Midterm Project Proposal

Daniel Zinobile and Anvesh Som

Objective

Our objective is to deliver a software pipeline that takes footage from a consumer-grade dashcam and tracks the position of people in the footage with respect to the camera's reference frame, within a critical safe distance. We will demonstrate functionality using footage taken from Daniel's personal dashcam while driving around the area. The program will ideally work with day and night conditions and perform tracking in real time, so that it could theoretically be integrated with a braking or steering system.

Software Plan

Purpose & objectives

Build a real-time C++ system that, from a single roof/bumper-mounted camera, continuously-

- calibrates/undistorts input frames (or loads supplied parameters),
- maps image pixels to ground coordinates via planar homography (assuming flat road, known camera pose/height)
- detects people, projects their foot points to ground, filters out far targets (> D), assigns persistent IDs, and raises a "brake" alert if any target is within D_close.

Scope (v1)

On-vehicle monocular processing for pedestrians on drivable flat(ish) surfaces at urban speeds; daytime and decent illumination, forward-looking FOV; single camera only.
Out-of-scope for v1: multi-camera fusion, radar/LiDAR fusion, full vehicle control, fine-grained trajectory prediction, curb-cuts/ramps modeling, steep grades.

Constraints

Real-time on single edge CPU; limited on-road testing; academic-license toolchains.

Design

- Camera Calibration

  Software performs camera calibration to find intrinsics and distortion parameters. Also allows users to enter their own parameters with a flag, or calculate those parameters by filming a calibration checkerboard with their camera, via a built-in calibration program.

- Tracking

  An open-source, and lightweight image classification model (to allow processing heavy extensions to pipeline) will be used to identify and place a bounding box around people in the footage. Post calculating image to planar-ground homography mapping, the coordinates of the pedestrian's position where they contact that plane (i.e. where their feet are) can be determined from planar homography - Assuming camera height is fixed and ground is planar.

- Range filtering and ID

  Humans too far from distance of interest are not assigned IDs. Rest are assigned, and are matched & tracked in the next frame by assigning that ID to the nearest person in the next frame. Humans too close to the car, trigger the alert signal.

Risks

The calibration and tracking ability of the program may be affected by the following:

- Camera vibration
  - Fallback: sanity check horizontal line; allow quick recalibration.
- Low resolution, motion blur, rolling shutter, and interlacing on dashcam footage
  - Implement pre-enhancer if processing limitations allow for it
- Variable lighting conditions
  - Add adaptive gamma correction/contrast normalization if processing limitations allow
- Image classification model may limit running in real time
  - Fallback: Use greyscale footage for faster processing
- Large inaccuracies associated with assumption of level camera and plane
  - Fallback: Using YOLO-pose or Blazepose for pose estimation, or use simple trigonometric calculations based on average human height for rough distance approximations

Development

The calibration program and tracking program will be written separately. Daniel will be the driver for the calibration program and will use TDD to create test cases and write code to pass the test cases. Anvesh will be the Navigator for this. The same will be done but with reversed roles for the tracking program.

Methods

- Programming Language/Build: C++17 (or later), CMake
- Revision Control: Github
- Libraries:
  - OpenCV (calibration/undistort/solvePnP/homography)
  - Eigen and Numpy for linear algebra involved in transformations and projective geometry
  - YOLO-tiny or similar image classification model

Metric and monitoring

Performance: end-to-end latency (human in camera -> alert); FPS stability
Perception: Homography reprojection error (measured at known markers); False alerts/min

Deliverables

The final deliverables will be as follows:

- Example dashcam calibration footage
- Multiple example files of dashcam driving footage
- Example text or csv file with camera calibration matrix
- Header and source files for class definitions
- Test files
- Main file to take input arguments and run program

To run the program, the user will do the following:

- Build executables per instructions in the README
- Run main executable with input arguments for:
    - Whether or not to run the calibration program
    - Location of existing calibration matrix file, or
    - Location of calibration footage
    - Location of dashcam footage to run with

While the program is running, a display will show the dashcam footage with overlaid bounding boxes and the xyz positions of people relative to the camera.

Definitions & Acronyms

FPS - Frame per Second
FOV - Field of View
TDD - Test-Driven Development
PnP - Perspective-n-Point