Learned Shape Affordances through Simple Tool Use

Jeremy Bennett
Poorvi Joebert
PJ Campbell

Table of Contents

Table of Figures

Introduction

A hallmark of greater intelligence is the ability to use objects in ones environment to perform actions that normally would not be possible. These tools allow an individual, human or animal, to extend their capabilities whether they are physical or mental. However, it is essential to know about the affordances of these objects and tools in order to utilize them effectively.

An expert pool player knows how far and at which angle a spherical object moves when pushed with enough force concentrated on a certain point and in a particular direction. The groundwork for this is laid as a child. At a very young age, children learn such affordances of shapes by moving and orienting objects while at play.
This project proposes an approach where an autonomous robot would learn and apply the same principles. It extends Stoytchev's 'Behavior Grounded Representation of Tools Affordances' which concentrated on learning affordances of varied tools when applied on a single object. In our project, we invert this to learn the affordances of differently shaped objects using a simple hooked stick as the primary tool. In particular, we are concerned with the movement and orientation of said objects for a 'PUSH' action.

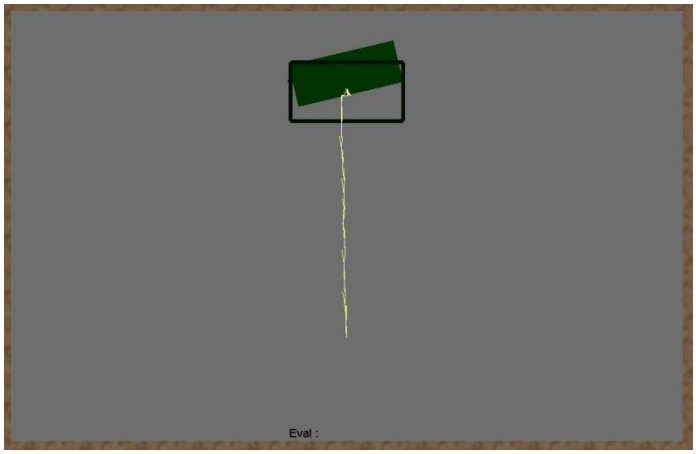

Figure 1: Shafttu Simulation

Motivation

One of the far reaching goals of robotics is to create robots that can do jobs which are repetitive and often tedious, or jobs that demand precision or simply those jobs which are harmful to humans; thereby freeing human beings to concentrate on finding the answer to life, the universe and everything, one assumes. Almost any work though, requires either tool use or object interaction and sometimes object manipulation through tool use.

Current robotic systems are programmed to work efficiently, but almost all are quite task specific and are incapable of being extended to work on other tasks without reprogramming. For example, a robot that utilizes tools will have to be programmed for a specific tool and a predefined set of objects that it can work on. If one were to generalize the task in any meaningful way, the amount of tools, objects and their interactions that need to be encoded, are

quite staggering. If this robot were to encounter a tool or object not in the database, it will be unable to perform any action.

Secondly, no matter how well programmed robots are today, they do not approach the levels of human intelligence. A robot may win at chess, but that is a testament to its data processing, memory and efficient search and retrieval routines than actual IQ. Similarly, while a robot might classify a saw as a tool, with property as cut logs; it has no understanding of what a saw, tool or log really is.

Thirdly, knowing the affordance of an object or tool is essential for manipulating the object for various tasks. Gibson defined affordance as “all ‘action possibilities’ latent in the environment, objectively measurable and independent of the individual's ability to recognize them, but always in relation to the actor and therefore dependent on their capabilities.” So, a desert would afford traversability to a mars rover but not to a roomba. It is therefore very important that a robot learn about affordances rather than having it programmed.

Developmental learning targets these prevalent issues in robotics and aims to create machines with robust intelligence that are autonomous and adaptable to novel situations. This can be accomplished by allowing the robot to undergo a developmental learning phase which helps it understand and verify things by itself without need of human interference.

Our motivation was to develop a generalized method whereby a robot could, through developmental stages, learn about object affordance independently. Previous work on autonomous tool use by Alex Stoytchev, was focused on learning the affordances of tools by using various tools on a single object. The setup is similar but our motivation is different. We are concerned with learning affordances of shapes and hence use a single tool to operate on differently shaped objects. Because of the complexity involved, we have chosen to start with regular shapes for this project.

Audience

The target audience for this project is researchers in the fields of Developmental Robotics and is relevant to the automation industry. It would also interest robot developers for games like snooker and golf.

Demonstrated Need

Tool use has not been well addressed in robotics [6]. It is usually hard-coded into the robotic system; for example, on the assembly line in a car factory, specialized robots are built to use tools for a specific task in a particular production phase. However, these robots cannot perform any other tasks and their tools are not meant for other phases. In most cases, they are physically bound to a single tool and are not built to use other types of tools. In the future, robots may be implemented to replace human activity in delicate areas such as elderly care and the military. Thus, robots will need to be capable of tool use [6].

Previous/Related Work

Affordances

Affordance is a biological concept that defines a relationship between an intelligent being and its environment [5]. Chemero and Turvey state that "affordances are opportunities for behavior" and "they are properties of the environment but taken relative to the animal" [1]. Ortmann and Kuhn believe "humans judge their environment according to the actions it affords." They claim that affordances are qualities. For instance, an affordance of stairs is „climbability". Climbability is an observable quality and physical quality of a step [6]. Affordances are tremendously powerful because they provide a perception of object properties and their role in the environment via actions executed on the objects [5]. An intelligent being can learn the affordances of an object though motor and sensing capabilities such as pushing, pulling, moving, and grasping. Affordances can indicate what will take place if a specific action is applied to an object. Thus they can help an intelligent being to decide which objects, and what actions on said objects, will accomplish a desired goal [5]. For instance, a bird – after learning the affordances of long and thin objects in the environment – may decide to use a twig to find food in openings that its beak cannot fit.

According to Don Norman, affordances are the actual properties of an object that can be perceived [12]. Although most objects often have many different attributes, only those that give a clue to how they are to be used are actually affordances. For instance, a table is usually a flat surface that one can sit comfortably in front of in order to do things such as eat breakfast, study or play a card game. Therefore, if an object exhibits these attributes regardless of whether or not it is traditionally labeled a table, the user will perceive these table affordances and the object will be implicitly considered a table. Sometimes, objects share affordances. One can sit on a table much like one can sit in a chair. Most objects have common affordances that are perceived by most human beings. For example, Norman points out that knobs afford turning, slots afford inserting things, plates are for pushing and balls afford bouncing and throwing [12]. These affordances give clues to how to interact with such objects. A thick circular disk affords sliding and gives the clue that it can be used as a hockey puck. A rock is a solid, heavy object and affords throwing which may give the clue that it can be used for holding a sheet of paper onto the ground or breaking a glass window. A stick affords pushing, poking and reaching which gives the clue that it can be used to push other objects and reach objects that would otherwise be unreachable.

Norman infers that for things that do not display cues as to how they should be used, a conceptual model must be learned [12]. He claims that a decent conceptual model will help determine results of particular actions on a specific object. He also points out that without such a model, all interactions with an object that is complex or questionable are arbitrary and meaningless. Furthermore, the outcomes of behaviors on this object cannot be predicted and there is no knowledge of how to apply this object in novel situations. Conceptual models should be relatively simple. Norman states that the “underlying physics or chemistry” is irrelevant. Rather one must know “the relationship between the controls and outcomes” [12]. Also important to note here, is that if the model is constructed conveyed incorrectly then it can pose complications and confusion.

Conceptual models are formed when intelligent beings interact with objects. Humans have the natural tendency to explore new objects and to learn a novel use for them, a behavior that is noticeable in young infants. Children impose many different exploratory behaviors on objects to

gain knowledge or improve existing knowledge about them. Such exploratory behaviors include touching, tasting, pushing, pulling, lifting, scratching, throwing and stacking. According to Norman, this is necessary for "helping us understand our experiences, predict the outcomes of our actions, and handle unexpected occurrences [12]". Often times models are developed from "fragmentary evidence" with little to no understanding of what is actually taking place and "with a kind of naïve psychology that postulates causes, mechanisms, and relationships, even when there are none" [12].

J. J. Gibson defines an affordance as "what it offers an animal" or "what it provides and furnishes, either for good or evil. He describes affordances as "complementarity" between animal and its surroundings. Gibson points out that affordances are unique and measure relative to an animal. For example, a high chair or a booster seat affords sitting for an infant or a child. However, it doesn't afford sitting for an adult since it would be unpleasant for an eighteen-year-old person to attempt sitting in it. A stick that measures about one yard in length affords pushing and reaching for a human being but it doesn't for a bird. A bird would not be able to manipulate a stick that long or thick since it doesn't have arms or cannot grasp large objects with its beak. So affordances are ultimately relative to the size of the person or living thing and its capacity to do certain behaviors.

Gibson describes two different types of objects: attached and detached objects. Some examples of detached objects are sticks, rocks, pens, forks and spoons. Attached objects are things that cannot be used without first being removed or broken off of something else. He claims that whether or not a detached object offers affordances depends on the size of the living thing that uses it. Living things with hands or opposable thumbs get the most out of detachable objects that offer affordances and the number of affordances that detachable objects have are increased. For example, a stick offers more affordances to a human being than to a bird even though there exists a decently sized stick that both organisms can use. This is because objects can be manipulated to form novel objects and/or affordances [3].

Although there are many different ways to classify or categorize different objects into groups, identifying affordances is not to classify them [3]. For example, Gibson states that since a stone is a projectile this does not mean that it cannot be anything else like "a paperweight, a bookend, or a hammer" [3]. Being a projectile or a paperweight or a hammer demonstrates three different classifications however all of these objects offer the same affordance. The fact that they are called different names has no bearing on the way they are perceived [3].

Tool Use

With the ability to perceive object affordances comes the ability to use objects as tools. Tool use is common amongst animals and is an indication of intelligence. A single tool can afford a variety of behaviors, especially if the animal can modify it [3]. Animals that can manipulate an object in its environment to accomplish a goal exhibit reasoning abilities [11]. For example, in a study presented by von Bayern et al, a crow was able to solve a task that required it to drop a stone into a vertical tube to release food from an out-of-reach platform inside a transparent box. Learning to use tools is a developmental process. E. J. Gibson stated that "active exploration of objects has important results for learning about what an object affords, what can be done with it, its functional possibilities and uses" [2] She pointed out that when a child perceives an object‟s use it is an indication that it recognizes potential affordances. Thus, in order for an object to be recognized as a tool, it needs to undergo exploratory behaviors by the manipulator [9].

Applications to Robotics

Several researchers have examined the applications of affordances in robotic systems. Chemora and Turvey compared Gibsonian and representationalist approaches to affordances. According to the Gibsonian view, the two components of affordances are the animal-environment system and the perception-action [1]. The representationalist approach believe that affordances must be internalized (this approach eliminates the perception-action component) and those who take the Gibsonian approach believe that affordances must be perceived. Chemora and Turvey claim that the Gibsonian approach is more suitable for developmental robotics.

Stoytchev developed a method for the robot to develop representations of the tools and their affordances using a behavior-based approach [8]. He equipped the robot with a set of exploratory behaviors which it could randomly apply to the tools to manipulate real physical objects. The robot then recorded and kept track of the outcomes of each tool and its behavior on an object. Stoytchev pointed out that his method has some shortcomings stating that in this approach some tool affordances are less likely to be discovered due to limited exploratory behaviors.

Montesano et al implemented a developmental approach for a general affordance model that could be applicable to new contexts. They proposed that affordances naturally connect low-level representations i.e. sensory-motor maps and higher level cognitive skills i.e. imitation [5]. Their approach involved a framework with three phases which include sensory-motor coordination, world interaction, and imitation. In the sensory-motor phase the robot learns new actions and builds perceptual skills. The world interaction phase helps the robot to learn a Bayesian network that symbolizes affordances via statistical dependencies between features, actions, and effects. In the imitation phase, the robot used its affordance model to recognize objects and actions performed by a human and imitate the goal of the human participant. The authors‟ developmental architecture is illustrated below.

Kraft et al implemented a robotic vision system to autonomously learn object representation and their grasp affordances through three-dimensional images and pure exploration without any prior knowledge of objects [6]. Using basic exploratory behaviors, the robot gains a visual representation of the objects appearance and geometric properties while learning various grasping actions. Successful grasping parameters determine the objects‟ grasping affordance, which are associated with the object model.

Sinapov and Stoytchev‟s research provided a different perspective of the way object properties are learned [7]. Specifically he showed why a combination of exploratory behaviors and sensory modalities can effectively increase object recognition rates. He developed a theoretical framework that utilizes a machine language concept called classifier diversity, in which a combination of multiple classifiers is more efficient in improving accuracy than a single classifier. He states that exploratory behaviors on an object are essentially like classifiers and therefore can be boosted to increase recognition rates [7]. In our study, only one exploratory behavior is necessary.

Uger et al implemented a model to identify the traversability affordance, a property of an object that allows it to be moved or rolled away, such as a sphere or a lying cylinder [10]. In their research, they found that only one percent of information about an object‟s features was needed to determine if it was "traversable". Thus they were able to utilize a range image of the environment and from that construct a perceptual representation of each object in the environment [10]. Using both a computerized physics-based simulation called the MACSim and

a real mobile robot – coupled with very little information of the objects‟ features via range images – they were able to identify the traversability affordance with 95% accuracy.

Approach

We will extend Stoytchev‟s behavior-based approach for tool use to manipulate differently shaped pucks with a hooked stick. We will impose a babbling period on the robotic simulator to learn the properties of the hooked stick and the effects of exploratory behaviors on each of the differently shaped objects or *pucks*. In addition, it will also learn the affordances of each shape. We will test the simulator‟s knowledge of affordances to measure its ability to maneuver a puck so that it is not only positioned but also oriented correctly over a marker representing its shape. This study will be conducted using a physics-based simulation of the robotic arm similar to the mobile manipulator in Stoytchev‟s experiments.

Equipment

Libraries

OpenGL

OpenGL is a standardized graphic application programming interface that was first introduced in 1992 in order to allow portable graphic applications. Through the years it has gone through numerous revisions and has earned its place as the mostly widely used graphics API accelerating the rendering of thousands of application across a wide variety of platforms It is known for its stability, reliability, scalability, and most importantly ease of use.
http://www.opengl.org/

Bullet

Bullet is a professional grade Open Source physics library that is supported on a variety of platform including the current generation of game consoles providing Collision Detection, Rigid Body Dynamics, and soft body dynamics. It has solid reputation for pushing the state of art and has even found its way into several movies and AAA game titles.
http://bulletphysics.org/wordpress/

OpenCV

Open Source Computer Vision Library (OpenCV) is programing library aimed at real time computer vision that was initially developed by Intel in 1999 to advance CPU intensive applications and later picked up by Willow Garage. It has gone through several iterations with the most recent adding C++ wrappers so as improve its ease of use by minimizing the lines of code required to execute functionality and to eliminate common coding mistakes such as leaking memory.
http://opencv.willowgarage.com/wiki/

Microsoft Visual Studios 2010

Visual Studios is a premier integrated development environment by Microsoft that enables one to design, implement, and test sophisticated programs for running on various platforms. The single interface design eliminates a lot of issues often encountered in software development and therefore is one of the preferred solutions in industry.
http://www.microsoft.com/visualstudio/en-us/

Google Code
Google Code is a web based repository and project tracking tool that is available to the Open Source Community.
http://code.google.com/

Virtual Objects
Table
The table is the flat surface on which all pucks are manipulated. It is represented by a flat plane that occupies the valid region of operation.

Hooked Stick
The manipulator in our robotic system is a simple hooked stick as seen below. A hooked stick was chosen over a straight stick in order to allow the system behaviors to include push operations that are towards itself as well as away. In order to ensure that the stick cannot press down on an object as it is being pulled the hooked section will be significantly taller than the pucks so that even if the end of the stick was laid on the table the shaft would still not touch the puck.

Figure 2: Stick

Pucks
There will be support a set of predefined pucks consisting of a circle, square, rectangle, and triangle.

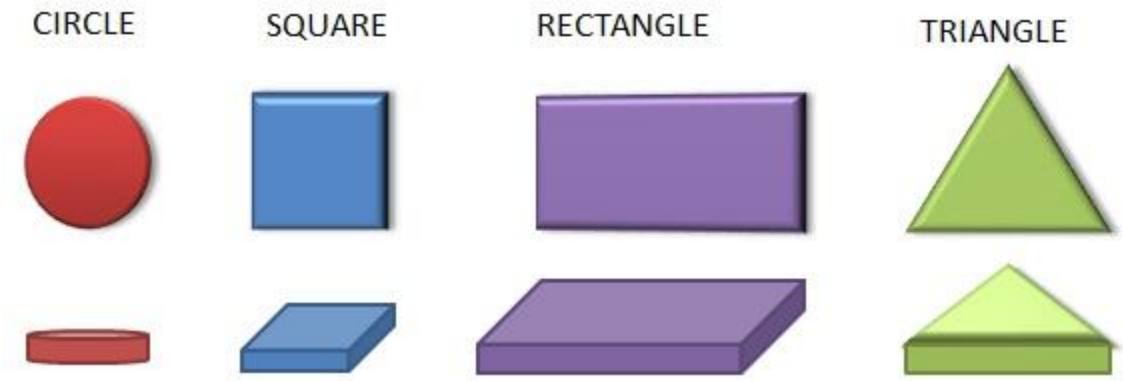

Figure 3: Predefined Shape Pucks

Each puck will have a yellow marker that can be used to identify its current orientation and will be push able along any of its edges. Note that while an object may be pushed along any edge the directions of push are limited to the four cardinal directions.

Figure 4: Predefined shapes pucks with orientation marker

Algorithms and Data Structures

Algorithms

Two main algorithms will be implemented: Babbling and Strategy.

Babbling

The babbling algorithm is responsible for determining an object's affordances in relation to the tool by applying a random set of behaviors. The combined affordances establish a model that allows desired results to be achieved. In the case of project the babbling algorithm will randomly select a push position, direction, and distance to be applied to the current puck and stores the results. Since the strategy chooses the best action based upon the desired results it is not necessary to ensure that the full set of permutations is executed during babbling, however the babbling algorithm will need to ensure that selections are applied to every orientation and that the puck is reset back to the start position if it should go out of bounds.

Babbling Implementation:

1. Set initial Babbling position and orientation for the chosen puck.
2. Apply a randomly generated push action on the puck.
3. Note the resulting puck position and orientation
4. If the puck is within the boundary then add the result to the database and update times used for the action, else reset with same orientation.
5. If the pucks moves or changes orientation update times successful for the action.
6. Steps 2 -6 are repeated for this new puck position.
7. The iteration occurs for a set number of times after which initial babbling position is incremented and the process repeats.

To ensure valid results for each orientation of the puck, initial babble positions were incremented 360 times, each initial orientation was then 'babbled' upon up to 20 iterations. But there were indications that the data set gathered by this was small and sometimes no equivalent action was returned during the evaluation phase. In order to gather larger data sets, the orientation each orientation was ensured to be executed 4 times resulting 1440 resets and the number of iterations before reset were increased to 50.

Further to optimize the amount of information gained from each babble action, successful actions were added for all equivalent orientation push direction pairs after applying the necessary translation.

Strategy

In the evaluation phase, the aim is to maneuver the object from its initial placement in order to position & orient it such that its contours match that of its shape marker. A greedy algorithm is

used as the Strategy to guide this exploratory behavior. This is based on previous work on "Affordances of Tools" by Alex Stoytchev.

Given the initial test setup, the robot first identifies the positions of the puck and the shape marker and calculates the distance between them. It then filters all exploratory behavior results for the particular shape from the database. All those with less than a predefined success rate are eliminated. Of the top few behaviors from this list, the one whose expected action brings it closest to the desired destination is chosen and applied. Beyond this, our strategy will rule out any behavior which moves the puck beyond the defined border.

If the actual outcome matches the expected outcome of the particular behavior, then its success rate is increased. It now recalculates the distance between the puck and the shape marker. If they are within a certain error tolerance, then the goal is reached, else, the above steps are repeated.

Strategy Implementation:

1. Calculate the distance, angle, and change in orientation from the shape to the marker.
2. If the distance is greater than 1
    a. Choose the action which results in the shapes moving closest to the marker. Actions that have less than a 50% chance of success are ignored.
3. If is the distance is less than 1
    a. Choose the action that will result in the greatest percent change towards ideal for the orientation or distance while minimizing the change to the other parameter. Again actions that have less than a 50% change of success are ignored.
4. Apply action resulting action to shape and update the results. Errant actions that produce no results will have their success rate reduce resulting in them being ignored on subsequent action selections.

Data Structures

Two main data structure will be implemented: Object Action and Object Result.

Object Action

The object action data structure will be used to capture all relevant state that represents a discrete event. This includes the initial orientation of the object and the location, direction, and distance of the push.

```
struct obj_act
{
   unsigned int iOrient; // Orientation
   unsigned int iPos;    // Position of push
   unsigned int iDir;    // Direction of push
   unsigned int iDis;    // Distance of push
   unsigned int iNumAct; // Number of times action executed
   unsigned int iSucAct; // Number of time successful
};
```

Object Result

The object result data structure will be used to capture all relevant state that represents a potential result of a given action. This includes the resulting motion of the object defined in polar coordinates and the degrees of rotation.

```
struct obj_res
{
   float    fMotDis; // Linear distance of motion
   float    fMotAng; // Angle of motion
   float    fOriAng; // Change in orientation
   obj_act *pAct;    // Associated action
};
```

Data Definitions

Orientation

Rotation of the puck in degrees based upon the change in location of its orientation marker.

Push Position

Angle in degrees relative to the orientation marker at which the hooked stick is placed against the object in order to perform a push behavior.

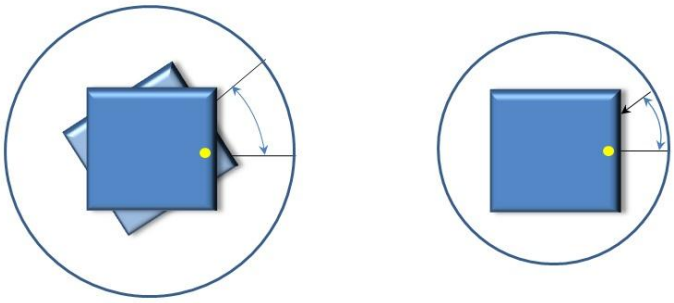

Figure 5: Shape Orientation (left) and Push Position (right)

Push Direction

Id used identifying one of the four unique directions in which the stick can push the puck.

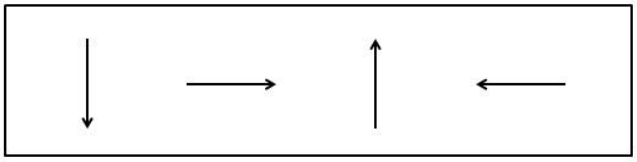

Figure 6: Push Directions

Push Distance

Push distance is the number of units the stick will be offset in the direction of push. In order to limit the number of permutation that will need to be tracked pushing will be limited to 3 and 5 units.

Implementation Methodology

This project will follow a clearly defined development model with each individual being responsible for different aspects of the projects.

Development Model

The project was implemented using a standard waterfall developmental model with each iteration having a clearly defined set of goals to be achieved. This model allowed issues with the general approach to be quickly identified and also provided a means by with results could be obtained at early stages while leaving open the door to stop earlier iteration if major problems should be considered. The project proposed 4 iterations, however only 2 were actually executed due to extreme difficulties with getting the physics engine to work correctly.

Iteration 1: Environment Setup

- Setup the Visual Studio Project and select Bullet and OpenCV versions
- Create simple application that renders top down view of a table like environment
- Implement objects for the various pucks that are to be supported
- Implement object for the hooked stick and create interface for it to be positioned and for it to execute a push
- Implement physics model such that the stick can interaction with the objects
- Implement marker object, ensure it does not participate in the physics

Iteration 2: 2D Perspective Simulation using Bullet for physics and data collection

- Implement algorithm for randomly selecting push direction and push points
- Implement data structure for storing results of a given push operation and update it based upon the simulation physic parameters before and after a push
- Implement babbling algorithm that randomly pushed a puck around the table
- Implement test algorithm that selects from the known list of affordances in order to maneuver the puck such that it ends up in the correct position and orientation
- Implement the defined test cases such that the setup can toggle between them and babbling against each of the object
- Execute babbling for each of the pucks
- Execute test cases for each of the pucks

Iteration 3: 2D Perspective Simulation using Bullet for physics and OpenCV for data collection

- Implement FBO interface that provides OpenCV the before and after frames for each push operation.
- Implement OpenCV based functions for determining the change in position and orientation
- Implement alternative update for the results data structure that uses the OpenCV results
- Execute babbling for each of the pucks

- Execute test cases for each of the pucks

Iteration 4: 3D Perspective Simulation using Bullet for physics and OpenCV for data collection

- Optional iteration to move simulation to a 3D perspective if time should allow.

Responsibilities

In order to facilitate development of the project each individual was given key responsibilities. These responsibilities were assigned such as to try and take advantage the talents of each of the individuals:

Jeremy: Framework, Open GL Rendering, Common Data Structures, and Simulation Strategy
PJ: Simulation, Physics, and OpenGL Rendering
Poorvi: Babbling, Evaluation, Website, and Poster

Timeline

The project was implemented over a period of a month and a half. Each of the non-optional iterations were initially given two weeks to complete, however due to difficulty it was necessary to extend each of the first two iterations by 1 week, thereby eliminating the time allocated for the third iteration.

Table 1: Timeline

Week of March 6 Implement Phase 1 *March 10 - Project Proposal Due **Week of March 13** Spring Break - No official work planned **Week of March 20** Finish Phase 1 **Week of March 27** Start Phase 2	**Week of April 3** Continue Phase 2 **Week of April 10** Finish Phase 2 Start Project Report **Week of April 17** Finish Project Report *April 21 - Project Report Due

Risks

Three key risks were identified for this project: Bullet, OpenCV, and Time.

Bullet
The project team had limited experience using this open source physics engine prior to this project. From this limited experience it was known the engine could be quite unforgiving,

however we felt it was the best choice and still stand by this decision. Ensuring the physics worked correctly for each the of desired shapes proved to be rather difficult. In order to try and limit the movement of the pucks the angular factor was initially set to limit them such that they could only rotated around the z axis. While theoretically this should have given the desired behavior in reality it resulted in the puck rotating in a very inconsistent manner. Allowing the pucks to rotate around all axes fixed this issue for the standard shapes; however it meant additional logic had to be put in place to guard against the puck accidentally being flipped over amongst other things.

OpenCV
The project team has limited experience with OpenCV and it has been over 4 years since the last time any of us has used it. However, the functionality we would have required was implemented in part while taking Computational Perception and should be easily adaptable to the needs of this project. Due to earlier difficulties OpenCV was never integrated into the project so this risk was never encountered.

Time
The timeline for completing the project was very tight and the goals were definitely aggressive. The iterative development plan was chosen since it allowed for the project to stop at a earlier milestone if necessary which turned out to be the case. Getting the project up and running and ensuring the physics worked correctly turned out to be a much more difficult problem than originally expected.

Evaluation Methodology

Objective

The aim is to get the robot to move the shape from its initial position to the marker which represents the particular shape. From the babbling phase prior to this, the robot would have formed some internal representation of the affordances of certain shapes. We have a preprogrammed strategy which guides the robot to move the shape closer to the destination by making use of the robot's representations on affordance to formulate the best way to position and orient the shape correctly over the marker.

Tests

There are four basic shapes which were used during the babbling phase. Each of these shapes will be tested based on placement -position relative to the marker-orientation at that position. This totals to 33 unique test cases.

CIRCLE
Owing to its shape, the circle has only one orientation. The shape is placed in three standard positions relative to the marker on the table - the bottom left, the bottom right and bottom center. Therefore we have only 3 test cases set up. In all the cases, the shape marker remains fixed in the top center position.

SQUARE

The square puck will be tested for two orientations. In one, the base of the square is set parallel to the table. In the second case, the base of the square makes a 45 degree angle with the table to form a diamond shape. The shape is placed in three standard positions relative to the marker on the table - the bottom left, the bottom right and bottom center. Therefore we have 6 test cases set up. In all the cases, the shape marker remains fixed in the top center position.

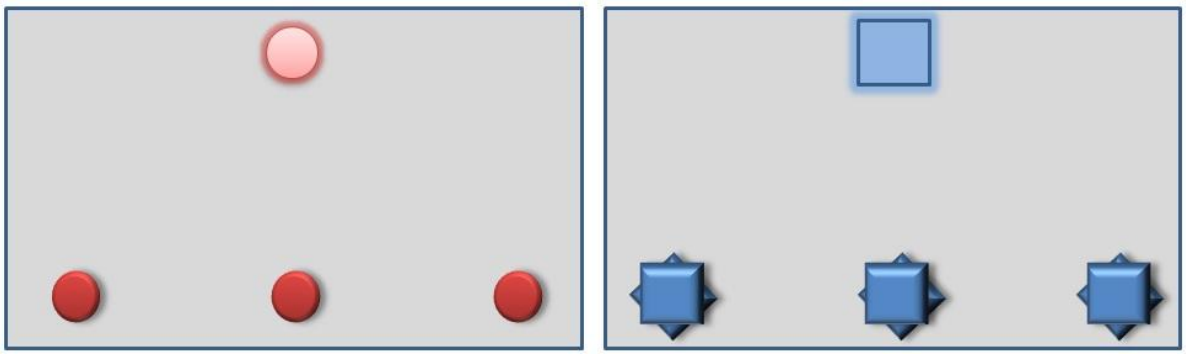

Figure 7: Circle and Rectangle Test Configurations

RECTANGLE

For the rectangle there are four specific orientations. The first one has the long axis of the rectangle laid parallel to the bottom edge of the table, while in the second case, it is the short axis. For the next two cases, long axis is angled at 45 degree and 135 degree to the table. For each case, the shape is placed in three standard positions relative to the marker on the table - the bottom left, the bottom right and bottom center. Therefore we have 12 test cases set up. In all the cases, the shape marker remains fixed in the top center position.

TRIANGLE

Similar to the rectangle, the triangle also has four specific orientations. The first one has the base of the triangle laid parallel to the bottom edge of the table, while in the second case, it is the tip. For the next two cases, the base is kept perpendicular with the tip pointing left or right. As in the above experiments, the shape is placed in three standard positions relative to the marker on the table - the bottom left, the bottom right and bottom Centre. Therefore we have a total of 12 test cases set up. In all the cases, the shape marker remains fixed in the top center position.

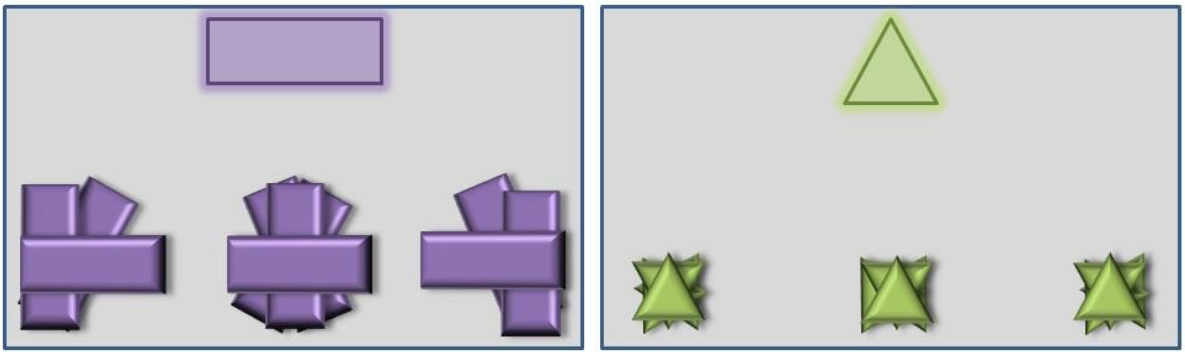

Figure 8: Rectangle and Triangle Test Configurations

Test Conditions

Precondition
The robotic system was given an opportunity to generate an affordance model for the given shape by performing babbling as described in the Algorithms and Data Structures Section.

Setup
The board is setup as shown in the test cases. The marker is in a fixed position near the head of the board. The active shape is placed away from the marker towards the foot of the board at one of the three pre-determined spots. The orientation of the shape is also varied for a given position.

Duration
The robotic system will be allowed 500 iterations to complete each test case. At the end of the period the simulation will be stopped and the results will be tabulated.

Conditions for Success
Success is defined by the ability to position and orient the active shape relative to the marker within an acceptable error tolerance. The success of position and orientation will be measured independently so as to be able to identify the situation in which the system is capable of performing one of the required actions, but fails to complete the other.

Test Subjects

The Robotic System in the Simulation is the test subject.

Results
After the completion of phase two, results were able to be captured for both babbling and simulation.

Babbling
Random exploratory behaviors were used to determine the affordances of each of the shapes through the use of simple hooked stick. In order to facilitate babbling and evaluation being executed as independent actions a means was put in place to serialize out the results of the babbling in a standard text file which also provided a means by which the output of babbling could be verified.

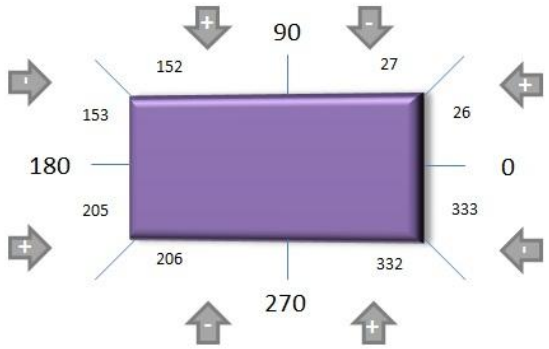

Figure 9: Rectangle Push Positions, Push Direction, and Orientation Change

For the initial verification the rectangle was selected since the outcome for almost actions can be reasonably predicted. The diagram above shows the general layout of rectangle at a orientation angle of 0 as implemented. The numbers represent the push positions at each of the midpoints as well as the corners of each edge. The grey arrows represent the direction of push that should effect each of the edges. The plus and minus signs in the arrows represent position and negative change in the orientation as the result of a push. Pushes in the center of the edges should result in more movement, while pushes toward the corners should results in more rotation. Based upon the above diagram one can expect to see four distinct regions of push distance alternating between large and small. Also based upon the above diagram one can expected to see 4 distinct regions of rotation alternating between large and small pairs of positive and negative changes in orientation.

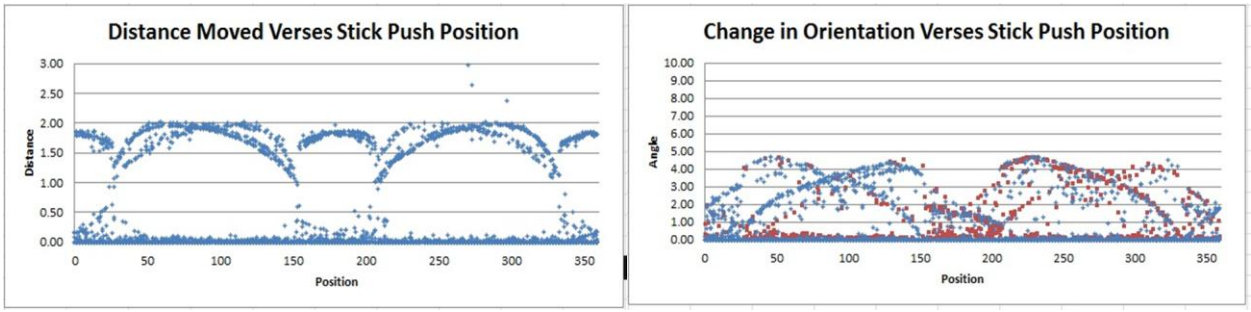

Figure 10: First Pass Rectangle Babbling Results

The diagrams above shows the results of the first babbling run. The distance graph at first glance shows the expected regions that match up to four edges. Upon closer examination a few details stick out. The push distance used was 3, so why is the maximum linear distance 2? Why are there two optimums for each region? What is with the near zero movement since it should have been discarded as invalid results? The change in orientation graph leaves a lot to be desired and obviously hints that physics rotation is not working correctly which might explain the artifacts in the other graph.

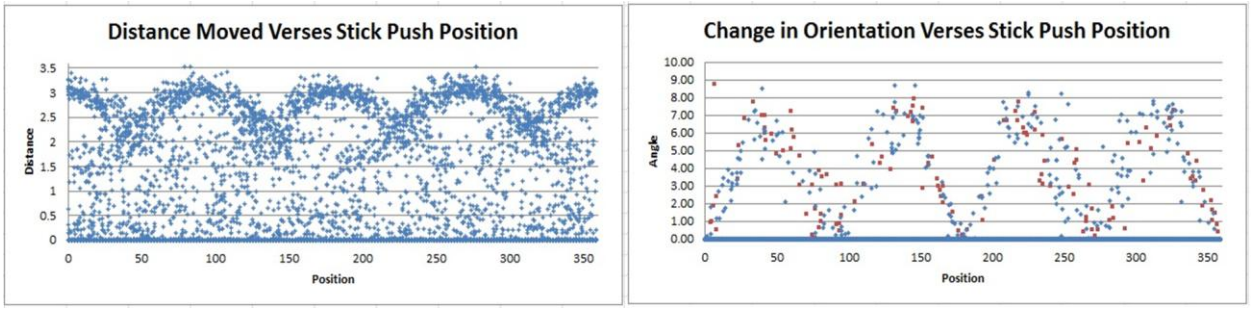

Figure 11: Second Pass Rectangle Babbling Results

The diagrams above shows the results of the second babbling run after fixing the physics so that rotations works correctly and ensuring that the stick is moved for the correct amount distance. The distance graph appears to have become a lot noisier and regions appear to now blend into one another. The change in orientation graph now appears to contain the correct number of regions; however there is no discernible pattern to the positive and negative rotations hinting that we may not be setting the correct sign for the rotation. This would explain the first graph since the initial orientation value could be wrong for a lot of case and therefore result in incorrect value.

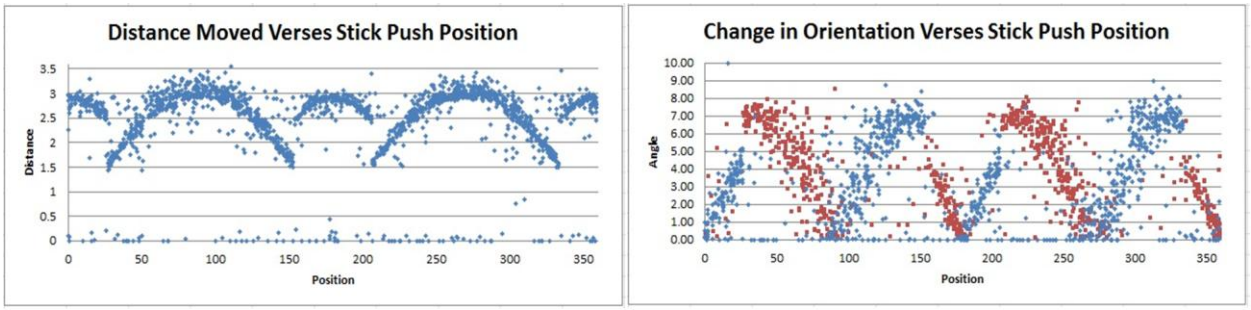

Figure 12: Final Rectangle Babbling Results

The diagrams above show the results for the final babbling run after fixing the change angle to correctly denote positive and negative changes in the orientation angle. The distance graph now shows the correct regions with a minimal amount of noise. Even better the change in orientation graph now shows the positive and negative pairs. For a sanity check the points for downward and left pushes were plotted. As seen below the graphs match with the push positions that one would expect to be affected by those directions of pushed. Based upon these result it is safe to say that babbling on the rectangle is nominally working correctly.

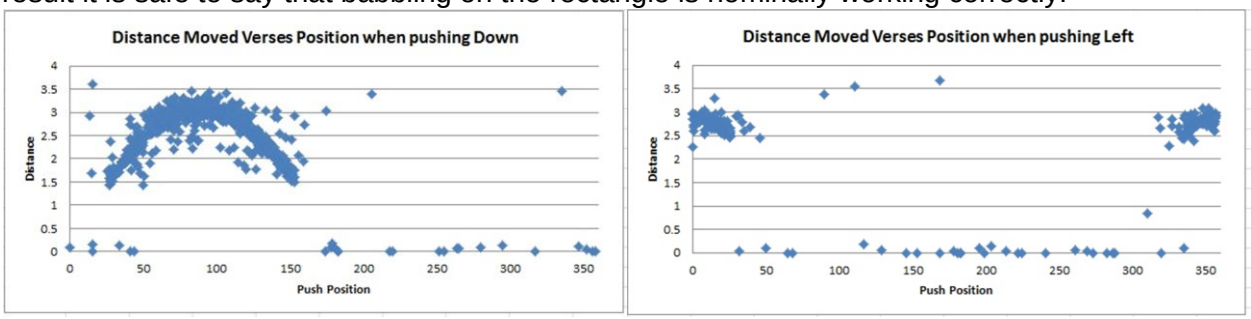

Figure 13: Rectangle Push Position to Push Direction Comparison

Performing the same analysis for the triangle babbling graphs produces similar observations. While definitely harder to read due to an increase in noise the distance graph does show 3 distinct regions of movement, one for each edges, as well 3 distinct regions of rotation pairs.

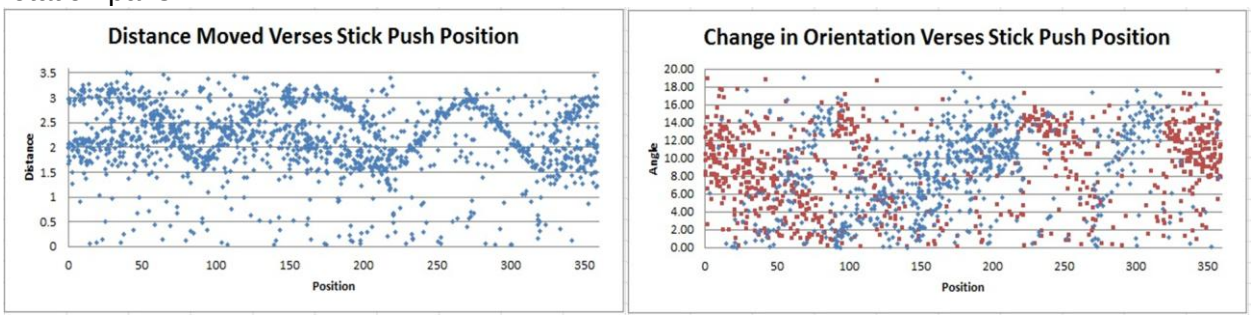

Figure 14: Triangle Babbling Results

Evaluation

The results from the babbling phase were combined with our strategy in order to evaluate the ability of our system to learn the affordance of the shapes through simple tool use.

Evaluation Implementation:

1) Save all configuration details for evaluation phase to a file
2) For each record from configuration file, repeat the following
3) Create the desired shape and corresponding shape marker and load the specific babble data-set. Update the positions and orientations of the same from the given configuration details
4) Calculate the liner and angular distance between current and destination positions.
5) Select the best result from the babbling set according to the strategy
6) Apply the corresponding action to the selected result
7) Check the resulting position. If valid, update times successful for the action, else just update times used
8) Repeat steps 4 to 8 until one of the following termination conditions is reached.
    1. Until the shape is within an acceptable tolerance of the marker position and orientation
    2. The number of required iterations has exceeded 500
    3. Oscillation is detected

Although in our proposal we had specific positions and orientation for puck and the marker, the evaluation is set up such that the above values can be varied and any number of test cases for the shapes can be run in one configuration file.

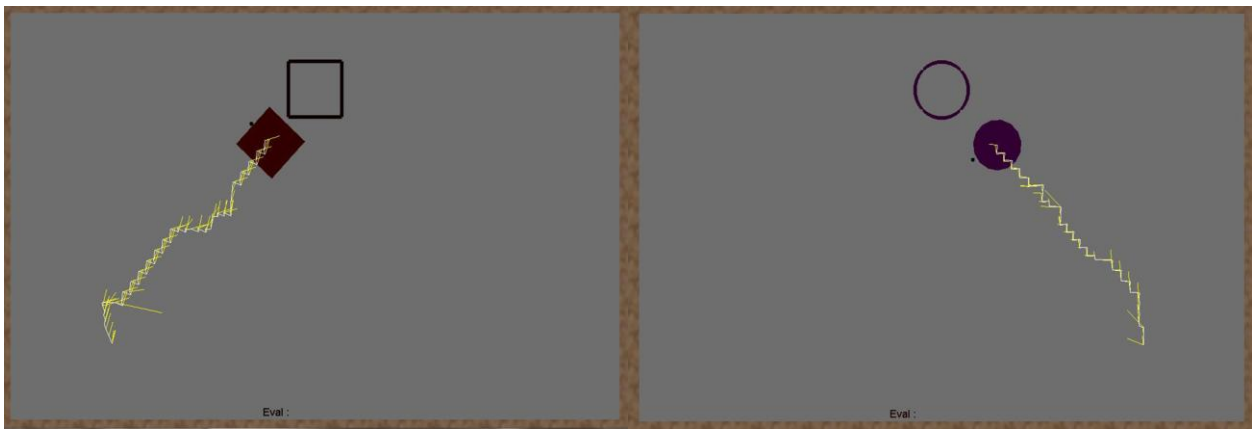

Figure 15: Evaluation Results for the Sphere and Square Shapes

The above diagrams above show how the simulation moves the shapes into its markers. Yellow markings show the predicted motion of the shape while the white markings show the actual motion.

Evaluation Analysis:

The two most difficult shapes to position into the marker are the triangle and the rectangle. The simulation had trouble pushing all of the shapes into the marker with only 200 tries. However, when the maximum number of tries was increased to 500, it was able to position the cylinder into the marker 100 percent of the time, an increase from zero percent. The success rate for the square went from 33 percent to 66 percent. The success rate for the triangle went from zero to 8.33 percent. The success rate for the rectangle did not improve; it remained at a zero percent success rate.

During our testing phase, we noticed that the simulation was able to position the shape into the marker but it was not able to orient the shape. So although the position of the shape was correct, the angle at which it was oriented was not. We believe that this is due to the amount of babbling data we've accumulated. We predict that if we generate more data, it will improve the simulation's ability to orient the shape correctly.

We also found that most of the errors that were generated were the results of the shapes moving off of the table. Whenever an action moves a shape off the table, the simulation regards it as invalid.

Conclusion

Learning shape affordances through tool use is a difficult a problem. Through this project we were able to show that using exploratory behaviors is sufficient for determining the movement and rotation characteristics of particular shapes. This was corroborated by the fact that the babbling results can be shown exhibit the expected characteristics of the shape data. Our simulation results however left a lot to be desired. While we easily able to position the pucks within the region of the marker, more often than not our strategy was unable to match up the orientation of the shape with that of the marker. Whether this is a flaw in our data model, a bug

in our simulation, or an invalid strategy is yet to be determined. Many lessons were learned while implementing this project and there is definitely room for future work

Lessons Learned

Through the course of this project several lessons were learned that are worth noting for future reference.

Data Collection

A complete set of babbling data for one shape would entail 360 x 360 x 4 records. That is, 360 degrees of puck orientation along with push on all 360 degree angles on the puck for each of the four cardinal directions. Assuming each babble takes a second for data collection, it would take 6 days to get a complete data set for one shape! We had realized the ramifications of this early on in the implementation phase and so tried a lot of optimizations that would lead to a simplified but workable data set which would produce good evaluation results.

As our evaluation phase sorted and chose babble data mostly based on puck orientation, we initially decided to increment babbling 360 times in order to get viable results for each orientation. But instead of 360 iterations for push angle, it was decided to collect a random set (including random direction) for a standard number of iterations of about 20 valid pushes, a valid push being one where the puck does not end up outside the boundary of the table.

But the data set generated was very small and had the following disadvantages:

1- During evaluation phase, when a record is sorted based on the babbling data on orientation, often times there wouldn't be a viable result -action pair for a push at a particular position and direction. So the puck would end up stuck in a loop, trying and failing to get a decent action that will enable it to move.

- The data was optimized such that an action with one cardinal direction during babbling could be translated into additional records for the other three directions for equivalent orientations and push positions of the puck. This is explained in Oversimplification.

- The physics engine was setup to be very precise with the calculations. As orientation was initially a floating point value, finding the exact match in the babble data set during the evaluation phase was in many cases next to impossible. This was circumvented by declaring orientation as an integer which decreased accuracy.

- Further, to prevent the strategy from stalling due to lack of orientation data, it was programmed to check all orientation values that fell within + or - one unit and to choose the closest action-result pair that matched.

- The initial 360 degree babble iterations were later increased to four rounds of 360 degree iterations in one degree increments making for a total of 1440 iterations and the random push number for each iteration was increased from 20 to 50 in order to generate a more complete data set.

2 - Secondly, our standard push distance of 3 on the puck would result in a movement anywhere between 0 to 4 units. In the final stages of evaluation, where the strategy is more focused on getting the right orientation, the puck would keep overshooting the targeted

destination trying to get it within the tolerance limits. This was termed as Oscillation, explained in detail in the limitations section below.

3 - Thirdly, with the smaller initial data-set, we were unable to get a clear pattern on our resulting babbling actions. On increasing the babbling data, the movement and orientation results for each shape began to take on a clearer picture as evidenced by the graphs in the previous section. It also served as an independent check as we were able to verify predicted patterns with the actual results. As expected, It was found that each puck had a clear and distinct pattern of movements and orientations based on its initial orientation, push position and direction, further corroborating our babbling strategy.

Over Simplification

Often times when dealing with complex problems it is desirable to try and simplify them as much as possible so as to remove as much of the complexity as safely possible. This was definitely the case when designing our data model and selecting the number of push directions that would be supported. Early on the choice was made to only push in the cardinal directions since this would limit the amount of space that would have to be explored while babbling. At first this seemed like a good idea, however while trying to optimize our babbling we came to an insight that if followed to its completion shows that this was not only unnecessary, but foolish.

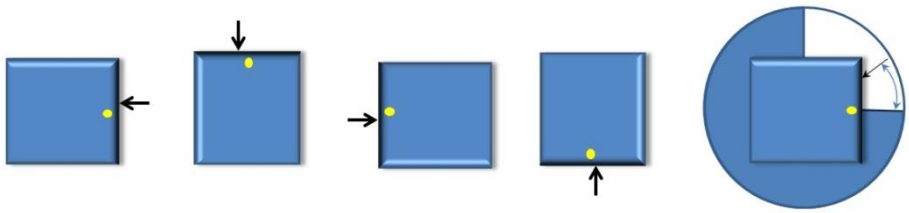

Figure 16: Equivalent Pushes

The key insight is that a push at any position for a given orientation can be translated into another push at a different orientation this is equivalent. In other words rotating the push, the orientation, and the motion angle by the same amount of degrees results in an equivalent action and results being generated. In our case each push has 3 equivalent pushes since there 4 potential push directions. At first this may seem like developers insight, however the equivalent pushes are actually the same thing as if orientations were to be mapped into a single quadrant. This is where the light bulb turns on.

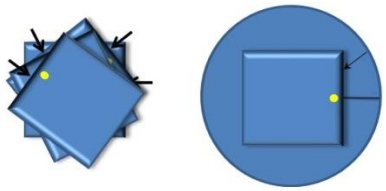

Figure 17: Equivalent Pushes Mapped to Orientation Coordinate System

Currently the pushes are defined relatively to the Cartesian coordinate system. This means that a given push is only valid for given orientation. This is a limitation caused by the fact that we only have 4 pushes. If the number of pushes were increased to 360 the coordinate system of

the push direction could be shifted to be shifted to that of the orientation. This would eliminate the need to track the current orientation, greatly simplifying the data model while improving the strategy since more all options would be available regardless of the orientation.

System Limitations

Below are some of the hurdles that were faced and partly fixed or circumvented during the limited time window of this project.

Floating Point

Initially the orientation and position were all saved as floating point values. As the calculations were very precise, there were a number of results where the orientation changed by a few decimals values. When it came to the evaluations phase, the strategy initially tried to choose the exact orientation from the babbling data-set. This approach did not work for the obvious reasons. As a work around, orientation was set to integer. This caused loss if information because of the precision limit and thus the accuracy in calculation was diminished.

A further modification was made to enable the strategy to choose closest orientation match within one unit of the desired angle. While this did enable to robot to choose an action instead of being stalled due to lack of data; it also lead to more puck Oscillations.

Oscillations

Due to the nature of the data model and babbling quite often the shape never reaches the desired tolerances, but rather starts to bounce around the solution. This behavior is commonly referred to as oscillation. In this project case oscillation can occur for two reasons: The babbling phase did not fully explore the potential data space and therefore the strategy is unable to move to the desired location. The strategy chose a path to the object that results in no action being available that can move the shape within defined tolerances.

Physics Simulation

While the simulations helped us gather more results at a faster pace than an actual robot implementation would have, it also was time consuming to get the physics of the shapes and the simulation setup to work correctly.

Secondly, the angles of the shape were calculated by the physics engine in radians; while orientation had been factored as degrees for our strategy. The constant conversions between the radians with floating point values and the degrees as integers caused loss of precision which affected the results.

The triangle was a special case in our project and it was definitely the hardest. Bullet does not provide a default 3D triangle therefore it was necessary for us to implement our using a convex hall shape. Because of this it proved very difficult to get the triangle to work in a consistent and correct manner and unfortunately the final solution was uncovered very late in the project resulting in very little babbling and evaluation being conducted on the triangle.

Implementation

The implementation for this project was complex to say the least. There were many components that need to be integrated together such as shape physics, strategy, babbling, and evaluation and often times these components shared common functionality such as creating a shape, setting the position of the shape, pushing the shape, and calculating. Initially it was

assumed that everyone was on the same page with what needed to implemented and how best to go about doing. It quickly became apparent this was not the case as functionality was often implemented in a incompatible manner and shared functionality was not being properly modularized so that it can be used in multiple places and tested independently from other functionality. This caused progress to be initially very slow with many headaches due to functionality not working correctly. With a little effort this oversight was able to rectified and by doing so we able to see vast improvement in both quality and the amount of work that was being accomplished.

Future work

There are a many avenues which can be expanded upon for this project.

Simulation

As mentioned in our initial proposal, one of the development model iterations was to integrate OpenCV and track and update the positions for the stick and marker from the pre and post frames. Secondly, isometric projection of the shapes could be done as the next logical step from the current 2D view.

Robot

The other option is to implement this entire project on a robot and collect the datasets from babbling. The robot would then undergo the evaluation phase and we can compare and contrast with the simulation results.

Strategy

There is currently only one strategy implemented for evaluation. It simply tries to narrow the linear distance to the desired position and then tries to get the closest orientation. Various other strategies could be implemented to see which one returns the optimal results for least number of steps to the destination. Evolutionary learning can also be implemented such that the robot may learn to evaluate different strategies in the long run and develop its own methodology.

Another strategy is to implement a method that will allow the simulation to learn symmetry of the shapes. For instance, the cylinder does not need to be oriented at a specific angle in order for it to fit in the shape marker. At any orientation, the cylinder will fit. However, the rectangle will only fit in the shape marker if it is at an orientation of zero degrees or 180 degrees.

Others

More shapes could be used for testing. Irregular shapes should also be taken into account. Furthermore, composite shapes and objects used in our daily lives could also be used for tests.

Further evaluation of the data-sets can be done to figure out specific patterns for different shapes. Knowing the pattern of behavior of a specific object or shape will help in object identification also. The robot after babbling for some time on a novel shape can check with its database of existing shapes and predict the behavior of the new shape.

Acknowledgement

We wish to thank professor Alex Stoytchev for his guidance for this project and his patience with our various topic proposals. We also thank Jivko Sinapov, our TA for this course who was very approachable and helped with the initial topic selection.

Project Team

Jeremy Bennett
Jeremy has over 13 years of programming experience in C++, Java, Asm, OpenGL, and Cg/GLSL with his main focus being Computer Graphics and Rendering Technologies. He holds a bachelor's degree in Computer Engineering and master's degree in Human Computer Interaction, both from Iowa State University, and is currently pursuing a PhD in Human Computer Interaction with a potential co-major in Computer Engineering. Jeremy is also an Advance Software Engineer at Siemens PLM Software working on their Direct Model Scene Graph library and has over 7 years of industry experience.

Poorvi Joebert
Poorvi earned a bachelor's degree in Computer Science from Anna University in India. Upon graduation she accepted a position with Wipro Technologies where she worked on Portal and Content Management as a Senior Software Engineer. She has six years of programming experience in Java and two years in C++. After about three years in the industry, she decided it was time to further her education and applied for Master's in Computer Science and is now co-majoring in Human Computer Interaction. She is currently working as Research Assistant on the Thinkspace project.

PJ Campbell
PJ has a bachelor"s degree in Computer Science from St. John"s University in New York. She has six years of programming experience in Java and three years in C++. She also has working knowledge of OpenGL, OpenSG, and Open Dynamics Engine. PJ is currently a research assistant in VRAC working on the Meta!Blast project.

References

[1] Chemero A., Turvey M. T. 2007. Gibsonian affordances for roboticists. *Adaptive behavior - animals, animats, software agents, robots, adaptive.*

[2] Gibson E. J. 1988. Exploratory behavior in the development of perceiving, acting, and the acquiring of knowledge. Cornell University. Ithaca, New York.

[3] Gibson J. J. 1979. The theory of affordances. *The Ecological Approach to Visual Perception.*

[4] Kraft D., Detry R., Pugeault N., Baseski E., Piater J., Kruger N. Learning objects and grasp affordances through autonomous exploration. *Proc. of the 7th international conference on computer vision systems.*

[5] Montesano L., Lopes M., Bernardino A., Santos-Victor J. 2007. Affordances, development, and imitation.

[6] Ortmann J., Kuhn W. 2010. Affordanes as qualities. *Proc of the 2010 conference on formal ontology in information systems.*

[7] Sinapov J., and Stoytchev, A. 2010. The boosting effect of exploratory behavior. *Proc of 24th national conference on AAAI*.

[8] Stoytchev, A. 2005. Behavior-grounded representation of tool affordances. *Proc. of IEEE ICRA*, pp. 3071–3076.

[9] Stoytchev, A. 2007. Robot tool behavior: a developmental approach to autonomous tool use (Doctoral Dissertation). Georgia Institute of Technology Atlanta, GA, USA.

[10] Ugur E., Dogar M., Cakmak M., Sahin E. The learning and use of traversability affordance using range images on a mobile robot.

[11] von Bayern A., Heathcote R., Rutz C., Kacelink A. 2009. The role of experience in problem solving and innovative tool use in crows. *Current Biology*

[12] Donald Norman, The Design of Everyday Things, ISBN 0-465-06710-7.