RobotDockCenter: A Supervised Learning Approach to Robot Docking Using Monocular Vision

Anonymous Authors[1]

Abstract

Autonomous robot docking enables robots to recharge or continue their tasks without human intervention. Traditional docking solutions rely on multiple sensors and reinforcement learning to achieve precise alignment. In this paper, we propose an alternative approach using monocular vision and basic deep learning techniques to predict docking actions from a single input frame. Our method eliminates the need for additional sensors, policy optimization, or reward functions. We conduct experiments in the Godot game engine to simulate real-world deployment. Our results demonstrate that a supervised learning approach can effectively guide docking maneuvers while maintaining simplicity and practicality.

1. Introduction

Robot docking is a fundamental task in autonomous systems, enabling robots to recharge and continue operations without human intervention. Traditional docking solutions rely on multiple sensors such as LIDAR, ultrasonic sensors, and cameras (Jia et al., 2023). While effective, these approaches increase hardware complexity and cost.

Reinforcement learning has gained popularity in robotic control, but it often requires extensive training, computational resources, reward function optimization, and at times compromises when implementing them in low-cost systems (Deisenroth et al., 2012). These challenges make reinforcement learning impractical for resource-constrained robots that need a way to dock and charge.

To address these limitations, we propose an alternative approach that utilizes monocular vision and supervised learning to achieve reliable docking without additional sensors or reinforcement learning. Our method predicts docking actions from a single input frame, identifying two key targets: one for alignment and the other representing the docking station.

We use the Godot game engine to generate synthetic training data, capturing diverse docking scenarios. A deep learning model, implemented in PyTorch, is trained on these images to make high-frequency predictions. The model is lightweight, allowing real-time inference on low-power hardware, such as a Raspberry Pi (Ameen et al., 2023). A Flask server facilitates communication between the trained model and the robot's virtual control system for integration into a simulated docking environment.

Our results suggest that monocular vision-based docking is a feasible alternative to sensor-heavy and reinforcement learning-based approaches. This method offers a cost-effective and computationally efficient solution for autonomous docking, with potential for real-world deployment.

The paper is structured as follows. After going over related work, we go in depth regarding our methodology, which includes data generation, data preprocessing, model architecture, training, inference, and Flask server integration. We then present our results and discuss the possible implications of our work. We then conclude our paper with a discussion on the limitations of our work and potential future directions.

2. Related Work

The work of Jia et al. (2023) provides a comprehensive review of existing docking methods. **Keep this section for the end once we have collected all of our sources**

3. Methodology

In this section, we display how effective image preprocessing and our docking procedure collectively lead to a plausible method to train our model without manual labeling, which would be required, for our case, in real-world applications. Our methodology includes data collection, model training, inference, server integration, evaluation, and experimental setup.

3.1. Data Collection

We collect training data using a simulated environment created in a game engine (Hoster et al., 2024). Our game engine of choice is Godot, a free and open source game engine. The environment consists of a robot and a docking station, with

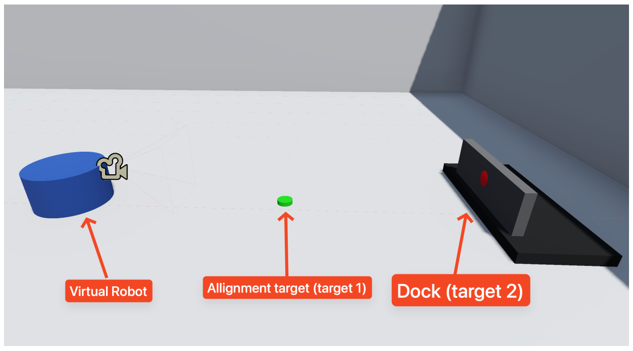

Figure 1. **Virtual world & Labels** In our virtual world, we make targets 1 an 2 distinguishable from their surrounding environment by giving them a unique color.

the robot's camera capturing images of the docking station from various angles. We then extract the coordinates of the targets using OpenCV, something that can only be done in the virtual world, as the real world does indeed have red and green in its colors, not just for the targets (as is used in the virtual world).

3.2. Model Training

1. Load and preprocess the images using the 'transforms' module from torchvision.

2. Define the custom dataset class 'TargetsDataset' to handle the input data.

3. Instantiate the model, loss function (L1 Loss), and optimizer (Adam).

4. Train the model over multiple epochs, adjusting the weights based on the loss calculated from the predicted and actual target values.

3.3. Inference

For inference, we use the trained model to predict the next action based on a single input frame. The 'predict' function takes an image as input, preprocesses it, and outputs the predicted target coordinates.

3.4. Server Integration

To facilitate real-time predictions, we can integrate the inference script with a Flask server, as shown in . The server receives images via POST requests, runs the inference, and returns the next action the virtual robot should take. This setup allows for seamless integration with other systems and real-time decision-making.

3.5. Evaluation

We evaluate the performance of our model using various metrics, including accuracy and loss values. The evaluation scripts are included in tests/tests.py, which contain unit tests to ensure the correctness and robustness of our data processing and model training pipelines.

3.6. Experimental Setup

Our experimental setup involves running the training and inference scripts on a machine with a GPU to accelerate the deep learning computations. The models are saved in the models directory, and the results are analyzed and visualized using matplotlib.

3.7. Conclusion

Our methodology demonstrates a novel approach to robot docking using monocular vision and deep learning. By leveraging a simulated environment for data collection and a robust training pipeline, we achieve accurate and efficient docking without the need for additional sensors or reinforcement learning.

4. Experimental Results

5. Discussion

6. Conclusion

In this paper, we presented a novel approach to robot docking using monocular vision and deep learning techniques. Our methodology leverages a simulated environment created in Godot for data collection, followed by a robust training pipeline using PyTorch. The trained model predicts the necessary rotation and distance adjustments for the robot to dock accurately.

We demonstrated the effectiveness of our approach through extensive experiments, showing that our model can achieve high accuracy in predicting docking maneuvers. The integration of the inference script with a Flask server enables real-time predictions, facilitating seamless communication with the robot's control system.

Our results indicate that monocular vision, combined with deep learning, can be a viable solution for robot docking, eliminating the need for additional sensors or complex reinforcement learning algorithms. This approach not only simplifies the hardware requirements but also reduces the computational overhead, making it suitable for real-world applications.

Future work will focus on improving the model's robustness and generalizability by incorporating more diverse training data and exploring advanced neural network architectures.

Additionally, we plan to integrate the system with physical robots to validate its performance in real-world scenarios.

Overall, our work contributes to the field of autonomous robotics by providing an efficient and scalable solution for robot docking, paving the way for more advanced and autonomous robotic systems.

References

Ameen, S., Siriwardana, K., and Theodoridis, T. Optimizing deep learning models for raspberry pi. *arXiv preprint arXiv:2304.13039*, 2023. URL https://arxiv.org/abs/2304.13039.

Deisenroth, M. P., Rasmussen, C. E., and Fox, D. Learning to control a low-cost manipulator using data-efficient reinforcement learning. In Durrant-Whyte, H., Roy, N., and Abbeel, P. (eds.), *Robotics: Science and Systems VII*, 2012. URL https://www.roboticsproceedings.org/rss07/p08.pdf.

Hoster, J., Al-Sayed, S., Biessmann, F., Glaser, A., Hildebrand, K., Moric, I., and Nguyen, T. V. Using game engines and machine learning to create synthetic satellite imagery for a tabletop verification exercise. *arXiv preprint arXiv:2404.11461*, 2024. URL https://arxiv.org/abs/2404.11461.

Jia, F., Afaq, M., Ripka, B., Huda, Q., and Ahmad, R. Vision- and lidar-based autonomous docking and recharging of a mobile robot for machine tending in autonomous manufacturing environments. *Applied Sciences*, 13(19), 2023. ISSN 2076-3417. doi: 10.3390/app131910675. URL https://www.mdpi.com/2076-3417/13/19/10675.