Target Driven Visual Navigation in Outdoor Scenes

Group 18

Jerry Xu -
Sai Charan Kotra
Satya Sai Bharadwaj Manthri
Vishnu Vardhan Reddy Kanamata Reddy
Abstract

- Target-driven visual navigation in outdoor environments, making an agent utilize vision to navigate through the environment in order to achieve a user-specified objectives has become one of the major difficulties in robotics in recent years.
- We present a novel outdoor environment simulated in the Gazebo framework and RL (Reinforcement learning)
- We show that we were able to replicate real-world outdoor scenarios and successfully conducted experiments in which we were able to make the robot navigate and interact with the objects.
- This creates innovative application possibilities where intelligent agents could pick up on information from their environment and adapt to a variety of settings with little assistance from humans.
Target Driven Visual Navigation

Present Camera View ≠ Target Not Found
Navigate and Check

Target Found
STOP
Main Goals

- Implement Target Driven Visual Navigation Concept
- Make Robot Work in Outdoor Environment With Target Driven Visual Navigation
- Test Robot for the following:
  - Changes in Lighting
  - Changes in Environment
Technologies and Hardware Specifications

- **ROS**
  - Robot Operating System (ROS) is a set of software libraries and tools that help you build robot applications.
  - It provides hardware abstraction, low-level device control, message-passing between processes, and package management.

- **Gazebo**
  - Gazebo is an open-source 3D robotics simulator.
  - It integrated the ODE physics engine, OpenGL rendering, and support code for sensor simulation and actuator control.

- **Windows**
Environment
We are using Model-free Reinforcement learning

States = \{ Target Found: 1, Target Not Found: 0 \}
  - state also includes image that the robot sees (aka a rgb image)

Action = \{ Left, Straight, Right \}

Reward Policy:
  - Collision => -100
  - Target Found => +100

On collision, Episodes End
Episodes - 50
Action per Episode - 1000
Important points of Implementation

Each action has a special designated moves for itself:

- Move Ahead - Moves 1 unit towards front.
- Left and Right - Turn and Move 0.5 unit front.

Learning is done by the following formula:

\[ G[\text{action}] + = \alpha (\text{target} - g[\text{action}]) \]

Target + = reward
Collision and Target Detection

- **Target Detection:**
  - Present view of head camera is taken with the help of a ROS topic `/head_camera/rgb/image_raw` and CvBridge
  - Comparison of target and present view is done with the help of following scores:
    - Mean Square Error (MSE)
    - Structural Similarity Index Measure (SSIM)

- **Collision Detection:**
  - For moving the robot, we developed a code. While developing itself, we created cases to detect collisions by the following measures.
DEMO (Output)
Expected output matches our output as seen in the previous slide - So, the navigation is successful
Thank You