Creation of a fine controlled action for a robot

Ellie Lin
Supervisor: Dr. Peter Stone
Department of Computer Sciences
University of Texas at Austin
Austin, TX

December 15, 2003

Abstract

A common problem facing roboticists is the creation of *fine controlled* actions for a robot that must be interspersed with a baseline motion. We define a fine controlled action to be one in which small errors can make the difference in the success or failure of the action. A baseline motion is one that is executed repeatedly over time, such as walking straight or remaining idle. We examine the considerations that can affect the success of a fine controlled action that transitions between baseline motions. We introduce a general technique for implementing a fine controlled action that transitions from and to a baseline motion using the example of pushing an elevator button. We implement this technique in the area of robotic soccer. Our results demonstrate that this technique can successfully create fine controlled actions for a robot.

1 Introduction

One of the fundamental challenges in robotics is the implementation of a *fine controlled* action for a robot that is executed from and to a baseline motion. We define a fine controlled action to be one in which small errors can make the difference in the success or failure of the action. One example of such is the pressing of an elevator button. Success in this task leaves little room for error. If the robot's arm contacts any part of the elevator other than the button while trying to press the elevator button, the robot runs the risk of pushing an incorrect button. Also, the success of a robot's fine controlled action is also dependent on the location at which the robot begins its action. In the elevator button example, if the distance between the robot and the elevator is too great, the robot will not make contact with the elevator button. A baseline motion such as walking or standing in place leads into and directly follows the fine controlled action. It is important for the robot to successfully begin the action from and return to the baseline motion.

Transitions between the fine controlled action and the baseline motion are tricky in that one could very well disrupt the operation of the other. Thus, creating a fine controlled action that executes from and to a baseline motion requires careful attention to detail. We explore the necessary considerations for creating a fine controlled action in the elevator button scenario. We then present a general technique for creating a fine controlled action and test it in an application relevant to robotic soccer, namely kicking the ball.

For the purposes of this research, we use the Sony Aibo robot. [1] The Aibo has multiple joints that can be set at certain angles. We specify the fine controlled action with a sequence

of poses, where each pose consists of all the angles of the joints. The Sony Aibo has a PID mechanism that moves the joint angles from one specified pose to the next over a specified time. PID is a control filter that helps the robot reach a pose without overshooting or oscillating. [2] Thus, we specify each action with a sequence of poses and a transition time between consecutive poses. The following table depicts the used joints and joint descriptions for the Sony Aibo robot.

joint	*joint description*
j_1	front right leg joint 1
j_2	front right leg joint 2
j_3	front right leg joint 3
j_4	front left leg joint 1
j_5	front left leg joint 2
j_6	front left leg joint 3
j_7	back right leg joint 1
j_8	back right leg joint 2
j_9	back right leg joint 3
j_{10}	back left leg joint 1
j_{11}	back left leg joint 2
j_{12}	back left leg joint 3
j_{13}	head joint 1
j_{14}	head joint 2
j_{15}	head joint 3
j_{16}	mouth joint

Table 1: Joints used in fine controlled action

2 Pushing the elevator button: a case study

We use a pretend button that is visible to the Aibo (orange) and placed at roughly the height of the Aibo's head when standing. The following picture shows the Aibo standing next to the elevator button.

We use walking as the baseline motion for the robot to transition from and to the button pushing action. The robot must walk to the elevator button, push the elevator button, and then walk away from the elevator button. The robot must not make contact with anyone else besides the elevator button. There are several considerations in this task:
(1) The difficulty of creating the button pushing action that transitions smoothly from all possible joint configurations of the walking baseline motion. It is easier to first create the button pushing action in isolation from the baseline motion and later address the transition.
(2) The negative effect the transition from walking to button pushing could have on the

success of pushing the button. An awkward transition (e.g. one that causes the robot to sway drastically to the side before pushing the button) can cause the robot to miss the elevator button. In this case, the beginning of the button pushing action needs to be modified to enable smoother transitions.
(3) The negative effect the transition from button pushing back to the walk could have on the success of the button pushing action. An awkward transition back to walking (e.g. one that causes the robot to push an incorrect button on the elevator after having already pushed the correct one) could negatively affect the success of the task. In the case of an awkward transition, the end of the button pushing action needs to be modified to enable a smoother transition.

We can now use the considerations from the elevator button problem to generalize a technique for creating fine controlled actions. We will discuss the implementation of the elevator button pushing action as we describe the technique in the following section.

3 The general technique

The technique assumes that there is a mechanism that enables us to position a robot in some pose and access all the joint angle values for the robot in that pose. We define a $Pose = (j_1, \ldots, j_n)$, $j_i \in \Re$, where j represents the positions of the n joints of the robot. The robot has a PID mechanism that moves joints 1 through n from one *Pose* to another over a time interval t. We specify each action as a series of moves $\{Move_1, \ldots, Move_m\}$ where a $Move = (Pose_i, Pose_f, \Delta t)$, and $Move_{jPose_f} = Move_{(j+1)Pose_i}$, $\forall j \in [1, m-1]$. Each t unit for the robots we used represents 8 milliseconds of execution time, so Δt is the amount of time (in 8 millisecond units) it takes to transition from one *Pose* to the next.

The technique is comprised of two steps:
(1) Critical action: Creating the fine controlled action in isolation from the baseline motion.
(2) Integration: Integrating the fine controlled action into the baseline motion.

3.1 Creating the critical action

The first step in creating the fine controlled action involves isolating the action from the baseline motion. We call the *Moves* that comprise this isolated action the *critical action*. To obtain the joint angle values for each *Pose*, we use a tool that captures all the joint angle values of the robot after physically positioning the robot in its desired pose. We first manually position the robot in the ending pose of the critical action and record $j_1, \ldots, j_n$ for that *Pose*. We call this $Pose_{critical_f}$. The ending pose is the *Pose* in which the robot completes its task. In the elevator button pushing task, $Pose_{critical_f}$ is the *Pose* in which the robot makes contact with the elevator button.

We then physically position the robot in the *Pose* from which we want the robot to move to $Pose_{critical_f}$. We called this $Pose_{critical_0}$. We then create a *Move* $m = (Pose_{critical_0}, Pose_{critical_f}, \Delta t)$ and watch the robot execute m via its PID control mechanism. At this point of fine controlled action creation, we are primarily concerned with the path the robot takes from $Pose_{critical_0}$ to $Pose_{critical_f}$. Thus, during this manual process of manipulating the robot's joints and capturing the *Pose*, we select a large Δt for the *Move* that enables us to watch the path from $Pose_{critical_0}$ to $Pose_{critical_f}$. Since each t unit for the robots we used represented 8 milliseconds of execution time, we typically selected Δt to be 64, so this *Move* took approximately 1/2 second to execute.

Above, the figure on the left is $Pose_{critical_0}$, and the figure on the right $Pose_{critical_f}$. Note that the front right leg is lifted in the air to the height of our button. This *Move* is specified by the following table:

	j_1	j_2	j_3	j_4	j_5	j_6	j_7	j_8	j_9	j_{10}	j_{11}	j_{12}	j_{13}	j_{14}	j_{15}	j_{16}	Δt
$Pose_{critical_0}$	-6	-10	-6	-6	-10	-6	-7	-10	146	-7	-10	146	38	89	24	0	
$Pose_{critical_f}$	95	0	0	-6	-10	-6	-7	-10	146	-7	-10	146	38	89	24	0	64

Table 2: Pushing elevator button critical action

If the *Move* does not travel a path that allows the robot to perform the action successfully, we then add an intermediary $Pose_{critical_1}$ between $Pose_{critical_0}$ and $Pose_{critical_f}$, create a sequence of two *Moves* $\{(Pose_{critical_0}, Pose_{critical_1}, \Delta t_i), (Pose_{critical_1}, Pose_{critical_f}, \Delta t_{i+1})\}$, and watch the execution. Again, we abstract away Δt_i and Δt_{i+1}, typically selecting 64. After watching the path for this sequence of *Moves*, we repeat the process if necessary.

We added intermediary $Pose_{critical_1}$ (above left) and $Pose_{critical_2}$ (above right) in between $Pose_{critical_0}$ and $Pose_{critical_f}$. Without these intermediary *Poses*, the robot would make contact with the elevator during the *Move* from $Pose_{critical_0}$ to $Pose_{critical_f}$ before it pushed the button. The added *Moves* are specified by the following table:

	j_1	j_2	j_3	j_4	j_5	j_6	j_7	j_8	j_9	j_{10}	j_{11}	j_{12}	j_{13}	j_{14}	j_{15}	j_{16}	Δt
$Pose_{critical_0}$	-6	-10	-6	-6	-10	-6	-7	-10	146	-7	-10	146	38	89	24	0	
$Pose_{critical_1}$	-6	-10	144	-6	-10	-6	-7	-10	146	-7	-10	146	38	89	24	0	64
$Pose_{critical_2}$	95	-10	144	-6	-10	-6	-7	-10	146	-7	-10	146	38	89	24	0	64
$Pose_{critical_f}$	95	0	0	-6	-10	-6	-7	-10	146	-7	-10	146	38	89	24	0	64

Table 3: Pushing elevator button critical action with intermediary Moves

After we are finally satisfied with the sequence of *Moves* in the *critical action*, we tune the Δt for each *Move*. We would like to execute each *Move*of the *critical action* as quickly as possible. Thus, we reduce Δt for each *Move* individually, stopping if the next decrement disrupts the action.

The following table depicts the *critical action* with tuned Δt.

	j_1	j_2	j_3	j_4	j_5	j_6	j_7	j_8	j_9	j_{10}	j_{11}	j_{12}	j_{13}	j_{14}	j_{15}	j_{16}	Δt
$Pose_{critical_0}$	-6	-10	-6	-6	-10	-6	-7	-10	146	-7	-10	146	38	89	24	0	
$Pose_{critical_1}$	-6	-10	144	-6	-10	-6	-7	-10	146	-7	-10	146	38	89	24	0	32
$Pose_{critical_2}$	95	-10	144	-6	-10	-6	-7	-10	146	-7	-10	146	38	89	24	0	16
$Pose_{critical_f}$	95	0	0	-6	-10	-6	-7	-10	146	-7	-10	146	38	89	24	0	64

Table 4: Pushing elevator button critical action with tuned Δt

3.2 Integrating the critical action into the baseline motion

The second step in creating the fine controlled action involves integrating the *critical action* into the baseline motion. There are two points of integration: (1) the transition from the baseline motion to the fine controlled action, (2) the transition from the fine controlled action to the baseline motion.

3.2.1 The initial action

We first focus on the *Move* $i = (Pose_{initial_0}, Pose_{critical_0}, \Delta t)$, where $Pose_{initial_0} \in \{$*all possible poses of the baseline motion*$\}$. Since i precedes the *critical action*, there may be cases in which i adds unwanted momentum to the *critical action* and disrupts the *critical action*. For example, in the elevator button scenario, the transition from the walk to the beginning of the critical action sometimes caused the robot to move sideways before executing the critical action. When the elevator button pushing action was isolated from the walk, the robot successfully pushed the button 85% of the time (in 20 trials). However, once the action was incorporated into the walk, the robot only successfully pushed the button 60% of the time (in 20 trials). In such cases where i disrupts the *critical action*, we find a $Pose_{initial_1}$, in which $\{(Pose_{initial_0}, Pose_{initial_1}, \Delta t), (Pose_{initial_1}, Pose_{critical_0}, \Delta t)\}$ does not disrupt the *critical action*. We call this the *initial action*.

$Pose_{initial_1}$ is shown above. The added *Move* is specified by the following table:

	j_1	j_2	j_3	j_4	j_5	j_6	j_7	j_8	j_9	j_{10}	j_{11}	j_{12}	j_{13}	j_{14}	j_{15}	j_{16}	Δt
$Pose_{initial_1}$	-5	23	102	-5	23	102	-35	6	75	-35	6	75	0	0	0	0	64
$Pose_{critical_0}$	-6	-10	-6	-6	-10	-6	-7	-10	146	-7	-10	146	38	89	24	0	32
$Pose_{critical_1}$	-6	-10	144	-6	-10	-6	-7	-10	146	-7	-10	146	38	89	24	0	32
$Pose_{critical_2}$	95	-10	144	-6	-10	-6	-7	-10	146	-7	-10	146	38	89	24	0	16
$Pose_{critical_f}$	95	0	0	-6	-10	-6	-7	-10	146	-7	-10	146	38	89	24	0	64

Table 5: Pushing elevator button initial action and critical action

Oftentimes, the $Pose_{initial_1}$ we use mirrors the *idle* position of the walk. The *idle* position of the walk is the *Pose* the robot assumes when walking with 0 velocity. Thus, it effectively causes the robot to stop before performing the critical action. We then add the *Move*

$(Pose_{initial_1}, Pose_{critical_0}, \Delta t)$, abstracting away the Δt, to the moves of the *critical action* and watch the path of execution.

As with the creation of the *critical action*, we then add intermediary *Poses* until we were satisfied with the sequence of *Moves* from $Pose_{initial_0}$ to $Pose_{critical_0}$. We then fine-tune the Δt for the added *Moves*.

3.2.2 The final action

We finally focus on the *Move* $f = (Pose_{critical_f}, Pose_{final_f}, \Delta t)$, where $Pose_{final_f}$ is the first position of the baseline motion after the robot completes its fine controlled action. For the walk, $Pose_{final_f}$ is the *idle* position of the walk. At the end the elevator button pushing action, the robot assumes $Pose_{final_f}$ before continuing the walk. Due to constraints of the movement module we used, this transition to $Pose_{final_f}$ takes 1 unit of t (8 milliseconds). [3] Thus, we consider the last *Move* of the action, f, to be $(Pose_{critical_f}, Pose_{final_f}, 1)$. Since f follows the *critical action*, there may be cases in which f causes unwanted side-effects (e.g. the hindering of the robot's ability to resume walking). In the elevator button scenario, f caused the robot to make contact with the elevator after it had already pushed the elevator button. Since the robot's task required that it not make contact with anything other than the elevator button, this resulted in the unsuccessful completion of the robot's task.

In cases where f causes undesirable side-effects, we find a $Pose_{final_0}$, in which $\{(Pose_{critical_f}, Pose_{final_0}, \Delta t), (Pose_{final_0}, Pose_{final_f}, \Delta t)\}$ does not cause unwanted side-effects. We call this the *final action*. As with the *critical action* and the *initial action*, we add intermediary *Poses* until we are satisfied with the sequence of *Moves* from $Pose_{critical_f}$ to $Pose_{final_f}$. We then fine-tune the Δt for the *final action*.

While transitioning from the button pushing action to the walk, for 80% of the time (out of 20 trials), the robot would make contact with the elevator after having already pushed the button. After adding $Pose_{final_0}$ (shown below) to the end of the *critical action*, the robot no longer exhibited the unwanted side-effect.

The sequence of *Moves* constituting the *initial action*, *critical action*, and *final action* make up the fine controlled action. The following table depicts the complete elevator button pushing action.

	j_1	j_2	j_3	j_4	j_5	j_6	j_7	j_8	j_9	j_{10}	j_{11}	j_{12}	j_{13}	j_{14}	j_{15}	j_{16}	Δt
$Pose_{initial_1}$	-5	23	102	-5	23	102	-35	6	75	-35	6	75	0	0	0	0	64
$Pose_{critical_0}$	-6	-10	-6	-6	-10	-6	-7	-10	146	-7	-10	146	38	89	24	0	32
$Pose_{critical_1}$	-6	-10	144	-6	-10	-6	-7	-10	146	-7	-10	146	38	89	24	0	32
$Pose_{critical_2}$	95	-10	144	-6	-10	-6	-7	-10	146	-7	-10	146	38	89	24	0	16
$Pose_{critical_f}$	95	0	0	-6	-10	-6	-7	-10	146	-7	-10	146	38	89	24	0	64
$Pose_{final_0}$	95	-10	144	-6	-10	-6	-7	-10	146	-7	-10	146	38	89	24	0	64

Table 6: Pushing elevator button initial action, critical action, and final action

By using this sequence of moves defined by the *Poses* in the above table, the Aibo successfully walks up to a small button, pushes the button, and walks away from the button. In the experiment, the Aibo has an 85% success rate (in 20 trials).

3.3 Implementation of the technique

This technique was first developed while creating a legged-league RoboCup team at the University of Texas at Austin in spring 2003. [3] The RoboCup is an international initiative to foster improvements in technology while using the game of soccer as a testbed. [4] We used this technique to create kicks for the Sony Aibo. The baseline motion that we used for the Sony Aibo was a walk on four legs.

Different situations call for different kicks. In this section, we detail the creation of two of the kicks using the technique presented in Section 2 and summarize three additional kicks created by the same method.

3.3.1 Chest push kick

In the field of robosoccer, it is often useful for the robot to kick the ball quickly and return to the walking motion quickly. The chest push kick was designed to address this issue. The chest push kick utilizes the robot's chest to propel the ball forward while the robot remains in a standing position. Since the robot's chest can contact the ball without drastically adjusting the robot's body height, the robot can quickly transition into and out of the kick from and to the walk.

The above images show the poses of the isolated kick. The following table shows the critical action for the chest push kick after tuning for Δt.

	j_1	j_2	j_3	j_4	j_5	j_6	j_7	j_8	j_9	j_{10}	j_{11}	j_{12}	j_{13}	j_{14}	j_{15}	j_{16}	Δt
$Pose_{critical_0}$	-12	30	91	-12	30	91	-70	45	104	-70	45	104	0	0	0	0	
$Pose_{critical_1}$	-120	90	145	-120	90	145	120	25	125	120	25	125	0	0	0	0	1
$Pose_{critical_f}$	-12	30	91	-12	30	91	-30	6	104	-30	6	104	0	0	0	0	64

Table 7: Chest push kick critical action

We then integrated the walk with the kick. Testing revealed that the robot successfully kicked the ball 55% of the time and fell over after 55% of the successful kicks (in 20 trials). Since $(Pose_{initial_0}, Pose_{critical_0}, \Delta t)$ added unwanted momentum to the critical action (causing the robot to fall over after the kick), we created an initial action to precede the critical action. $\{(Pose_{initial_0}, Pose_{initial_1}, 64), (Pose_{initial_1}, Pose_{critical_0}, 64)\}$ does not lend unwanted momentum to the critical action. Testing revealed that the robot now successfully kicked the ball 100% of the time (in 20 trials).

The above picture shows the added $Pose_{initial_1}$. The following table shows the initial action with the critical action after tuning for Δt.

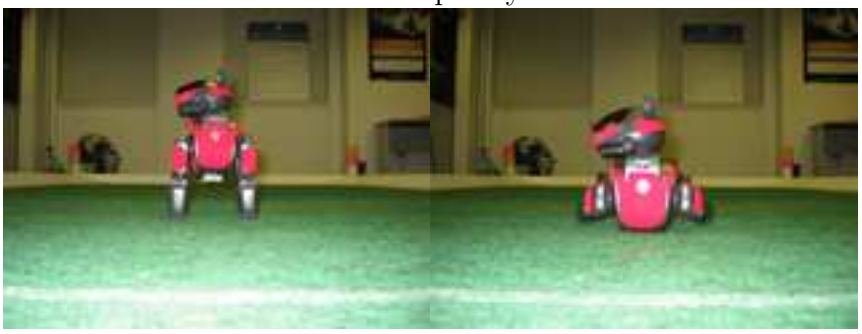
	j_1	j_2	j_3	j_4	j_5	j_6	j_7	j_8	j_9	j_{10}	j_{11}	j_{12}	j_{13}	j_{14}	j_{15}	j_{16}	Δt
$Pose_{initial_1}$	-12	30	91	-12	30	91	-30	6	104	-30	6	104	0	0	0	0	64
$Pose_{critical_0}$	-12	30	91	-12	30	91	-70	45	104	-70	45	104	0	0	0	0	64
$Pose_{critical_1}$	-120	90	145	-120	90	145	120	25	125	120	25	125	0	0	0	0	1
$Pose_{critical_f}$	-12	30	91	-12	30	91	-30	6	104	-30	6	104	0	0	0	0	64

Table 8: Chest push kick initial action and critical action

Since the critical action did not add unwanted momentum that hindered the robot's ability to resume its baseline motion, there was no need to create a final action. The above table represents the complete fine controlled action.

3.3.2 Fall forward kick

It is also important in robosoccer for the robot to have a strong kick that propels the ball a great distance away. Thus, we created the fall forward kick. The fall forward kick makes use of the forward momentum of the robot as it falls from standing position to lying position. Since the kick begins in a standing position, the robot can quickly transition from the walk to the kick. However, since the kick ends in a lying position, the robot does not transition from the kick back to the walk as quickly.



The above images show the poses of the isolated kick. The following table shows the critical action for the fall forward kick after tuning for Δt.

	j_1	j_2	j_3	j_4	j_5	j_6	j_7	j_8	j_9	j_{10}	j_{11}	j_{12}	j_{13}	j_{14}	j_{15}	j_{16}	Δt
$Pose_{critical_0}$	-5	0	20	-5	0	20	-35	6	75	-35	6	75	45	-90	0	0	
$Pose_{critical_f}$	-100	23	0	-100	23	0	100	6	75	100	6	75	45	-90	0	0	32

Table 9: Fall forward kick critical action

We then integrated the walk with the kick. There was no unwanted momentum resulting from $(Pose_{initial_0}, Pose_{critical_0}, 32)$, so there was no need to create an initial action. Testing

revealed that $(Pose_{critical_f}, Pose_{final_f}, \Delta t)$ caused the robot to fall forward on its face every time. Although the robot had successfully kicked the ball, the robot could not immediately resume walking. In this situation, the robot had to wait for its fall detection to trigger and tell it to get up (which takes time) before resuming the walk. Thus, we found a $Pose_{final_1}$ such that $\{(Pose_{critical_f}, Pose_{final_1}, 32), (Pose_{final_1}, Pose_{final_f}, \Delta t)\}$ does not hinder the robot's ability to resume walking.

The above image shows the $Pose_{final_1}$. The table shows the critical action with $Move(Pose_{critical_f}, Pose_{final_1}, 32)$.

	j_1	j_2	j_3	j_4	j_5	j_6	j_7	j_8	j_9	j_{10}	j_{11}	j_{12}	j_{13}	j_{14}	j_{15}	j_{16}	Δt
$Pose_{critical_0}$	-5	0	20	-5	0	20	-35	6	75	-35	6	75	45	-90	0	0	32
$Pose_{critical_f}$	-100	23	0	-100	23	0	100	6	75	100	6	75	45	-90	0	0	32
$Pose_{final_1}$	90	90	0	90	90	0	100	6	75	100	6	75	45	-90	0	0	32

Table 10: Fall forward kick critical action and $\{(Pose_{critical_f}, Pose_{final_1}, 32)\}$

From observation, it is noted that transitioning from $Pose_{critical_f}$ directly to $Pose_{final_1}$ is not ideal. The robot would fall over 25% of the time (in 20 trials) during $(Pose_{critical_f}, Pose_{final_1}, 32)$. Thus, we added $Pose_{final_0}$ to precede $Pose_{final_1}$ in the final action.

The above image shows the $Pose_{final_0}$. Afterward adding $Pose_{final_0}$, the robot no longer fell over when transitioning from the kick to the walk. The following table shows the entire fine controlled action, consisting of the contact action and the final action.

	j_1	j_2	j_3	j_4	j_5	j_6	j_7	j_8	j_9	j_{10}	j_{11}	j_{12}	j_{13}	j_{14}	j_{15}	j_{16}	Δt
$Pose_{critical_0}$	-5	0	20	-5	0	20	-35	6	75	-35	6	75	45	-90	0	0	32
$Pose_{critical_f}$	-100	23	0	-100	23	0	100	6	75	100	6	75	45	-90	0	0	32
$Pose_{final_0}$	-100	90	0	-100	90	0	100	6	75	100	6	75	45	-90	0	0	32
$Pose_{final_1}$	90	90	0	90	90	0	100	6	75	100	6	75	45	-90	0	0	32

Table 11: Fall forward kick critical action and final action

3.3.3 Other kicks

We created several other kicks with this technique. These included the front power kick, the head kick, and the arms together kick.

Front Power Kick

The front power kick was the first kick we created for the RoboCup. We modeled this kick after what seemed to be the predominant goal-scoring kick from previous RoboCup competitions. [5] During the kick, the robot raises its two front legs up and drops them onto the sides of the ball from a 'broadbase' position. The force of the falling legs propels the ball forward.

We wanted our front power kick to transition from any walk without prematurely tapping the ball out of the way. Thus, we started the kick in a 'broadbase' position in which the robot's torso is on the ground with its legs spread out to the side. If the robot were to transition into the front power kick from a standing position, the robot would drop to the ground while pulling its legs away from the ball.

From this broadbase position, the robot then moves its front legs together to center the ball.


After the ball has been centered, the robot moves its front legs up above its head and then quickly drops the front legs onto the sides of the ball, kicking the ball forward.



We found that the kick moves the ball relatively straight forward a distance of up to 3 meters. However, we noticed that the robot's front legs would miss the ball if the ball lies within 3 cm of the robot's chest. We resolved this issue by utilizing the robot's mouth to push the ball slightly forward before dropping its legs on the ball.

Head Kick

The head kick was created to satisfy the need for a kick in a non-forward direction. We decided that the head could be used to kick the ball to the left or to the right. During the head kick, the robot first leans in the direction opposite of the direction it intends to kick the ball.

The robot then moves its front leg (left leg when kicking left, right leg when kicking right) out of the way.

Finally, the robot leans in the direction of the kick as the head turns to kick the ball.

The head kick moves the ball almost due left (or right) a distance of up to 0.5 meters. We discovered that the head kick was especially useful when the ball was close to the edge of the field. The robot could walk to the ball, head kick the ball along the wall, and almost immediately continue walking, whereas the front power kick oftentimes kicked the ball against the wall, in essence moving the ball very little, if at all.

Arms Together Kick

After creating kicks geared toward ball scoring, we realized that we needed a kick for the goalie to block the ball from entering its goal. Deciding that speed and coverage area were more important than the direction of the kick, we created the arms together kick. During the arms together kick, the robot first drops into broadbase position.

The robot then swings its front left leg inward, swings it back out, swings its front right leg inward, and swings it back out.


The arms together kick proved successful at quickly propelling the ball away from the goal.

4 Conclusions

In this paper, we introduce a general technique for creating fine controlled actions for robots. Our experiments indicate that a wide variety of fine controlled actions can be created with this technique. The technique that had originally been developed to create kicks that transitioned from and to a walk has been generalized to apply other kinds of motion. The technique successfully created a fine controlled action for pushing an elevator button and transitioning from and to a baseline walking motion. The results demonstrate the improvement in an action's success from the inclusion of the initial action and/or final action.

References

[1] Sony. Sony Global - AIBO Global Link, 2003. URL=http://www.sony.net/Products/aibo/index.html.

[2] Bill Messner and Dawn Tilbury. Control Tutorials for Matlab: PID Tutorial, 1996. URL=http://rclsgi.eng.ohio-state.edu/matlab/PID/PID.html.

[3] Stone, Peter, Kurt Dresner, Selim T. Erdogan, Peggy Fidelman, Nicholas K. Jong, Nate Kohl, Gregory Kuhlmann, Ellie Lin, Mohan Sridharan, Daniel Stronger, and Gurushyam Hariharan. "UT Austin Villa 2003: A New RoboCup Four-Legged Team." The University of Texas at Austin, Department of Computer Sciences. AI Technical Report 03-304. October 6, 2003. 76 pages. URL=http://www.cs.utexas.edu/users/UTCS/ailab/index/html/Abstracts.2003.html.

[4] The RoboCup Federation. RoboCup Official Site, 1998-2003. URL=http://www.robocup.org/02.html.

[5] Manuela Veloso, Scott Lenser, Douglas Vail, Maayan Roth, Ashley Stroupe, and Sonia Chernova. CMPack-02: CMU's Legged Robot Soccer Team. 2002. URL=http://www.openr.org/robocup/code2002SDK/CMU/cmu_teamdesc.pdf.