Hole in One: Using Qualitative Reasoning for Solving Hard Physical Puzzle Problems

Xiaoyu Ge
Research School of Computer Science
Australian National University

Jae Hee Lee
QCIS, FEIT
University of Technology Sydney

Jochen Renz and **Peng Zhang**
Research School of Computer Science
Australian National University

Abstract

The capability of determining the right physical action to achieve a given task is essential for AI that interacts with the physical world. The great difficulty in developing this capability has two main causes: (1) the world is continuous and therefore the action space is infinite, (2) due to noisy perception, we do not know the exact physical properties of our environment and therefore cannot precisely simulate the consequences of a physical action.

In this paper we define a realistic physical action selection problem that has many features common to these kind of problems, the minigolf hole-in-one problem: given a two-dimensional minigolf-like obstacle course, a ball and a hole, determine a single shot that hits the ball into the hole. We assume gravity as well as noisy perception of the environment. We present a method that solves this problem similar to how humans are approaching these problems, by using qualitative reasoning and mental simulation, combined with sampling of actions in the real environment and adjusting the internal knowledge based on observing the actual outcome of sampled actions. We evaluate our method using difficult minigolf levels that require the ball to bounce at several objects in order to hit the hole and compare with existing methods.

1 Introduction

One of the grand visions of Artificial Intelligence is to build robots with similar everyday capabilities as humans, who can live among us and assist us with many of our daily tasks. There is a multitude of applications such as caring for the sick, the young or the elderly or household robots that can relief us from many of our daily chores.

In order to progress towards more capable and more human-like robots, we need to develop methods and technology that allow robots to successfully interact with their environment. It requires AI or robots to perceive their environment using their available sensors and to select and to perform physical actions or a sequence of physical actions that achieves a given task. Dealing with physical actions is very hard for a number of reasons:

1. Since the available information about the environment is based on perception, it will most likely be incomplete and imprecise
2. Since the world is continuous, there are typically infinitely many different actions available, each of which could have a different outcome. For example, the exact angle or force that is used to interact with another object determines its behavior.
3. The outcome of a physical action might be unknown before executing it or before accurately simulating it.

Accurately predicting the outcome of physical actions is essential for selecting the right action, but there are potentially infinitely many possible physical actions to consider.

When we humans are faced with a "*physical action selection problem*", i.e., a problem that requires selecting a physical action [1] that achieves the desired goal (out of an infinite number of possible actions), we are very good at coming up with a qualitative solution and with a qualitative prediction of the consequences of an action. A qualitative solution means that we can describe the physical action in words as well as what we expect will happen as a consequence of executing the physical action. Based on these predictions we can describe a physical action or action sequence that could potentially achieve the goal. Whether it does achieve the goal or not, we can only find out once we execute the action.

Physical action selection problems can come in many variants and it is not possible to formalize all of them as a single meaningful problem that covers all cases. We have therefore selected one particular physical action selection problem that is an actual real-world problem and that covers many common aspects of physical action selection problems. We call our problem the "*Hole-in-One*" problem in reference to the problem in mini golf of identifying and executing a shot that sinks the ball with this single shot: Given a static environment of physical objects C (the mini golf "obstacle course"), an object B (the "ball") at start location S, and a target location H (the "hole"), all of which we define more precisely in section 3. Identify the force vector P (the "putt") that, when applied to B at location S, results in B reaching the target H. The idea is that in order to achieve this, the ball needs to bounce at several objects that are part of C in order to reach H with only one shot. But which objects have to be hit and in which order needs to be determined.

[1] An action has several parameters e.g. the exact angle and force of a putt. Actions are different if their parameters have different values

(a) (b)

Figure 1: (a) A real-world mini golf course (b) Illustration of the problem domain in this paper. The goal region H is the green region. The trajectory of B given by an identified solution is shown as the sequence of red dots.

This is a major difference to papers in the literature who study physical action selection of robots, some of which even study minigolf (Zickler and Veloso 2009). Variants of the hole-in-one problem occur frequently, not just in mini golf, in Pachinko, in pool billiard, curling or in a multitude of physics-based video games such as Angry Birds, but also in many everyday situations. These variants can be in 2D or in 3D environments and involve gravity or not.

What these problems have in common is that there are infinitely many possible force vectors. Once a force vector is given and the physical setting, that is the physical properties of all objects and the environment is exactly known, it is possible to compute the exact trajectory of the ball and to see if that force vector solves the problem. However, the task we need to solve is the inverse problem and therefore much harder. We have to identify a force vector out of infinitely many possibilities that solves the problem. While a geometrical or analytical solution of these problems is typically not possible if the obstacle "course" is non-trivial, humans are very successful in solving these kind of problems. We also very much enjoy solving these problems, which is demonstrated by the fact that they often occur in a game-like setting. These problems become even harder to solve when we consider that we usually do not know the exact physical setting. We often only know what we can see and our perception is thus the limiting factor in what we know about the physical setting. Because of the uncertainty about the physical environment, every potential solution to the problem needs to be executed in the actual environment before we can be sure that it is a solution. If it is no solution, we need to find ways of adjusting it so that it will eventually lead to a solution.

In this paper we propose to solve this problem similar to how humans are believed to be doing it: by a combination of qualitative reasoning and mental simulation as well as through a repeated process of limited sampling in the actual environment, observation of the consequences and adjusting our mental simulation. The method can solve even very complicate instances of the hole-in-one problem, which we demonstrate in Section 5.

2 Background

There are two key research streams in reasoning about physical systems, namely qualitative physical reasoning (Davis 2008) and simulation-based reasoning (Battaglia *et al.* 2013). One goal of qualitative physics is to formalise the commonsense knowledge (Kuipers 1989) about real-world physics and solve physical reasoning problem within the framework. The main advantage of qualitative physical reasoning is that it can rapidly draw useful conclusions from incomplete information (Davis *et al.* 2013). However, these approaches are often specific to a narrow domain and there have been very few implementations of these theories. The most relevant work in this field is (Forbus 1981) which proposed a framework for reasoning about the motion of a 2D ball by qualitative simulation. The rules used for state transitions are based on qualitative process theory (Forbus 1984). While most of the previous work focuses on representing physical systems and describing (or predicting) consequences of actions, our method is solving a considerably harder problem as it has to find applicable actions from the infinite action space.

Simulation-based reasoning was inspired by findings in cognitive psychology that mental simulation may play a pivot role in intuitive physical reasoning (Craik 1967). Mental simulation is viewed as a probabilistic simulation in which inferences can be drawn over the principles of classical mechanics. The prediction power drops off drastically when the model is inaccurate or the observation is radically incomplete. (Davis and Marcus 2015) discussed the limitations of the simulation-based reasoning methods. (Smith *et al.* 2013) analysed how humans make physical predictions about the destination of a moving object in the simulated environment where a ball is moving on a bumper table.

In robotics, there has been extensive research on motion planning (Kumar and Chakravorty 2012) or manipulation planning (Dogar 2013) For example, (Westphal *et al.* 2011) uses qualitative reasoning for generating manipulation plans. It models the spatial layout of objects using a spatial constraint network. The plan is found when there is a consistent constraint network under the goal constraints. (Kunze and Beetz 2015) combines a qualitative reasoning method and physics simulations to envision possible effects of actions when making a pancake. The actions and plans of making a pancake seem hard-coded. Our problem is different from most manipulation or motion planning problems where robots can actively adjust their motion path while executing it. In our case, we need to identify a single impulse that solves the given task and no further adjustments are possible. Our task is not easier compared with those complex robotic tasks. For example, a recent paper (Wolter and Kirsch 2015) developed a framework aiming at combining learning and planing and employing qualitative reasoning and linear temporal logic. They used their framework to solve a "ball throwing" problem which is a simplified version of our problem. Their problem is to throw an object to hit a fixed goal location. Their approach does not plan the path, instead, it only looks at the final result (e.g., is the ball left or right of the goal) and adjusts the initial action accordingly. There has been some work on teaching robots to play mini golf. (Khansari-Zadeh

et al. 2012) proposed mathematical models for learning control parameters of hitting motions while they do not focus on solving the planning problem. (Zickler and Veloso 2009) proposed a framework for physics-based planning that integrates high level behavior planning and lower level motion planning. The method uses random-based sampling to find solutions. In the evaluation, we will show a comparison between our method and a random-based sampling method.

3 Modeling the Physical Environment

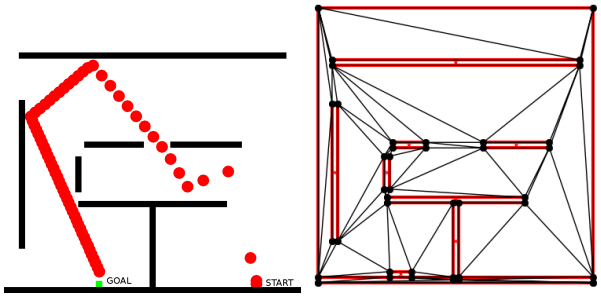
In this paper, we choose the following idealisation of the physical environment, which is often used in physics puzzle games such as mini golf:

- The environment is a restricted 2D plane.
- Objects are 2D rigid bodies with polygonal or circular shape.
- There may be a uniform downward gravitational force.
- Object movements and interactions obey Newtonian physics.
- Physics parameters of objects and the environment remain constant.

We call this environment PHYS2D. An instance of PHYS2D, or a *scenario* is a tuple $\langle E, \mathbf{O}, \mathbf{P}, \mathbf{T} \rangle$, where E is the restricted plane where the objects are located, $\mathbf{O}$ a finite set of static rigid objects O, each of which has a shape, a location, an angle and a type, $\mathbf{P}$ is a set of physics parameters that hold in the environment, such as gravity, and $\mathbf{T}$ is a set of object types and their respective physics properties such as mass and friction, or whether the object can move after being hit or remains static. We assume that all objects are initially static and stable under gravity.

Given such a scenario we can now apply physical actions to it and define physical reasoning problems and tasks. A *physical action* (short: *action*) can be applied to an object O by exerting an impulse at a certain position p of the exterior boundary of O (denoted as ∂O). An *impulse* is defined as a pair (θ, μ) where $\theta \in [0, 2\pi)$ is the direction and $\mu \in [0, \mu_{\text{max}}]$ the magnitude of the impact. μ_{max} is the maximal magnitude allowed in the environment, both θ and μ are continuous, therefore the number of possible actions is infinite. An action, i.e., a triple $\langle p, \theta, \mu \rangle$ applied to O can bring O into motion and, as an effect, can cause a chain of new actions on other objects. We call the initial action a *direct* action and the resulting new actions *indirect* actions.

While there are many possible problems that can be defined in this environment, the problem we want to solve is to identify a single action applied to a specified *start object*, such that the action results in a specified *goal region* to be hit by at least one of the objects. This problem is quite general in the sense that it can be applied to various physical games, such as computational pool (Archibald *et al.* 2010), Angry Birds (Renz 2015) and digital curling (Yamamoto *et al.* 2015). These games can have several objects that could be used as start objects or have several goal regions (e.g. holes in pool or pigs in Angry Birds) that the agent has to reach or destroy.In this paper we assume there is only *one* start object and *one* goal region. Despite this restriction, the problem



(a) (b)

Figure 2: (a) Scenario MINI7 (b) Triangulation of Mini7

is very challenging as there are many ways to use the chain of indirect actions on intermediate objects to reach the goal region after acting upon the start objects (see Example 2). Furthermore there are infinitely many possible actions, each of which might have a different outcome that needs to be determined in advance. We call this physical action selection problem the Hole-in-One problem.

Definition 1. (Hole-in-One)

Instance: An instance of the physical action selection problem Hole-in-One is a tuple $\langle E, \mathbf{O}, \mathbf{P}, \mathbf{T}, B, H \rangle$, where we use a scenario of PHYS2D and determine a ball $B \in \mathbf{O}$ as the start object and H as the target hole, a polygon in E with a given location.

Solution: A *solution* is an action or *putt* $P = \langle p, \theta, \mu \rangle \in E \times [0, 2\pi) \times [0, \mu_{\text{max}}]$ applied to object B such that B is delivered to the hole H as a consequence of the putt. To simplify the problem, we fix p to be the centroid of B. Formally, the problem can be described as finding a putt P such that the forward model f w.r.t. scenario $\langle E, \mathbf{O}, \mathbf{P}, \mathbf{T} \rangle$ induces a continuous function $R : [0, 1] \to E$, called the route R of ball B given P, such that $R(1) \in H$.

As the forward model f is not explicitly given and the search space is infinite, it is difficult to devise a systematic method for Hole-in-One and to analyze its complexity.

Example 2. In Figure 2a the start object B is the red object. The goal region H is the green region. Here, the solution is to make the start object bounce several times to reach the region. Since the action space to hit B is infinite, intelligent search is required to solve the problem.

What we have described is the actual physical environment we are dealing with. Solving physical action selection problems in this continuous environment is hard, but the problems we are facing are even harder as we do not have complete information about this environment. We only know what we can perceive and perception is typically noisy. The method we develop in this paper aims at solving these physical action selection problems under noisy perception, which requires an extended approach compared to having perfect information.

4 A Method for Identifying Physical Actions

The Hole-in-One problem distinguishes itself from common AI planning problems in that its search space is *infinite* and in particular the action space is *continuous*: Infinitely many

different actions can allow an object to take infinitely many different paths. We propose the following method to solve this problem:

- As input scenario, we take the information about the physical environment that we obtained through potentially noisy perception.
- We first partition the free space of the given scenario into finitely many triangular zones (Figure. 2b).
- We defined qualitative rules that describe the physics that govern the transition of moving objects between the triangular zones. We use these rules to generate sequences of qualitative transitions between zones that coincide with potential real paths a moving object can take to achieve the goal. We call such a sequence a *qualitative path*.
- Once all qualitative paths are determined, we rank them by their likelihood of being realized, before we try to realize them (see Section 4).
- We now use a physics simulator based on our perceived input scenario (that is the simulator does not know the real environment, denoted as Sim_I) to search for actions that realize the qualitative paths in their ranked order, i.e., actions that allow objects to follow the qualitative paths.
- The solutions we obtain here are not necessarily solutions in the real environment. Therefore, whenever we obtain a solution in Sim_I, we immediately apply the solution to the real environment. If it does not lead to a real solution, we adjust the object information in Sim_I according to the observation made in the real environment before we continue with the previous step. We will not adjust the triangulation or the qualitative paths when we adjust objects in Sim_I.

We now describe these points in more detail.

Qualitative Rules for State Transitions

Given a instance, we triangulate the free zone (i.e., the space not occupied with objects) of the scene using constrained Delaunay triangulation (Shewchuk 1996) where the object boundaries and the boundaries of E are part of the triangulation. We then generate *qualitative paths* based on the triangulation, that are sequences of qualitative states Q that represent how objects can move through the triangulation.

Definition 3. The *qualitative state* $\langle \triangle, e, \Theta, \mathtt{mt} \rangle$ of the ball B is defined by

- $\triangle$: the triangular zone where the ball is located;
- e: the edge of $\triangle$ the ball crossed to enter $\triangle$;
- Θ: possible direction range when entering $\triangle$ via e;
- $\mathtt{mt}$: the motion type of the ball. We distinguish three types of motions: FLY, BOUNCE, and SLIDE.

We obtain a qualitative path by expanding a qualitative state Q_i subsequently to a set of its next possible states $\mathbf{Q_i}$. Qualitative paths form a tree of states with branching factor at most 6, two outgoing edges with three possible motion types.

The procedure for expanding a qualitative state $Q_1 = \langle \triangle_1, e_1, \Theta_1, \mathtt{mt}_1 \rangle$ to a state $Q_2 = \langle \triangle_2, e_2, \Theta_2, \mathtt{mt}_2 \rangle \in \mathbf{Q_1}$ is as follows:

1. For each edge e_2 of triangle $\triangle_1$ with $e_2 \neq e_1$ we determine whether the current direction range Θ_1 allows the ball to move from e_1 to e_2.
2. If possible, we choose a motion type $\mathtt{mt}_2$ for the next state according to a set of physical rules (see below).
3. We also set the zone of he next state to $\triangle_2$ and set e_2 and determine Θ_2 according to our rules.

In the following, we describe the rules that govern state transitions between different motion types. Note that we always write "ball" to denote the moving object, as that is the moving object we use in the Hole-in-One problem. But the rules equally apply to other moving objects, not just to balls.

Rule 1 (FLY $\rightarrow$ FLY)**:** This rule is triggered when the current motion type is FLY and e_2 is an edge between $\triangle_1$ and $\triangle_2$, as the ball can keep flying after it entered $\triangle_2$ through e_2. Since the ball has entered $\triangle_1$ through e_1, the range of directions that the ball can fly from e_1 to e_2 is limited. Let v be the vertex of triangle $\triangle_1$ that is shared by e_1 and e_2, and let v_1 and v_2 be the remaining other vertices of e_1 and e_2, respectively. To compute Θ_2, we set first $\Theta_{\mathtt{Ran}}$ as the range between the direction from v_1 to v and the direction from v to v_2. Then we can ensure that only free flying point-like objects with direction $\theta \in \Theta_{\mathtt{Ran}}$ can fly from e_1 to e_2. To compute Θ_2 we also take the effect of gravity into consideration and apply gravity to the current direction Θ_1, obtaining a new range $\Theta_{\mathtt{Gra}}$. Then the method computes Θ_2 by intersecting $\Theta_{\mathtt{Gra}}$ with $\Theta_{\mathtt{Ran}}$. The next state $\langle \triangle_2, e_2, \Theta_2, \mathtt{FLY} \rangle$ will be created if $\Theta_2 = \Theta_{\mathtt{Gra}} \cap \Theta_{\mathtt{Ran}} \neq \emptyset$. •

The following two rules are triggered when $\mathtt{mt}_1 = \mathtt{FLY}$ and e_2 is a surface of an object. In this case the ball flying to e_2 can bounce. We assume it will be bounced back in the reflection direction range with respect to the outward normal of the surface. To this end, we determine Θ_2 in the same way as described in the (FLY $\rightarrow$ FLY) rule.

Rule 2 (FLY $\rightarrow$ BOUNCE $\rightarrow$ FLY)**:** This rule will always be applied when $\Theta_2 \neq \emptyset$. The method computes the range $\Theta_\uparrow$ of bouncing directions using Θ_2 and the normal vector of the surface. and adds $\langle \triangle_1, e_2, \Theta_\uparrow, \mathtt{FLY} \rangle$ to $\mathbf{Q_1}$. •

Rule 3 (FLY $\rightarrow$ BOUNCE $\rightarrow$ SLIDE)**:** In addition, there is the possibility that the ball *slides* on e_2 when the gravity force is towards the e_2, which assures that the ball can receive support from the surface, which we call the SLIDE condition. If the condition hold, we add $\langle \triangle_2, e_2, \Theta_2, \mathtt{SLIDE} \rangle$ to $\mathbf{Q_1}$. •

The following three rules are triggered when $\mathtt{mt}_1 = \mathtt{SLIDE}$. We assume that once the ball enters into the SLIDE motion, it will keep sliding until it hits a "wall" or until the surface cannot support it anymore. Let e_1 be the surface on which the ball is sliding, and e_2 be the surface connected to e_1 through their common vertex.

Rule 4 (SLIDE $\rightarrow$ SLIDE)**:** We add state $\langle \triangle_2, e_2, \Theta_2, \mathtt{SLIDE} \rangle$ to $\mathbf{Q_1}$, where Θ_2 is the direction from the common vertex to the other vertex of e_2. $\triangle_2$ is the triangular zone to which e_2 belongs. •

Rule 5 (SLIDE $\rightarrow$ BOUNCE $\rightarrow$ SLIDE)**:** This rule applies if e_2 forms a "wall" in front of the current direction. Specifically, when Θ_2 as defined in Rule 3 is between the surface e_1 and

the surface normal, then the ball will bounce back and slide on e_1 in the opposite direction. •

Rule 6 ($\texttt{SLIDE} \rightarrow \texttt{FLY}$)**:** If e_2 can neither give any support to the ball nor allow the bounce, the ball will start to fly from the end of e_1. Hence, we modify the current state by changing the motion type from SLIDE to FLY and apply the $\texttt{FLY} \rightarrow \texttt{FLY}$ rule to infer the next states. •

So far we have only considered changes to the state of a moving ball, but whenever a moving ball bounces off another object, the other object can start moving too (provided it is a movable object). While this is not allowed in the Hole-in-One problem where all objects remain static, we still add this possibility for the sake of generality. This is covered by the following rules.

Rule 7 (Hitting Movable Objects)**:** For each edge e that belongs to a movable object we generate a state $\langle \triangle, e, \Theta, \texttt{FLY} \rangle$ where $\triangle$ is the triangle in the free space that also has e as its edge. This triangle is uniquely determined, as edge e represents a surface of an object that is shared by only one triangle in the free space. Θ will be the same direction range as the direction range Θ_1 of the object that hit the movable object. Should the SLIDE condition (see Rule. 3) apply, we will also add a state with motion type SLIDE. We then continue expanding each of the generated qualitative states using the other applicable rules. •

Note that once an object starts moving, we would have to adjust the triangulation as the free space changes. However, in this paper we do not explicitly handle more complicated cases where moving objects repeatedly interact. Instead, we assume here that these cases are covered by the simulator when trying to realize paths and leave more detailed rules capturing this to future work should it be necessary.

Generating Qualitative Paths

To derive qualitative paths from the ball B to hole H, we starts with all possible initial states. An initial state $\langle \triangle_{\texttt{ini}}, e_{\texttt{ini}}, \Theta_{\texttt{ini}}, \texttt{mt}_{\texttt{ini}} \rangle$ of B is given as follows:

- $\triangle_{\texttt{ini}}$: the triangular zone containing the centroid of B;
- $e_{\texttt{ini}}$: the surface on which B is located;
- $\Theta_{\texttt{ini}}$: possible direction range to reach the next edge.
- $\texttt{mt}_{\texttt{ini}}$: there will be at most four different initial states as B can SLIDE on $e_{\texttt{ini}}$ in two different directions or FLY to each of the other edges of $\triangle_{\texttt{ini}}$.

We expand each initial state by successively applying all applicable rules to determine possible successor states. We stop expanding a state when it reaches the goal state $\langle \triangle_H, e_H, \Theta_H, \texttt{mt}_H \rangle$ where e_H is a surface of $\triangle_H$ that contains hole H. A qualitative path is a sequence of states from an initial state to a goal state where the successor state of each state is obtained by using our qualitative rules. We record which rule is applied at which state in order to rank qualitative paths.

Any qualitative path that does not lead to the goal state will be deleted. We use the following rules to ensure that,

Rule 8 (Do not move through a smaller edge)**:** Whenever an object O is required to pass through an edge e that is smaller than O itself, we remove any qualitative path that does not include a bounce transition at e. •

Rule 9 (Limit the number of states)**:** If a state is exactly the same as a previous state (including the same direction range Θ), we delete that qualitative path as it may generate infinite cycles. If a qualitative path reaches a preset maximum of states without reaching the goal state, we also delete it. •

Ranking Qualitative Paths

Before trying to physically realize the different qualitative paths, we will rank them according to the likelihood of leading to a solution. The idea is to take into account the magnitude of an impulse required to realize a qualitative path. If a qualitative path is too long or involves many bounces, which reduces the speed of the moving object, then the path will be less likely to be realized. Therefore, we assume that two factors play a role in ranking qualitative paths: the actual length of a path and the number of bounces along the path.

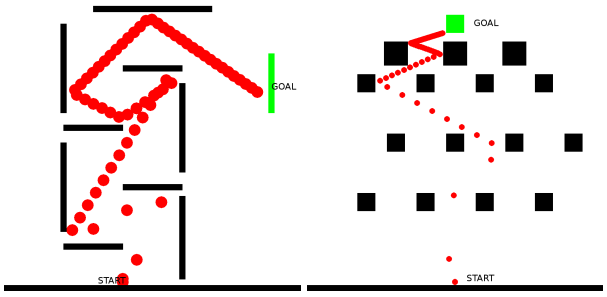
Let $be_0, be_1, \ldots be_n$ be the sequence of edges, where be_0 is the initial edge from which B is launched, $be_1, \ldots, be_{n-1}$ are all surfaces where bouncing takes place, and be_n is the entering edge of the goal state. Then the cost of a qualitative path is given by

$$\sum_{i=0}^{n-1} d(be_i, be_{i+1}) \cdot (1+\gamma)^i, \tag{1}$$

where $d(be_i, be_{i+1})$ is the Euclidean distance between two edges and $(1+\gamma)$ with $\gamma \in (0,1)$ is a penalty term. The penalty term penalizes the part of a qualitative path that happens after each bouncing. Therefore, given two paths of similar lengths, a path with less bounces will be preferred to a path with more bounces. γ can be set to a smaller value when the ball does not lose much kinetic energy after each bouncing, and vice versa.

Realizing Qualitative Paths

In this section we describe how to use these qualitative solutions to reduce the search space of finding a real solution. Recall that a solution is a physical action, i.e., an impulse $\texttt{imp} := \langle \theta, \mu \rangle$, on the centroid of ball B that delivers it to hole H. The idea is as follows: for each qualitative path, we sample possible actions using Sim_I within a range $\Theta_{\texttt{imp}}$ of directions that can potentially realize the path. We cluster qualitative paths that share similar direction ranges $\Theta_{\texttt{imp}}$ and sample only within these shared ranges. (We mainly focus on the direction parameter θ, as it has a larger effect on the solution. For the magnitude μ of the impulse, we always sample within its full range.) We also rank clusters by their average costs based on the cost function (1). If we do not find any action that can realize any qualitative path after going through all the clusters, we discard less promising clusters and restart sampling. A cluster is identified as less promising when none of the sampled actions has achieved the different bounces required to follow a qualitative path.



(a) (b)

Figure 3: Mini golf scenarios usually require several bounces. (a) Scenario MINI6 and (b) Scenario MINI2 with identified solutions

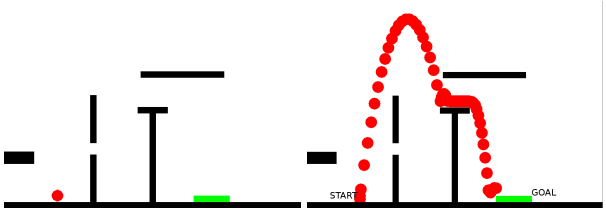
Testing Potential Solutions in the Real Environment

The visual input to the internal simulation can be noisy with imperfect perception. Therefore, applying an action in the real environment may have a different outcome from that predicted by Sim_I. To deal with this, we progressively adjust the internal simulation whenever a proposed action does not lead to a solution in the real environment. This is done to minimize the difference between the outcomes. The outcome of an action is represented by the sequence of triangular zones of the qualitative path generated by that action. This qualitative representation is less sensitive to visual noise compared with using trajectory points which are highly affected by noise. We use the Levenshtein distance (Levenshtein 1966) to quantify the similarities between two sequences. Once our method found an action that solves the problem in the internal simulation, it will execute the action in the real environment and observes the trajectory of the moving object. It then obtains the sequence of triangular zones from the observed trajectory and compares the sequence with its counterpart in the internal simulation. If the edit distance between them is greater than a threshold ϵ, the method will adjust the spatial configuration of relevant objects to minimize the distance. The relevant objects are the objects with which the moving object collided in the both internal simulation and real environment.

5 Evaluation

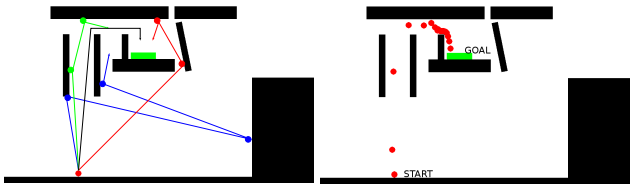
We simulated the real environment using the Box2D (www.box2d.org) physics engine. The method also uses Box2D for its internal simulator Sim_I with an incomplete knowledge of the real environment. We generated scenarios that allow us to evaluate different aspects of our proposed method. A scenario contains a set of movable and immovable objects. Objects have three physical properties, namely, density, friction, restitution. The goal region H is specified as rectangular region that is initially away from any movable object. An action is performed by exerting an impulse $(\mu, \theta), \mu \in I_\mu, \theta \in I_\theta$ on the centroid of B with $I_\mu = (0, 5000]$ and $I_\theta = [0, 2\pi)$. A problem is solved when H is contacted by B.

We first tested if our method can find different qualitative paths. In Figure. 4a, one possible solution is to let B hit the platform and fall to the green goal region. To realize this path, one has to know how an object files under gravity. To test the effectiveness of our clustering and ranking strategy, we designed some scenarios that have many qualitative paths that may lead to solve the problem. For example, in S2 (Figure 5a), our method detected 595 qualitative paths and divided them into 14 groups by their $\Theta_{\texttt{imp}}$. The figure illustrates four interesting groups of qualitative paths. Each colored arrow represents a group of qualitative paths, and shows the rough direction B takes. The four groups of paths were ranked in descending order of average costs as: black, green, red, blue. The blue group is ranked last because it takes several long distance bounces. The black group is ranked first because it suggests to hit the goal region via a direct trajectory. However, the black group was identified as an un-realizable group after a few sampled actions in the real environment. We further created several mini-golf scenarios. The scenario designs are inspired by the game levels of a popular video game of mini-golf[2]. Unlike S_1 and S_2, there is no gravity in these mini-golf scenarios. The scenarios usually require more than 5 bounces to solve (e.g. see Figure. 3).



(a) (b)

Figure 4: (a) Scenario S_1 (b) An identified solution to S_1



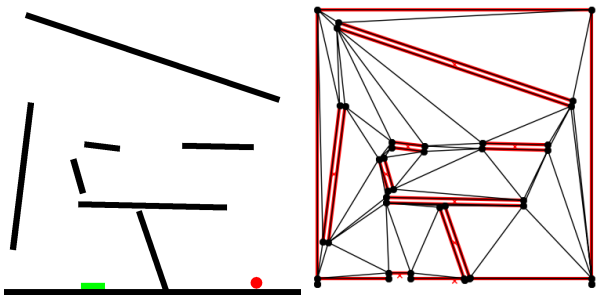
(a) (b)

Figure 5: (a) Groups of qualitative paths detected in S_2 (b) An identified solution to S_2

Dealing with Noisy Information To test the effectiveness of our method with imperfect perception, we perturb the visual input of a scenario by rotating each object at an angle θ in radians. The angle is sampled from a zero mean Gaussian with a variance $\sigma \in \{0.1, 0.2, 0.3\}$. The method uses the perturbed visual input for its internal simulation and keeps adjusting the angle of objects using the method described in Section 4. Figure 7 shows a perturbed mini-golf scenario in Figure 2a with $\sigma = 0.3$. It is clear to see that even a small rotation will substantially distort the scenario.

We use each designed scenario as a template to automat-

[2]http://www.eivaagames.com/games/mini-golf-game-3d/



(a) (b)

Figure 6: Perturbed scenario MINI7 with $\sigma = 0.3$ and its corresponding triangulation

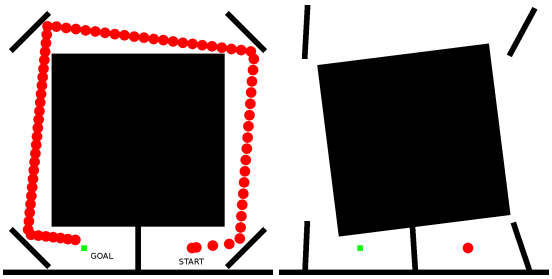
ically generate scenarios for testing. Given a template scenario, we randomly vary the spatial configuration of the objects. In the end, we obtained 72 levels. The proposed method uses penalty term $\gamma = 0.7$. At each round, the method samples 200 impulses with $\theta \in \Theta_{\texttt{imp}}, \mu \in [0, 5000]$ for each group of the identified qualitative paths.

We compare our method M_1 with a solver M_2 which uses a goal-oriented sampling strategy. The sampling strategy of M_2 is similar to the one used in (Wolter and Kirsch 2015) that adjusts actions according to the distance between the final position of the ball and the target destination. Specifically, M_2 evaluates an action using the minimum distance D between the trajectory of any movable object and the goal region. The goal is to find an action with $D = 0$, which solves the problem. M_2 obtained trajectory points directly from the actual environment (simulator), which are noise-free. Given a problem, M_2 performs several rounds of searching. At each round, M_2 makes several random trial actions in the actual environment and selects the action that yields the minimum D_1. It then performs a local search around the action, and picks the sample with the minimum $D_2 < D_1$ and repeat the procedure until it finds a solution or reaches a cut-off threshold. The evaluation result is summarized in Table 1. To give an indication of how much our method can reduce the sampling space, we show the proportion (AR) of the direction range of all qualitative paths to the entire range $[0, 2\pi)$. The actual solution space can be even smaller.

Summary of the Evaluation Our method outperforms M_2 in all the scenarios. M_2 is less efficient because there could be many local optima in a problem. By contrast, our method tries to realize each group of qualitative paths, which helps to avoid these local optima. Consequently, it can detect more different types of solutions (if there are any). It can still effectively find solutions when the solutions are far apart and potentially disconnected in the solution space. Qualitative reasoning and triangulation can be achieved efficiently; it takes on average 4 seconds to generate qualitative paths based on a triangulation with around 60 zones. The reason why no solution was found for all the ten MINI5T levels is that the solution space of MINI5 is very small (see Figure 7a); Varying positions of any object (especially the middle black square) will eliminate these solutions.

Table 1: Results on the generated scenarios with imperfect perception. SN: The number of actions made in the actual environment until the first solution is found. *MINI1T: the scenarios created based on MINI1. The average of AR and respective SN are shown for those entries.

Scenarios	AR	SN(M1)			SN(M2)
		$\sigma = 0.1$	$\sigma = 0.2$	$\sigma = 0.3$	
S1	0.05	421	621	829	4229
S2	0.05	375	837	837	>10000
MINI1	0.07	320	223	407	8529
MINI1T	0.05	371	332	319	426
MINI2	0.03	216	230	288	1736
MINI2T	0.04	32	23	36	280
MINI3	0.03	7	34	34	370
MINI3T	0.02	3	12	7	223
MINI4	0.02	133	509	967	987
MINI4T	0.04	42	93	172	254
MINI5	0.04	28	56	31	537
MINI5T	N/A	>10000	>10000	>10000	>10000
MINI6	0.04	68	199	208	2932
MINI6T	0.04	32	239	757	529
MINI7	0.05	41	77	236	3208
MINI7T	0.03	75	236	852	706



(a) (b)

Figure 7: (a) An identified solution to MINI5 (b) A randomly generated scenario based on MINI5 using $\sigma = 0.3$

As the noise level increases, our method can still detect and realize qualitative paths that lead to the goal. Such qualitative paths have similar bounce sequences as their counterparts derived from perfect triangulation. However, it takes on average longer to detect a solution than with accurate perception. Because as the noise increases, accuracy of the triangulation is getting worse and we generate more unrealisable qualitative paths. There will be triangular zones where there is supposed to be a surface while actually not or vice versa. These information can only be potentially adjusted after making several trial shots in the actual environment. Inaccurate perception also affects the ranking of qualitative paths.

6 Discussion

The Hole-in-One problem is just one example of a physical action selection problem, where a physical action has to be determined that achieves a given goal in a physical environment. The difficulty of these problems lies in the fact that the action space is infinite Another source of difficulty is that the sequence of interactions with other objects as well as the required number of interactions is unknown. The problem becomes even harder when the exact physical properties of

objects and their locations are not exactly known.

Despite having considered only one example problem, the method we developed to solve the Hole-in-One problem for 2D environments, under gravity and noisy perception is more general. Other physical action problems in 2D environments with different optimization criteria can be solved in a similar way, by triangulating the free space and by determining qualitative paths between the triangles that obey the general physics rules we defined. Our approach of sampling proposed solutions in the real environment and adjusting our internal knowledge through observations can be used for other physical action problems.

While some of the techniques we use are not new, clearly we have not invented triangulation and also sampling based adjustment has been done before, what is novel in our work is that we can solve arbitrary instances of a complex physical action selection problem without hardcoding or predefining actions or action sequences. We only define standard physics rules that determine what changes can happen to an object when it interacts with other objects or moves through empty space. We can do this under gravity and under noisy perception, and we do it in a similar way to how humans supposedly solve these kind of problems (Trial and error). One possible extension of our work is to consider 3D environments, where we could partition the free space into zones similar to how we do it in 2D. Another possible extension is to lift the restriction that all objects other than the ball are static.

7 Conclusion

We studied a realistic problem that contains some of the essential components AI needs to successfully interact with the real world: being able to predict the consequences of physical actions and to select a physical action out of an infinite number of actions that achieves a specified goal. This problem becomes even harder with noisy perception. The proposed method involves a combination of qualitative reasoning and internal simulation together with testing proposed actions in the real environment and, if necessary, adjusting our internal knowledge based on the new observations. While it is not our intention to build a robot that becomes the new minigolf world champion, we have seen in our experiments that our approach is able to identify some remarkable shots involving several bounces before achieving a hole in one. We are rather interested in coming closer to being able to solve physical action selection problems in general, particularly in noisy environments. A solution to this problem will have a major impact on AI and we believe that our approach forms a good starting point to achieving that. As a side note, we just read about golf playing robot LDRIC that managed to score a hole in one. But of course (still somehow surprisingly) a hole in one in golf seems to be easier to achieve than a hole in one in a difficult minigolf level that involves several physical interactions with other objects.

References

Christopher Archibald, Alon Altman, Michael Greenspan, and Yoav Shoham. Computational Pool: A New Challenge for Game Theory Pragmatics. *AI Magazine*, 31(4):33–41, December 2010.

Peter W Battaglia, Jessica B Hamrick, and Joshua B Tenenbaum. Simulation as an engine of physical scene understanding. *Proceedings of the National Academy of Sciences*, 110(45):18327–18332, 2013.

Kenneth James Williams Craik. *The nature of explanation*. CUP Archive, 1967.

Ernest Davis and Gary Marcus. The scope and limits of simulation in cognitive models. *arXiv preprint arXiv:1506.04956*, 2015.

Ernest Davis, Gary Marcus, and Angelica Chen. Reasoning from radically incomplete information: The case of containers. In *Proceedings of the Second Annual Conference on Advances in Cognitive Systems ACS*, volume 273, page 288, 2013.

Ernest Davis. Physical reasoning. *Handbook of Knowledge Representation*, page 597, 2008.

Mehmet R Dogar. *Physics-Based Manipulation Planning in Cluttered Human Environments*. PhD thesis, Massachusetts Institute of Technology, 2013.

Kenneth D. Forbus. A study of qualitative and geometric knowledge in reasoning about motion. Technical report, Cambridge, MA, USA, 1981.

Kenneth D Forbus. Qualitative process theory. *Artificial intelligence*, 24(1):85–168, 1984.

Seyed Mohammad Khansari-Zadeh, Klas Kronander, and Aude Billard. Learning to play minigolf: A dynamical system-based approach. *Advanced Robotics*, 26(17):1967–1993, 2012.

Benjamin Kuipers. Qualitative reasoning: modeling and simulation with incomplete knowledge. *Automatica*, 25(4):571–585, 1989.

Sandip Kumar and Suman Chakravorty. Adaptive sampling for generalized probabilistic roadmaps. *Journal of Control Theory and Applications*, 10(1):1–10, 2012.

Lars Kunze and Michael Beetz. Envisioning the qualitative effects of robot manipulation actions using simulation-based projections. *Artificial Intelligence*, 2015.

VI Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. In *Soviet Physics Doklady*, volume 10, pages 707–710, 1966.

Jochen Renz. AIBIRDS: The Angry Birds Artificial Intelligence Competition. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, March 2015.

Jonathan Richard Shewchuk. Triangle: Engineering a 2d quality mesh generator and delaunay triangulator. In *Applied computational geometry towards geometric engineering*, pages 203–222. Springer, 1996.

K Smith, Eyal Dechter, J Tenenbaum, and Edward Vul. Physical predictions over time. In *Proceedings of the 35th annual meeting of the cognitive science society*, pages 1–6, 2013.

Matthias Westphal, Christian Dornhege, Stefan Wölfl, Marc Gissler, and Bernhard Nebel. Guiding the generation of manipulation plans by qualitative spatial reasoning. *Spatial Cognition & Computation*, 11(1):75–102, 2011.

Diedrich Wolter and Alexandra Kirsch. Leveraging qualitative reasoning to learning manipulation tasks. *Robotics*, 4(3):253–283, 2015.

M. Yamamoto, S. Kato, and H. Iizuka. Digital curling strategy based on game tree search. In *2015 IEEE Conference on Computational Intelligence and Games (CIG)*, pages 474–480, August 2015.

Stefan Zickler and Manuela Veloso. Efficient physics-based planning: sampling search via non-deterministic tactics and skills. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pages 27–33. International Foundation for Autonomous Agents and Multiagent Systems, 2009.