UNDERGRADUATE THESIS
School of Engineering and Applied Science
University of Virginia

Finding a Give-And-Go In a Simulated Soccer Environment

Submitted by
Dev Batta
Computer Science/Computer Engineering

TCC 402
April 22, 2002

On my honor as a student, on this assignment I have neither given nor received unauthorized aid as defined by the Honor Guidelines for Papers in TCC Courses.

Signed ___________________

Approved ___________________________ Date _____________
Technical Advisor – David Evans

Approved ___________________________ Date _____________
TCC Advisor – Richard Jacques

Table of Contents

ABSTRACT

A dynamic, unpredictable world complicates independent decision-making in a team of mobile robots. The number of possible actions a robot can take is vast and environmental variables are constantly changing. Therefore, a robot must make a decision quickly and efficiently. This is true in all team environments, including a soccer match. This research uses a simulated soccer match and concentrates on finding when the opportunity for a special situation known as a give-and-go exists. The focus is to break down a continuous model of the world into discrete steps that evaluate the environment. A give-and-go is a sequence of two passes between teammates that attempts to leave a defender behind the play.

The evaluation of a give-and-go is broken down into five steps: 1) Does my team have the ball? 2) Is there a teammate further up field to which the player with the ball can pass? 3) Is there a defender in between the passer and the receiver that will not be able to intercept the initial pass? 4) Is there an open area of the field to which the passer can run? 5) Can the receiver redirect the ball to the open area of the field that the passer is running without the other team intercepting? For a give-and-go to exist, these questions must be evaluated in order and the answer must be yes to each of them.

CHAPTER 1: INTRODUCTION

1.1 Summary

How can the continuous world of a robot in a simulated team environment be broken down into a discrete model of the world? This is an important question when trying to decide what action to take next. An agent in a team environment needs to know the state of the world around it before it decides on how to respond.

1.2 Problem Definition

A dynamic, unpredictable world complicates independent decision-making in a team of mobile robots. The number of possible actions a robot can take is vast and further complicated by environmental variables that are constantly changing. In a real-time environment, an action must be chosen before there is a drastic change in the world. Therefore, a robot must make a decision in a quick and efficient manner. Artificial decision-making extends into such fields as E-commerce, medicine, and military tactics. My medium of research is a simulated soccer environment.

My focus is on identifying when an opportunity for a give-and-go pass is available. A give-and-go pass is a combination of passes in which the player with the ball passes to a teammate, runs to an open area of the field, and then immediately gets the ball in return (Fig. 1). What view of the world is necessary for the opportunity of a give-and-go? The main challenge is simplifying the continuous world to recognize when this special situation exists.

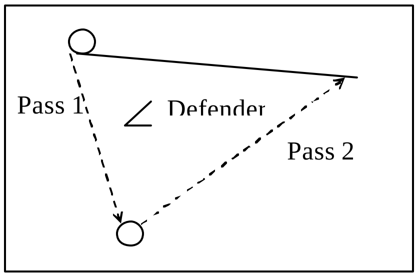


Figure 1: Schematic of a Give-and-go Pass

1.3 <u>Rationale</u>

The RoboCup initiative is an international symposium that promotes research in artificial intelligence and robotics. RoboCup uses soccer as a fun and interesting way to research multi-agent systems using a simulated environment. My primary focus will be to evaluate a model of the world to recognize special situations when a player on my team has the ball. Examples of special situations can include passing to a teammate or a give-and-go. In past work, the focus of a player with possession of the ball has been to decide whether to pass, shoot, or dribble. Little focus has been spent on recognizing if a special situation exists.

The ability to recognize an opportunity for a give-and-go places the offense a considerable advantage. After delivering the first pass, a decision does not need to be made about what to do next; the offensive player is running to its next spot before the defender has time to react to the pass. If the player receiving the initial pass also recognizes the give-and-go situation, the defender is left behind.

Additionally, I had to find a way to adapt my research to work in a *swarm* programming environment. Swarm programming uses a set of behaviors to program and constrain the single agents for the sake of the desired global behavior of the team (e.g. *disperse* is a behavior with the goal of keeping the players spread out by constraining each member to keep a minimum distance from all teammates). Each behavior is contained in its own module and is combined with other behaviors to form a strategy. The behavior of the swarm depends on the total behavior of the single units but is resilient to a few misbehaving members and to changes in the environment [Evans 2000].

CHAPTER 2: LITERATURE REVIEW

The RoboCup simulation league held its first international competition in 1997. Since then, many competitions have been held with respected research teams competing from around the world. Fortunately, many of the teams make their research public over the web. Much of the literature that I use is from documentation of past teams. Each team has its own method of interpreting the data that is presented in the world model and acting on it. Things can be learned from the mistakes and successes of each of these teams. This leads me to conduct most of my research from documents found online rather than from published articles in engineering journals.

The UVA RoboCup team has decided to use code libraries from the Mainz Rolling Brains (MRB) team as skeleton code. Mainz uses a rule-based concept to decide on an action. MRB’s code is well documented and gives our team a starting point to build upon.

CMUnited, a team from Carnegie Mellon University and winners of the 1999 competition, searches hundreds of possible passes to each teammate to find the one that maximizes a score based on probability of an interception and the resulting strategic value of the position. This year they are also introducing the concept of a leading pass where the ball is kicked to a position that a teammate is running toward instead of where the teammate is positioned when the pass is delivered. All of the passing options are considered against keeping the ball or shooting at the goal to select the best action. [CMU2000]

The Karlsruhe Brainstormers, from the University of Karlsruhe in Germany, uses agent based learning to make decisions based on previous experiences. Initially a player does not know how to accomplish any of its goals so it has to record the outcome of a random action (e.g. the player attempts to intercept the ball but doesn't know what actions to take so it runs towards the ball at a random speed and angle). If the outcome of a series of actions is successful, those actions are assigned a cost using a learning function. The player continues to improve its actions by lowering its total cost. As the cost decreases, the player is more likely to achieve its goal. The player is essentially learning that the higher cost series of actions are not the most efficient way and will try different actions to find a better way to succeed. [Karl2000]

Behavior modules are another way to formulate decisions. As described earlier, swarm programming breaks up behaviors into modules and combines these modules to form a strategy. These modules interact by taking the world model and evaluating the importance of the behavior. For example, if a player feels it is far away from the closest teammate, *disperse* would evaluate itself as unimportant and allow other behaviors to dictate the player's actions. A research team for Ohio University uses a similar layered approach by having the behaviors return its applicability, priority, and duration to a higher-level controller [OHIO2000]. The controller then decides which behaviors to use and issues commands to the robots.

Swarm programming works in a similar way as the Ohio University layered approach. Each agent is run by a master controller. The master controller initializes an array of behavior modules that is used to decide from which behaviors the agent can choose. Because the number of behavior modules can grow to a large size, it is unreasonable to calculate the outcome of each behavior. Consequently, each behavior module has a function that quickly computes whether it is useful. For example, the give-and-go module would not need to be considered if the other team has the ball.

If a behavior determines that it is useful, it then calculates an action or a series of actions for the agent to perform. Along with the action, the behavior also returns a recommendation to the master controller. The recommendation is a numeric indicator how helpful the behavior is to team's overall goals. The higher a recommendation, the more likely the behavior is chosen. The master controller then chooses the behavior that returns the highest recommendation and commands the robot to perform the actions returned from the behavior. [WWFC2001]

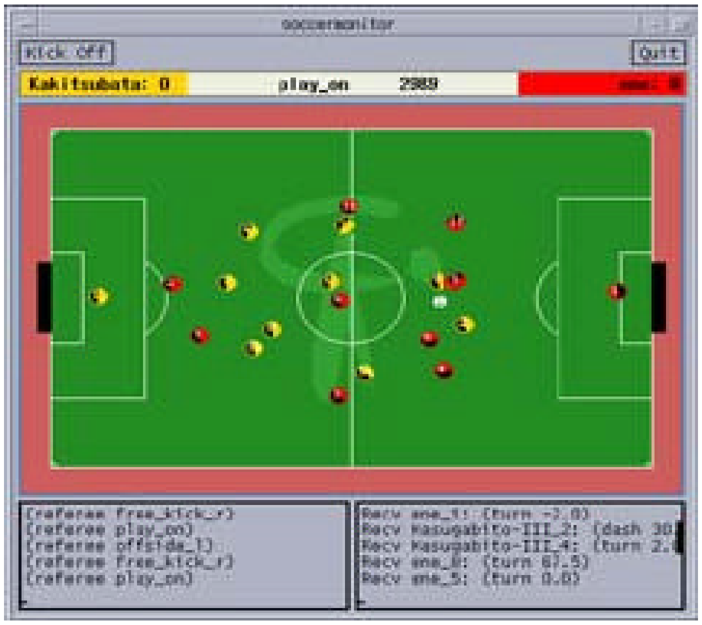
The reigning RoboCup champion, Tsinghuaeolus from China, has a unique way of considering different passing options. In their architecture, the player with the ball considers up to thirty different passing routes and chooses the one that will help the team the most. The agent with the ball assigns teammates to control different sections of each pass route. The control assignments are determined based on each agent's ability to get a specific point on the considered pass route. Whichever agent can get to a point on the pass route, that agent will be assigned control of the point. This method of assigning control of pass routes is superior to the methods used by other teams I have researched because of the ability to pass to an open part of the field rather than directly to teammates only. If a pass route to open space is considered, then the passer can find how successful the pass will be based on who will get to their controlled portion of the pass route first. [TSIN2001]

Ideas from most of these strategies can help further my research. The idea of modules and rule-based decision-making has already been used in the development of our team. The learning strategy is not an area that will be researched for this year’s team but is an interesting subject and could be undertaken in future years. To evaluate past competitors, available source code will be critical to understand the successes and failures of each team.

CHAPTER 3: METHODOLOGY

3.1 <u>Research Medium</u>

The RoboCup initiative is an international symposium that promotes research in artificial intelligence and robotics using soccer. The motivation for my research is to program part of an offense to compete in the RoboCup simulatio n league. An international competition was held this past August in Seattle. A match in the simulation league consists of two teams of eleven players competing on a simulated soccer field (Fig. 2). The SoccerServer is the medium for my research.



```
(referee free_kick_r)
(referee play_on)
(referee offside_l)
(referee free_kick_r)
(referee play_on)
```

```
Recv ene_1: (turn -2.0)
Recv Kasugabito-III_2: (dash 30
Recv Kasugabito-III_4: (turn 2.
Recv ene_8: (turn 67.5)
Recv ene_5: (turn 0.0)
```

Figure 2: Snapshot of the SoccerServer During Competition

3.2 <u>RoboCup Team</u>

The RoboCup team that competed in August 2001, Wahoo Wunderkind FC (WWFC), consisted of University of Virginia Computer Science faculty members David Evans (project leader) and David Brogan and University of Virginia Computer Science undergraduate students Dev Batta, Keen Browne, Jon McCune, and Adam Trost.

3.3 <u>Resources</u>

All of the necessary resources are available on-grounds at the University of Virginia. The graphics lab in Olsson Hall has enough UNIX computers to accommodate all members working on the project. The simulator and skeleton code necessary to develop the team is available online due to the open source nature of RoboCup. All of the coding is done in C++.

3.4 <u>Project Activities</u>

The initial step for conducting my research was learning how the SoccerServer works. For testing purposes, it is necessary to be able to set up situations with just a few players rather than two complete teams of eleven players competing. It is possible to limit the number of players on the field but it is not possible to give them an initial position. Therefore, I need to observe the game or sort through log files to determine if players are taking advantage of give-and-go situations.

The next step was learning the architecture of the current WWFC team. One of the more important aspects of the architecture was understanding how player positioning is determined in the world model. In the WWFC architecture, a global absolute field position and a local position relative to other agents can both be determined. Position, along with direction and speed, determines where a player is going and helps the passer decide if a give-and-go is feasible. Once the positions of the players are determined, the agent with the ball can look at different scenarios and determine which action to take.

Issues that must to be taken into account when deciding if a give-and-go is feasible are: 1) Does my team have the ball? 2) Is there a teammate further up field to which the player with the ball can pass? 3) Is there a defender in between the passer and the receiver that will not be able to intercept the initial pass? 4) Is there an open area of the field to which the passer can run? 5) Can the receiver redirect the ball to the open area of the field that the passer is running without the other team intercepting? If the answer to all of these questions is yes, then there exists a chance for a give-and-go pass. The give-and-go module then computes the recommendation and passes it back to the master control.

CHAPTER 4: TEAM ARCHITECTURE

The architecture for the WWFC RoboCup team divides the agent into three separate components, the WorldModel, behaviors, and the body. The MasterControl is responsible for controlling the three components of the agent (also known as the client). MasterControl is the class that is the heart of each agent and is divided into listen, think, and act. Understanding the WWFC team architecture is essential in understanding the implementation of the give-and-go behavior.

4.1 MasterControl: Listen

Listen is the connection to the SoccerServer and is responsible for receiving sensor data from the body of the robot and inserting the information into the WorldModel. The information received from the server is stored in the WorldModel in an object-oriented format. Listen is in charge of storing the location of the ball, the client, and other players.

4.2 MasterControl: Think

Think is the primary decision-making component of the client. As previously mentioned, behaviors are divided into modules that interpret the data of the WorldModel. The modules represent a set of primitive actions such as disperse or center. Each behavior module consists of a behavior, perception, and recommendation. When a behavior is evaluated, it relies on a set of perceptions to decide its usefulness. For example, if the client is already spread out from the rest of its teammates, the disperse behavior will return a low recommendation to the MasterControl.

The perceptions for the behavior modules are functions that return interpretations of the WorldModel. Perceptions can be as simple as finding the closest teammate to the ball or as complex as predicting if the other team will intercept a pass to a teammate. The particular perceptions of a behavior are taken into account when determining the recommendation. If a behavior is useful, the perceptions will allow the behavior to know if its actions are feasible. For example, if a player with the ball is trapped between defenders, the pass behavior can be called upon to decide if a teammate is open. The perceptions for pass will find the closest teammates to the client and determine whether a successful pass can be made.

Once the perceptions are evaluated, the behavior module can determine a recommendation and a course of action. The higher the recommendation, the more likely the MasterControl will select the actions of the behavior. For the pass behavior, if a pass can be made to a teammate that is in position to shoot on goal, the pass module will return a high recommendation and an action to kick the ball in the direction of the teammate with a certain amount of power.

Behavior modules contain two main methods, one to determine its usefulness and another to generate a recommendation. [WWFC2001] The usefulness method is a quick check to see if the behavior should even be considered at the current time and is provided to eliminate unnecessary calculations. For example, a give-and-go is useful only if the client has the ball and will deem itself useless in all other situations. If a behavior is considered useless, it will automatically return a recommendation of zero. If it is useful, the perceptions will be called upon to calculate a recommendation and a set of actions. Once all behaviors have returned their recommendations and actions the MasterControl selects the behavior with the highest recommendation and sends the set of actions for interpretation to the act function.

4.3 MasterControl: Act

Act is a representation of the body of the client. When MasterControl selects a behavior, the actions are sent to the server through the body using the act function. The body takes the action list sent from think and translates it into language that the server can understand. All commands sent to the server pass through the body. [WWFC2001]

The basic structure of a client is summarized in Figure 4. Listen retrieves sensor data and injects it into the WorldModel. Think interprets the WorldModel and decides on the best behavior to follow with a set of actions. Act takes the actions from think and sends movement commands to the server.

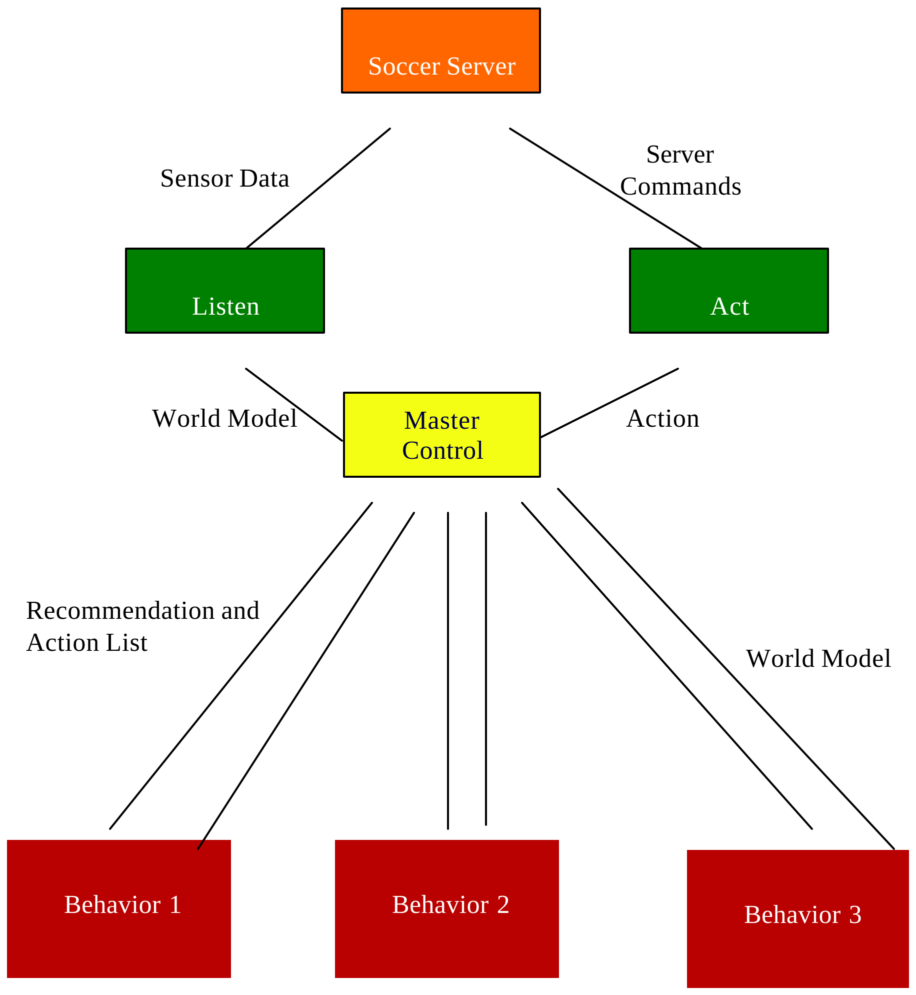


Figure 4: Interactions of the agent control system. First, Listen develops a WorldModel from the server for the MasterControl. The behaviors use the contents of the WorldModel to form recommendations and actions. Act uses the recommended actions to send commands to the server as instructions for the agent.

CHAPTER 5: THE PROJECT

5.1 Requirements

Before writing the algorithms that looks for the opportunity of a give-and-go, the necessary perceptions have to first be identified. These perceptions take the continuous world model and break it into a series of discrete steps that will be used to decide the recommendation. Each player on the team will evaluate these perceptions before deciding which action to take next. Another concern is the synchronization of the two players involved in the give-and-go.

5.2 Give-and-Go Perceptions

The perceptions that have been identified for a give-and-go situation are:

1. Am I in control of the ball?
2. Is there a teammate that is closer to the goal than I am?
3. Which teammate, that is closer to the goal than me, is closest to me?
4. Is there an opponent in between my teammate and I that can be a victim of the give-and-go?
5. Will a pass to this teammate be intercepted by the other team?
6. Is there an open area of the field that I can run to and receive an immediate return pass from my teammate?

These perceptions will be evaluated in order. If the answer to any of the questions is no, the give-and-go behavior module will automatically return a recommendation of zero to the MasterControl.

5.2.1 Ball-Control Perception

The perception that checks to determine if the agent has the ball is actually the usefulness method (described in the section on the think part of MasterControl) for the give-and-go behavior. If the agent does not have possession of the ball, a give-and-go cannot be initiated by the agent. If this perception determines the agent does not have the ball, many unnecessary calculations are prevented.

The implementation of this method is quite simple. A check is done to see if the ball is within a certain distance to the player. The method returns true if the ball is within the specified distance and returns false if it is outside the distance. This distance is a constant specified in a file of global constants.

5.2.2 Closer To The Goal Perception

If the agent decides that it has possession of the ball, it needs to next find out if there is a player closer to the goal that is in position to receive a pass. To simplify and speed up the algorithm, only the closest teammate is considered for a give-and-go.

In the implementation of this perception, the players in between the player with the ball and the opponent's goal are the only ones considered. The agent loops through the position of each of its teammates, stores the identity of the closest teammate, and finally returns true. If there are no teammates closer to the opponent's goal than the player with the ball, the perception returns false and the behavior's recommendation is set to zero.

5.2.3 Opponent Between Ball and Teammate Perception

After identifying the closest teammate in front of the ball, the agent must determine if there is an opponent between the ball and the teammate. If no defender is present between the two teammates then there is no opportunity for a give-and-go. A defender between the ball and the teammate is defined in this situation as being positioned in a triangular region bounded by the horizontal location of the player with the ball, the vertical location of the closest teammate, and the line connecting the two teammates. Figure 5a illustrates a situation where a defender is in the bounded box and Figure 5b shows a situation where the defender is outside.

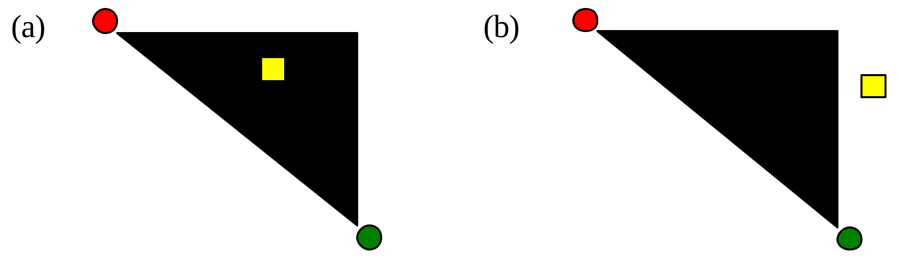


Figure 5: The bounded triangle (black area) is created using the horizontal position of the player with the ball (red circle), the vertical position of the teammate (green circle), and the line connecting the two players. In (a), the opponent (yellow box) is in the bounded triangle and the opportunity for a give-and-go is open. In (b), the opponent is outside the box and the give-and-go is aborted. Note that the initial pass in both situations is possible but the second pass would most likely be intercepted by the defender in (b).

This perception loops through each opponent player to find who is in the triangle. If exactly one opponent is found to be inside the triangle, the perception returns true and the behavior continues. If there are no opponents or more than one opponent in the triangle, the perception returns false and the recommendation for the behavior is set to zero. The hypotenuse of the bounding triangle is the slope-intercept form of the line connecting the two players.

5.2.4 First Pass Perception

At this point in the behavior, there is sufficient information to determine if there is an opportunity for a give-and-go. The remaining issues that need to be resolved are the two passes that complete the give-and-go. Predictions are made about whether the passes will be intercepted by the other team before getting to the teammate. The first pass of the give-and-go is easier to assess because the preferred destination of the ball is known because the location of the teammate is known.

Predicting an interception by the opponent is a time consuming calculation. Therefore, predictions only on players within a certain range are made. For instance, it is unlikely that an opponent that is positioned above the player with the ball will intercept the ball if a pass is made to a teammate below the ball. Figure 6 shows several player positions and which opponents will be considered and which will not.

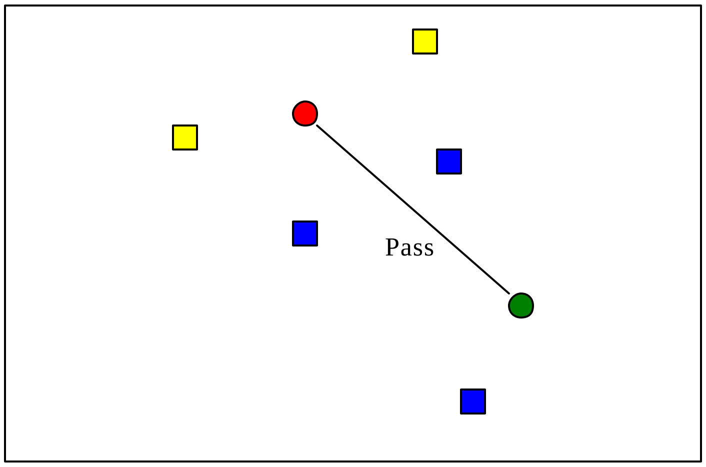


Figure 6: Example of opponents who will be considered when predicting if a pass will be intercepted. The pass is made from the red circle to the green circle. The yellow squares represent opponents that are not considered threats to intercept because of their positions relative to the direction of the pass. The blue squares are opponents that are considered threats to intercept.

If an opponent is considered a threat, a prediction is made on whether that player will intercept the ball at some future time. The future position of the player is predicted by simulating its movement using its current position, velocity, maximum velocity, and acceleration. The movement of the ball is also simulated in a similar fashion and if the opponent crosses a point in the path of the pass before the ball, the pass will be intercepted. If an opponent that will intercept the pass is found, the perception returns false and the recommendation of the behavior is set to zero. Otherwise, the behavior will be ready to evaluate the second pass.

5.2.5 Second Pass Perception

The second pass in the give-and-go is more difficult and more costly to compute than the first pass. The reason for this is the uncertainty in the final position of the player receiving the second pass. The passer can consider several locations to place the pass. Other than the location uncertainty, the second pass is evaluated in a similar manner to the first pass.

To simplify the calculations, five different passes are considered. The first consideration is to the location where the vertical of the passer and the horizontal of the receiver cross (e.g. the location of the right angle in Figure 5). Predictions are made on the success of this pass in the same manner as the predictions on the first pass. If the first consideration is found to be intercepted, the next pass to be considered will be at the location the receiver will be one time-step after reaching the first considered pass location. One time-step is the amount of time a player has to decide on its next action. This pass is assessed in the same manner as the first. Up to five time-steps will be predicted until a suitable pass is found. When a pass that can be made is found, the perception returns true and the recommendation is set to a high number. If a pass is not found after the fifth time-step, the perception returns false and the recommendation is set to zero. Figure 7 displays the progression of considered second passes.

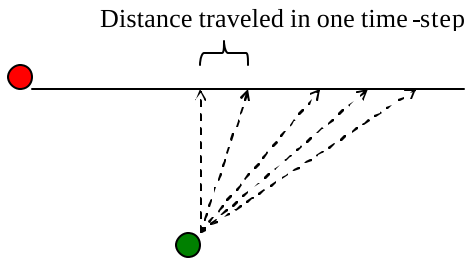


Figure 7: The considered passes for the second pass of the give-and-go. The progression goes from left to right. Once a suitable pass is found remaining passes are not considered. Notice the distance between passes varies due to agent acceleration and fatigue.

5.3 Player Synchronization

Once the player with the ball has evaluated the perceptions, it is ready to perform the next action. If a give-and-go is the chosen behavior, the agent will pass the ball to its closest teammate. How does the other teammate know that the passer is expecting a return pass? The answer to this question is a feature in the SoccerServer that allows limited communication between players in close proximity to one another. A message can be sent between teammates. The format for the message will tell the teammate to stop, expect a pass, and send a return pass to a certain location. The location sent in the message is the same location that the second pass perception computed. After the message is sent, the passer knows that the teammate is aware of the give-and-go to be executed.

CHAPTER 6: RESULTS

When the give-and-go behavior was used in a controlled environment of two offensive players and one defensive player it hardly succeeded. More than half of the time, the defender would intercept the ball before reaching the teammate. One several occasions, it was unclear whether the pass was being intercepted or if the defender was stealing the ball from the passer. The other half of the time, the passer would not attempt to pass to its teammate and would instead dribble the ball or have it stolen by the defender.

The few times the ball got to the teammate of the passer, it was uncertain if the ball had actually been passed or if the passer had just dribbled too far in front of itself. An attempt for a second pass was never made during testing. The receiver would either dribble the ball or lose control and have the ball stolen.

According to the log files, the intercept function returned false very often for the first pass. When intercept returns false, the behavior aborts and the recommendation is set to zero. Intercept usually returned false for the second pass.

CHAPTER 7: CONCLUSIONS

7.1 Interpretations

Many of the situations that represented good opportunities for a give-and-go were not exploited. This can be attributed largely to an inaccurate intercept function. This function returned false too often and did not give the players a chance to make an attempt to execute a give-and-go. Just from watching the game, it was apparent that many opportunities were lost.

One obvious problem with the function is the amount of computation involved when assessing a pass. The passer can potentially run the intercept function sixty times before deciding what to do (one for each of the ten players for the first pass and ten for each of the five considered second passes). If the computation takes too long, the agent cannot make a decision on what to do next. In addition, inaccuracies in intercept calculations cause the behavior to abort and set the recommendation to zero.

Although the give-and-go behavior was not efficient, an improvement in the intercept algorithm would be the first step in a remedy. The underlying methods of recognizing the opportunity for a give-and-go do a quick and accurate job of breaking down the world into discrete steps. The perceptions provide a logical evaluation of the state of the world when deciding if a give-and-go is possible. These basic perceptions can form the basis of any future give-and-go behavior.

Another problem with the behavior was the lack of synchronization between the players. After making the first pass, the second pass should be made automatically. The receiver did not get a clear message of where to make the return pass and was slow to react. This delayed reaction usually resulted in an easy steal by the defensive player.

7.2 Recommendations

The most glaring flaw of the give-and-go behavior is the intercept routine. Replacing the current method with a quicker, more accurate algorithm would greatly improve the behavior. Implementing the behavior as a combination of more primitive swarm behaviors is another option. For instance, writing separate pass and intercept behaviors would adhere more clo sely to the swarm architecture. The autonomous nature of swarm programming would also be preserved because the communication between the two agents would be eliminated. The two players involved in the give-and-go could evaluate their own pass separately and increase the speed of the behavior. Again, the basic perceptions used in the current give-and- go behavior would still apply,the only difference being in the separation of implementation.

BIBLIOGRAPHY

[CMU2000] ATT-CMUnited RoboCup-2000 Simulator Team. Carnegie Mellon University. July 1, 2001. <http://www.cs.cmu.edu/~pstone/RoboCup/ATTCMUnited2000-sim.html>

[EVANS2000] Evans, David. Programming the Swarm. University of Virginia. July 24, 2000.

[KARL2000] Kalrsruhe Brainstormers Team Page. University of Karlsruhe. July 1, 2001. <http://www.karlsruhe-brainstormers.de/html/nav.htm>

[OHIO2000] Chelberg, David, Lonnie Welch, Arvind Lakshmikumar, Matthew Gillen, and Qiang Zhou. Meta-Reasoning For a Distributed Agent Architecture. Ohio University. January 19, 2000. <http://zen.ece.ohiou.edu/~robocup/papers/HTML/SSST/SSST.html>

Blumberg, Bruce and Tinsley A. Galyean. Multi-Level Direction of Autonomous Creatures for Real-Time Virtual Environments. MIT Media Lab.

Drucker, Christian, Sebastion Hubner, Ubbo Visser, and Hans-Georg Weland. “As time goes by” - Using time series based decision tree induction to analyze the behaviour of opponent players. University of Bremen. June 29, 2001. < http://www.informatik.uni-bremen.de/~visser/liter/DrueckerEtAl.pdf>

Heintz, Fredrik. RoboSoc: a system for Developing RoboCup Agents for Educaional Use. Linköpings University. March 23, 2000.

RoboCup Homepage. June 2000. <http://www.robocup.org>.

Stone, Peter and David McAllester. An Architecture for Action Selection in Robotic Soccer. Fifth International Conference on Autonomous Agents. October 2000.

Wunstel, M., D. Polani, T. Uthmann, and J. Perl. "Behavior Classification With Self-organizing Maps". In Fourth International Workshop on RoboCup.

APPENDIX A: giveandgoperception.h

The header file for the give-and-go perception.

APPENDIX B: giveandgoperception.cpp

The source file for the give-and-go perception.

APPENDIX C: giveandgobehavior.h

The header file for the give-and-go behavior.

APPENDIX D: giveandgobehavior.cpp

The source file for the give-and-go behavior.