Perception and Strategy Learning in Robot Soccer

Yongbo Qian, Alex Baucom and Daniel D. Lee

Abstract—Soccer robots need to reason efficiently and make real-time decisions in an uncertain game. However, with more objects of interest being changed to white and games being pushed towards variable lighting conditions, advanced vision algorithms are crucial for robot’s ability to perceive new surroundings in real-time. Upon accurately perceiving a dynamic game environment, intelligent role switching, team positioning, and path planning methods are also key to winning the game. We explored some novel techniques in lighting-invariant perception and high-level team strategies which allowed our team UPennalizers to stay competitive in the Standard Platform League (SPL).

I. INTRODUCTION

In order to accomplish the goal that “a team of fully autonomous humanoid robot soccer players shall win a soccer game against the winner of the most recent World Cup by 2050”, the RoboCup competitions are evolving towards more realistic game scenarios by applying challenging rules in various disciplines. In the Standard Platform League for example, the outdoor league was first introduced in 2016 to encourage games under natural light, and such uneven lighting will continue being the favorable lighting condition in 2017. The 2017 competition is also going to start a mixed team tournament that might be played 10 vs 10 on a larger field. Those rule changes pose significant challenges in both areas of perception and strategy in robot soccer. In order to step up to these challenges and having a team of robots play well in a dynamic, uncertain environment, this paper presents some learning algorithms for robots to adapt to the lighting changes and reason about real-time team behaviors during the game.

We continued our effort in developing a robust and efficient perception system on humanoid soccer robots. The manually tuned color look-up table approach [1] became less effective at seeing the new white ball and goalposts, especially under natural light. Instead, we developed a new perception pipeline starting with field color detection and then extracting other key elements on the field using non-color features.

As the first vital step in this pipeline, an accurate and fast green classifier could provide contextual information and more computational capacity for other object recognition routines. In this work, we explored an approach that projects the 3-dimensional color space of the image pixels to a lower dimensional representation which could preserve the linear separability of green and non-green classes, while discounting the effect of illumination variation. This was implemented using Fisher Linear Discriminant analysis (LDA) to discriminate the color features affected by lighting changes. The projection maximizes the ratio of between-class variance to within-class variance in the training process. The projection direction can be also learnt in an online self-supervised manner to adapt to the current lighting conditions.

On the team strategy side, we aimed to develop a strategic positioning scheme for our robots. Player’s intelligent positioning on the field is important in the game of soccer. Even if a player does not have the ball, where they choose to move to can have a dramatic impact on the game. A supporting player near the goal can place themselves in a position to recover a rebounded shot, a defensive player can place themselves in between the ball and the goal to deflect an opponent’s shot, and so on. Programmatically defining such strategic positioning is difficult and testing it on real robots is even more difficult. For this reason, a simulation was designed for the UPennalizers team with the goal of designing and testing new positioning strategies. Inspired by the recent success of DeepMind’s AlphaGo, we also wanted to test simulation self-play to see if the computer could learn better strategies than what we could hand program.

II. PERCEPTION

Although previous competitions were all indoor under static lighting sources, researchers in the RoboCup community have already done a lot of work to investigate vision algorithms that could be immune to the lighting changes. There are three viable ways [2] to solve this lighting variation problem:

1) Apply algorithms to improve color constancy at pixels [3]. The idea is to form an illumination invariant color space so that color space distribution could remain similar when lighting varies, and resulting color segmentation and the rest of the color-based image processing tasks could still be performed without a problem.

2) Apply algorithms to improve color constancy over time. This could be achieved by using some techniques to correct the color classification in run-time to improve the reliability of color feature and have it adapt to the current lighting condition. An exemplary work was from Sridharan [4], which used KL-divergence measure of the color space distribution to determine and adapt to three illumination conditions. Houliston et. al. [5] proposed another feasible approach that used the feedback from object detections to add missing color class information to the lookup table.
3) Use additional visual features besides color for objection detection and tracking. Some teams have already abandoned color and completely relied on edge, shape or other features instead. While this approach showed some promising results, those features might require more time for computational processing and contain less representative information of the image than color potentially could.

In this work, we implemented the ideas from both 1) and 2) on field color detection. Most teams from SPL [6][7] use histogram analysis for this task, based on the assumption that the image is mostly covered by the field. We intend to generalize to more scenarios and improve the efficiency through dimensionality reduction. With field color being detected, algorithms from our previous work [8] could be applied to extract the field boundary lines and the perform other objects (ball, line) recognition routine within field boundary.

A. Log Chromaticity Projection

Since inconsistent green distribution may appear across the unevenly lit soccer field, we need to first do some transformation on the RGB space to extract the color content without intensity (chromaticity). Using the L1 norm on the G channel as chromaticity coordinate would work for green detection [8], but we want to generalize our approach to other colors as well. Therefore, without favoring any particular channel, we use log chromaticity projection proposed by Finlayson et. al. [9].

We first divide the 3D RGB triplets by the geometric mean, then take the logarithm to generate ρ which carries all three components of chromaticity:

\[
\rho_k = \log \frac{R_k}{\sqrt[3]{\prod_{i=1}^{3} R_i}} \quad k = 1, 2, 3
\]

Since in log space, ρ is orthogonal to the unit vector \(u = \frac{1}{\sqrt{3(1+1+1)}}\); thus only two coordinates are independent. We could then project ρ using a 2 × 3 orthogonal matrix \(U\) determined by eigenvalue decomposition to form a 2D representation \(\chi\) for this log chromaticity space:

\[
\chi = U\rho
\]

B. Linear Discriminant Analysis

We want to further project the 2D chromaticity feature space to 1D, with the objective of detecting the field color rather than forming an illumination invariant image. Therefore, we could simply perform a two-classes (green and non-green) linear discriminant analysis (LDA) to achieve dimensionality reduction while preserving the class discriminatory information. The full derivation of LDA can be found in [10]. The main idea is to find a optimal projection vector where the ratio of between-class scatter and within-class scatter is maximized, so that the class separation is maximized.

C. Results & Comparison

We collected a dataset consists of our field carpet using NAO’s on-board cameras in both indoor and outdoor environment under various lighting conditions. We first trained our LDA by manually selecting green and non-green areas in the image shown in Figure 2 left. The red line in figure 2 right shows the linear boundary that separates green and non-green classes in 2D chromaticity space \(\chi\). If we project \(\chi\) onto a line orthogonal to that red boundary line, the separability of those two classes could still remain.

We then used other images in the dataset for testing. With the same linear boundary learnt in the training process, the green classification worked surprisingly well in both static indoor field and outdoor carpet with strong shadow (Figure 3). This promising result suggests that this method could work consistently well in some new, different environment.

We could also extend this mechanism to the images from bottom camera (Figure 4 left) to classify green and white classes to help objects (line, ball) detection. Similarly, we perform descrimitive analysis on the 2D log chromaticity space \(\chi\) (Figure 4 middle). Quadratic discriminant analysis (QDA) yields to a better result here, using different covariance matrix of each class. The red curve in figure 4 middle separates green and white classes, achieving good classification result in figure 4 right.

To generalize the discriminant analysis for better classification result, we could use the kernel trick [11] to learn a non-linear mapping from the input space to a high-dimensional feature space to obtain linear separability.

Figure 5 compares our proposed method with previously used color-table method. We simulated two illumination
conditions, normal (top) and bright (bottom) in our lab environment. We trained both our linear classifier on 2D chromaticity feature space and color look-up table using Gaussian Mixture Model on YUV channel under the normal lighting condition first. When the lighting remained the same, both methods worked well (top middle & right). However, when the illumination condition changed to bright, the green classification result (bottom middle) for our proposed method was still accurate, while the color-table method failed to classify lots of green pixels on the field. This comparison shows clear advantage of our approach.

D. Online Self-Supervised Learning

Although the results shown in the previous section are all reasonably well, it is still impossible to find a completely illumination invariant feature space. A good training set consisting green samples under different illumination is desired to achieve a better classification result. For example, if we only used the brightest pixels on the field in figure 5 bottom left to train the classifier, the classification result could be really poor if the lighting significantly changed to dark (Figure 6 top left). Therefore, it is within our best interest to extend our method an online self-supervised learning approach so that the color classifier could adapt to the current lighting condition. This could reduce the effort of on-site vision calibration before games, and eventually allow robots to play on the field in any environment.

Some ground classification research [12] combines geometric estimates of the ground model with appearance-based classification to achieve an online self-learning scheme from vision. However, it is difficult to recover depth to estimate ground plane for soccer playing NAO robots. Structure from motion (SfM) is impossible to perform due to the lack of translational speed and computation capacity in real-time. The depth cannot be estimated using stereo vision neither because the fields of view of Nao’s top and bottom cameras do not overlap.

Our online self-supervised learning approach for soccer-playing robots is pictured in Figure 6. If the ratio $\alpha$ of classified green pixels to the total pixels within field boundary is lower than a certain threshold, which certainly indicates significant lighting variation, our online learning scheme will be triggered. Instead of having fixed pre-trained projection vector, the linear classifier will be re-trained to slightly
modulate the projection vector which separates green and non-green classes. The non-green samples for this re-training process is selected from pixels above robot’s horizon. For the positive samples, we need to add more missing green pixels on the field due to misclassification from the previous classifier. We calculated the posterior probability $P(g|x)$ for $k$ point $x$ within the field boundary, which indicates the possibility of a point $x$ belongs to class $g$ (green). Since the classification was using the maximum a posteriori (MAP) principle, the points $x$ which satisfy $P(g|x) > 0.5$ were already classified as green, we then need to add $x$ which $P_l < P(g|x) < 0.5$. $P_l$ can be determined by $\alpha$: lower $\alpha$ means we need to add more misclassified pixels for re-training, then the smaller $P_l$ should be used for the lower bound. Both positive and negative samples are still in 2D chromaticity space for re-training. After the re-training process, the projection direction for LDA will be adapt to the current lighting condition. To avoid field color changes when robot looks out of the field or close to other robots, a large adjustment of the projection vector is not allowed.

III. TEAM STRATEGY

In the field of robot soccer, there are many different approaches to team and individual strategy. Some of the simulation leagues can have very complicated layered strategies [16] but since the simulation we want is designed to benefit the SPL league, a simpler strategy was needed. Many teams use artificial potential fields [13] to handle velocity generation and path planning during the soccer match [18] and other work has been done using Voronoi tessellations [14] to do this more intelligently; however, these types of approaches still require a way to define a goal position to reach. Defining this goal can often require a lot of checks and switches to cover all cases and we were more interested in having the robot learn where it should position itself given the current environment. Some other work in artificial potential fields [17] illustrated how to use potential fields to define additive heuristic functions which could be used to position players without having to define a specific goal position. This method was used as a starting point for this work.

A. Simulation Framework

The simulation was designed with two primary goals in mind. First, it should be easy to prototype and test different behaviors and strategic positioning algorithms. Second, it should be easy to use a learning algorithm with the simulation. To accomplish these goals, the simulation was designed in MATLAB and was made very configurable. In addition, the simulation can accept a set of functions that define strategic positioning algorithms for each player so that new functions can be easily tested.

The simulation contains four main components: the game controller, the ball, the world, and the players. The game controller handles all of the high level simulation details. The ball, world, and players are all defined as objects which the game controller instantiates and updates at every time step. The game controller also utilizes several other utility functions that handle field animation, detecting collisions, and resetting the ball or players when they go out of bounds.

The ball object is fairly simple. It keeps track of the ball’s location and velocity on the field as well as the last player to touch it (for enforcing out of bounds rules). When the ball is kicked or collides with an object, it is given a velocity based on the kick strength or the collision physics. The ball update method then simulates the ball motion at each timestep and performs deceleration on the ball based on a configurable friction parameter.

The world object contains all of the information about the state of the world. It gathers data from each player and the ball and aggregates all of it into ‘observations’ that the players can make about the world. This centralized world model allows the simulation to run significantly faster than having every player object make these ‘observations’. The world object also is fully configurable with regards to the noise of player observations. Parameters can be set to allow players to have a perfect observation of the world or a very noisy observation. This allows for testing the robustness of different positioning methods to noise in the environment and observations. Lastly, the world object also handles dynamic player role switching. Once again, this computation was centralized in order to speed up the runtime of the simulation. However, care was taken to ensure that the algorithm was implemented in a way that would not make the role switching behave any differently than if the computations were performed locally on each robot.

The player object is the most complicated class in this simulation. Each player makes an observation by querying the world object and then uses this information in a behavior function to determine what it should do. The behavior is primarily responsible for setting a desired velocity. The actual velocity is then computed from the desired velocity based on acceleration and the maximum velocity of the player. The behavior function for each player is a simple finite state machine that has states for searching, approaching the ball, kicking, and moving. The move state calls a function handle that was given to the player at initialization which allows for specifying different movement strategies for different players. This function is responsible for taking in information about the world and specifying a desired velocity for the player to execute.

B. Potential Field Function Implementation

A potential field function approach was chosen to test the simulation and its learning capabilities. Potential field functions are quick to compute and offer a lot of flexibility in defining positioning strategies. Our implementation defines one positioning function and the weights of the function change depending on the players role. This one function is made up of several heuristic functions that define how attracted or repelled a player should be from different features such as the sidelines, the ball, other players, etc. Since these functions are all additive, it is straightforward to create more heuristic functions and add them to the main function.
The key aspects of the potential field function that have to be hand-designed are the metrics to measure as part of computing the potential field. Starting from the work of [17] and building upon it through trial and error, we came up with the following metrics to use: distance to each sideline, distance to each teammate, distance to the ball, distance to the attacking shotpath (defined as the vector from the ball to the center of the attacking goal), distance to the defending shotpath (defined as the vector from the ball to the center of the defending goal), distance to the attacking goal, distance to the defending goal, and an interaction term between the distance to the attacking shotpath and the distance to the ball.

This interaction term was a key breakthrough in defining a very specific behavior that would have otherwise been very complicated to create. The problem that necessitated this term was if the attacking player was in front of the ball and trying to get in position to shoot. The attacking player was attracted to the ball, but had no way to know to go around the ball and position itself behind the ball to shoot. Similarly, if a defending player was in front of the ball, it would try to get back to a defending position, but would not know to avoid the ball. These cases could be solved by adding in several more terms to the potential field function, but capturing this with one term is much more efficient. Using the equation

$$U_{att} = \frac{1}{2} gain(H(d) - H(d - r))(d - offset)^2$$

$$U_{rep} = \frac{1}{2} gain(H(d) - H(d - r)) \left( \frac{1}{d - offset} - \frac{1}{r} \right)^2$$

Where $H$ is the heaviside step function, $d$ is the computed metric, $r$ is a parameter to define the range of effectiveness, $gain$ is a parameter to define the strength of this heuristic, and $offset$ is a parameter to shift the minimum point in the field. Each heuristic has its own set of weights for each role and the total function is simply a sum of all the weighted attractive and repulsive functions. The functions for each role are machine generated before running the simulation for runtime optimization. During the simulation, each player only has to calculate the metrics previously mentioned and then pass them into the proper function for their role. Local gradient descent on the potential function is used to determine player velocity and they have reached the desired position whenever the reach the field minimum. Local minima is not a major concern due to the dynamic nature of the environment.

C. Learning Methodology and Initial Testing

While it is possible to hand tune parameters for each role of the potential field function, we wanted to see if the simulation could be used to learn better parameters. To do this, the simulation was modified to accept a vector of weights as an input and output a score that reflects how well each team performed. This score was based on the number of goals scored by each team, whether they were own goals, and how often players went out of bounds.

The Nelder-Mead simplex learning algorithm was chosen for learning since it was easy to implement and, most importantly, doesn’t need large numbers of function evaluations to converge. Due to the linear nature of the simulation, massive parallelization was not an option and so every effort was made to optimize the simulation code; however, a 10 minute simulation still required about 20 seconds to run on a single thread of an i5 desktop processor. This made it important to use an algorithm that minimized the number of required function evaluations.

The learning methodology is as follows. Two ‘default’ positioning strategies have been defined. Both programmatically choose a goal state for each role and then generate a
velocity to that goal. Default 1 uses a proportional controller to define the robot’s velocity and Default 2 uses a more standard potential field function [13] to generate a velocity. Prior testing shows that Default 2 behavior almost always beats Default 1 behavior when averaged over many games. The Default 1 behavior is used as the opponent team for all of the learning matches and Default 2 is used to test the learned behavior and ensure it isn’t overfitting. Additionally, the learned behavior is tested against the initial potential field function behavior to attempt to quantify the performance improvement.

To make the search space tractable, a layered learning [15] based approach is used to learn weights of individual roles and then combine them together later and do learning on the joint set. The full scale learning trials have not taken place yet, but qualitative analysis of small scale runs indicate that the learned positioning is improving the performance of the team as a whole. Intelligent behaviors such as the supporter moving to empty space in anticipation of a rebounded shot or the defenders working together to block more of the goal are appearing occasionally in testing, which shows promise for this method.

IV. CONCLUSION & FUTURE WORK

In this work, we proposed learning algorithms in both areas of perception and strategy. Applying linear discriminant analysis on 2D chromaticity feature space shows promising results in detecting the field color in a lower dimension representation. An online self-supervised approach is also explained for more robust field classification. This paper has also explained a simulation framework that was designed to test strategic positioning algorithms and perform learning on them. Strategic positioning using artificial potential field functions have been implemented in the simulation and appear to hold promise for improving the teams overall behavior.

The next steps for the future are to run the full scale learning trials and evaluate the results quantitatively. For perception, the online learning approach needs to be validated in outdoor fields and optimized to guarantee a fast and efficient runtime. For strategy, other types of strategic positioning algorithms could be tested and compared in simulation and implementation and testing on the actual robots could be pursued to fully validate the potential field method.

REFERENCES


[19] https://github.com/alexbaucom17/RoboCupSoccerSim/