NASA Space Mission AI Project Plan - Conceptual Design Track

Project Overview

Student Name: Andrew Kuruvilla
Project Track: Conceptual Design Track
Project Name: Andrew_Kurvilla_Conceptual_Design_Track
GitHub Repository: https://github.com/drewvilla1/Andrew_Kurvilla_Conceptual_Design_Track

Project Description: This project proposes a detailed conceptual design for an AI system to optimize autonomous navigation for a NASA space mission. The AI solution will focus on real-time path planning and obstacle avoidance for a Mars rover, leveraging machine learning and sensor data fusion to enhance mission efficiency and safety.

Project Proposal (20 Points)

Problem Statement

The exploration of Mars requires rovers to navigate complex terrains with minimal human intervention due to communication delays. Current navigation systems rely on pre-programmed routes and limited real-time adaptability, which can lead to inefficiencies or mission risks when encountering unexpected obstacles (e.g., rocks, craters, or sand traps).

Proposed Solution

Develop an AI-based autonomous navigation system for a Mars rover that:

- Uses a combination of computer vision, reinforcement learning, and sensor fusion to dynamically plan paths.
- Processes data from LiDAR, stereo cameras, and inertial measurement units (IMUs) to detect and classify obstacles.
- Adapts to changing environmental conditions (e.g., dust storms, lighting changes) in real-time.
- Optimizes for energy efficiency and mission objectives (e.g., reaching scientific targets).

Objectives

1. Design an AI model architecture for real-time path planning.
2. Create a sensor fusion framework to integrate multiple data sources.
3. Develop a testing plan to validate the system under simulated Mars conditions.
4. Ensure the system is robust, scalable, and aligns with NASA's mission requirements.

Scope

- Focus on navigation and obstacle avoidance for a Mars rover.
- Exclude onboard scientific analysis or communication systems.
- Assume a simulated environment for testing and validation.

Detailed Solution Plan (40 Points)

System Architecture

The proposed AI system consists of the following components:

1. **Data Acquisition Module**:
    - Inputs: LiDAR point clouds, stereo camera images, IMU data.
    - Function: Collects and preprocesses raw sensor data for analysis.
    - Technology: Python-based data pipelines, OpenCV for image processing, PCL (Point Cloud Library) for LiDAR.
2. **Sensor Fusion Module**:
    - Approach: Use a Kalman filter to integrate LiDAR, camera, and IMU data for accurate environmental mapping.
    - Output: A unified 3D map of the rover's surroundings.
    - Rationale: Combining multiple sensor inputs improves robustness against noise and environmental variability.
3. **Obstacle Detection and Classification**:
    - Model: Convolutional Neural Network (CNN) trained on labeled Mars terrain data.
    - Function: Identifies obstacles (e.g., rocks, slopes) and classifies them by risk level.
    - Training Data: Simulated Mars terrain datasets (e.g., NASA's Mars Yard) and synthetic data augmentation.
4. **Path Planning Module**:
    - Algorithm: Reinforcement Learning (RL) with a Deep Q-Network (DQN) to optimize paths.
    - Inputs: 3D map, obstacle classifications, mission objectives.
    - Outputs: Optimal path avoiding obstacles while minimizing energy use.
    - Rationale: RL allows the rover to learn from experience and adapt to new terrains.
5. **Control Interface**:

- Function: Translates planned paths into rover motor commands.
- Technology: ROS (Robot Operating System) for communication between AI and rover hardware.

Workflow

1. Sensors collect raw data (LiDAR, cameras, IMU).
2. Data is preprocessed and fused into a 3D environmental map.
3. CNN identifies and classifies obstacles in the map.
4. RL-based path planner generates an optimal route.
5. Control interface sends commands to the rover's motors.
6. System continuously updates based on new sensor data.

Assumptions

- Rover hardware supports real-time data processing (e.g., onboard GPU).
- Simulated Mars environment is available for testing.
- NASA provides access to relevant terrain datasets for training.

Risk Mitigation

- **Risk**: Sensor noise or failure.
  - **Mitigation**: Implement redundancy in sensor fusion and fallback to conservative navigation if data quality drops.
- **Risk**: Model overfitting to simulated data.
  - **Mitigation**: Use data augmentation and domain randomization during training.
- **Risk**: Computational limitations.
  - **Mitigation**: Optimize algorithms for efficiency and prioritize critical tasks.

Testing Plan (25 Points)

Testing Objectives

1. Validate the accuracy of obstacle detection and classification.
2. Ensure path planning adapts to dynamic environments.
3. Confirm system robustness under simulated Mars conditions.
4. Verify energy efficiency and mission objective alignment.

Test Scenarios

1. **Static Obstacle Avoidance**:
   - Environment: Simulated Mars terrain with rocks and craters.

    - Objective: Rover navigates to a target without collisions.
    - Metric: Success rate (target reached without collision).
2. **Dynamic Environment**:
    - Environment: Terrain with changing conditions (e.g., dust storm reducing visibility).
    - Objective: Rover adjusts path in real-time.
    - Metric: Path adaptation time and collision avoidance.
3. **Edge Cases**:
    - Environment: Extreme slopes, narrow passages, or sensor noise.
    - Objective: Rover avoids mission failure (e.g., getting stuck).
    - Metric: Failure rate and recovery time.
4. **Energy Efficiency**:
    - Environment: Long-distance navigation with multiple obstacles.
    - Objective: Minimize energy consumption while reaching target.
    - Metric: Energy usage compared to baseline (e.g., shortest path).

Testing Tools

- **Simulation Platform**: Gazebo with ROS for Mars terrain simulation.
- **Dataset**: NASA’s Mars Yard dataset and synthetic terrain data.
- **Metrics Tracking**: Custom Python scripts to log success rates, energy usage, and adaptation times.

Validation Criteria

- Obstacle detection accuracy: >90% precision and recall.
- Path planning success rate: >95% in static environments, >85% in dynamic environments.
- Energy efficiency: At least 20% improvement over shortest-path baseline.
- Robustness: No mission failures in >90% of edge case tests.