BLG561E Deep Learning Project Report CNN-based Localization System for an Autonomous Mobile Robot

Emre Can Contarlı
Control and Automation Engineering
Istanbul Technical University
contarli@itu.edu.tr
504232106

Fulya Yenilmez
Computer Engineering
Istanbul Technical University
fulya.yenilmez@tau.edu.tr
504232517

***Abstract*—This project has developed a CNN-based localization system for wheelchairs equipped with an RGB-D camera. The system aims to improve current localization methods using CNN. It uses synchronized data from the RGB-D camera and AMCL outputs. The CNN model is trained to predict the wheelchair's position and orientation within a predefined environment. The methodology includes data preprocessing, model selection, and training, with a custom CNN architecture and a benchmark using a pre-trained VGG19 model. The models are evaluated using an unseen test dataset. This system shows potential for visual cues in development of localization accuracy, and reducing dependence on traditional sensors, and improvement of autonomous navigation systems.**

I. Project Description

The project aims to develop a Convolutional Neural Network (CNN)-based localization system for an autonomous mobile robot equipped with an RGB-D camera. The robot in question is an autonomous wheelchair that features components such as LIDARs, an RGB-D camera, and wheel encoders. The autonomous wheelchair is shown in Fig. 1. The system will be trained to predict the wheelchair's pose and orientation within the known map of the Faculty of Electrical & Electronics Engineering's new building corridor. Currently, the robot's localization is managed by the Adaptive Monte Carlo Localization (AMCL) algorithm within the Robot Operating System (ROS). AMCL employs LIDAR and odometry data to probabilistically estimate the robot's position and orientation. By training with synchronized data from the RGB-D camera and AMCL outputs, the project will explore the viability of visual cues for localization. This approach could offer an innovative alternative to conventional localization algorithms, potentially enhancing robustness and decreasing the dependency on wheel encoders and LIDAR [1]–[4].

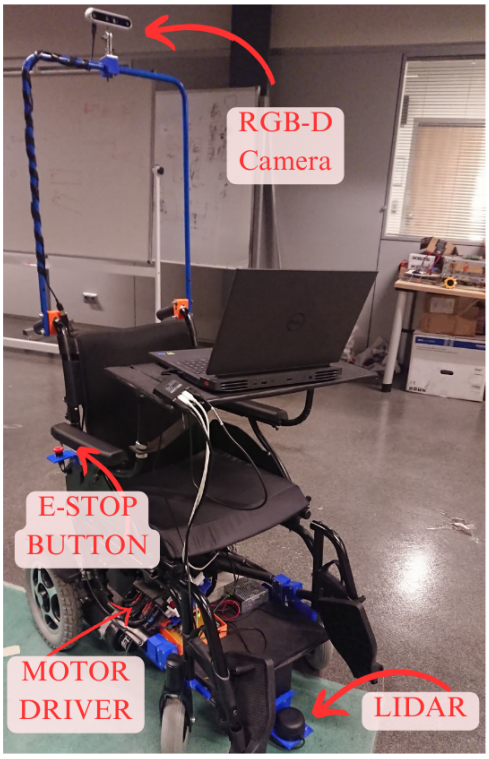

Fig. 1. The autonomous wheelchair.

II. Problem Definition

From a robotics standpoint, the core problem this project aims to address is the adaptation of CNN-based visual processing to perform accurate wheelchair localization, a task traditionally managed by sensor fusion techniques involving LIDAR and odometry. The initiative seeks to demonstrate that a CNN can reliably infer spatial information from RGB-D imagery to determine the wheelchair's position and orientation within the dynamic environment of a building's corridor. This approach necessitates an understanding of the spatial dimensions within robotics but also the ability to interpret complex visual cues against the variability introduced by real-world interactions.

In parallel, the project confronts several challenges from a

deep learning perspective. The first is developing a robust data acquisition and preprocessing pipeline capable of delivering high-quality, annotated visual data for training the CNN. This includes solving image distortion problems, varying lighting conditions, and partial occlusions that can mislead the learning process. Furthermore, selecting a suitable CNN architecture and its hyperparameter optimization are pivotal to ensure that the network can generalize well from the training data to unseen environments. The model must be lightweight enough to facilitate real-time inference yet sophisticated enough to capture the subtleties required for precise localization.

III. Dataset

The dataset essential for this project will be constructed by collecting synchronized RGB-D images and AMCL-generated pose and orientation data as the wheelchair maneuvers through the mapped corridors of the Faculty of Electrical & Electronics Engineering. Special attention will be paid to capturing a diverse range of scenarios that the wheelchair may encounter, including variations in lighting, transient obstacles, and different times of day to simulate real-world conditions. This approach mitigates overfitting and ensures CNN's robust performance across various environmental contexts.

Data integrity will be a focal point, with procedures in place to identify and rectify any imbalances or biases in the dataset. Furthermore, preprocessing techniques such as image normalization, augmentation, and filtering will be applied to enhance data quality, providing CNN with a comprehensive learning foundation. The collection process itself will be methodical, employing a systematic approach to ensure that data is not only extensive but also representative of the operational domain of the autonomous wheelchair. The dataset used for this project is available on GitHub. You can access the dataset at the following link:

https://github.com/econtarli/BLG561E_Final_Project

IV. Methodology

The methodology involves data preprocessing, CNN model selection, and training. The data will be preprocessed to align images with their corresponding AMCL data. The dataset is saved as a CSV file containing image path info and x, y, and yaw values. The x and y values represent the wheelchair's position, while the yaw value gives the orientation angle. The obtained dataset is randomly split into %20 test set and %80 train set.

In the data pre-processing step, the x and y values, which are in meters, are normalized between 0 and 1, and the 'yaw' angle, which is in degrees, is marked with negative values for north and positive values for south. The images are resized to 224x224 to fit the input size of the CNN model. As a data augmentation step, %20 rotation, %20 width/height shift, %20 shear, %20 zoom, and horizontal flip is applied to the images.

A. Custom CNN Model

After some experiments, the optimal CNN architecture is selected. The CNN architecture is selected based on its ability to handle spatial data and learn pose and orientation feature representations. The architecture consists of multiple convolutional layers followed by max pooling and batch normalization layers. At the end of the architecture, dense layers are included to learn the feature representation and dropout layers are added to regularize and prevent overfitting. The final dense layer has three outputs for x, y, and yaw values. While regression is performed for the x and y values, classification is used for yaw value. The architecture of CNN model is shown in Figure3.

Layer (type)	Output Shape	Param #
conv2d_12 (Conv2D)	(None, 222, 222, 32)	896
max_pooling2d_12 (MaxPooling2D)	(None, 111, 111, 32)	0
batch_normalization_12 (BatchNormalization)	(None, 111, 111, 32)	128
conv2d_13 (Conv2D)	(None, 109, 109, 64)	18,496
max_pooling2d_13 (MaxPooling2D)	(None, 54, 54, 64)	0
batch_normalization_13 (BatchNormalization)	(None, 54, 54, 64)	256
conv2d_14 (Conv2D)	(None, 52, 52, 128)	73,856
max_pooling2d_14 (MaxPooling2D)	(None, 26, 26, 128)	0
batch_normalization_14 (BatchNormalization)	(None, 26, 26, 128)	512
conv2d_15 (Conv2D)	(None, 24, 24, 256)	295,168
max_pooling2d_15 (MaxPooling2D)	(None, 12, 12, 256)	0
batch_normalization_15 (BatchNormalization)	(None, 12, 12, 256)	1,024
flatten_3 (Flatten)	(None, 36864)	0
dense_6 (Dense)	(None, 128)	4,718,720
dropout_3 (Dropout)	(None, 128)	0
dense_7 (Dense)	(None, 3)	387

Total params: 5,109,443 (19.49 MB)
Trainable params: 5,108,483 (19.49 MB)
Non-trainable params: 960 (3.75 KB)

Fig. 2. Architecture of the custom CNN model

B. Pre-trained VGG19 Model

To benchmark to a custom CNN model, a pre-trained VGG19 model is used. The VGG model, sequential deep architecture, is fine-tuned by adding top layers to adapt it to the task. The same training techniques, including data augmentation and normalizations, are applied to the VGG model.

C. Training and Evaluation

Both model are compiled using Adam optimizer, chosen for its efficiency and ability to handle sparse gradients. The loss function used is mean square error, appropriate for regression tasks.

Early stopping and model checkpoints are applied to prevent overfitting and save the best model. Early stopping cuts the training process if the validation loss does not improve for 15 epochs, while the model checkpoints save the best model based on the validation loss.

The model's performance is evaluated by displaying the training and validation losses graph over the epochs. Results are also tested on test set. The loss graph can be seen in Figure 4. The prediction is made on 5 random images on the trained model. The predicted values are compared with true values to check the model's accuracy.

Layer (type)	Output Shape	Param #	Connected to
input_layer_1 (InputLayer)	(None, 224, 224, 3)	0	–
block1_conv1 (Conv2D)	(None, 224, 224, 64)	1,792	input_layer_1[0]…
block1_conv2 (Conv2D)	(None, 224, 224, 64)	36,928	block1_conv1[0][…
block1_pool (MaxPooling2D)	(None, 112, 112, 64)	0	block1_conv2[0][…
block2_conv1 (Conv2D)	(None, 112, 112, 128)	73,856	block1_pool[0][0]
block2_conv2 (Conv2D)	(None, 112, 112, 128)	147,584	block2_conv1[0][…
block2_pool (MaxPooling2D)	(None, 56, 56, 128)	0	block2_conv2[0][…
block3_conv1 (Conv2D)	(None, 56, 56, 256)	295,168	block2_pool[0][0]
block3_conv2 (Conv2D)	(None, 56, 56, 256)	590,080	block3_conv1[0][…
block3_conv3 (Conv2D)	(None, 56, 56, 256)	590,080	block3_conv2[0][…
block3_pool (MaxPooling2D)	(None, 28, 28, 256)	0	block3_conv3[0][…
block4_conv1 (Conv2D)	(None, 28, 28, 512)	1,180,160	block3_pool[0][0]
block4_conv2 (Conv2D)	(None, 28, 28, 512)	2,359,808	block4_conv1[0][…
block4_conv3 (Conv2D)	(None, 28, 28, 512)	2,359,808	block4_conv2[0][…
block4_pool (MaxPooling2D)	(None, 14, 14, 512)	0	block4_conv3[0][…
block5_conv1 (Conv2D)	(None, 14, 14, 512)	2,359,808	block4_pool[0][0]
block5_conv2 (Conv2D)	(None, 14, 14, 512)	2,359,808	block5_conv1[0][…
block5_conv3 (Conv2D)	(None, 14, 14, 512)	2,359,808	block5_conv2[0][…
block5_pool (MaxPooling2D)	(None, 7, 7, 512)	0	block5_conv3[0][…
flatten_1 (Flatten)	(None, 25088)	0	block5_pool[0][0]
dense_1 (Dense)	(None, 128)	3,211,392	flatten_1[0][0]
dropout_1 (Dropout)	(None, 128)	0	dense_1[0][0]
xy_output (Dense)	(None, 2)	258	dropout_1[0][0]
yaw_output (Dense)	(None, 1)	129	dropout_1[0][0]

Total params: 24,350,027 (92.89 MB)
Trainable params: 3,211,779 (12.25 MB)
Non-trainable params: 14,714,688 (56.13 MB)

Fig. 3. Architecture of the custom Pretrained VGG model

V. Results

The performance of the custom CNN-based localization system is evaluated using both the custom CNN model and a pre-trained VGG model. The evaluation includes comparing training and validation losses and testing results for predicted and actual wheelchair positions and orientations.

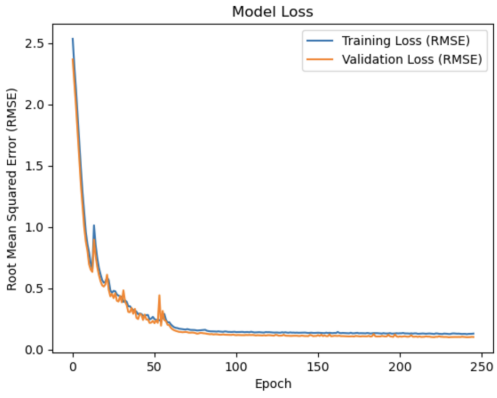

Fig. 4. Training and Validation Losses for CNN model

The above graphs show the training and validation losses for the custom CNN model and pre-trained VGG model. The Root Mean Squared Error (RMSE) was used as the loss metric. For the custom CNN model, the training loss decreased steadily and converged after approximately 200 epochs, indicating good training progress. The validation loss is similar to the training loss, but with slight fluctuations, showing that the model generalizes well to unseen data. On the other hand, the pre-trained VGG model showed a higher validation loss compared to the custom CNN model, with more visible fluctuations. The model might be approaching overfitting. Therefore, the custom CNN model is a more suitable model for this localization task. The custom CNN model demonstrated better performance regarding training/validation losses and test results, as shown in the figures below.

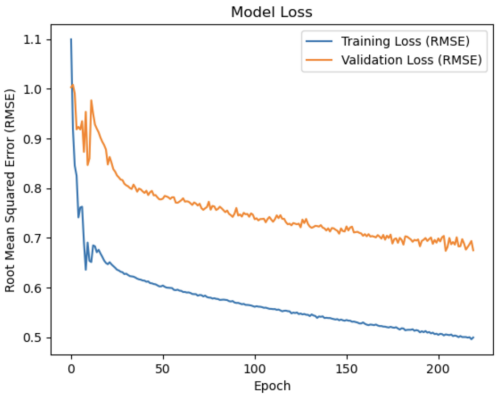

Fig. 5. Training and Validation Losses for Pretrained VGG19 model

Fig. 6. Test results from Custom CNN model.

Fig. 7. Test results from Pretrained VGG model.

REFERENCES

[1] A. Forechi, T. Oliveira-Santos, C. Badue, and A. F. D. Souza, "Visual global localization with a hybrid wnn-cnn approach," in *2018 International Joint Conference on Neural Networks (IJCNN)*, 2018, pp. 1–9.

[2] M. Ballesta, L. Paya, S. Cebollada, O. Reinoso, and F. Murcia, "A cnn regression approach to mobile robot localization using omnidirectional images," *Applied Sciences*, vol. 11, no. 16, p. 7521, 2021.

[3] F. Foroughi, Z. Chen, and J. Wang, "A cnn-based system for mobile robot navigation in indoor environments via visual localization with a small dataset," *World Electric Vehicle Journal*, vol. 12, no. 3, p. 134, 2021.

[4] S. Xu, W. Chou, and H. Dong, "A robust indoor localization system integrating visual localization aided by cnn-based image retrieval with monte carlo localization," *Sensors*, vol. 19, no. 2, p. 249, 2019.