1 Introduction / Overview

The objective of this project is to demonstrate mastery of robotic perception concepts through the design and implementation of a simulated mobile robot operating in a physics-enabled environment. The final system will integrate computer vision, localization, mapping, and navigation to allow a robot to autonomously reach a goal while avoiding both static and dynamic obstacles.

This report documents **Milestone 1**, which focuses on **computer vision**. At this stage, the robot uses a monocular camera to observe the environment and identify dynamic obstacles based on visual motion cues. The implementation emphasizes understanding and applying the underlying mathematical models of motion perception rather than relying on high-level computer vision libraries.

The computer vision algorithms developed for this milestone are implemented in code and made publicly available for reference and reproducibility. The source code corresponding to the computer vision component of this project can be found in the following GitHub repository:

`https://github.com/pepepepepordu/perception-of-cognitive-robot-project`

2 Robot Simulation

This section is intentionally left blank for integration with the simulation and environment description provided by other team members.

3 Computer Vision

3.1 Mathematical Model

The robot perceives the environment through a sequence of camera images, modeled as a discrete **luminance (intensity) function**:

$$I(x, y, t)$$

where I represents the grayscale luminance at pixel coordinates (x, y) at time t. This formulation is appropriate since motion estimation relies on intensity variations rather than color information.

Motion in the image is modeled using the **brightness constancy assumption**, which states that the luminance of a moving point remains constant between consecutive frames:

$$I(x, y, t) = I(x + u, y + v, t + 1)$$

Assuming small inter-frame motion, a first-order Taylor expansion yields the **optical flow constraint equation**:

$$I_x u + I_y v + I_t = 0$$

where:

- I_x and I_y are spatial intensity gradients,
- I_t is the temporal intensity derivative,
- (u, v) represents the pixel displacement between frames.

Spatial gradients are approximated using central differences:

$$I_x \approx \frac{I(x+1, y) - I(x-1, y)}{2}, \quad I_y \approx \frac{I(x, y+1) - I(x, y-1)}{2}$$

The temporal gradient is computed as:

$$I_t = I(x, y, t + 1) - I(x, y, t)$$

Rather than solving for a continuous flow field, motion is estimated by evaluating a discrete set of candidate motion vectors:

$$(u, v) \in \{(-1, 0), (1, 0), (0, -1), (0, 1), (0, 0)\}$$

For each candidate motion, the mean squared error of the optical flow constraint is computed:

$$E(u, v) = \frac{1}{N} \sum (I_x u + I_y v + I_t)^2$$

The motion vector that minimizes this error is selected as the estimated motion for a given image feature.

3.2 Method of Implementing

Each camera frame is first converted to grayscale and smoothed using a Gaussian filter to reduce noise and stabilize gradient estimation. Rather than processing all pixels, a small number of salient feature points are selected using the **Shi–Tomasi corner detection method**, implemented via the *Good Features to Track* algorithm. This method identifies pixels with strong intensity variation in two orthogonal directions, making them well suited for motion estimation.

Only a limited number of feature points are retained to reduce computational complexity while preserving robustness. These feature points are extracted from the previous frame and serve as reference locations for local motion analysis.

For each selected feature point, a local 5×5 image patch is extracted from both the previous and current frames. Spatial and temporal gradients are computed within this region. If the magnitude of the spatial gradients is below a threshold, the feature point is discarded to avoid unstable motion estimates.

The optical flow constraint is evaluated for each candidate motion vector, and the motion direction that minimizes the mean squared error is selected. Feature points exhibiting non-zero motion are classified as belonging to **dynamic objects**, while points with zero estimated motion are treated as **static elements** of the environment.

4 Implementation and Demonstration

The computer vision pipeline operates in real time and follows these steps:

1. Capture consecutive camera frames
2. Convert frames to grayscale and apply Gaussian smoothing
3. Select feature points using the Shi–Tomasi method
4. Extract local image patches around each feature
5. Compute spatial and temporal gradients
6. Evaluate candidate motion vectors using mean squared error
7. Classify motion direction for each feature point

Detected feature points are visualized on the camera feed, and motion direction estimates are printed for debugging and verification. Moving objects generate consistent non-zero motion estimates, while static background features exhibit minimal or no detected motion. This behavior demonstrates the system's ability to distinguish dynamic obstacles using visual information alone.

5 Future Work

Future work will focus on the next project milestone, which targets **Simultaneous Localization and Mapping (SLAM)**. In this phase, the robot will construct a map of the environment while simultaneously estimating its own position within that map. Visual information from the camera will be integrated with range-sensing data to enable consistent map construction and pose estimation. The distinction between static and dynamic obstacles developed in this milestone will support SLAM by preventing moving objects from being incorporated into the map.