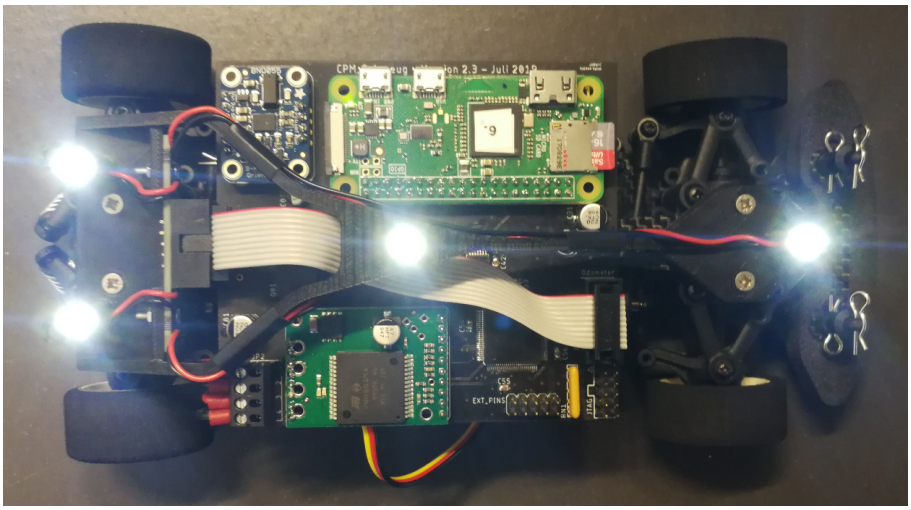
Indoor Positioning System

The Indoor Positioning System (IPS) detects the position and orientation (pose) and identity of multiple vehicles simultaneously. The poses are updated at 50 Hz.

Optics

The Indoor Positioning System Setup works with active light sources (LEDs) on the vehicles and a camera looking down from the ceiling. The LEDs are detected based on their high brightness. The camera is set to a very short exposure (~100 microseconds). Thus, the ambient light creates a very small signal (almost black), while the LEDs still appear as white dots. The short exposure time also eliminates the problem of motion blur. At a top speed of 4 m /s the vehicle travels 0.4 mm during the exposure.

Other light sources and reflective surfaces on the vehicle can create false signals and must be covered with tape. This includes the connectors on the Raspberry Pi and the status LEDs on the motor speed controller.



Vehicle Pose

The outer three LEDs indicate the vehicle pose and are permanently illuminated. The section Pose Calibration describes how the LED positions are related to the reference pose.

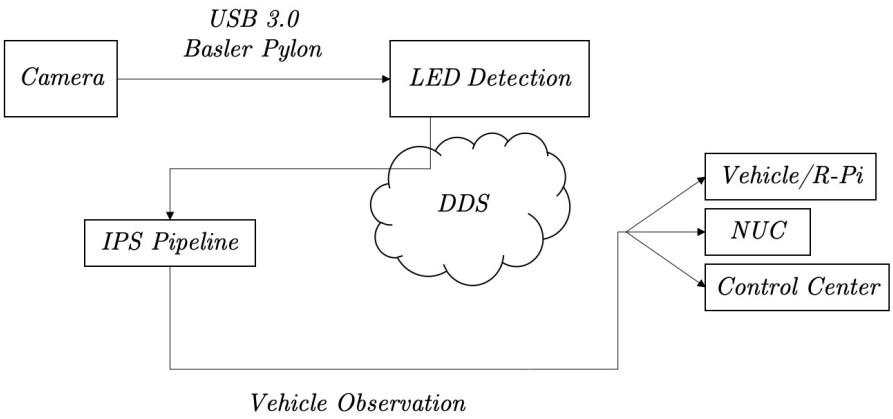
Vehicle Identification

The central LED flashes in a pattern that is different for each vehicle. The patterns are chosen such that sampling effects do not create ambiguous signals. See our paper Vision-Based Real-Time Indoor Positioning System for Multiple Vehicles for more information.

Vehicle ID	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
Period of flashing, in number of frames	4	7	10	13	16	7	10	13	16	19	10	13	16	19	22	13	16	19	22	25	16	19	22	25	28
LED illumination duration, in number of frames	2	2	2	2	2	5	5	5	5	5	8	8	8	8	8	11	11	11	11	11	14	14	14	14	14

Software Architecture

The IPS software has two major components, the LED detection and the IPS pipeline. The LED detection reads images from the camera at 50 Hz and extracts the list of image coordinates for all visible LEDs. These `LedPoints` are published via DDS. The raw images are not saved or published, as this would create impractically large data volumes. The IPS pipeline processes the LED points and publishes `VehicleObservations`, which include the vehicle's position, orientation, identity and a timestamp.

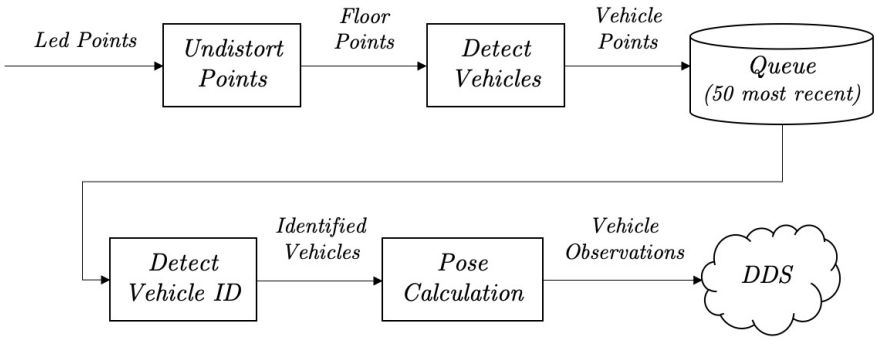


LED Detection

The LED detection uses the OpenCV functions `cv::threshold`, `cv::findContours` and `cv::moments` to find the centers of the LEDs.

IPS Pipeline

The IPS pipeline performs a relatively complex data processing task. To make the software easier to understand, it is broken down into independent, stateless processing steps.



The **UndistortPoints** function transforms the image/pixel coordinates to metric coordinates on the driving surface (floor). The calculation is based on a calibration image. See https://github.com/embedded-software-laboratory/cpm-lab/tree/master/matlab_scripts/camera_calibration

The **DetectVehicles** function groups and classifies points as `front`, `center`, `back_left` or `back_right`. Points that appear to not belong to a vehicle are discarded.

The **Queue** collects the 50 most recent samples of the vehicle points. This is necessary, as the vehicle ID can not be determined from a single image.

The **DetectVehicleID** function extracts the ID for each vehicle from the last second (50 frames) of vehicle points.

The **PoseCalculation** function calculates the vehicle reference pose based on the `front`, `back_left` and `back_right` points. The calculation is based on manually collected calibration data. See https://github.com/embedded-software-laboratory/cpm-lab/tree/master/matlab_scripts/ips_pose_calibration

The implementation can be found at https://github.com/embedded-software-laboratory/cpm-lab/tree/master/ips2/src

Calibration

Camera Calibration

First, other PVC canvases are removed from the floor to reveal the checkerboard canvas. Then an image of the checkerboard pattern is taken using the `PylonViewerApp`. The exposure is adjusted such that the checkerboard corners are well resolved. If the exposure is too short or long, the corners are "rounded off", which degrades the quality of the calibration.

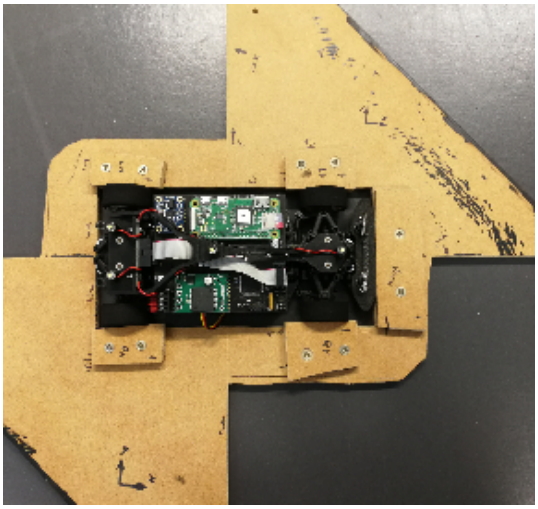
The Matlab function `detectCheckerboardPoints` is used to determine a list of matching points `(x_floor, y_floor, x_image, y_image)`. Then a 5th order 2D polynomial is fitted to the data using linear least squares.

See https://github.com/embedded-software-laboratory/cpm-lab/tree/master/matlab_scripts/camera_calibration

Pose Calibration

The vehicle pose has the format `(x, y, yaw)`. The yaw is the rotation angle in radians of the vehicle around the vertical axis in the counter-clockwise direction. When yaw=0, the vehicle points in the x-direction. The coordinates (x, y) give the location of the vehicle's *reference point*. The reference point is defined as the geometric center between the front and rear axle.

To accurately implement this definition of the pose, another measurement and calibration procedure is performed. First we need a means of accurately placing the vehicle on the floor, such that the true pose is known. The vehicle is clamped into a piece of wood, which extends the vehicle's local x/y coordinate system. This makes it simple to manually align the vehicle with the checkerboard pattern.



While the IPS is running, the vehicle is placed in various poses on the floor, following a particular calibration sequence. The calibration features `(back_x, back_y, direction_x, direction_y)` are recorded from the running IPS. A linear calibration is then fitted using least squares.

Coordinate Systems

IPS Image Coordinates

The image coordinates correspond to the IPS image sensor pixels. They are only relevant for the IPS LED detection.

Floor Coordinates

(mathjax-inline(x \in [0, 4.5], y \in [0, 4])mathjax-inline)

X/Y units: Meter

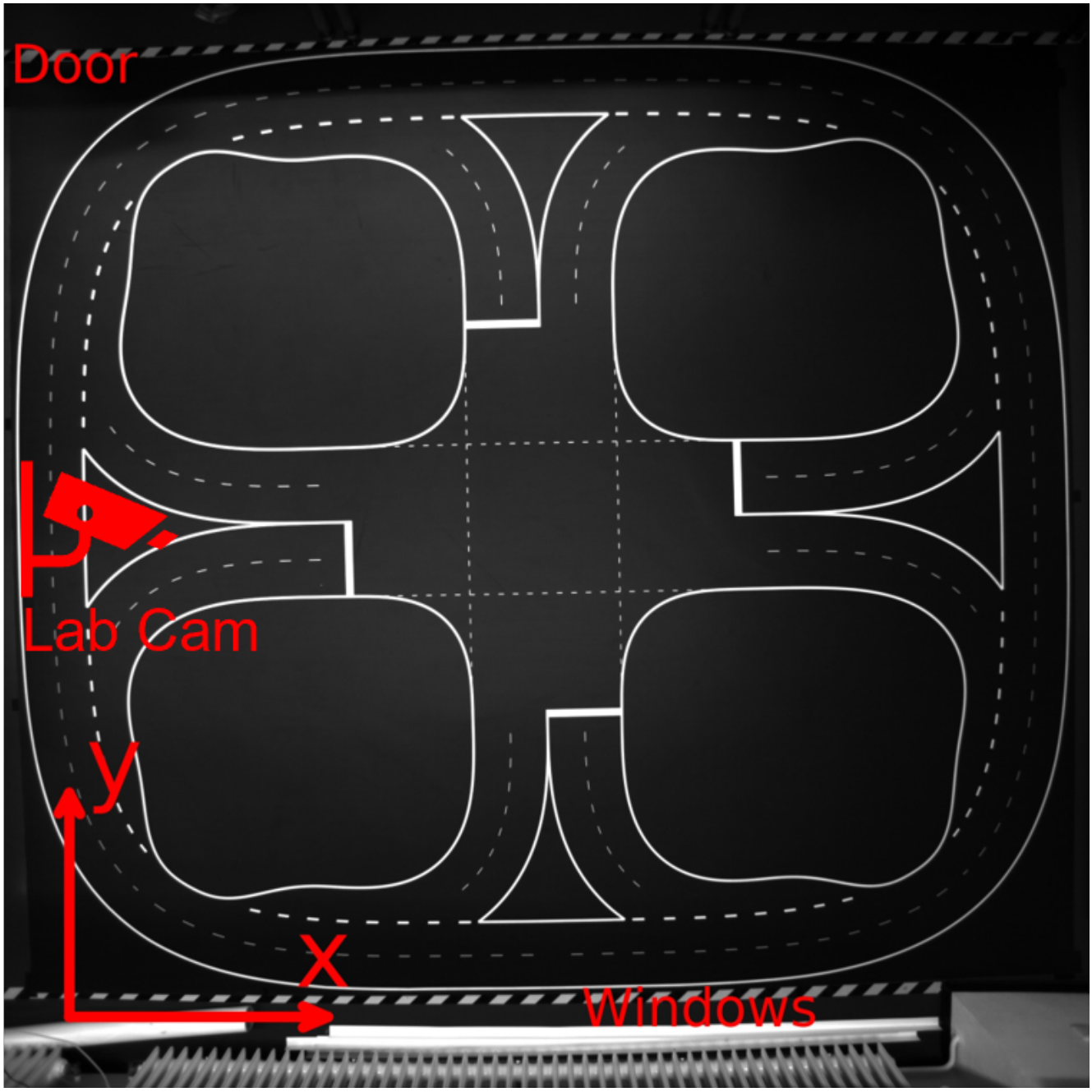
Yaw unit: Radian

Origin: corner of the PVC canvas, towards the windows on the right.

Yaw angle: Measured from the x-axis, counterclockwise.

Useful yaw equations:

```
direction_x = cos(yaw)
direction_y = sin(yaw)
yaw = atan2(direction_y, direction_x)
```



Vehicle Coordinates

X-axis: forwards

Y-axis: left

Origin: center point between both axles.

In practice, this coordinate system is realized with a calibration jig.