Efficient Physics-Based Planning: Sampling Search Via Non-Deterministic Tactics and Skills

Stefan Zickler
Computer Science Department
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213, USA
szickler@cs.cmu.edu

Manuela Veloso
Computer Science Department
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213, USA
veloso@cs.cmu.edu

ABSTRACT

Motion planning for mobile agents, such as robots, acting in the physical world is a challenging task, which traditionally concerns safe obstacle avoidance. We are interested in physics-based planning beyond collision-free navigation goals, in which the agent also needs to achieve its goals, including purposefully manipulate non-actuated bodies, in environments that contain multiple physically interacting bodies with varying degrees of controllability. Physics-based planning is computationally hard due to the large number of continuous motion actions and to the difficulty in accurately modeling the rich interactions of such controlled, manipulatable, and uncontrolled, potentially adversarial, bodies. We contribute an efficient physics-based planning algorithm that uses the agent's high-level behaviors to reduce its motion action space. We first discuss the general physics-based planning problem. We then introduce *Tactics and Skills* as a model for infusing goal-driven, higher level behaviors into a randomized motion planner. We present a physics-based state and transition model that employs rigid body simulations to approximate real-world interbody-dynamics. We introduce and compare two variations of our tactics-driven, physics-based planning algorithm, namely Behavioral Kinodynamic Balanced Growth Trees and Behavioral Kinodynamic Rapidly-Exploring Random Trees. We tested our physics-based planners in a variety of rich domains and show results in simulated domains where the agent manipulates an object in a dynamic non-adversarial and adversarial environment, namely in a robot minigolf and robot soccer domain, respectively.

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search—*Plan execution, formation, and generation*

Keywords

Planning, Physics-Based, Kinodynamic, Tactics, Behavioral, Robot, Rigid Body, Control

Cite as: Efficient Physics-Based Planning: Sampling Search Via Non-Deterministic Tactics and Skills, Stefan Zickler, Manuela Veloso, *Proc. of 8th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2009),* Decker, Sichman, Sierra and Castelfranchi (eds.), May, 10–15, 2009, Budapest, Hungary, pp. 27 – 34

1. INTRODUCTION

Autonomous mobile agents, such as robots, face the challenge of safely navigating in their environments. Traditional motion planning focuses on the sole problem of the safe navigation through an obstacle-ridden environment. However, when embedded in a task, such mobile agents have behaviors and goals to achieve which could be used combined with their motion plans. We are hence interested in planning problems where an agent has to achieve goals which go beyond plain collision-free navigation. In particular, we aim to solve control problems involving multiple physically interacting bodies with varying degrees of controllability. In such *physics-based* domains, an agent may not only need to generate trajectories that are collision-free, but may also need to purposefully manipulate non-actuated bodies, in environments that are potentially adversarial.

Planning in such domains is very challenging for several reasons. The search space of an agent's possible control actions in a continuous physical environment is vast and the set of valid solutions is likely to be highly constrained, especially if complex dynamics, such as purposeful object manipulations, are required. The presence of adversaries increases this difficulty by further narrowing the solution space and by introducing the need for opponent models which can predict the reactive controls of adversarial bodies. Another computational challenge is to accurately model all of the bodies' rich physical interactions and dynamics, such as momentum, friction, and collisions.

We have extensively worked with domains where robots navigate with sophisticated motion planning, but which does not take into account the agent's behaviors and goals [17, 6, 4]. In this work, we introduce and evaluate an efficient physics-based planning approach that uses high-level behavioral control models to effectively reduce the agent's motion search space and thus provide an efficient way for solving tactically and physically complex domains. We consider non-deterministic high-level behavioral control models that allow the planner to effectively sample its search space, as well as predict actions of similar bodies in the environment.

The paper is organized as follows. We first review related work. We then formally define the physics-based planning problem and present a state and transition model that employs rigid body simulations to approximate real-world interbody-dynamics. We define the different types of bodies that can be encountered in a physics-based domain and discuss some of the unique challenges associated with planning

goals involving these bodies. Next, we introduce sampling-based *Tactics* and *Skills* as a model for infusing goal-driven, high-level behaviors into a randomized motion planner. We introduce and compare two variations of our tactics-skills-driven randomized physics-based planning algorithm, namely Behavioral Kinodynamic Balanced Growth Trees (BK-BGT) and Behavioral Kinodynamic Rapidly-Exploring Random Trees (BK-RRT). Finally, we evaluate and compare these algorithms experimentally, discuss future work, and conclude with a summary of the contributions.

2. RELATED WORK

There is a vast body of work regarding motion planning in continuous domains. One of the most popular algorithms is Rapidly-Exploring Random Trees (RRT). Introduced by LaValle [11], RRTs are based on the idea of growing a search tree through the configuration space with the hope that one of the tree's leafs will eventually reach the goal-state. The advantage of the RRT algorithm is that it employs sampling-techniques which ensure that the resulting tree will tend to rapidly cover the reachable configuration space. RRT has been shown to be usable for navigation planning problems which involve dynamics and kinematic constraints [12].

Traditional sampling-based motion planning algorithms, such as RRT, are typically employed to solve the problem of collision-free robot navigation [5, 13, 15]. We are however interested in solving planning problems with goals that go beyond the simple navigation to a target point, and which can require complex, multi-body interactions and manipulations. To tackle such difficult problems, our work aims to integrate behavior-based agent control methods into the motion planning algorithm. There exist many behavior-based robot control architectures [3, 14, 2, 8, 10]. Our work integrates and significantly extends the single agent control blocks of the *Skills, Tactics, and Plays (STP)* architecture [3].

There are some related approaches which have also integrated behavioral models into dynamics-based motion planning [9, 16]. However, these approaches are aimed at solving computer graphics problems, and therefore assume full controllability of all agents and other bodies in the domain. As such, these approaches are inapplicable to solve physics-based robot planning problems that require the manipulation of passive bodies. Furthermore, the "behaviors" used in these graphics approaches consist of pre-recorded computer animation sequences that are unable to make state-dependent decisions as required in most robot domains.

3. PHYSICS-BASED PLANNING

Similarly to a general planner, given a state space X, an initial state $x_{\text{init}} \in X$, and a set of goal states $X_{\text{goal}} \subset X$, a motion planner searches for a sequence of actions $a_1, \ldots, a_n$, which, when executed from x_{init}, ends in a goal state. The state space is assumed to represent a physical geometry, and the actions correspond to bodies' actuation controls.

We use the term *Physics-Based Planning* for action models that aim to reflect the inherent physical properties of the real world. The *Rigid Body Dynamics* model [1] provides a computationally feasible approximation of basic Newtonian physics, and allows the simulation of the physical interactions between multiple mass-based bodies, under the assumption that such bodies are non-deformable. The term *Dynamics* means that rigid body simulators are second order systems, able to simulate physical properties over time, such as momentum and force-based inter-body collisions.

More formally, let the Physics State Space X describe the entire variable space of the physical domain and let us assume the presence of n rigid bodies. A state $x \in X$ is defined as $x = [t, r_0, \ldots, r_n]$, where t represents time, and r_i represents the state of the i-th rigid body. A rigid body state in a second order system is described by its position, rotation, and their derivatives. That is, $r = [p, q, v, \omega]^T$ where:

- p : position *(3D-vector)*
- q : rotation *(unit quaternion or rotation matrix)*
- v : linear velocity *(3D-vector)*
- ω : angular velocity *(3D-vector)*.

The action space A is the set of the applicable controls that the physics-based planner can search over. An action $a \in A$ is typically defined as a vector of subactions $[a_{r1}, \ldots, a_{rn}]$, where a_{ri} is a pair of 3D force and torque vectors applicable to a corresponding rigid body state r_i.

A physics-based planner chooses actions by reasoning about the states resulting from the actuation of possible actions. The state computations are done by simulation of the rigid body dynamics. There are several robust rigid body simulation frameworks, freely available, (e.g., Open Dynamics Engine (ODE), Newton Dynamics, and NVIDIA PhysX). Frequently referred to as *physics engines*, these simulators are used as a "black box" by the planner to simulate state transitions in the physics space, as illustrated in Figure 1.

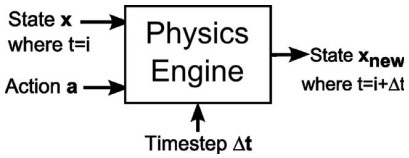

Figure 1: A Physics Engine returns state transitions.

Given a current physics state of the world x in combination with a control action vector a, the physics engine is then able to simulate the rigid body dynamics forward in time by a fixed timestep Δt. During the planner's forward search, the physics engine then internally resolves any inter-body collisions and delivers a complete new planning state x_{new}.

3.1 Rigid Body Types

Planning for a solution sequence of physical actions is clearly related to the types of bodies present in the domain. We classify the types of bodies in the domains of the physics-based planner, using a hierarchy as shown in Figure 2.

Every body is by definition a *rigid body*. There are *static* rigid bodies that do not move, even when a collision occurs, which are often used to model the ground plane and all non-movable bodies, such as walls and heavy objects. All other bodies are *manipulatable*, meaning that they react to collision forces exerted upon them. Among these, the planner can directly control the *actively controlled* bodies, i.e., it has available actions directly applicable to these bodies.

Interestingly, there are two different types of bodies that are manipulatable but not actively controlled, namely the *passive* and *foreign controlled* bodies. *Passive* bodies can

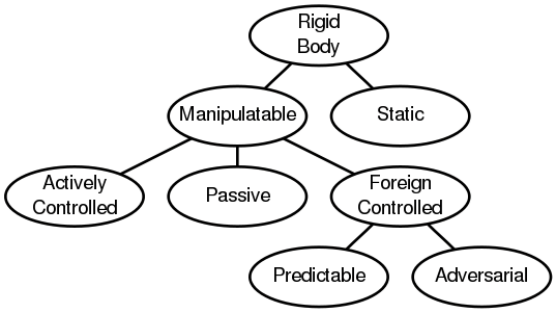

Figure 2: Rigid body classes

only be actuated by external influences and interactions, such as being carried or pushed. The *foreign controlled* bodies are actively actuated, but by external control to our planner. But such *foreign controlled* bodies can have *predictable* motion, such as an escalator or a windmill, or be *adversarial* capturing the challenges of their motion modeling by the planner. In physics-based games, such *adversarial* bodies can represent the intelligently controlled opponent agents.

3.2 Planning Challenges

Particularly challenging physics-based planning problems involve planning goals defined in terms of the state of a passive body. A solution then requires the manipulation of the passive body through the means of collisions exerted by the actively controlled body. The search is difficult as there are no simple heuristics to guide the selection of the actuation choices of the controllable body in order to bring the passive body closer to its goal state, in presence of the complex effects of the rigid body dynamics. The planner encounters even further complications if the domain contains adversarial bodies which might actively attempt to prevent such manipulations from succeeding.

This disconnect between the action space of actively controlled bodies and the goal state of passive bodies makes it virtually impossible to construct a simple metric-based planning heuristic. However, we still need to bias the search by some heuristic, as otherwise the search time grows dramatically, making online planning completely infeasible. The contributions of our work include the use of high-level behaviors as such heuristics for the physics-based planner.

4. TACTICS AND SKILLS

To overcome the challenge of searching through the vast physics-based control space, we use a high-level behavioral control model in a novel way to reduce the search space.

Among the many reactive behavioral control architectures, e.g., [2, 8, 10], we choose the *Skills, Tactics, and Plays (STP)* architecture, as it has effectively been used in real-time adversarial robotics domains [3]. In STP, a *Play* captures the behaviors of a group of multiple agents. A single agent's behavior is modeled as a reactive *Tactic*, representing a finite state machine (FSM) of lower-level *Skills*, which act as pre-programmed, reactive control blocks. Our work involves single-agent planning, therefore does not use Plays. Traditionally, the STP runs online, as a policy-based controller, without any physics-aware planning at the Tactics level.

In our work however, our physics-based planner uses Tactics and Skills as an action sampling model, and we need to extend Tactics and Skills to be non-deterministic. Instead of greedily executing a single Tactic online, we effectively use planning to simulate the outcome of many different variations of the Tactic executions in simulated physics-space. The planner selects a good goal-achieving simulated solution for execution. We modify the traditional definition of a Tactic to be modeled probabilistically as a non-deterministic FSM. Similarly, we create a non-deterministic version of Skills which uses random sampling to choose from a set of possible control actions. In summary, instead of being a reactive controller as in traditional STP, the Tactic's new role is to guide the planner's search by imposing constraints on the searchable action space. Before we explain in detail in the next section the use of the Tactics in the planning algorithm, we now further define the non-deterministic Tactics and Skills.

Figure 3 shows an example diagram of a Tactic. Internally, a Tactic contains a FSM where each state represents a Skill. Each Skill within the Tactic acts as a control block, being able to read the current state of the world x and producing a control sub-action a_{ri}. The tactic itself determines how to transition between different Skills over time, thus taking the state of the world x and a pointer to the currently active Skill $\text{fsm-state}_{\text{source}}$ as an input, and producing a pointer to the newly active Skill $\text{fsm-state}_{\text{new}}$ as output.

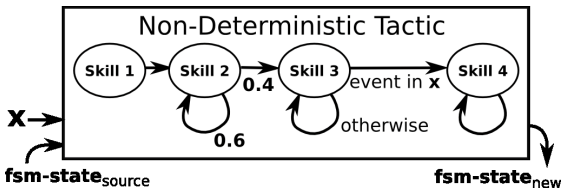

Figure 3: Example of a non-deterministic Tactic

Transitions between these Skills can be deterministic (such as between "Skill 1" and "Skill 2" in Figure 3), they can be modeled non-deterministically through user-defined transition probabilities (such as the outgoing transitions of "Skill 2"), or they can programmatically depend on any desired property of the current state of the world x (such as the outgoing transitions of "Skill 3"). The selection of Skills and the actual transition probabilities and events are defined by the developer. By being modeled in this non-deterministic fashion, each execution of the Tactic can result in a different sequence of state-transitions, thus covering different parts of the search space. However, a developer is in full control over how much "freedom of search" he or she wants to provide to the Tactic by adjusting its transition probabilities.

The purpose of the Tactic's individual Skills is to act as a control model that provides a range of different actions from which the planner can choose when searching. Traditionally, in its reactive mode, a Skill is deterministic and returns a single action. Instead, we extend a Skill to compute a continuous set of actions and make it non-deterministic through the use of sampling among the possible actions (see Figure 4).

Instead of deterministically producing a single control action a_{ri} based on the current state vector x, the Skill is defined to evaluate x and to produce a continuous set of actions A_{ri}. It then uses random sampling to return a single

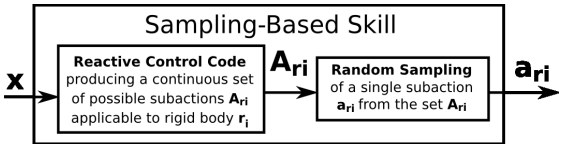

Figure 4: Structure of a sampling-based Skill

non-deterministic action a_{ri}. Effectively, this means that each Skill execution produces a different viable perturbation of the Skill's typical control behavior, thus providing a sampling-based model which our planner uses to search.

The Skill's action generation function and sampling model are predefined and assume a certain higher level knowledge about the domain's goal and its physical properties, such as the reasonable ranges of applicable forces and torques. Typically, a single Skill implements a particular control objective such as "drive towards a sampling-based location," "push some target rigid body towards a sampling-based target region," or "turn a sampling-based amount." Designing these actual Skills does take some programmatic effort and it can be argued that some of the Skills should be considered domain-dependent. However, the STP architecture exactly provides for Skills to be used as template-like "building blocks" that apply to a variety of domains. Given a library of Skills, it is feasible to rapidly create an intelligent tactical model that is well-suited for solving a particular domain. The effectiveness of our approach arises from the fact non-deterministic Tactics and Skills can encode as much or as little domain-dependent information as desired. Such encoding provides full continuous control over the trade-off between choosing a more general, but possibly less efficient Tactic, or a highly domain-optimized and efficient Tactic. The use of different high-level Tactics and Skills allows for the feasibility of the search for physics-based planning in many domains.

5. PLANNING ALGORITHM

We are now ready to introduce the planning algorithm. We present two variations of the algorithm that share the same fundamental physics-based, forward-planning loop, but differ in the way they control the growth of the search tree. We name these two variations as Behavioral Kinodynamic Rapidly-Exploring Random Trees (BK-RRT) and Behavioral Kinodynamic Balanced Growth Trees (BK-BGT).

In order to use Tactics and Skills as a behavioral sampling model for our planner, we integrate their internal states into the state space. More formally, a state $x \in X$ is now defined as $x = [t, \mathit{fsm\text{-}states}, r_0, \ldots, r_n]$ where *fsm-states* represents a vector of the current internal FSM states of all Tactics used in the domain. The number of Tactics is dependent on the number and types of bodies within the domain. Each actively controlled body has a corresponding non-deterministic Tactic consisting of sampling-based Skills from where its actions are sampled. Additionally, we construct reactive, deterministic Tactics with deterministic Skills to act as prediction models for each foreign-controlled, and especially also to approximate the model of each adversarial body, if existing. Even if an adversary body's exact Tactic is not known, it may be useful to still model its roughly expected behavior rather than assuming it is static. Finally, passive bodies in the domain are non-actuated and do not require a Tactic.

Algorithm 1 shows the main planning loop. We initialize the search with a tree T containing an initial state $x_{init} \in X$. We then enter the main planning loop, which runs for a predefined domain-dependent maximum number of search iterations k, if no solution is found earlier. On each iteration, the algorithm selects which node x_{source} to extend by using the `SelectNode` node selection function.

Algorithm 1: BK-BGT / BK-RRT Planning Loop

```
T.AddVertex(x_init);
for k ← 1 to k do
    x_source ← SelectNode (T);
    fsm-states ← FsmTransitions(x_source.fsm-states);
    a ← ApplySkills(fsm-states,x_source);
    x_new ← Simulate(x_source,a,Δt);
    x_new.t ← x_source.t + Δt;
    x_new.fsm-states ← fsm-states ;
    if IsValidState(x_new) then
        T.AddVertex(x_new);
        T.AddEdge(x_source,x_new,a);
        if x_new ∈ X_goal then
            return x_new ;
        end
    end
end
return Failed;
```

The difference between BK-RRT and BK-BGT lies precisely on the node selection function. BK-RRT selects nodes similarly to the Rapidly-Exploring Random Trees (RRT) search, which allows rapid growth of a search tree through a continuous space by probabilistically sampling the space towards the goal or towards a random exploration target. RRT has been used for collision-free motion planning in many robotics applications, including dynamics-based navigation [11, 5, 12]. The Function `SelectNodeRRT(T)` shows the node selection scheme used in the BK-RRT search.

Function `SelectNodeRRT(`T`)`

```
s_random ← SampleRandomState();
return NNeighbor(T,s_random);
```

The function `SampleRandomState` uses an internal probability distribution to provide a sample s_{random} taken from the sampling space $\mathbf{S} \subseteq X$. The function `NNeighbor` then finds the state-node in the tree which is the nearest neighbor to s_{random}, according to some predefined distance metric. As with traditional RRT, it is important that the sampling space $\mathbf{S}$, the underlying probability distribution, and especially the distance metric are all carefully chosen to match the domain. For the rigid body domains tested, we define this metric as the shortest possible time that it could take an actively controlled body to reach s_{random}, defined as a randomly sampled target configuration of the actively controlled body within the confines of the domain. The advantage of such an RRT-based node selection scheme is that it results in a relatively efficient and probabilistically uniform coverage of the domain's geometric workspace. The downside of RRT-based node selection is that good distance metrics can be difficult to define and to compute in a physics-based environment. Furthermore, the nearest neighbor lookup becomes slower with growing tree sizes, resulting in a quadratic runtime of algorithm.

It is questionable whether it is worth to perform the wide sophisticated RRT node selection scheme if the agent's actions are already constrained by a Tactics model, therefore not really requiring a uniform growth through the domain. Considering the tactically constrained models, it makes more sense to abandon any attempts in modeling complex distance metrics that involve knowledge about bodies and the state-space, and instead focus on a fast and random growth of the search tree itself. To test this hypothesis, we introduce a novel less informed approach, but which has the objective of growing the search tree in a well-balanced fashion. We call this approach "Balanced Growth Trees," and its corresponding node selection function is shown in `SelectNodeBGT(T)`.

Function `SelectNodeBGT(`T`)`

if $\left(\frac{\texttt{AvgLeafDepth(T)}}{\texttt{AvgBranchingFactor(T)}}\right) > \mu$ **then**
  return `PickRandomNonLeaf(T)`;
else
  return `PickRandomLeaf(T)`;
end

The only parameter of the BGT search is a single constant μ that represents the desired ratio between the average leaf depth and the average tree branching factor. A large value of μ leads the algorithm to expand further into the future, but creates a "thinner" tree. A smaller value of μ focus on a more dense expansion, but with a limited average time horizon. As it is possible to keep running values of the average branching factor and leaf depth as the tree grows, this node selection scheme is able to run in constant time per node selection. Our experimental evaluation compares the use of the RRT and BGT node selection variations.

The search continues from the selected node x_{source}. Algorithm 1 calls `FsmTransitions` to perform all Tactics' internal FSM state transitions. Recall that the non-deterministic Tactic switches non-deterministically between Skills and the state vector's component fsm-states identifies which Skill is currently selected for each existing Tactic. The algorithm then applies the Skills associated with the fsm-states through the function `ApplySkills`, which invokes the sampling-based Skills as described in Section 3. The Skill takes the state of the world x as an input, and returns a non-deterministically chosen action a_{ri}. The algorithm is now ready to call the physics-engine through the function `Simulate`, with input the source state x_{source}, and the forces and torques defined by a, to simulate Δt forward in time of the physics of all the bodies. The physics-engine returns a new state x_{new}. The algorithm then checks the resulting state with a user-defined function `IsValidState` to ensure that the simulation from x_{source} to x_{new} did not violate any constraints which may be required for the domain. This additional validity check is optional and for many domains `IsValidState` simply returns *true*. If accepted, the algorithm adds x_{new} to the search tree T as a child of the chosen node x_{source}. The complete loop is repeated until the algorithm either reaches the goal, or until it reaches the maximum allowed number of iterations k, at which point the search returns failure for this state. (In robot domains, the algorithm can be repeatedly invoked within the sensing cycle for a new state.) Once the goal is reached, the algorithm simply traverses back to the root the root of T and returns the reversed path as the planned action sequence.

6. RESULTS

We tested our algorithm in a variety of simulated domains. We implemented the planner in C++, we chose NVIDIA PhysX as the underlying physics engine, and the results were computed on a Pentium 4. Animations of the planning and execution of the resulting plans and additional experiments can be found in videos available at (link removed for blind review). We used an action timestep Δt of 1/60th of a second. Table 1 shows average planning times and tree sizes in two testbeds, namely the "Minigolf" and the "Soccer" domains.

Minigolf Domain			
Algorithm	Time & StdDev	Nodes & StdDev	μ
BK-RRT	$5.3\text{s} \pm 1.1\text{s}$	13042 ± 1200	n/a
BK-BGT	$2.5\text{s} \pm 1.3\text{s}$	8094 ± 1067	1000

Soccer Domain			
Algorithm	Time & StdDev	Nodes & StdDev	μ
BK-RRT	$9.6\text{s} \pm 5.1\text{s}$	9752 ± 2157	n/a
BK-BGT	$11.1\text{s} \pm 2.1\text{s}$	11088 ± 1267	100

Table 1: Performance comparison of BK-RRT and BK-BGT. Each value represents an average over 8 experiments.

The goal in the "Minigolf" domain is to have a ball at a final location. The domain includes one controllable body, the robot, which needs to manipulate the ball through a course with one or more moving obstacles. Figure 5 shows the Tactic for the controllable robot body, which allows the body to wait for a sampling-based amount of time (which is significant as there are moving obstacles), then drive into the ball at a sampling-based speed and angle, and actuate the ball towards a sampling-based target.

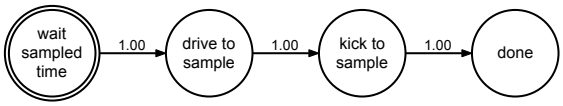

Figure 5: "Minigolf" Tactic for the controlled robot.

Note that the ball in this domain is a completely passive body that our planner can only move by generating purposeful collisions exerted from the robot body. Figure 6 shows a visual example from this domain. As can be seen in Table 1, the planner successfully finds solutions.

The "Soccer" domain (a single soccer attacking situation) significantly enhances the concept of goal-driven manipulation of a passive body, and truly demonstrates the unique tactical planning abilities of our planner. The robot-body features a complex tactical model, as shown in Figure 7.

The Tactic allows the robot to dribble the ball to a sampled location, kick or chip the ball towards a sampled narrow target, or towards a sampled point in the goal.

Additionally, we include in the domain three dynamic adversary rigid bodies which were running deterministic adversarial behaviors to block the ball from the goal. Note that from a *planning* perspective, this domain represents a very difficult problem. Not only does the robot need to navigate around moving bodies, but it also needs to exert accurate

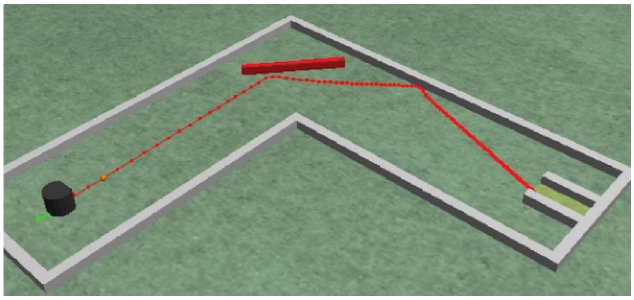

Figure 6: The "Minigolf" domain. The bar-shaped obstacle at the center of the course is rotating at constant velocity and thus represents a predictable, foreign-controlled body. The path shows an example of a legal solution found by the planner for the controllable body, the robot (dark cube, left): the robot waits an appropriate amount of time and then accurately manipulates the ball to use the rotating obstacle as a bounce-platform, leading it into the goal position (bottom right).

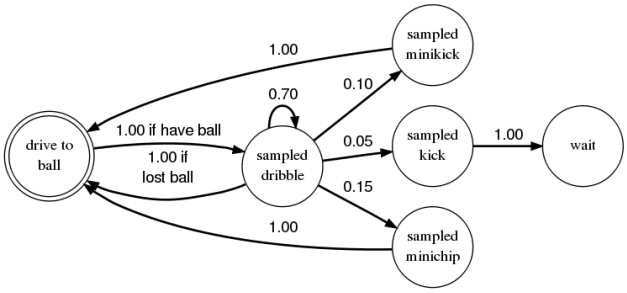

Figure 7: "Soccer" Tactic for the controlled robot.

control on the ball to achieve the high level scoring goal in an adversarial environment. This domain generated various interesting solutions, one of which is shown in Figure 8.

Looking at the performance values in Table 1, we can see that neither of the two node selection schemes has a constant advantage over the other. Instead, BGT seems to outperform RRT in the "Minigolf" domain, whereas the opposite is the case in the "Soccer" domain. A possible explanation for this behavior is that the "Minigolf" Domain is already tightly constrained by its Tactics model, thus not gaining much advantage of RRTs more elaborate selection scheme. The "Soccer" domain on the other hand, features a broader Tactics model which heavily relies on the agent exploring the workspace, which might give RRT a slight advantage in this domain.

In addition to analyzing our algorithm's performance and showing visual solutions, we are also interested in comparing the qualitative performance of our planning approach to traditional reactive control methods as they are currently used in many applications. To do so, we designed an experiment in the simulated soccer environment where the player agent and the soccer ball are placed at a randomly initialized location. The agent's goal is to deliver the soccer ball into the goal-box which is protected by two reactive defenders. We executed a traditional, deterministic version of Tactics and Skills on this agent over multiple randomly initialized trials. This deterministic control approach would follow a simple robot-soccer policy of dribbling the ball and shooting it into the corner of the goal-box with the widest opening. We compared the success rate (resulting in a goal) of the deterministic execution with trials generated with our BK-BGT approach, using different maximum search tree sizes. Table 2 shows the results.

Method	Tree Size	% Success
Deterministic Tactic	n/a	30%
BK-BGT	2500	40%
BK-BGT	5000	55%
BK-BGT	10000	60%

Table 2: Average success rate of a simulated "attacker vs. two defenders" scenario (20 randomly initialized trials).

We can see that even with relatively small search tree sizes, we obtain a greater average success rate than traditional methods. The reason for this outcome is that our physics-based planning approach has the power to find many intricate, physics-based solutions which the reactive version will never execute based on its fixed policy. This experiment highlights the potential advantage of a physics-based planning approach over purely reactive policies.

7. CONCLUSION AND FUTURE WORK

We presented the physics-based planning problem and its challenges. We introduced a non-deterministic version of *Tactics and Skills* as a model for infusing goal-driven, high level behaviors into a sampling-based motion planner, thus reducing its action space and making search feasible. We introduced two techniques for the search tree expansion, one based on RRT, and a new "Balanced Growth Tree" one. We experimentally demonstrated the effectiveness in challenging physics-based planning testbed domains. We further compared and reported the success of physics-based planning versus reactive deterministic control.

We can identify several directions for future work, which could be contributions on their own. Further improving the planning performance is one of our own main focus points. On the computational aspect, there are several promising approaches, such as performing the physics-computations on the Graphical Processsing Unit (GPU). Many modern physics-engines are now starting to offer GPU-based hardware-acceleration which has the potential to significantly speed up the forward-simulation component of the algorithm. Additionally, it might be interesting to parallelize the entire planning algorithm to be run on modern multi-core processors. One way to achieve this would be by designating different branches of the search tree to different processing cores.

On the algorithmic side, it would be interesting to look into the concept of finite horizon planning. In particular, if used in real robot domains, our planner would need to operate within a fixed amount of time and under uncertainty. Limiting the depth of the search tree in combination with frequent replanning is a promising approach to achieve this. Re-planning also brings along other additional challenges. A very interesting question is whether it is possible to reuse a previously generated plan to increase the speed of replanning and reduce the amount of oscillations between consecutive

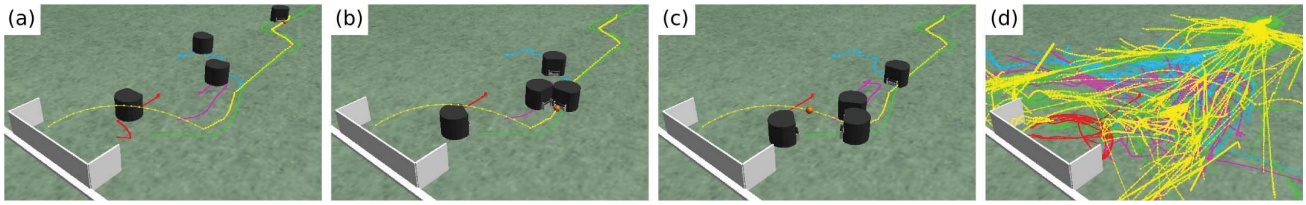

Figure 8: An example of the "Soccer" domain. (a) The initial configuration of the bodies. The controllable robot is initially at the top right, and the planning goal is to deliver the ball to the goal while avoiding the defenders and goalie robot bodies; (b), (c) Snapshots of one solution found by the planner; (d) The entire search tree representing the positions of all rigid bodies.

planning iterations. Existing planners, such as ERRT [7] or Multipartite RRTs [18],have made use of past results during replanning by caching previously used planning data. Whether this is possible for tactically constrained, physics-based planners, such as ours, is an open research question.

Finally, it might also be worthwhile to investigate the use of supervised machine learning to automatically "train" a particular tactical model by optimizing its internal transition probabilities and sampling distributions, thus creating the strongest and most efficient planner possible for a given domain. This could be achieved by executing the planner under many random domain instantiations, and recording which sampling choices frequently lead to a goal and which ones do not. Such an optimized planner is very likely to deliver significantly improved planning times because it will focus its search on tactical branches which are more likely to succeed.

Acknowledgments

This research was partly sponsored by Intelligent Automations, Inc. under subcontract no. 654-1, and by United States Department of the Interior under Grant No. NBCH-1040007. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of any sponsoring institution.

8. REFERENCES

[1] D. Baraff. Physically Based Modeling: Rigid Body Simulation. *ACM SIGGRAPH Course Notes*, 2001.

[2] S. Behnke and R. Rojas. A Hierarchy of Reactive Behaviors Handles Complexity. *Balancing Reactivity and Social Deliberation in Multi-Agent Systems: From Robocup to Real-World Applications*, 2001.

[3] B. Browning, J. Bruce, M. Bowling, and M. Veloso. STP: Skills, Tactics and Plays for Multi-Robot Control in Adversarial Environments. *IEEE Journal of Control and Systems Engineering*, 219:33–52, 2005.

[4] J. Bruce, M. Bowling, B. Browning, and M. Veloso. Multi-Robot Team Response to a Multi-Robot Opponent Team. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2003.

[5] J. Bruce and M. Veloso. Safe Multi-Robot Navigation within Dynamics Constraints. *Proceedings of the IEEE*, 2006.

[6] J. Bruce, S. Zickler, M. Licitra, and M. Veloso. CMDragons: Dynamic Passing and Strategy on a Champion Robot Soccer Team. In *Proceedings of IEEE International Conference on Robotics and Automation*, 2008.

[7] Bruce, J. and Veloso, M. Real-Time Randomized Path Planning for Robot Navigation. *Proceedings of IROS-2002*, October 2002.

[8] R. D'Andrea. The Cornell RoboCup Robot Soccer Team: 1999-2003. *New York, NY: Birkhauser Boston, Inc, 2005.*, pages 793–804, 2005.

[9] M. Lau and J. Kuffner. Behavior planning for character animation. *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 271–280, 2005.

[10] T. Laue and T. Röfer. A Behavior Architecture for Autonomous Mobile Robots Based on Potential Fields. *8th RoboCup Symposium*, 2004.

[11] S. LaValle. Rapidly-exploring random trees: A new tool for path planning. *Computer Science Dept, Iowa State University, Tech Report*, 1998.

[12] S. LaValle and J. Kuffner Jr. Randomized Kinodynamic Planning. *The International Journal of Robotics Research*, 20(5):378, 2001.

[13] N. Melchior, J. Kwak, and R. Simmons. Particle RRT for Path Planning in very rough terrain. In *NASA Science Technology Conference (NSTC)*, 2007.

[14] L. Parker. ALLIANCE: an architecture for fault tolerant multirobot cooperation. *IEEE Transactions on Robotics and Automation*, 14(2):220–240, 1998.

[15] N. Vahrenkamp, C. Scheurer, T. Asfour, J. Kuffner, and R. Dillmann. Adaptive motion planning for humanoid robots. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2127–2132, 2008.

[16] K. Yamane, J. Kuffner, and J. Hodgins. Synthesizing animations of human manipulation tasks. *ACM Transactions on Graphics (TOG)*, 23(3):532–539, 2004.

[17] S. Zickler and M. Veloso. Playing Creative Soccer: Randomized Behavioral Kinodynamic Planning of Robot Tactics. In *Proceedings of the RoboCup Symposium*, 2008.

[18] M. Zucker, J. Kuffner, and M. Branicky. Multipartite rrts for rapid replanning in dynamic environments. *Proc. IEEE Int. Conf. on Robotics and Automation*, pages 1603–1609, 2007.

This page is intentionally left blank.