Cognitive Image Processing for Humanoid Soccer in Dynamic Environments[1]

Gabriëlle E.H. Ras

June 22, 2015

Abstract

In this paper we consider the problem of object detection. A simple cognitive image processing module (CIP-module) based on the *Recognition-By-Components* (RBC) is used to extract important features from an image without using the information about color of the features. The developed CIP-module shows that object detection can be achieved without having to rely on a specific color of the object. In this case, a Standard Platform League (SPL) goal and ball are successfully detected in realistic SPL game scenarios. The CIP-module is compared to the most widely used method of object detection within the SPL which is image segmentation based on the scanline method. A comparison is made on grounds of runtime and accuracy.

Keywords Robocup, SPL, object-detection, cognitive image processing

1 Introduction

1.1 The RoboCup and the Standard Platform League

RoboCup
The RoboCup is an international robotics competition to promote Artificial Intelligence and robotics research. It was founded in 1997 with the following objective: *"By the middle of the 21st century, a team of fully autonomous humanoid robot soccer players shall win a soccer game, complying with the official rules of FIFA, against the winner of the most recent World Cup."*[19] Much progress has been made since the humble beginnings of 1997. Currently, the RoboCup has a multitude of leagues within leagues, one of which is the Standard Platform League(SPL) within the RoboCup Soccer League.

[1]This thesis has been prepared in partial fulfilment of the requirements for the Degree of Bachelor of Science in Knowledge Engineering, Maastricht University, supervisor: Dr. Nico Roos.

Standard Platform League
The Standard Platform League (SPL) is a league within the RoboCup Soccer League. Each participating team uses the same type of humanoid robot, the NAO robot made by Aldebaran. In this league the goal is to play soccer with a team of robots and to win from the opposing team. In order to achieve this goal, the robots have to be equipped with a set of skills to be able to correctly interact with the environment, of which visual perception is the most important one. The robot has to be able to correctly perceive where the ball, the goals and the other robots are.

Nowadays, the SPL is an environment with reasonably stable conditions: A game of SPL takes place on a green soccer field with white lines. An orange ball is used to play with and the robots have to score in a white goalposts. Each team has no more than 5 robots and 1 coach robot. The robot that must be used is the gray NAO made by Aldebaran. So far, SPL matches have always taken place in an indoor setting with stable lighting conditions provided by large ceiling lights. The exact lighting conditions depend on the competition site. Also dependant on the competition site, fields can be located close to each other without a barrier between adjacent fields. In the case that there is no barrier present, there will be at least 3 meters between the carpets of the adjacent fields[18].

Technical Challenges
Each year new technical challenges are conceived to make the SPL matches more realistic to a human soccer match. Teams can score points by completing technical challenges successfully. Technical challenges often lead to modification of the SPL rules to make the matches resemble actual human soccer matches to a greater extent. In the future some of the match parameters will change as there are already many SPL challenges that deal with a more dynamic environments[15]. Because of this it is worth investigating techniques that make perception in dynamic environments more robust.

1.2 Problem Statement and Research Questions

The problem of object detection in the SPL domain has been approached in many ways, most of which include methods that specifically search for the color of the object. These methods search within a certain range of the to be detected color or using color look-up tables[12],[4]. While this has the advantage of being able to quickly and accurately determine the possible places where the object could be located, these methods have the disadvantage of needing color calibration, stable lighting conditions and the knowledge of what color to search for[12],[4],[3]. These requirements put strong limits on how dynamic the SPL environment can be, and ultimately, these requirements impair the ability of the robots to handle a realistic match scenario. If the goal is to play against humans in 2050, the robot should be able to detect objects under varying environment circumstances.

Since the objects that need to be recognized are distinct in shape, the shape can be considered a reliable source to base perception on. Instead of relying on color, an approach is chosen that relies on the shape of the object.

Recognition-By-Components

In this paper the problem of object detection is approached from a cognitive perspective that incorporates the use of shapes for human object recognition. The *Recognition-By-Components* (RBC) theory proposed by Biederman [2] is used as the structural model for the development of a object detection module. In short, the RBC theory is a bottom-up process to explain object recognition in humans. The theory states that there are basic *geons* (for "geometrical ions") that describe the shape of any object. According to the theory, geons form an alphabet for the compositions of complex objects, i.e. complex objects are composed of 2 or more geons. Biederman's original geons are generalized-cone components and it is theorized that there are $N \leq 36$ geons. Biederman hypothesizes that geons are simple, typically symmetrical volumes lacking sharp concavities, such as blocks, cylinders, spheres and wedges. Biederman's original geons are 3-dimensional volumes, however, in this implementation of the theory, geons are 2-dimensional shapes because the input image is 2-dimensional. In Figure 1 it can be seen how SPL objects can be decomposed into geons.

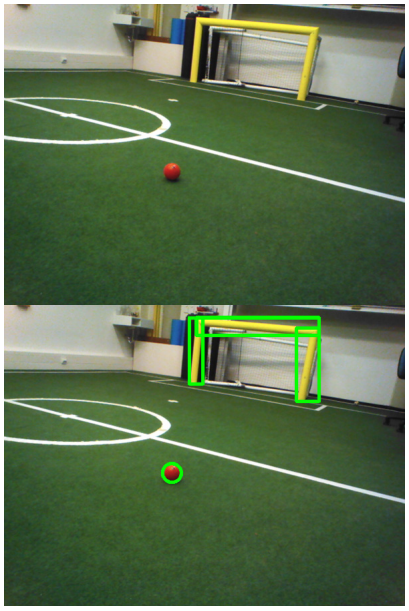

Figure 1: Decomposition of SPL objects into geons.

This thesis deals with the following problem statement:

- Is the RBC theory applicable to an SPL environment?

And it investigates the following research questions:

1. How does the color space representation of the image affect the saliency of the to be detected objects?
2. Is object detection in the SPL possible without searching for the color of the specified object?
3. Is the RBC-approach better than a color-approach in terms of accuracy and run-time?

2 Background

The software is developed and tested on an ASUS X53S laptop with 2.2 GHz CPU and 4 GB of RAM. In reality, the software would be run on a NAO robot with an ATOM Z530 1.6 GHz CPU and 1 GB RAM. As can be seen in Figure 2, the Nao has 2 cameras in it's head, one above the other. Both cameras cannot be used simultaneously and the views from both cameras make stereo vision impossible. The developed module uses both cameras depending on the scenario. Each time, an image with the resolution of 640x480 taken by the NAO robot is used as input. The language that the CIP-module was programmed in is C++11. The module was developed using methods from the OpenCV library version 2.4.9.

3 Related Work

In the past there have been several SPL Technical Challenges to motivate the research of SPL in more

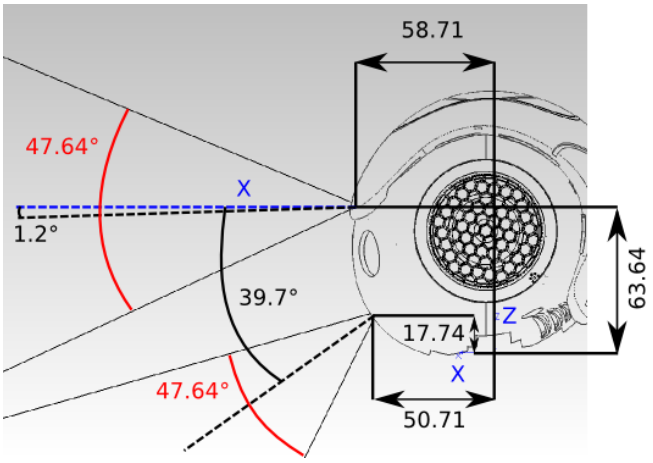

Figure 2: Location of the cameras and their angles.

dynamic environments such as the *Any Place Challenge* in 2014[15], *Any Ball Challenge* in 2009[14], *Variable Lighting Challenge* in 2005[16] and the *New Goal Challege*[17] in 2006.

To solve the *Any Ball Challenge*, García *et al.* [8] use a visual attention model based on a saliency map. Their findings show that their approach is successful with 100% success rate for the images that were used in their experiments. However, a high computation time is needed for the algorithm to work and this makes it unable to operate at real time.

For the *Variable Lighting Challenge*, the Dutch AIBO Team did a study on color invariance algorithms. In this study a comparison was made between algorithms based on a Gaussian color model (Geuzenbroek, 2000) or a color-histogram (Zivkovic & Kröse, 2004). The conclusion of this study was that the algorithm based on color-histograms is applicable on the Aibo platform. The computational resources on the Aibo ERS-7 are clearly not enough to facilitate an algorithm based on a Gaussian color model [13]. We bear in mind that this was performed on the Aibo ERS-7 model, which came out in 2003. Nowadays the Nao robots are much more sophisticated. The usage of color-histograms indicate that the team was working with bounded values of color, which are not robust if the lighting would change dramatically.

Trifan *et al.* [20] used a real-time vision module detect SPL objects. Their module uses a look-up table for fast color classification, an intensity histogram for self-calibration and color segmentation as the basis for object detection. However, the module is completely based on the SPL object remaining the same color and would not be suitable in future SPL settings where the color of the objects will change.

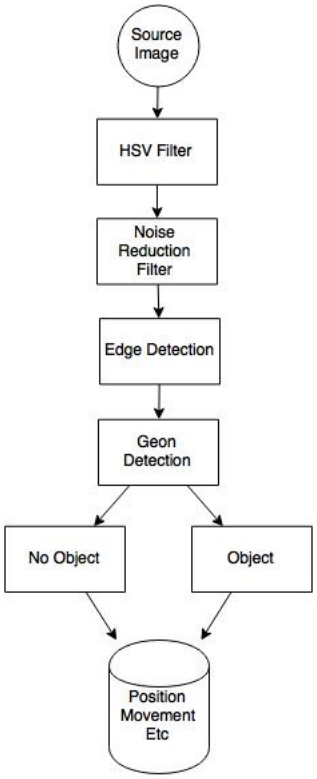

Figure 3: Data flow in the CIP-module

García *et al.* [9] believe that large parts of the image are not usable and use a selective attention model to only analyse a part of the image. The model utilizes color segmentation and shape analysis to detect the ball.

In general, most teams use techniques that are based on (a combination of): Color segmentation with the use of scanlines, edge detection, saliency-scan[1] and selective visual attention to determine the locations of the objects on the field.

4 The CIP-module

The CIP-module is a partial implementation of Biedermans Recognition-By-Components theory. It has no knowledge of the color of the object that needs to be detected, it relies heavily on edge-detection techniques and it uses geons to detect the object based on their shape. The saturation channel of HSV is used to enhance the saliency of the objects in an attempt to increase the accuracy of edge detection. The entire process can be described by the following 4 phases. Figure 11 describes the data flow in the CIP-module.

4.1 Enhancing the Saliency of Objects

Saliency

Saliency is defined as the distinct subjective perceptual quality which makes some items in the world stand out from their neighbors and immediately grab our attention[11]. There are many factors that can make an object salient, such as color, orientation and movement. In the SPL case, the objects are salient due to their loud colors and due to the fact that there are not many objects on a SPL field.

It is believed that the more salient an object is, the more easily it is detected by an edge detector. Edge detectors are sensitive to significant changes in the local contrast of neighbouring pixels[5]. Changing representation of the image can enhance or reduce this local contrast. Saliency detectors also take local contrast of neighbouring pixels into account since they take into account the color and intensity information of the pixels in an image[11]. Thus, a representation that enhances the saliency should also increase the local contrast. In Figure 4 a saliency map has been obtained from the saturation channel of the HSV representation of the original image.

Color Spaces

When the NAO robot captures an image, it is returned as an image represented by the RGB color space. Rather than keeping this automatically selected color space and using all 3 channels, experiments were done to determine if a different color space and a different channel would have an effect on the saliency of the objects. This was achieved by by experimenting with the HSV and HLS color spaces and their channels. It is expected that the saliency will be affected, because when the representation of an image changes, the information that determines the saliency of an object also changes.

HSV and HLS were chosen because they represent color in a way that not only lets us look at the color property of a pixel, but also many other properties of a pixel, such as saturation, brightness, lightness, hue and value. This makes it possible to analyze the value of a pixel regardless of the perceived color of the pixel. Other color spaces do not encode their representation of color in such a useful way. RGB and CMYK only represent color in such a way that makes it easy to reproduce a color by using a additive and subtractive color model, respectively. CIEXYZ, CIELAB and LMS represent color in a way that is comparable to how human beings perceive color by mimicking the function of the rods and cones in the retina of the eye. Luma plus chroma color spaces such as YCrCb are used in

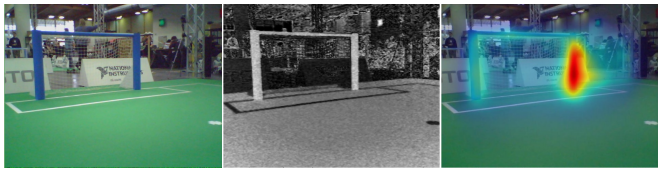

Figure 4: Original Image, HSV - S channel, Overlay of Saliency Map

video and digital photography systems, where the luma represents the brightness of a pixel and the chroma values correspond approximately to the amounts of blue and red in the color.

Experiments and Results

A saliency map is obtained from all channels of the HSV and HSL color spaces. The saliency maps are obtained using a graph-based visual saliency (GBVS) implementation in MATLAB[10]. GBVS attempts to predict the location of the salient regions in an image as a human observer would perceive it. The original image and the images of all the channels are compared to each other. Then the results are analyzed to see if changing color spaces and channels has an effect on the output of GBVS.

A set of 20 soccer and SPL images was used to produce the saliency maps. For the calculation of the measure score that is given in Table 1, each of the 20 images is analyzed a total of 7 times: 1 time for the original image, 3 times for each channel of HSV and 3 times for each channel of HSL. Each time the saliency map is given a score depending on where it thinks the salient regions are. If the detected salient regions are foreground objects it is considered correct. If the the detected salient regions are background objects it is considered incorrect. If the detected salient regions are background and foreground objects then it is considered neutral. Let S be the detected saliency region and f' the score per image.

$$f = \begin{cases} 1 & \text{if S = correct} \\ 0 & \text{if S = neutral} \\ -1 & \text{if S = incorrect} \end{cases} \quad (1)$$

The measurement score M of an image is calculated by taking the average of all the scores. n is the number of images.

$$M(image) = \frac{1}{n}\sum_{i=1}^{n} f_i \quad (2)$$

The results given in Table 1 indicate that the saturation channel in the HSV color space enhances the saliency of the objects in the images by a difference of $M(HSV_S) - M(original) = 0.1$.

Measure score		
Original image	RGB	0.35
HSV	H	-0.1
	S	0.45
	V	0.25
HLS	H	-0.1
	L	0.2
	S	-0.1

Table 1: Color Spaces Saliency Experiment Results

4.2 Noise Reduction

Digital cameras are prone to Gaussian sensor noise in circumstances where the image is underexposed. Because the CIP-module relies heavily on edges, it is not expected to be able to handle noisy images very well. In an image with much noise, many edges of irrelevant features will be detected. To reduce the impact that noise can have in an image, the image is convoluted with a 2-dimensional Gaussian kernel:

$$G_0(x,y) = Ae^{\frac{-(x-\mu_x)^2}{2\sigma_x^2}+\frac{-(y-\mu_y)^2}{2\sigma_y^2}} \quad (3)$$

where G_0 is the Gaussian distribution, A is the amplitude, μ is the mean and σ the standard deviation, one for each dimension.

Gaussian Parameters

Using the OpenCV function *cv::GaussianBlur*, we can control the kernel size K_G and the standard deviations σ_x, σ_y. Through empirical results it was found that $K = 1$ and $\sigma_x = \sigma_y = 0$ sufficiently reduces the amount of noise while preserving the details for the images with a resolution of 640x480. The bigger K is and the smaller σ, the better the approximation of the kernel to the continuous distribution. This is favourable when having the luxury of processing capacity, but not in the situation where there is little processing power to spare.

4.3 Feature Extraction

In this phase, the goal is to find the meaningful features in the image. The previous phases eliminated almost all information about color in the image and reduced the noise. That has left us only with color intensity information. This makes it easy to discern the boundaries between high intensity objects and the background of the image. To extract the boundaries from the image, edge detection is performed using the Canny Edge detector[5]. This significantly reduces the amount of information that has to be processed in the future phases.

The Canny Edge detector was developed by J. Canny and is generally considered to be an optimal detector because it aims to satisfy 3 main criteria:

1. Low error rate
2. Good localization
3. Minimal response

It does this by finding the intensity gradient of the image, using non-maximum suppression to determine if the pixel is a better candidate for an edge than its neighbours. And then by applying hysteresis as the final step in order to decide where an edge begins and ends[5].

Canny Edge Parameters

In the implementation we used the function *cv::Canny* which gave us the ability to specify the parameters of the aforementioned relevant steps. The function uses an extended Sobel operator to find the intensity gradient of the image. Here a kernel size $K_S = 3$ is found to deliver better results. To increase the accuracy of *cv::Canny*, the L_2 norm is used to calculate the image gradient magnitude:

$$L_2 = \sqrt{(\delta I/\delta x)^2 + (\delta I/\delta y)^2} \quad (4)$$

Large intensity gradients are more likely to correspond with to edges compared to small intensity gradients. In most cases it is not possible to indicate whether a threshold corresponds to a gradient being an edge or not, which is why the Canny Edge detector uses thresholds with hysteresis[6]. Hysteresis is the dependence of the output of a system on its current input and its past inputs. In the context of edge detection this means that to classify a pixel as an edge, it depends if their neighbour pixels were classified as an edge. First one threshold is used to determine is a pixel is an edge. If the intensity value lies underneath the first threshold and a neighbour pixel was classified as an edge, then the second threshold is used to see if the intensity is still enough to make it an edge. This requires 2 thresholds: τ_1 and τ_2. It was found through experimentation that $\tau_1 = 60$ and $\tau_2 = 180$ are good values.

4.4 Geon Matching

Geons

This approach is inspired by the RBC theory that utilizes geons which, when combined, can generate an infinite set of complex objects. In the SPL setting we focused on the ball and goal. Hence the geons that are described are strictly the ones used to generate these objects and can be seen in Figure 6. Geons are not fixed shapes, but rather they are dynamic descriptions of quadrilateral shapes (quad) and *blob* shapes. A blob is defined as a round shape that may be dented or have protrusions. A *quad* is defined as a shape with 4 edges.

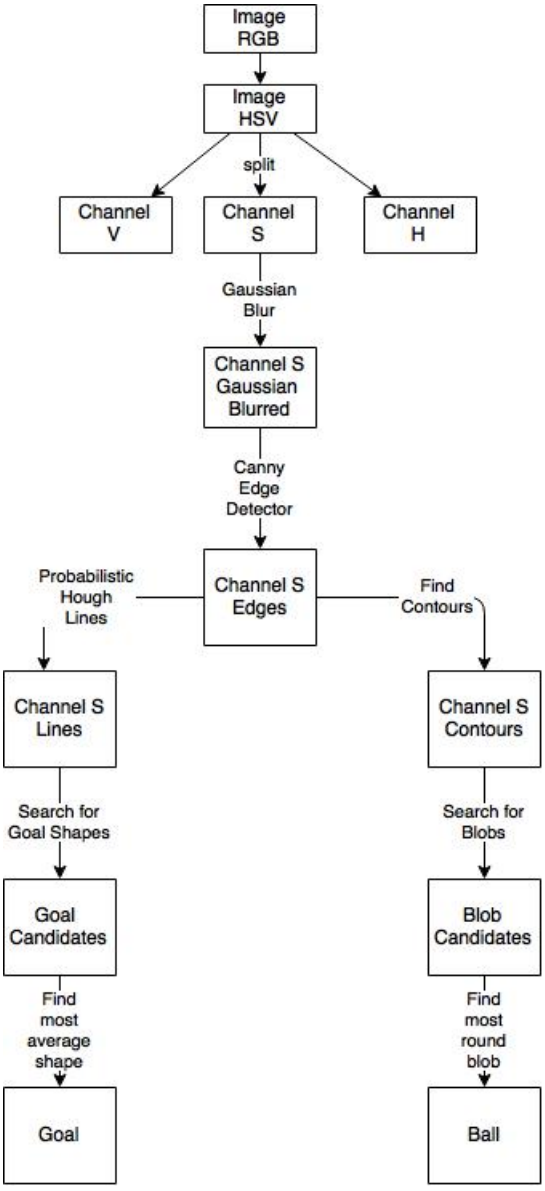

Figure 5: Finding the correct shapes in the CIP-module

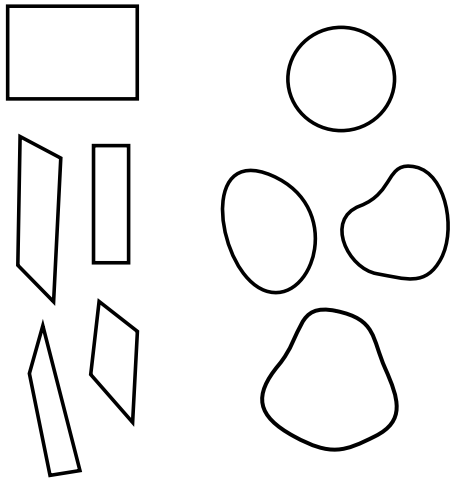

Figure 6: Quadrilateral and blob geons

Figure 7: 3 different goal angles

Goal

To find a goal, lines that form the shape of a goal need to be located. The shape of the goal can differ depending on the angle that the NAO robot is facing the goal. This can be seen in Figure 7. But in general, independent of the facing angle, only 3 lines need to be found that make up the goal. This is done by applying a Hough transform on the set of edges to find straight lines. The function *cv::HoughLinesP* is used to detect the lines. After that, the set of returned lines is analyzed to find 3-tuples of lines that intersect each other in the way that the top goal-bar intersects with the 2 goalposts. Lastly, the tuples are analyzed for the width-height ratio $r = width/height$. Since it is not known from which angle the NAO robot is looking at the goal, and thus, which shape the goal will be, the mean ratio r of all potential candidates is computed and the candidate with the ratio that is closest to this mean will be returned.

Ball

To find the ball, a more complicated procedure is used. Because the ball is such a small object in a larger image, it is highly susceptible to noise caused by shadows and light reflection in such a way that the shape rarely resembles a circle. This means that techniques such as the Hough circle transform will not work well on this problem. See the Appendix for a comparison between *cv::HoughCircles* and the ball-finding function of the CIP-module.

To find the ball, the contours are computed from the previously found set of edges using *cv::findContours*. This function returns each contour as a set of edges. To clarify, an edge is an image point where the intensity difference between pixels is significant. A contour denotes an object boundary.

A contour consists of n points. For each contour that is found, each point is given an error value, denoted by $e(c_i)$. The error value is computed in the following way:

$$e(c_i) = |D_P - BB_{rad}| \tag{5}$$

where

$$D_P = \sqrt{(BB_{cp,x} - P_x)^2 + (BB_{cp,y} - P_y)^2} \tag{6}$$

and

$$BB_{rad} = \frac{BB_{width} + BB_{height}}{4} \tag{7}$$

For each contour, first calculate it's bounding box BB and center point of the bounding box BB_{cp}. Next, calculate what is called the radius of the bounding box BB_{rad}. Then, for each point P in the contour, calculate the euclidean distance D_P between P and BB_{cp}. Next, calculate the difference $\triangle D_P$ between D_P and BB_{rad}. Do this for all points in the contour and divide by the amount of points in the contour to get the average error of a contour:

$$e(c) = \frac{\sum_{i=1}^{n} e(c_i)}{n} \tag{8}$$

where This error indicates how similar the shape is to a circle. The closer to 0, the more circular the blob is. The blob with the lowest error is returned.

5 Color-Based Image Segmentation

Color-based image segmentation (CBIS) is the process of partitioning a digital image into multiple segments by using the colors of the image. In Figure 8 a schematic of the implemented CBIS algorithm can be seen. First the image is scanned with horizontal and vertical scanlines. These search the image for pixels that are similar to the colors in the lookup-table (LUT). The scanlines scan the image for the pixels that are red, yellow and white. These are indicated in the LUT with RGB-values. After the scanlines find the desired pixels, these are grouped into clusters using the DBSCAN algorithm[7]. DBSCAN is a density-based clustering algorithm that groups together points that are close to each other. The clusters are compared in size and the cluster with the biggest size is returned.

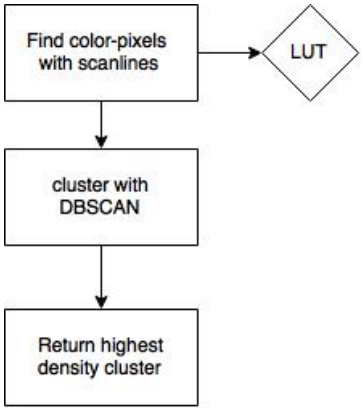

Figure 8: A schematic of the CBIS algorithm

6 Experiments

First a control experiment was done between CIP and CBIS. This was on a dataset of 60 images containing images taken by the robot in SPL environments and non-SPL environments.

The CIP method is compared with CBIS in 5 experiments by altering the following image parameters— of the images in the control dataset— with *Adobe Photoshop CS5*:

- Exposure
- Contrast
- Uniform noise
- Gaussian blur
- Saturation

Note that the individual geon functions are compared due to the different algorithms used to locate them. That is also the reason why each comparison with CBIS mentions the color used; each time CBIS was searching for a different color. Here red is used to find the ball and yellow and blue to find the goals.

A separate experiment was done on the CIP module to determine if changing the hue of the image would have any impact. Changing the hue is analogues to changing the color of the objects and the environment. The hue of the images in the control dataset was changed to a random hue-value each time.

As previously mentioned in section 4.4, *cv::HoughCircles* is compared with the blob-finding function of CIP. A set of 48 images coming from the control dataset was used to compare the 2 algorithms.

The experiments are set up as follows:

- Experiment 1: Control

Unaltered images in the dataset are analyzed by both algorithms.

- Experiment 2: Exposure
  A setting with an Exposure Value(EV) of -2.15 and a setting with an EV of 1.8.
- Experiment 3: Contrast
  A setting with a contrast of 100 and a setting with a contrast of -50.
- Experiment 4: Hue
  The hue of the image is changed to a random number.
- Experiment 5: Noise
  A setting with a noise level of 15% and a setting with a noise level of 8%.
- Experiment 6: Gaussian blur
  A setting with a gaussian blur with a radius of 4 pixels.
- Experiment 7: Saturation
  A setting with a saturation of 65 and a setting with a saturation of -48.
- Experiment 8: CIP-blob vs. hough circle transform
  Hough circle transform with the following parameters:
  Inverse ratio of the accumulator resolution to the image resolution = 1. Minimal distance between detected centers = 60. Upper threshold for the internal Canny edge detector = 60. Threshold for center detection = 50. Minimum and maximum radius of the to be detected circles = 0, in this case it means that they are unknown.

Results

- Experiment 1: CBIS outperforms CIP with an accuracy of 0.7 versus 0.42 respectively when it has to search for the ball. CBIS also outperforms CIP with an accuracy of 0.5 versus 0.3 respectively when it has to search for the goal. In terms or runtime for finding the ball, on average, CIP-ball scores 18.2 ms and CBIS scores 19.4 ms. For the goal, CIP-goal scores 503.9 ms and CBIS scores 88.6 ms.
- Experiment 2: Exposure
  CBIS performs bad both with goal and ball search, with an accuracy of 0.21 and 0.2 respectively when EV=1.8. When EV=-2.15 CBIS scores 0.72 when finding the goal. The accuracy of CIP is not significantly influenced.
- Experiment 3: Contrast
  Both algorithms seem not to be influenced by the change in contrast on both tasks.
- Experiment 4: Hue
  The hue does not have any effect on CIP's performances.

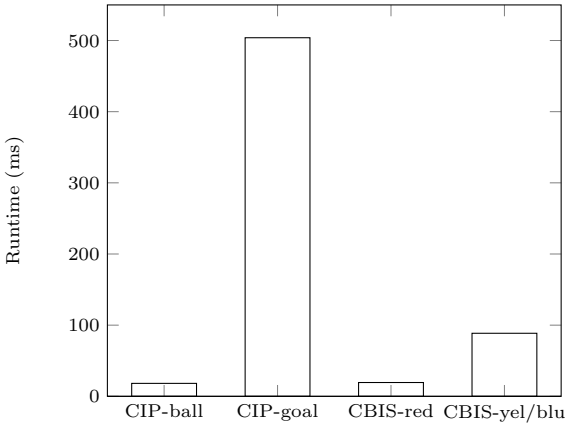

Figure 9: Comparison of the runtime of CIP and CBIS on the control dataset.

- Experiment 5: Noise
  CIP-goal was unable to finish any experiments with noise—the program kills the task because it took so long. CIP-ball is affected, having an accuracy of 0.02. CBIS seems affected when the noise level is 15%, having an accuracy of 0.2 for ball finding. The goal finding is not affected, accuracy of 0.47.
- Experiment 6: Gaussian blur
  CIP is affected, having a accuracy of 0.22 and 0.2 for ball and goal finding respectively. CBIS seems unaffected.
- Experiment 7: Saturation
  CIP-ball is affected by a saturation of 65, accuracy scoring 0.18. CBIS is affected by both saturation levels, as it performs worse accuracy-wise
- Experiment 8: CIP-blob vs. Hough circle transform
  Hough circle transform is only able to locate 3 out of 48 balls correctly. CIP-ball locates 16 out of 48 balls.

7 Conclusion and Future Work

Conclusion

Experiments have demonstrated that RBC can be applied on the SPL problem, in particular when no information is given about the color of the objects. As experiment 8 shows, the color of the objects and the background are unimportant and do not affect detection rates or runtime. Unfortunately, the accuracy is not satisfactory at 0.42 and a standard color-based image segmentation algorithm still outperforms it when the color of the objects is known beforehand. Additionally, RBC should not be used in cases where the image is affected by uniform noise or gaussian blur as the performances suffer greatly. This is because the algorithm is greatly dependant on edges. If an image is noisy, many false edges

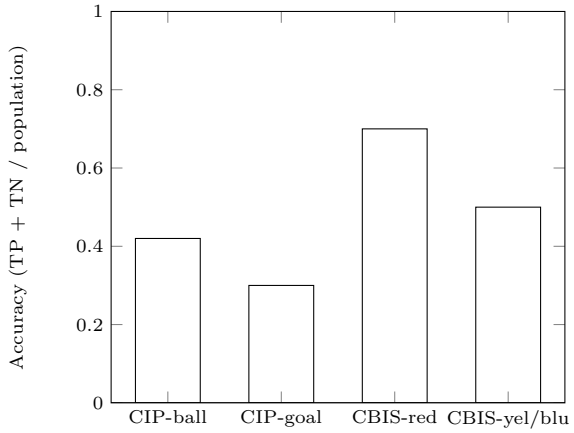

Figure 10: Comparison of the accuracy of CIP and CBIS on the control dataset.

will be found. However, in an environment where the colors of the objects are unknown, or if the background is unknown, then CIP will outperform a standard CBIS.

Future Work

CIP shows good potential as a simple yet effective algorithm for identifying SPL objects when their colors are unknown. Further research should include:

- An alternative to edges: most problems occur because the edge cannot be found.
- Identification of saliency regions: if the algorithm knows where the interesting objects are, the chances that it finds the correct edges are increased.
- The representation of more geons and how to combine them: With more complex shapes need to be detected, it is better to have dedicated geons.

References

[1] Ashar, J., Claridge, D., Hall, B., Hengst, B., Nguyen, H., Ratter, M. Pagnucco A., Robinson, S., Sammut, C., Vance, B., et al. (2010). Robocup standard platform league-rUNSWift 2010. *Australasian Conference on Robotics and Automation*.

[2] Biederman, I. (1987). Recognition-by-components: a theory of human image understanding. *Psychological review*, Vol. 94, No. 2, p. 115.

[3] Budden, D., Fenn, S., Walker, J., and Mendes, A. (2012). A novel approach to ball detection for humanoid robot soccer. *AI 2012: Advances in Artificial Intelligence*, pp. 827–838. Springer.

[4] Cañas, J., Puig, D., Perdices, E., and González, T. (2009). Visual goal detection for the robocup standard platform league. *X Workshop on Physical Agents, WAF*, pp. 121–128.

[5] Canny, J. (1986). A computational approach to edge detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, , No. 6, pp. 679–698.

[6] Canty, Morton J (2014). *Image Analysis, Classification and Change Detection in Remote Sensing: With Algorithms for ENVI/IDL and Python*. Crc Press.

[7] Ester, M, Kriegel, HP, Sander, J, and Xu, X (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. *Kdd*, Vol. 96, pp. 226–231.

[8] García, J., Rodríguez, F., Martín, F., and Matellán, V. (2010). Using visual attention in a nao humanoid to face the robocup any-ball challenge. *5th Workshop on Humanoids Soccer Robots. Nashville, TN, USA*.

[9] García, Juan F, Rodríguez, Francisco J, Fernández Llamas, Camino, Matellán Olivera, Vicente, et al. (2012). Designing a minimal reactive goalie for the robocup spl.

[10] Harel, Jonathan, Koch, Christof, and Perona, Pietro (2006). Graph-based visual saliency. *Advances in neural information processing systems*, pp. 545–552.

[11] Itti, L. (2007). Visual salience. Vol. 2, No. 9, p. 3327. revision 72776.

[12] Khandelwal, P., Hausknecht, M., Lee, J., Tian, A., and Stone, P. (2010). Vision calibration and processing on a humanoid soccer robot. *The Fifth Workshop on Humanoid Soccer Robots*.

[13] Oomes, S., Jonker, P., Poel, M., Visser, A., M. Wiering, W. Caarls, Leijnen, S., van Weers, S., Wijngaards, N., and Dignum, F. (2005). The dutch aibo team report on robocup 2004. *The Dutch RoboCup team, Holland*.

[14] RoboCup SPL Technical Committee, Technical Challenges for the RoboCup 2009 Standard Platform League Competition. www.informatik.uni-bremen.de/spl/pub/Website/Downloads/Challenges2009.pdf.

[15] RoboCup Technical Committee, RoboCup Standard Platform League (Nao) Technical Challenges. https://www.informatik.uni-bremen.de/spl/pub/Website/Downloads/Challenges2014.pdf.

[16] RoboCup Technical Committee, Technical Challenges for the RoboCup 2005 Legged League

Competition. www.informatik.uni-bremen.de/spl/pub/Website/Downloads/Challenges2005.pdf.

[17] RoboCup Technical Committee, Technical Challenges for the RoboCup 2006 Legged League Competition. www.informatik.uni-bremen.de/spl/pub/Website/Downloads/Challenges2006.pdf.

[18] RoboCup Technical Committee, (2015). Standard Platform League (NAO) Rule Book. www.informatik.uni-bremen.de/spl/pub/Website/Downloads/Rules2015.pdf.

[19] The Robocup Federation, About RoboCup. www.robocup.org/about-robocup/.

[20] Trifan, A., Neves Antonio, JR., and Nuno, L. (2012). A modular real-time vision module for humanoid robots. *Conference on Intelligence and Computer Vision XXIX*.

size: 640 x 480	Runtime (ms)				True Positive				True Negative				False Positive				False Negative			
Experiment 1: Control	CIP-ball	CIP-goal	CBIS-red	CBIS-yel+blu	CIP-ball	CIP-goal	CBIS-red	CBIS-yel/blu	CIP-ball	CIP-goal	CBIS-red	CBIS-yel/blu	CIP-ball	CIP-goal	CBIS-red	CBIS-yel/blu	CIP-ball	CIP-goal	CBIS-red	CBIS-yel/blu
Original	18.20116	503.8513233	19.39175	88.61319333	16	9	31	24	9	9	11	6	11	34	7	17	24	8	11	13
standard dev, n=60	2.047735251	980.2177365	5.423099582	302.8323132																
Experiment 2: Exposure																				
EV = -2.15	17.19243833	533.0195917	17.14447167	118.6801817	10	11	24	40	7	7	7	3	18	35	15	12	25	7	14	5
	2.049790671	1117.572549	4.654658156	255.0711508																
EV = 1.8	16.323785	121.0552617	17.592975	23.37744333	10	6	0	0	10	7	13	12	8	14	0	0	32	33	47	48
	1.964915348	222.0361934	2.090574556	2.822894927																
Experiment 2: Contrast																				
Level = 100	17.93319833	698.7253817	36.939435	518.89723	15	11	29	24	6	6	7	5	13	39	24	22	26	4	0	9
	1.986880707	1340.344685	104.5246819	1710.77057																
Level = -50	17.12594833	418.3901617	18.05787833	27.96099667	14	7	26	18	7	6	12	9	11	29	3	15	28	18	19	18
	1.929381556	832.4400115	4.412300223	14.4688565																
Experiment 3: Different image hue																				
Random Hue	18.33231167	504.2063983	n/a	n/a	16	9	n/a	n/a	9	9	n/a	n/a	11	34	n/a	n/a	24	8	n/a	n/a
	2.144829557	989.2567448																		
Experiment 4: Uniform Noise																				
Level = 15%	47.12167833	n/a	43.040845	85.074465	1	n/a	11	19	0	n/a	1	9	59	n/a	49	32	0	n/a	0	0
	3.0640105	n/a	70.9639516	123.0616761		n/a				n/a				n/a				n/a		
Level = 8%	33.89250667	n/a	22.33768	74.125105	0	n/a	22	26	0	n/a	7	12	60	n/a	28	19	0	n/a	3	3
	3.127139983	n/a	15.57222815	141.1682072		n/a				n/a				n/a				n/a		
Experiment 5: Gaussian Blur																				
Radius = 4 pixels	14.61735167	28.54795167	16.33837667	98.49788667	0	0	27	20	13	12	10	8	0	0	5	8	47	48	18	24
	1.564546281	3.710236233	4.666299328	419.0193551																
Experiment 6: Saturation																				
Level = 65	18.88388333	894.7302267	130.377395	1617.616467	5	15	12	16	6	5	4	8	25	38	42	30	24	2	2	6
	1.999136222	1894.803309	593.8514121	4044.523558																
Level = -48	17.77105167	532.069285	16.87461	23.33486333	11	8	17	6	9	6	12	12	17	35	2	7	23	11	29	35
	2.110490438	998.4557184	2.940075395	2.000629787																

47 images contained a ball
48 images contained (part of) a goal

	Runtime (ms)		True Positive		True Negative		False Positive		False Negative	
Experiment 7: CIP vs HouchCircle	CIP-ball	HoughCircle	CIP-ball TP	HoughCircle	CIP-ball	HoughCircle	CIP-ball	HoughCircle	CIP-ball	HoughCircle
original	19.14681875	1861.880746	16	3	0	0	8	41	24	4
st.dev. n=48	2.015092662	3341.417928								

48 images contain a ball

Figure 11: Appendix: results of all the experiments