Real-Time Localization Framework for Autonomous Basketball Robots

Naren Medarametla
School of Computer Science Engineering
Vellore Institute of Technology
Chennai, India
naren.medarametla2023@vitstudent.ac.in

Sreejon Mondal
School of Electrical and Electronics Engineering
Vellore Institute of Technology
Chennai, India
sreejon.mondal2023@vitstudent.ac.in

***Abstract*— Localization is a fundamental capability for autonomous robots, enabling them to operate effectively in dynamic environments. In Robocon 2025, accurate and reliable localization is crucial for improving shooting precision, avoiding collisions with other robots, and navigating the competition field efficiently. In this paper, we propose a hybrid localization algorithm that integrates classical techniques with learning-based methods, relying solely on visual data from the court's floor to achieve self-localization on the basketball field.**

Keywords*—*Robot Localization, Autonomous Navigation, Neural Networks, Robocon

I. Introduction

The DD-Robocon is the Indian national entry-level competition to Asia-Pacific Robot Contest (ABU Robocon). The objective of the competition is for college teams to design, build, and operate robots to complete certain tasks. The tasks vary each year and are usually based on cultural, historical, or technological themes related to the host country of the international contest. For Robocon 2025, the theme was Robot Basketball, where two robots had to play the game of basketball following specific rules. The arena consisted of a court of dimensions 15m $\times$ 8m and baskets at a height of 2.4m. Due to the nature of the game, the robots had to move quickly and unpredictably, and take shots from various positions on the court. The robots had to have robust and real-time self-localization determine it's distance from the basket. In this paper we propose a vision-based localization system based on the white field lines of the basketball court. **Section II** reviews existing methods and prior research related to this work. **Section III** provides a detailed description of our proposed algorithm, approach, and model architecture. **Section IV** provides the results obtained from our experiments. **Section V** evaluates the accuracy of our approach and discusses potential directions for future work.

II. Related Work

In our earlier methodology, we employed an odometry-based approach for the self-localization of the robots. This system integrated wheel encoders and an Inertial Measurement Unit (IMU), complemented by a TF-Luna LiDAR sensor to estimate the depth between the robot and the basket. However, the TF-Luna sensor demonstrated a limited effective range of approximately 3 to 4 meters and exhibited suboptimal accuracy within this range. Additionally, a significant challenge arose due to the absence of a clearly distinguishable object or surface on the basket, which hindered reliable depth estimation.

In the subsequent approach, we employed an object detection model based on YOLOv8n, trained on a custom dataset to detect the basket. This model was utilized not only for basket detection but also for correcting the robot's yaw orientation to ensure it consistently faced the target. To estimate the horizontal distance from the basket, we defined discrete zones, with each zone representing a specific distance range. The corresponding zone, selected manually by the pilot based on visual assessment, was then used in conjunction with the detected orientation to determine the appropriate RPM for the shooting mechanism. While this approach facilitated a simplified method for distance estimation, it was inherently dependent on manual selection and coarse approximations, which could lead to reduced accuracy and consistency.

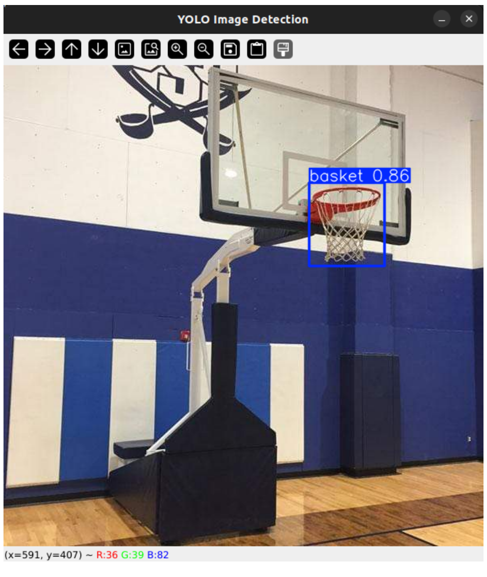

Figure I: basket detection

We improved upon the object detection approach by using the MiDaS (Multiple Depth Estimation Accuracy with Single Network) deep learning model. MiDaS is a designed for monocular depth estimation, i.e., predicting depth from a single image. It uses a ResNet-based encoder-decoder architecture, where:

- The encoder extracts features from the input image.
- The decoder upsamples these features to produce a depth map.

The approach combines the YOLOv8 model, with the MiDaS depth estimation framework for relative depth prediction. The pipeline begins by loading an input image, followed by inference using the YOLO model to localize the basket and identify its center. Simultaneously, MiDaS processes the same image to generate a dense relative depth map. Once the basket is detected, the relative depth at the center of the bounding box is extracted from the depth map. An interactive calibration step is used to relate MiDaS's relative depth to real-world distance. The user is prompted to enter known ground-truth distances for three distinct positions. These pairs of relative depth and real distance are then used to fit a linear regression model, enabling real-time conversion of relative depth to metric distance. During tracking mode, the system periodically predicts the basket's distance using the learned regression model and overlays the result on the original image alongside the visualized depth map. The output is displayed in a side-by-side format showing both RGB and color-mapped depth views for easier interpretation.

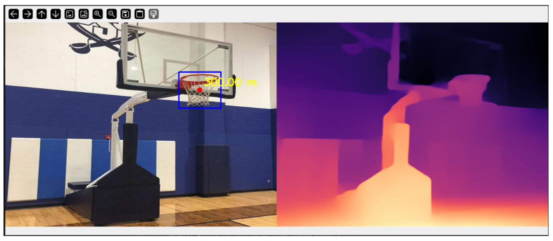

Figure II: basket depth estimation

A key limitation of this approach was its reliance on prior calibration, which required manually measuring the distance between the robot and the basket at least three times before accurate predictions could be made. In response to these constraints, we developed the current method presented in this paper, building upon insights gained from the previous techniques.

III. Methodology

Our approach is a two-step process that begins with **Preprocessing** the image, followed by passing it to the model for **Inference**.

A. *Preprocessing*

The input image having dimensions ($640 \times 480 \times 3$) is converted from the RGB color space to the HSV color space, then the white regions are masked out using two predefined HSV ranges. The image is downsampled through a radial scan, flattened, and finally passed through the neural network. Figure I shows the preprocessing pipline and Algorithm 1 gives the implementation of the downsampling algorithm.

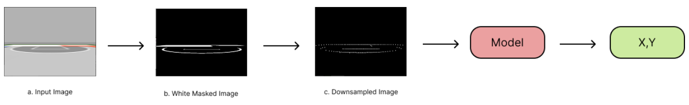

Figure III: preprocessing pipeline

Algorithm 1 Downsampling

```
    Input: Image
 1: $H \leftarrow$ Image.height
 2: $W \leftarrow$ Image.width
 3: $R \leftarrow$ Black Image
 4: for angle $\leftarrow$ 0 to 180 step 2 do
 5:     $lastPixel \leftarrow 0$
 6:     $cx \leftarrow$ W / 2
 7:     $cy \leftarrow$ H
 8:     for d $\leftarrow$ 0 to max(H, W) do
 9:         $x \leftarrow cx + d \times \cos(angle)$
10:         $y \leftarrow cy - d \times \sin(angle)$
11:         if $0 \leq x <$ W and $0 \leq y <$ H then
12:             $pixel \leftarrow Image[y][x]$
13:             if lastPixel = 255 and pixel $\neq$ 255 then
14:                 $R[y][x] \leftarrow 255$
15:             end if
16:             $lastPixel \leftarrow pixel$
17:         end if
18:     end for
19: end for
20: Return R
```

B. *Model Architecture*

The proposed model is a feedforward neural network consisting of a flattening layer followed by four fully connected layers. The first 200 pixels from the top of the image are removed to reduce the input size, as they do not carry useful information and the pixel values are scaled down to the range [0, 1].

The resulting image with dimensions ($640 \times 280 \times 1$) is flattened into a vector of size 1,79,200 and passed through the first linear layer, followed by a ReLU activation function. The subsequent layers have sizes 1024, 256, and 64, each followed by a ReLU activation. The final layer outputs a 2-dimensional vector representing the predicted x and y positions of the robot.

The rationale for using this relatively simple model lies in the simplicity of the input images and to reducing inference time.

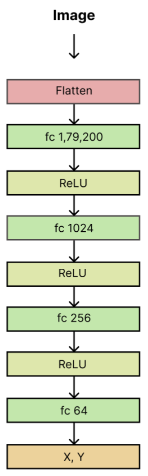

Figure IV: architecture diagram

C. Dataset

A digital twin of the robot was created and simulated in a replica of the Robocon 2025 arena using the Gazebo [1] simulator with the help of ROS 2 [2].

The robot was driven through the arena capturing images and corresponding x, y coordinates. A TimeSynchronizer was used to synchronize the frame header and the position header. A total of 6283 images were captured and split into training and test datasets with a 0.9 to 0.1 ratio. Care was taken to include every part of the arena for an unbiased dataset.

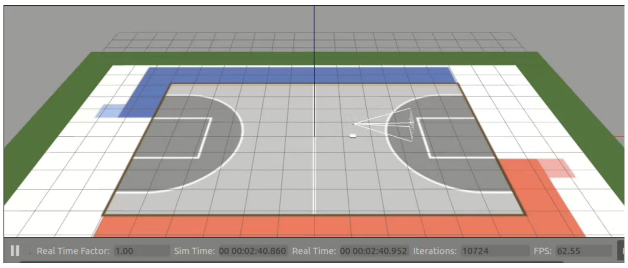

Figure V: gazebo simulation

D. Training

The model was trained for 15 epochs using an Adam optimizer [3] with an initial learning rate of 10^{-4} and a Mean Squared Error (MSE) cost function in batches of size 8.

IV. Results

Figure IV depicts the loss at each epoch throughout the training process. 8 independent images at different points of the court were again captured from the simulation and fed into the model, Figure V shows the plot between the ground truth and the prediction made with an average loss of 0.06 meters.

The code for the paper can be found in the following repository https://github.com/NarenTheNumpkin/Basketball-robot-localization

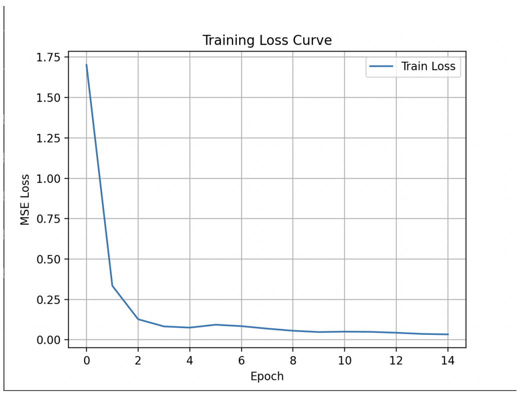

Figure VI: loss curve

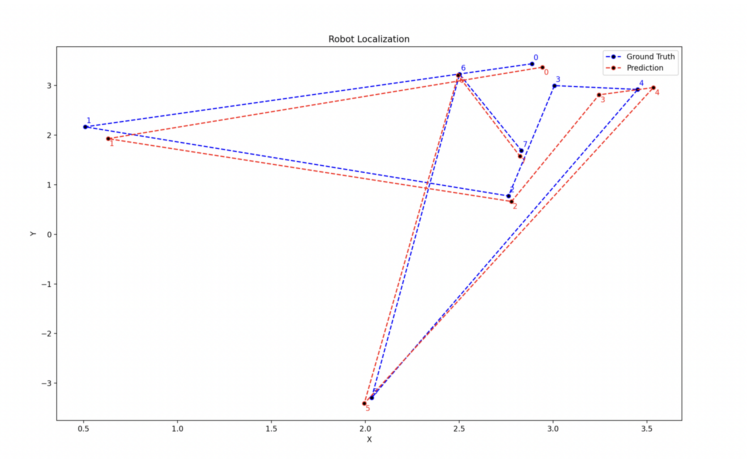

Figure VII: truth vs prediction

V. Conclusion

In this work, we presented a hybrid localization framework for autonomous basketball robots competing in the Robocon 2025 arena. Our approach relies solely on floor images processed through a lightweight feedforward neural network. By preprocessing the images to highlight salient floor features and using a simple architecture, we achieved an average prediction error of approximately 0.06 meters, demonstrating the potential of learning-based localization methods. This result shows that even with minimal sensor inputs and modest model complexity, it is possible to achieve sufficiently accurate localization. Future improvements could focus on integrating other sensors such as IMU, wheel odometry, or LIDAR with the vision-based system through filtering techniques like Kalman or particle filters to achieve more robust and reliable localization. Additionally, exploring more neural network architectures may help to improve prediction accuracy. Finally, optimizing the model for deployment on embedded platforms with limited computational resources through techniques such as pruning, quantization, or hardware acceleration could be worked upon.

References

[1] N. Koenig and A. Howard, ”Design and use paradigms for Gazebo, an open-source multi-robot simulator,” 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566), Sendai, Japan, 2004, pp. 2149-2154 vol.3, doi: 10.1109/IROS.2004.1389727.

[2] Steve Macenski, Tully Foote, Brian Gerkey, Chris

Lalancette, and William Woodall. *Robot Operating System 2: Design, architecture, and uses in the wild.* Science Robotics, vol. 7, no. 66, 2022.

[3] Diederik P. Kingma and Jimmy Ba. *Adam: A method for stochastic optimization.* arXiv preprint arXiv:1412.6980, 2014.