Manipulation of Objects using an AIBO

Somchaya Liemhetcharat
Advisor: Professor David S. Touretzky

Senior Research Thesis
School of Computer Science, Carnegie Mellon University
May 2nd, 2007

Abstract

Present techniques for manipulation with an AIBO involve specialized routines written for specific object shapes and domains, such as ball manipulation in RoboSoccer, where fixed sets of key-frames or fixed motion sequences have often been used, which the AIBO performs without feedback, similar to acting out a pre-defined script. This approach is difficult to generalize, and thus the AIBO is unable to adapt to new situations.

This research focuses on general methods that allow an AIBO to manipulate classes of objects, such as boxes, spheres, and cylinders. The manipulation techniques are as general as possible, so that the AIBO is able to act on a variety of objects. Visual feedback is used to guide the AIBO as it performs the desired action. This allows the AIBO to adapt to changing circumstances and subtle differences in the objects being manipulated. In this paper, we first describe a technique we use to create a coherent model of a box object by composing various camera images, so as to allow the AIBO to have some understanding of the object. Next, we describe an approach technique that allows the AIBO to approach an object from any direction, while maintaining visual contact with the object and keeping a safe distance from it. Finally, we describe a general manipulation technique that uses the AIBO's body to push the object, such that the AIBO can effectively manipulate the object's position and orientation.

CONTENTS

I. INTRODUCTION

The Sony AIBO robot is built like a dog, having four limbs, a head and a tail. Typically, all four limbs are used in locomotion, which makes it difficult for the AIBO to manipulate objects with its limbs. The head has three degrees of freedom, which allows the AIBO to have a clear view of the world around it. The limbs and head of the AIBO allow it to manipulate objects, but present techniques involve specialized routines written for specific object shapes and domains, such as ball manipulation in RoboSoccer. These techniques often involve a fixed set of key-frames or fixed motion sequences, which the AIBO performs without feedback, much like acting out a pre-defined script [7]. While these techniques allow the AIBO to perform very well in domains such as RoboSoccer, it is difficult to generalize these manipulation techniques to other objects or domains. Furthermore, using such techniques, the AIBO does not have an internal model of the object it is manipulating other than the object's position. Other attributes of the object are irrelevant since the motion sequences do not make use of such information. However, without such information, it is difficult for the AIBO to have an understanding of the world around it, and plan meaningful actions.

The goal of this project is to create techniques that allow an AIBO to create internal models of objects, and then manipulate the objects effectively using the models. These manipulation techniques are designed to be as general as possible, so as to allow the AIBO to act on classes of objects, such as boxes, spheres, and cylinders. Our techniques are implemented within the Tekkotsu framework for AIBO development [1] [2], which allows for iconic (pixel-based) and symbolic (shaped-based) representations of visual objects in its DualCoding engine [3] [5]. In order to create internal models of objects, the AIBO composes various shapes into a coherent whole, and infers attributes of the objects such as length, width, and orientation. After a model of an object is created, the AIBO can then perform the manipulation techniques developed by this project, using visual feedback to guide its actions and adapt to changing circumstances. As such, these techniques are not limited to specific objects or domains, and can be generalized to classes of objects.

II. OBJECTS IN THE AIBO'S MIND

A. Definition of an Object

In the Tekkotsu framework, visual objects can be differentiated into two kinds of representations: iconic (pixel-based) and symbolic (shape-based) representations using its DualCoding engine. Simple shapes defined in the present framework, such as lines, ellipses and blobs, provide valuable information such as length and orientation, but are generally symmetrical about one or more axes, which lead to inconsistencies referring to the left or right of a shape, when the shape is viewed from more than one perspective (see Figure 1). Also, shapes are assumed to lie in the ground plane, which allows an easy projection from the camera frame to the world frame [3]. However, objects in the world are three-dimensional, and knowledge of all three dimensions is critical when building an accurate model of an object. Three dimensional objects that are presently defined in the Tekkotsu framework, such as spheres [3] and pyramids [4], do not provide information about orientation or are not refined enough for general use.

In order to overcome these limitations, we have defined a new type of shape, called a Target. A Target shape can be used to represent any three-dimensional object in the world, and has attributes of length, width, height, and orientation. A Target shape is named this way because such objects can be manipulated by the AIBO, and thus become a target for the manipulation techniques developed in this project.

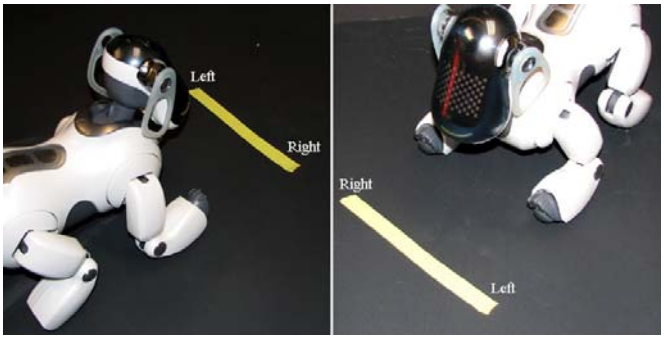

Figure 1: Symmetry of simple shapes

B. Cognition of an Object

An object has three-dimensions, which makes it difficult to create an accurate model using only a simple shape, even with assumptions on attributes such as the height of the object. As such, the method we developed to create a model relies on composing various simple shapes, known *a priori* to be in a certain layout. While this technique cannot be generalized across different classes of objects, such as between boxes and spheres, it can be used to create models of different objects of the same class, such as two boxes with different sizes. Because the goal of this project was the manipulation of objects, and not the modeling of general objects, the method we used relies on some assumptions about the object, such as its height and class. However, the manipulation techniques developed, which presently use these model-creating techniques, do not rely on such assumptions and can be used for other kinds of models created with more advanced techniques.

For this project, we focused on creating a model of a box object, using three lines of different colors, placed on the top of the box in a layout that breaks its symmetry (see

Figure 2). In this way, the model created has a unique orientation, regardless of the relative angle between the object and the AIBO. In order to overcome the limitation that shapes are projected onto the ground plane, we used *a priori* knowledge of the height of the box, in order to project the lines correctly into the world shape space (see Figure 3).

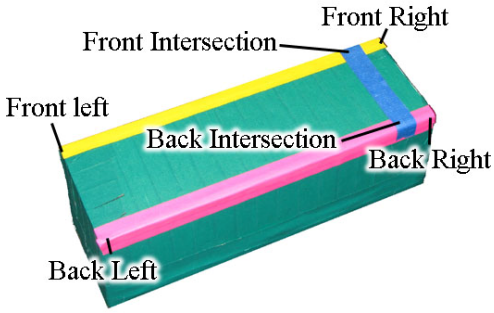

Figure 2: Layout of lines on a box and its feature points

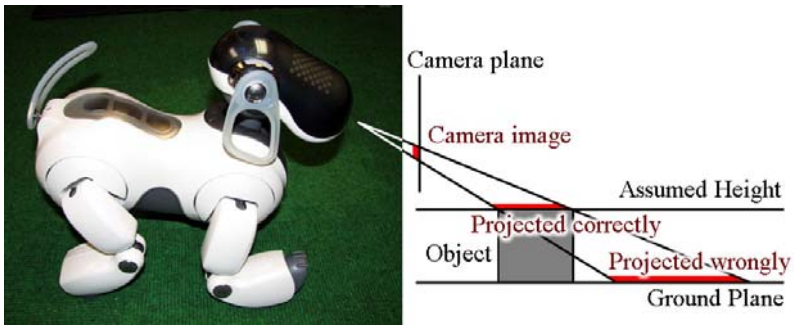

Figure 3: Projection of shapes to 3d-space

C. Creating a model

1) Extracting feature points

The box model can be defined by its six feature points (see Figure 2). To retrieve these feature points from a camera frame, we use line extraction techniques developed by Tekkotsu's DualCoding engine [3] [5]. However, since the AIBO has a narrow field of view, the entire object is seldom within the camera frame. This is especially apparent when the AIBO approaches the object, which is an essential part of manipulation. As a result, we had to develop methods that compose different camera images to retrieve as many feature points as possible.

In the direct case, the entire object is within the camera frame, which occurs when the object is a good distance away from the AIBO. In this case, the end-points of the two horizontal lines are easily extracted, and the other two feature points are retrieved by obtaining the intersection points between the vertical and horizontal lines.

In the general case, only part of the object can be seen within the camera frame. Because the three lines that define the box are of different colors, we can easily distinguish them. The present DualCoding techniques relay information about whether the end of a line is visible in the camera image, and we use this information for our feature points. However, because lines are symmetric, it is difficult to distinguish between the left and right end of the object based on a single line. As a result, while we

are able to distinguish between the front and back lines (since they are of different colors), we may not be able to determine the left and right feature points of a line correctly. As such, in the general case, as well as the direct case above, further analysis is performed on the feature points to distinguish between the left and right feature points, as well as to infer attributes of the model such as length, width, and orientation.

2) Inferring object attributes

The following are known about the box object *a priori*:

- The height of the box is provided by the user. Length and width are not.
- The box is characterized by 3 lines: two horizontal lines arranged length-wise across the top face, and a vertical line arranged width-wise across the top face (see Figure 2).
- The box has 6 feature points: front left, front right, back left, back right, front intersection, and back intersection. The first four points refer to the end-points of the horizontal lines, and the last two points refer to intersections between the vertical line and the two horizontal lines.
- The front intersection and back intersection points are closer to the right end-points of their corresponding lines.
- If an intersection point and one end-point of the corresponding line can be seen in the same camera image, then the end-point must be the right end-point of the line.

Using the *a priori* knowledge given above, it is possible to distinguish between the left and right feature points and infer the attributes of the object. It is important to note that the algorithm described below can be performed on subsets of the feature points, since not all feature points are readily visible in a camera image.

1. The front and back lines of the box are re-created using the 4 feature points.
2. The lengths of the lines are compared, and if one line is substantially longer than the other, the shorter line is discarded as its end-points are probably noisy.
3. The front and back intersection points are used to distinguish between the left and right edges of the line.
    a. If only one end-point is valid (i.e. the end of the line was seen in a camera image), then that end-point is the right end-point of the line (see Figure 4a)
    b. If both end-points are valid (i.e. both ends of the line have been seen), or if both are invalid (i.e. both ends of the line have not been seen), then the right end-point must be the one closest to the intersection point (see Figure 4b)
4. If only one intersection point has been seen, then distinguishing the left and right edges of the other horizontal line can be done by comparing it to the line whose orientation has been determined (see Figure 4c).
5. In cases where both intersection points cannot be seen, the orientation of the box can be determined by looking at the relative positions of the two horizontal lines, since the front and back lines can be differentiated by their colors (see Figure 4d).

6. When only one line can be seen without an intersection point in view, it is not possible to break the symmetry of the line and determine the orientation of the box accurately.
7. After performing the steps above, the length, width, and orientation of the box can be determined by analyzing the position and orientation of the horizontal lines.

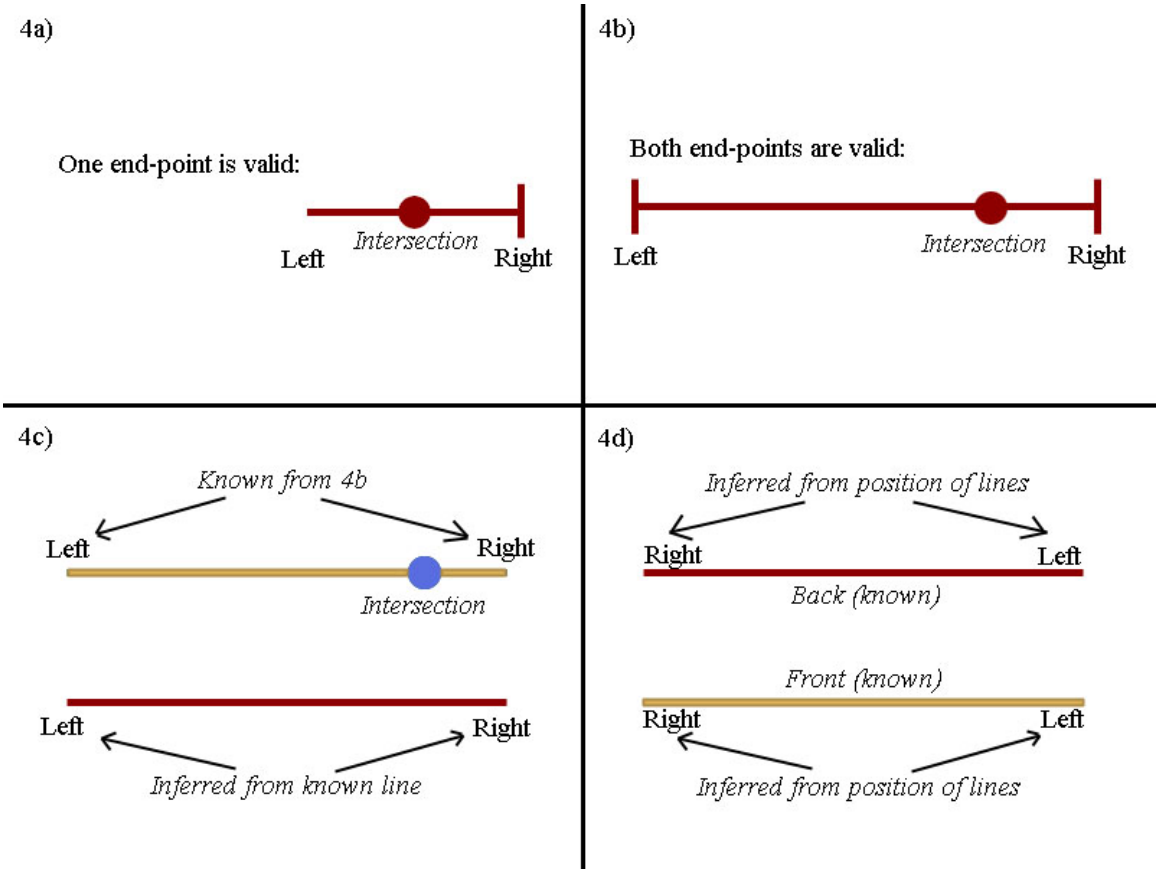

Figure 4: Inferring attributes of the box object

3) Merging models into a coherent whole

After performing the above algorithm to distinguish feature points and infer attributes, the internal model of the object created by the AIBO may not be complete, since the AIBO may not have the entire object within its camera frame. Thus, it is necessary to compose multiple camera images and merge the internal models into a coherent whole.

Firstly, the AIBO must be able to decide on the next view-point to look at in order to gain a new perspective on the object. To do so, the AIBO determines whether its present model of the object contains valid end-points of the horizontal lines (i.e. whether the ends of the lines have been seen). If the feature points are missing or incomplete (when the end of a line has not been seen in any camera image), the AIBO turns its head and focuses on that feature point. The AIBO then repeats the model extraction and inference phase, to create a separate model based on the new camera image. Following that, it performs a merge to combine this newly-constructed model with its present

internal model, updating the model's feature points and attributes. In this way, the AIBO improves its model with each step and eventually creates a coherent model of the entire object (see Figure 5).

However, it is not always possible to view all the feature points of an object. For example, when the AIBO is extremely close to the object, which occurs during manipulation, it may not be possible to view the closest horizontal line, due to joint constraints of the AIBO's head, as well as shadows cast by the AIBO on the object, which interferes with the color segmentation. Thus, the feature points that the AIBO chooses to search and complete may only be a subset of all the feature points of the object. Since the object's attributes can be inferred from a subset of feature points, doing so is not necessarily detrimental to the AIBO's manipulation of the object.

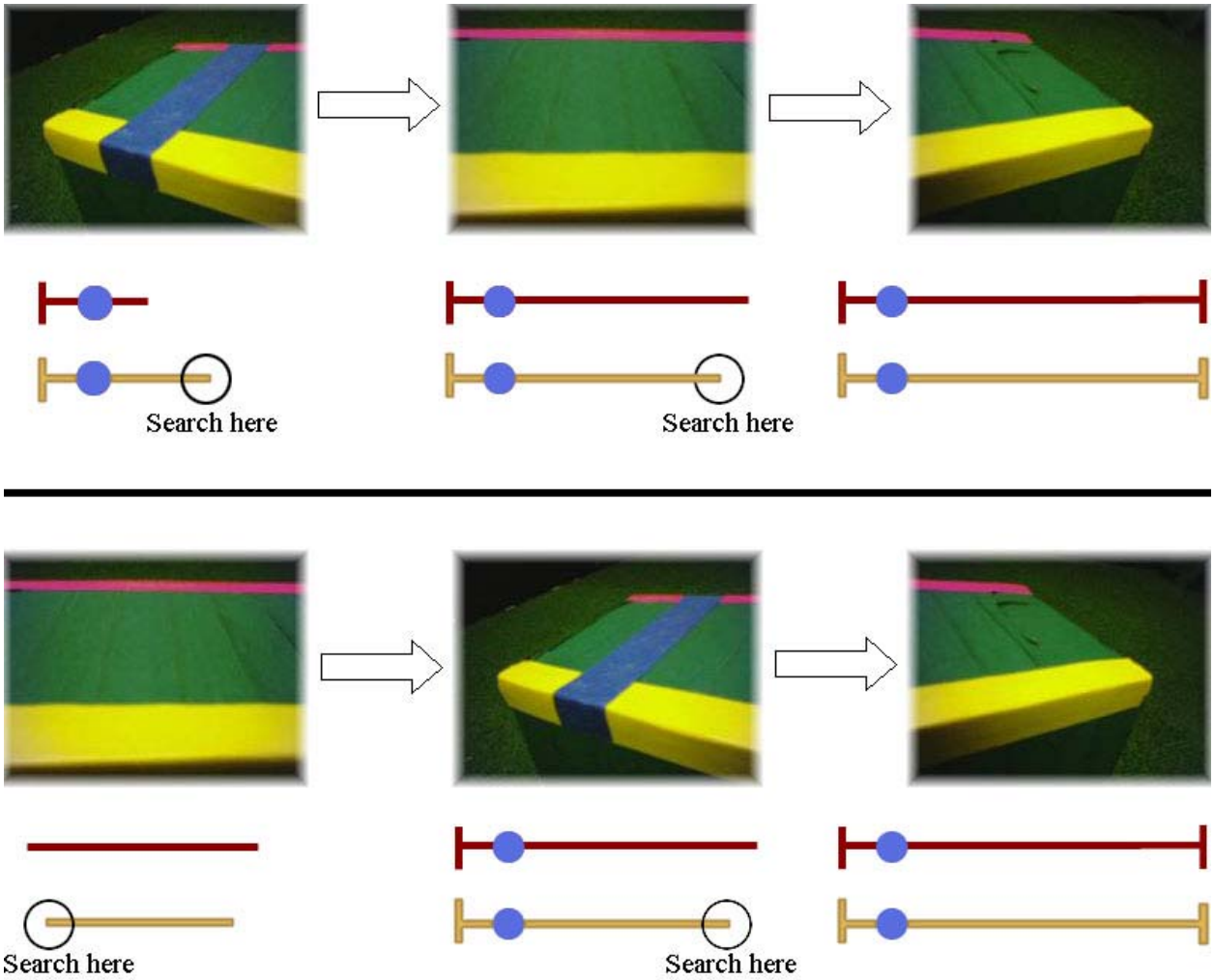

Figure 5: Searching and combining feature points of an object

III. MANIPULATION OF OBJECTS

A. Approaching an object

Before the AIBO can perform any kind of manipulation on an object, it must first be able to approach the object. By using the model creation technique described above, the AIBO is able to create an accurate model of the object it is approaching. However, due to errors in odometry and inconsistencies in the walking, it is unlikely that the AIBO can approach the object accurately if it only builds the model once and does the entire approach. Instead, our technique for approaching the object involves visual feedback, where the AIBO constantly updates the relative position of the object, and makes adjustments in its approach.

Also, another factor taken into consideration for our approach technique is that the AIBO must be able to approach the object from all directions. We decided to develop a manipulation technique that involves pushing rather than grasping and pulling, for reasons described in the next section. Due to these reasons, it is important that the approach must be able to position the AIBO anywhere around the object. Also, being able to approach the object from any direction is more general than a specific approach from a particular direction.

1) Defining a desired location

In the present Tekkotsu framework, there are three different forms of spatial representation: camera-centric (position in the camera frame), egocentric (relative position to the robot), and world-centric (fixed position in the world) (see Figure 6) [2]. For the purposes of the approach as well as manipulation techniques we developed, we introduce a fourth kind of spatial representation: object-centric. An object-centric representation is similar to the egocentric one, except that the object is treated to be at the origin, and the axes in this representation are dependent on the object's orientation (see Figure 6).

The reason why we do this is two-fold. Firstly, defining a desired position relative to the object is intuitive to the user. For example, the AIBO might be required to approach the back of the object. In this case, it is more intuitive to define the desired position to be some (-x, 0) from the object's centroid (x points along the orientation of the object, similar to the AIBO's egocentric axes), instead of coordinates in the egocentric or world-centric representations. Secondly, using egocentric coordinates would not be appropriate since the coordinates would change each time the AIBO moved, and using world-centric coordinates would not allow the AIBO to adapt to changes in the object's position or orientation. An object-centric representation allows the AIBO to reach its desired position regardless of where the object is in the world, and allows the AIBO to adapt to changes in the object's position and orientation as it performs the approach.

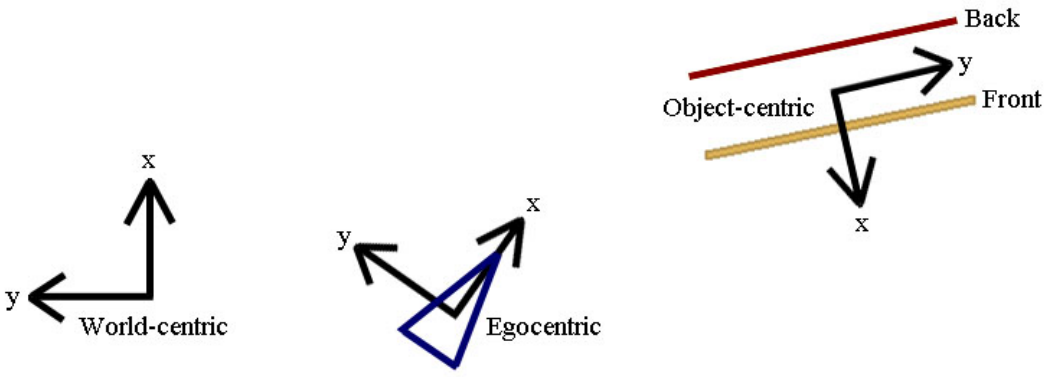

Figure 6: Coordinate representations

2) Creating waypoints

After the desired location represented in object-centric coordinates has been defined, the AIBO can begin its approach. Since the approach may require the AIBO to go around the object, the approach technique developed involves a form of circling around the object. This allows the AIBO to keep visual contact with the object, while maintaining a safe distance away from it as the AIBO goes around the object. This prevents the AIBO from coming into physical contact with the object during the approach. In order to do this effectively, we use the Waypoint system developed in Tekkotsu. The Waypoint system allows us to define waypoints for the AIBO to walk, which the AIBO can then perform in sequence. The steps taken to generate the waypoints are as follows:

1. The AIBO compares its present bearing to the object against the desired position's bearing to the object to obtain a bearing error θ (see Figure 7a). If the two bearings are radically different (i.e. θ is large), then the circling must be performed.
2. In order to perform the circling, the AIBO selects a point along the approach circle, at an angle ϕ closer to the desired position (see Figure 7b). Upon reaching this waypoint, it repeats the process until θ is small enough (see Figure 7c), at which point it proceeds to the next step.
3. Once the two bearings are close (i.e. θ is small), the AIBO can perform a direct approach to the desired location, without having to maintain its distance away from the object (see Figure 7d). After doing so, the AIBO will be at the desired position around the object.

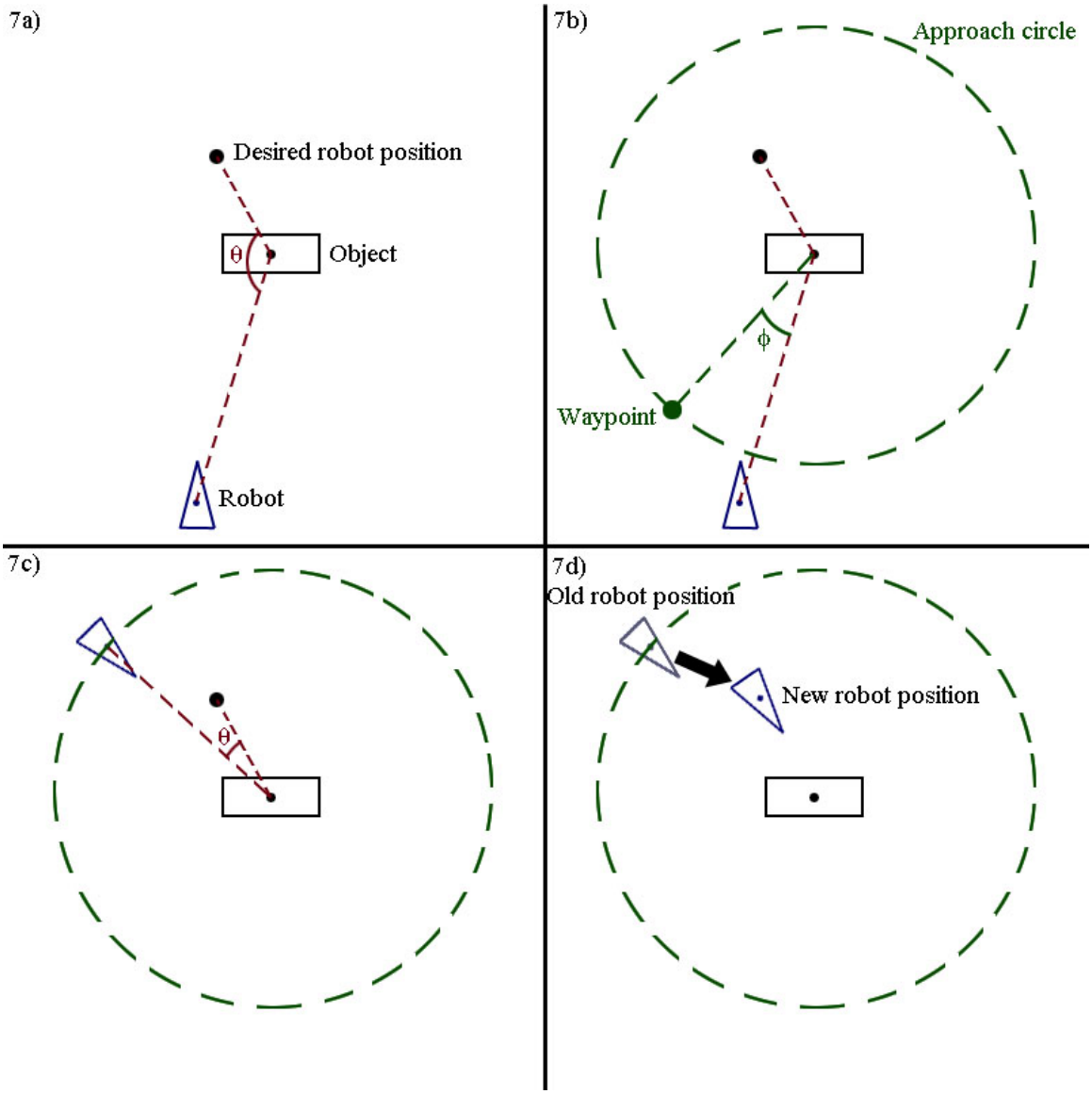

Figure 7: Creating waypoints

By performing the above steps, the AIBO is able to construct a list of waypoints that will go around the object if required, before directly approaching the desired location (see Figure 8). However, due to errors in walking and odometry, it is unlikely that the AIBO will arrive at the desired location if the AIBO walked from waypoint to waypoint without feedback. As such, during each phase of the approach, our technique allows the AIBO to walk only a short distance before it updates its model of the object, and uses the updated model to refresh its waypoints. In this way, the approach performed by the AIBO is more tolerant to errors and can adapt to changes in the object's position and orientation during the approach.

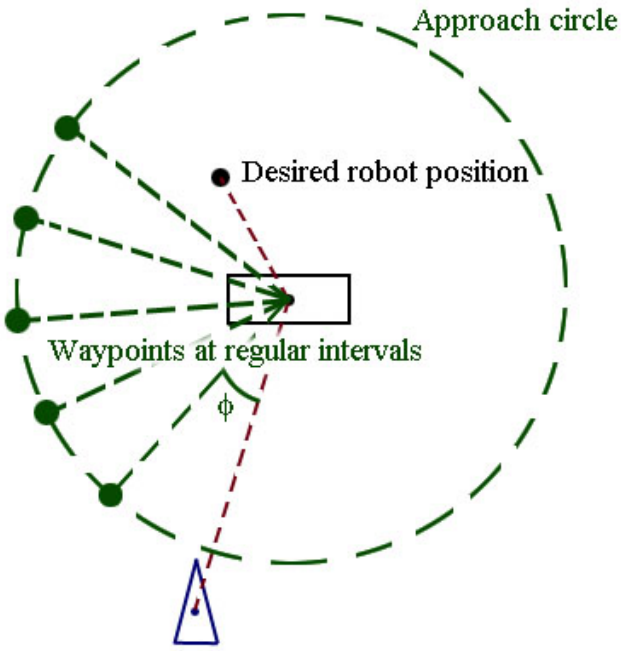

Figure 8: Waypoints selected to approach an object

B. Pushing an object

After the AIBO reaches the desired position around the object, direct manipulation can be performed. The manipulation technique developed assumes that the AIBO starts at the desired location around the object, as defined above. In order to manipulate the object, we decided to use a pushing force generated by the AIBO as it walks into the object. This is mainly because manipulation involves translation and rotation, and since the AIBO does not have limbs that allow it to grasp the object from the front and pull it, it is easier to push it instead. While the AIBO's head and mouth can be used to grasp objects, it is limited by the short length of the neck, which prevents it from grasping large objects. Also, grasping techniques are not easy to generalize within a class of objects, since a technique to grasp would depend on the relative size of the object to the AIBO; an AIBO may be able to grasp a small object, but not a large one with the same technique.

We had a variety of choices in deciding how the AIBO could push an object. Several techniques immediately came to mind, such as pushing it with a limb or head, similar to how object manipulation is sometimes performed in Robosoccer. However, because the limbs of the AIBO are used for locomotion, pushing an object with its limbs would require the AIBO to get very close to the object, stop walking and balance on three legs, and push the object with the other leg. Since the AIBO's legs are considerably short, manipulation in this way will not cause much translation or rotation on the object. We ruled out using the head for manipulation due to similar reasons, and we also wanted to leave the head available for visual feedback.

As such, we experimented with other methods of manipulation, and settled on using the AIBO's body to push an object as it walks. By doing so, the AIBO is able to

manipulate the object for a distance, without being limited by the length of its limbs or neck. Also, while manipulation in this manner is less precise than pushing a specific part of the object with a limb, we found that coarse manipulation is sufficient to position an object in a general area in the world, at a greater speed than pushing with a limb.

1) Defining a push

A push manipulation performed by the AIBO consists of two components: an initial starting position and a direction (see Figure 9). Both components are defined in the object-centric representation. The initial starting position is taken to be the desired position around the target of the approach phase, as we assume the approach is carried out prior to manipulation. The direction is the angle in which the AIBO should face and walk in order to push the object with its body. By varying the initial starting position and direction, we were able to vary the manipulation of the object's position and orientation (see Figure 10). As such, a combination of such motions will allow the AIBO to position an object around the world.

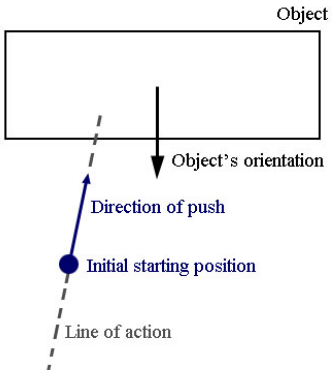

Figure 9: Defining a push

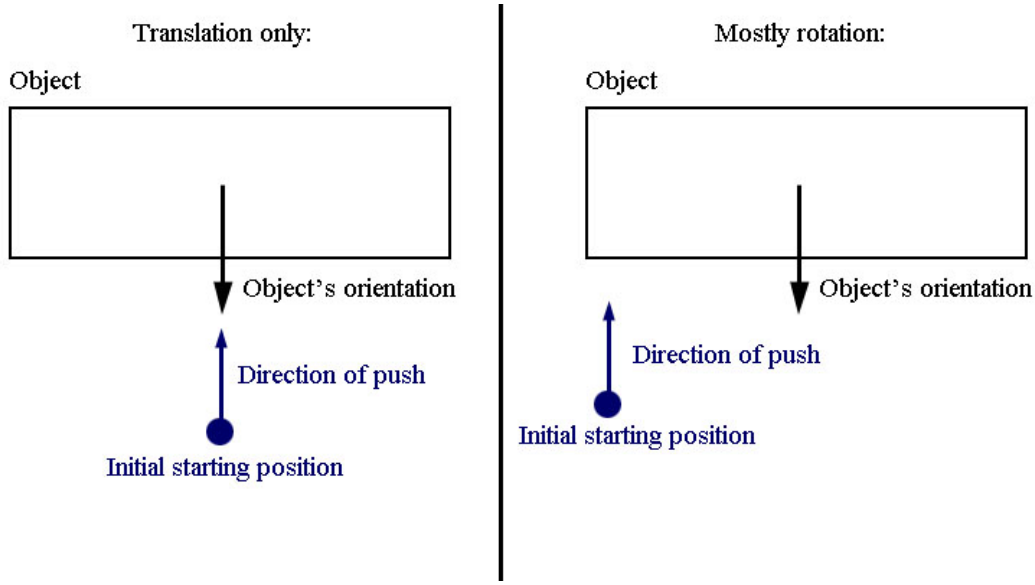

Figure 10: Different manipulations performed on an object

2) Performing the push

In order to perform the push, the AIBO uses the Waypoint system, in a similar fashion to the approach phase. In this way, a queue of actions can be created and executed at once, before updating its internal model of the object.

1. Firstly, the AIBO compares its orientation to the direction of the push. If the two directions are different, a waypoint is created such that the AIBO turns in place until its orientation matches the push direction (see Figure 11a).
2. The AIBO then checks if it is on the line of action (see Figure 9). If it is not, then a waypoint is created such that the AIBO moves sideways until it lies on the line on action (see Figure 11b). In this way, its orientation is undisturbed and it is ready to perform the final manipulation action.
3. The final manipulation action involves the AIBO simply walking forward. Since the previous two steps have ensured that the AIBO lies on the line of action and that it is facing the push direction, a forward walk causes the body to stay within the line of action and generate a pushing force on the object (see Figure 11c).

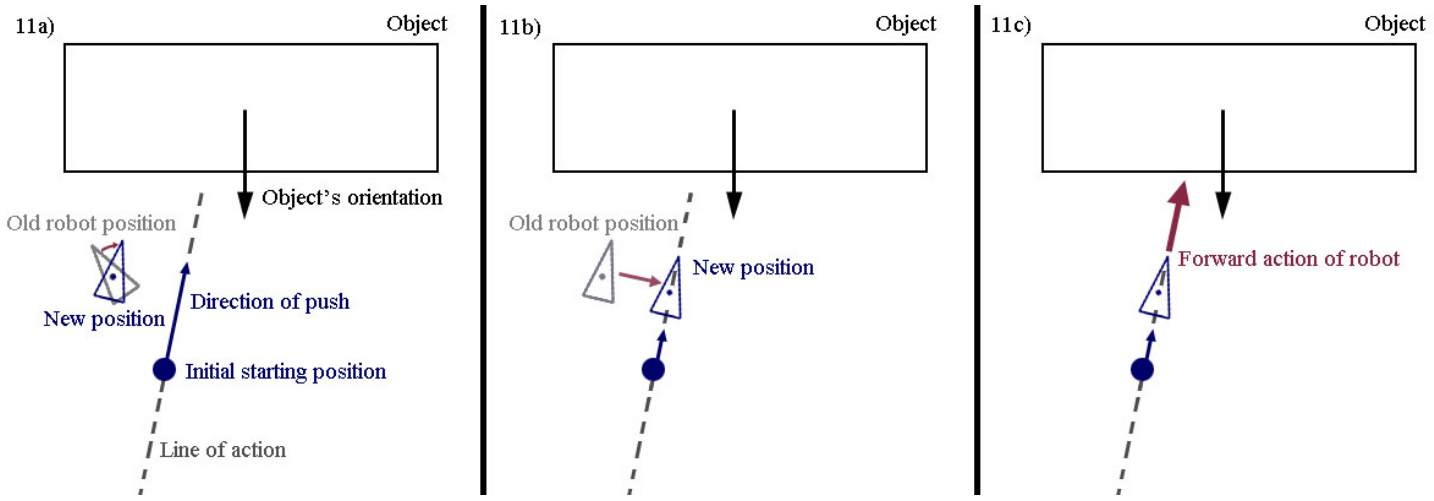

Figure 11: Pushing an object

After performing these steps, the AIBO updates its model of the object, and adjusts its position accordingly before pushing again. In this fashion, the AIBO maintains its line of action on the object as the object moves, and hence causes a consistent change in the position and orientation of the object.

IV. USING THE PILOT TO MANIPULATE AN OBJECT

A. Introduction to the Pilot

In the Tekkotsu framework, there exists a Pilot class that handles the control of the AIBO to perform common tasks, such as approaching or searching for a shape [2]. Because the manipulation of objects is likely to be a common task of the AIBO as well, we decided to implement our techniques within the Pilot class. In this way, it is easier for the user to perform manipulation tasks on the AIBO by generating a Pilot request, compared to using a separate behavior that deals with manipulation.

B. GotoTarget

The GotoTarget Request performs the approach techniques described in the earlier sections. In order to execute the GotoTarget request, the following fields must be filled in by the user:

Variables	**Description**
targetShape	The target object that the AIBO should approach.
positionRelativeToTarget*	The desired position around the object.
safeDistanceAroundTarget*	The distance at which the AIBO keeps away from the object as it circles.
subtendAngle*	The angle at which the AIBO subtends the circle (see Figure 7b).
approachAngle*	The angle at which the AIBO can directly approach the desired position (see Figure 7c).
mapBuilderRequestFn	This function generates a MapBuilder request, allowing the AIBO to find the target object again after movement.
buildTargetMapBuilderRequestFn*	This function generates a MapBuilder request, which extracts a target object at a given point in space.
buildTargetParamsFn*	This function returns the feature points of the model that should be searched for.

* New fields added into the Pilot request

The MapBuilder request functions above are used by the Pilot when the model of the object should be rebuilt. The Pilot executes these MapBuilder requests through the MapBuilder, a class that handles building maps of the world [2]. By allowing a general function to be used, the user can specify the kind of search to locate the target object, as well as the class and colors of the object to be found.

When executing the GotoTarget request, the Pilot makes use of the above parameters and performs the approach techniques described in the earlier sections.

C. PushTarget

The PushTarget Request performs the manipulation techniques described in the earlier sections. In order to execute the PushTarget request, the following fields must be filled in by the user:

Variables	**Description**
targetShape	The target object that the AIBO should approach.
positionRelativeToTarget*	The desired position around the object.
angleToPushTarget*	The direction to push the object (see Figure 9)
mapBuilderRequestFn	This function generates a MapBuilder request, allowing the AIBO to find the target object again after movement.
buildTargetMapBuilderRequestFn*	This function generates a MapBuilder request, which extracts a target object at a given point in space.
buildTargetParamsFn*	This function returns the feature points of the model that should be searched for.

* New fields added into the Pilot request

The parameters of a PushTarget request are similar to the GotoTarget request, with the exception of the parameter *angleToPushTarget*. When executing the PushTarget request, the Pilot uses the parameters above to perform the manipulation techniques described in the earlier sections.

V. DISCUSSION

The manipulation technique described in this paper allows the AIBO to perform more than an observer role, and actually make changes to the world around it. The modeling of objects allows the AIBO to go one step beyond understanding simple shapes, and have a more complete view of the objects in its world. While the techniques described to create the model rely on a number of assumptions, better vision algorithms can be employed to extract the model in the future. As they do not depend on the assumptions used in extracting the model, the same manipulation techniques can still be performed. Also, the techniques developed here are not constrained to a particular object, and can be extended to classes of objects. This allows the AIBO to manipulate a large range of objects with the infrastructure developed here, and not be limited to specific objects or domains.

In practice, the manipulation technique developed in this paper perform very well. However, problems occur during manipulation when the model of the object is not created accurately. This happens occasionally when the AIBO is unable to resolve the orientation of the object, either because of a lack of information, or because of noise in the image. An error in model creation can cause the AIBO to detect an object with a flipped orientation, which in turn causes the AIBO to turn around. Such errors can be detected when the AIBO rotates a large amount, and checks can be written in code to detect this and repeat the model creation process. With an improved model-creation technique, such errors that occur can be minimized further. Also, the manipulation fails when the AIBO is not able to locate the object after moving. In the present framework, the MapBuilder request provided by the user is required to locate the object after movement, but if the MapBuilder fails to find the object, the Pilot request to manipulate it fails as well.

VI. FUTURE WORK

Better vision algorithms that rely on fewer or no assumptions can be developed to extract more information from a single camera image. In this way, the AIBO will be able to create accurate models of objects in the world more easily and interact with them.

Also, the present techniques do not keep track of the AIBO in the world space; this is left to the user to maintain. Because of this, the AIBO is able to manipulate the object, but it is not able to keep in mind where it wants the object to end up and react accordingly. An updated manipulation technique would involve looking at markers and localizing, as well as choosing the more relevant manipulation to perform on the object. By having a world model in its mind, the AIBO will also be less susceptible to failures in model creation, since it can fall back on previous knowledge of the world.

ACKNOWLEDGEMENTS

I would like to thank Professor David S. Touretzky for his invaluable help in mentoring my senior research thesis, as well as the brainstorming sessions we had to solve the many issues that came up. I would also like to thank Ethan Tira-Thompson for his help regarding the Tekkotsu framework, and his useful suggestions throughout the development of this project.

REFERENCES

[1] E. J. Tira-Thompson, "Tekkotsu, A rapid development framework for robotics," Master's Thesis, Robotics Institute, Carnegie Mellon University, May 2004.

[2] (2007) Tekkotsu website. [Online]. Available: http://www.tekkotsu.org/

[3] D. S. Touretzky, N. S. Halelamien, E. J. Tira-Thompson, J. J. Wales and K. Usui, "Dual-coding representations for robot vision in Tekkotsu," *Autonomous Robots*, 22(4): pp. 425-435, 2007.

[4] M. Carson, "Blocks World Vision for the AIBO Robot," Senior Research Thesis, School of Computer Science, Carnegie Mellon University, May 2006.

[5] N. S. Halelamien, "Visual Routines for Spatial Cognition on a Mobile Robot," Senior Research Thesis, School of Computer Science, Carnegie Mellon University, May 2004.

[6] S. Chernova and M. Veloso, "Learning and Using Models of Kicking Motions for Legged Robots," *Proceedings of ICRA-2004*, New Orleans, May 2004.