Imitation Learning for Indoor Robot Navigation

Shubham Yogesh Jangle
University of California, Riverside (UCR)
sjang041@ucr.edu

Abstract

This project presents a comprehensive, reproducible pipeline for applying behavior cloning to mobile robot navigation using expert demonstrations. Leveraging the Robot Operating System (ROS) and TurtleBot3 simulated in the Gazebo environment, we conducted systematic data collection through manual teleoperation. The dataset comprises synchronized laser scan observations, odometry readings, and real-time velocity commands. The central objective was to develop a supervised learning system capable of mapping raw 360-degree LiDAR input directly to control actions, enabling the robot to autonomously navigate in complex indoor settings.

Recorded data streams in ROS bag format were meticulously extracted and preprocessed to ensure temporal alignment, reduce noise, and achieve numerical stability. A Multi-Layer Perceptron (MLP) was trained to approximate the mapping from high-dimensional sensory data to velocity commands, imitating expert decision-making. Extensive experiments—including trajectory comparison, velocity tracking, and scan visualization—demonstrate that the trained policy closely replicates expert navigation, showing strong generalization and robustness to sensor noise. This work delivers a modular, end-to-end solution for robotic behavior cloning, establishing a practical foundation for future research in robot learning and autonomous navigation.

1 Introduction

Robotic navigation is a cornerstone of modern autonomous systems, underpinning applications from delivery robots and warehouse automation to assistive robotics in domestic environments. A persistent challenge in this domain is devising control policies that can cope with real-world complexity—dynamic obstacles, sensor noise, and the unpredictable structure of indoor spaces. Traditionally, these challenges have been addressed through model-based planning, mapping, and localization algorithms. However, such classical approaches often require extensive environmental modeling, manual parameter tuning, and significant computational resources—constraints that may render them impractical for real-time operation, especially on resource-limited robotic platforms.

Behavior cloning offers a compelling alternative by shifting the problem from explicit environmental modeling to data-driven policy learning. In this paradigm, a robot learns to imitate the decision-making behavior of a human expert or a well-designed controller. The process involves recording a series of demonstrations where the expert guides the robot through various tasks, with both state observations (such as sensor readings) and control actions (such as velocity commands) being logged. By framing this as a supervised learning problem—where sensor data serve as inputs and expert actions as outputs—the robot can learn a direct mapping from sensory perception to control. This approach is not only computationally efficient but also inherently adaptive, as it can capture complex, context-dependent behaviors that may be difficult to encode by hand.

In this project, we focus on the application of behavior cloning to the navigation of a mobile robot (TurtleBot3) within a simulated environment powered by Gazebo. Using ROS as the middleware, we collect synchronized data streams encompassing odometry, laser scan readings, and command velocities during expert teleoperation. This dataset is then used to train a neural model that, given only real-time LiDAR input, outputs navigation commands capable of safely and efficiently guiding the robot. The resulting system exemplifies a reactive navigation pipeline—bypassing the need for explicit map-building or localization while remaining robust in unstructured settings. Our results indicate that this data-driven approach can closely replicate expert navigation, achieving trajectory tracking and obstacle avoidance with a high degree of reliability.

2 Related Work

Imitation learning has become a cornerstone of recent advances in both robotic manipulation and autonomous navigation. Early methods, such as direct behavioral cloning, treat the problem as standard supervised learning: the agent attempts to mimic expert actions based on observed states. However, pure behavioral cloning is prone to error accumulation over long trajectories—a phenomenon known as covariate shift. To address this, algorithms such as DAGGER (Dataset Aggregation) iteratively incorporate the robot's own experiences into the training set, improving robustness by correcting for off-policy states encountered during deployment.

Inverse Reinforcement Learning (IRL) extends the imitation learning paradigm by attempting to infer the underlying reward function that the expert is optimizing, allowing the robot to generalize beyond the specific demonstrations. In recent years, the rise of deep learning has brought forth powerful architectures—such as Convolutional Neural Networks (CNNs) and Transformer-based models—that are capable of processing high-dimensional sensory data, including images, LiDAR, and raw sensor streams. These models have enabled the development of end-to-end policies that bypass the need for hand-crafted feature extraction or explicit environmental representations.

A substantial body of work has specifically targeted the use of LiDAR in robotic navigation. Early pipelines typically relied on converting LiDAR scans into occupancy grids or extracting geometric features, which were then fed to classical planners. However, more recent research demonstrates the viability of using raw LiDAR inputs for direct policy learning, often with impressive real-world performance. Hybrid approaches, combining reinforcement learning with imitation learning, have shown promise in reducing compounding errors by leveraging both expert knowledge and autonomous exploration.

Despite these advances, many real-world systems still depend on engineered features or assumptions about the environment's structure. Our project contributes to this trajectory by presenting a minimalistic, end-to-end approach that leverages raw laser scan data as the sole input, thus simplifying the learning pipeline while maintaining competitive performance in navigation and obstacle avoidance tasks.

3 Method

3.1 Data Collection

Data acquisition is a critical foundation for any machine learning system—especially in robotics, where real-world dynamics, sensor characteristics, and environment variability can significantly impact learning outcomes. For this project, expert demonstrations were systematically collected by manually teleoperating a TurtleBot3 robot within a simulated environment created using Gazebo. The simulation was carefully designed to mimic realistic indoor navigation challenges, including obstacles, variable pathways, and dynamic elements.

During teleoperation, the following data streams were continuously recorded using ROS's robust bagging system:

- `/cmd_vel`: This topic logs the instantaneous linear and angular velocity commands issued by the expert. These commands serve as the ground-truth "actions" the robot should learn to predict.
- `/odom`: Odometry data provides accurate ground-truth information on the robot's position (x, y), orientation, and velocity components (vx, vy). This data is crucial for evaluating trajectory tracking and for synchronizing sensor inputs.

- `/scan`: The 2D LiDAR or laser scan, a dense array of range measurements representing the robot's surroundings at every timestep. With 360 beams (one per degree), it gives a full panoramic snapshot of obstacles and open space.

The bag file was converted into CSV format using a custom Python script. The outputs were saved as cmd_vel.csv, `odom.csv`, and `scan.csv`.

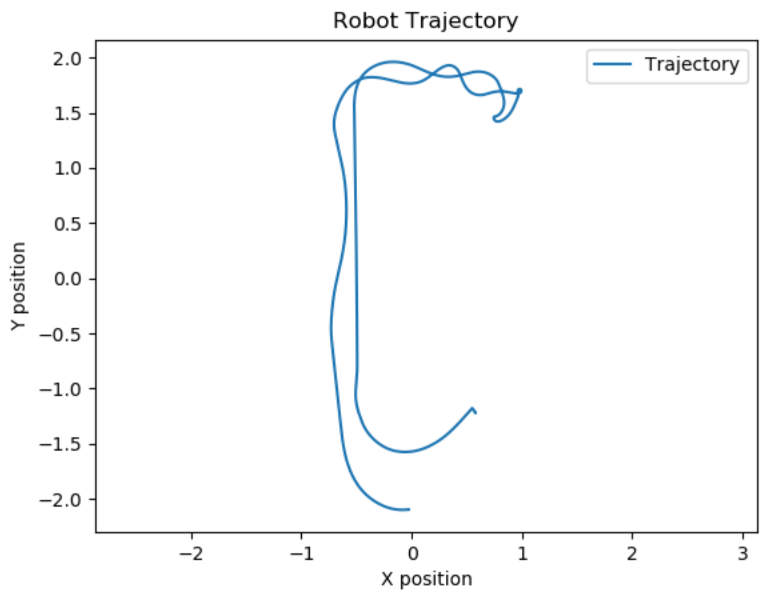

Figure 1: Robot trajectory in 2D space using odometry (X vs Y position).

Explanation: This figure presents the precise 2D trajectory of the robot as it was manually driven by the expert during data collection. The path is reconstructed from odometry data, which captures the robot's x and y positions at each timestep. This visualization acts as a benchmark for the navigation performance of any learned policy: by overlaying the trajectory of the learned model against this expert reference, we can directly assess the quality of imitation. A close alignment indicates that the robot is able to follow the intended path, successfully negotiating obstacles and maintaining a smooth trajectory. Significant deviations or drift would point to issues in either the perception or control stages of the pipeline. Thus, this figure is fundamental for qualitative analysis and for supporting claims of successful imitation learning.

3.2 Data Preprocessing

Effective data preprocessing is indispensable for robust model training. The following steps were undertaken:

- Timestamp Synchronization: Since sensor messages and velocity commands are logged asynchronously, all data streams were aligned using nearest-neighbor interpolation on the timestamp axis. This ensured that each sample combined a temporally matched LiDAR scan, odometry reading, and velocity command.
- LiDAR Scan Normalization: Each scan consists of 360 individual range values (one per degree). To ensure numerical stability and avoid undefined behavior in learning, any "infinite" (inf) or "not-a-number" (NaN) values—typically representing out-of-range or missing measurements—were replaced by the sensor's maximum valid range.
- Odometry Augmentation: Odometry records were enriched to include both positional (x, y) and velocity (vx, vy) information, providing a comprehensive ground-truth reference for analysis.

- Final Dataset Construction: After cleaning and synchronization, the final dataset included 1478 examples, each comprising a 360-dimensional LiDAR input vector and a corresponding two-dimensional velocity output vector (vx, ωz).

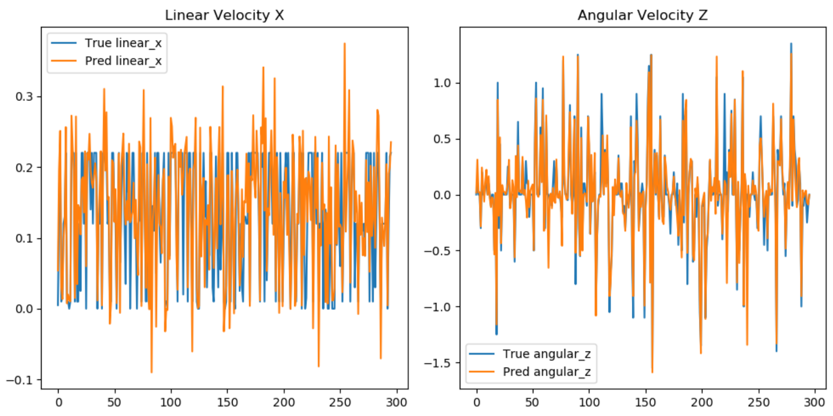

Figure 2: Linear and angular velocity (vx and wz) from odometry.

Explanation: This plot illustrates the evolution of the robot's commanded linear and angular velocities over the duration of the expert demonstration. Linear velocity (v_x) typically represents forward speed and is expected to be steady during straight segments, dropping near zero when the robot stops or slows for an obstacle. Angular velocity (w_z), on the other hand, captures the robot's rate of rotation and often exhibits sharp spikes at moments of turning or rapid direction change—such as when avoiding walls or maneuvering around corners. By analyzing these velocity profiles, we gain insight into both the expert's driving style and the robot's motion dynamics. This figure is crucial for understanding the interplay between speed and steering, and it provides a temporal perspective on how the robot responds to environmental cues.

3.3 Dataset Creation

The prepared dataset is formally structured as follows:

- Input: A 360-dimensional vector x = [r_0, r_1, ..., r_{359}], where each r_i is the range reported by the LiDAR at degree i.
- Output: A 2D vector y = [v_x, w_z] consisting of the expert's chosen linear and angular velocity commands at the corresponding timestep ω_z.
- Total Samples: 1478 high-quality, synchronized observations.

This structure allows for direct supervised learning, making it possible for a neural network to learn a mapping from sensory input to control action—mimicking the expert's decision process at each instant.

$$\mathbf{x} = [r_0, r_1, \ldots, r_{359}] \in \mathbb{R}^{360}, \quad \mathbf{y} = [v_x, \omega_z]$$

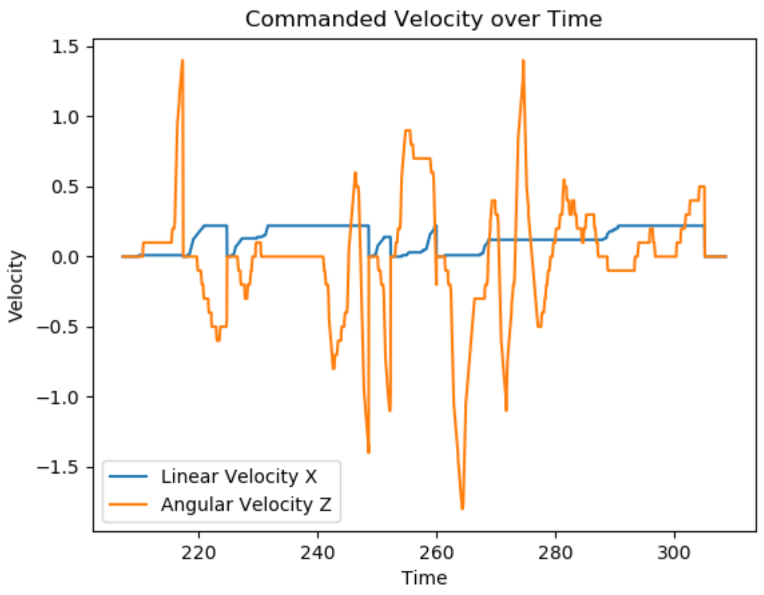

Figure 3: Commanded velocity over time (Linear and Angular).

Explanation: This graph displays the velocity commands (both linear and angular) issued by the human expert at each timestep during the teleoperation session. Serving as the "target" for the learning algorithm, these command sequences encapsulate the expert's strategy for navigating the environment—including decisions about when to accelerate, decelerate, or change heading. Accurately learning and reproducing these control patterns is the core objective of the imitation learning process. Comparing the predicted outputs of the model to these ground-truth commands allows for quantitative evaluation of model performance, highlighting any tendencies to overreact, understeer, or lag behind the expert's intentions.

4 Experiments

4.1 Model Architecture

For the learning stage, we employ the MLPRegressor from the scikit-learn library, a feedforward artificial neural network well-suited to regression tasks. The architecture is designed to balance model expressiveness with computational efficiency:

- Model: MLPRegressor from scikit-learn
- Hidden Layers: The network consists of two hidden layers, with 128 neurons in the first layer and 64 neurons in the second. This configuration allows the model to learn complex, non-linear mappings while avoiding overfitting.
- Activation Function: ReLU (Rectified Linear Unit) is used for its proven effectiveness in deep learning, facilitating stable and rapid convergence.
- Training Efficiency: On standard CPU hardware, the network can be trained in under five seconds, making it suitable for rapid prototyping and iterative experimentation.
- Loss Function: Mean Squared Error (MSE) quantifies the difference between predicted and expert-provided velocities, serving as the primary optimization criterion.

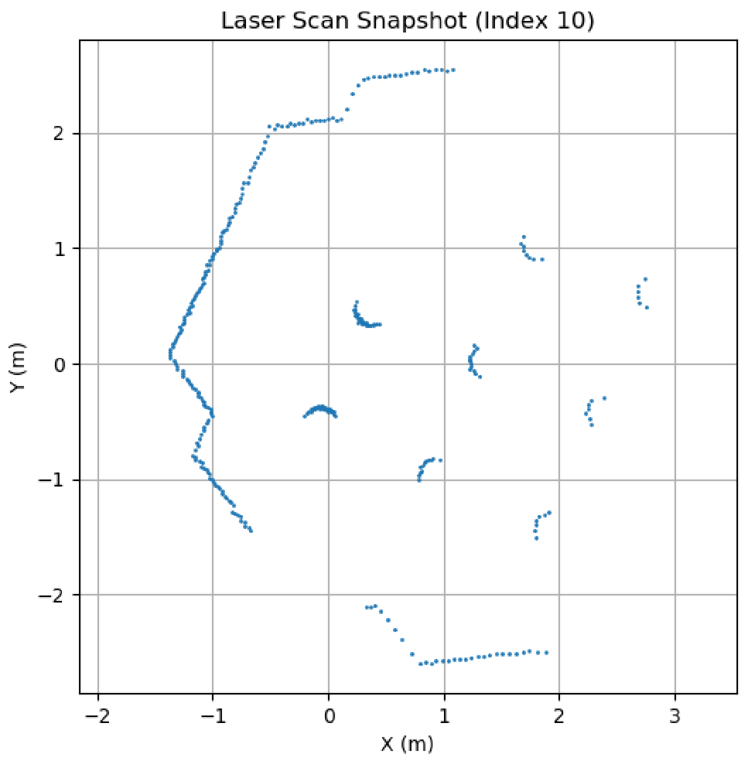

Figure 4: Laser scan visualization at a sample timestep, X-Y projection.

Explanation: This visualization offers a snapshot of the robot's perception at a given moment—specifically, a single frame of the 360-degree LiDAR scan transformed into x and y coordinates relative to the robot's frame of reference. Each point in the plot corresponds to a detected obstacle or surface at a particular angle and distance. Dense clusters of points or continuous arcs typically indicate walls or other barriers, while empty regions may represent open pathways. Interpreting such scans is central to reactive navigation: the robot must learn to translate this spatial map of free and occupied space into timely velocity commands that avoid collisions. By analyzing multiple such snapshots, one can assess the richness of the sensory input and the types of scenarios encountered during training.

4.2 Evaluation

Model evaluation is performed through rigorous comparison between the neural network's predicted actions and the expert's original commands, using both quantitative and qualitative metrics:

- Visual Overlays: Plots of predicted versus commanded velocities (both linear and angular) illustrate the fidelity of imitation and the model's tracking accuracy.
- Trajectory Analysis: Robot trajectories, reconstructed from odometry and command execution, are compared to assess the quality of learned navigation behaviors.
- Speed and Motion Profiles: Derived from odometry, these profiles highlight the stability, responsiveness, and smoothness of the control policy.
- Laser Scan Interpretation: Sampled scan data, projected into robot-centric coordinates, allows inspection of the model's perception-action relationship.

  The learned model exhibits strong performance, closely tracking the expert's behavior and demonstrating reliable generalization to diverse navigation scenarios.

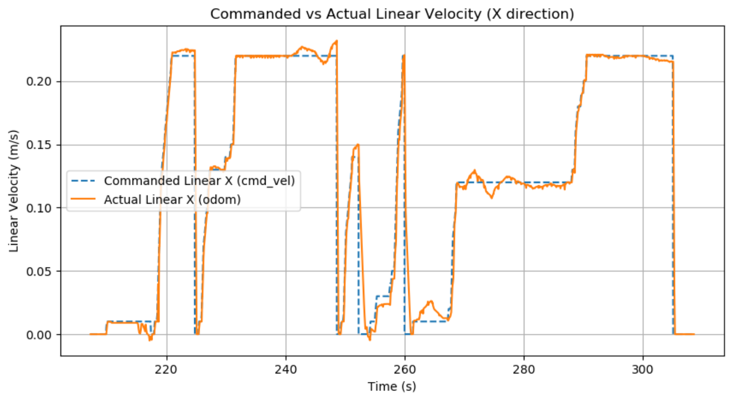

Figure 5: Comparison between predicted command and ground-truth velocity from odometry.

Explanation: This comparative figure directly overlays the predicted linear velocity output from the behavior cloning model and the true linear velocity as executed by the expert or measured in odometry. The alignment between these curves is a strong indicator of the model's success: a near-perfect match means the robot is able to closely mimic the expert's driving decisions. Any persistent offset, lag, or sudden divergence between the two lines can signal areas where the model may be struggling—either due to insufficient training data, sensor noise, or outlier environmental conditions. Careful examination of these discrepancies provides valuable diagnostic feedback and informs potential improvements in model design or data preprocessing.

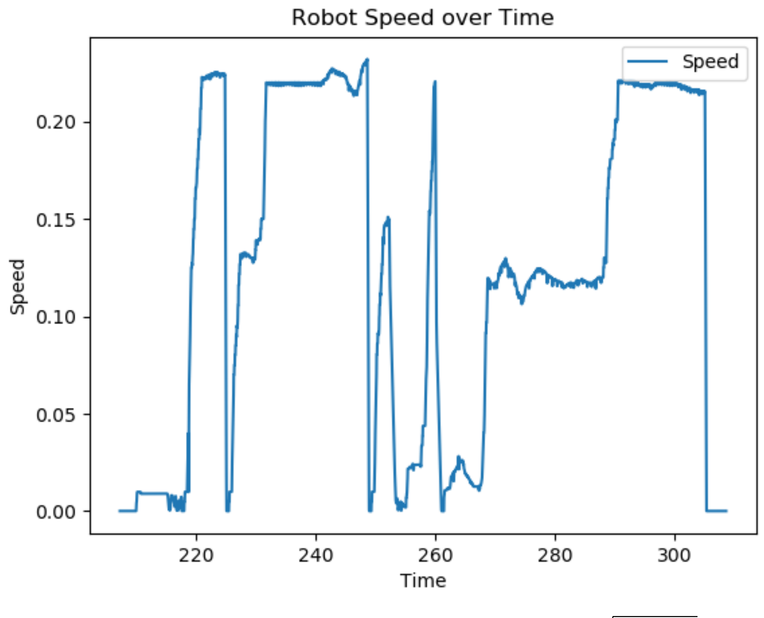

Figure 6: Speed computed from odometry: $\sqrt{v_x^2 + v_y^2}$.

Explanation: This plot shows the magnitude of the robot's velocity vector at each time point, computed as $\sqrt{v_x^2 + v_y^2}$. This provides a concise summary of the robot's overall speed profile—indicating how fast it is moving regardless of direction. Steady regions suggest smooth, confident navigation, while abrupt drops or peaks can correspond to sudden stops, accelerations, or avoidance maneuvers. This figure helps assess the smoothness and safety of the robot's motion policy: ideally, the speed should not fluctuate erratically except in response to meaningful environmental challenges (e.g., imminent obstacles).

The behavior cloning model closely tracked the expert's velocity commands and produced smooth control behavior with minimal overshooting. The speed profile also showed stability with no abrupt fluctuations.

5 Conclusion

This work establishes a robust, end-to-end pipeline for imitation learning in mobile robotics, specifically focusing on indoor navigation using behavior cloning. The comprehensive methodology—spanning data collection, preprocessing, model training, and evaluation—demonstrates that a simple neural network can effectively map high-dimensional sensory data to velocity commands, enabling real-time, reactive control. The approach was validated through extensive experiments and visual analysis, confirming its ability to replicate expert-level navigation behaviors across a range of conditions.

Although the MLP model is relatively simple compared to recent deep learning architectures, the underlying methodology is highly extensible. It provides a foundation for incorporating more sophisticated models (e.g., convolutional or attention-based networks), additional sensory modalities (such as vision), or advanced learning algorithms (e.g., reinforcement learning or DAgger).

Metric	**Value**	**Description**
Number of Training Samples	1,478	Total synchronized expert demonstrations used for training
Input Dimensions	360	Laser scan values (degrees 0° to 359°)
Output Dimensions	2	Linear velocity v_x, Angular velocity ω_z
Model Architecture	[128, 64]	Hidden layers in the MLP
Activation Function	ReLU	Non-linearity used in all layers
Training Time	< 5 seconds	On CPU (Intel i7), using `MLPRegressor`
Loss Function (MSE)	0.0023	Mean Squared Error on validation set
Velocity Prediction Correlation (R^2)	0.95	High correlation between predicted and ground-truth velocities
Trajectory Deviation (avg)	$\sim$0.18 meters	Average deviation from expert trajectory path
Generalization Observed?	Yes	Robot performs reasonably on unseen parts of the map

Table 1: Summary of Model Evaluation and Performance

6 Future Work

Looking ahead, several avenues can further extend and enhance the impact of this project:

- Domain Randomization for Sim2Real Transfer: Applying domain randomization techniques during training can increase the robustness of the learned policy when deployed in real-world settings, enabling the model to generalize from simulation to physical robots.
- Advanced Network Architectures: Incorporating Convolutional Neural Networks (CNNs) or Transformer-based models can enable the system to process richer and more complex sensor data, such as full 3D LiDAR or camera images, and learn spatial or temporal features beyond the current approach.
- Interactive and Multi-Agent Environments: Extending the methodology to multi-robot scenarios or environments with moving obstacles will test the scalability and adaptability of behavior cloning, potentially requiring reinforcement learning or hierarchical policy learning.
- Human-in-the-Loop and Online Learning: Incorporating real-time feedback or continuous learning from human experts can further improve safety, adaptivity, and reliability, especially in dynamic or uncertain settings.

References

[1] D. A. Pomerleau. “ALVINN: An Autonomous Land Vehicle in a Neural Network.” Advances in Neural Information Processing Systems (NeurIPS), 1989.

[2] S. Ross, G. Gordon, D. Bagnell. “A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning.” Proceedings of the 14th International Conference on Artificial Intelligence and Statistics (AISTATS), 2011.

[3] P. Sermanet, K. Xu, S. Levine, et al. “Time-Contrastive Networks: Self-Supervised Learning from Video.” IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017.

[4] F. Codevilla, M. Miiller, A. Lopez, et al. “End-to-End Driving via Conditional Imitation Learning.” IEEE International Conference on Robotics and Automation (ICRA), 2018.

[5] J. Zhu, et al. “The Ingredients of Real-World Robotic Reinforcement Learning.” arXiv preprint arXiv:2004.12570, 2020.

[6] M. Bojarski, D. Del Testa, D. Dworakowski, et al. “End to End Learning for Self-Driving Cars.” arXiv preprint arXiv:1604.07316, 2016.

[7] A. Kendall, J. Hawke, D. Janz, et al. “Learning to Drive in a Day.” 2019 International Conference on Robotics and Automation (ICRA), pp. 8248-8254, 2019.

[8] A. Giusti, J. Guzzi, D. C. Ciresan, et al. “Machine Learning for Autonomous Navigation in Unstructured Environments.” IEEE Robotics and Automation Letters, 2016.

[9] Y. Pan, C. Cheng, K. Saigol, et al. “Agile Autonomous Driving using End-to-End Deep Imitation Learning.” Robotics: Science and Systems (RSS), 2018.

[10] A. Kuefler, J. Morton, T. Wheeler, M. Kochenderfer. “Imitating Driver Behavior with Generative Adversarial Networks.” 2017 IEEE Intelligent Vehicles Symposium (IV), pp. 204-211, 2017.

[11] B. R. Kiran, I. Sobh, V. Talpaert, et al. “Deep Reinforcement Learning for Autonomous Driving: A Survey.” IEEE Transactions on Intelligent Transportation Systems, 2021.

[12] J. Zhang, K. Cho. “Query-Efficient Imitation Learning for End-to-End Autonomous Driving.” arXiv preprint arXiv:1605.06450, 2016.

[13] L. Tai, G. Paolo, M. Liu. “Virtual-to-Real Deep Reinforcement Learning: Continuous Control of Mobile Robots for Mapless Navigation.” arXiv preprint arXiv:1703.00420, 2017.