Learned Shape Affordances through Simple Tool Use

Jeremy Bennett
PJ Campbell
Poorvi Joebert

Introduction

A hallmark of greater intelligence is the ability to utilize 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. There have been numerous studies in the realm of developmental psychology to study tool use in animals; however the same is not true in developmental robotics. In this project we will propose a novel approach to extend Stoytchev's Behavior Grounded Representation of Tools Affordances for a variety of differently shaped pucks using a simple hooked stick as the primary tool.

Motivation

Most of today's robotic systems are programmed to work efficiently, but they are also 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.

Developmental robotics targets this prevalent issue in robotics and aims to create machines with robust intelligence that are autonomous, 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 on its own without need of human interference.

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.

Previous work on learning the affordances of tools and autonomous tool use by Alex Stoytchev, looked at manipulating a single object using various tools. The focus of our project is on using a single tool to operate on different objects. Knowing the affordance of an object - which is determined by its shape and the entity using it - is important for manipulating the object for various tasks. Because of the complexity involved in having a robot learn about affordances of different objects, we have chosen to start with regular shapes for this project.

The results of our experiments in developmental robotics could also be used to extend and gain deeper insight into developmental psychology of humans and animals.

Audience

The target audience for this project is researchers in in the fields of Developmental Robotics and Developmental Psychology as well as the automated processing industry.

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 Work / 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.

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.

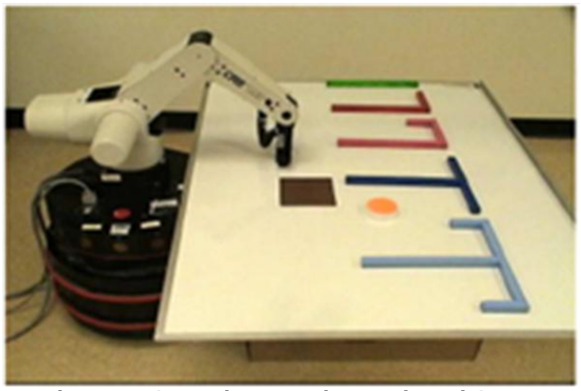


Figure 1: Stoytchev's Robot and Tool Setup

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.

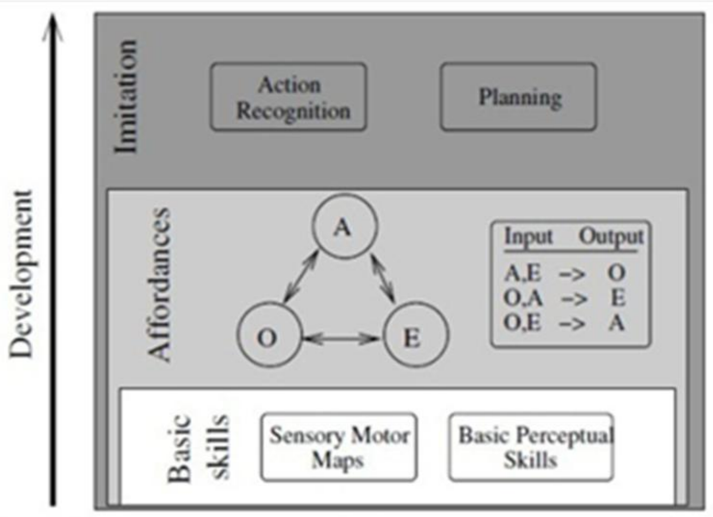


Figure 2: Montesano Developmental Architecture

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].

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.

Previous Experience

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 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.

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

Development

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 Studio 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/

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.

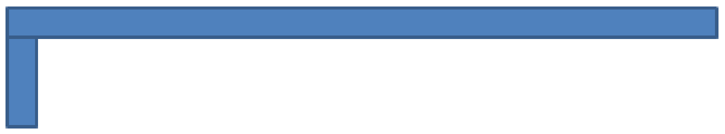


Figure 3: Hooked Stick

Pucks

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

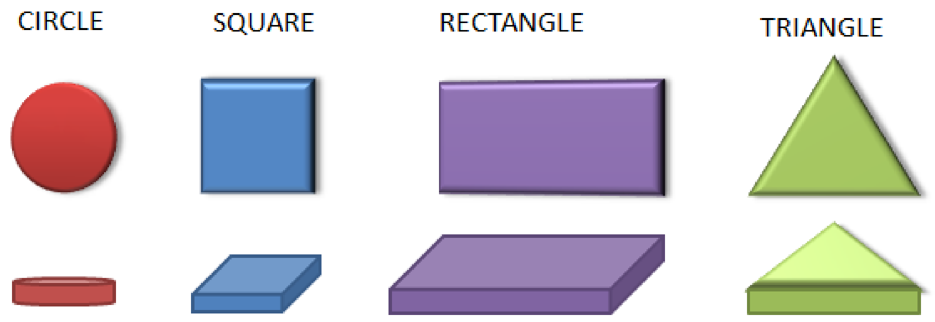


Figure 4: Set of Defined Pucks

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.

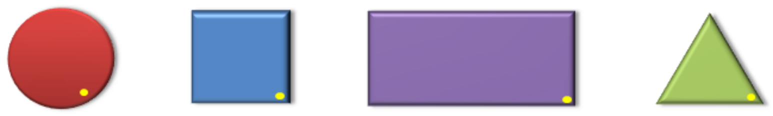


Figure 5: Puck Markers and Push Positions

Algorithms and Data Structures

Algorithms

Two main algorithms will be implemented: Babbling and Strategy.

Babbling

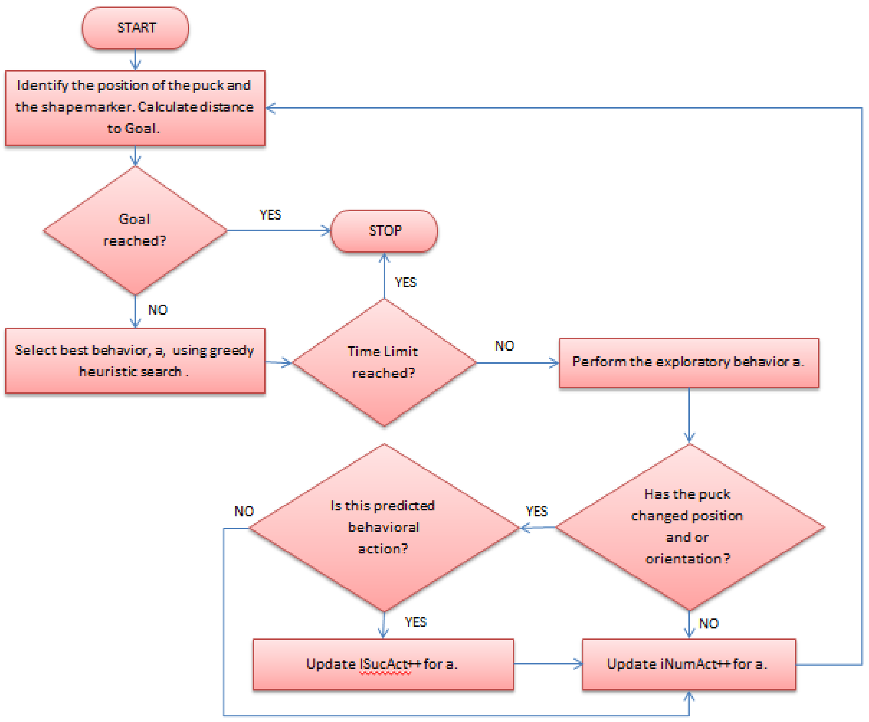
The babbling algorithm is responsible for determining an object's affordances in relation to the tool by applying a random set of behaviors. The combined affordances establish a model that allows desired results to be achieved. In the case of project the babbling algorithm will randomly select a push 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.

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.



START
Identify the position of the puck and the shape marker. Calculate distance to Goal.
Goal reached?
YES
STOP
NO
Select best behavior, a, using greedy heuristic search .
YES
Time Limit reached?
NO
Perform the exploratory behavior a.
Has the puck changed position and or orientation?
YES
Is this predicted behavioral action?
NO
YES
Update ISucAct++ for a.
NO
Update iNumAct++ for a.

Figure 6: Strategy for moving puck over marker

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 offset of the object and degrees of rotation.

```
struct obj_res
{
   float      vOff[2];  // Change in position
   float       fAng;     // Change in orientation
   obj_act *pAct      // Associated action
};
```

Data Definitions

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.

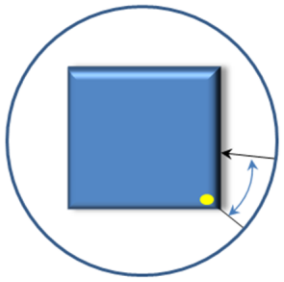


Figure 7: Push Position

Push Direction

ID used to identify one of the four unique directions in which the stick can push the puck.

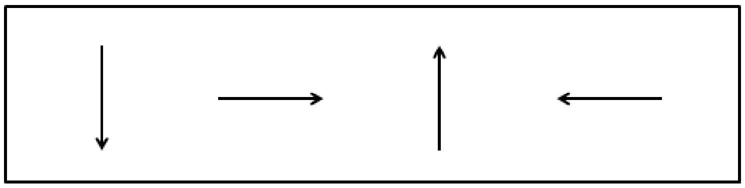


Figure 8: Set of Defined Push Directions

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.

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

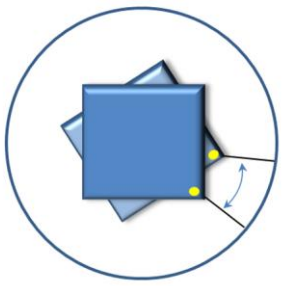


Figure 9: Orientation Angle

Implementation Methodology

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

Developmental Model

The Shape project will be implemented using a standard waterfall developmental model with each of its iterations having a clearly defined set of goals to be achieved. This should allow issues with our general approach to be quickly identified without having to worry about the additional complexities that will be introduced in later iterations.

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

The project responsibilities will tentatively be broken up as listed. As the project progresses it may become necessary to shift responsibilities to accommodate something be easier or harder than originally expected.

Jeremy

- Framework and Architecture
- OpenGL Rendering
- Common Data Structures

PJ

- Simulation
- Physics

Poorvi

- Strategy
- OpenCV

Timeline

The project will implemented over a period of a month and a half with each of the non-optional faces given approximately two weeks to be completed. Dates are subject to change if we should happen to get ahead/behind schedule or decide to proof of concept an item from a later iteration.

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
Start Phase 2

Week of March 27
Finish Phase 2

Week of April 3
Start Phase 3

Week of April 10
Finish Phase 3
Start Project Report

Week of April 17
Finish Project Report
*April 21 - Project Report Due

Risks

This project has three key risks: Bullet, OpenCV, and Time.

Bullet
The project team has limited experience using the open source physics. Further the limited experience we have has shown us the engine can be quite unforgiving. This being said some of the work from the previous experience can serve as solid basis for which simulation can be derived.

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 require was implemented in part while taking Computational Perception and should easily be adapted to our needs.

Time
The timeline for completing the project is very tight and the goals are definitely aggressive. The iterative development plan should help substantially, however we will definitely be cutting it close.

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.

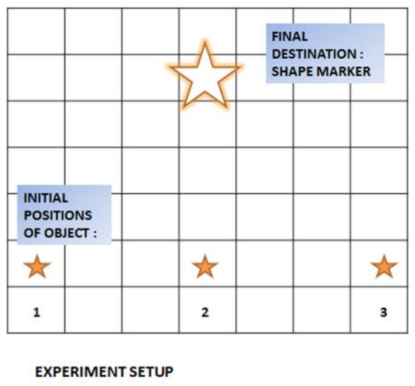

FINAL DESTINATION : SHAPE MARKER
INITIAL POSITIONS OF OBJECT :
1
2
3
EXPERIMENT SETUP

Figure 10: Test Setup

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.

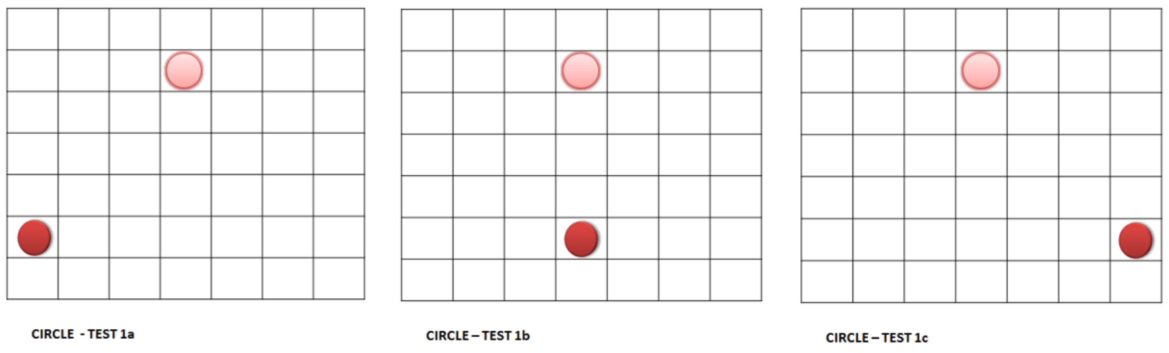


Figure 11: Circle Tests

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 45degree 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.

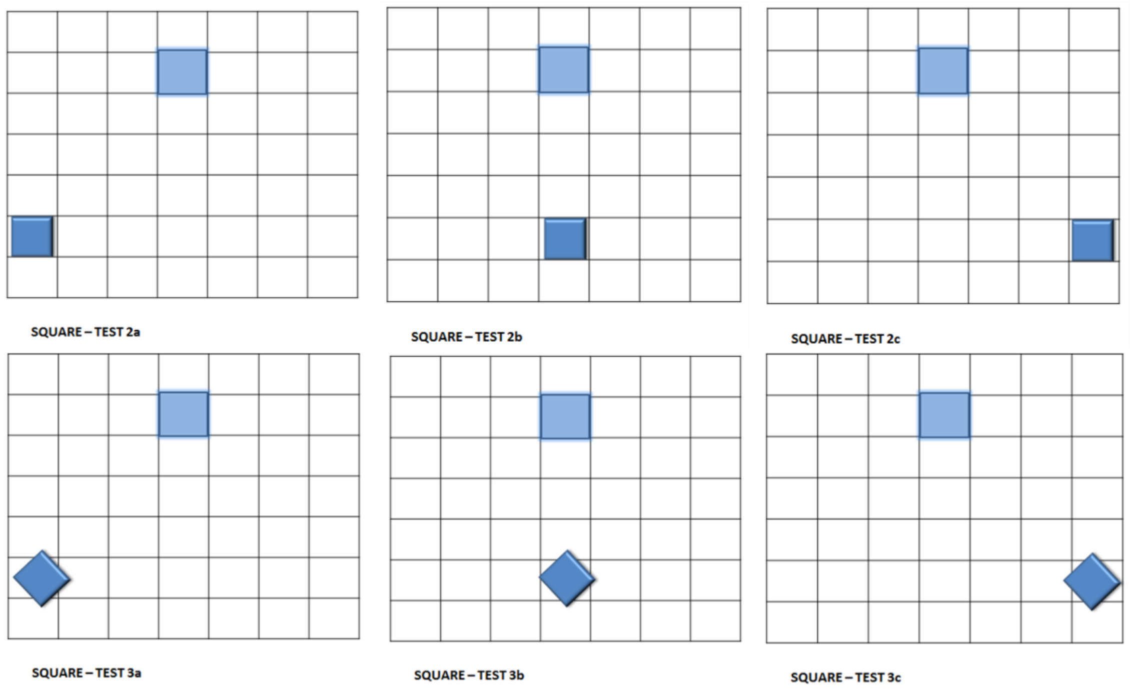


Figure 12: Square Tests

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.

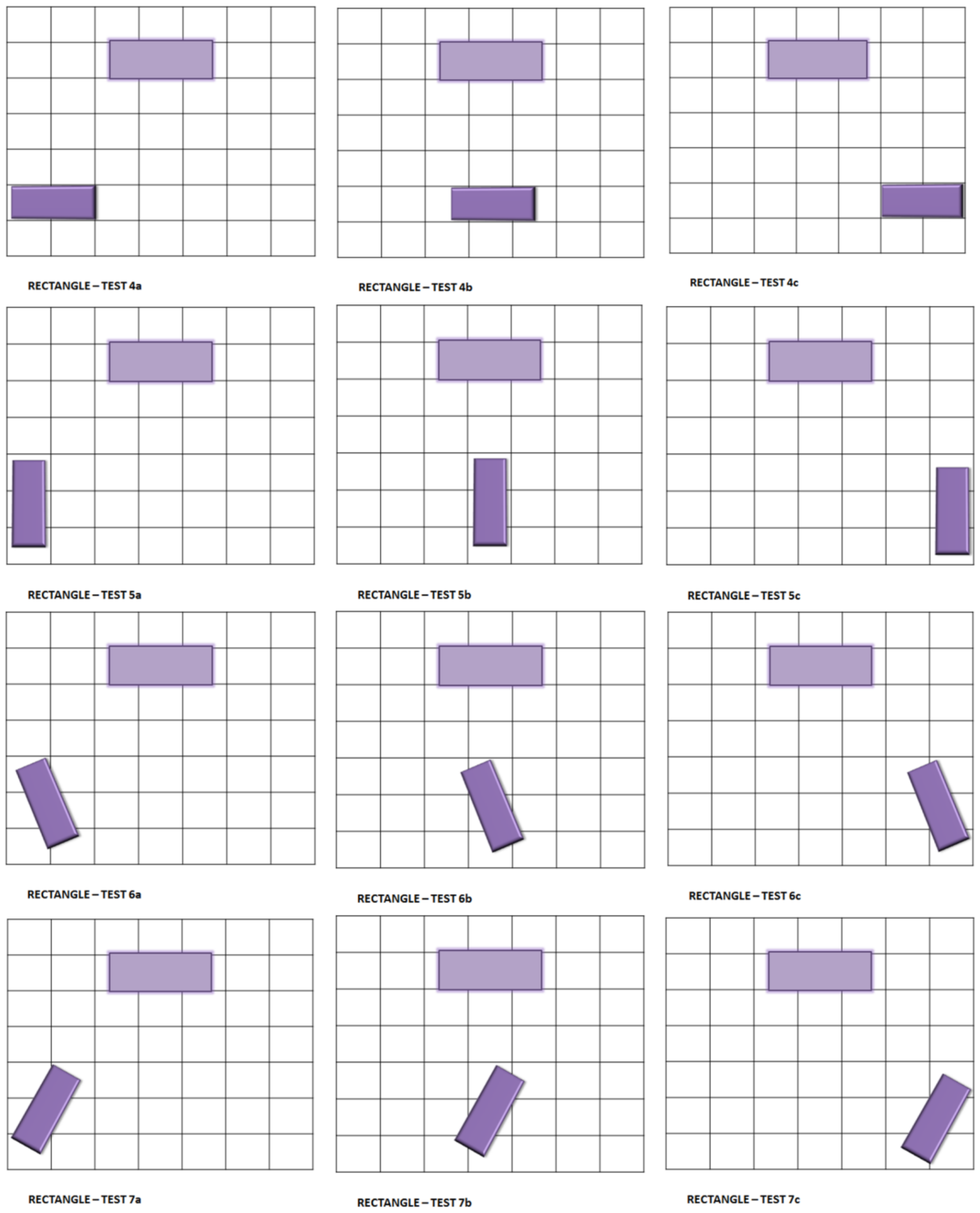


Figure 13: Rectangle Tests

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.

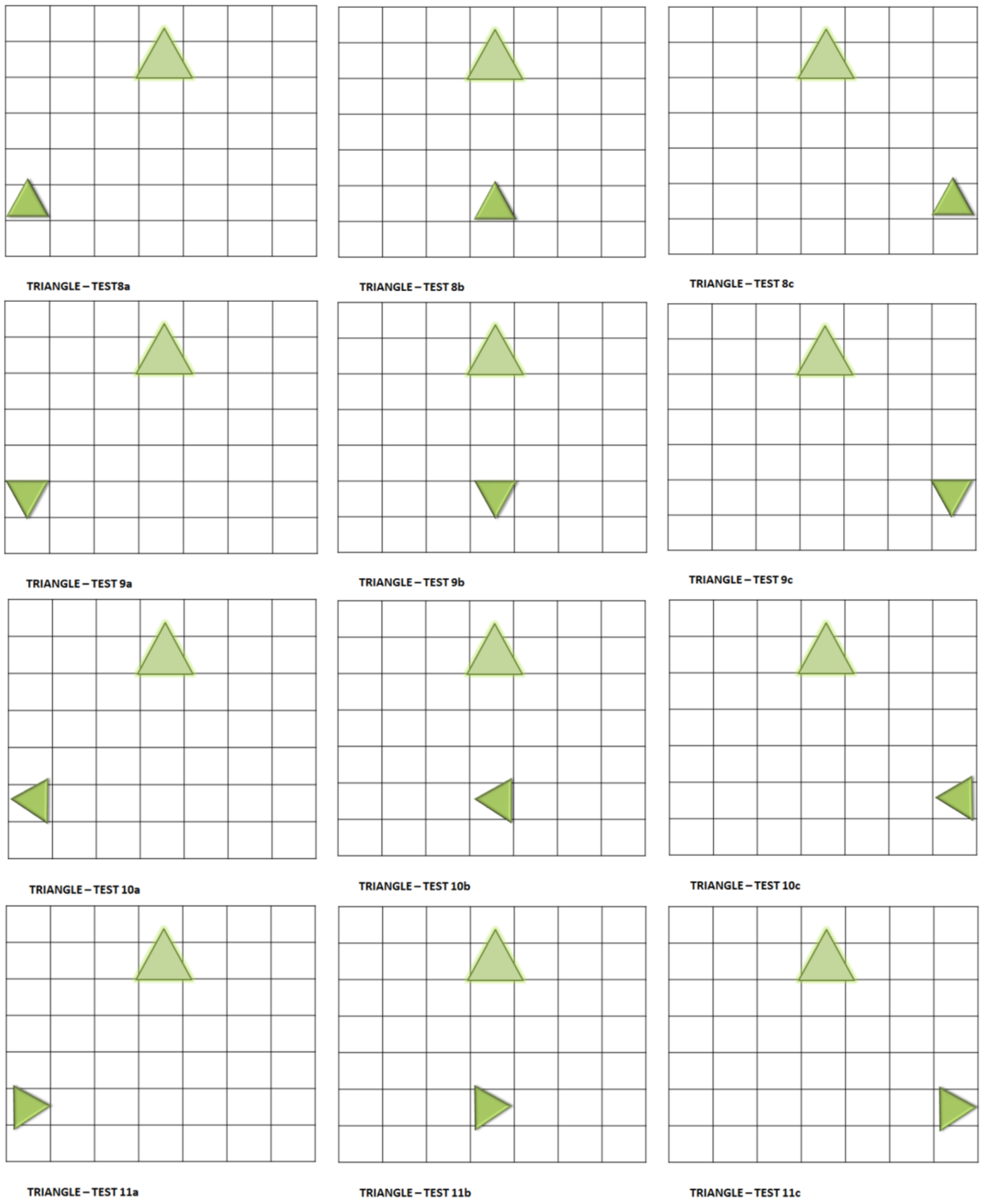


Figure 14: Triangle Tests

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 two minutes to complete each test case. At the end of a two minutes 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.

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