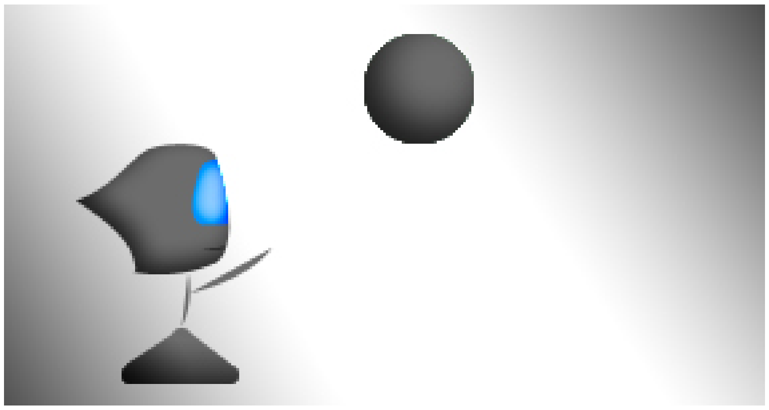
Robots Playing Catch

Brandon Tolsch

December 9, 2014



Abstract

We present a hybrid system which represents two robots playing catch. The ball is passed through the air between the two robots. We discuss various simplifications and versions of this problem which can be described using differential dynamic logic. We develop a hybrid program for describing this game with the robots constrained to 1D motion where the robots can adjust the power with which they shoot the ball at each other. We prove the ball is never dropped by the robots and the robots never collide in the process of playing the game. We also provide some guidance for avoiding difficulties with evolution domain constraints in differential dynamic logic.

1 Introduction

The study of hybrid systems is about systems with both discrete and continuous dynamics. Discrete dynamics include any change that is instantaneous, such as switches which can only be in one of several states at a time. Continuous dynamics include phenomena that evolve continuously over time and can therefore be described by differential equations. Each domain is very complicated to study on its own, so it seems that combining them would exacerbate the difficulty. Examples of hybrid systems include aircraft control systems, train control systems, and many kinds of robots.

It is true that hybrid systems are very complex, but a powerful proof logic is provided by [1] for dealing hybrid systems. The goal when studying hybrid systems is usually to prove that the system always maintains some property. This property is usually a statement of safety or efficiency or both. For example, for a train control system a safety property would be that there is never a collision and an efficiency property would be that trains never arrive later than some constant time buffer.

The problem we consider here is two robots playing catch with a ball. Each robot has a basket on top of it for catching the ball and a cannon for shooting the ball. One of the robots starts with the ball loaded in its cannon and it needs to shoot the ball so the other robot can catch it. When the other robot catches it in its basket, it is automatically loaded into its cannon. This continues back and forth. The goal is to never let the ball touch the ground so the robots can happily continue playing catch forever.

This problem can be related to many applications, but there are two that we present here: Cirque du Soleil and RoboCup robot soccer. Cirque du Soleil is a type of circus performance that involves elaborate tricks. One type of trick that is performed in these shows is to have many people on stage at once doing many different things that could easily interfere with each other, but don't. For example, imagine two people juggling balls quickly between each other. With enough balls in circulation between them, it almost seems like a continuous arc is created between them. Then consider another pair of jugglers throwing their balls through the first stream of balls. It seems like the two streams of balls should interfere with each other and they should all collide and fall to the ground but if everything is well timed the two streams will interleave perfectly.

Another example which is more closely related to the exact problem we are considering is when objects are thrown to or between swinging trapeze artists. While a trapeze artist is swinging, their motion is that of a pendulum and they can't suddenly choose to speed up or slow down. If someone on the ground tries to throw them an object, they have to plan for the trajectory of the object to intersect the trajectory of the trapeze artist at exactly the right time.

In the small-size league of RoboCup robot soccer, two teams of small cylindrical robots move around on a bounded field with a small ball. There is a goal for each team as in regular soccer and a point is scored by getting the ball into the opposing team's goal box. Normal passes and shots occur by rolling the ball on the ground, but it's possible for the robots to chip the ball and send it up into the air. Making passes like this can be very advantageous because you can avoid opponents that would otherwise block a pass. Similarly, it would also be advantageous to predict where the ball will land after it has been chipped by an opponent so you can plan to either intercept it or block the receiving opponent more quickly. The model of our problem is slightly more restrictive since we require both robots to catch the ball while in robot soccer the ball can roll after it lands without issue; the penalty for missing the ball is removed and there are additional dynamics present when the ball hits the ground.

2 Related Work

Previous work has been done by Colin for modeling robot soccer passes as a hybrid system [2]. Colin assumed stationary robots and showed that the ball will stay within the legal bounds of the field when the robots know where the other players are.

RoboCup teams have also spent some time dealing with modeling or predicting chip kicks. The SKUBA robot team has done work on predicting the trajectory of a chip-kick in small-size league robot soccer [3]. This is especially difficult since the robot's perception of the ball comes from an overhead camera. This makes perceiving depth very difficult, which is an important part of determining the trajectory. They are able to derive a reasonable prediction of the trajectory after just a few frames. This information is given to their strategy software module to help decide where their team's players should move. The CMDragons robot soccer team has been using a stochastic model of passing which assigns a relatively low probability of pass-success to chip kicks [4]. Finally, Nunome et al. [5] have worked on detecting the spin of the ball in the air from camera frames. This is important because the spin on the ball can have a huge effect on where the ball goes when it lands. It might move forward quickly, move forward slowly, stop, or even roll backwards depending on the spin imparted by the kicker.

3 Model

Both the shooting and receiving robots have many choices to make. The receiving robot could remain still while the ball is in the air and hope the shooter has good aim. This makes the receiver a relatively easy target because the shooter knows exactly where it should try to make the ball land. The receiving robot could remain still only until the ball is shot and then move to where it thinks the ball will land by observing its trajectory. This also makes the receiver an easy target but the receiver can still adjust its position to improve its chance of catching the ball. The receiving robot could also wander around aimlessly until the ball is shot and then try to move to where it thinks the ball will land. This makes the receiver a harder target. The shooter can also choose whether to move or remain still before shooting, though what it does after shooting is not relevant until the next round of shooting. In all of these cases, the burden is largely on the shooter to decide a good target location that is safely reachable by the catcher.

We chose to model the robots as two circular regions moving in an infinitely large plane. The ball moves in three dimensional space, where the robots are in the plane $z = 0$. Although giving the robots boundaries is more realistic, it doesn't add much to the result here. In the case of modeling chip kicks in robot soccer directly, where the ball can also roll, boundaries would be more important. If the ball ever goes below height 0 then the game is over and the robots have lost. The ball can be caught if it lands anywhere in the robot's circular bounding region. Although the collision boundary region of the robots and the region in which they are able to catch the ball could be different, it doesn't add anything to the result; collision detection and catch detection are already completely separate. The cannon is not considered to interfere with the robot's ability to catch the ball. The ball is fired from the center of the robot at height $z = 0$. These are both reasonable assumptions since we could consider a robot where the opening of the cannon is at the bottom of the basket. This would not interfere with catching the ball and it would be fired from the same height at which it is caught. Firing the ball is modeled as projectile motion without air resistance where the shooting robot gives the ball some initial velocity as a result of the cannon firing.

We will refer to the two robots as robot A and robot B. We will use the following notation for this model:

- $IR_A(t)$ - Predicate which is true when the ball is in the circle of robot A, ignoring the ball's z position, at time t.

- $(A_x(t), A_y(t))$ - Center of the circle of robot A at time t.
- $(A_{vx}(t), A_{vy}(t))$ - Velocity of the robot A in the x and y directions at time t.
- $(A_{ax}(t), A_{ay}(t))$ - Acceleration of the robot A in the x and y directions at time t.
- $(b_x(t), b_y(t), b_z(t))$ - Center of the circle of robot A at time t.
- $(b_{vx}(t), b_{vy}(t), b_{vz}(t))$ - Velocity of the robot A in the x and y directions at time t.
- $(b_{ax}(t), b_{ay}(t), b_{az}(t))$ - Acceleration of the ball in the x, y, z directions at time t.
- a_A - Maximum forward acceleration of the robots.
- a_B - Maximum braking acceleration of the robots.
- r - Bounding radius of both robots.
- $C_v(A)$ - Initial velocity imparted on the ball by A's cannon.
- $(C_x(A), C_y(A), C_z(A))$ - Direction vector of A's cannon.
- g - Acceleration due to gravity.

We are primarily concerned with two types of properties for our hybrid system: safety and efficiency. Liveness is also sometimes considered in hybrid system models. Liveness is usually a statement either that the system always makes progress in some way or that the system doesn't diverge or separate to become trivially safe. For example, liveness for our system without the ball could be that the robots don't simply stay still or get arbitrarily far away from each other. In our system, this type of liveness isn't an issue because no matter how the robots move, they have to continue to shoot the ball to each other. Liveness in our system could be that the robots don't always shoot exactly to the other robot so they are each forced to move occasionally.

3.1 Safety

Safety in this system primarily means two things. The first, which is more intuitive, is that the robots never collide with each other. If the robots decide never to move then this is trivially satisfied. Remember that this doesn't violate liveness for this system though. The second is that the ball can never be dropped. This is a safety condition because we defined the game to be lost if the ball is dropped. If the model were extended to include boundaries in the plane, safety would also mean that the robots stay within the boundaries.

There are several cases of motion for which the collision safety condition is not trivial: motion of both robots before the shot, motion of the shooter before the shot, shooter is stationary before the shot. If both robots are moving before the shot, they both need to be mindful of the other to avoid collisions. Additionally, the shooter needs to plan to shoot the ball to a location such that the catcher can reach the ball in time without hitting the shooter.

If only the shooter is moving before the shot, it needs to make sure not to hit the stationary catcher and the shooter still needs to ensure that the catcher can reach the ball without hitting the shooter. The shooter's motion after the shot needs to be taken into consideration for both of these cases; it continues to move after the shot since it can't immediately stop and this could interfere with the catcher's ability to reach the ball's landing location in time. If the shooter is not moving before the shot, it only needs to consider itself as a stationary object to calculate safe locations for the ball to land. Calculating these safety regions is very difficult for arbitrary motion of the robots, so several simplifications will be used. These will be described in section 4.

The second safety condition is that the ball doesn't get dropped. Recall that this means that the ball is in one of the robot's circles if the ball's z coordinate is 0. In our notation, this is $b_z(t) = 0 \rightarrow (IR_A(t) \vee IR_B(t))$. It can also be implemented by letting the ball fall below $z = 0$ if it outside of a robot's circle. The is described by the formula $(IR_A(t) \rightarrow b_z(t) \geq 0) \wedge (IR_B(t) \rightarrow b_z(t) \geq 0)$. These restrictions can be implemented directly by using them as a domain constraint on the evolution of the ball's differential equation. This means that as the ball is moving through the air in our hybrid system, it must stop if it would violate the implication (by going below $z = 0$ inside one of the circles). The region where the ball can land to satisfy this safety condition will be called the *reachability region*. Although both formulations of the condition seem very similar, we will show in section 6 that the latter is more difficult to use.

3.2 Efficiency

Efficiency is not an easy concept to apply to this system. You could require the shooter to choose the target location that requires the least amount of work from the other. However, for example, this is neither the goal in Cirque du Soleil nor the goal in robot soccer. Although efficiency conditions weren't implemented or proved, let's look at what interesting conditions might be considered efficiency for each of these domains.

For Cirque du Soleil the goal is entertainment value and awe. One way to achieve this is to have very narrow margins, as illustrated by the juggling example earlier. So an efficiency condition might be that you want to minimize the minimum distance between the robots as well as preferring fast motion over slow motion when the robots are close. In addition, you could also maximize the maximum distance between the robots. Combining these would create fast motion that may start slow and separated but culminates in a near collision. This would be very exciting and it seems it would be tricky to achieve a successful pass in this case. This condition could be given as an upper bound on the minimum distance between the agents or an upper bound on the distance between them when the catch occurs. This would be difficult to prove in the general motion case. If both robots are stationary before the shot and the catcher only moves according to a simple path (straight lines, parabolas, or along a single circle) the shooter could conceivably calculate a path for the catcher to achieve this efficiency condition.

Another interesting condition that would provide entertainment value is to have the catcher moving quickly when it catches the ball. This gives a similar appearance of having achieved a fantastic coincidence of motion. This could be given as a lower bound on the catcher's motion when it catches the ball. This condition is easier to achieve by itself because the shooter can simply choose a point that is both safe and relatively far from the catcher, forcing it to accelerate for a long time.

For robot soccer, the goal is to position the ball such that a teammate can catch it before an opponent and also hopefully score a goal. This scenario isn't very interesting when talking about only two robots who are on the same team. A stationary object could be added to represent the opponent goalie (adding a moving goalie moves into the realm of planning and adversarial models). Efficiency in this case would be to have the ball land in a location where it can be immediately shot to the goal and is also reachable by the catcher. This could be accomplished by computing the intersection of the safety region as before and a new region which represents places from which a goal can be scored. A procedure using occluded regions similar to [2] could be used.

4 Stepping Stones

In order to make this very general and complex problem more manageable, we divided the problem into three areas of variability that can all be simplified to different degrees. These simplifications were used as stepping stones to make progress towards proving safety properties of the most general system. The three categories are as follows:

1. Cannon
    - Fixed vertical angle.
    - Adjustable vertical angle.
    - Fixed shot power.
    - Adjustable shot power.
2. Robot Motion
    - Positions fixed.
    - 1D motion (line of motion fixed from the start).
    - 2D linear motion (can only change direction when stopped).
    - 2D curvilinear motion (motion along continuous and connected arcs).
3. Robot Motion before Shot
    - Receiver stationary.
    - Receiver moving.
    - Shooter stationary.
    - Shooter moving.

Furthermore, for all of the problems where motion is allowed, it can either be the case that the catcher will have to observe the ball to determine where it will land or that it is told the trajectory by the shooter. As a further simplification of these problems, we assumed that the catcher doesn't have to wait to observe the ball before moving to the ball's landing location. Without these simplifications, the controller of each robot would have to account for the delay that the catcher will experience. Given some worst-case bound on how long it will take the catcher to decide when and where to move, the shooter can still determine the safety region.

The various cannon stepping stones are simple to think about relative to the other two categories. Having fixed angles and only variable power means that the shooter only needs to consider the range of the ball along a single line. The motion of the catcher can still complicate the computation of the region from which this line is reachable. Having adjustable angles adds complexity primarily because the shooter can make itself into an obstacle by shooting to opposite sides of itself on each turn. This of course is only valid for cases with two-dimensional motion.

The robot motion stepping stones are the most complex. One-dimensional motion is fairly simple with constant acceleration. Allowing multiple stages of change in acceleration would add more complexity. Two-dimensional linear motion, which forces the robots to only change direction while stopped and otherwise move along straight lines, is a nice compromise between the restictiveness of one-dimensional motion and the complexity of arbitrary two-dimensional motion. In this case, an optimal path for the catcher can still be derived by the shooter and can therefore be used to compute the reachability region. For arbitrary two-dimensional motion, which can be modeled as motion along continuos arcs that are also continuously differentiable, is difficult both to prove and to control. The shooter would have to make some simplifying assumptions about the path the catcher can take in all situations in order to compute a reachability region. This would be more restrictive than necessary. The catcher would also need to make simplifying assumptions in planning its path and adjust its velocity or path as it travels if it determines that it won't reach the target location at the right time. This motion is the most realistic but is also the most difficult to control and prove.

Finally, allowing the robots to move before the shot occurs adds complexity in proportion to the complexity of the motion. So adding motion before the shot for one-dimensional motion is less complex than either of the two-dimensional motion variants. The one-dimensional case, for example, is still determined by the displacement between the robots. This makes the analysis and control much simpler than if they could move in arbitrary directions. In addition to the evolution of the ball and robots together, there is also evolution for a non-deterministic amount of time of only the robots. During this new evolution period, the catcher has to constantly consider what would happen if the shot were fired in the next step. Additionally this complicates collision avoidance greatly.

5 Proof Techniques

In terms of the stepping stones, we only considered models where both robots are stationary before the shot and both robots have completely fixed cannon positions. Additionally, the robots always catch the ball in the center of their circles. The circles, therefore, only really function as collision buffers. Proving that the robots could also catch the ball anywhere in their circle would make the safety region slightly larger. For the one-dimensional motion case, this would amount to extending the reachability region by r in both directions. The stepping stones that we successfully proved are:

- Fixed cannon power, fixed robot positions.
- Variable cannon power, 1D linear motion.

Although the first case is very simple, some interesting things were learned from that proof which will be discussed in section 6. The second case consisted of the robots being constrained to one line for the entire game. This is illustrated in Figure 1. The shooting robot can choose any cannon power to use, which will change the range of the ball. This figure shows that B can only move left and right on the black line. Therefore, B can never move to the other side of A and A can never move to the other side of B.

Imagine that A is currently shooting. There are four places the ball can land: to the left of A, between A and B, on B, and to the right of B. A cannot shoot to its left or else the game is lost so we can assume the cannon is pointing to the right of A. The other cases are illustrated by Figures 2-4. Because both robots have a maximum acceleration they can use to catch the ball, there is also a limit to how far right A can shoot the ball and allow B to still catch it. The region in which B can reach the ball with its center is implicitly described by this formula:

$$B_x-\frac{1}{2}a_A\cdot(2\cdot C_v(A)C_z(A)/g)^2 \leq A_x+2\cdot C_v(A)^2C_z(A)C_x(A)/g \leq B_x+\frac{1}{2}a_A\cdot(2\cdot C_v(A)C_z(A)/g)^2$$

This is termed *reachability*. The center term is derived from the range of the ball with initial velocity $C_v(A)$ since $2\cdot C_z(A)C_x(A) = \sin(2\theta)$ where θ is the angle at which the ball is shot. The terms on the left and right are derived from allowing B to accelerate with maximum acceleration in either direction for the duration the ball is in the air.

Again, the region is automatically safe in terms of collisions if the ball will land at or to the right of B. If it lands to the left, however, A also needs to ensure that B is able to stop before it hits A after catching the ball. This can be expressed as the initial position of B minus the distance travelled to catch the ball minus the stopping distance given maximum braking and the velocity B reached to catch the ball. Since the distance B will travel to catch the ball is known to be the distance from B's current center to the landing location (by the assumption that we are actually only using the center of our robots to catch), the braking distance is what we need to maximize to find the worst case. Because B always starts moving when the ball is shot and can only pick its acceleration at the start, this automatically determines its final velocity. The acceleration that B must choose is:

$$2(A_x + 2\cdot C_v(A)^2C_z(A)C_x(A)/g - B_x)/t_c^2$$

where $t_c = 2\cdot C_v(A)C_z(A)/g$ is the time the ball returns to $z = 0$. So the safe region for this case is additionally constrained by the following formula (after simplification):

$$2\cdot C_v(A)^2C_z(A)C_x(A)/g - (2\cdot(A_x + 2\cdot C_v(A)^2C_z(A)C_x(A)/g - B_x)/t_c)^2/(2\cdot a_B) \geq 2r$$

This is termed the *collision* formula. A's choice of $C_v(A)$ is limited unconditionally by the reachability formula and it is limited by the collision formula if $C_v(A)$ would cause the ball to land between A and B.

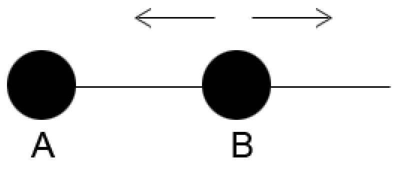


Figure 1: 1D motion problem.

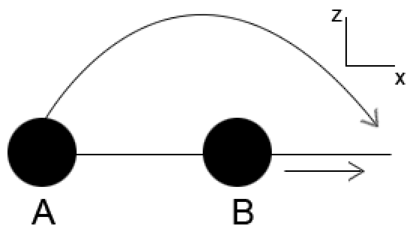


Figure 2: 1D motion problem.

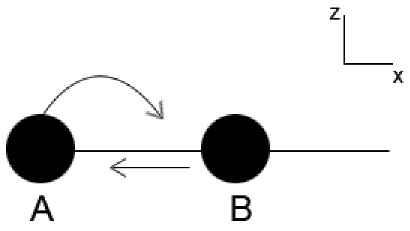


Figure 3: 1D motion problem.

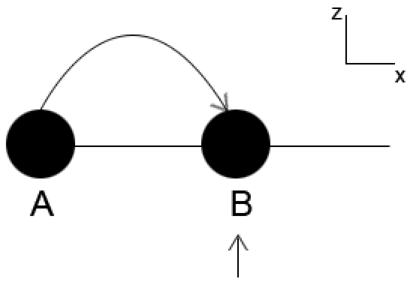


Figure 4: 1D motion problem.

Given these constraints on A's behavior, we can now prove that no collisions will occur and the ball will always be caught. Because reachability always applies, for either robot shooting, it is very straightforward to prove that the ball is never dropped. In order to prove that there are no collisions, we are most interested in choices of cannon power for which the robots have to move towards each other. During the proof of this system, we don't know what velocity A chose, we can only know that A obeyed these constraints. An important step of proving safety against collisions is to condition on which region A chose. We know A chose the reachability region, but we can further divide that into two disjoint regions: the ball will land between A and B or the ball will land on or to the right of B. Together these cover every possibility in the reachable region, so as long as we prove both are safe we have proved that all choices A can make are safe. With this conditioning, however, proving collision safety is very simple. The former region can be proven to be safe because we now know for sure that the collision formula was a factor in A's decision. The collision formula ensures that B will not hit A after catching the ball. The latter region is trivially safe because B is moving away from A.

6 Challenges

Recall from section 3.1 that one logical formulation of the safety condition for the ball to never be dropped is $(IR_A(t) \rightarrow b_z(t) \geq 0) \wedge (IR_B(t) \rightarrow b_z(t) \geq 0)$. When this is used to constrain the evolution of the differential equation describing the ball, the ball is only allowed to go into the region $z < 0$ if it's outside the imaginary cylinders of both robots.

Now consider a problem where everything is fixed and both robots are always stationary. The cannons of each robot are pointed such that the ball will hit the center of the other robot. In the process of proving this system, we can solve the differential equations describing the ball's motion ($b_{az}(t) = -g$) to obtain a parabola:

$$b_z(t) = C_v(A)C_z(A)t - \frac{1}{2}gt^2$$

We can then easily solve this equation to get that the time at which the ball hits $z = 0$ again is $t_c = 2 \cdot C_v(A)C_z(A)/g$. All that's left to prove safety is to show that the domain of the solution is $t \leq t_c$ because for all $t > t_c$, there is actually a point where the solution violated our domain constraint. This is true because the ball must pass through the center of the receiving robot's bounding circle, so when it goes below $z = 0$ it must still be in the robot's circle.

This condition can be expressed logically as:

$$\forall t(t > t_c \rightarrow \exists s(s \leq t \wedge \neg((IR_A(s) \rightarrow b_z(s) \geq 0) \wedge (IR_B(s) \rightarrow b_z(s) \geq 0))))$$

The trouble with this is that it mentions two quantifiers and a lot of variables. This makes it computationally intensive (exponential time in the number of variables) to compute the validity of this formula. Using a similar argument as the proof described earlier, we can divide the solution into two regions: $t \leq t_c$ and $t > t_c$. Now we can assume that the consequent of the implication is true under the branch where we assume $t > t_c$. However, this quantifier is still computationally difficult to directly prove. It's impossible to choose a witness t_w for which this is always true. For any witness we might choose, there is a $t_c < t_s < t_w$ which the witness fails to cover.

Although there might be another approach to proving this type of evolution domain constraint, we found that it is much easier to instead change the hybrid program. Instead of using a complicated evolution domain constraint like the one we just discussed, we can instead explicitly restrict the ball to stay at or above $z = 0$ and then test that it is always in a robot's circle at $z = 0$. This describes the same safety condition because anytime the ball is in one of the circles at $z = 0$ it will not fall through and anytime it is outside both of the circles at $z = 0$ it is about to fall below the robots and be dropped.

7 Conclusion

We have developed a hybrid program for describing a game of catch between two robots constrained to 1D motion with the ability to adjust the power with which they shoot the ball at each other. We also proved that this system is safe in two ways: the ball is never dropped by the robots and the robots never collide. We showed that evolution domain constraints on differential equations can be difficult to deal with when they involve implications. It can be very difficult to show that the implications were violated for certain parts of the solution without using quantifiers. The solution we found to this problem is to reformulate the hybrid program to move the conditions and tests outside of the evolution domain constraint.

References

[1] A. Platzer *Logical Analysis of Hybrid Systems: Proving Theorems for Complex Dynamics.* Spring-Verlag Berlin Heidelberg 2010.

[2] A. Colin *Safe Ball Passing in RoboCup.* http://symbolaris.com/course/fcps13/projects/acolin.tgz.

[3] K. Sukvichai, T. Ariyachartphadungkit, K. Chaiso *Robot Hardware, Software, and Technologies behind the SKUBA Robot Team.* Springer-Verlag Berlin Heidelberg 2012.

[4] J. Biswas, J. Mendoza, D. Zhu, B. Choi, S. Klee, M. Veloso *Opponent-Driven Planning and Execution for Pass, Attack, and Defense in a Multi-Robot Soccer Team.* Proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems 2014.

[5] Y. Nunome, K. Murakami, M. Ito, K. Kobayashi and T. Naruse *A Method to Estimate Ball's State of Spin by Image Processing for Improving Strategies in the RoboCup Small-Size-Robot League.* The 18th Annual RoboCup International Symposium 2014.