Conflict Resolution for Autonomous Vehicles: A Case Study in Hierarchical Control Design

Magnus Egerstedt, and Clyde F. Martin

magnus@ece.gatech.edu
School of Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, GA 30332, U.S.A.

martin@math.ttu.edu
Department of Mathematics
Texas Tech University
Lubbock, TX 79409, U.S.A.

Abstract

In this paper we consider the problem of conflict avoidance for autonomous systems using a hierarchical control strategy that we establish using the notion of predicate consistent abstractions. We apply our proposed hierarchical design methodology to the case when a number of autonomous vehicles are asked to approach a single terminal in a computationally feasible, yet safe and orderly manner.

1 Introduction

As the technological and scientific advancements are making unmanned, autonomous vehicles readily available, the problem of controlling multiple robots in a coordinated fashion is becoming an important scientific issue [4, 9, 10, 14, 22, 23]. As a consequence, the need for computationally feasible solutions to time-critical multi-agent control problems, such as coordinated conflict resolution, is accentuated. This has given rise to an area of research, where various types of simplifications of the original problems have been suggested in such a way that critical safety properties are provably consistent with the simplification. In other words, if the simplified model produces "safe" trajectories then these trajectories remain "safe" for the real system as well. Different proposed ways for addressing this issue range from local, decentralized, worst-case decision making [13, 23] to simplified vehicle models [4]. In this paper, we propose a *hierarchical control strategy* for solving the multi-agent collision avoidance problem. The idea is to view the individual agents from three different layers of abstractions in such a way that we can prove that the overall system exhibits certain desired properties.

It can be argued that the emergence of hybrid control as a control design paradigm is driven by the need to manage complexity [15]. By breaking up a given control problem into subproblems, e.g. by identifying modes of operation, the result is a collection of simpler problems, whose solutions may or may not be globally compatible. One way of making sure that the solutions are in fact compatible is to use formal verification techniques, but, as pointed out in [3], for realistic problems the computational costs may be prohibitive. Another possible route is to construct the subproblems in a provably correct manner, and the main contribution of this paper is a formal method for inducing layers into the control problems, drawing inspiration from the literature on

abstractions [19, 20], as well as hierarchical control design [14]. In order for this program to be successful we first need to establish a language for describing hierarchical control systems, which will constitute the first part of this paper (Section 2). We then apply our methodology to the case where multiple autonomous vehicles are asked to approach a terminal or docking station in a safe and orderly manner (Section 3). This example is to be thought of as one instance in which a hierarchical structure is useful, and it arises for instance in air-traffic control around airports, as well as in the control of cooperative, multi-agent robots.

2 Hierarchal Control

The need for hierarchical control strategies has been brought to the fore by the development of increasingly complex engineering systems, such as embedded systems, distributed multi-agent systems, or sensor network systems. Hierarchical, or layered, control strategies are introduced as a mean to overcome the complexity of the system, and the idea is to construct controllers at a high level of abstraction in such a way that desired properties propagate down through the hierarchy, ultimately producing real, low-level controllers with a guaranteed desired behavior.

In this paper we propose a general formal framework for capturing just what we mean by consistent hierarchical strategies, and we illustrate the usefulness of this construction by solving a collision-avoidance problem in some detail for multiple autonomous robots. The contribution of this paper is thus twofold: First we show how a general theory of hierarchical control can be constructed. This theory is a more general version of the recent work suggested in [19, 20], where abstractions are defined as property preserving mappings. For instance, mappings have been derived that preserve controllability or stabilizability of linear control systems. The second contribution of this paper is the application of our hierarchical control strategy to the multi-agent collision avoidance problem. This problem has been studied quite extensively [10, 23], but the proposed solutions are relying on time consuming and computationally expensive numerical techniques that limit their usefulness in time critical applications. The solution that we propose in this paper is, in contrast, theoretically sound as well as computationally inexpensive, thanks to the layered control design.

2.1 Predicate Consistent Abstractions

Let X_i be a set admitting a binary partition under the predicate P_i, i.e if we set

$$P_{X_i} = \{x \in X_i \mid P_i(x) \text{ is true}\}$$
$$\tilde{P}_{X_i} = \{x \in X_i \mid P_i(x) \text{ is false}\},$$

then $\tilde{P}_{X_i} = \{x \in X_i \mid x \notin P_{X_i}\}$. From a hierarchical control point of view, the idea is to find objects that exhibit the desired property P_i (e.g. stability, optimality, etc.), i.e. to find $x_i \in P_{X_i}$ by solving the simpler problem of finding $x_{i+1} \in P_{X_{i+1}}$. This new predicate P_{i+1} and set X_{i+1} have to be chosen in such a way that it is possible to propagate the solution down through the hierarchy while keeping the relevant predicates true at the lower levels in the hierarchy. The reason why we index the predicates, the P_i's, is that a desired predicate (such as stability) at one level in the hierarchy might correspond to another predicate (such as eigenvalues with negative real parts) at another level.

If we now combine the set X_{i+1} and the predicate $P_{i+1} : X_{i+1} \to \{\text{true}, \text{false}\}$ with the mapping $\mathcal{A}_{i+1} : X_{i+1} \to X_i$, we can define the following key concept:

Definition 2.1 (Predicate Consistent Abstraction) *Given the sets X_i, X_{i+1} and the corresponding predicates P_i, P_{i+1}. We say that the triple $(X_{i+1}, P_{i+1}, \mathcal{A}_{i+1})$ is a predicate consistent abstraction of (X_i, P_i) if $\mathcal{A}_{i+1}(x_{i+1}) \in P_{X_i}$ for all $x_{i+1} \in P_{X_{i+1}}$.*

The reason for introducing the predicate consistent abstractions, and thus imposing a hierarchical structure on the problem, is that this allows us to cast the problem on a high level of abstraction, which is desirable for two reasons:

1. It may be easier to find an object $x_{i+1} \in P_{X_{i+1}}$ and to construct $\mathcal{A}_{i+1}$ than to find an object $x_i \in P_{X_i}$ directly.

2. Even if $\mathcal{A}_{i+1}$ might be computationally very expensive to construct, we only have to do this once. In real-time applications we can thus construct $x_{i+1} \in P_{X_{i+1}}$ (hopefully in real-time), and immediately compute $\mathcal{A}_{i+1}(x_{i+1}) \in P_{X_i}$ using the precomputed and stored abstraction mapping $\mathcal{A}_{i+1}$.

As a somewhat trivial, yet illustrative example of applying this concept one can consider the classic problem of finding a stabilizing, linear state feedback control law to the system

$$\dot{x} = Ax + Bu,$$

where $x \in \mathbb{R}^n, u \in \mathbb{R}^m$, and A, B is a controllable pair of compatible dimension. In the previously established terminology we can let X_0 be the set of all $m \times n$-matrices. Thus P_{X_0} takes on the form $\{K \in X_0 \mid \mathrm{Re}(\lambda_i(A - BK)) < 0,\ i = 1, \ldots, n\}$, where $\lambda_i(\cdot)$ denotes the i:th eigenvalue.

Instead of trying to find $K \in P_{X_0}$ directly, we know that by finding the symmetric, positive definite matrix solution $P \succ 0$ to the Riccati equation

$$A^T P + PA - 2PBB^T P + Q = 0,$$

given $Q \succeq 0$, we obtain the exponentially stabilizing feedback control law

$$u = -B^T P x.$$

(See any textbook on LQ-design, for example [5].) We can thus let X_1 be the set of all positive definite $n \times n$-matrices, and let P_{X_1} denote the subset of X_1 that solves the previously mentioned Riccati equation. By letting $\mathcal{A}_1(P) = B^T P \in X_0$ we, by construction, have that $(X_1, P_1, \mathcal{A}_1)$ is a predicate consistent abstraction of (X_0, P_0), since $\mathcal{A}_1(P) \in P_{X_0}$ for all $P \in P_{X_1}$.

Assuming that the original control problem consists of finding $x_0 \in X_0$ such that $P_0(x_0)$ is true, and that we have designed a collection of predicate consistent abstractions $(X_i, P_i, \mathcal{A}_i),\ i = 1, \ldots, N$, we can of course reiterate this argument. If we can find an object $x_N \in X_N$ such that $P_N(x_N)$ is true, then the following algorithm is guaranteed to terminate, per construction, with the desired result $x_0 \in P_{X_0}$.

```
Given x_N ∈ P_{X_N}
Set i = N
While i > 0
  Construct x_{i-1} = A_i(x_i)
  i := i - 1
End
```

2.2 Collision-Avoidance

The problem that we consider in the remainder of this paper is how to generate paths that lead multiple, planar, autonomous vehicles to a desired goal state, such as an airport terminal or a robot docking station, in a safe and orderly manner. However, we do not want to solve the problem of both controlling the nonlinear robot dynamics at the same time as we plan collision free routes, which is why a hierarchical approach is called for.

If we assume that each of the m individual robots' states evolve on the smooth manifold $\mathcal{M}$, and that the dynamics is defined by the smooth control system

$$\dot{x}_i(t) = f(x_i(t), u_i(t)),\ x_i(0) = x_{i0},\ i = 1, \ldots, m,$$

where $x_i(t) \in \mathcal{M}$ and $u_i(t)$ belongs to the set of admissible inputs U. Now, the problem that we investigates concerns driving these robots toward a terminal or docking station in the plane, and for this we associate an output equation

$$y_i(t) = h(x_i(t)) \in \mathbb{R}^2$$

to each robot that projects the robot states onto $\mathbb{R}^2$.

Now, the specifications when landing airplanes are typically given by discretizing the plane, resulting in a planar, Cartesian grid, with respect to which the waypoints are defined. We can, of course, always scale this grid in order to have it coincide with $\mathbb{Z}^2$, which will be done throughout this paper. This grid structure will furthermore be referred to as a cell-partitioning of $\mathbb{R}^2$, where each cell is defined as $\{y \in \mathbb{R}^2 \mid |y(1) - z(1)| \leq 1/2 \text{ and } |y(2) - z(2)| \leq 1/2\}$ for a given $z \in \mathbb{Z}^2$.

In order to formulate the specifications in terms of the robot dynamics, we introduce the mapping $\mathcal{Z} : \mathbb{R}^2 \to \mathbb{Z}^2$, that maps points in $\mathbb{R}^2$ to the midpoint of the particular cell that the point resides within, i.e.

$$\mathcal{Z}(y) = \text{argmin}_{z \in \mathbb{Z}^2} \|y - z\|_2,$$

where $\|\cdot\|_2$ denotes the standard Euclidean norm in $\mathbb{R}^2$. In order to avoid ambiguities we can furthermore always define $\mathcal{Z}$ uniquely on the boundary of each cell.

On $\mathbb{Z}^2$ we can now define the Manhattan metric $\|z\|_M$ as the minimum number of cells that need to be traversed in order to reach the cell containing the origin, and if we let

$$p_i = \|\mathcal{Z}(h(x_i(0)))\|_M \in \mathbb{Z}^2$$

we are ready to formulate the particular multi-agent specifications under consideration:

Definition 2.2 (Control Layer Specifications)

1. *Terminal Approach:* $\|\mathcal{Z}(h(x_i(k)))\|_M + 1 = \|\mathcal{Z}(h(x_i(k-1)))\|_M,\ k = 1, \ldots, p_i,\ i = 1, \ldots, m$; *and*
2. *Collision Avoidance: Given* i, $\|h(x_i(t)) - h(x_j(t))\|_2 > \rho,\ \forall j \neq i$, *as long as* t *satisfies* $\|\mathcal{Z}(h(x_i(t)))\|_M \neq 0$, *where* ρ *is the desired safety margin.*

It should be noted that by specifying the requirements in this way, we have implicitly, by virtue of Specification 1, required an additional property for our solution to exhibit:

3. *Docking:* $\|\mathcal{Z}(h(x_i(p_i)))\|_M = 0,\ i = 1, \ldots, m.$

2.2.1 Control Layer

At the lowest layer in the hierarchy, a control law has to be defined for each of the individual robots such that the specifications are met.

Let

$$p = \max_{i \in \{1, \ldots, m\}} p_i,$$

and define U^T as

$$U^T = \{u : [0, T] \to U\},$$

for a given $T \geq 0$. Under the input trajectory $u_i \in U^T$, the corresponding vector field $f(x_i, u_i)$ is well defined. If we let $U^\star = U^p \times \cdots \times U^p = (U^p)^m$, the multi-agent control problem can thus be cast in terms of finding $\mathbf{u} = (u_1, \ldots, u_m) \in U^\star$ such that the specifications are met. In the abstraction formalism we can thus define X_0 and P_0 as:

Definition 2.3 (Level 0) *Let* $X_0 = U^\star$ *be the set that defines the lowest level in the hierarchy, and let the multi-agent collision avoidance predicate* P_0 *be true for all* $\mathbf{u} = (u_1, \ldots, u_m) \in X_0$ *such that they, when applied to the individual robots, make the robots satisfy the specifications in Definition 2.2.*

2.2.2 Path Planning Layer

Instead of trying to find $\mathbf{u} \in P_{X_0}$ we can take advantage of the fact that there already exist a number of efficient tracking algorithms with provable performance, and we refer to the literature [1, 6, 7, 8, 11, 18, 22] for a treatment of this topic. For the purpose of this paper we assume that given a twice differentiable path in $\mathbb{R}^2$, i.e. $h_{id} \in C_2^2[0,p]$, with $h_{id}(0) = h(x_i(0))$, we can always design an output feedback control law $\phi_i : Y \to U$ such that

$$\|h(x_i(t)) - h_{id}(t)\|_2 < \epsilon, \ t \in [0,p].$$

Here $\epsilon > 0$ is a number smaller than $1/2$, i.e. smaller than the grid dimensions, and x_i is generated by the action of the control law ϕ_i.

The main idea now is to assume that we can produce a collection of curves in $C_2^2[0,p]$ that satisfy certain conditions, and then use the tracking laws (the ϕ_i's) in order to guarantee that the specifications are met. The specifications that we propose are simply:

Definition 2.4 (Path Planning Layer Specifications)

1. *Terminal Approach:* $\|\mathcal{Z}(h_{id}(k))\|_M + 1 = \|\mathcal{Z}(h_{id}(k-1))\|_M, \ k = 1, \ldots, p_i, \ i = 1, \ldots, m;$
2. *Interior Passage:* $\|\mathcal{Z}(h_{id}(k)) - h_{id}(k)\|_2 < 1/2 - \epsilon, \ i = 1, \ldots, m;$ *and*
3. *Collision Avoidance: Given* i, $\|h_{id}(t) - h_{jd}(t)\|_2 > \rho + 2\epsilon, \ \forall j \neq i,$ *as long as* t *satisfies* $\|\mathcal{Z}(h_{id}(t))\|_M \neq 0.$

If we let $(C_2^2[0,p])^m$ denote the product of m copies of $C_2^2[0,p]$, and denote an element in this space by $\mathbf{h}$ we can formulate the following abstraction parameters:

Definition 2.5 (Level 1) *Let* $X_1 = (C_2^2[0,p])^m$ *and let* $P_1(\mathbf{h})$ *hold true for all* $\mathbf{h} \in (C_2^2[0,p])^m$ *such that the specifications in Definition 2.4 are met. Furthermore, let* $\mathcal{A}_1(\mathbf{h}) \in U^\star$ *be given by the control inputs generated by the feedback laws* $\boldsymbol{\phi} = (\phi_1, \ldots, \phi_m)$.

Theorem 2.1 $(X_1, P_1, \mathcal{A}_1)$ *is a predicate consistent abstraction of* (X_0, P_0), *i.e.* $\mathcal{A}_1(\mathbf{h}) \in P_{X_0}$ *for all* $\mathbf{h} \in P_{X_1}$.

Proof: Let $\mathbf{h} \in P_{X_1}$ and let $\boldsymbol{\phi} = (\phi_1, \ldots, \phi_m) = \mathcal{A}_1(\mathbf{h})$. Then, if we let each individual system evolve according to the dynamics generated by the feedback law $\phi_i, \ i = 1, \ldots, m$, we have that, by the triangle inequality:

$$\begin{aligned} \|h(x_i(k)) - \mathcal{Z}(h_{id}(k))\|_2 &\leq \|h(x_i(k)) - h_{id}(k)\|_2 + \|h_{id}(k) - \mathcal{Z}(h_{id}(k))\|_2 \\ &< \epsilon + 1/2 - \epsilon = 1/2, \end{aligned}$$

where $k = 1, \ldots, p_i$. In other words, $\mathcal{Z}(h(x_i(k))) = \mathcal{Z}(h_{id}(k))$, and hence the first specification in Definition 2.2 is satisfied.

The triangle inequality furthermore gives that

$$\begin{aligned} \|h(x_i(t)) - h(x_j(t))\|_2 &\geq \|h_{id}(t) - h_{jd}(t)\|_2 - \|h(x_i(t)) - h_{id}(t)\|_2 - \|h(x_j(t)) - h_{jd}(t)\|_2 \\ &> \rho + 2\epsilon - \epsilon - \epsilon = \rho, \end{aligned}$$

as long as t satisfies $\|\mathcal{Z}(h_{id}(t))\|_M \neq 0$. Hence the second specification in Definition 2.2 is satisfied, and the theorem follows. ∎

2.2.3 Discrete Time Layer

So far so good, but ultimately we would like to be able to specify only discrete points, i.e. points on the grid, and then generate the appropriate paths automatically. In other words, if we let $\mathbb{Z}^{2p}$ be strings of length p over $\mathbb{Z}^2$, we can set $X_2 = \mathbb{Z}^\star$, where $\mathbb{Z}^\star = \mathbb{Z}^{2p} \times \cdots \times \mathbb{Z}^{2p}$. What remains to be done in order to solve the original problem is thus to produce the correct specifications, i.e. to define P_2 as well as find $\mathbf{q} \in P_{X_2}$. Furthermore, an appropriate abstraction map $\mathcal{A}_2$ must be defined. This will be the focus of the following section, where we will show how to explicitly solve the multi-agent collision avoidance problem in a hierarchical, yet provably correct manner.

3 Hierarchical Multi-Agent Control

3.1 Discrete Time Planning

The purpose of this section is to find $\mathbf{q} \in P_{X_2}$ and to characterize P_2, i.e. to find safe paths through the Cartesian grid. We do this by characterizing all possible unsafe situations that can occur.

At the highest level of abstraction in our hierarchical control strategy, we let each individual vehicle be in a particular square in the Cartesian grid for one unit of time. For safety reasons, two vehicles in a square is forbidden since this case could potentially result in a "near miss", or a collision. We let the terminal, or goal point, have coordinates (0,0), and we give the grid a metric based on the norm $||(a,b)|| = |a| + |b|$, which in the air-traffic control literature is known as is the Manhattan metric [21].

When a vehicle enters the region around the terminal, it should trace a trajectory of minimal distance to the terminal, while avoiding contact with all other vehicles. Thus, if the vehicle is at (7,-5), it must reach the terminal in 12 time units. At the next instant of time, it must be at either (6,-5) or at (7,-4). We can denote this by the following controlled equation

$$(x_k, y_k) = (x_{k-1}, y_{k-1}) + (\delta_1, \delta_2),$$

where $|\delta_i| = 1$ or 0, and the following rules apply $\delta_1\delta_2 = 0$, $|x_k| \leq |x_{k-1}|$, $|y_k| \leq |y_{k-1}|$, and $\delta_1^2 + \delta_2^2 = 1$. These rules apply anytime the position of the vehicle is described in terms of its location with respect to the terminal.

It is straightforward to see that the conditions concerning blockages surrounding the terminal can be completely characterized by the following three examples: The first case is displayed in Figure 1(a). It is the simplest blockage case, where it is impossible to move without x_2 being in conflict with either x_1 or x_3. The real problem is that all three vehicles are exactly three time units from the terminal, and there are only two possible routes for them to eventually take.

The fact that there are exactly four directions from which the robots can enter the terminal means that at most four vehicles may enter at any given instant of time. Thus, if five or more vehicles are anywhere in the grid, equidistant from the terminal, then there is an unavoidable blockage. This is illustrated in Figure 1(b). Similarly, if there are four vehicles equidistant from the terminal in two adjacent quadrants then this will force four vehicles to reach the terminal simultaneously from three different directions, as illustrated in Figure 1(c).

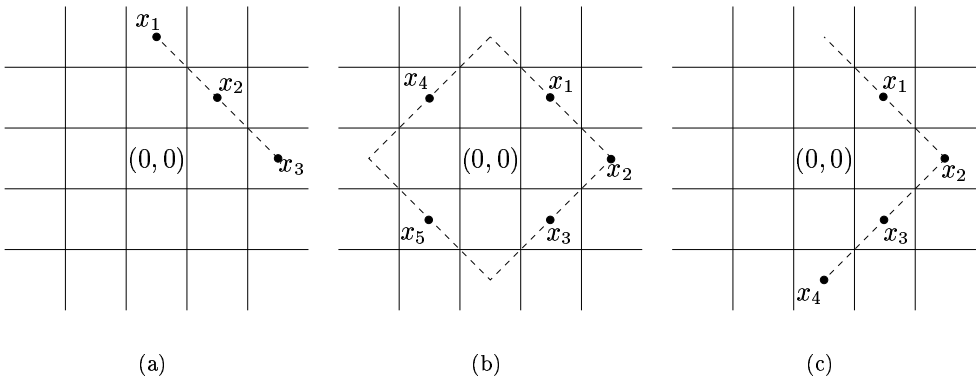


Figure 1: The three possible blockage cases.

Can any other conflicts occur? The answer is no, and thus only the conflicts illustrated in Figures 1(a)–(c) lead to inevitable conflicts. This can be shown by working backwards. Suppose

that vehicles x and y are in conflict at time $k+1$. Then at time k they were in a position where they had no choice on their next move. This requires that either one or the other was heading straight towards the terminal, or that there were other vehicles on the same level line. Thus we are either in the situation of Figure 1(a) as a consequence of the situation in Figure 1(b), or we are in the situation of Figure 1(c), with several vehicles on the same level line in a single quadrant.

A safe route can thus be planned (and hence $\mathbf{q} \in P_{X_2}$ can be generated in the hierarchical control strategy) in a straightforward manner using the standard graph geodesics if and only if the following assumption is satisfied:

Assumption 3.1 (Conflict-Free Paths)

1. *No more than two vehicles in a single quadrant are equidistant from the origin.*
2. *No more than three vehicles in two adjacent quadrants are equidistant from the origin.*
3. *No more then four vehicles are equidistant from the origin.*

We can thus define the specifications at this level in the hierarchy as:

Definition 3.1 (Discrete Time Layer Specifications)

1. *Terminal Approach: For each individual robot (robot i) it holds that* $(x_k, y_k) = (x_{k-1}, y_{k-1}) + (\delta_1, \delta_2),\ k = 1, \ldots, p_i,$ *where* $|\delta_i| = 1$ *or* 0 *and* $\delta_1^2 + \delta_2^2 = 1$.
2. *Collision Avoidance: The robots should satisfy Assumption 3.1 and the paths through* $\mathbb{Z}^2$ *should be given by standard graph geodesics.*

3.2 Trajectory Planning Using Splines

Now that we can find $\mathbf{q} \in P_{X_2}$ the next topic under investigation thus becomes the search for $\mathcal{A}_2$. In other words, how should appropriate paths be planned at the path-planning level such that the predicate P_1 is satisfied? Our solution to this problem is to use control for driving a linear system through a series of waypoints in such a way that the safety properties from the discrete-time level are preserved.

Similar types of planning problems have been studied in [2, 17, 24, 25], and in particular, techniques have been developed for producing so called *monotone splines* [12]. These types of curves are of particular importance to the problem at hand since the vehicles should approach the goal monotonically, and in the paragraphs to follow, we outline the development of these trajectories (and hence of the construction of $\mathcal{A}_2$).

Assume that we are given a set of safe, shortest-distance points through the grid, $\xi_i,\ i = 1, \ldots, N$, that constitute one instance of a conflict-free path in a multi-vehicle scenario. Furthermore, these points are located in the middle of squares in the grid, $\xi_i \in \mathcal{G}_i,\ i = 1, \ldots, N$, with side length 1. Thus $\mathcal{G}_i$ has mid-point $(\xi_{xi}, \xi_{yi}),\ i = 1, \ldots, N$. What we want to achieve is to generate a feasible route that *only goes through these specified squares in the grid.* We want to do this at the same time as we want to prevent the trajectories from oscillating, due to the fact that we ultimately want the vehicles to be able to follow the paths.

We, without loss of generality, assume that we are working with grid squares in the third quadrant, and the task is to drive the path through the squares to the terminal, located at the origin. This means that (ξ_{xi+1}, ξ_{yi+1}) is either $(\xi_{xi} + 1, \xi_{yi})$ or $(\xi_{xi}, \xi_{yi} + 1)$.

If we decouple the path planning problem into two subproblems, one along each axis, a preliminary version of the interpolation problem can be formulated as: Drive $(x(t), y(t)) \in \mathbb{R}^2$ close to (ξ_{xi}, ξ_{yi}) at the corresponding interpolation times, t_i, while staying in the specified grid squares for all times. At the same time we want the path to be smooth, which gives that we,

for instance, could minimize the L_2-norm of the second derivatives in the x and y-directions. In other words, given the system dynamics

$$\ddot{x}(t) = u_x(t),\ \ddot{y}(t) = u_y(t),\ u_x, u_y \in L_2[0,T],$$

what we want to do is minimize

$$\int_0^T \Big(u_x(t)^2 + u_y(t)^2\Big)dt + \sum_{i=1}^{N}\Big(\tau_{xi}(x(t_i) - \xi_{xi})^2 + \tau_{yi}(y(t_i) - \xi_{yi})^2\Big),$$

where T is the total time of the maneuver, and $\tau_{xi} > 0$ and $\tau_{yi} > 0$ determine how much importance should be given to the waypoint fitting around (ξ_{xi}, ξ_{yi}). We furthermore impose the following constraint on our optimal control problem:

$$\forall t \in [0,T]\ \exists i \in \{1,\ldots,N\} \mid (x(t), y(t)) \in \mathcal{G}_i.$$

This last constraint is an infinite dimensional constraint since it is a property that has to hold for all times, making the optimization problem very hard to solve [16]. Instead we would like to reformulate this as a problem where the grid constraints are finite. This can be achieved if we require that the curve is monotonously increasing in both the x and the y-direction (since we are in the third quadrant) while demanding that $(x(t_i), y(t_i)) \in \mathcal{G}_i, i = 1,\ldots,N$, which allows us to trade one infinite dimensional constraint for another, i.e. that $\dot{x}(t), \dot{y}(t) \geq 0\ \forall t \in [0,T]$.

We can thus reformulate the problem as

$$\min_{u_x,u_y} \int_0^T (u_x^2(t) + u_y^2(t))dt + \sum_{i=1}^{N}\Big(\tau_{xi}(x(t_i) - \xi_{xi})^2 + \tau_{yi}(y(t_i) - \xi_{yi})^2\Big),$$

subject to

$$\begin{array}{l} \ddot{x}(t) = u_x, \quad \ddot{y}(t) = u_y,\ u_x, u_y \in L_2[0,T] \\ \dot{x}(t) \geq 0, \quad \dot{y}(t) \geq 0, \quad \forall t \in [0,T] \\ (x(t_i), y(t_i)) \in \mathcal{G}_i, i = 1,\ldots,N, \end{array}$$

where we have assumed that $x(0), y(0) \in \mathcal{G}_1, \dot{x}(0), \dot{y}(0) \geq 0$.

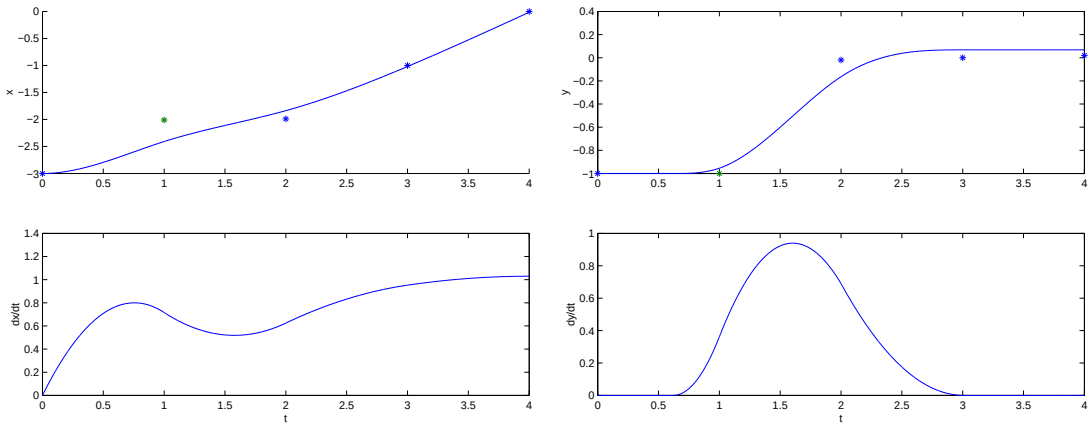
In [12] it was found that the set of controls in $L_2[0,T] \times L_2[0,T]$ that satisfy the constraints is a closed, convex, and non-empty set, which is a strong enough result to guarantee the existence of a unique optimal solution. In that work it was also shown that the optimal control in $L_2[0,T] \times L_2[0,T]$ is piecewise linear. Furthermore, u_x only changes from different linear cases at the waypoints, or at times when $\dot{x}(t) = 0$, and similarly for u_y.

Based on these two facts it is possible to find the optimal, piecewise linear control inputs in a computationally feasible way by solving a dynamic programming problem along each axis. An example of applying this method to the problem of planning planar, feasible paths are shown in Figures 2-3.

3.3 Final Remarks

It should be pointed out that by making the τ_i's large in the cost function in the previous section, we can make the curve pass arbitrarily close to the waypoints as long as these points are feasible with respect to the monotonicity constraint. What this means it that if we think of $\mathcal{A}_2$ as a mapping from$\mathbb{Z}^\star$ to $(C_2^2[0,p])^m$ we can break this mapping down into m components, where each component produces a monotone smoothing spline.

Definition 3.2 (Level 2) *Let $X_2 = \mathbb{Z}^\star$ and let $P_2(\mathbf{q})$ hold true for all $\mathbf{q} \in \mathbb{Z}^\star$ such that the specifications in Definition 3.1 are met. Furthermore, let $\mathcal{A}_2(\mathbf{q})$ be given by a collection of m monotone smoothing splines through the waypoints specified by $\mathbf{q}$.*



(a) (b)

Figure 2: In the left figure, x (top) and $\dot{x}$ (bottom) are displayed, while the right figure shows the solution in the y-direction. The stars correspond to the waypoints, and it should be noted that the derivatives are non-negative for all times.

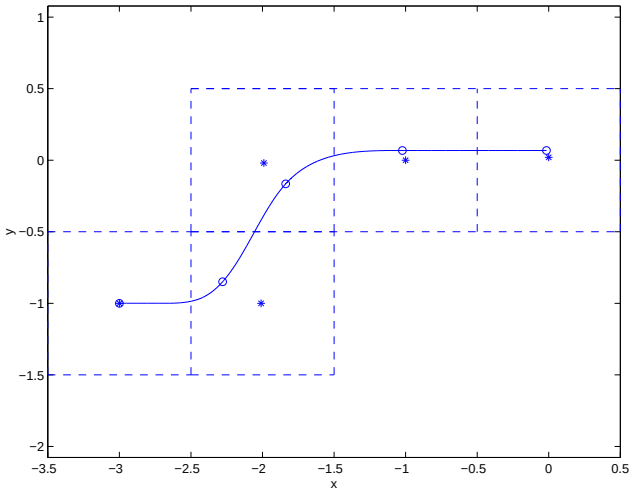


Figure 3: A path-planning example, corresponding to the solution in Figure 2, is displayed. The circles and stars correspond to the actual positions at the interpolation times and the waypoints respectively.

Theorem 3.1 $(X_2, P_2, \mathcal{A}_2)$ *is a predicate consistent abstraction of* (X_1, P_1)*, i.e.* $\mathcal{A}_2(\mathbf{q}) \in P_1$ *for all* $\mathbf{q} \in P_{X_2}$.

Proof Outline: By making the τ_i's large enough, when producing the monotone splines, we have directly guaranteed that both the Terminal Approach and Interior Passage specifications are met at Level 1. Furthermore, Collision Avoidance follows, as long as ρ (the safety distance) is sufficiently smaller than the dimensions of the grid, from the choice of waypoints in combination with the fact that the monotone splines never leave their prespecified cells.

Corollary 3.1 $\mathcal{A}_1(\mathcal{A}_2(\mathbf{q})) \in P_{X_0}$ *for all* $\mathbf{q} \in P_{X_2}$.

4 Conclusions

In this paper we establish an effective language for describing hierarchical control strategies, based on predicate consistent abstractions. Given a set X_i, together with a predicate P_i that we want our solution to satisfy, we show that we can generate solutions $x_i \in P_{X_i}$ by solving the problem at a higher level of abstraction, i.e. by finding $x_{i+1} \in P_{X_{i+1}}$ and then computing x_i using the abstraction mapping $x_i = \mathcal{A}_{i+1}(x_{i+1})$.

We furthermore apply this methodology to the problem of multi-agent collision avoidance as an illustration of the practical soundness of these concepts. Our hierarchical solution to the collision-avoidance problem is produced by generating safe routes through a Cartesian grid, combined with a computationally sound method for mapping these routes onto the set of monotone smoothing splines. As shown in Figure 4, the price we have to pay for using a hierarchical control strategy is a potentially significant reduction of the set of possible solutions. Instead of letting the collision-free multi-agent paths satisfy P_0, they in fact have to lay in the much smaller set $\mathcal{A}_1(\mathcal{A}_2(P_{X_2}))$. However, by only requiring that the specifications are met, i.e. by not imposing any optimality concerns, the resulting solutions are provably correct.

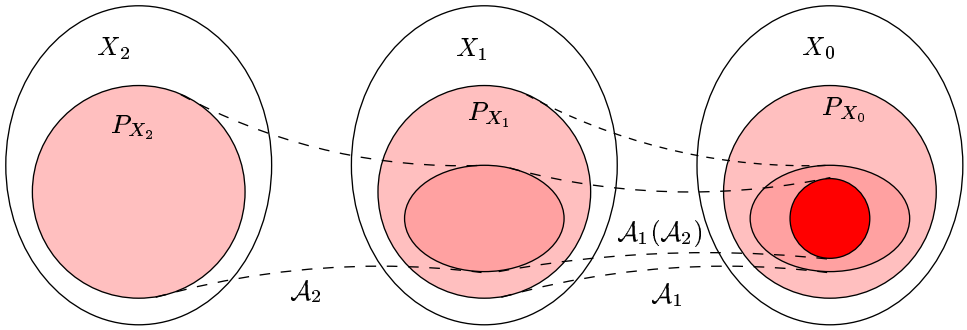


Figure 4:

References

[1] J. Ackermann. *Robust Control.* Springer-Verlag, London pp. 371-375 1993.

[2] N.N. Agwu and C.F. Martin. Optimal Control of Dynamic Systems: Application to Spline Approximations. *Appl. Math. Comput.* 97 (1998), no. 2-3, 99–138.

[3] R. Alur, J. Esposito, M. Kim, V. Kumar, and I. Lee. Formal Modeling and Analysis of Hybrid Systems: A Case Study in Multirobot Coordination. *Proceedings of the World Congress on Formal Methods*, LNCS 1708, pp. 212–232, Springer, 1999.

[4] T. Balch and R.C. Arkin. Behavior-Based Formation Control for Multi-Robot Teams. *IEEE Transactions on Robotics and Automation*, Vol. 14, No. 6, Dec. 1998.

[5] R.W. Brockett. *Finite Dimensional Linear Systems.* John Wiley and Sons, New York, 1970.

[6] G. Campion, G. Bastin, and B. D'Andréa-Novel. Structural Properties and Classification of Kinematic and Dynamic Models of Wheeled Mobile Robots. *IEEE Transactions on Robotics and Automation*, Vol. 12, No. 1, Feb. 1996.

[7] C. Canudas de Wit, Bruno Siciliano and G. Bastin. Theory of Robot Control. *Springer Verlag, 1996.*

[8] C. Canudas de Wit. Trends in Mobile Robot and Vehicle Control. *Control Problems in Robotics*, Lecture Notes in Control and Information Sciences 230, pp. 151–176, eds. B. Siciliano and K.P. Valavanis, Springer-Verlag, London, 1998.

[9] J. Desai, J. Ostrowski, and V. Kumar. Control of Formations for Multiple Robots. Proceedings of the *IEEE International Conference on Robotics and Automation, Leuven*, Belgium, May 1998.

[10] M. Egerstedt and X. Hu. Formation Constrained Multi-Agent Control. *IEEE Transactions on Robotics and Automation*, Vol. 17, No. 6, Dec. 2001.

[11] M. Egerstedt, X. Hu, and A. Stotsky. Control of a Car-Like Robot Using a Virtual Vehicle Approach. Proceedings of the *37th IEEE Conference on Decision and Control*, pp. 1502–1507, Tampa, Florida, USA, Dec. 1998.

[12] M. Egerstedt and C.F. Martin. Monotone Smoothing Splines. *Mathematical Theory of Networks and Systems*, Perpignan, France, June 2000.

[13] D. Fudenberg and J. Tirole. *Game Theory.* MIT Press, Cambridge, MA, 1991.

[14] T. J. Koo, B. Sinopoli, A. Sangiovanni-Vincentelli, and S. Sastry. A Formal Approach to Reactive System Design: A UAV Flight Management System Design Example. In *Proceedings of IEEE International Symposium on Computer-Aided Control System Design*, Kohala Coast, Hawaii, August 1999 .

[15] T. J. Koo, G. J. Pappas, and S. Sastry. Mode Switching Synthesis for Reachability Specifications. *Hybrid Systems: Computation and Control*, M. D. Di Benedetto and A. Sangiovanni-Vincentelli (Eds.), Lecture Notes in Computer Science, Vol. 2034, pp. 333-346, Springer-Verlag, 2001.

[16] D.G. Luenberger. *Optimization by Vector Space Methods.* John Wiley and Sons, Inc., New York, 1969.

[17] C.F. Martin, P. Enqvist, J. Tomlinson, and Z. Zhang. Linear Control Theory, Splines and Interpolation. *Computation and Control, IV* (Bozeman, MT, 1994), 269–287, Progr. *Systems Control Theory*, 20, Birkhuser Boston, Boston, MA, 1995

[18] R. Murray and S. Sastry. Nonholonomic Motion Planning: Steering Using Sinusoids. *IEEE Transactions on Automatic Control*, Vol. 38, No. 5, pp. 700-716, 1993.

[19] G. Pappas and S. Simic. Consistent Hierarchies of Nonlinear Abstractions. In *Proceedings of 39th IEEE Conference on Decision and Control*, Sydney, Australia, Dec. 2000.

[20] G.J. Pappas, G. Lafferriere, and S. Sastry. Hierarchically Consistent Control Systems. *IEEE Transactions on Automatic Control*, Vol. 45, No. 6, pp. 1144–1160, June 2000.

[21] T.S. Perry. In Search of the Future of Air-Traffic Control. *IEEE Spectrum*, Vol. 34, No. 8, pp. 18–35, 1997.

[22] N. Sarkar, X. Yun, and V. Kumar. Dynamic Path Following: A New Control Algorithm for Mobile Robots. *Proceedings of the 32nd Conference on Decision and Control*, San Antonio, Texas, Dec. 1993.

[23] C. Tomlin, G.J. Pappas, and S. Sastry. Conflict Resolution for Air Traffic Management: A Study in Multi-Agent Hybrid Systems. *IEEE Transactions on Automatic Control*, Volume 43, Number 4, April 1998.

[24] Z. Zhang, C.F. Martin. Convergence and Gibbs' Phenomenon in Cubic Spline Interpolation of Discontinuous Functions. *J. Comput. Appl. Math.* 87 (1997), No. 2, 359–371.

[25] Z. Zhang, J. Tomlinson, and C.F. Martin. Splines and Linear Control Theory. *Acta Appl. Math.* 49 (1997), No. 1, 1–34.