Online Motion Planning for Multi-Robot Interaction Using Composable Reachable Sets

Aris Valtazanos and Subramanian Ramamoorthy

School of Informatics
University of Edinburgh
Edinburgh EH8 9AB, United Kingdom
a.valtazanos@sms.ed.ac.uk, s.ramamoorthy@ed.ac.uk

Abstract. This paper presents an algorithm for autonomous online path planning in uncertain, possibly adversarial, and partially observable environments. In contrast to many state-of-the-art motion planning approaches, our focus is on decision making in the presence of adversarial agents who may be acting strategically but whose exact behaviour is difficult to model precisely. Our algorithm first computes a collection of *reachable sets* with respect to a family of possible strategies available to the adversary. Online, the agent uses these sets as *composable behavioural templates*, in conjunction with a particle filter to maintain the current belief on the adversary's strategy. In partially observable environments, this yields significant performance improvements over state-of-the-art planning algorithms. We present empirical results to this effect using a robotic soccer simulator, highlighting the applicability of our implementation against adversaries with varying capabilities. We also demonstrate experiments on the *NAO* humanoid robots, in the context of different collision-avoidance scenarios.

Keywords: Online Motion Planning, Autonomous Decision Making, Composable Behavioural Templates

1 Introduction

As robots become increasingly more autonomous, they require online decision making skills for adversarial environments, in the presence of other agents with conflicting objectives. Although there are many theoretical frameworks for addressing such decision making problems, nearly all assume significant knowledge on the capabilities and actions of the adversaries. In actual practice, agents must synthesise their beliefs from limited observations of the world and devise plans that are likely to succeed, despite significant imprecision regarding actions and *strategic profiles* corresponding to the adversary. Even though an agent's own goals may be concretely formulated, the behaviours of its adversaries may not be easily characterised to the level required by formal algorithmic tools.

Our proposed approach draws on the notion of a *reachable set*, as used in the literature on hybrid systems in control theory, to encode unsafe sets of state configurations up to infinite time horizons. We extend the basic concept by using

a collection of infinite-horizon reachable sets corresponding to coarse profiles. These sets, computed offline, form *composable templates* that can be used online to generate safe trajectories with respect to an adversary's realised behaviour. In partially observable environments, we use a particle filter algorithm for estimating *beliefs* about other agents. These beliefs are used to select the template that most closely matches the perceived coarse capabilities of the adversary, which in turn informs selection of appropriate landmarks for motion planning.

The reachable set formulation has a strong game-theoretic flavour, as it builds on the notion of selecting optimal control inputs with respect to a class of disturbances available to an adversary. However, it also differs from traditional game theory in not assuming any explicit knowledge of payoffs associated with specific actions. When used in conjunction with probabilistic filtering, reachable set composition is less sensitive to errors in the estimation of the adversary's strategy. This leads to the generation of safer trajectories in reactive path planning.

2 Background and Related Work

2.1 Hybrid System Modeling

A key concern in our work is the modeling of an opponent's behaviour, which is dictated by choices over discrete behavioural modes and underlying continuous dynamics. A good framework for thinking about such problems is available within the control theory literature, where systems with joint discrete and continuous dynamics are known as hybrid systems [13].

A major application of hybrid system modeling has been the formal description of aircraft collision avoidance as a pursuit-evasion game between two adversaries [13]. Each aircraft assumes the role of an *evader* seeking to avoid collision with an adversary, who is modeled as a *pursuer* with the exact opposite goal. An evader has a notion of a *target* set of unsafe states, which must be avoided to prevent collision.

A key innovation of this approach is the introduction of *reachable* sets of states, which can be classified in one of two ways. A **forward reachable set** is the set of states that can be reached from some given initial configurations. A **backward reachable set** is the set of states that may give rise to trajectories terminating in a target set of unsafe states. In path planning, backward reachable sets provide an elegant way of determining *the entire set of trajectories* that are likely to lead to the satisfaction of a goal. These sets can be computed directly, without recourse to exhaustive simulation over all possible state transitions.

One popular approach for approximating reachable sets is their estimation as the zero sublevel set of the solution of the Hamilton-Jabobi-Isaacs PDE with respect to the system dynamics [13]. Other works use overapproximations of reachable sets to compute optimal trajectories [6]. A comprehensive toolbox with level set implementations for reachable sets was developed in [10].

2.2 Reactive Path Planning

Several mobile robots require planning routines that can adapt to dynamically changing obstacles and environmental context. *Reactive* path planning refers to

the class of algorithms that are used to solve this problem online, based on continuously updated information on a robot's environment.

In *velocity obstacle* algorithms [7], agents estimate the velocities of their surrounding obstacles, which are used to define regions that should be avoided. Reciprocal collision avoidance [15] is a special case of the above, which assumes that all agents follow the same planning procedure. Other state-of-the-art motion planning examples for a variety of multi-agent domains are surveyed in [4].

In comparison to the work in this paper, most of the above examples do not address sensing uncertainty, partial observability, physical capability modeling, and unknown adversarial strategy profiles. The path planning problem is most challenging precisely when all these constraints arise simultaneously, which highlights the need for robust and efficient solutions. This is the focus of our paper.

2.3 Template-based Planning and Control

Many practical problems in robotics are hard to solve using direct optimisation, due to their high dimensionality or complex dynamics. An approach that is increasingly gaining traction is to devise coarse abstractions that simplify the overall problem. [5] presents a framework of local feedback planning policies, represented as funnels, which can be sequenced through back-chaining to achieve a global goal. Extending this idea, [12] defines these local controllers as linear quadratic regulators. Our approach extends these notions by explicitly considering control in the presence of other, possibly strategic, agents.

A related theme is found in risk-sensitive planning and control [16]. Here, obstacles are associated with the risk they impose on a given trajectory. Potential functions have also been proposed as a means of expressing the desirability of different objects and landmark points for candidate trajectories [11].

3 Algorithm

In this paper, we focus on robotic soccer with humanoid robots, a domain capturing the features we have discussed so far: strategic adversaries with unknown behavioural models, partial observability, uncertainty in sensing and actuation, and dynamically varying goals. We first define the dynamics of our system and the corresponding reachable sets following the conventions of Tomlin et al [13].

3.1 System Dynamics

We consider a system involving two holonomic robots r_A and r_B (Figure 1). As the robots are autonomous and are not provided with external input, all coordinate frames are egocentric. Throughout this section, we consider the case of r_A planning trajectories with respect to the adversarial agent r_B.

The state x of r_B relative to r_A is defined in terms of the vector:

$$x = \left[x_1, x_2, x_3\right]^T \tag{1}$$

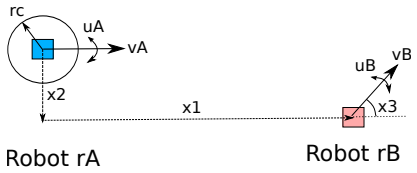


Fig. 1. Coordinate system centred on robot r_A

where x_1, x_2 are the planar coordinates and x_3 is the heading of r_B relative to r_A. Because of the partial observability assumption, we approximate the relative heading x_3 as the difference between the directions of their linear velocities. A particle filter is used to deal with the uncertainty in the estimation of relative positions (Section 3.4). Furthermore, r_c is defined as a fixed-length distance, within which collisions between the two robots occur.

The dynamics of the system can then be defined as

$$\dot{x} = \frac{d}{dt}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} -v_A + v_B \cos x_3 + u_A x_2 \\ v_B \sin x_3 - u_A x_1 \\ u_B - u_A \end{bmatrix} = f(x, u_A, u_B). \tag{2}$$

where v_A and v_B are the linear velocities of the two robots, and u_A and u_B are their angular velocities. In accordance with most works in the hybrid systems literature (e.g. [13]), linear velocities are treated as *fixed* parameters, whereas angular velocities are *control* inputs selected by the two robots.

3.2 Reachable Set Calculation

In our problem domain, backward reachable sets represent the states the agents must avoid over some time horizon, in order to prevent collisions with each other. We compute these sets following the methodology in [14]. Each robot may select a control input from a set of admissible values:

$$u_A \in \mathcal{U_A} = [u_{Am}in, u_{Am}ax],\ u_B \in \mathcal{U_B} = [u_{Bm}in, u_{Bm}ax] \tag{3}$$

where $\{u_{Am}in, u_{Am}ax, u_{Bm}in, u_{Bm}ax\}$ are predefined upper and lower angular velocity bounds. The backward reachable set is then obtained by solving the Hamilton-Jacobi-Isaacs Partial Differential Equation (HJI PDE):

$$\frac{\partial v(x,t)}{\partial t} + \min[0, H(x, \nabla v(x,t))] = 0,\ v(x,0) = g(x), \tag{4}$$

with Hamiltonian (replacing p with $\nabla v(x,t)$)

$$H(x,p) = \max_{a \in \mathcal{U_A}} \min_{b \in \mathcal{U_B}} p \cdot f(x,a,b), \tag{5}$$

where $g(x)$ is a scalar function representing the target set:

$$g(x) = \sqrt{{x_1}^2 + {x_2}^2} - r_c. \tag{6}$$

The HJI PDE is solved backwards in time up to some time horizon τ, to give the backward reachable set

$$S(\tau) = \{x \mid v(x, -\tau) \leq 0\}. \tag{7}$$

Figure 2 shows some examples of reachable sets computed for the system dynamics defined above, for various initial conditions and parameters. These sets and their approximations form the basis of our motion planning algorithms, by defining state space regions that should be avoided to prevent collisions. For a more detailed coverage of the convergence properties of this method, see [14].

3.3 Template Estimation

Strategic adversarial domains like robotic soccer require agents to cope with the *unknown* and potentially *non-stationary* behaviour of their opponents. The objective of a soccer-playing agent may dynamically change, for example, from navigating to the ball to making a maneuver to mark an adversary. This is in contrast to the aircraft collision avoidance example described in Section 2.1, where aircrafts are modeled as consistently seeking to avoid collision.

Time horizon τ (s)	**0.1**	**0.5**	**1.0**	**2.0**	**3.0**	**4.0**	**5.0**
Running time (s)	2.94	9.43	15.56	31.30	46.56	60.61	76.01

Table 1. HJI PDE running time for various horizon times.

We model different strategic behaviours as a collection of reachable sets, each corresponding to a hypothesis on the capabilities of the adversaries. Ideally, robots would compute different sets at each time step, depending on their aggregated observations. However, this is not feasible because of the computational expense associated with the solution of the HJI PDE backwards in time (Table 1). Instead, we select a fixed number of hypotheses, for which we compute the reachable sets *offline*. We complement the definitions of Section 3.2, by defining a set of *admissible linear velocities* for the adversarial agent r_B:

$$v_B \in \mathcal{V}_\mathcal{B} = [v_{Bm}in, v_{Bm}ax] \tag{8}$$

A similar argument applies to the set of admissible bounds $\mathcal{V}_\mathcal{A}$ for agent r_A. Both pairs of bounds depend on the *physical* capabilities of the robots (e.g. the maximum velocities permitted by their hardware). We first discretise $\mathcal{V}_\mathcal{A}$ and $\mathcal{V}_\mathcal{B}$ to obtain a countable set of pairs $\mathcal{VP} = \{v_A \in \mathcal{V}_\mathcal{A},\, v_B \in \mathcal{V}_\mathcal{B}\}$. For each pair $(v_A, v_B) \in \mathcal{VP}$, we compute a backward reachable set as in the previous section, thus obtaining a finite collection of *templates*.

Figure 2 shows an example of estimation of different reachable set templates, using Mitchell's toolbox [10]. The parameters reflect different extrema in the combinations of strategies followed by the two robots. For example, the *red* set depicts the worst case where r_B is moving at full speed and r_A is stationary. The right half of Figure 2 illustrates an irregular set that is not easily approximable.

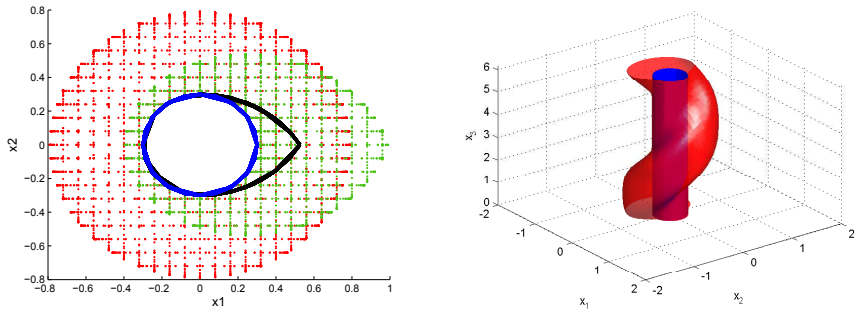


Fig. 2. Left: Two-dimensional projection of reachable set templates. Fixed parameters: $r_c = 0.3m$, $u_Amin = u_Amax = -u_Amin = -u_Bmin = 1.6rad/s$. Blue set: $v_A = 0.0m/s$, $v_B = 0.0m/s$. Black set: $v_A = 0.5m/s$, $v_B = 0.0m/s$. Green set: $v_A = 0.5m/s$, $v_B = 0.5m/s$. Red set: $v_A = 0.0m/s$, $v_B = 0.5m/s$. Right: The full three-dimensional version of the red and blue sets.

3.4 State Estimation and Reachable Set Composition

We use a particle filter estimation algorithm, in order to convert *observations* of the adversaries into *beliefs* with associated probabilities (Algorithm 1). The algorithm runs continuously, generating new beliefs based on the most recent observations. At each step, the robot queries its sensors (GetLatestObservations), and attempts to match them to the best past beliefs (ClusterObservations). The clustered observations are then passed through the particle filter to update the likelihood of each belief. We have implemented a variant of the Sampling-Importance-Resampling algorithm introduced in [9].The particle filter also helps alleviate some of the oscillation problems that arise in reactive path planning.

Beliefs are used to compute the relative state x of the adversary, as defined in Section 3.1. These estimates are then used to select the best reachable set **dynamically**, and plan trajectories that are optimal with respect to the current beliefs. When a robot is *outside* the chosen reachable set, it plans trajectories with respect to its boundaries. Similarly, when it is *inside*, it seeks to find the nearest point on its boundary, so it can exit and reach a safe configuration again.

4 Results

We demonstrate the application of our algorithm in the context of various robotic soccer scenarios. First, we present results on a 3-D strategy simulator, which was developed to comply with the specification of the RoboCup Standard Platform League [3]. Then, we present experiments on real NAO robots [2].

4.1 Simulation

We have developed a MATLAB 3-D strategy simulator for the NAO robots. The soccer field (Figure 3-top left) has dimensions of 6x4m, and the ball is

Algorithm 1 Adversary State Estimation

```
1: BeliefEstimator(NA, NP, π)
2: INPUT:no. of adversaries NA, particles per adversary NP, proposal distribution π
3: Ps, Ws, Os, Bs ← ∅ {Particles, weights, observations, beliefs}
4: for i = 1 to NA do
5:     Bs(i) ← RANDOM_POINT
6:     for j = 1 to NP do
7:         Ps(i,j) ← RANDOM_POINT
8:         Ws(i,j) ← 1/NP
9:     end for
10: end for
11: while true do
12:     Os ← GETLATESTOBSERVATIONS {vision}
13:     Os ← CLUSTEROBSERVATIONS(Os, Bs) {find nearest past beliefs}
14:     for i = 1 to NA do
15:         if Os(i) == NULL then
16:             Os(i) ← Bs(i) {no current observation, use last belief}
17:         end if
18:         Bs(i), Ps(i), Ws(i) ← PARTICLEFILTER(Os(i), Ps(i), Ws(i))
19:     end for
20: end while
```

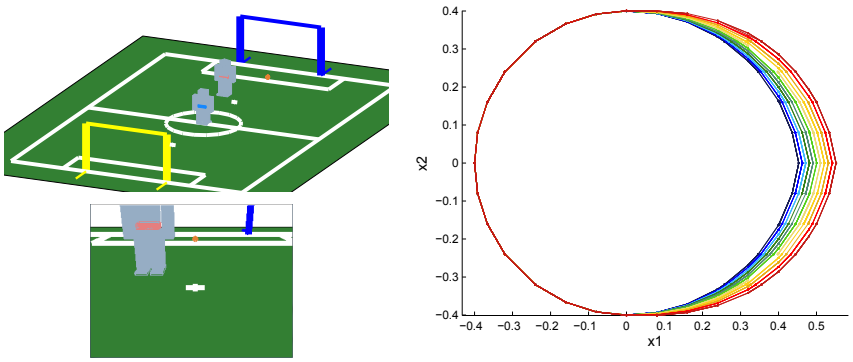
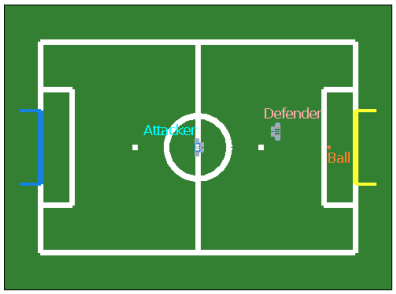


Fig. 3. *Top Left:* Simulated field with two robots. *Bottom Left*: The view of the blue robot. *Right:* Two-dimensional projection of reachable set templates for the NAO humanoid robots. The selected parameters are $r_c = 0.4m, u_Amin = u_Amax = -u_Amin = -u_Bmin = 1.7rad/s$, $v_A = 0.1m/s$, $v_B \in \{0, 0.1\}$, with v_B sampled every $0.01m/s$

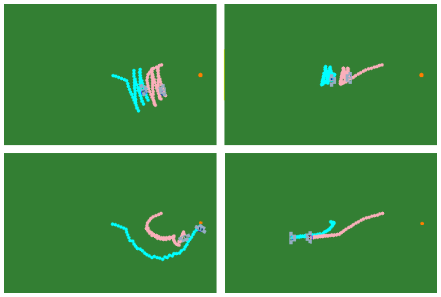
an orange sphere with a radius of 3cm. Each simulated robot is equipped with a 58° field-of-view camera (Figure 3-bottom left) with which it perceives the world. Sonar sensing is not modeled in this setup. Robots detect each other through waistbands, which are coloured light blue and pink for the two teams. Locomotion dynamics are not modeled explicitly, although noise can be added to executed movements. Uncertainty in sensing arises from the *egocentric* nature of belief estimation, which impacts the visibility of adversarial robots.

We first compute reachable set templates based on the hardware specifications [2] of the NAO (Figure 3-right). The velocity bounds were selected to comply with the physical limits of the NAO. The outer sets represent conservative hypotheses, where adversary r_B moves with high velocity towards r_A.

One-versus-one game We first consider the case of a one-versus-one game of soccer between two robots (Figure 4(a)). We assign a different role to each robot - the goal of the *attacker* is to navigate to the ball, while avoiding the *defender*. In a reachable set context, the defender selects the template which best models his adversary, and then tries to move towards its boundary.



(a) Initial configuration. The attacker (blue robot) is located on the halfway line, with the defender further away.



(b) 1-v-1 examples. Format: (ATTACKER, DEFENDER). *Top left:* (RSC,RSC). *Top right:* (TSB,TSB). *Bottom left:* (RSC,TSB). *Bottom right:* (TSB,RSC).

Fig. 4. One-versus-one experiments

The reachable set composition heuristic is compared against a benchmark inspired from the collision-avoiding velocity and dynamic window ideas presented in [8] and [15]. This algorithm allows the robot to move with its current velocity v_A, as long as it is outside the *target* set of unsafe velocities (defined in terms of the adversary's current *velocity* v_B). When inside, it computes the nearest point p_b on the boundary of the unsafe set, and uses it to define a set of safe velocities:

$$\mathcal{V}_{\mathcal{S}} = \{v \mid (v - (v_A + p_b)) \cdot \frac{p_b}{|p_b|} \geq 0\} \tag{9}$$

that may lead it out of the unsafe set. The robot then selects the velocity $v_S \in \mathcal{V}_{\mathcal{S}}$ that will minimise the distance to its current goal. This variant was chosen to reflect the difference between planning with respect to a *target* set and planning with respect to multiple *reachable* sets. We refer to the two algorithms as **RSC** (Reachable Set Composition) and **TSB** (Target Set Benchmark) respectively. In summary, the two algorithms differ in the state set they use when planning (reachable vs target), as well as the space they operate in (position vs velocity).

Strategies are initially evaluated in a **fully observable** setting. Agents are provided with exact information on the location of the ball and their adversary.

Table 2. 1v1, partial observability. Success rate and mean time to goal (MTTG)

(a) Reachable sets vs benchmark

(Att,Def)	Success %	MTTG
(RSC,RSC)	72%	63.28 steps
(TSB,TSB)	44%	64.5 steps
(RSC,TSB)	98%	39.5 steps
(TSB,RSC)	16%	60.5 steps

(b) Different reachable set algorithms.

(Att,Def)	Success %	MTTG
(RSC,SRS-BC)	38%	58.5 steps
(SRS-BC,RSC)	42%	72.5 steps
(RSC,SRS-WC)	82%	67.8 steps
(SRS-WC,RSC)	40%	60.1 steps

Figure 4(b) shows all strategy combinations for the attacker and the defender. In the RSC-TSB combination, the attacker benefits from the adaptive nature of the template algorithm, eventually finding a path to the goal.

We now investigate the more challenging **partially observable** case. We also add random Gaussian noise to the commanded movements of the robot, with mean equal to the magnitude of the command and a standard deviation of 1. We ran 50 trials for each strategy combination. Each trial was simulated for 80 discrete simulation steps, or until the attacker reached the goal.

Table 2(a) summarises these results, where RSC maintains its superiority over TSB. As with full observability, this difference is stronger when the attacker and the defender use different algorithms. However, even when the heuristics are the same ((RSC,RSC), (TSB,TSB)), reachable set composition manages to steer the attacker to the goal more reliably. This also means that RSC defenders are less successful against RSC attackers, as the reactive nature of the game prevents them from both tracking and blocking their opponents' moves robustly.

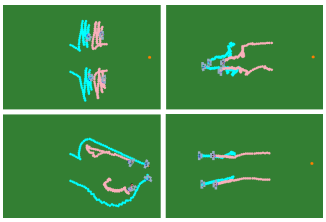
As a further comparison, we also compare the RSC algorithm to two variants of path planning algorithms using a *single* reachable set. The first variant plans with respect to the *best-case* reachable set, where the adversary is hypothesised as always moving with a minimum linear velocity. The second variant takes into account only the *worst-case*, where the adversary moves with the maximum allowed velocity. We term the two variants **SRS-BC** (Single Reachable Set - Best Case) and **SRS-WC** (Single Reachable Set - Worst Case) respectively.

Table 2(b) summarises the computed metrics for this case. In terms of success rate, the different variants are comparable; this is largely due to the small discrepancy between the best- and the worst-case reachable set (Figure 3). The results also appear to favour optimistic defensive strategies (e.g.(RSC,SRS-BC)), which allow defenders to move closer to the attacker when marking. Nevertheless, the RSC algorithm offers some improvement in the time to reach the goal.

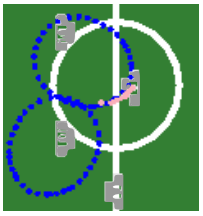
Multi-robot games In this example, the blue team now consists of two attackers, and the pink team of two defenders. However, no cooperation or information exchange between teammates is allowed. The attacking team succeeds if at least one of its members reaches the ball; the defenders succeed if they prevent both attackers from reaching the ball. Figure 5(a) shows the initial configuration.

The algorithms are once again evaluated on the four different permutations of the RSC and TSB algorithms under full observability. Figure 5(b) displays a pattern similar to Figure 4(b). In most cases, the resulting trajectories vary among

(a) Initial configuration.



(b) 2-v-2 examples. Format: (ATTACKER, DEFENDER). *Top left:* (RSC,RSC). *Top right:* (TSB,TSB). *Bottom left:* (RSC,TSB). *Bottom right:* (TSB,RSC).



(c) Multiple reachable set defending example - agent plans path with respect to the intersection of the sets.

(Att,Def)	**Success %**	**MTTG**
(RSC,RSC)	90%	61.4 steps
(TSB,TSB)	62%	56.0 steps
(RSC,TSB)	94%	46.2 steps
(TSB,RSC)	22%	75.0 steps

(d) Performance statistics.

Fig. 5. Two-versus-two experiments

teammates. This discrepancy is due to the presence of the additional adversary, who influences the selection of appropriate landmarks for path planning.

Table 5(d) summarises the results for 50 runs in the partially observable case. Compared to the one-versus-one experiment, partial observability is more favourable towards attackers, as seen by the higher success rate in most cases. However, the RSC algorithm still behaves more robustly than the TSB variant, despite the additional constraints. The significant improvement in the attackers' success rate is partly explained by the presence of multiple robots in the field, which makes disambiguation and tracking difficult during close interactions.

The robustness of the RSC algorithm under both full and partial observability is a strong indication of its suitability to multi-agent path planning. When planning a path with respect to multiple adversarial agents, it is sufficient to consider the intersection of their reachable sets as a safety criterion. Figure 5(c) illustrates how a defender uses this intersection to plan a path that accounts for both attackers. The plotted sets correspond to the templates computed in Figure 3, which are translated and rotated based on the current output of the particle filter algorithm - this estimate may not always correspond to the exact true state of the adversary. In this noisy setting, RSC may generate safer trajectories than simpler heuristic algorithms that do not explicitly consider the evolution of system dynamics over some time horizon.

4.2 Physical Robot

We conclude by presenting some illustrative applications of our algorithm on the NAO humanoid robots. Similarly to the simulation, robots identify each other

using coloured waistbands, with the same observability constraints as in section 4.1. Imperfect sensing arises naturally in this setup, so each robot runs a variant of Algorithm 1 to deal with incomplete and/or noisy vision estimates.

The behavioural models have been adjusted to comply with the hardware and processing limitations of the humanoids. We consider one-versus-one collision avoidance tasks, against a variety of adversaries with unknown strategies. The attacking robot always runs the RSC algorithm. The defender was programmed to execute one of the following behaviours: move towards the attacker, move in a circle around the ball, move vertically with respect to the attacker, also execute the RSC algorithm in order to navigate to a second ball, and move randomly.

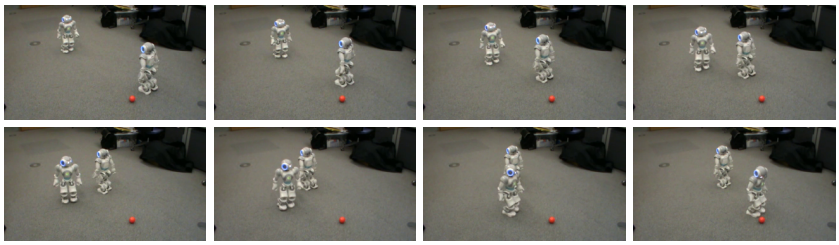


Fig. 6. NAO going to the ball while avoiding an adversary moving directly towards it.

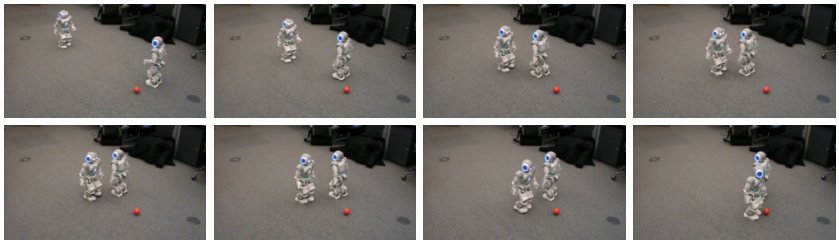


Fig. 7. NAO going to the ball while avoiding an adversary moving randomly.

Our supporting **video** [1] illustrates the above examples on NAO robots. Screen shots from two examples are given in Figures 6 and 7. In both cases, the robot manages to successfully reach the goal, despite losing track of the ball or its adversary for long periods of time. This demonstrates the robustness of the composable reachable set algorithm under realistic constraints and conditions.

5 Conclusions

We have presented an algorithm for autonomous online path planning in realistic adversarial conditions, where the adversary's capabilities and strategy profiles can not be characterised or estimated precisely. Our algorithm extends the concept of reachable set computation for safe trajectory generation, to incorporate bounds and hypotheses on adversarial behaviours. By generating suitable templates offline and composing them dynamically online, the algorithm outperforms

simpler modeling heuristics. Thus, our results favour the notion of planning with respect to an infinite-horizon reachable set of states. This superiority is reflected in both the success rate of reaching a given goal and the time taken to reach it.

Our reachable set template computation builds on very simple hypotheses on the adversary's linear velocity bounds. Depending on the parameters of the problem, this may lead to small (Figure 3-right) or large (Figure 2) between best- and worst-case templates. Future work would be to utilise more sophisticated distributions over reachable sets, while also incorporating more elaborate coarse dynamics models that go beyond the scope of rectangular velocity bounds.

Composable reachable sets may also be beneficial in numerous applications beyond multi-robot path planning. One obvious example is Human-Robot Interaction, where the human agent's strategies and behaviours can not be modeled exactly but we still want strategically meaningful behaviour on the part of the robot. This is an area of future work for us.

References

1. Supporting material - http://www.youtube.com/watch?v=BfJgWz4TwlE.
2. NAO robot documentation - http://academics.aldebaran-robotics.com/.
3. RoboCup SPL rules - http://www.tzi.de/spl/. pages 1–27, 2010.
4. J. Bruce. *Real-Time Motion Planning and Safe Navigation in Dynamic Multi-Robot Environments*. PhD thesis, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, January 2006.
5. R. R. Burridge, A. A. Rizzi, and D. E. Koditschek. Sequential composition of dynamically dexterous robot behaviors. *IJRR*, 18(6):534–555, 1999.
6. J. Ding and C. J. Tomlin. Trajectory optimization in convex underapproximations of safe regions. In *CDC*, pages 2510–2515, 2009.
7. P. Fiorini and Z. Shillert. Motion planning in dynamic environments using velocity obstacles. *International Journal of Robotics Research*, 17:760–772, 1998.
8. D. Fox, W. Burgard, and S. Thrun. The dynamic window approach to collision avoidance. *IEEE RAM*, 4(1):23 –33, March 1997.
9. N. J. Gordon, D. J. Salmond, and A. F. M. Smith. Novel approach to nonlinear/non-gaussian bayesian state estimation. *Radar and Signal Processing*, 140(2):107–113, August 1993.
10. I. M. Mitchell and J. A. Templeton. A toolbox of hamilton-jacobi solvers for analysis of nondeterministic continuous and hybrid systems. In *HSCC 2005*, pages 480–494. Springer-Verlag, 2005.
11. E. Rimon and D. Koditschek. Exact robot navigation using artificial potential functions. *IEEE Trans. Rob. Aut.*, 8:501–518, 1992.
12. R. Tedrake. LQR-trees: Feedback motion planning on sparse randomized trees. In *Proceedings of Robotics: Science and Systems*, Seattle, USA, June 2009.
13. C. J. Tomlin, J. Lygeros, and S. S. Sastry. A game theoretic approach to controller design for hybrid systems. In *Proc. IEEE*, pages 949–970, 2000.
14. C. J. Tomlin, I. Mitchell, A. M. Bayen, and M. Oishi. Computational techniques for the verification of hybrid systems. In *Proc. IEEE*, pages 986–1001, 2003.
15. J. van den Berg, S. J. Guy, M. C. Lin, and D. Manocha. Reciprocal n-body collision avoidance. In *International Symposium on Robotics Research*, 2009.
16. P. Varaiya. Hierarchical control of semi-autonomous teams under uncertainty. In *Final report of Darpa Contract F33615-01-C-3150*, 2004.