it. Thus, in a certain way, it is expected that the planned trajectory will obey the tunnel constraints with no or slight modifications to the control solution. The final result should be that the on-line control problem will have fewer decision variables, which, in turn, will allow the optimization to be accomplished at a faster rate.

The remainder of this paper is organized as follows. In Section II, the trajectory planning architecture employed in the present work is presented. The main contribution of this work is introduced in Section III, in which a MILP problem integrating the waypoint trajectory planning technique proposed in [19] with the tunnel-MILP approach proposed in [16] is formulated. The simulation scenarios are described in Section IV. Section V presents the simulation results. Conclusions are drawn and suggestions for future work are given in section VI.

A. Notation

- $x \in \mathbb{R}^n$: plant state;
- $x_0 \in \mathbb{R}^n$: initial plant state;
- $u \in \mathbb{R}^p$: control signal;
- $r \in \mathbb{R}^2$: vehicle position;
- $b_{i,l}^{WP} \in \{0,1\}$: binary variables associated to the positions of the waypoints;
- $b_m^t \in \{0,1\}$: binary variables associated to the tunnel-MILP constraints;
- k: current time;
- $\hat{\diamondsuit}(k+i|k)$: predicted value of the variable $\diamondsuit$ at time $k+i$ based on the information available up to time k;
- $N(k) \in \mathbb{N}$: MPC control and prediction horizon;
- $C_r \in \mathbb{R}^{2\times n}$: matrix that extracts position information from the state vector;
- $\mathbb{U} \subset \mathbb{R}^p$: set of admissible control values;
- $\mathbb{X} \subset \mathbb{R}^n$: set of admissible state values;
- $\mathbb{Q} \subset \mathbb{R}^n$: set of terminal state values at the end of the horizon;

- $\mathcal{Z}_m \subset \mathbb{R}^2$: polygon defining the m-th obstacle;
- $\mathcal{T}_j \subset \mathbb{R}^2$: polygon defining the j-th convex polytope of the tunnel;
- $V_i \in \mathbb{R}^2$: i-th vertex in the planned path;
- $\bar{N} \in \mathbb{N}$: maximal horizon in the one-step formulation;
- $\bar{N}_P \in \mathbb{N}$: maximal horizon between waypoints;
- $N_{WP} \in \mathbb{N}$: number of waypoints;
- $N_{obs} \in \mathbb{N}$: number of obstacles;
- $N_f \in \mathbb{N}$: number of sides in each obstacle;
- $N_V \in \mathbb{N}$: number of vertices in the planned path;
- $N_T \in \mathbb{N}$: number of convex polytopes in the tunnel;
- $\alpha_i \in \mathbb{R}$: variable that determines the position of the i-th waypoint along the planned path;
- $M_{WP} \in \mathbb{R}_+$: constant large enough to make waypoint location constraints inactive;
- $M_m^t \in \mathbb{R}_+$: constant large enough to make tunnel-MILP constraints inactive;
- $r_x \in \mathbb{R}$: position along a coordinate axis in a horizontal plane regarding an arbitrary origin;
- $r_y \in \mathbb{R}$: position along a coordinate axis (perpendicular to the first) in a horizontal plane regarding an arbitrary origin;
- $v_x \in \mathbb{R}$: velocity associated to r_x;
- $v_y \in \mathbb{R}$: velocity associated to r_y;
- $a_x \in \mathbb{R}$: acceleration associated to v_x;
- $a_y \in \mathbb{R}$: acceleration associated to v_y;
- $\gamma \in \mathbb{R}$: weight of the term associated to the fuel consumption in the control cost function;
- $\gamma_p \in \mathbb{R}$: weight of the term associated to the fuel consumption in the waypoint trajectory planning cost function;

II. Trajectory planning architecture

The path planning, trajectory planning and control framework adopted herein divides the planning and control in hierarchical levels. This chain may be summarized in three different layers, as in [19]:

1) Path planner.
2) Trajectory planner.
3) Predictive Control layer.

For the purpose of obtaining a collision-free path, a number of points are taken over the faces of the obstacles represented as convex polytopes and the boundaries of the known environment, over which perfect knowledge is assumed and the movement of the vehicle is constrained to remain. Figure 1 shows these points as black stars for one arbitrary scenario. They are used to obtain a Voronoi graph [20]. Subsequently, the nodes of the graph that are inside the prohibited region are removed along with the edges that connected these nodes to the other ones in the graph. What then remains is a graph whose edges (depicted in Fig. 1 as black dotted lines) form paths that present a good property of having equal distance between the obstacles which are the closest to an edge and the edges themselves. This property comes from the nature of the Voronoi graph construction, which maximizes the minimum distance to all points. Two nodes are added to this graph: the initial position and another node inside the terminal set (any can be used, but in this work one at the center of the terminal set is arbitrarily chosen). These two additional nodes are connected to the closest nodes of the graph to each of them, which produce edges that do not cross the obstacles. This is a critical phase, because if any of these two additional nodes (initial or goal position) cannot be connect to the graph, any search on the graph for a collision-free path will fail, unless the initial position is itself the goal. Thus, the problem is considered to be inadequately described and either the obstacles, the known region, the initial or the goal position will have to be changed in order to allow for a solution. Figure 1 shows the resulting graph after the addition of the initial position (depicted as a black square) and the goal point (represented by a black circle) to the graph.

Afterwards, a graph search is performed to find the shortest path between the initial node and the final one, via an A^* search algorithm [20], [21]. The ensuing path obtained from the graph is depicted in Fig. 1 by the black continuous lines that cover the dotted lines of the portion of the graph constituting the path. The cost herein adopted is composed of the real distance cost to get to the particular node under evaluation plus the straight-line distance from this node to the final one. It is important to remark that it is not required that a collision-free straight-line exists between the evaluated node and the goal one. Therefore, this cost is a lower bound to the total cost to get from the evaluated node to the goal [20]. The adopted cost function of a node X is

$$f(X) = g(X) + h(X) \tag{1}$$

where f is the total cost, g is the cost to move from the current node to node X and h is the straight-line distance between node X and the goal node (note that the presence of obstacles is not considered in calculating h). The A^* algorithm starts at the initial node, adds all nodes connected to this one to the frontier and moves to the node with the least cost in the frontier, repeating this procedure. Any node that has already been explored is removed from the frontier. The algorithm stops either when a path is found to the goal, returning this solution or when the whole graph has been explored and no path has been found, returning *failure* in this case. If this happens, since it was possible to connect the initial and goal positions to the graph before reaching this phase, it indicates that the graph presents subgraphs which are not connected. In this case, again the problem is inadequately posed and either the obstacles, the known region, the initial or the goal position will have to be changed in order to allow for a solution.

This initial path is then filtered in order to remove some edges while maintaining it free of collisions with the obstacles, if possible, as a larger number of edges entails a larger number of decision variables, both continuous and binary, in the optimization problem solved to determine the positions of the waypoints. For this purpose, an algorithm tries to connect node i to node $i+2$. If the resulting edge is collision-free, then node $i+1$ is removed and the edges between nodes i

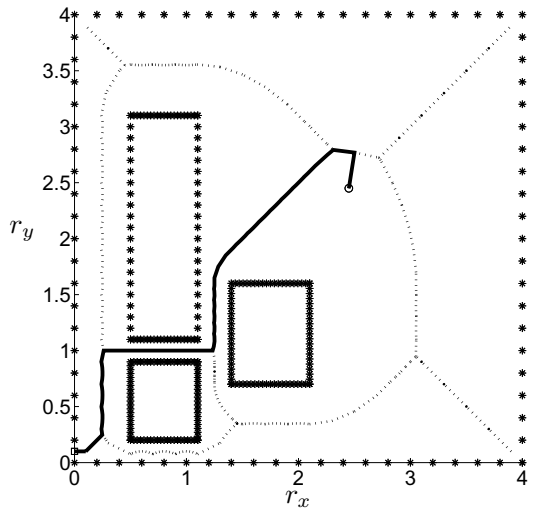

Fig. 1. Example of Voronoi graph construction (dotted lines). The stars $*$ indicate the points taken at the faces of the obstacles and boundaries of the environment for the construction of the graph. The path resulting from the A^* search performed on the Voronoi graph extended with the initial (square) and goal (circle) positions is represented by the black continuous line.

and $i+1$ and between nodes $i+1$ and $i+2$ are replaced by one between the nodes i and $i+2$. If it collides with an obstacle, then i is incremented and the algorithm is repeated for node $i+1$, until the final node is reached. This is done according to Algorithm 1 presented below.

Algorithm 1: Reduce number of edges in Path

1: $i \leftarrow 1$
2: **while** $V_{i+2} \neq Goal\_Node$ **do**
3: **if** $\mathcal{K}(\overline{V_i V_{i+2}}) == \varnothing$ **then**
4: $Path \leftarrow Remove(Path, V_{i+1})$
5: **else**
6: $i \leftarrow i+1$
7: **end if**
8: **end while**

□

In Algorithm 1, the $Path = \{V_i\}$ variable is the ordered sequence of vertices in the path from initial node to goal node, and thus, V_i is the i-th vertex in this sequence ($V_1 = Initial\_Node$); $\mathcal{K}$ is the set formed by the intersection of the straight-line segment $\overline{V_i\ V_{i+2}}$ with the set of obstacles $\bigcup_{m=1}^{N_{obs}} \mathcal{Z}_m$, $\mathcal{Z}_m = \{r | P_m^{obs} r \leq q_m^{obs},\ m = 1, \ldots, N_{obs}\}$; function $Remove(\clubsuit, \diamondsuit)$ removes the node $\diamondsuit$ from the path $\clubsuit$ and returns the reduced resulting path. By applying this filtering algorithm to the example path of Fig. 1, the reduced path shown in black continuous lines in Fig. 2 is obtained.

After that, a Delaunay triangulation of the space is computed [22]. It divides the bidimensional space into triangles that cover the whole space and whose intersection is empty, save possibly for their sides. For the triangulation, a number of points needs to be taken in the space. In the present work, these points are the vertices of the obstacles and of the known region. The resulting Delaunay triangulation is depicted in Fig. 2 for the example of Fig. 1. The sides of the resulting triangles are represented by the black dotted lines in Fig. 2. The triangles that are crossed by the planned path are computed and merged to form convex polytopes if possible, attempting to reduce the number of polytopes in the tunnel. These are used to impose the obstacle avoidance constraints as in the tunnel-MILP formulation [16]. Figure 2 shows the resulting tunnel after merging the triangles composed of the polytopes in shades of gray.

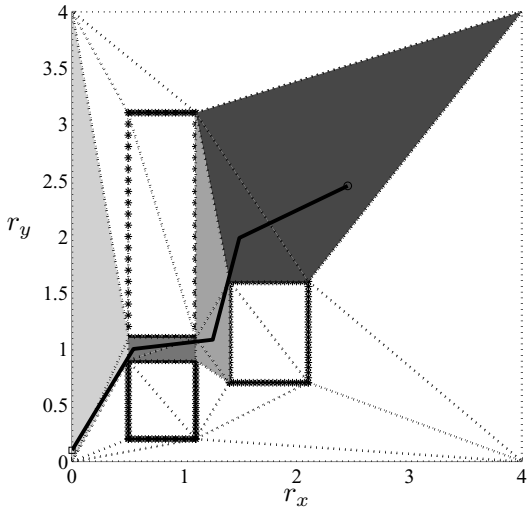

Fig. 2. Reduced path resulting from the application of Algorithm 1 to the example path of Fig. 1 (black continuous lines). The Delaunay triangulation of the scenario is represented in dotted lines and the merged polytopes crossed by the reduced path are displayed in shades of gray.

Fig. 3 presents the trajectory planning/control framework adopted in this work, which is the same as in [19]. The integration of the tunnel-MILP approach to the proposed waypoint trajectory planning technique is done in the "Trajectory planner" block. The "Active target selection logic" is responsible for commuting the current target in the following manner: if the position of the vehicle is equal (within an arbitrary numerical tolerance) to the current active target (starting from the first waypoint), then the active target is changed to the next waypoint in the sequence; otherwise, the active target remains the same. If the last waypoint is the current active target and it is reached, then the next active target will be the terminal set.

III. Trajectory planning technique

The trajectory planning technique employed in this paper builds upon the one proposed in [19]. It determines a sequence of waypoints between the initial position and the goal set. By following this sequence, the vehicle approaches the

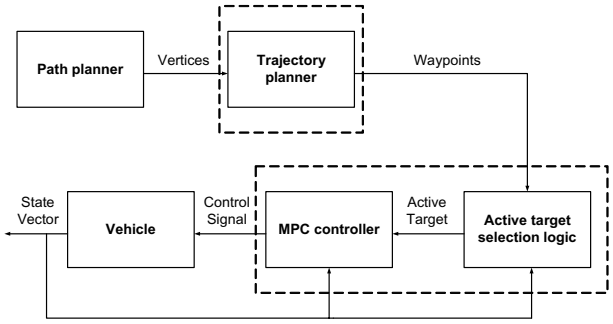

Fig. 3. Trajectory planning and control architecture used in this work.

goal set. The smallest horizon necessary to reach waypoint i from waypoint $i-1$, as well as the ones necessary to reach the first one from the initial position and the goal set from the last one, is smaller than the horizon necessary to reach the goal set directly from the initial position.

The positions of the waypoints are constrained to remain on a pre-planned path. This leads to a reduction on the complexity of the trajectory planning task, as the search space for the positions of the waypoints is simplified.

The main difference between the trajectory planner proposed in [19] and the one employed herein is the incorporation of the tunnel-MILP formulation for collision avoidance proposed in [16]. This formulation of the collision avoidance problem reduces the number of binary variables necessary to avoid obstacles by using a decomposition of the space in a sequence of convex polytopes crossed by a pre-planned path that form a tunnel from the initial position to the goal. Since this path is already used in the trajectory planner proposed in [19], the only additional effort necessary to calculate the tunnel is to divide the space in a sequence of convex polytopes and compute the ones that are crossed by the pre-planned path. This is solved by employing a Delaunay triangulation. Afterwards, the triangles that are crossed by the path are identified and, in order to reduce the number of necessary binary variables, these are united to form the smallest number of convex polytopes possible.

Let the N_T convex polytopes forming the tunnel be given as linear inequalities $\mathcal{T}_j = \{P_j^T r \leq q_j^T\},\ \ j = 1,\ldots,N_T$. Then, in order to avoid collisions with the obstacles, the position of the vehicle must remain inside one of the polytopes $\mathcal{T}_j$ at every instant. The waypoint trajectory planning problem with collision avoidance can be posed as a MILP problem as follows:

Problem 3.1: Let N_{WP}, $\bar{N}_P$, N_T, and $\{V_i\}$, $i = 1,\ldots,N_V+1$ be the preset number of waypoints, the maximal horizon to reach an waypoint from the previous one, the number of convex polytopes in the tunnel and the ordered sequence of vertices whose connection via straight-line segments produces the collision-free path (V_1 is the initial position of the vehicle), respectively. The waypoint determination problem is stated as

$$\min_{\hat{u},\,\alpha_i,\,b_m^t,\,b_{i,l}^{WP}} \sum_{i=1}^{N_{WP}} \alpha_i + \mu \sum_{i=1}^{N_{WP}} \sum_{l=1}^{N_V} l b_{i,l}^{WP} + \gamma_p \sum_{j=1}^{(N_{WP}+1)\bar{N}_P} \xi_j \tag{2}$$

s.t.

$$\begin{aligned}&\hat{r}(k+i\bar{N}_P|k) \leq (\alpha_i - N_V + l)V_l + \\ &+ [1-(\alpha_i - N_V + l)]\,V_{l+1} + M_{WP}(1-b_{i,l}^{WP}), \\ &1 \leq i \leq N_{WP},\ 1 \leq l \leq N_V\end{aligned} \tag{3a}$$

$$\begin{aligned}&-\hat{r}(k+i\bar{N}_P|k) \leq -\{(\alpha_i - N_V + l)V_l + \\ &+ [1-(\alpha_i - N_V + l)]\,V_{l+1}\} + M_{WP}(1-b_{i,l}^{WP}), \\ &1 \leq i \leq N_{WP},\ 1 \leq l \leq N_V\end{aligned} \tag{3b}$$

$$\sum_{l=1}^{N_V} (N_V - l) b_{i,l}^{WP} \leq \alpha_i,\ 1 \leq i \leq N_{WP} \tag{3c}$$

$$\begin{aligned}&\sum_{l=1}^{N_V} b_{i,l}^{WP} = 1,\ 1 \leq i \leq N_{WP} \\ &b_{i,l}^{WP} \in \{0,1\},\ 1 \leq i \leq N_{WP},\ 1 \leq l \leq N_V\end{aligned} \tag{3d}$$

$$0 \leq \alpha_{N_{WP}} \leq \alpha_{N_{WP}-1} \leq \cdots \leq \alpha_1 \leq N_V \tag{3e}$$

$$\hat{x}(k+(N_{WP}+1)\bar{N}_P|k) \in \mathbb{Q} \tag{3f}$$

$$\hat{u}(k+j|k) \in \mathbb{U},\ 0 \leq j \leq (N_{WP}+1)\bar{N}_P - 1 \tag{3g}$$

$$\hat{x}(k+j|k) \in \mathbb{X},\ 1 \leq j \leq (N_{WP}+1)\bar{N}_P - 1 \tag{3h}$$

$$P_m^t \hat{r}(k+j|k) \leq q_m^t + M_m^t(1-b_m^t(k+j|k)) \tag{3i}$$

$$\begin{aligned}&\sum_{m=1}^{N_T} b_m^t(k+j|k) \geq 1,\ b_m^t(k+j|k) \in \{0,1\}, \\ &1 \leq j \leq (N_{WP}+1)\bar{N}_P,\ 1 \leq m \leq N_T\end{aligned} \tag{3j}$$

$$\begin{aligned}&-\xi_j \leq \hat{u}(k+j|k) \leq \xi_j, \\ &0 \leq j \leq (N_{WP}+1)\bar{N}_P - 1\end{aligned} \tag{3k}$$

where μ, $\gamma_p > 0$ are scalars, $\hat{r}(k+i\bar{N}_P|k) = C_r\hat{x}(k+i\bar{N}_P|k)$ is the predicted position at sampling time $(k+i\bar{N}_P)$, and M_m^t is a vector with the same size of q_m^t, obtained by stacking a scalar $M^t > 0$ large enough to make the inequalities $P_m^t\hat{r}(k+j|k) \leq q_m^t$ inactive. Thus, instead of using $(N_{WP}+1)\bar{N}_P N_f N_{obs}$ binary variables to impose the collision avoidance inequalities as in [19], now only as many as $(N_{WP}+1)\bar{N}_P N_T$ are necessary.

The minimization of the first term of the cost function in Eq. (2) prioritizes solutions that have the waypoints farther from the initial position and closer to the terminal set, in order to avoid low initial speeds.

In order to position a waypoint between the vertices V_l and V_{l+1}, this point must lie within the segment $\overline{V_l V_{l+1}}$. This means that the point may be written as $V_{l+1} + \beta(V_l - V_{l+1})$, with $\beta \in [0,1)$. In Problem 3.1, this is done by making $\beta = \alpha_i - N_V + l$. With the maximization of $(N_V - l)b_{i,l}^{WP},\ 1 \leq l \leq N_V$ (equivalent to the minimization of the second term

in the cost function) subject to the constraint in Eq. (3c), the value of $(N_V - l)b_{i,l}^{WP}$ is the greatest integer which is smaller or equal to α_i for any positive value of the scalar μ. The constraints in Eqs. (3a), (3b) and (3d) impose that each waypoint is located at one and only one segment of the path. For this purpose, the scalar M_{WP} is chosen large enough to render the constraints in Eqs. (3a) and (3b) inactive when $b_{i,l}^{WP} = 0$.

The positions of the waypoints between the initial position and the last vertex, are uniquely determined by the values of α_i, $1 \leq i \leq N_{WP}$. Each waypoint (from 1 to N_{WP}) lies farther from the initial position than the one before, because of the constraint in Eq. (3e).

The last term of the cost function in Eq. (2) penalizes the sum of the absolute values of the control variables. This aims at minimizing the amount of actuator effort during the maneuver.

As the number of waypoints N_{WP} and the horizon between them $\bar{N}_P$ are fixed, if Problem 3.1 is infeasible, an increase either in N_{WP} or $\bar{N}_P$ is necessary, to allow for more optimization variables in order to reach the terminal set.

It is important to remark that the trajectory planner cost function used to obtain the positions of the waypoints and the control loop one are not the same. Therefore, there may be differences between the planned trajectory and the one that is actually executed. The control loop cost function employed in the present work is the same proposed in [14] and adopted in [16] and [19]. When compared to the trajectory planner cost function of Eq. (2), it can be seen that the maximal horizon in these are different ($\bar{N}$ for the control loop cost function and $(N_{WP}+1)\bar{N}_P$ for the trajectory planner cost function) and the horizon itself is not minimized in the latter. Moreover, the trajectory planner cost function involves terms related to the position of the waypoints, which are not present in the control loop cost function.

In this paper, robustness to disturbances is not considered. Nevertheless, the trajectory planning technique could be extended to cope with unknown but limited disturbances by replacing the concept of waypoints by waysets, which the state of the vehicle must reach at a given time. The waysets would be determined so that they can be reached within the desired horizon despite the disturbances. Robustness to disturbances during the control phase would then be achieved using the control formulation in [14] with the appropriate wayset as the terminal set and an active target set switching logic similar to the one used in the present paper.

IV. Simulation scenarios

A model of a particle moving in two dimensions was employed for simulation, as in previous works concerning the use of MPC and MILP for trajectory planning [14], [16], [17]. The continuous-time model equations are:

$$\dot{r}_x = v_x,\ \dot{v}_x = a_x,\ \dot{r}_y = v_y,\ \dot{v}_y = a_y \tag{4}$$

where the position of the particle in a plane with respect to an arbitrary reference frame is represented by r_x and r_y. For control purposes, it is interesting to re-cast these equations in state-space form ($\dot{x} = A_c x + B_c u$). This is done by defining the state and control vectors as $x = [r_x \quad v_x \quad r_y \quad v_y]^T$, $u = [a_x \quad a_y]^T$. The MPC approach herein used is one of discrete time. Therefore, a discrete-time model of the form $x(k+1) = Ax(k) + Bu(k)$ was obtained with

$$A = \begin{bmatrix} 1 & T & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & T \\ 0 & 0 & 0 & 1 \end{bmatrix}, B = \begin{bmatrix} 0.5T^2 & 0 \\ T & 0 \\ 0 & 0.5T^2 \\ 0 & T \end{bmatrix} \tag{5}$$

where T is the sampling period. For the simulations in the present paper T was normalized to one time unit.

The constraints imposed on the velocities and accelerations are $-1 \leq x_2, x_4 \leq 1$ and $-1 \leq u_1, u_2 \leq 1$, respectively. As means to limit the position of the vehicle to a terrain over which information was assumed to be available, constraints were also imposed on the position: $0 \leq x_1, x_3 \leq 10$.

The goal was to reach a terminal set in the form of a rectangle described by the following inequalities on the positions $2.4 \leq x_1, x_3 \leq 2.5$ from an initial state that was arbitrarily set to $x_0^T = [0\ 0\ 0.1\ 0]^T$.

The three obstacles were also represented as rectangles with $0.6 \leq x_1^1, x_1^2 \leq 1$, $0.3 \leq x_3^1 \leq 0.8$, $1.2 \leq x_3^1 \leq 3.0$, $1.5 \leq x_1^3 \leq 2.0$ and, $0.8 \leq x_3^3 \leq 1.5$, where the superscript refers to each of the obstacles.

It is worth remarking that the obstacle avoidance constraints are only enforced at the sampling times involved in the discrete time predictions of the position. As a consequence this does not avoid stretches of the continuous-time trajectory crossing the obstacle. One alternative involves incorporating restrictions on the transition of the vehicle to each region of the space defined by obstacle inequalities to handle this issue, as proposed in [23]. This approach requires an additional number of binary variables, increasing the complexity of the MILP problem. In the present work, the length and width of the obstacles were expanded by an amount determined through the maximal admissible absolute value of the velocity in each axis. Thus, the avoidance constraints that were in fact imposed were constructed based on the following expanded obstacles: $0.5 \leq x_1^1, x_1^2 \leq 1.1$, $0.2 \leq x_3^1 \leq 0.9$, $1.1 \leq x_3^1 \leq 3.1$, $1.4 \leq x_1^3 \leq 2.1$ and, $0.7 \leq x_3^3 \leq 1.6$,

The weight γ of the fuel in the control cost function was set to 0.1. The maximal horizon was set to $\bar{N}_{OS} = 45$ for the one-step solution. Meanwhile, for the planner solution $\bar{N}_P = 10$ was adopted and the number of waypoints was set to $\bar{N}_{WP} = 4$. The weight of the fuel cost in the planning phase was $\gamma_p = 0.1$ and the weight of the binary variables was $\mu = 1$. The average computation time of 10 runs of each simulation was used to determine the computation times presented in Section V, in order to eliminate fluctuations due to external factors. All simulations were carried out in a personal computer equipped with a Pentium® Dual-Core E5400 processor with $2.7GHz$ clock. For solution of the

MILP, the CPLEX toolbox from IBM ILOG was used in Matlab environment, under an academic license.

V. Results and discussion

In this section, the trajectories of the vehicle and the generated control signals with the waypoint trajectory planner and replanning the whole trajectory at every sample time in the control phase are compared. The mean and largest computational times are also compared for both cases.

Figure 4 shows the trajectory with the waypoint trajectory planner. It can be seen that the vehicle reaches the target set from the initial position while avoiding collisions with the obstacles. All the four waypoints are crossed and the total time to reach the terminal set was 45 sample times.

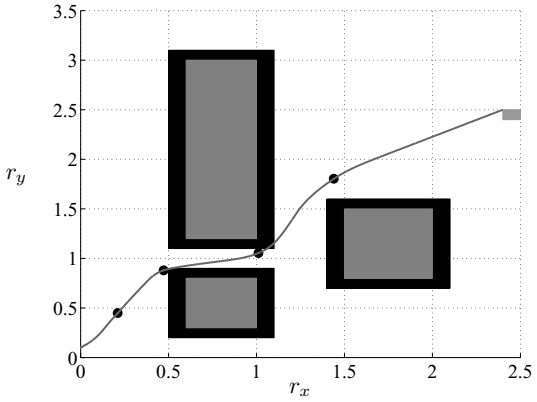

Fig. 4. Trajectory with waypoint trajectory planner.

For comparison, the trajectory with replanning at every sample time can be seen in Fig. 5. The vehicle also reaches the target set from the initial position while avoiding collisions with the obstacles. It can be noted that the resulting trajectory is more aggressive than the former one in terms of proximity to the obstacles. This is due to the constraint of the waypoints to the initial planned path in the first case, which causes the trajectory to remain closer to the initial planned path. This path, in turn, comes from a filtering of a path obtained via a Voronoi graph. Therefore, it steers the trajectory to remain closer to a path which maximizes the minimum distance between the obstacles and itself. The total amount of time to reach the terminal set was 36 sample times.

Regarding the control action, Figs. 6 and 7 show the commanded accelerations with the employment of the waypoint trajectory planning technique and without it, respectively. It can be seen that the control signals vary more often with the waypoint trajectory planning. In this case, the fuel cost was 31.57 and 31.07 with the replanning. It must be recalled from section IV that the weight of the fuel expense in the cost function γ is 0.1. Therefore, the overall cost involving the total time to reach the terminal set and the fuel expense was 39.11 in with the replanning and 48.16 with the waypoint trajectory planning. This shows that constraining

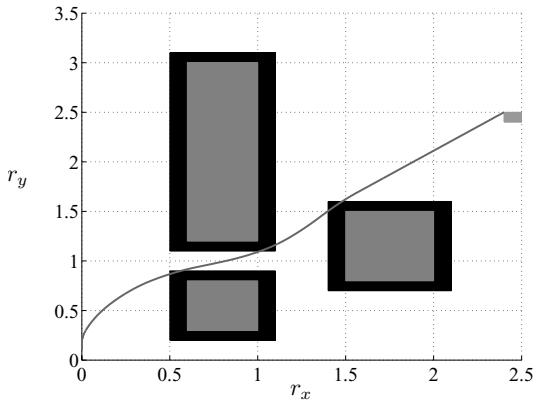

Fig. 5. Trajectory replanning at every sample time.

the waypoints to remain in the pre-planned path compromises the capacity of finding the optimal trajectory. However, this drawback is compensated by a substantial reduction in the required computation time.

The times taken to plan and execute the trajectory are presented in Table I. The times taken to execute the trajectory (T_{exec} and T_{exec}^{max}) are respectively, the mean and largest computation times per sample time. The times to plan the trajectory, for their part, are only measured once per test, since planning occurs before the maneuver starts. The time T_{total} is the average of the summation of the planning time and the total execution time for all the 45 sample instants in the case of the waypoint trajectory planner. As for the replanning case, it is the average of the total execution time during the 36 sample instants taken to execute the maneuver.

A significant reduction both in mean as well as largest computation times during the execution may be observed when the proposed waypoint trajectory planning is used. The planning phase lasted $35.60s$ in average and the largest planning time was $72.49s$, which are small in comparison to a largest execution time of $58.22s$ at one sample time without waypoint trajectory planning.

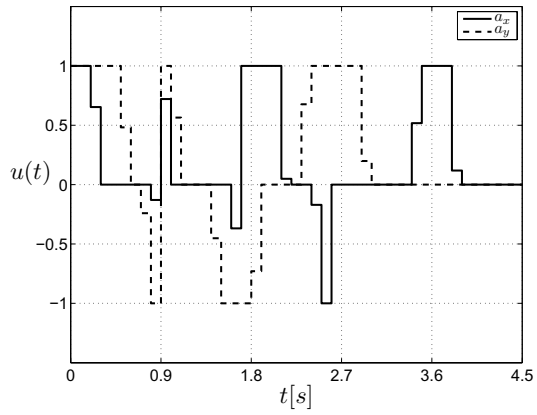

Fig. 6. Control signals with waypoint trajectory planner.

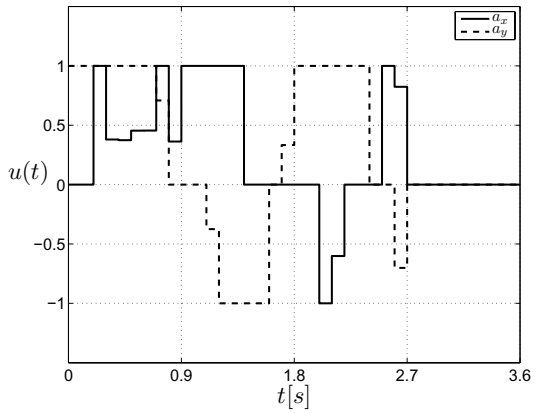

Fig. 7. Control signals obtained replanning the trajectory.

TABLE I

MEAN AND LARGEST TIMES WITH THE TWO SIMULATED EXAMPLES

Time (s)	Simulation	
	Waypoint	Replanning
T_{plan}	35.60	0
T_{exec}	0.04	11.81
T_{plan}^{max}	72.49	0
T_{exec}^{max}	0.13	58.22
T_{total}^{*}	37.41	425.06

$*$ T_{total} is the sum of the amount of time to carry out the maneuver plus the planning time.

VI. CONCLUSIONS

The integration of the waypoint trajectory planning with the tunnel-MILP delivered a feasible trajectory and reduced considerably the computational burden. The grade of optimality required and the computation time present a trade-off, as seen in the results that showed a greater cost in terms of the fuel expense and arrival time. Thus, if a more economic trajectory is necessary and there are more computational resources available, one can reduce the number of waypoints and increase the horizon between them.

It is important for real-world applications that the planned trajectory presents robustness to an unknown but limited disturbance. Thus, this remains a possible future work.

The planner can be extended for use with a periodic trajectory by setting a sequence of terminal sets along it. A state of the vehicle inside the previous terminal set is used as the initial state in Problem 3.1 and the next terminal set in the sequence is used as the terminal set of Problem 3.1. This procedure is repeated for all terminal sets in the sequence.

Future works could include the use of points in the intersection of adjoint polytopes of the tunnel as waypoints and have the cost calculated in each intersection to allow for the use of Dynamic Programming [1], [24], [25], [26].

ACKNOWLEDGMENTS

The authors acknowledge the support of FAPESP (2011/18632-8, 2011/17610-0) and CNPq (research fellowships). We thank Prof. Arthur Richards for bringing the Tunnel-MILP paper to our attention.

REFERENCES

[1] C. Goerzen, Z. Kong, and B. Mettler, "A survey of motion planning algorithms from the perspective of autonomous UAV guidance," *J. Intell. and Robotic Systems*, vol. 57, no. 1, pp. 65 – 100, 2010.

[2] S. LaValle and P. Konkimalla, "Algorithms for computing numerical optimal feedback motion strategies," *Int. J. Robotics Research*, vol. 20, no. 9, pp. 729 – 752, 2001.

[3] S. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," *Int. J. Robotics Research*, vol. 20, no. 5, pp. 378 – 399, 2001.

[4] J. Redding, J. N. Amin, and J. D. Boskovic, "A real-time obstacle detection and reactive path planning system for autonomous small-scale helicopters," in *AIAA Guid., Nav. Control Conf.*, no. 2007 - 6413, 2007.

[5] S. LaValle, "Motion planning: the essentials," *IEEE Robotics and Automation Society Mag.*, vol. 18, no. 1, pp. 79 – 89, 2011.

[6] E. Frazzoli, M. A. Dahleh, and E. Feron, "Real-time motion planning for agile autonomous vehicles," *J. Guid., Control, Dyn.*, vol. 25, no. 1, pp. 116 –129, 2002.

[7] C. Dever, B. Mettler, E. Feron, and J. Popovic, "Nonlinear trajectory generation for autonomous vehicles via parameterized maneuver classes," *J. Guid., Control, Dyn.*, vol. 29, no. 2, pp. 289 – 302, 2006.

[8] E. F. Camacho and C. Bordons, *Model Predictive Control*. London: Springer-Verlag, 1999.

[9] J. A. Rossiter, *Model-based Predictive Control: a practical approach*. Boca Raton: CRC Press, 2003.

[10] L. Singh and J. Fuller, "Trajectory generation for a UAV in urban terrain, using nonlinear MPC," in *Proc. American Control Conf.*, vol. 3, Arlington, 2001, pp. 2301–2308.

[11] N. Slegers, J. Kyle, and M. Costello, "Nonlinear model predictive control technique for unmanned air vehicles," *J. Guid., Control, Dyn.*, vol. 29, no. 5, pp. 1179 – 1188, 2006.

[12] I. D. Cowling, O. A. Yakimenko, J. F. Whidborne, and A. K. Cook, "A prototype of an autonomous controller for a quadrotor UAV," in *Proc. European Control Conf.*, 2007.

[13] R. B. Patel and P. J. Goulart, "Trajectory generation for aircraft avoidance maneuvers using online optimization," *J. Guid., Control, Dyn.*, vol. 34, no. 1, pp. 218 – 230, 2011.

[14] A. Richards and J. P. How, "Robust variable horizon model predictive control for vehicle maneuvering," *Int. J. Robust and Nonlinear Control*, vol. 16, no. 7, pp. 333 – 351, 2006.

[15] J. Bellingham, A. Richards, and J. P. How, "Receding horizon control of autonomous aerial vehicles," in *Proc. American Control Conf.*, Anchorage, 2002, pp. 3741–3746.

[16] M. P. Vitus, V. Pradeepz, G. M. Hoffmann, S. L. Waslander, and C. J. Tomlin, "Tunnel-MILP: Path planning with sequential convex polytopes," in *Proc. AIAA Guid., Nav., Control Conf.*, no. 7132, Honolulu, 2008, pp. 1–13.

[17] I. Prodan, F. Stoican, S. Olaru, and S.-I. Niculescu, "Enhancements on the hyperplanes arrangements in mixed-integer programming techniques," *J. Optim. Theory Appl.*, vol. 154, pp. 549–572, 2012.

[18] J. P. Vielma and G. L. Nemhauser, "Modeling disjunctive constraints with a logarithmic number of binary variables and constraints," *Math. Program., Ser. A*, vol. 128, pp. 49–72, 2011.

[19] R. J. M. Afonso, R. K. H. Galvão, and K. H. Kienitz, "Predictive control with trajectory planning in the presence of obstacles," in *Proc. UKACC Int. Conf. on Control, Cardiff, UK*, no. 60, 2012, pp. 508–514.

[20] S. Russel and P. Norvig, *Artificial Intelligence: a modern approach*, 3rd ed. Prentice Hall, 2009.

[21] D. S. Yershov and S. LaValle, "Simplicial dijkstra and A* algorithms: from graphs to continuous spaces," *Advanced Robotics*, vol. 26, no. 17, pp. 2065–2085, 2012.

[22] D. Lee and B. Schachter, "Two algorithms for constructing a delaunay triangulation," *Int. J. Comp. Inf. Sci.*, vol. 9, no. 3, pp. 219–242, 1980.

[23] M. H. Maia and R. K. H. Galvão, "On the use of mixed-integer linear programming for predictive control with avoidance constraints," *Int. J. Robust and Nonlinear Control*, vol. 19, pp. 822 – 828, 2009.

[24] A. Gasparetto and V. Zanotto, "A new method for smooth trajectory planning of robot manipulators," *Mechanism and Machine Theory*, vol. 42, pp. 455–471, 2007.

[25] J. T. Betts, "Survey of numerical methods for trajectory optimization," *J. Guid., Control, Dyn.*, vol. 21, no. 2, pp. 193–207, 1998.

[26] A. G. Barto, S. J. Bradtke, and S. P. Singh, "Learning to act using real-time dynamic programming," *Artificial Intelligence*, vol. 72, pp. 81–138, 1995.