Assistive Parking Systems Knowledge Transfer to End-to-End Deep Learning for Autonomous Parking

Omar Gamal
Chair of Automatic Control Engineering
University of Siegen
D-57076 Siegen, Germany
e-mail: omar.gamal@uni-siegen.de

Mohamed Imran
Chair of Automatic Control Engineering
University of Siegen
D-57076 Siegen, Germany
e-mail: mohamed.pmohamed@student.uni-siegen.de

Hubert Roth
Chair of Automatic Control Engineering
University of Siegen
D-57076 Siegen, Germany
e-mail: hubert.roth@uni-siegen.de

Jürgen Wahrburg
Chair of Automatic Control Engineering
University of Siegen
D-57076 Siegen, Germany
e-mail: wahrburg@zess.uni-siegen.de

***Abstract*—In numerous spots, parking a vehicle is challenging task and requires an experienced driver to maneuver and park the vehicle efficiently. With the advent of Automatic Parking Assist Systems (APAS), drivers can park their vehicles automatically and safely. These systems, however, still require driver intervention and constant attention while parking. The APAS system uses the onboard sensors to perceive the environment to identify the obstacles around and a proper parking space. The system then plans a collision-free trajectory and follows that trajectory to park the vehicle in the designated parking space. This paper presents an intelligent parking system for parking Unmanned Ground Vehicle (UGV) perpendicularly using Convolution Neural Networks (CNNs). To overcome the problem of dataset scarcity and quality APAS system is used to generate training data. The neural network model is trained to mimic the APAS system behavior captured in the generated dataset. The evaluation of the trained CNN model showed that the proposed intelligent parking system is able to park the vehicle perpendicularly with accurate orientation.**

Keywords-automatic assistive parking system; end-to-end learning; perpendicular parking; ROS; path planning; path tracking

I. INTRODUCTION

There is no doubt that parking a vehicle is one of the difficult tasks that require experienced driver and familiarity with vehicle's dimensions. To park the vehicle, the driver should find a suitable spot that fits vehicle's dimensions and maneuver the vehicle to safely park it in the designated space without harming his vehicle or other vehicles around. Therefore many research has been conducted to automate the parking process however most of the developed systems require driver constant attention and intervention [1]. Moreover, most of these systems don't consider all parking techniques. Automatic Parking Assist System (APAS) has been adopted in various commercial car models. The system consists of four main modules; namely Environment recognition, path planning, path tracking and actuation system [2].

Parking has been addressed by researchers and many solutions have been proposed. Nowadays, Artificial intelligence (AI) based methods have drawn much attention and interest from researchers such as fuzzy logic, genetic algorithms, neural networks and hybrid intelligent techniques. AI is considered to be one of the key enablers of autonomous parking system to overcome the limitations and the problems we have in traditional methods. The authors in [3] used a self-organizing fuzzy controller. Razinkova A., Cho H. C. and Jeon H. T in [4] introduced Fuzzy-logic based trajectory generation algorithm for parallel parking. The authors of [5] proposed a hybrid intelligent control scheme for automatic parallel parking that consists of three different AI approaches; namely genetic algorithm, Petri net, and fuzzy logic. Azadi, S. H., Nedamani, H. R. and Kazemi in [6] demonstrated the feasibility of automatically parking an Articulated Vehicle using Adaptive-Network-based Fuzzy Inference System (ANFIS). The authors of [7] introduced a hybrid intelligent technique for autonomous parallel parking using neuro-fuzzy controller. Moon, J., Bae I. and Kim, S. in [8] approached the problem by using twin artificial neural network architecture to mimic the human driver behavior. The authors of [9] adopted Jordan-net neural network model to autonomously park and pull out the vehicle from the parking space. Other research involving the usage of artificial intelligence techniques in autonomous parking are in [10]-[14].

This paper presents an intelligent parking system that mimics the behavior of the APAS system to park the vehicle perpendicularly in the designated parking space. The proposed approach is based on End-to-End deep learning where the network learns by itself the relevant features that map the input image to the corresponding responses, i.e.

978-1-7281-5739-9/20/$31.00 ©2020 IEEE

velocity and steering angle. The frozen inference graph is deployed to verify the effectiveness of the system.

II. METHODOLOGY

In perpendicular parking, the vehicles are parked next to each other perpendicularly to the wall or anything else. The strategy of perpendicular parking is considered to be the easiest between all other parking techniques, e.g. parallel, angle, reverse parking etc. Nevertheless, parking the car efficiently with this technique can be challenging for inexperienced drivers especially when the free space lies between two other cars. In this parking technique, the driver should keep enough distance between his car and the row of other parked cars, turning the wheel all the way on one side and proceeds to the parking space. Once the car reaches halfway into the parking space the driver should straighten the wheels and stop when reaches the proper spot.

In this work, we will tackle the problem of perpendicular parking by using Deep Convolutional Neural Networks (CNNs). End-to-End learning has proved itself in various applications and most of the research conducted nowadays is concerned with the application of deep learning to enhance the accuracy of conventional systems or even replacing them. However, training CNNs from scratch or fine-tuning pre-trained models requires a huge amount of data which depends mainly on the task/problem in hand and how deep the network architecture is. To our knowledge, there is no dataset available for autonomous parking. Most of the datasets available are related to the identification of emergency, safe and secure parking locations, occupancy of public parking areas, parking signs, parking space detection etc. Collecting the data ourselves is expensive and requires experienced and patient driver who drives the car hundred or even thousands of times collecting images and steering angles that translates to efficient parking behavior. On the other hand, repeating this process hundreds or even thousands of times might affect the quality of collected data.

One way to overcome these problems is to build an Automatic Parking Assist System (APAS) to automatically park the vehicle in the parking space while simultaneously collecting the dataset. The experience of the APAS system is reflected in the collected dataset which can be easily transferred to CNN model. The trained model can then autonomously park the vehicle perpendicularly and even adapt itself to unseen scenarios. In order to collect the dataset and investigate the performance of the trained model, an Unmanned Ground Vehicle model (UGV) will be used.

III. SYSTEM SETUP

Fig. 1 and 2 illustrate the UGV model and the hardware architecture of the system. The UGV model is based on the Ackermann steering mechanism and equipped with NVIDIA Jetson TX2 developer kit, Front Camera, Arduino 2560, DC Motor, Servo Motor, RC Controller Module, etc.

The system incorporates two control levels; namely high-level and Low-level control. The low-level, i.e. Arduino board is used for sensor data collection and sending to high level controller as well as motor drivers control based on the commands from the high-level controller. The high-level control is where all high processing tasks are performed such as dataset collection, APAS and running inference model. The front camera is used for dataset collection and inference. It is placed on top of the vehicle exactly at the center axes. The camera is made to focus on the ground by tilting it to an angle of 70° and focusing on the ground. The vehicle's motors, as well as on-board hardware, are supplied from an external power source.

Figure 1. Unmanned ground vehicle.

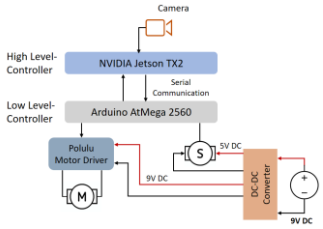

Figure 2. System architecture.

IV. ASSISTIVE PARKING SYSTEM

The APS system generally consists of four main modules; namely Environment perception, Path planning, Path tracking and Actuation system [2]. In environment perception, the target parking space, vehicle position and obstacles are identified and given as an input to path planning module which generates a path from the vehicle position to the parking space [6]. Finally, the path tracking module controls the vehicle's speed and steering angle to follow the generated path and park the vehicle in the designated space efficiently.

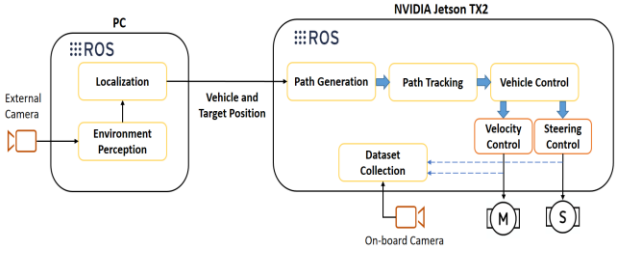

Figure 3. Proposed APAS system architecture.

The APS system architecture built for our use case is shown in Fig. 3. We used a Personal Computer (PC) and an external camera for UGV and parking space localization using ROS Framework. The target and vehicle pose are then packed in ROS geometry pose stamped messages and sent to

the path generation and tracking modules for further processing.

The vehicle's onboard camera and dataset collection program are used to collect the incoming frames along with the velocity and steering angles applied to the vehicle's motors. Fig. 4 illustrates the test environment built for dataset collection and inference. We assumed that we have one free parking space and there are no obstacles present around the vehicle.

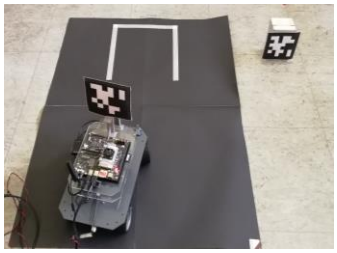

Figure 4. Test environment.

A. UGV and Parking Space Localization

The position of UGV and parking space are identified using the ROS wrapper of AprilTag 3 system. The apriltag_ros package is capable of detecting multiple tags in a given image frame or video stream coming from a camera. The pose of all tags relative to the camera frame is maintained by the tf ROS package. The tf holds the relationship between all coordinate frames in the system over time and allows us to get the pose, i.e. position and orientation of one frame relative to another.

Two tags are placed in the test environment, see Fig. 5. The first Tag is placed on top of the UGV to get its position relative to the camera frame. The second Tag is placed at an offset of -75 cm in X-axis from the center of the parking area. The Y-axis which represents the height does not play a role in path planning and tracking algorithms and thus is kept without alteration. The pose of the UGV and parking space are obtained using a tf transform listener and sent to the path planning and tracking modules.

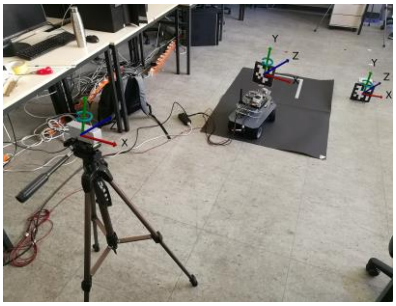

Figure 5. Vehicle and parking space localization using Aprilags.

B. Path Generation

A B´ezier curve-based path planning algorithm is used to generate a path from the vehicle position to the target parking space. It is smother and computationally inexpensive compared to other geometric curves based path planning algorithms like cubic splines, polynomials, etc. The algorithm uses approximation and interpolation methods for generation of paths/curves [15]-[16]. Initial and final point's works on interpolation methods while the in-between point's works on the approximation method. The smoothness of the curve can be varied by adjusting the in-between control points. The curve will connect the initial and final point but it will not pass through the in-between points due to the approximation. A Bézier curve of n-control points is defined as

$$P(\lambda) = \sum_{i=0}^{n} B_i^n(\lambda) P_i. \quad (1)$$

Where Pi is the control points of the ith vectors. $B_i^n(\lambda)$ represents Bernstein polynomial (1). Where n represents the order of the curve and λ normalized time.

$$B_i^n(\lambda) = \binom{n}{i} \lambda^i (1-\lambda)^{n-i} \quad (2)$$

C. Path Tracking

A geometric path tracking algorithm is used to autonomously navigate the vehicle over the generated path. The algorithm computes the required control commands, i.e. vehicle velocity and steering angles that minimize the difference between the generated and actual path. The pure pursuit algorithm is used owing to its simplicity and robustness against disturbances [17]-[18]. The algorithm computes the curve that connects the origin of the vehicle's coordinate system with the next waypoint on the path at a distance L_{wf}, see Fig. 6. The origin of the vehicle's coordinate system is preferable to be placed at the vehicle's rear axle or shifted with an offset ϵ.

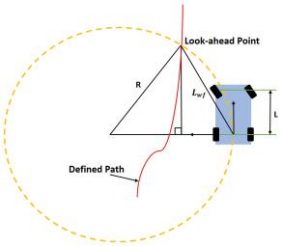

Figure 6. Pure Pursuit path tracking algorithm.

The turning radius can be calculated as:

$$\frac{L_{fw}}{2 \sin \eta} = R. \quad (3)$$

Where η is the lookahead heading. The vehicle's steering angle δ can be calculated using vehicle kinematic model as:

$$\delta(t) = \tan^{-1}\left(\frac{2.L.\sin(\eta(t))}{L_{fw}}\right). \quad (4)$$

D. Dataset Collection and Preparation

The UGV is autonomously driven using the APS system to park it in the designated parking space while simultaneously collecting the dataset.

TABLE I. DATASET CLASS LABELS

Class	Steering Angle	Velocity
050F100	50	100%
060F100	60	100%
070F100	70	100%
080F100	80	100%
090F100	90	100%
100F100	100	100%
110F100	110	100%
120F100	120	100%
130F100	130	100%
090S000	90	0%

To create more variations in the dataset the UGV is made to start from different points in the test environment. The captured images are stored in different folders which correspond to the different class labels. The defined class labels are shown in the above table where F means forward and S stop. Fig. 7 shows an example of the collected dataset and Fig. 8 depicts the number of images collected per class.

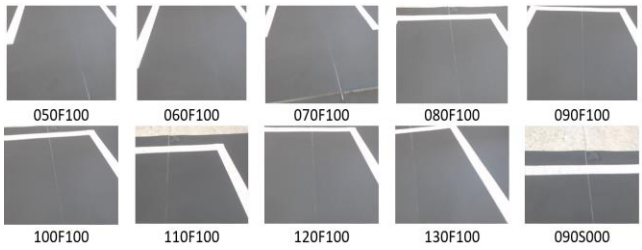

Figure 7. Example of generated dataset.

Figure 8. Generated dataset.

V. CNN MODEL ARCHITECTURE DESIGN AND TRAINING

The CNN model architecture consists of 12 layers. Three Convolutional layers, one Maxpooling layer, four dropout layers, one Image input layer, one flatten layer and two fully connected layers, see Fig. 9.

The network takes an input image of size (160X213X 3). The next layer is conv2d layer which consists of 32 kernels of size 11 X 11, strides of 7 X 7, valid padding and RelU activation. The output dimension of the layer is 22 X 29 X 32. It is connected to the dropout layer with a rate of 0.2. The next layer is Maxpooling layer with a pool size of 3 X 3, strides of 2 X 2, and valid padding. The output of the layer is 10 X 14 X 32. The next layer is conv2d layer which consists of 16 kernels of size 9 X 9, strides of 1 X 1, same padding, and ELU activation. The output dimension of the layer is 10X14 X16 which is connected to dropout layer with a rate of0.1. The next layer is conv2d layer which consists of 32 kernels of size 6 X 6, strides of 2 X 2, same padding, and Tanh activation. The output dimension of the layer is 5 X 7 X 32 which is connected to dropout layer with a rate of 0.1. The next layer is flatten layer that converts the input into a vector of dimension 1120 X 1, which is given as an input to a fully connected layer, consisting of 50 units with sigmoid activation. The next layer is the Dropout layer with a rate of 0.1. The final layer is fully connected layer with 10 units and soft-max activation.

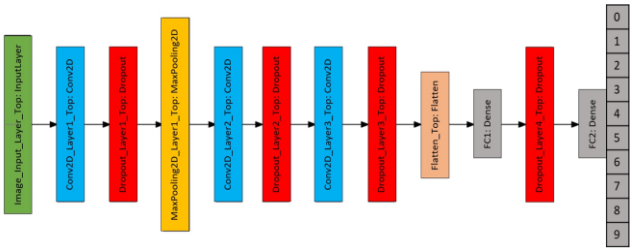

Figure 9. CNN's model architecture.

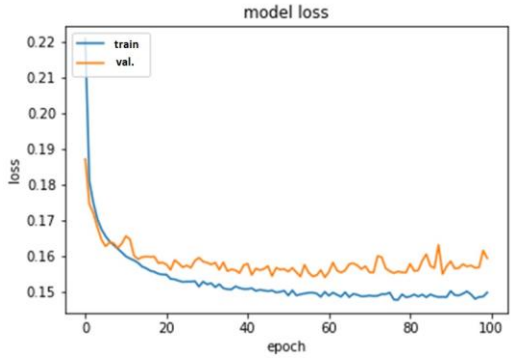

Figure 10. Training and validation Loss graphs.

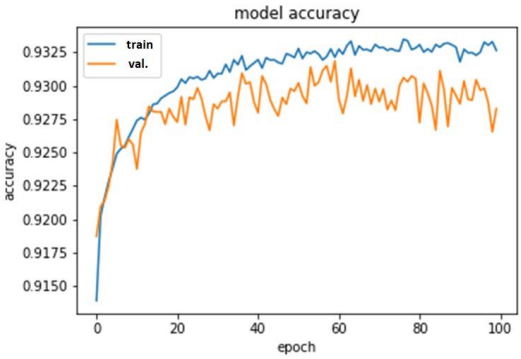

Figure 11. Training and validation accuracy graphs.

To train the network, Adam optimizer and binary cross-entropy loss functions have been used. The batch size used for training the network is 32 and the learning rate is 0.001. The training is started with 100 epochs and the weights of the model at epoch 59 are selected because after that epoch the model overfits. The corresponding loss and accuracy values are 0.153973 and accuracy of 93.18% respectively, see Fig. 10 and 11.

VI. Testing and Evaluation of the Trained CNN Model

To test and evaluate the trained CNN model the frozen inference graph is obtained and deployed on the NVIDIA Jetson TX2 board. The model takes a live video stream from the on-board camera and for each incoming frame, it predicts the required speed and steering angle which is then sent to Arduino to drive the vehicle towards the parking space, see Fig. 12. A multiple test runs have been made to evaluate the model performance. Fig. 14 depicts car behavior during one of the test runs. The UGV was able to reach the parking space with good orientation regardless of the starting initial position.

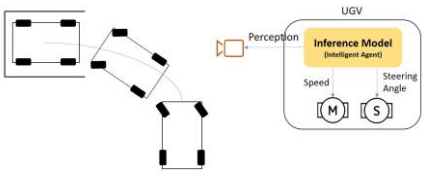

Figure 12. Deployment of frozen inference graph.

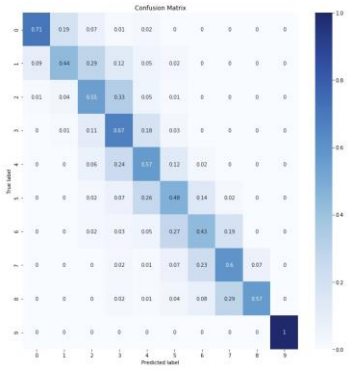

Figure 13. Confusion matrix of test dataset (3278 images).

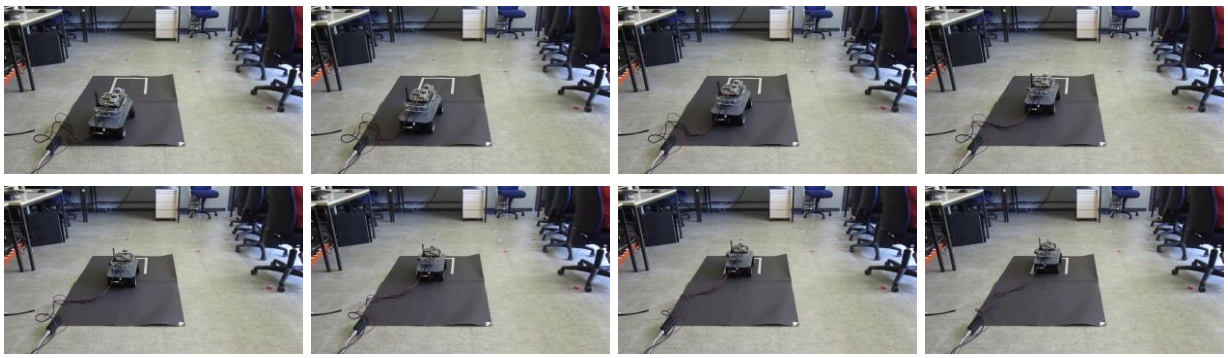

Figure 14. Example of a test run.

To evaluate the model performance further the confusion matrix is generated. Fig. 13 illustrates the confusion matrix where the diagonal elements denote the true quantity of each class. As can be seen, a major confusion occurred between each class and its direct neighbor's classes. This, however, is due to the small steering angle (10°) between one class and another.

VII. Conclusions

In this paper, the knowledge of the APS system has been successfully transferred to CNN model. The trained model was able to mimic the APS system behavior and park the vehicle perpendicularly with accurate orientation. Using the APS system as a tutor has proven to be practical and reliable approach to overcome the problem of having experienced driver during dataset collection and low-quality dataset that doesn't capture the best practice.

The proposed approach tackles the problem of perpendicular parking and the feasibility of achieving autonomous parking using CNN. The approach can be applied to self-driving cars with the same localization system or it can be replaced with GPS and high-definition (HD) maps.

To further our research we are planning to investigate other parking techniques as well as parking in the presence of static and dynamic obstacles, e.g. Pedestrians, other vehicles, etc.

References

[1] X. Du and K. K. Tan, "Autonomous reverse parking system based on robust path generation and improved sliding mode control," pp. 1225-1237, 2014.

[2] Y. Song and C. Liao, "Analysis and review of state-of-the-art automatic parking assist system," pp. 1-6, 2016.

[3] S. J. Huang and G. Y. Lin, "Parallel auto-parking of a model vehicle using a self-organizing fuzzy controller," pp. 997-1012, 2010.

[4] A. Razinkova, H. C. Cho, and H. T. Jeon, "An intelligent auto parking system for vehicles," pp. 226-231, 2012.

[5] C. K. Lee, C. L. Lin, and B. M. Shiu, "Autonomous vehicle parking using hybrid artificial intelligent approach," pp. 319-343, 2009.

[6] S. H. Azadi, H. R. Nedamani, and R. Kazemi, "Automatic Parking of an Articulated Vehicle Using ANFIS," pp. 93-104, 2013.

[7] Z. L. Wang, C. H. Yang, and T. Y. Guo, “The design of an autonomous parallel parking neuro-fuzzy controller for a car-like mobile robot,” pp. 2593-2599, 2010.

[8] J. Moon, I. Bae, and S. Kim, “Automatic Parking Controller with a Twin Artificial Neural Network Architecture,” 2019.

[9] M. R. Heinen, F. S. Osório, F. J. Heinen, and C. Kelber, “Autonomous vehicle parking and pull out using artificial neural networks,” 2006.

[10] A. Kuefler, J. Morton, T. Wheeler, and M. Kochenderfer, “Imitating driver behavior with generative adversarial networks,” pp. 204-211, 2017.

[11] R. J. Oentaryo and M. Pasquier, “Self-trained automated parking system,” pp. 1005-1010, 2004.

[12] P. Zhang, L. Xiong, Z. Yu, P. Fang, S. Yan, J. Yao, and Y. Zhou, “Reinforcement Learning-Based End-to-End Parking for Automatic Parking System,” 2019.

[13] Y. W. Ryu, S. Y. Oh, and S. Y. Kim, “Robust automatic parking without odometry using an evolutionary fuzzy logic controller,” pp. 434-443, 2008.

[14] Singh Prabhdeep and Bajaj Ankita, "Accident warning system based on vehicular ad-hoc network (vanet): a review," International Journal of Electrical and Electronic Engineering & Telecommunications, Vol. 5, No. 4, pp. 53-57, October 2016.

[15] A. Sakai, D. Ingram, Dinius J., Chawla K., Raffin A., and Paques A., “PythonRobotics: a Python code collection of robotics algorithms,” 2018.

[16] Zhou F., Song B., and Tian G., “B´ezier Curve Based Smooth Path Planning for Mobile Robot,” pp. 2441-2450, 2011.

[17] Andersen H., Z. J. Chong, Y. H. Eng, S. Pendleton, and M. H. Ang, “Geometric path tracking algorithm for autonomous driving in pedestrian environment,” pp. 1669-1674, 2016.

[18] R. C. Conlter, “Implementation of the Pure Pursuit Path'hcking Algorithm,” 1992.