[Project Title: e.g., Autonomous Object Tracking Robot] Project Report

Team Members:

- Team Member 1 (ID: 20xxxxxx)
- Team Member 2 (ID: 20xxxxxx)
- Team Member 3 (ID: 20xxxxxx)
- Team Member 4 (ID: 20xxxxxx)
- Team Member 5 (ID: 20xxxxxx)

Date: [Current Date]

---

1. Project Overview

1.1 Project Title and Task Description

This project aims to develop a ROS-based autonomous mobile robot capable of [describe the main task, e.g., tracking and following colored objects in real-time using RGB-D camera].

1.2 Project Objectives

- Objective 1: [e.g., Implement robust color-based object detection]
- Objective 2: [e.g., Develop smooth trajectory planning algorithm]
- Objective 3: [e.g., Achieve real-time visual servoing control]

2. System Design

2.1 System Architecture

System Architecture Diagram Placeholder *Figure 1: System architecture showing ROS nodes and their communication relationships.*

2.2 Module Division

The system consists of the following modules:

- **Perception Module**: Processes camera data for object detection and localization.
- **Planning Module**: Generates optimal paths based on detected object positions.
- **Control Module**: Executes velocity commands for smooth robot motion.
- **Integration Module**: Coordinates all subsystems and handles exceptions.

3. Algorithm Principles

3.1 Core Algorithm Description

3.2 Algorithm Selection Rationale

Table 1: Comparison of Detection Algorithms

Method	Speed	Accuracy	Complexity
HSV Thresholding	Fast	Medium	Low
Template Matching	Slow	Low	Medium

We selected HSV thresholding because [explain your reasoning].

3.3 Key Parameters

- HSV Range
- Control Gain
- Maximum Velocity

4. Implementation Details

4.1 Technical Challenges and Solutions

1. **Challenge**: Noisy depth data causing unstable distance estimation. **Solution**: Applied median filter and temporal smoothing.
2. **Challenge**: Robot oscillation during tracking. **Solution**: Implemented velocity ramping and dead zone.

4.2 Debugging Notes

- Issue: TF frame mismatch between camera and base_link. Fixed by adding static transform publisher.
- Issue: Message queue overflow. Resolved by setting queue_size=1 for real-time topics.

5. Experimental Results

5.1 Quantitative Results

Path planning evaluation encompasses three main categories of metrics.

Path Quality Metrics assess the generated path's characteristics including length (total distance using Euclidean or Manhattan measurements), smoothness (evaluating turns and curvature), safety (obstacle clearance and collision risk), and time efficiency.

Algorithm Performance Metrics measure computational efficiency (runtime, memory, node expansions), solution quality (success rate, optimality ratio), and robustness across varying conditions

Dynamic Environment Metrics evaluate the algorithm's adaptability through replanning frequency and real-time performance capabilities.

Path Quality Metrics

- Path Length: The total distance traveled from start to goal, measured either as straight-line (Euclidean) or grid-based (Manhattan) distance.
- Smoothness: The continuity and fluidity of the path, quantified by the number of direction changes, curvature variations, and turning angles.

- Safety: The degree of collision avoidance, measured by minimum clearance from obstacles, probability of collision, and safety buffer zones.
- Time Efficiency: The temporal performance of path execution, including total travel duration and expected arrival time.

Algorithm Performance Metrics

- Computational Efficiency: Resource consumption during path computation, including processing time, memory footprint, and number of search nodes explored.
- Success Rate: The percentage of scenarios where the algorithm successfully finds a valid path to the goal.
- Suboptimality Ratio: The ratio comparing the found path's cost to the theoretical optimal path cost (value of 1.0 = optimal).
- Robustness: The algorithm's stability and consistent performance across diverse environments and varying parameter settings.

Dynamic Environment Metrics

- Replanning Frequency: How often the algorithm must recalculate the path due to environmental changes or new obstacles.
- Adaptability: The algorithm's ability to adjust to dynamic obstacles, moving targets, or changing constraints.
- Real-time Performance: The capability to compute and update paths within strict time constraints for time-critical applications.

5.2 Qualitative Results

Detection Screenshot *(a) Object detection visualization*

Trajectory Plot *(b) Robot trajectory during tracking*

Figure 2: Experimental visualization results.

5.3 Result Analysis

The system achieved [good/satisfactory] performance because [analysis]. However, performance degraded under [conditions] due to [reasons]. Potential improvements include:

- Implementing adaptive thresholding for varying lighting conditions
- Adding Kalman filter for smoother trajectory estimation

6. Team Contribution

Table 3: Team Member Contributions

Name	Student ID	Module	Specific Work	Contribution
Member 1	20xxxxxx	Perception	HSV tuning, depth fusion	30%
Member 2	20xxxxxx	Planning	A* algorithm implementation	25%

Name	Student ID	Module	Specific Work	Contribution
Member 3	20xxxxxx	Control	Visual servoing, velocity smoothing	25%
Member 4	20xxxxxx	Integration	System debugging, video recording	20%
Member 5	20xxxxxx	...	...	...%
Total				**100%**

Member Signatures:

Member 1	Member 2	Member 3	Member 4	Member 5
___________	___________	___________	___________	___________

7. Summary and Reflection

7.1 Project Achievements

- Successfully implemented a complete ROS-based robotic system
- Gained hands-on experience with computer vision and robot control
- Developed teamwork and project management skills
- Deepened understanding of sensor fusion and real-time systems

Appendix

A.1 Code Repository

Complete source code is available at: https://github.com/username/project-repo

A.2 Additional Experimental Data

A.3 References

1. ROS Wiki: http://wiki.ros.org/
2. OpenCV Documentation: https://docs.opencv.org/
3. Additional references as needed