RoboCup SPL 2018 rUNSWift Team Paper

Kenji Brameld, Fraser Hamersley, Ethan Jones, Tripta Kaur, Liangde Li, Wentao Lu, Maurice Pagnucco, Claude Sammut, Qingbin Sheh, Peter Schmidt, Timothy Wiley, Addo Wondo, and Kelvin Yang

School of Computer Science and Engineering
UNSW Sydney
Sydney 2052 Australia
http://www.cse.unsw.edu.au

Abstract. RoboCup continues to inspire and motivate our research in artificial intelligence and robotics. The 2018 UNSW Sydney team (*rUNSWift*) consists of undergraduate students, Masters and PhD students, alumni, and supervisors. *rUNSWift* has a rich history, with involvement in RoboCup for over a decade in the Standard Platform League (SPL). We work in research areas that include computer vision, localisation, locomotion, machine learning, and layered hybrid architectures. Major developments in 2018 include a robust system to handle dynamic lighting, significant improvements to the ball detector, a new field feature detection system, a restructure of the behaviours framework and the use of the torso to improve walk stability.

1 Introduction

Team *rUNSWift*, has been competing in the Standard Platform League (SPL) since 1999. Every year, we strive to improve the weakest aspects of our system and adapt it to new challenges presented by the committee through rule changes. In 2018, we focused on improvements to the black and white ball detection, efficiency of field feature detection, adaptability to natural lighting, robot detection and stability. We also revised our behaviours to incorporate rule changes involving the free kick, and refined the behaviour architecture to allow greater flexibility in role switching. These changes allowed us to remain competitive in the league, earning a quarter finalist position in the main competition and first in the mixed team tournament.

2 Team Organisation and Development Methodologies

Consistent team organisation and procedures have played a significant role over the past year and has allowed the team to reach the quarter-finals at RoboCup 2018.

The team maintained regular weekly meetings and testing routines that have been used in past years. We would meet once a week in person, with Google Hangouts for remote communication. At these meetings we would discuss team

Fig. 1: A subset of the 2018 team.

updates and technical direction for the next week. It also allowed team members to ask for assistance on difficulties they had experienced, fostering a culture of collaboration and support for other team members.

Our main form of communication is through Slack, a team-messaging platform, which allows messages to be sent out to specific members through the use of channels. This encouraged clean, organised team communication, allowing team members to only listen in to conversations of significance to them. Additionally our team maintains the code base in a git repository hosted on GitHub which enables effective collaboration with a large code base. We also utilise the GitHub wiki for internal documentation and GitHub issues to track ideas, issues and their completion.

We also maintained regular testing schedules. The simplest of these is a 'striker test', where a single robot and the ball are placed in set positions around the field, and the efficiency in scoring a goal is observed and compared against previous tests. This tests the integration of all our modules and improvements over the past week to ensure that changes are actually improving our overall soccer play. We also ran regular practice drills and matches involves multiple robots to evaluate team play, positioning, contention and obstacle avoidance.

3 Vision

The basic structure of the 2018 vision system remains the same as that of 2017 [1]. The image is colour classified, (Section 3.1), regions of interest are detected based on this classification (Section 3.2) and finally regions of interest are classified into objects by detectors (Sections 3.3, 3.4 and 3.5). This layout is illustrated in Figure 2.

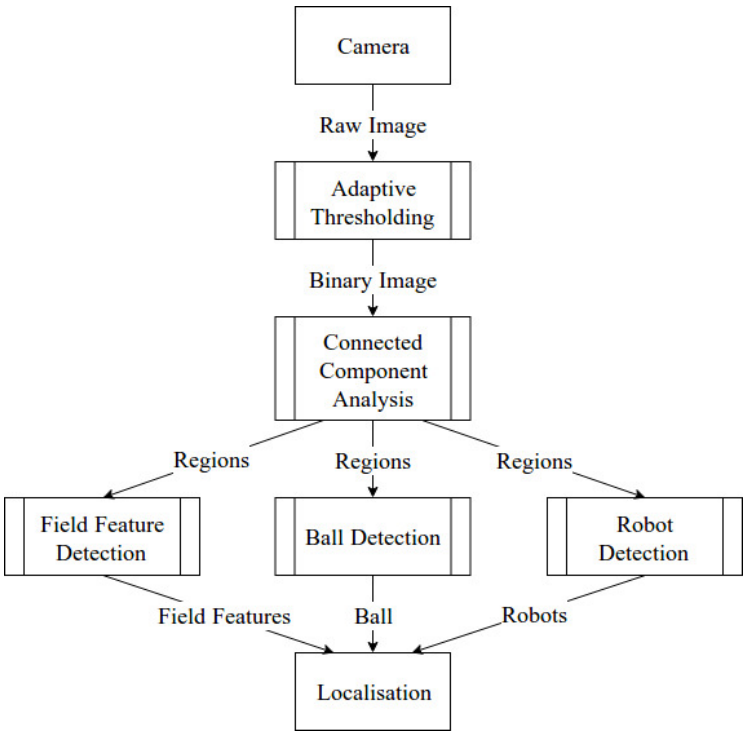

Fig. 2: Overview of the 2018 vision system.

The key improvements to our vision system were:

- **Dynamic Lighting** - Previously, we relied on hand colour classification, which struggled under dynamic lighting conditions. In 2018, we switched to a binary image, classified with adaptive thresholding (Section 3.1). Additionally, the camera's auto exposure setting was turned on to allow the robot to see where lighting varied across the field. Our vision is now robust to a wide variety of lighting conditions and no longer requires manual calibration.
- **New Algorithm to Detect Field Features** - The field features detector was redesigned to work with ROI, cutting the processing time of this detector (Section 3.3). Although the old system was effective at detecting the field lines, it was more computationally expensive. Consequently, we struggled to meet our goal of consistent 30 fps vision - required for the robots to be reactive in a real game. The new field feature system has largely alleviated this issue.
- **Improved Ball Detection** - We continued to improve the ball detection and re-engineered some of the features that were used. This resulted in an overall improvement to the detection rate and efficiency of the detector. The dynamic time management system used for handling complex detection cases was also refined to reduce the average and worst case time to run 3.4.

– **New Robot Detector** - Improvements were also made to the robot detector, which was primarily used for obstacle avoidance at close range. Our changes increased the range and accuracy. This allows our robots to determine a kickoff play, actively avoid moving directly toward opponents and attempt to maintain possession of the ball during contention (Section 3.5).

3.1 Adaptive Thresholding

In 2018 we converted the vision system to work with binary images to be more robust to varied lighting. This is done with adaptive thresholding, which is an effective method for calculating the threshold dynamically in scenes with uneven levels of brightness. This is a significant improvement on the static colour calibrated tables used by the team in previous years.

We utilise an efficient implementation of adaptive thresholding using the integral image [2] to achieve real-time thresholding on the Nao. The algorithm determines the average y value (in the YUV422 image) of the pixels within a square window surrounding each pixel. If the value of the centre pixel is sufficiently lower than the average within the window it is set to black, otherwise it is set to white. This process can be performed efficiently, regardless of window size, with the integral image.

Fig. 3: An example of the adaptive thresholding process. White pixels are considered light whilst green pixels are dark.

As this method does not rely on specialised colour tables the exposure of the image may be allowed to vary. This has allowed us to enable auto exposure on the camera, further improving our system's robustness to changes in lighting. Target exposure was brought down to minimise motion blur. Furthermore, manual colour classification of the environment before each game is no longer necessary, reducing the setup time. The main downside is the loss of colour information, previously used to determine the field boundary.

3.2 Regions of Interest

Currently, everything of interest on the RoboCup Soccer SPL field is primarily white. In the 2017 vision system, regions of interest (ROIs) were created by finding white areas in a colour calibrated image. Minimal changes was required to move to the binarised image in 2018. These ROIs are located using a connected component analysis algorithm [8], with some modifications.

Since we just need the bounds of the regions, only the first pass of the algorithm is performed, recording the axis aligned extents of each group of pixels. When this is done connected groups can be quickly merged by taking the maximums of these bounds.

This algorithm tends to produce large groups containing most of the objects in the frame due to field lines connecting other white objects, such as the ball. As this is undesirable for the purposes of locating important objects the we split the scene into a grid, and prevents components from being connected across grid lines. Unfortunately, this may split useful objects like the ball so the system merges nearby groups where the edged of the bounding box have a sufficient ratio of white to not white pixels. This achieves bounding boxes that are reasonably likely to contain interesting objects, tightly bounded when those objects are not close to other white objects.

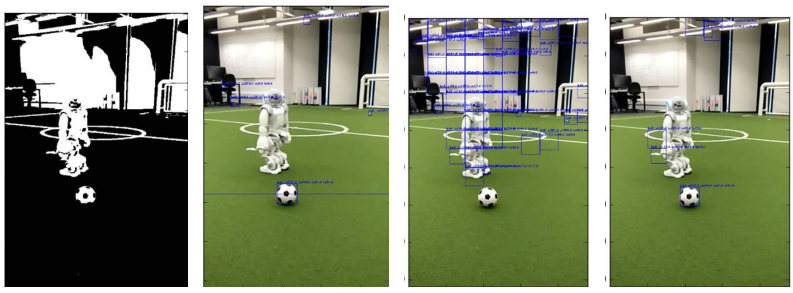

Fig. 4: The region finding system. From left to right: The binary image. Regions generated without grid splitting. Regions generated with grid splitting, culled to show only ball like regions. Final regions generated with grid splitting and merging, culled to show only ball like regions.

3.3 Field Features

The new field feature detector takes advantage of the ability of the ROI system to locate line segments. All lines of the field are white, such that they are detected and segmented by the ROI system 3.2. Some computationally efficient checks are performed on each region, allowing us to determine whether it contains a field feature. This is more efficient than the previous approach, which scanned the frame for candidate points and ran RANSAC on these points to locate field

features. The new detector is compatible with the binarisation of vision as it considers the light pixels as white (for lines) and dark pixels as green (for the field).

Our system detects regions that appear to contain line segments, corners, T-intersections and curved line segments. Curved line segments are combined together to form centre circle candidates, which are further checked for a line passing through the middle of the candidate circle to provide confirmation and orientation. If line segments intersect near the detected corner and T-intersection features these features can be confirmed, or even new, possibly occluded, ones detected. These lines and more distinct field features are passed to localisation to determine the position of the robot. This system is outlined in figure 5.

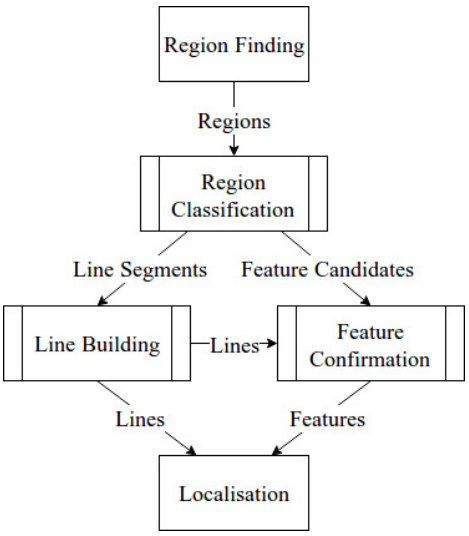

Fig. 5: Overview of the 2018 field feature detector.

3.3.1 Region Expansion There are times when a region does not contain a sufficient portion of a feature to identify it and accurately distinguish its direction. In such cases, we attempt to expand the region to obtain more information on the feature. For corners, this ensures the feature is fully included in the region. For curves, a larger curvature segment can be detected from the region. We can't expand a region arbitrarily, as in some situations, particularly near the goal box, features are located close to each other and expanding arbitrarily would put several features in a single region. We perform the expansion by combining some neighbouring regions, whilst trying to ensure that no other features have been included into the region. This additional step is effective for increasing the accuracy and consistency of detecting field features. The following pseudocode describes how this region expansion is performed:

```
For every region
    Count number of white pixel segments along the region
        edge
    Find all neighbouring regions
    For each neighbouring region
        Create region containing the original and
            neighbour region
        Count number of white edge sections in this region
        If number of white edge sections in original
            region
        AND combined region is same
            replace the original region with the combined
                region
```

3.3.2 Line Segments The line classification system primarily examines the borders of regions. If exactly two white segments are found along the region boundary, they are assumed to be the ends of a line (Figure 6). We then check that the width of the two ends are approximately the correct size for a field line based on an estimated projection from the robot's pose. We also check that the two ends are connected by following the edge of the white area. If these checks pass the system then attempts to determine if the segment is a straight line, a curve or a corner.

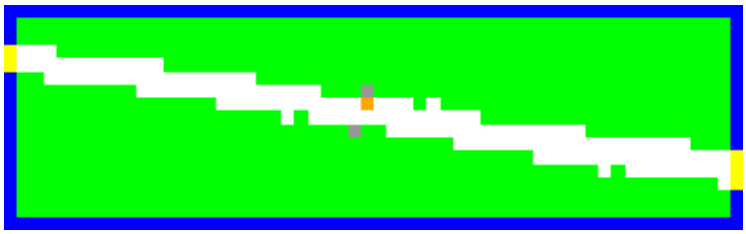

Fig. 6: An example of a region classified as a line. Green areas are dark pixels, white areas are bright pixels. Blue pixels are dark areas on the border and yellow pixels are bright areas on the border. The orange pixel is the centre point between the line ends. Grey pixels are the extents of the line identified by the curve check.

3.3.3 Curve Segments A region must be sufficiently large to be classified as a curve. Very small regions are always marked as lines. To check if a segment is a curve, the pixel half way between the two line ends is checked. If that pixel is green, this is a curve. If the pixel is white, the edges of the line around the centre point are found. If these are unevenly spaced from the centre point, the region is marked as a curve. If it is evenly spaced, it is considered a line. This is

shown in Figures 6 and 7. If the segment passes these checks it is finally checked to see if it is merely curved or is an actual corner.

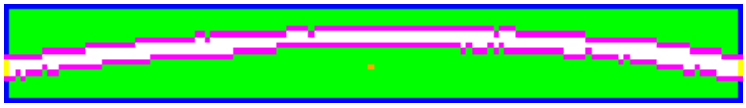

Fig. 7: Curve detection on a curve. Most colours are as in Figure 6. As the centre point is not in the line there are no grey pixels. Pink pixels show the edges of the curve identified.

A region marked as curved must finally be checked to determine if it is actually a curved line or really a sharp corner. To do this the point along the edge of the white part of the line segment farthest from the line between the two ends is found as tips, the middle point of two tips is the intersection of this potential corner. If half of the edge points of white part is one a straight line, the region is marked as a corner (Figure 8). Otherwise it is left as a curve.

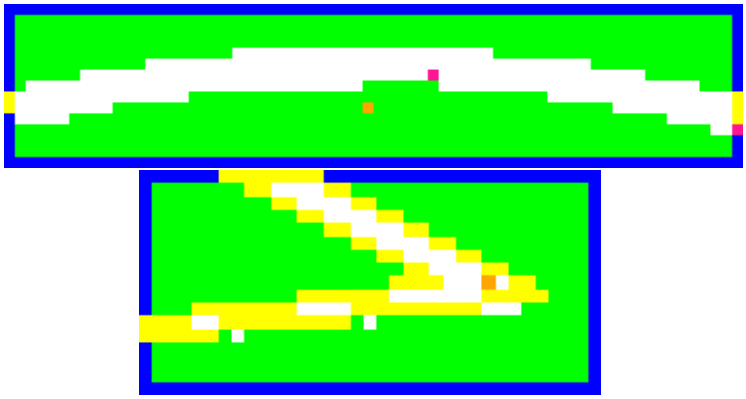

Fig. 8: Examples of the corner check being performed on a curve (above) and corner (below). Here the orange pixels represent the "point" of the corner. The yellow lines represent the straight lines identified as composing the corner. The pink dots represent the best attempts to find the ends of those lines on the curve.

3.3.4 Centre Circle Construction Where curved line segments have been identified further checks are made to determine if the area can be confidently said to contain a centre circle. The possible centre circles corresponding to each curve segment are calculated. Line segments that lie on the edge of those circles are then identified. Three checks are made:

1. The total length of curved line segments along the circle edge must be above a threshold.
2. The total length of curved and straight line segments along the circle edge must be above a (higher) threshold.
3. There must be a sufficiently long merged line passing through close to the centre of the centre circle (i.e. the field centre line).

This is illustrated in Figure 9.

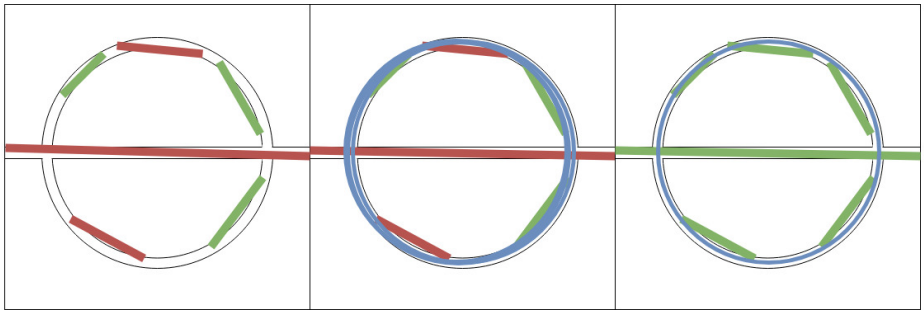

Fig. 9: Basic process of constructing a centre circle. Left: The detected lines overlaid on the true centre circle, with green for curves and red for lines. Middle: All possible centre circles based on the curved lines. Right: One circle selected, with all the lines related to it in green.

3.3.5 T Junctions If there are exactly three white segments along the edge of a ROI, it is potentially a T intersection. As before, the white areas are checked to determine if they are roughly the correct size and connected to each other. If they are, the are further checked by ensuring the line between two of the ends is entirely white pixels (the top of the "T"). Finally the straight lines between these two ends and the other end (the base of the "T") must not be entirely white pixels.

3.3.6 Feature Extrapolation The final step performed by the system is to check that the corners and T intersections it has identified match the lines it has found. The system may also extrapolate corner and T intersection locations it did not find in regions. For every pair of (merged) lines the intersection point is identified. A quality value is then calculated for that intersection. If a corner or T intersection is detected nearby a large quality boost is added. The attributes of the lines forming the intersection are then examined. Longer lines with ends closer to the intersection add more quality. If one of the lines passes through the intersection quality is added to it being a T intersection, while corner quality is penalised (and vice versa). The line based intersection checks are illustrated in Figure 10.

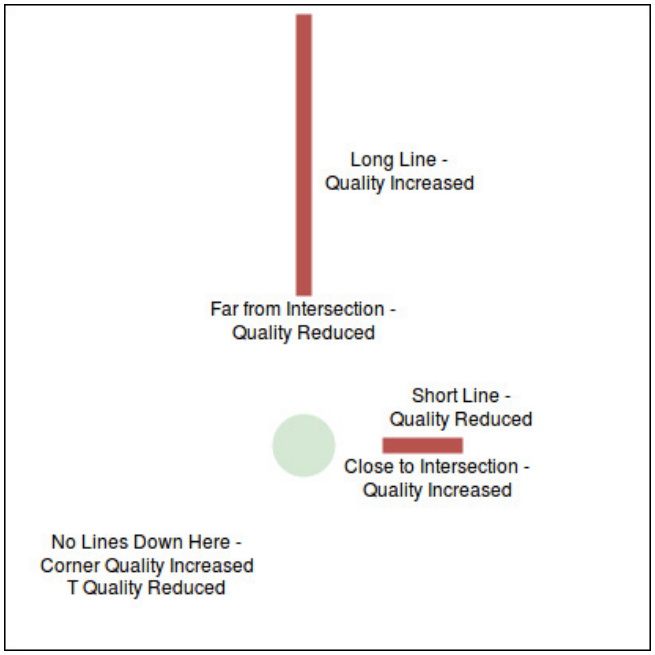

Fig. 10: An example of the quality modifiers used by the intersection detector. Lines are shown in red and the intersection point is shown as a blue circle.

3.3.7 Penalty Spot Penalty spot detection is an important feature for the localisation of a goalkeeper. When standing in the goal box it is surprisingly difficult for a goalkeeper to see any major field features. The goal box is often occluded by its own body, particularly the shoulder pads. Furthermore, to respond to the moving ball the goalkeeper must track the ball at all times, so in general nearby field features will not even be in field of view. The result is that our standard field features function poorly unless the goalie moves in a manner detrimental to its role.

The penalty spot is an effective feature to handle this case. The detector is only concerned with penalty spots that are very close to and in clear view from the goalkeeper, and is not enabled for other robots. Firstly, simple heuristics including the width and height of the bounding box and the projected size of the penalty spot are used to eliminate most regions at a low computational cost. Secondly, connected component analysis is applied to the region in the adaptive thresholded image. If any dark region fully encased in white is found the candidate is rejected. Finally, the region is passed through a Gaussian mixture model based classifier to eliminate remaining false positives. The machine-learning algorithm is the GMM used by Ball Detector explained in 3.4, trained to detect penalty spots instead of balls.

3.4 Ball Detector

Ball detection is a critical component of any robot soccer system. The 2016 transition to a black and white ball has made the task significantly more difficult. The ball can no longer be distinguished from other objects by its colour alone, instead requiring a more complex system that considers a candidate's appearance. This rendered the 2015 orange ball detector entirely ineffective. The 2017 ball detector adequately met this challenge primarily making use of the circular shape of the ball and its distinctive pattern. Despite this, the 2017 ball detector had a number of limitations and assumptions. The structure of the 2017 ball detector remained in 2018, with improvements made to address its limitations, increasing the reliability and accuracy.

Inside the ball detector, all regions are passed initially filtered through a ball candidate finder. The ROI finder at the vision level, described in 3.2 segments the white sections of the frame, however, these regions are not always a perfect crop of the ball. Namely, field lines, other robots and goal posts can be included in an ROI that also includes a ball. A tightly cropped and consistently scaled region of interest is desirable as it significantly simplifies the problem of ball pattern detection. Achieving a tight crop is the role of the ball candidate finder.

For "easy" cases cropping is done by Naive ROI (Section 3.4.1.1), while more complex regions require Blob ROI (Section 3.4.1.2). We also eliminate regions that are clearly not ball like due to size and shape. Scaling is performed in the region regeneration step (Section 3.4.2.3).

Next, region specific pre-processing (Section 3.4.2) is performed to adjust the image and make the major features of the ball candidate more salient. The white pixels are used to perform a circle fit, and we discard regions for which an adequate circle cannot be found (Section 3.4.2.2).

A ball candidate that reaches this stage undergoes two final heuristic checks to analyse the likelihood that it is a ball (Sections 3.4.3.1 and 3.4.3.2). Each of these checks give the candidate a score if passed. To be considered a ball a candidate must achieve a score above a certain score threshold.

3.4.1 Ball Candidate Generation

3.4.1.1 Naive ROI If a region is the correct size and shape for a ball, it is adjusted and considered as a candidate. The definition of correct size is calculated from the kinematics chain, given the ROI's position in the frame. The basic adjustments includes adding a small padding to the region in case the side or corner of the ball is lost and resizing the ROI to a fixed size. This is the most frequently occurring case in the game, where the ball is in the middle of the field with no obstructions.

3.4.1.2 Blob ROI When an ROI is larger than the expected size of a ball, there is a possibility of the ball being inside the region, but overlapping with other white objects. In this case, an attempt is made to create a tight region around the ball, by looking for the black blobs on the ball. A connected component analysis

is run on the binarised region to locate all black blobs within the region. If multiple ball blobs are detected, they are compared to find the blob with the highest possibility of being from a ball. This is achieved by comparing the density of black within the bounding box around the blob. The density is determined by dividing the number of black pixels in the blob by the product of the width and height of the rectangular bounding box around the blob. The blob with the highest density is considerd the most likely to be a blob from the ball in the region. If the blob's width to height ratio is close to 1:1, a new ball candidate region is created around it. This region is then passed through the remainder of the detection pipeline as though it was a naive ROI. This helps us locate the ball when it is adjacent or overlapping with other white objects, such as when the ball is in front of a goal post or robot or on a line.

3.4.2 Preprocessing

3.4.2.1 ROI Specific Thresholding To sharpen the ball pattern, a second pass of the adaptive thresholding is run, local to the region. This is similar to the one described in Section 3.1, although the brightness parameter is determined dynamically for each region, based on its content. Specifically, if the region is in the top camera, the threshold percentage is one tenth of the average brightness of that ball candidate. For regions in the bottom camera, a single pre specified value is used, because there will be less variance in a single frame.

3.4.2.2 Circle Fit The ball candidate is refined further by using a circle fit. Candidate edge points are generated by scanning from the left, top, and right of the region to find the first transition from black to white in the binary image. A scan from the bottom is not conducted, as the shadow of the ball makes often makes its bottom edge difficult to identify. A circle fitting algorithm is then used to determine the existence of a circle, along with its centre and radius. The following describes the algorithm used for circle fitting, for a single region.

```
Create 2D array, size of the region for centre point
    voting
Create 1D array for radius voting
Find edge points by scanning region from top, left,
    and right of region
For 100 iterations
    Randomly select three points
    Determine centre point and radius from points
    Vote on centre point
    Vote on radius
    IF max(centre point voting 2D array) > centre point
        voting threshold
    AND max(radius voting array) > radius voting threshold
        Found circle, with most-voted centre point and
            most-voted radius
```

3.4.2.3 Region Regeneration If a region has passed all the preceding checks it is considered a strong enough candidate to process further. The first step of this is 'zooming in' on the candidate region. If the region contains too few pixels when considered at the standard resolution of binary classification the classification is redone at higher resolution. The resolution is increased by the smallest factor of 2 that causes the region to contain the minimum number of pixels, or the maximum resolution of the Nao camera if needed. As a result, we are able to extend the range of the ball detection considerably, without having to process a high resolution image upfront in the stages of ROI generation 3.2.

3.4.3 Heuristic Checks

3.4.3.1 Triangle Check The regular shape of the pattern on the ball can be used to eliminate many non ball cases. To achieve this we check that the dark spots on the ball form a rough equilateral triangle. To detect these dark spots adaptive thresholding and connected component analysis is run specifically for the region. The parameters of these algorithms are, for this case, optimised to cleanly distinguish the black spots on the ball. Additionally, the circle fit is treated as a hard edge, preventing spots on the edge of the ball from being connected via the dark field. The black blobs found this way are checked to ensure they are of appropriate size and aspect ratio to be a ball spot, relative to the size of the circle.

The blobs that have passed these tests are then checked to determine if their centre points form an imperfect equilateral triangle. Each possible triangle is checked. As the triangle should be (roughly) equilateral this check can be easily performed by comparing the distances between each centre point. If all three distances are nearly the same they form a sufficiently equilateral triangle.

3.4.3.2 Gaussian Mixture Model If a ball region passes all the previous checks, it will be sent to the final classifier. Our ball detection classifier is a specially designed semi-supervised module composed of a Gaussian Mixture Model (GMM) [7] and a simple classifier.

Two GMMs have defined a set of Gaussian models of both real balls and samples of non ball images that pass the heuristic checks. These unsupervised, generative models have data efficiency advantages (that is, they require fewer examples to learn) over many other machine learning methods, such as convolutional neural networks[6]. When a new candidate is sent to the GMM, it will output the probability of each Gaussian in the model. Then, a simple Max A posterio algorithm is applied to determine the most probable classification by calculating the probability of each mixture and picking the most probable as the final output.

3.5 Robot Detector

As robots are significantly larger than the other objects on the field the standard region of interest finder (3.2) does a poor job of finding candidates. To solve this

the region finder is run again at a low resolution with different grid and merging settings. Clusters of these regions are grouped in order to generate candidate robot regions, containing all regions forming the cluster.

The second part of the pipeline is the classifier. As with every component of vision the key factor here is balancing the constraints of accuracy and efficiency. Decision trees are extremely efficient, however, they struggle with the highly variable appearance of robots. A machine learning method, random forest, is used to bag the regions into the correct categories. In addition, careful feature engineering is essential to achieve good performance. A large variety of features were tested, and the following features were found to be the most informative:

- Width/height ratio
- White/black pixels ratio
- Size of the bounding box
- Mean of the pixels (binary image)
- Variance of the pixels (binary image)
- Scaled Mass Centre's X
- Scaled Mass Centre's Y
- Hu moments [4]
- Zernike moments [5]

The scaled mass centre is calculated by dividing the zero order moments by the first order moments. Hu and Zernike moments are values that relate to the shape of the pattern in the image.

Each tree in the random forest is given access to a subset of these features. This results in a variety of different decision trees, which leverage different aspects of the region's appearance. These trees are then composed together, with the vote of the "forest" giving a better classification quality than the individual trees.

4 Localisation

A Multi-Modal Kalman Filter is used for robot pose estimation. Information on odometry and field features are used for the prediction and measurement update steps (respectively) in the Kalman Filter.

Odometry is calculated from the movement of the robot's feet and the gyroscope reading, then applied to the predicted pose of the robot. The variance of the robot is increased in proportion to the magnitude of the pose change predicted by odometry.

In the measurement update step field features are used. A complication to this is the existence of visually identical instances of field features at several different locations (i.e. there are six T-intersections on the field). To handle this, we assume a sighted field feature is the instance of the feature most consistent with the current robot pose estimate. A field feature observation is a polar observation, but must be applied to a Cartesian Kalman Filter. This creates a non-linear relationship between the observation measurement and Kalman filter state. Two techniques are used to manage this:

1. Measurement to Pose conversion. For field features that provide distance, heading, and orientation, specifically corners, T-junctions and centre circles, a set of possible robot poses are calculated, corresponding to the multiple corners/ tjunctions/ centre-circles on the field. Measurements with relatively small discrepancy with the current pose hypothesis are passed through to the Kalman Filter for a linear update.
2. Extended Kalman Filter. For a penalty spot, that provides only distance and heading in our system, an EKF is used to linearise the polar measurements around the current estimate.

To note, the system has moved mostly away from non-linear filters, due to the removal of polar features such as goal posts and beacons.

In 2014, the 2006 distributed multi-modal Kalman Filter localisation developed for the AIBOs were ported to the Naos. We track multiple hypothesis modes for the pose of the robots and ball on the field. Each hypothesis mode consists of the robot pose, ball position and velocity, and the poses of the teammate robots. The robots share information regarding the ball's state and the robot's pose so that teammates can incorporate this information into their own filters.

To handle the symmetry of the field, the ball is used for disambiguation between the two symmetric sides of the field, through majority consensus on which side of the field the ball is on between team mate robots.

5 Motion

rUNSWift's motion is primarily based off the Hengst's walk generator [3] developed in house and used since the 2014 RoboCup competition. Today, it is still robust choice for many teams and we have continued to improve it through several modifications. The kick motion, integrated directly into the walk generator for smooth transitions, was also developed around the same time, and has also not varied much from its original design. For the 2018 competition significant improvements were made to both of these motions to improve the robots stability and recovery from disturbances. The motion of the kick was also modified to improve its dynamic reach, allowing the robot to powerfully strike balls in a larger area in front of the kicking foot.

In 2017 artificial grass fields were introduced, a surface that made balance recovery from disturbances more difficult. Additionally, the rough field surfaces at the 2017 RoboCup Nagoya competition farther complicated stable movement. These uneven surfaces introduce unexpected disturbances, causing instability. These factors, combined with an ageing and worn team of robots, influenced the decision to improve the stability of rUNSWift's walk generator. When a large destabilisation of the robot is detected by the gyro the robot will now react by moving its hip to compensate. As field construction proved similarly challenging in 2018 this development was tested in practice and proved itself a significant improvement to walk stability.

6 Behaviours

6.1 New Behaviour Structure

In 2018, we modified our behaviour architecture to improve dynamic role switching. From 2010, the higher level layers of rUNSWift's behaviours have been written in Python, allowing faster development. Our behaviours are modelled off a decision tree. Nodes are divided into two categories, roles and skills. Roles, such as "Goalie" and "Penalty Striker", define what the robot should be doing during a game. Skills are specific actions a robot has chosen to take.

Skills range from relatively high level actions, such as approaching the ball or walking to a global point on the field, to lower level actions, such as stand or crouch. This allows the low level skills to be inherited as components of several higher level skills, which in turn are used by one or more roles.

State transitions can happen a frequently when a robot is involved in ball play. Under the old system, when transitions take place in high level skills, all child skills must be reinitialised. This process creates a computational overhead that reduced robot responsiveness, particularly during critical periods of play around the ball. To mitigate this we redesigned our system so that all objects are initialised at the start, and a reset routine is called on only the nodes of interest when a transition takes place.

6.2 Rule Changes

A major new rule in 2018 was the introduction of free kicks. When on the offending side of a free kick the team needed to ensure they moved beyond 0.75m from the ball to prevent illegal defender penalties. Robots prioritise walking out of the 0.75m radius around the ball, whilst remaining as close as can be reliably achieved to enable faster repossession of the ball after the free kick. Where the robot has lost sight of the ball, the robot moves to a position close to (but still more than 0.75m from) the known goal kick ball placement locations. In a pushing free kick the robot starts a search for the ball if they do not know its location, moving away when they find it.

2018 also brought about changes to the packet size. We adapted to this requirement by removing some data duplicated in our message and the SPL standard message. Another change was the reduction in the packet rate, so the rate of transmission was reduced accordingly.

6.3 Mixed Team

This year, rUNSWift participated in the Mixed Team Tournament with team B-Human to form the team B-Swift, where we placed first. This is the second year we participated in the mixed team tournament.

Mixed team games are played with six robots on each side, with each team contributing three robots. The B-Swift strategy was to split offence and defence

between the two teams. rUNSWift's three robots were primarily performed defence defence, whilst offence was handled by the B-Human robots. Our robots were split into three roles; the goalie and defender used in non mixed team game play and a specially designed Defensive MidFielder (DMF).

When attacking, the defender will calculate the ball-to-goal vector and position itself to intercept while the DMF plays more aggressively, activating immediately when it may be needed. The DMF also has some limited responsibility to support the attackers. While in this year's tournament the DMF did not score directly, the role created some good opportunities for the strikers to shoot goals.

The core strategy for the DMF in defence was to contend with the opponent's striker while the defender keeps a constant distance to respond to failed contention. The defender was also able to offer vision support, such as passing on the location of the ball when it was kicked behind the DMF.

7 Tooling

While they have no direct effect on gameplay, tools are critical to developing effective solutions to the problems found in robot soccer. Here the two primary tools used by rUNSWift are outlined.

7.1 Offnao

Offnao is a tool built in house that collects, saves, restores and streams data from Nao robots. It can record all the data from the robot's sensors, including either the colour classified or raw camera data. If given the raw camera data Offnao can also run the vision system offline. Offnao also collects information about the internal state of the robot, such as its beliefs about its position, team mate position and ball position, along with what the vision system has detected.

All this information can be displayed through various visualisation methods, including annotated versions of the robot's camera image and an overhead field view (Figure 11. Camera settings may also be changed through Offnao. Camera pose offsets can be calibrated to compensate for differences in the head mounting and looseness in the robot's joints. Debug logs recorded by the main code may be viewed. Finally, raw data from the non vision sensors, such as sonar and foot sensors, can be accessed.

This tool is also used to calibrate the camera pose and camera settings, by streaming live data from a robot that is running the code. We can tune these parameters in real time, before saving them to a config file for the robot to use.

7.2 Vatnao

As part of our 2016-17 Vision System changes we developed a tool to assist in development and debugging. The tool, Vatnao, runs the vision system on a local machine using logs recorded on the robot. It allows the developer to examine variables, annotate images based on how Vision processes the raw frames, and

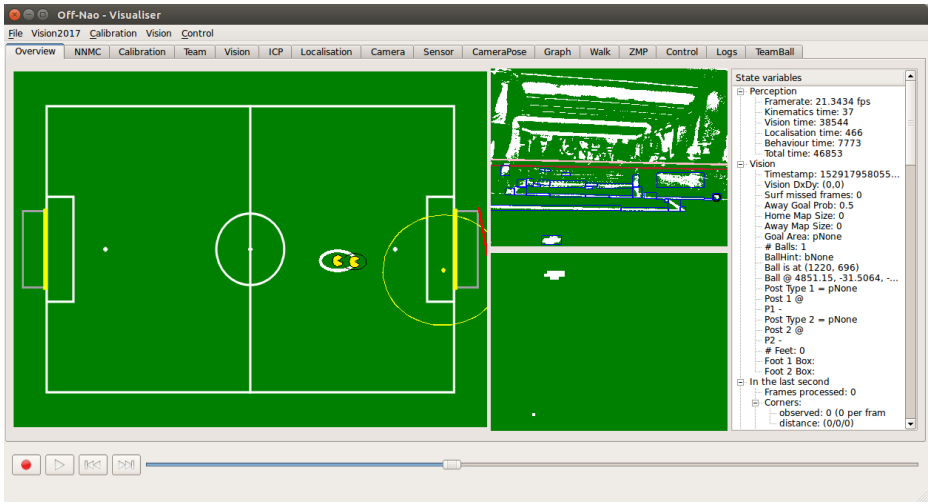

Fig. 11: The primary Offnao display. The yellow dot is the current ball position belief. The little pac-man figure is the robot's position belief. Blue boxes are detected regions. The black circle is a detected ball.

even adjust configuration in real time. We used it heavily as part of development of the ball detector to speed up the development cycle, reducing testing time, quickly identifying problem cases and allowing us to quickly evaluate the effects of tweaks and changes to our code base.

Vatnao has also allowed us to break down the components of each detector. For example, in the case of the ball detector, each heuristic can be broken down and displayed separately to determine optimal values for the checks. The tool also handles other modules of vision, such as field feature and robot detection. In 2018, we have continued to develop and use Vatnao to improve vision as we port more of the vision system to use regions of interest 3.2.

8 Concluding Discussion

rUNSWift placed in the quarter finals this year, winning 3 games out of 5. Although this is the same standing as 2017, we are proud of the notable improvement to gameplay through the number of goals scored. Additionally, we placed in the quarter finals in the Penalty Shootout Challenge, and first place in the mixed team tournament with our partners B-Human.

In 2018, our team continued to further improve vision, tweak kinematics and began restructuring behaviours. During the competition it was apparent our vision and kinematics work had paid off, however, there is considerable room for improvements in field features, localisation and behaviours. In several instances, our robots suffered from getting mislocalised on the field which had detrimental

effects on our gameplay. Our behaviours can be refined to leverage our team's strengths of long range vision and flexible motion.

For the upcoming year, we plan to focus on behaviours and localisation. We also welcome the update to the SPL robot, which brings about new challenges, opportunities and capabilities. With the updated hardware, we are excited by the new potential it delivers in moving the league forward.

Acknowledgements

The 2018 team wish to acknowledge the legacy left by previous rUNSWift teams and thank the School of Computer Science and Engineering, University of New South Wales for their continual financial, administrative and financial support to our team. We would like to thank the Nao Devils team for the use of their camera driver, with some modifications. We also wish to pay tribute to other SPL teams that inspired our innovations in the spirit of friendly competition.

References

1. Bai, G., Brady, S., Brameld, K., Chamela, A., Collette, J., Collis-Bird, S., Hall, B., Hendriks, K., Hengst, B., Jones, E., Pagnucco, M., Sammut, C., Schmidt, P., Smith, H., Wiley, T., Wondo, A., Wong, V.: runswift 2017 team report and code release. Tech. rep., The University of New South Wales (2017)
2. Bradley, D., Roth, G.: Adaptive thresholding using the integral image. Journal of graphics tools 12(2), 13–21 (2007)
3. Hengst, B.: rUNSWift Walk2014 report. : http://cgi.cse.unsw.edu.au/'tilde'robocup/2014championteampaperreports/20140930-bernhard.hengst-walk2014report.pdf, University of New South Wales (2014)
4. Hu, M.K.: Visual pattern recognition by moment invariants. IRE transactions on information theory 8(2), 179–187 (1962)
5. Khotanzad, A., Hong, Y.H.: Invariant image recognition by zernike moments. IEEE Transactions on pattern analysis and machine intelligence 12(5), 489–497 (1990)
6. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems. pp. 1097–1105 (2012)
7. Nielsen, F.: k-mle: A fast algorithm for learning statistical mixture models. In: Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on. pp. 869–872. IEEE (2012)
8. Rosenfeld, A., Pfaltz, J.L.: Sequential operations in digital picture processing. Journal of the ACM (JACM) 13(4), 471–494 (1966)