Ball Distance Estimation Using Machine Learning on an Omnidirectional Camera for a Middle Size Soccer Robot

Danni Syahputra
Department of Electrical Engineering
Politeknik Negeri Batam
Batam, Indonesia
dannisyahputra36@gmail.com

Anugerah Wibisana
Department of Electrical Engineering
Politeknik Negeri Batam
Batam, Indonesia
wibisana@polibatam.ac.id

Masdika Aliman
Department of Electrical Engineering
Politeknik Negeri Batam
Batam, Indonesia
masdikaaliman@gmail.com

Ichsan Fajar Yudika
Department of Electrical Engineering
Politeknik Negeri Batam
Batam, Indonesia
Ichsanfyudika@gmail.com

***Abstract*—The RoboCup Middle Size League (MSL) is a robotic soccer competition where fully autonomous robots play soccer using onboard perception and decision-making systems. Accurate ball perception, especially distance estimation, is crucial for effective navigation and control. This paper presents a ball distance estimation system for MSL robot soccer using an omnidirectional camera combined with machine learning models. Bounding box parameters from object detection, along with the width and height of the omnidirectional camera frame, are used as inputs for regression models to predict the ball's distance, which is then converted into local X and Y coordinates. Four regression models—K-Nearest Neighbor (KNN), Support Vector Regression (SVR), Random Forest Regression (RFR), and Multilayer Perceptron (MLP)—were evaluated in ground and airborne ball conditions. RFR and MLP achieved the highest accuracy, with MLP outperforming all other models based on experimental results. These findings demonstrate the feasibility of the proposed approach, with MLP emerging as the most reliable model for robust and accurate ball distance estimation in MSL robot soccer.**

Keywords—Robot Soccer, Machine Learning, Omnidirectional Camera, Object Detection.

I. INTRODUCTION

The RoboCup Middle Size League (MSL) is a robotic soccer competition that features two teams, each with five independent robots, playing on an indoor pitch. Each robot relies on its own sensors and embedded computing units to perceive the game environment, process information, and actuate movements, all without human assistance [1]. To support effective gameplay strategies, robots must be able to estimate how far the ball is from their current location. Many MSL teams equip their robots with omnidirectional vision, usually achieved through catadioptric designs that combine an upward-facing camera and a convex mirror, often spherical, parabolic, or hyperbolic, to capture a full 360º view [2]. In addition, while monocular cameras or depth cameras are also commonly used, omnidirectional cameras offer a distinct advantage due to their ability to capture the full 360º field of view. This comprehensive coverage allows the system to detect and estimate the distance of the ball from any direction, which is essential in dynamic game scenarios.

In previous implementations, the distance to the ball detected by the omnidirectional camera was estimated using a linear interpolation method [3]. This method used a lookup table with pixel distances and their corresponding real-world distances. The distance was estimated by finding the nearest reference points and applying linear interpolation between them. After obtaining the ball's distance, it was converted into local X and Y coordinates, which were then used by the robot's main system and subsystems for decision-making. This approach worked reasonably well when the ball was on the ground. However, it caused noticeable errors for airborne balls because the lookup table was built only from ground-level data and assumed a linear mapping that does not apply when the ball is in the air.

To address this issue, a machine learning method is employed to estimate the ball's distance from omnidirectional images. The approach utilizes features extracted from the ball detection stage, particularly the bounding box parameters of the detected ball, along with the overall frame dimensions (width and height). These data are then processed and converted into meaningful features for model training. The target label for training is the actual distance between the robot and the ball, obtained through ground-truth measurements. The dataset was collected under various conditions, including when the ball was on the ground, airborne, and partially hidden. In this study, several machine learning models are assessed, including K-Nearest Neighbor (KNN), Support Vector Regression (SVR), Random Forest Regression (RFR), and Multilayer Perceptron (MLP).

The use of these conventional machine learning models is motivated by the simplicity of the regression task and the need for computational efficiency. Since the object detection stage already requires substantial processing power, employing lightweight regression models ensures real-time performance on the robot's onboard system without compromising estimation accuracy. The proposed implementation is intended to reduce significant errors that often arise when the ball is airborne, while preserving reliable accuracy when the ball remains on the ground. Ultimately, enhancing distance estimation precision is expected to improve the robot's overall performance and increase its competitiveness in matches.

This paper's remaining sections are arranged as follows: The method employed in this study is described in Section II. The results of the experiment are shown and discussed in Section III. Section IV wraps up the paper by discussing the general findings.

II. METHODOLOGY

The proposed method uses input from an omnidirectional camera to estimate the distance to the ball. As shown in Fig. 1, the camera captures an image that is first processed through an object detection stage to identify the ball. Object detection involves classifying objects and localizing them within the image by generating a bounding box around each detected instance [4]. The bounding box parameters of the detected ball, together with frame dimensions of the omnidirectional image, are extracted and transformed into input features for the machine learning regression model. This model estimates the distance between the robot and the ball, which is subsequently converted into local position coordinates (X, Y). These coordinates are then utilized by the robot's main system and subsystems to support decision-making during gameplay.

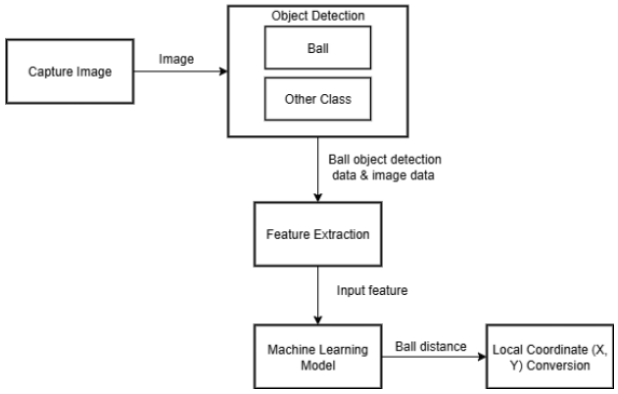


Fig. 1. System Diagram.

A. Object detection

Ball detection in the omnidirectional camera uses an object detection approach. In this study, the detection model employed is YOLOv11, developed by Ultralytics [5]. Each detection branch ends with convolutional layers followed by a Detect layer, which outputs bounding box predictions, objectness scores, and class scores [6]. In this work, only the bounding box predictions for the ball are used, which are then converted into features for the machine learning regression model.

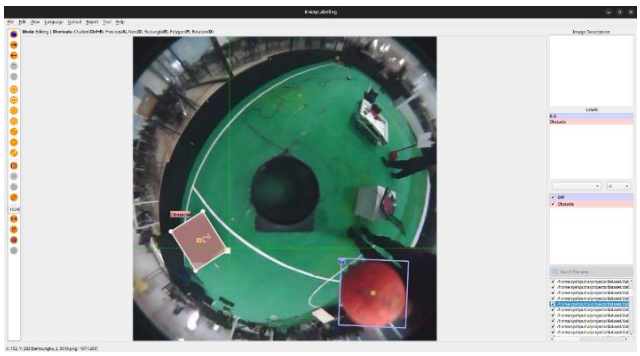


Fig. 2. Dataset annotations using X-AnyLabeling.

The object detection model was trained on a custom dataset of 1,800 annotated images. The annotations were made using the X-AnyLabeling software [7]. As shown in Fig. 2 and exported in the YOLO format, the dataset was divided into 20% for validation, 10% for testing, and the rest for training. This dataset was specifically used for the object detection model. The images were collected from an indoor soccer field environment, with the ball placed in various conditions, including on the ground, airborne, partially occluded, and positioned at different distances and angles relative to the robot. The trained object detection model was then integrated into the system to produce ball detection outputs, which were further processed as input features for the distance regression model.

B. Feature Extraction

After detecting the ball, the bounding box outputs are processed to extract numerical features that are used as inputs for the regression model during both training and real-time operation. In this study, six features are used, derived from the bounding box parameters and the corresponding image frame. The first feature is the pixel distance between the image center and the center of the ball's bounding box, calculated as in (1).

$$d_{pix} = \sqrt{\left(x_{ball} - x_c\right)^2 + \left(y_{ball} - y_c\right)^2} \tag{1}$$

Here, x_{ball} and y_{ball} represent the coordinates of the ball's bounding box center, while x_c and y_c correspond to the coordinates of the image frame's center. The second feature is the angle of the ball relative to the image center, calculated as in (2).

$$\theta = arctan2\left(y_{ball} - y_c,\ x_{ball} - x_c\right) \tag{2}$$

$$\theta_{deg} = \begin{cases} \theta \times \dfrac{180}{\pi}, & if\ \theta \geq 0 \\ \theta \times \dfrac{180}{\pi} + 360, & if\ \theta < 0 \end{cases} \tag{3}$$

The angle is then converted into degrees and normalized to the range [0°, 360°], as shown in (3). The third feature is the aspect ratio of the bounding box, which can provide useful information when the ball is partially occluded. The aspect ratio is calculated as in (4).

$$r_{asp} = \frac{w_{box}}{h_{box}} \tag{4}$$

Here, w_{box} and h_{box} denote the width and height of the ball's bounding box, respectively. The fourth feature is the bounding box area, which generally increases when the ball is airborne. This characteristic helps the regression model distinguish between ground-level and airborne conditions, thereby improving distance estimation accuracy. The bounding box area is defined in (5).

$$A_{box} = w_{box} \times h_{box} \tag{5}$$

The fifth and sixth features are the horizontal and vertical offsets, which indicate the displacement of the ball's center relative to the center of the image frame along the x- and y-axes. These features reflect the directional bias of the ball's position within the frame and are given in (6) and (7).

$$\Delta x = x_{ball} - x_c \tag{6}$$

$$\Delta y = y_{ball} - y_c \tag{7}$$

Where Δx is the horizontal offset and Δy is the vertical offset. The input features and target of the model can be defined as follows:

$$x = \left[d_{pix},\ \theta_{deg},\ r_{asp},\ A_{box},\ \Delta x,\ \Delta y \right] \quad (8)$$

$$y = d_{real} \quad (9)$$

Here, in (8) x represents the input feature vector of the model, and in (9) y is the target output. The value d_{real} denotes the ground-truth distance (in meters) from the robot to the ball, which was manually measured during dataset collection for training the machine learning model. A total of six numerical features are extracted from the ball's bounding box and the image frame: pixel distance, angle, aspect ratio, bounding box area, horizontal offsets, and vertical offsets. These features capture both positional and geometric characteristics of the ball, allowing the regression model to learn the relationship between visual observations and the actual distance.

The dataset for regression was collected under diverse experimental conditions, including when the ball was placed on the ground, when it was airborne, and at multiple distances (up to 5 meters, beyond which detection becomes difficult), as well as from various angular positions relative to the robot. In total, 510 samples were collected to train and evaluate the regression model. Although this experiment focuses on estimating the distance to a ball, the same feature extraction method can generally be applied to other objects. The key factor is the object's physical size, since the bounding box area feature depends directly on the object's dimensions, while the other features are usually applicable across different objects.

C. Machine Learning Models

The regression model dataset was constructed from the features extracted in the previous stage, consisting of 510 samples. These samples were divided into 80% for training and 20% for testing. Prior to training, all features were standardized using the StandardScaler method to ensure a uniform scale. Three regression models, K-Nearest Neighbor (KNN) regression, Support Vector Regression (SVR), and Random Forest Regression (RFR), were developed using the Scikit-learn library [8]. The Multilayer Perceptron (MLP) model was developed using the PyTorch deep learning framework [9]. To provide a clearer understanding, each regression model is briefly described as follows:

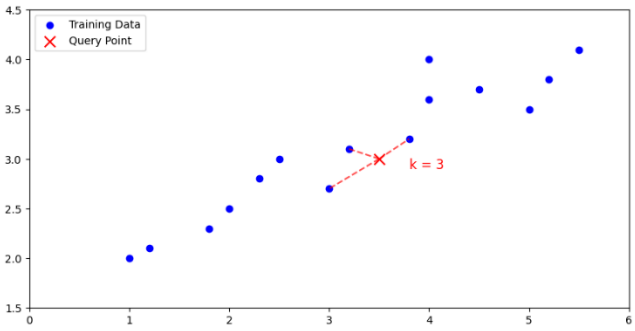


Fig. 3. KNN regression visualization (k=3).

1) K-Nearest Neighbor (KNN) Regression: The KNN algorithm is a straightforward yet powerful regression technique. By averaging the goal values of its k closest neighbors in the training set, it forecasts the output for a new sample. Because it maintains the training data and uses it directly for prediction, this method eliminates the need for an explicit training phase. Its performance is significantly influenced by the distance metric, usually the Euclidean distance, and the number of neighbors (k) [10]. In this study, the optimal hyperparameters, including k, the weighting scheme, and the Minkowski power parameter (p), were determined using grid search with cross-validation. The query point is predicted by averaging the target values of its nearest neighbors, as illustrated in Fig. 3.

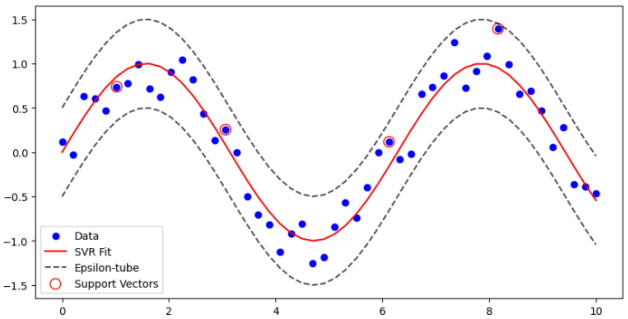


Fig. 4. Support Vector Regression visualization.

2) Support Vector Regression (SVR): SVR is a regression technique that estimates target values within a certain tolerance margin, represented by the symbol ε. SVR creates a ε-insensitive tube around the regression function by permitting deviations up to ε instead of minimizing the error of each training point. The final model is defined by support vectors, which are training samples outside of this tube. Both linear and non-linear correlations between features and the target variable can be modeled by SVR by integrating kernel functions such as the radial basis function (RBF), linear, or polynomial [11]. In this study, SVR was implemented using the RBF kernel, with hyperparameters C (Regularization parameter), ε (Epsilon-insensitive tube), and γ (kernel coefficient) optimized through grid search with cross-validation. As illustrated in Fig. 4, the regression function (red line) is bounded by an ε-tube, with support vectors highlighted as red circles.

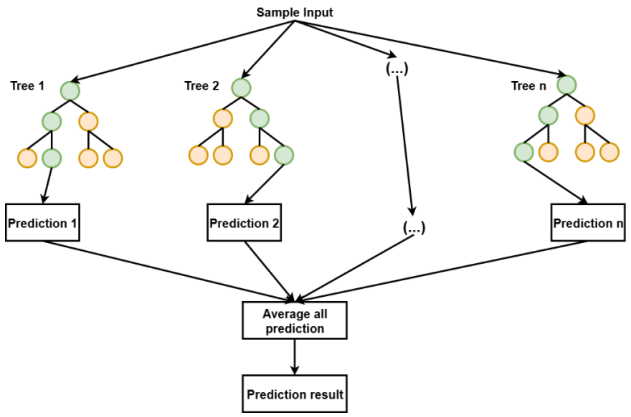


Fig. 5. Random Forest Algorithm.

3) Random Forest Regression (RFR): As illustrated in Fig. 5, RFR is an ensemble learning technique that performs

regression by combining several decision trees. Each tree independently produces a prediction, and the final output is obtained by averaging the results of all trees. By combining predictions from many decision trees, RFR improves robustness and generalization in contrast to a single tree, which is prone to overfitting [12]. In this study, RFR was implemented with three key hyperparameters: n_estimators (number of trees), max_depth (maximum tree depth), and min_samples_split (minimum samples required to split an internal node). These hyperparameters were optimized using grid search with cross-validation.

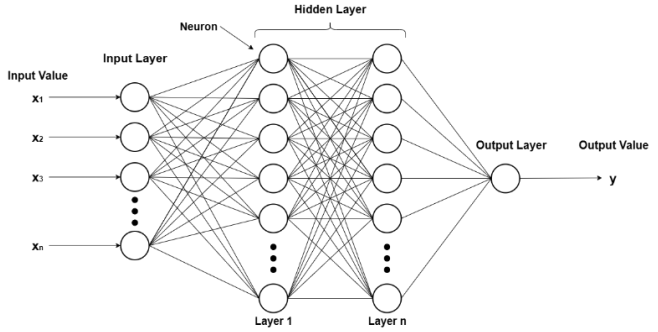


Fig. 6. Multilayer Perceptron Architecture.

4) Multilayer Perceptron (MLP): As illustrated in Fig. 6, the Multilayer Perceptron (MLP) is a type of artificial neural network (ANN) composed of an input layer, one or more hidden layers, and an output layer. The input layer has as many neurons as there are features in the dataset, while the output layer in this regression task comprises a single neuron to estimate the distance of the ball from the robot. Unlike linear models, MLP can identify more complex patterns due to their hidden layers, which allow the learning of non-linear relationships between features and the target variable [13] [14]. Training is performed through forward propagation to compute outputs, followed by backpropagation to calculate errors, with weights updated via an optimization algorithm such as Adam. The network architecture, including the number of hidden layers and neurons, is generally determined empirically to balance underfitting and overfitting [15]. This study used the hyperbolic tangent (Tanh) activation function with two hidden layers of 16 neurons each.

The ball distance estimation model was trained using the four regression methods described above. For the KNN, SVR, and RFR models, hyperparameters were optimized using grid search with 3-fold cross-validation. Meanwhile, the MLP model parameters were empirically tuned based on experimental observations to achieve stable convergence and prevent overfitting. The final training parameters used for each model are summarized in Table I.

TABLE I. REGRESSION MODEL TRAINING PARAMETERS

Model	Training Parameters
KNN	n_neighbors = 3, p = 8, weights = 'distance'
SVR	Kernel = RBF, C = 10, ε = 0.01, γ = 'scale'
RFR	n_estimators = 50, max_depth = None, min_samples_split = 2
MLP	Activation = Tanh, Hidden layers = [16, 16], Batch size = 16, Learning rate = 0.001, Epochs = 500

D. Ball Distance to Local X and Y Positions

After the machine learning model estimates the ball's distance from the robot, the value is converted into local coordinates (X, Y). The transformation is defined as follows:

$$X = d_{pred} \times sin\left(\frac{\theta_{deg} \times \pi}{180}\right) \quad (10)$$

$$Y = d_{pred} \times cos\left(\frac{\theta_{deg} \times \pi}{180}\right) \quad (11)$$

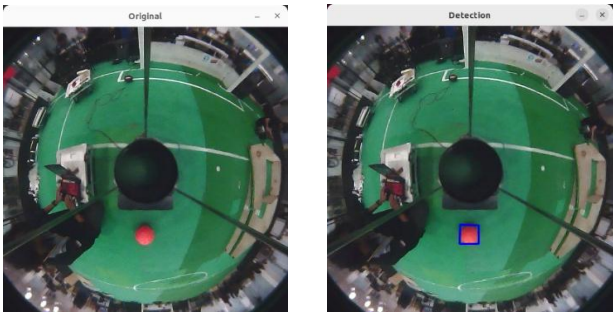
In (10) and (11), d_{pred} represents the estimated distance to the ball, while θ_{deg} denotes the ball's angle relative to the center of the omnidirectional camera frame, as defined in (3). This conversion shows where the ball is in the local two-dimensional coordinate system (X, Y) of the robot. After that, these values are sent to the robot's main system or other subsystems.

III. RESULTS

This section presents the study's results, starting with object detection of the ball, then evaluating trained regression models on the test dataset. The models are then tested on sample cases under different conditions, including ground and airborne balls, as well as various angular positions. Finally, predicted distances are converted into the robot's local X and Y coordinates.

A. Object Detection Testing

Ball detection was tested using the trained object detection model. Fig. 7 shows the detection results: frame (a) without detection, and frame (b) with the detected ball highlighted by a bounding box.



(a) (b)

Fig. 7. Ball detection: (a) frame without detection, (b) frame with detection.

The extracted bounding box information, along with the width and height of the omnidirectional camera frame, was used as input features for the regression models to estimate the distance from the robot to the ball.

B. Machine Learning Training Evaluation

The machine learning models for ball distance estimation were trained using the extracted features described in the previous section. The performance of each model was evaluated using four standard regression metrics: mean squared error (MSE), root mean squared error (RMSE), mean absolute error (MAE), and the coefficient of determination (R^2). Table II summarizes the evaluation results. These metrics collectively assess model accuracy, consistency, and generalization capability.

TABLE II. EVALUATION OF EACH REGRESSION MODEL

Model	MSE	RMSE	MAE	R^2
KNN	0.0864	0.2939	0.164	0.9472
SVR	0.1486	0.3854	0.1916	0.9091
RFR	0.0095	0.0974	0.0644	0.9942
MLP	0.0092	0.096	0.0529	0.9943

The results indicate that MLP and RFR achieved the highest performance, both yielding very low errors (MSE below 0.01, RMSE around 0.096–0.097, and MAE between 0.05 and 0.06) and high determination coefficients (R^2 above 0.994). This demonstrates that both models accurately captured the relationship between the extracted features and the ball's actual distance. In comparison, KNN and SVR showed weaker results. SVR had the lowest R^2 value of 0.9091 and higher error metrics (MSE = 0.1486, RMSE = 0.3854, MAE = 0.1916), while KNN performed moderately better with R^2 = 0.9472 but still exhibited higher errors than RFR and MLP.

C. Model Testing with Sample Data

To further evaluate the predictive performance of the trained models, a set of sample data was tested under two different conditions: when the ball was on the ground and when it was airborne. In the ground condition, the ball could be detected up to 5 meters, while in the airborne condition, detection was limited to approximately 1.5 meters before the ball left the camera's frame. In this experiment, the ball was placed in front of the robot as well as at various angular positions around it.

TABLE III. BALL ON GROUND (FRONT POSITION)

No	Actual Distance (meter)	Predicted Distance			
		KNN	*SVR*	*RFR*	*MLP*
1	0.5	0.5	0.5003	0.5122	0.5001
2	1	1.05	0.9611	1.0282	1.0035
3	1.5	1.6067	1.377	1.531	1.518
4	2	1.8567	2.0203	2.059	2.0272
5	2.5	2.5467	2.6067	2.4902	2.5075
6	3	3.59	2.9155	3.1462	2.9668
7	3.5	3.54	3.4909	3.3882	3.5434
8	4	4.1	3.8374	4.0526	4.0597
9	4.5	4.3933	4.0982	4.5008	4.5012
10	5	4.6933	4.398	4.912	4.9523

TABLE IV. EVALUATION FOR GROUND (FRONT)

Model	MSE	RMSE	MAE	R^2
KNN	0.0502	0.224	0.149	0.9757
SVR	0.0586	0.242	0.1549	0.9716
RFR	0.005	0.0706	0.054	0.9976
MLP	0.001	0.0316	0.0242	0.9995

The experimental results obtained with the ball positioned on the ground directly in front of the robot, as shown in Table III and the evaluation metrics in Table IV, indicate all models produced accurate predictions. RFR and MLP consistently outperformed KNN and SVR. MLP had the best performance with MSE = 0.001, RMSE = 0.0316, MAE = 0.0242, and R^2 = 0.9995, nearly perfect predictions. RFR also performed well with MSE = 0.005, RMSE = 0.0706, MAE = 0.054, and R^2 = 0.9976, slightly less precise but still reliable. KNN and SVR gave acceptable results with R^2 values between 0.97 and 0.98, though their accuracy decreased beyond 3 meters.

TABLE V. AIRBORNE BALL (FRONT POSITION)

No	Actual Distance (meter)	Predicted Distance			
		KNN	*SVR*	*RFR*	*MLP*
1	0.25	0.2667	0.4313	0.2770	0.2124
2	0.5	0.6167	0.3074	0.4812	0.4442
3	0.75	0.9000	0.9467	0.8432	0.8414
4	1	1.1233	1.3985	0.9126	1.0141
5	1.25	1.3233	1.9931	1.2324	1.2596
6	1.5	4.7333	2.7887	1.3670	1.4722

TABLE VI. EVALUATION FOR AIRBORNE (FRONT)

Model	MSE	RMSE	MAE	R^2
KNN	1.7519	1.3236	0.6189	-8.6104
SVR	0.4134	0.643	0.5002	-1.2678
RFR	0.0059	0.0768	0.0628	0.9676
MLP	0.0023	0.0482	0.0394	0.9872

For the airborne ball condition, the test results are shown in Table IV, with the evaluation metrics summarised in Table V. Differences between models became more pronounced. RFR delivered accurate predictions with low error metrics: MSE = 0.0059, RMSE = 0.0768, MAE = 0.0628, R^2 = 0.9676. MLP performed slightly better: MSE = 0.0023, RMSE = 0.0482, MAE = 0.0394, R^2 = 0.9872. KNN and SVR struggled when the ball was airborne. KNN had large errors (MSE = 1.7519, RMSE = 1.3236, MAE = 0.6189) and a negative R^2, indicating worse predictions than the mean. SVR also performed poorly, with R^2 = –1.2678, showing ineffective generalisation

TABLE VII. BALL AT VARIOUS ANGULAR POSITIONS

No	Cnd	Ang (°)	Act Dist (m)	Predicted Distance			
				KNN	*SVR*	*RFR*	*MLP*
1	Grd	90	0.25	0.3433	0.2271	0.3858	0.2167
2	Air	90	0.7	0.8833	0.6723	0.6056	0.7427
3	Grd	145	0.75	0.64	0.762	0.7452	0.7358
4	Air	145	1	1.3	1.4375	1.1754	1.206
5	Grd	180	1.25	1.0067	1.102	1.1264	1.1111
6	Air	180	0.65	0.7167	0.7118	0.5942	0.6527
7	Grd	270	1.5	2.4733	1.2972	1.5318	1.6987
8	Air	270	0.8	1.2067	-0.3548	0.8142	0.7749
9	Grd	315	2	1.99	2.012	1.9362	1.9136
10	Air	315	1.5	2.1733	2.8669	1.3238	1.359
11	Grd	0	2.9	2.9833	2.8977	3.0052	2.8794
12	Air	0	1.3	2.6433	2.5248	1.3032	1.3003
13	Grd	45	3.3	4.2367	3.8054	3.463	3.2137
14	Grd	45	4.6	4.3967	4.5894	4.2128	4.2091
15	Air	45	0.4	0.5833	0.3632	0.5166	0.4904

TABLE VIII. EVALUATION FOR VARIOUS ANGLES

Model	MSE	RMSE	MAE	R^2
KNN	0.3025	0.55	0.3873	0.7796
SVR	0.3479	0.5898	0.3484	0.7465
RFR	0.0209	0.1447	0.1101	0.9847
MLP	0.0201	0.1417	0.0985	0.9854

Test results in Tables VII and VIII indicate that all models' accuracy varied with viewing angle and condition, whether on the ground or airborne. Among the four models, RFR and MLP achieved the best, most stable results across various angles. MLP had the lowest errors with MSE = 0.0201, RMSE = 0.1417, MAE = 0.0985, R^2 = 0.9854, showing high

accuracy. RFR was close with MSE = 0.0209, RMSE = 0.1447, MAE = 0.1101, R^2 = 0.9847. KNN and SVR showed degraded performance as angles deviated, with R^2 = 0.7796 and 0.7465, and higher errors, especially at steeper or partial angles.

Overall, the results show performance differences among models. KNN and SVR did well when the ball was on the ground and directly in front, where features were stable. Their performance worsened with airborne or angled views, showing less ability to handle complex spatial variations. RFR and MLP, however, remained accurate and stable across all conditions. RFR was notably robust with low error in difficult scenarios, while MLP achieved the highest accuracy.

D. Results of Predicted Distance to Local Coordinates

The predicted distance was converted into the robot's local X and Y coordinates as defined in (10) and (11). Fig. 8 illustrates the results of this conversion, showing the predicted distance of the ball, its angular position within the omnidirectional camera frame, and the corresponding local coordinates relative to the robot. These X and Y coordinates provide essential information for ball localization and serve as an important input for the robot's navigation and decision-making processes.

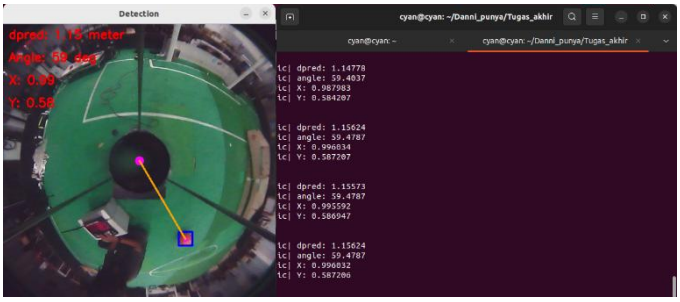


Fig. 8. Predicted distance converted to local x and y coordinates.

Overall, the results showed that the proposed distance estimation and coordinate transformation approach can be effectively integrated into the robot's perception system, providing reliable support for higher-level control and strategy in the Middle Size League robot soccer.

IV. Conclusion

This paper presented a ball distance estimation system for the Middle Size League robot soccer using an omnidirectional camera combined with machine learning models. The proposed method employed bounding box features extracted from object detection, together with the width and height of the omnidirectional camera frame, as inputs for regression models to predict the ball's distance. The predicted distances were then transformed into local X and Y coordinates, providing essential inputs for robot navigation and decision-making.

From the experimental evaluation, the four machine learning models—KNN, SVR, RFR, and MLP—showed distinct performance patterns. Under the ground condition in front of the robot, MLP achieved the highest accuracy, with the best values among all models (MSE 0.001, RMSE 0.032, MAE 0.024, and R^2 0.9995), while RFR also performed well but with slightly higher errors. KNN and SVR were less precise, with higher errors and lower R^2 values. In the airborne condition, MLP consistently achieved the highest accuracy (MSE 0.0023, RMSE 0.0482, MAE 0.0394, R^2 0.9872), followed by RFR, whereas KNN and SVR produced large errors and negative R^2 values, indicating poor generalisation. When tested at various angular positions, MLP again achieved the best results across all metrics (MSE 0.0201, RMSE 0.1417, MAE 0.0985, R^2 0.9854), with RFR slightly behind, and KNN and SVR showing significant drops in accuracy, particularly at wider or oblique angles.

Based on these results, the MLP proved to be the most suitable model for ball distance estimation, achieving the highest accuracy across ground, airborne, and angular scenarios. Its robustness makes it a strong candidate for real-time robotic perception. However, the 510 sample dataset may limit generalization, and expanding it could further improve model robustness and performance. Overall, the proposed approach successfully combines omnidirectional vision and machine learning to detect the ball, estimate its distance, and transform this information into local coordinates, demonstrating its feasibility for Middle Size League robot soccer.

References

[1] D. Nardi, I. Noda, F. Ribeiro, P. Stone, O. von Stryk, and M. Veloso, "RoboCup soccer leagues," AI Magazine, vol. 35, no. 3, pp. 77–85, 2014.

[2] G. Lopes, F. Ribeiro, and N. Pereira, "Catadioptric system optimisation for omnidirectional RoboCup MSL robots," in RoboCup 2011: Robot Soccer World Cup XV, LNCS 7416, T. Röfer et al., Eds. Berlin, Heidelberg: Springer, 2012, pp. 318–328.

[3] F. Rahmat, A. Wibisana, M. R. Fauzi, M. Aliman, and N. Firmansyah, "Implementation of intercept ball algorithm for wheeled soccer robot Barelang63," in Proc. 7th Int. Conf. Appl. Eng. (ICAE), Advances in Engineering Research, vol. 251, Atlantis Press, 2024, pp. 190–205.

[4] A. R. Pathak, M. Pandey, and S. Rautaray, "Application of deep learning for object detection," in Proc. Int. Conf. Computational Intelligence and Data Science (ICCIDS), Procedia Comput. Sci., vol. 132, pp. 1706–1717, 2018.

[5] G. Jocher and J. Qiu, "Ultralytics YOLO11," GitHub, 2024. [Online]. Available: https://github.com/ultralytics/ultralytics [Accessed: May. 28, 2025].

[6] R. Khanam and M. Hussain, "YOLOv11: An Overview of the Key Architectural Enhancements," Oct. 2024, [Online]. Available: http://arxiv.org/abs/2410.17725.

[7] W. Wang, "Advanced auto labeling solution with added features," GitHub, 2023. [Online]. Available: https://github.com/CVHub520/X-AnyLabeling.git [Accessed: May. 24, 2025].

[8] F. Pedregosa et al., "Scikit-learn: Machine learning in Python," J. Mach. Learn. Res., vol. 12, pp. 2825–2830, 2011.

[9] A. Paszke et al., "PyTorch: An imperative style, high-performance deep learning library," in Adv. Neural Inf. Process. Syst. (NeurIPS), vol. 32, Vancouver, Canada, Dec. 2019.

[10] H. A. Alsayadi, A. A. Abdelhamid, E. S. M. El-Kenawy, A. Ibrahim, and M. M. Eid, "Ensemble of Machine Learning Fusion Models for Breast Cancer Detection Based on the Regression Model," Fusion: Practice and Applications, vol. 9, no. 2, pp. 19–26, 2022.

[11] F. Zhang and L. J. O'Donnell, "Support vector regression," in Machine Learning, Academic Press, 2020, pp. 123–140.

[12] Y. Li et al., "Random forest regression for online capacity estimation of lithium-ion batteries," Appl Energy, vol. 232, pp. 197–210, Dec. 2018.

[13] H. Ramchoun, M. Amine, J. Idrissi, Y. Ghanou, and M. Ettaouil, "Multilayer Perceptron: Architecture Optimization and Training," International Journal of Interactive Multimedia and Artificial Intelligence, vol. 4, no. 1, p. 26, 2016.

[14] S. Sharma, S. Sharma, and A. Athaiya, "Activation functions in neural networks," Int. J. Eng. Appl. Sci. Technol., vol. 4, no. 12, pp. 310–316, Apr. 2020. [Online]. Available: Available: http://www.ijeast.com.

[15] G. Panchal, A. Ganatra, Y. P. Kosta, and D. Panchal, "Behaviour analysis of multilayer perceptrons with multiple hidden neurons and hidden layers," Int. J. Comput. Theory Eng., vol. 3, no. 2, pp. 332–337, Apr. 2011.