ARTIFICIAL INTELLIGENCE FOR GAMES

Advanced Soccer Skills and Team Play of RoboCup 2017 TeenSize Winner NimbRo

Seminar report by

YEBEI HU

Matrikel-Nr. 3565290

Submitted to

Prof. Dr. Köthe

September 8, 2019

SS 2019 Heidelberg University

Contents

1 Introduction

"Robot Soccer World Cup" (RoboCup) is an annual international robotics competition proposed and founded in 1996. This competition aims at promoting robotics and AI research by offering a publicly appealing but formidable challenge. The official goal of RoboCup is that a team of fully autonomous humanoid robots shall beat the winner of the most recent World Cup by 2050. Every year the RoboCup Humanoid League raises the bar of its competitions. In 2017, the league increased the allowed number of players in the TeenSize soccer tournament to 3 vs. 3, which encourages the development of more complex team play strategies. At the same time, a new competition called Drop-in games was introduced in which a team is composed of robots from different institutes or universities. Improvements in individual skills were also demanded through a set of technical challenges including push recovery, high jump, high kick, goal kick from moving ball. The team NimbRo from University of Bonn was the winner of the 2017 TeenSize soccer tournament, the technical challenges, and the Drop-in games. This report presents individual skills and team play developments used in RoboCup 2017 which lead the team NimbRo to be the winner.

2 Robot Platforms

In RoboCup 2017 the team NimbRo used their fully open-source 3D printed platform, the igus Humanoid Open Platform [1]. Moreover, they upgraded one of their classic robots, Dynaped, so that it is able to get up from the prone and supine lying positions in order to satisfy the rules.

Fig. 1: The Team NimbRo and two kinds of platforms.

2.1 Igus Humanoid Open Platform

The igus Humanoid Open Platform (Fig. 1 left) has been developed as an open-source hardware and software project over the last four years. Because of its 3D printed exoskeleton, the weight of this 92 cm tall robot is only 6.6 kg. This platform incorporates an Intel Core i7-5500U CPU running a 64-bit Ubuntu OS, and a Robotis CM730 microcontroller board, which electrically interfaces with its Robotis Dynamixel MX actuators. The CM730 also incorporates 3-axis

accelerometer and gyroscope sensors, for a total of 6 axes of inertial measurement. For visual perception, the robot is equipped with a Logitech C905 USB camera fitted with a wide-angle lens. The robot software is based on the ROS middleware, which is a continuous evolution of the ROS software that was written for the NimbRo-OP [2].

2.2 Dynaped

Dynaped (Fig. 1 right) has been playing for team NimbRo since RoboCup 2009. Its original design had 14 degrees of freedom (DOF): 5 DOF per leg, 1 DOF per arm, and 2 DOF in the neck. Dynaped is distinguished by its effective use of parallel kinematics coupled with high torque, provided by pairs of EX-106 actuators in the roll joints of the hip and ankle, and pitch joints in the knee. All other DOFs are driven by single actuators. The torso is constructed entirely of aluminum and consists of a cylindrical tube and a rectangular cage that holds the electronics. Similar to the igus Humanoid Open Platform, Dynaped is equipped with an Intel Core i7-5500U CPU, a CM740 controller board (newer version of CM730), and a USB camera with a wide-angle lens. Dynaped used the same software components as the igus Humanoid Open Platform such as perception, bipedal walking, team communication, soccer behaviors, among others, due to their modularity and robustness.

Dynaped needed to be upgraded to satisify the new rule of RoboCup 2017. Dynaped was not able to get up either from prone or supine lying position with only 1 DOF per arm. So NimbRo included an additional DOF in each arm, namely a pitch elbow. Both the arm and the forearm are made from carbon fiber and reinforced against torsion with aluminum bars.

3 Visual Perception

3.1 Field Detection

Field detection of the robots can be based on color segmentation in the HSV color space, because the soccer field used in RoboCup competitions is green. Then a binarized image is constructed using a user-selected green color range, and all connected components that appear below the estimated horizon are found. Connected regions that have an area greater than some chosen threshold are taken into consideration for the extraction of a field boundary. If too many separate green regions exist, a set maximum number of regions are considered, prioritized by size and distance from the bottom of the image. Finding a convex hull of all green areas directly in the image is a very common approach. However, more care needs to be taken in this case because of the significant image distortion. The vertices of the connected regions are first undistorted before calculating the convex hull, so that these unwanted areas can be excluded. The convex hull points and intermediate points on each edge are then distorted back into the raw captured image, and the resulting polygon is taken as the field boundary.

3.2 Ball Detection

In previous years, the ball used in RoboCup was orange. So, most RoboCup teams used simple color segmentation and blob detection based approaches to realize ball detection. Now that the ball is mostly white, and generally with a pattern, such simple approaches no longer work so effectively, especially since the lines and goal posts are also white. A new method including two stages is used to solve this problem. In the first stage, ball candidates are generated based on color segmentation, color histograms, shape, and size. White connected components in the

image are found, and the Ramer-Douglas-Peucker algorithm [3] is applied to reduce the number of polygon vertices in the resulting regions. This is to make the subsequent detection of circle shapes quicker. The detected white regions are searched for at least one third full circle shapes within the expected radius ranges using a technique very similar to Algorithm 3. Color histograms of the detected circles are calculated for each of the three HSV channels and compared to expected ball color histograms using the Bhattacharyya distance [4]. Circles with a suitably similar color distribution to expected are considered to be ball candidates.

In the second stage, a dense histogram of oriented gradients (HOG) descriptor [5] is applied in the form of a cascade classifier, with use of the AdaBoost technique. Those candidates that do not have the required set of HOG features are rejected using this cascade classifier. The HOG descriptor is only applied to the regions suggested by the ball candidates to save computational time. The aim of using the HOG descriptor is to find a description of the ball that is largely invariant to changes in illumination and lighting conditions. However, the HOG descriptor is not rotation invariant, so to detect the ball from all angles, and to minimize the user's effort in collecting training examples, each positive image is rotated by $\pm 10^{\circ}$ and $\pm 20^{\circ}$, selectively mirrored, with the resulting images being presented as new positive samples. Greater rotations are not considered to allow the cascade classifier to learn the shadow under the ball. So, in summary, each positive sample that the user provides is extended to a total of 10 positive samples. With a set of about 400 positive and 700 negative samples that are gathered directly from the robot camera and with 20 weak classifiers, training takes approximately 10 hours.

The described approach can detect balls with very few false positives, even in environments cluttered with white and with varying lighting conditions. This approach can detect the FIFA size 4 ball up to four meters away. An interesting result is that our approach can find the ball in undistorted and distorted images with the same classifier. A full set of generalized Haar wavelet features [6] was tested on the same data set, and although the result was comparable, the training time was about 10 times slower than for the HOG descriptor. A Local Binary Patterns (LBP) feature classifier [7] was implemented in another comparative test, but the ball detection results were poor.

3.3 Field Line Detection

In previous years, many teams based their line detection approaches on the segmentation of the color white. But now the field lines are no longer clearly visible and identifiable due to the introduction of artificial grass in the RoboCup humanoid competition. Such approaches are no longer robust due to the increased number of white objects on the field, and due to the visual variability of the lines. The new robust approach is to detect spatial changes in brightness in the image using a canny edge detector [8] on the V channel of the HSV color space. The V channel encodes brightness information, and the result of the canny edge detector on this input is quite robust to changes in lighting conditions, so there is no need to tune parameters for changing lighting conditions. A probabilistic Hough line detector [9] is used to extract line segments of a certain minimum size from the detected edges. This helps to reject edges from white objects in the image that are not lines. The output line segments are filtered in the next stage to avoid false positive line detections where possible. A thresholding technique is used to try to ensure that the detected lines cover white pixels in the image, have green pixels on either side, and are close on both sides to edges returned by the edge detector. The last of these checks is motivated by the expectation that white lines, in an ideal scenario, will produce a pair of high responses in the edge detector, one on each border of the line. Ten equally spaced points are chosen on each line segment under review, and two normals to the line are constructed at each of these points, of approximate 5 cm length in each of the two directions. The pixels in the captured

image underneath these normals are checked for white color and green color, and the output of the canny edge detector is checked for high response. The number of instances where these three checks succeed are independently totaled, and if all three counts—V_W, V_G and V_E respectively—exceed the configured thresholds, the line segment is accepted, otherwise the line segment is rejected.

In the final stage, similar line segments are merged with each other, as appropriate, to produce fewer and bigger lines, as well as cover those line segments that might be partially occluded by another robot. Prior to merging, each line segment is projected to the egocentric world coordinates. The merging algorithm is detailed in Algorithm 1, which in turn uses Algorithm 2 to perform the individual merge operations. A list of the simple helper functions that are used in the algorithms in this paper is presented in Table I. Note that the constant thresholds used in Algorithm 1, are indicating the maximum allowed difference in angle and distance between two lines to be merged, and the numeric values are suitable for the current humanoid league field dimensions.

Algorithm 1 Merge similar line segments.

Input: A set of line segments S
Output: A set of line segments N where $|N| \leq |S|$

$N \leftarrow S$
repeat
  $M \leftarrow N$
  $m \leftarrow |M|$
  $N \leftarrow \emptyset$
  while $M \neq \emptyset$ **do**
    $X \leftarrow$ Any element of M
    $M \leftarrow M \setminus \{X\}$
    $\hat{m} \leftarrow |M|$
    for $Y \in M$ **do**
      if $MinDistance(X, Y) < 0.25\,\mathrm{m}$ **and** $AngleDiff(X, Y) < 15°$ **then**
        $N \leftarrow N \cup \{Merge(X, Y)\}$
        $M \leftarrow M \setminus \{Y\}$
        break
      end if
    end for
    if $|M| = \hat{m}$ **then**
      $N \leftarrow N \cup \{X\}$
    end if
  end while
until $|N| = m$
return N

Algorithm 2 $Merge(X, Y)$: Merge two line segments.

Input: Two line segments X and Y
Output: Merged line segment Q

$\theta \leftarrow 0$
$X_m = Midpoint(X)$
$Y_m = Midpoint(Y)$
$r \leftarrow \|X\|/(\|X\| + \|Y\|)$
$P \leftarrow rX_m + (1 - r)Y_m$
if $\|X\| \geq \|Y\|$ **then**
    $\theta \leftarrow Slope(X)$
else
    $\theta \leftarrow Slope(Y)$
end if
$Z \leftarrow$ The line through point P of slope θ
$S \leftarrow$ Orthogonal projections of the endpoints of X and Y onto Z
$Q \leftarrow$ The largest line segment defined by points in S
return Q

TABLE I: Helper functions used in the algorithms.

$MinDistance(X, Y)$	Returns the minimum distance between two line segments X and Y
$AngleDiff(X, Y)$	Returns the angle between two line segments in degrees
$Merge(X, Y)$	See Algorithm 2
$Midpoint(X)$	Returns the midpoint of a line segment
$Slope(X)$	Returns the slope of a line segment
$MaxPtDistance(S)$	Returns the maximum distance between any pair of points in the set S
$GetBisector(X)$	Returns the perpendicular bisector line of the given line segment
$HasIntersection(X, Y)$	Returns whether two line segments intersect at a point
$GetIntersection(X, Y)$	Returns the intersection point of two line segments
$DistanceToLine(P, X)$	Returns the Euclidean distance between point P and line segment X
$Mean(S)$	Returns the mean of the points in the set S

The final result is a set of line segments that relates to the lines and center circle on the field. Line segments that are under a certain threshold in length undergo a simple circle detection routine, detailed in Algorithm 3, to find the location of the center circle. This approach can detect circle and line segments up to 4.5 m away in experiments.

Algorithm 3 Detect circle from line segments.

Input: A set of line segments S
Output: A boolean flag whether a circle is detected
Output: The centre $C \in \mathbb{R}^2$ of the detected circle

$P \leftarrow \emptyset$
for all Line segments $X, Y \in S$ such that $X \neq Y$ **do**
  $X_B \leftarrow GetBisector(X)$
  $Y_B \leftarrow GetBisector(Y)$
  if $HasIntersection(X_B, Y_B)$ **then**
    $C \leftarrow GetIntersection(X_B, Y_B)$
    if $DistanceToLine(C, X) \approx 0.75$ **and** $DistanceToLine(C, Y) \approx 0.75$ **then**
      $P \leftarrow P \cup \{C\}$
    end if
  end if
end for
if $|P| \geq 5$ **and** $MaxPtDistance(P) < 0.75$ **then**
  return true, $Mean(P)$
end if
return false, $(0, 0)$

*: Note that the number 0.75 in this algorithm is related to the expected circle radius in metres.

3.4 Goal Detection

In the vision module, detection of the white goal posts includes two stages. In the first stage, the image is binarized using color segmentation of the color white, defined as a particular region of the HSV color space by the user. Horizontal and vertical lines are then extracted from the binarized image using probabilistic Hough line detection [9]. The detected vertical and horizontal line segments are merged together to get fewer, bigger lines using a similar approach as for field line detection. The main idea behind merging is to have one-line segment on each goal post, instead of two on each side. Moreover, by merging line segments this method can deal with only partially observable goal posts, and motion-blurred images in which the background may blend into some part of the goal post.

In the second stage of goal post detection, each of the vertical line segments that do not meet the following criteria are rejected.

a) The length of the line segment must be within a certain range, dependent on the distance from the robot to the projected bottom point of the line.
b) The bottom of the line must be within the field.
c) The top of the line must be above the estimated horizon.
d) On a kid-size field, if the goal post candidate is more than 2 m from the robot there must be one horizontal line segment close to the candidate.

3.5 Obstacle Detection

The rules of the RoboCup humanoid competition state that all robots should have mostly black feet. Thanks to these rules, obstacle detection is based on color segmentation of the black color range that is defined by the user. Then search for black connected components within the field

boundary, and if a detected connected component is large enough and within a certain predefined distance interval from the robot, the lowest point of the component is returned as the egocentric world coordinates of an obstacle. An image mask is implemented that is dependent on the position of the head to prevent the detection of own body parts as obstacles. This allows the regions of the image that are expected to contain parts of the own body to be ignored. This approach is able to detect robots at distances of up to 2 m.

4 Bipedal Walking

The gait of the robots is formulated in three different spaces: joint space, inverse (Cartesian) space and abstract space. The last one is a convenient formulation for humanoid robots for balancing and walking. Starting from a halt pose defined in the abstract space, the central pattern adds waveforms features like leg lifting, leg swinging, arm swinging, among others. The resulting abstract configuration is then transformed to the inverse space where more motion components are added. The result is finally converted into joint space to command the actuators. Several feedback mechanisms have been implemented on top of the open-loop gait that help to stabilize the robot. Each of these mechanisms acts as a PID-feedback controlling the fused angle deviations of the robot by adding corrective actions to the central pattern generated waveforms. The fused angles are an intuitive representation of orientations that offer benefits over Euler angles for balance. These mechanisms, shown in Fig. 2, include arm angle, hip angle, continuous foot angle, support foot angle and the centre of mass (CoM) shifting. The step timing is computed using a capture step framework, based on the lateral CoM state.

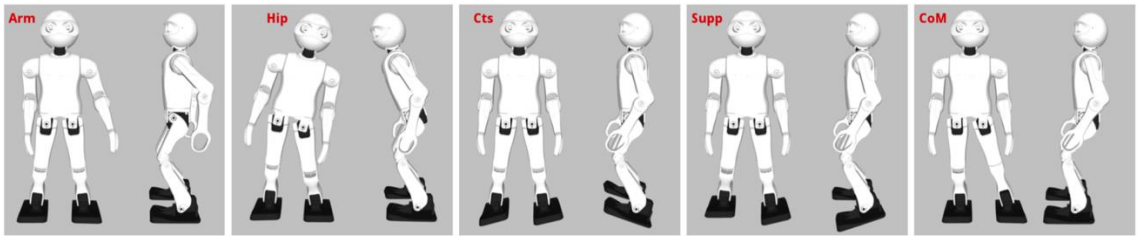

Fig. 2: Feedback mechanisms to stabilize the robot.

5 Soccer Behavior

Based on the visual perception of the game state, including ball detections, obstacle detections, and the estimated pose of the robot on the field, the robots still need to decide on and execute a strategy for playing soccer. This primarily involves localizing the ball and scoring a goal while avoiding obstacles, but also extends to team communications, team play, and cooperation of the game using the information from the RoboCup Humanoid league game controller. A custom two-layered hierarchical finite state machine (FSM) has been implemented for this purpose and runs in a separate behavior node. The lower of these two layers is referred to as the Behavior FSM and is responsible for implementing low-level skills such as searching for the ball, going to the ball, dribbling and kicking. The upper layer is referred to as the Game FSM, and builds on the skills implemented in the lower layer to implement game behaviors such as default ball handling, which attempts to kick or dribble a ball into goals, and positioning, which is used for the auto-positioning setup phase during a kickoff. In general, the game states combine groups or sequences of skills to execute certain soccer game state-specific behaviors.

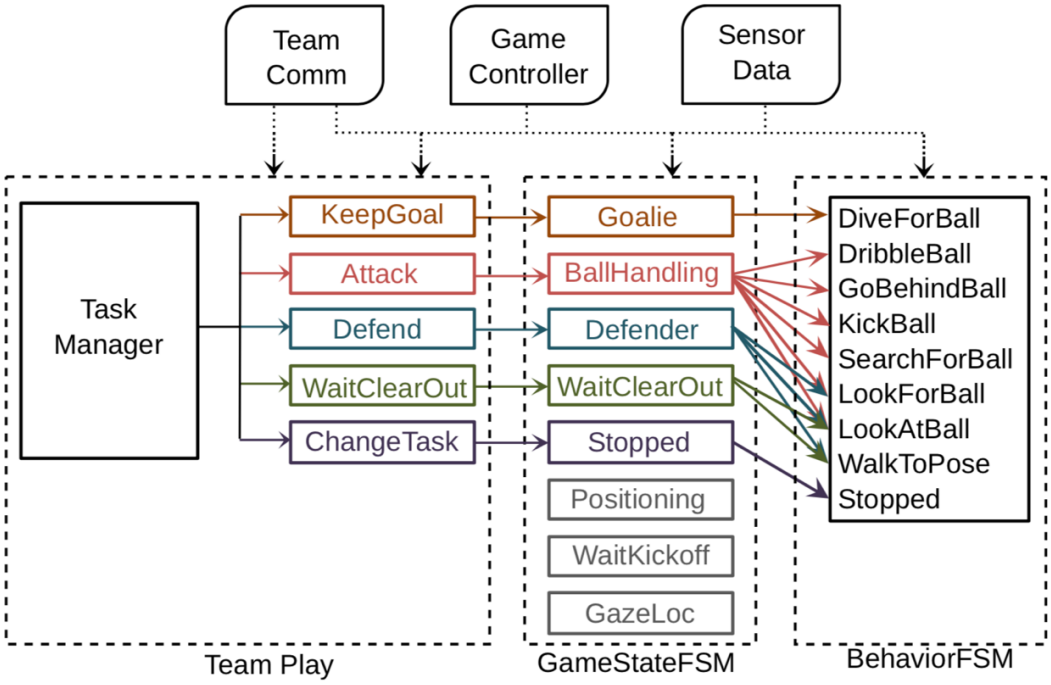

Fig. 3: The structure of soccer behaviors and team play.

6 Team play

Teams participating in the TeenSize class in RoboCup 2017 can be composed by a maximum of three robots: one goalkeeper and two field players. They define dynamic Player Tasks which are frequently reassigned during the game. This task tells the robot what it is supposed to do according to its own state in the field and the state of its teammates. The following five tasks are defined: Attack, Defend, KeepGoal, ChangeTask and WaitClearOut. A goalkeeper can only be assigned the task KeepGoal, while a field player can alternate between Attack, Defend, ChangeTask and WaitClearOut. In addition, a task manager is defined which is in charge of the safe assignment of these tasks. Each of these tasks is associated with a respective state of the game FSM.

A robot with Attack task, known as striker, has active interaction with the ball. In possession of the ball, the robot will try to score either by kicking directly or by dribbling to get a better position for kicking the ball. The robot will also reach the ball and search for the ball in case the robot does not possess it.

A robot with Defend task, is known as defender. The defender robot is not supposed to have contact with the ball but to be ready to change its task and approach the ball if necessary. Its position is defined by a vector coming from the middle of its own goal towards the position of the ball. The magnitude of this vector is defined proportional to the distance of the ball to the own goal and saturated in order to avoid collisions with team members. In this manner, the robot is able: a) to block opposite direct shots, b) to be ready for one-vs-one fights, and c) to get possession of the ball in case the previous striker is taken out of the match. With respect to the orientation, the robot tries to look in the direction of the ball. In order to avoid collisions

with other teammates, the initial shortest path (straight line) is modified such that any teammate in between the path is surrounded.

A robot with KeepGoal Task, known as goalkeeper, is supposed to dive in some direction or clean out the ball (managed by the task manager).

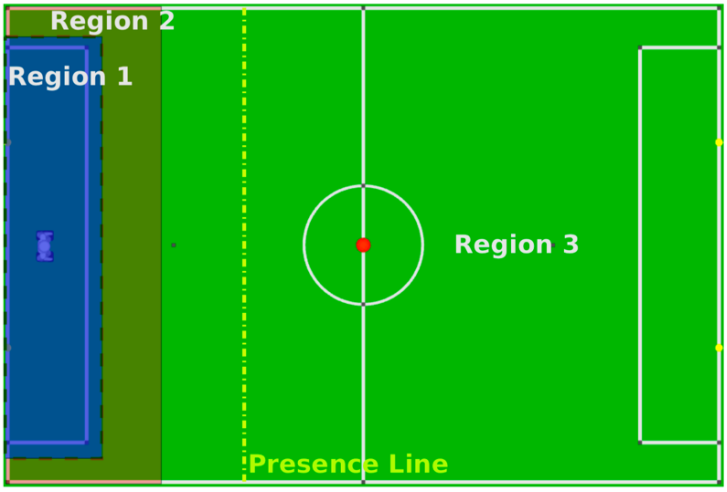

Fig. 4: The behavior of goalkeeper in different regions.

Areas determining goalkeeper ball clearing behavior. In region 1, which is the goal area with an additional outer tolerance, goalkeeper needs to clear out the ball. In region 2, the goalkeeper clears out the ball if there is no field player between their own goal and the presence line (yellow dotted line). In region 3, the robot remains in its goal.

Because the RoboCup Humanoid League has a special rule that prohibits more than one robot to be in the own area for more than 10 s, there need to be a special task called WaitClearOut. When a robot announces that it will clear out the ball from the own goal area, a WaitClearOut task reassignment occurs. Other robots go closer to the ball without entering the own goal area to avoid the illegal defense. The path is planned such that the robot will not block the shot from any other robot clearing out the ball and two robots will not collide with each other.

The task assignment is based on an asynchronous request-and-response system that ensures that there is only one robot actively interacting with the ball, which prohibits, for example, two robots try to kick the ball simultaneously which could lead to team self-collisions.

There can be task reassignment between two field players. In such conditions, the striker has a higher priority than the defender. If the defender finds itself in a better position to possess the ball, the defender sends a request to change task. If the striker confirms that its teammate is in a better position, the request from the defender is accepted. Then the striker is assigned to ChangeTask and sends back a response. After receiving this response, the defender changes its task to Attack and sends a confirmation such that the robot with task ChangeTask can change to the Defend task. If the request is refused, both robots keep their current tasks.

7 Conclusion

The team NimbRo had a very good game performance in RoboCup 2017. Their robots scored 37 goals in 12 matches and did not receive any goal. 27 goals were scored during seven games of the TeenSize tournament and the remaining 10 goals were scored in five Drop-in games. In the Drop-in games, the individual skills were tested, and their robots have obtained 21 points in total, having a margin of 19 points to the team in second place. Dynaped withstanded the push from a 1.5 kg pendulum which was retracted by 55 cm, and also completed high kick with a height of 8 cm. Igus performed a jump of height 4.5 cm, remaining 0.192 s in the air and stand stable afterwards. This proved the robustness of the methods they used. However, there are some problems with their robots, for example, the moving speed is quite low, and the stability is not so good. Although the stability of their robots has made great progress in the latest RoboCup 2019, they still have to work hard to update their robots if they want to achieve the official goal of RoboCup.

References

[1] P. Allgeuer, H. Farazi, M. Schreiber, and S. Behnke, "Child-sized 3D Printed igus Humanoid Open Platform," in *Proceedings of 15th IEEE-RAS Int. Conference on Humanoid Robots (Humanoids)*, 2015.

[2] P. Allgeuer, M. Schwarz, J. Pastrana, S. Schueller, M. Missura, and S. Behnke, "A ROS-based software framework for the NimbRo-OP humanoid open platform," in *8th Workshop on Humanoid Soccer Robots, Int. Conf. on Humanoid Robots*, 2013.

[3] U. Ramer, "An iterative procedure for the polygonal approximation of plane curves," *Computer graphics and image processing*, vol. 1, no. 3, pp. 244–256, 1972.

[4] G. Bradski, *Dr. Dobb's Journal of Software Tools*, 2000.

[5] N. Dalal and B. Triggs, "Object detection using histograms of oriented gradients," in *Pascal VOC Workshop, ECCV*, 2006.

[6] R. Lienhart and J. Maydt, "An extended set of haar-like features for rapid object detection," in *International Conference on Image Processing*, vol. 1, 2002, pp. I–900.

[7] S. Liao, X. Zhu, Z. Lei, L. Zhang, and S. Z. Li, "Learning multiscale block local binary patterns for face recognition," in *Advances in Biometrics*. Springer, 2007, pp. 828–837.

[8] J. Canny, "A computational approach to edge detection," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, no. 6, pp. 679–698, 1986.

[9] J. Matas, C. Galambos, and J. Kittler, "Robust detection of lines using the progressive probabilistic Hough transform," *Computer Vision and Image Understanding*, vol. 78, no. 1, pp. 119–137, 2000.