Shantanu Parab
3426 Tulane Drive, Hyattsville, MD
shantanuparab +1 301-905-7946 sparab@umd.edu shantanu-parab shantanuparabumd

Perception and Path Planning

Autonomous Robot | C++, Python, Perception, Path Planning **Jan 2023- May 2023**

- Developed an autonomous vehicle capable of navigating through complex obstacle courses and performing object manipulation tasks.
- Integrated Raspberry Pi with Ubuntu 20.04 to support ROS2 Galactic, leveraging YOLO for object detection and MiDAS for depth estimation using a monocular camera.
- Implemented localization using encoders and IMU sensors, while employing RRT* algorithm for global path planning and potential field algorithm for dynamic path planning.

Explanation

The project involved designing and constructing an autonomous vehicle capable of navigating through dynamic environments and performing object manipulation tasks autonomously. Utilizing Raspberry Pi with Ubuntu 20.04 as the hardware platform, the project ensured compatibility with ROS2 Galactic, allowing seamless integration with various sensors and algorithms for enhanced functionality.

To enable real-time object detection, the project integrated YOLO (You Only Look Once), empowering the vehicle to identify colored objects within its environment efficiently. Furthermore, MiDAS (Monocular Depth Estimation in Autonomous Driving Scenarios) was leveraged for depth estimation, providing accurate depth perception essential for obstacle avoidance during navigation.

Localization was achieved using encoders and IMU sensors, enabling the vehicle to estimate its position and orientation relative to its surroundings accurately. For path planning, the project implemented the RRT* (Rapidly-exploring Random Trees) algorithm, allowing the vehicle to navigate efficiently through complex environments while avoiding obstacles. Additionally, the potential field algorithm was utilized for dynamic path planning, enabling real-time adaptation of the vehicle's trajectory based on the presence of dynamic obstacles.

Custom libraries were developed to interface Raspberry Pi hardware with C++ programs on the Linux environment, facilitating seamless communication via UART, SPI, and I2C protocols for enhanced control and data exchange. Through meticulous integration of hardware and software components, the project successfully realized an autonomous vehicle capable of robust navigation and object manipulation in dynamic environments.

Path Planning for Autonomous Vehicle | C++, Python, Path Planning **Jan 2024- March 2024**

- Developed an autonomous car simulation in Gazebo using ROS2, integrating sensor fusion techniques to detect and map obstacles accurately with camera and LiDAR data.
- Implemented RRT* path planning algorithms for efficient navigation and dynamic obstacle avoidance, enhancing the car's ability to maneuver through complex environments.
- Leveraged OpenCV for obstacle identification and lane detection, ensuring precise lane following and enhancing overall navigation capabilities of the autonomous car.
- Utilized C++ and Python languages in an Agile Software Development context, emphasizing CI/CD, Code Coverage, and Unit Testing with Google Test Suite to ensure software robustness and reliability.
- Implemented a dynamic simulation of an Autonomous Vehicle with lane-changing capabilities, using RRT* algorithm for path planning. Integrated advanced sensors to detect obstacles and craft alternate routes for enhanced safety.

Explanation

In the project, I utilized a Toyota Prius car model's URDF and integrated sensors and actuators to interact with ROS2 and Gazebo simulation environments. By creating a mathematical model of the car, I generated an action set to drive its behavior within the simulation. Employing the RRT* algorithm, I facilitated the car's navigation by finding the optimal path on the global map, effectively circumventing static obstacles along the way.

To address dynamic obstacles encountered during navigation, I employed a combination of camera and LiDAR sensors to assess the distance between the car and the objects in its vicinity. Leveraging this data, I implemented the potential field algorithm to locally plan a path around dynamic obstacles within the environment.

Throughout the project, I adhered to the Agile Software Development approach, ensuring iterative development cycles and continuous improvement. As part of this methodology, I implemented comprehensive unit testing using the Google Test Suite and conducted code coverage analysis to ensure the reliability and robustness of the software components.

SLAM for Autonomous Robot | C++, Python, Path Planning **Jan 2024- March 2024**

- Simulated the implementation of a Naive SLAM approach, integrating perfect odometry and LiDAR readings to establish foundational knowledge of SLAM techniques
- Implemented particle filter for precise localization of a turtle robot within a known environment, leveraging its robustness for localization tasks.
- Enhanced SLAM techniques by integrating Pose Graph Optimization for elevated localization accuracy and mapping robustness in dynamic environments.

Explanation

The project focused on enhancing Simultaneous Localization and Mapping (SLAM) techniques for a turtle robot within ROS environment. Initially, a Naive SLAM approach was simulated, incorporating perfect odometry and LiDAR readings to lay the groundwork for understanding SLAM principles. This approach provided insights into the basic concepts of localization and mapping.

To improve localization accuracy, a particle filter was implemented to precisely determine the turtle robot's position within a known environment. The particle filter proved robust in handling localization tasks, contributing to the project's overall accuracy.

The project further advanced by integrating Pose Graph Optimization, elevating localization accuracy and mapping robustness in dynamic environments. By refining mapping accuracy through pose graph optimization, particularly in scenarios with imperfect odometry, the turtle robot achieved precise localization and mapping capabilities, crucial for navigating unknown environments effectively. The combination of particle filter and pose graph optimization enhanced the turtle robot's SLAM capabilities, paving the way for reliable autonomous navigation and mapping tasks.

Robot Modelling and Manipulation

Quadruped Gait Simulation: Gazebo|ROS | C++, Python, Robot Manipulation **Nov 2022-Dec 2022**

- Modeled and simulated a quadruped robot to navigate obstacle paths using various gaits, employing SolidWorks for design and creating a URDF for ROS and Gazebo integration.
- Developed the mathematical model of the robot using DH parameters and verified its accuracy using the Peter Corke toolbox in MATLAB, treating each leg as a single serial manipulator.
- Utilized Inverse Kinematics (Jacobian method) to compute joint velocities for the legs, enabling the simulation of correct walking motion, and implemented obstacle detection using cameras and LiDAR for adaptive trajectory switching.

Explanation

The project centered on engineering a quadruped robot simulation capable of traversing obstacle paths and employing diverse gaits for locomotion within simulated environments. It began with a comprehensive robot design process using SolidWorks, where intricate details of the robot were crafted. A URDF (Unified Robot Description Format) was created to seamlessly integrate the robot model with ROS (Robot Operating System) and the Gazebo simulation environment, ensuring compatibility and functionality within the simulation setup.

To ensure an accurate representation of the robot's kinematics and dynamics, the mathematical model of the robot was formulated using DH (Denavit-Hartenberg) parameters. This model underwent rigorous validation using the Peter Corke toolbox in MATLAB, confirming its precision in simulating the robot's motion and behavior. Inverse Kinematics was then implemented using the Jacobian method to compute joint velocities for the robot's legs, enabling the generation of realistic walking motion patterns.

Integration of cameras and LiDAR sensors into the simulation environment enabled obstacle detection and mapping, empowering the robot to dynamically switch trajectories based on real-time environmental cues. Algorithms were developed to interpret sensor data and adapt the robot's gait and trajectory, allowing it to navigate diverse terrain and obstacles efficiently and autonomously. The project's comprehensive approach to simulation

and robotics engineering underscored its ability to tackle complex challenges in robotic locomotion and environment interaction within virtual settings.

Decision Making for Robots

Swarm Robot: Firefly Algorithm | C++,ROS2,CMake.GitCI **Nov 2022-Dec 2022**

- Implemented a bio-inspired decision-making algorithm based on the Firefly algorithm for swarm robots in Gazebo simulation using ROS2.
- Utilized Turtle robots equipped with camera sensors in the Gazebo environment to identify objects of specific colors and create a global map.
- Integrated IMU for global position determination and OpenCV for object identification, enabling robots to attract each other based on intensity and distance, culminating in the gathering near objects of interest.

Explanation

The project aimed to develop a bio-inspired decision-making algorithm for swarm robots, leveraging the Firefly algorithm's principles in Gazebo simulation with ROS2. Turtle robots were utilized in the simulation environment, equipped with camera sensors for object identification. The primary objective was to identify objects of specific colors within the environment and map their locations globally.

Each robot determined its global position using IMU data and utilized camera information processed through OpenCV to locate objects. The Firefly algorithm was adapted to attract robots towards objects based on intensity and distance, simulating collective behavior akin to swarm intelligence. Robots attracted each other towards objects, forming suboptimal solutions through iterative iterations.

The project extended the Firefly algorithm's application from suboptimal optimization problems to scenarios like search and rescue. By gathering near objects of interest, the swarm robots demonstrated the algorithm's effectiveness in collaborative decision-making and object localization tasks, showcasing its potential in real-world applications beyond traditional optimization problems.

Reinforcement Learning

RL using Gazebo and ROS2 | C++, Python, Reinforcement Learning, Pytorch **October 2023-December 2023**

- Implemented Twin-Delayed Deep Deterministic Policy Gradient (TD3) algorithm using PyTorch for quadruped locomotion training in Gazebo and ROS2 simulation environments.
- Developed custom plugins to optimize the reinforcement learning process by seamlessly interfacing with the Gazebo environment.
- Utilized SolidWorks for robot design, C++ libraries for sensor creation, and PyTorch for TD3 network architecture, achieving successful generation of a walking gait for the quadruped robot within a training time of approximately 20 hours.

Explanation

The project focused on simulating quadruped locomotion training within the Gazebo and ROS2 environments using the Twin-Delayed Deep Deterministic Policy Gradient (TD3) approach. Initially, the robot was designed in SolidWorks, and necessary actuators were added before exporting it into the simulation environment. Custom C++ libraries were developed to create sensors that generated appropriate feedback from the Gazebo environment. The TD3 algorithm, implemented using PyTorch, facilitated reinforcement learning for the locomotion task.

Custom plugins were created to seamlessly interface with the Gazebo environment, optimizing the reinforcement learning process. The training process involved testing various rewards for the network, prioritizing forward speed, and displacement, and maintaining the height of the torso above the ground. After approximately 20 hours of training, the project successfully generated a walking gait for the quadruped robot, demonstrating the effectiveness of the TD3 approach in simulated locomotion tasks.

3D Vision

Point Cloud Classification and Segmentation using PointNet | Python, PointNet **Nov 2023-Dec 2022**

- Employed PointNet architecture to classify point clouds into three categories: chairs, vases, and lamps, and segment chairs into six distinct sections.
- Analyzed the impact of point cloud density and rotation on classification and segmentation accuracy.
- Found that reducing point cloud density from 10,000 to 50 points resulted in a notable accuracy drop, with segmentation accuracy decreasing to 30% and classification accuracy decreasing to 66%.
- Extreme rotations along principal axes led to further accuracy drops, with segmentation accuracy dropping to 30% and classification accuracy dropping to 66%.

Explanation

This project provided insights into the robustness of PointNet architecture for point cloud tasks, highlighting the importance of point cloud density and object rotation in accurate classification and segmentation. The findings underscored the need for careful consideration of these factors when deploying point cloud models in real-world applications.

3D Model Generation with Neural Radiance Fields (NeRF) | Python, Neural Radiance Field **Oct 2023-Nov 2023**

- Implemented NeRF to generate detailed 3D models of objects, specifically focusing on reconstructing a Lego bulldozer.
- Applied theoretical concepts of NeRF, leveraging neural networks to predict radiance and capture detailed geometry and appearance of the object.
- Experimented with different epoch configurations to quantify their effect on rendering quality, optimizing model training for enhanced output.

Explanation

This project provided practical experience in advanced computer vision techniques, specifically in the domain of 3D model generation. By effectively applying NeRF, I demonstrated the ability to navigate complex methodologies and translate theoretical knowledge into tangible results. Moreover, the exploration of epoch variations underscored my proficiency in optimizing model performance for realistic renderings.

Overall, this project showcases my expertise in cutting-edge computer vision technologies and my commitment to pushing the boundaries of 3D vision applications. It highlights my capacity to tackle challenging assignments and deliver innovative solutions in computer vision and machine learning.

Deep Learning

Predicing ESport Outcome Using Deep learning | Python, Pytorch, Deeplearning **Oct 2023-Dec 2023**

- Utilized the ResNet50 architecture within a PyTorch-based CNN framework to extract and process intricate visual features essential for outcome prediction in Valorant Esports.
- Integrated Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) architectures with PyTorch-based CNN to discern temporal patterns in gameplay dynamics, enhancing round winner predictions for Valorant matches.
- Developed a custom dataset by segmenting Valorant matches into rounds and converting round videos into frames (images), enabling the extraction of valuable insights from gameplay sequences.

Explanation

The project aimed to predict the win percentage of teams in Valorant Esports by analyzing gameplay dynamics. Initially, a custom dataset was created by segmenting Valorant matches into individual rounds, and round videos were converted into frames (images). These frames were processed using the ResNet50 architecture within a PyTorch-based CNN framework to extract complex visual features crucial for outcome prediction.

To discern temporal patterns in gameplay dynamics, Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) architectures were integrated with the PyTorch-based CNN. This integration facilitated the identification of player behaviors and strategies that led to better chances of winning rounds. The LSTM and GRU models provided insights into the sequential nature of gameplay, enhancing the precision of round winner predictions.

Furthermore, the project involved training the algorithms on video rounds to evaluate the effectiveness and limitations of the architectures. By analyzing the performance of both LSTM and GRU architectures, the project identified areas for improvement and potential future capabilities, contributing to the advancement of predictive analysis in Valorant Esports.

Control of Non-Linear System | Matlab, Control Theory **Dec 2022**

- Linearized a two-pendulum cart non-linear system using Jacobian Linearization, to comment on the controllability and Observability of the system.
- Integrated an LQR and LQG controller into it to improve the output response and compare the two.
- Developed a Luenberger Observer for the system to estimate the state of the system.

Human Object Detection | Software Development, OpenCV, C++ **Oct 2022-Nov 2022**

- Developed a Human Object detection program to be integrated into a mobile robot.
- Used the AIP(Agile Iterative Process) and TDD(Team Driven Development) to develop the project.
- Focused on the software development aspect of the project by making use of Version Control, Sprint Planning, and Backlog Tracking.

Trajectory Design for Panda Robot | Robot Modelling, Python **Oct 2022-Nov 2022**

- Derived the transformations for the robot manipulator using the Denavit-Hartenberg Method.
- Designed the trajectory for a 7-link robot manipulator (Panda) to draw a circle in 3D space using Jacobian, Inverse, and Forward Kinematics.