Perceivable Affordances and Mobile Robots

Dilan C. Edirisinghe
Advisor: Professor David S. Touretzky
April 29, 2005

I. Introduction

Intelligently manipulating arbitrary objects using a mobile robot is a hard problem. A programmer who whishes to do so must take care of problems ranging from the visual recognition of the object to the low-level details that are associated with manipulating that object. Low-level details include the shape and geometry of the object to be manipulated, the robot's position and orientation within the environment, and the kinematics associated with manipulating the robot and the object.

This project abstractly merges the perception and manipulation frameworks into a concrete mobile robotic programming framework by utilizing the idea of "affordances". Coined by the noted psychologists J.J. Gibson, "affordances" describe the realizable actions on an object that an environment "offers, provides, or furnishes to an animal."

Many classical mobile robotic programming frameworks force the developer to determine viable movements and actions from object representations provided by a visual module without considering the robot's perception of what the object is. Problems range from a technical issue involving the robot's physical constraints that determine possible actions it may perform, to problems of a more aesthetic nature where the robot does not portray intelligent behavior (thus, it more or less follows a set of pre-programmed rules). In contrast, the affordance method develops a methodology in which a robot recognizes the utilities associated with an object representation and uses this set of utilities to allow the user to determine an action to perform.

The Sony AIBO robot dog provides the testing ground for the affordances. This method is implemented as a component of Tekkotsu, an open source development framework for the AIBO dog. Although the kinematics package of Tekkotsu is used, this method generally works for kinematics packages as long as that package provides the necessary functionality in controlling the low-level aspects of the dog. Tests include the dog recognizing a ball to be a representation of an object that "affords" the ability to be stared at or moved by the chest or the paw (this itself is determined by the size of the ball), as well as "affording" movements to the dog which sever to correctly position and orient it with relation to the object.

II. Affordances

Originally coined by the psychologist J. J. Gibson, the word "affordance" describes the relationship between an animal, some object, and the environment—the object is what the environment "offers, provides, or furnishes to an animal." His claim is that animals

obtain affordances in objects directly and immediately in the context of the environment. An affordance can be considered a property of an object in relation to animal and its environment. For example, consider the case of opening a door. The door has a handle that can be twisted to open the door—it does not matter what type of handle the door has, since the handle itself "affords" the action of opening the door.

Norman helped popularize the idea affordances in his book *The Psychology of Everyday Things*. He further discusses the difference between the affordances Gibson described with perceived affordances—affordances that are not necessarily real, but arise for some animal perceiving and making a mental representation. Thus, the real question with Norman and perceived affordances is the following: does the user finds the action afforded by the affordance meaningful?

Affordances, provides a powerful abstraction that allow objects to be described not by their visual qualities, but by the utility they provide an animal with respect to its environment. Using Gibson's idea, an implementation of a sample Affordance system has been completed on the AIBO robot dog; this system helps demonstrate the usefulness of the Gibson's affordance concept (from cognitive science) in programming mobile robots.

III. The Affordance Implementation

Affordances are organized into a type hierarchy with the base type as being the Affordance class. In the proposed formulation, four components characterize affordances: the action afforded, the estimated achievability, the means of achieving the action, and the means of assigning a "successfulness" metric to the outcome of the attempted action. The formulation organizes actions into a type hierarchy based on the object being operated on and the result to be achieved. Currently, the formulation requires each object to have its own set of affordances programmed for it, with the manipulation of lines and balls being the main objects of focus in this research. The perception of lines and balls and what they afford encompass the components currently implemented; the robot can touch objects, visually explore them, and, in the case of the ball, manipulate the object. Manipulation of the ball consists of pushing the ball using the dog's chest or the paw. Experimentation, at the moment, is focused on the fine positioning of objects using the AIBO dog.

Affordances are not recursive: they do not internally refer to another affordance to insure the completion of their actions. Each affordance acts by itself to perform its specified task, but certain affordances may not be recognized as viable options unless another other affordances is performed. For example, the dog needs to be next to a ball for a push ball affordance to be recognized. The dog generally is not next to the ball and is some distance away from it; however, if the user invokes the dog's walk to ball push position affordance, then the dog should end up next to the ball and a push ball affordance would be recognized the next time the user requests the affordances that the dog recognizes. Another decision was to determine what kind of actions could be classified as affordances. For example, high-level state machines to control some kind of behavior organization or advanced path planning algorithm are not classified as possible

affordances. Thus, the affordances this system considers are those that robot can perform directly without too much overhead on resource and path considerations.

Affordances are organized into a hierarchy of inherited classes. At the base level is the Affordance class: this class contains the basic information of an affordance including the affordance name (as a string) and the affordance type (as a value). This class also contains an abstract function that deals with the setup of the state machine. The Affordance class is then partitioned into four subclasses: the Visual Affordance, the Touch Affordance, the Push Affordance, and the Object Affordance. The purpose of the first three classes is to provide some general functionality associated with the tasks of visual processing and the kinematics involving touching and pushing objects. The Object Affordance comprises a general description for tangible objects (in this case, lines and balls) that the dog sees. Its two subclasses are the Line Affordance and the Sphere Affordance, each of which provide information associated with the object they represent. All other affordances are created by inheriting (utilizing multiple inheritance in C++) from one of the above-mentioned classes. For example, the Explore Line affordance inherits from the Visual Affordance and the Line Affordance classes.

IV. Affordances as State Machines

To perform the afforded actions of an object, each affordance type has an associated state machine implemented using the Tekkotsu state machine formalism (which uses event passing for states transitions). Tekkotsu itself relieves the user of some programming by providing head motion and body locomotion state nodes. An affordance's state machine determines when an affordance has been successfully completed as well as whether the affordance encountered an error when attempting to perform its action. No enforcement exists on exactly how complicated the state machine for any particular affordance is. Also, the programmer can still fully utilize the visual systems of the dog: this is particularly useful when the programmer needs to update the camera to check if an affordance has properly completed it specified task.

An example of a simple affordance state machine is the Explore Line affordance. This affordance is recognized when a line segment becomes visible on the camera. The action this affordance performs is to incrementally move the dog's head along the line and visit both endpoints of the line, determining not only the coordinates of the line in relation to the dog, but also the length of the line. The state machine for this affordance is particularly simple and consists of the three state nodes: a Wait for Image Update state node, an Analyze Image state node, and a Move Head state node.

The Wait for Image Update state node's purpose is to wait for the dog to update its camera and formulate a new image. The Analyze Image state node determines a variety of details: it checks to see if the dog still sees the line the user specified, it determines whether (and calculates if necessary) the next gaze target, and it checks if an endpoint of the line is found. A new gaze target is calculated if one of the following end state conditions is not met: both endpoints of the line are found, the newly generated image does not contain the line the user specified, some head joint reached one of its joint limits. Once the new gaze target is calculated, control moves to the Move Head state node which simply moves the head to specified gaze target; after completing this head motion, control returns back to the Wait for Image Update state node.

This example illustrates the basic use of a state machine in the affordance model. The use of return codes helps prevent cases of infinite looping and allows the user to be informed when the affordance completes (or when an error occurs impeding it from completing) its specified action; for instance, if some head joint reaches its joint limits, the affordance will return out of the state machine and produce a return code for the user specifying the error it encountered.

V. Achievability Codes

Before a user attempts to perform an affordance, he or she has the option of obtaining information of how successfully the system thinks this affordance will complete. The following classification describes the different "achievabilites" of an affordance: achievable, partially achievable, potentially achievable, and unachievable. Requiring detailed knowledge of the robot's structure and physical constraints, the process of predicting achievability is aided by Tekkotsu's inverse kinematics module, which helps determine the necessary joint motions to achieve the intended action.

Affordances chosen as "achievable" consist of those affordances whose designated object's location, as well as the robot's pose with respect to said objects, permits the required motion. The robot's pose comes into consideration due to the limited range of joint motion as well as the issue of the robot obscuring the target object with its own body (thus, it cannot see the object). Examples include a push affordance being achievable if the object is within reach of the robot, and an Explore Line affordance being partly achievable if the robot's head reaches the pan and tilt limits preventing any further attempts at examination of the line.

Before a user attempts to perform an affordance, he or she has the option of obtaining information of how successfully the system thinks this affordance will complete. The following classification describes the different "achievabilites" of an affordance: achievable, partially achievable, potentially achievable, and unachievable. Requiring detailed knowledge of the robot's structure and physical constraints, the process of predicting achievability is aided by Tekkotsu's inverse kinematics module, which helps determine the necessary joint motions to achieve the intended action.

Affordances chosen as "achievable" consist of an object whose location permits the required motion. Additionally, the robot's pose comes into achievability consideration due to the limited range of joint motion as well as the issue of the robot obscuring the target object with its own body (thus, it cannot see the object). Examples include a push affordance being achievable if the object is within reach of the robot, and an Explore Line affordance being partly achievable if the robot's head reaches the pan and tilt limits preventing any further attempts at examination of the line.

Partially achievable classification describes those affordances that can partly achieve the action to be performed. For example, if the dog has a partial image of a ball in its camera and if attempts to look at the center of the ball (via Look at Center of Ball Affordance) would make some joints exceed their limits, then the affordance would be classified as partially. Potentially achievable affordances describe those affordances that would be achievable if they require only a single additional affordance (whose action alleviates the achievability difficulty of the first) to be performed. Otherwise, the affordance obtains the unachievable label.

VI. Return Codes

The topology of implemented action results consists of a success signal or one of the following failure signals: lost contact or position problem. These action results are termed the affordance's Return Code. A lost contact failure signal manifests in two different ways depending on whether the affordance is an instance of a Visual Affordance or an instance of a Touch/Push Affordance. For a Visual Affordance, a lost contact failure implies the disappearance of the object from the dog's field of view; for a Touch/Push Affordance, lost contact implies that the ball rolled away or the robot lost physical control over the ball. A position problem failure signal occurs from a lack of solutions to the inverse kinematics arising from restrictions imposed by joint limitations associated with the dog.

To deal with the position problem failure signal (as well as to be able to actually push an object around) the Walk Affordance is introduced. Each non-Walk manipulation affordance has an associated Walk Affordance, which repositions the robot so as to allow the intended action to be undertaken. For example, suppose the Explore Line affordance is potentially achievable. Then invoking the Walk to View Line affordance compels the robot to walk to the line and to orient itself in the desired direction. This results in the Explore Line achievability changing from a potentially achievable action to an achievable action, thus alleviating the position problem failure signal.

An affordance signals completion by sending an Affordance Completion Event; the user must subscribe to this event to receive information about the success or failure of an executed affordance. Once this event is sent, the affordance stops running and a new affordance is then allowed to start when the user asks for it to occur. Use of this inherited hierarchy allows for the programmer to implement extra return codes for more specific affordances.

VII. Affordance Recognition

Recognition of the appropriate affordances is important to this research. The visual recognition of objects is accomplished using of the Visual Routines within the Tekkotsu framework. Given the visual information associated with an object, affordances have to determine the object's description and location relative to the robot as well as to generate any necessary information regarding the physical constraints the robot imposes on the affordance's action. For example, consider the task of pushing balls of various sizes: small balls should be pushed by dog's the paw, while large balls should be pushed by the dog's chest (the paw just becomes too cumbersome and awkward of a tool to use if the ball is large). At each time step, an "affordance recognizer" generates a list of affordances for each object in a camera image; the user then chooses the appropriate affordance (that is associated with the desired object) to perform. In theory, a robot should be able to determine the most appropriate affordance through trial and error—currently the affordance is determined through experimentation on the part of the programmer.

VIII. Push Ball Across Line Affordance

For a more detailed example of how complex an affordance can become the push ball across line affordance is discussed. Pushing a ball is an unreliable action since this action depends on well-grounded information such as the size of the ball, the density of the ball, and the type of environment the ball is pushed in. This means, that the execution of the affordance requires close monitoring in order for the action to be successful, but even with close monitoring this affordance can still fail. This discussion will consider the case when the ball is pushed across the line using the dog's chest.

The architecture of this affordance may be described by the following four primary state nodes: the Wait for Image node, the Verify Ball node, the Verify Line node, and the Push Ball node. Both the Verify Ball node and the Verify Line node have extra nodes attached to them to make sure the correct information of the ball and the line are obtained. The first node in the state machine is the Wait for Image node; its sole task is to wait for a new camera image. Once a new camera image is obtained, the state machine transitions to the Verify Ball node which checks the camera image to see if the ball is within sight of the camera. If the ball is not found, the state machine transitions to the Move Head node, which changes the viewing direction of the head. After this action completes, the state machine transitions back to the Wait for Image node. This cycle continues until the ball is found or until a certain number of attempts have been made at trying to locate the ball. If the ball is not found, an affordance completion event is sent, ending the execution of this affordance, and a return code of type Lost Ball is returned to the user. Otherwise, information concerning the ball (for example, its bounding circle and center) is calculated and the state machine transitions to the Verify Line node. Note: if the ball somehow moves too far away from the chest vertically or horizontally, the Verify Ball node will end the execution of the Affordance and send an affordance completion event with a return code of type Lost Ball back to the user.

The Verify Line node works in very much the same way as the Verify Ball node. If the current image is not detected, a state transition occurs and a Move Head node moves the dog's head to a different viewing position. Once that action completes, another state transition occurs and the current node of execution is the Wait for Image node; after a new camera image is obtained, the state machine transitions to the Verify Line node. This cycle repeats until either a line is found or a certain number of failed head motions occur. If the line is not found, a Lost Line return code is returned to the user and the affordance stops its execution. Otherwise, information concerning the line is calculated and corrections are made to the heading just in case the dog's pushing direction is a bit off from the line; then, the state machine transitions to the Push Ball node. This node just pushes the ball a small distance, and once this action completes the state machine transitions back to the initial Wait for Image node. The success completion condition occurs at the Verify Line node. If the ball is position ahead of the line an affordance completion event with the success return code is sent to the user.

IX. Discussion

The application of affordances to mobile robots allows for an intuitive approach in binding objects and the robot through the multiple actions the robot can perform on the objects. Since the difficult low-level problems related to joint constraints, positional constraints, and low-level vision problems are removed, a framework based on this approach allows the programmer to deal with the high-level issues of intelligent action planning. Additionally, affordances provide an alternative view of objects by allowing users to consider the utilities objects offer the robot, as opposed to considering information about raw object itself.

There have been some other interesting studies in applying affordances in robotics. Murphy conducted studies in applying affordances to a mobile robot as method for direct perception of objects. His four step methodology consists of the following: defining the task to see if affordances are suitable for the task, defining the environment and objects in consideration for the robot, recognizing the potential affordances the environment and its objects allow, implementing the affordances. The three tasks he found affordances worthy for are fine positioning, path following, and picking up the trash. His conclusions include that affordances should be carefully designed so that the system using the affordances does not become brittle. While Murphy also designs specific affordances, others have attempted "learn" affordances of various objects. For example, Stoychev describes a behavior-based approach where a robot chooses random sequences of "exploratory" behaviors to autonomously learn affordances for various objects.

X. Future Work

The previous discussion introduces the overall goal of the proposed method. Based on the topology and location of an object, a robot should be able to formulate a set of affordance for the object. However, a drawback of the approach is if other objects are near the object at hand: this would cause a rich supply of potentially perceived affordances to rapidly accumulate in which different objects interact in order to complete a task. For example, the affordances which allow object A to be pushed next object B grows as $O(N^2)$. To alleviate this issue, the following solution is proposed: introduce two levels of affordance recognition, goal oriented affordance and undirected affordance. Undirected affordances allow for the robot to randomly explore the environment using only general afforadances such Push Ball or Explore Line. Goal oriented affordances allow the completion of certain tasks (the goal is provided by the programmer) by returning a list of perceived affordances whose actions will allow the completion of the goal.

The current affordance recognizer recognizes affordances for those objects directly visible to dog's camera. The next version of this system will store the affordances in the working memory of the dog through use of two mechanisms: the local map and the global map. The local map is obtained after taking a camera image using the dog's camera; this approach is 'local' because the information obtained (orientation and position of the various objects) from the camera image is stored relative to the dog's current orientation and position. Thus, the dog's position can be considered a fixed point in terms of the local map. In the global map, however, all objects are stored in some

positional manner extracted from a series of local maps and the dog's position is not considered a fixed point; thus, the dog's position information is also listed and stored in the global map. Each object in this global map will then specify the associated affordances that can be performed on them given information about the dog.

Finally, this approach should eventually extend to allow a mechanism for different affordances to interact with each other. For example, if a ball affords to be kicked and a goal affords an object to be pushed into it, some mechanism should be designed to allow for the two affordances to interact allowing a ball to be kicked into the goal.

XI. Conclusion

The use of affordances to recognize utilities associated with an object and its environment to allow the user to determine an action to perform, allows the user to focus on the high-level details of what he wants the robot to perform. By allowing programmers and users to use this system, we hope to allow mobile robot programmers to more quickly focus on the actual tasks they want completed.