TURTLEBOT3 LINE FOLLOWING MISSION

Robotics Project

Emmanouil Koutoulakis

Emmanouil Markodimitrakis

December 2021

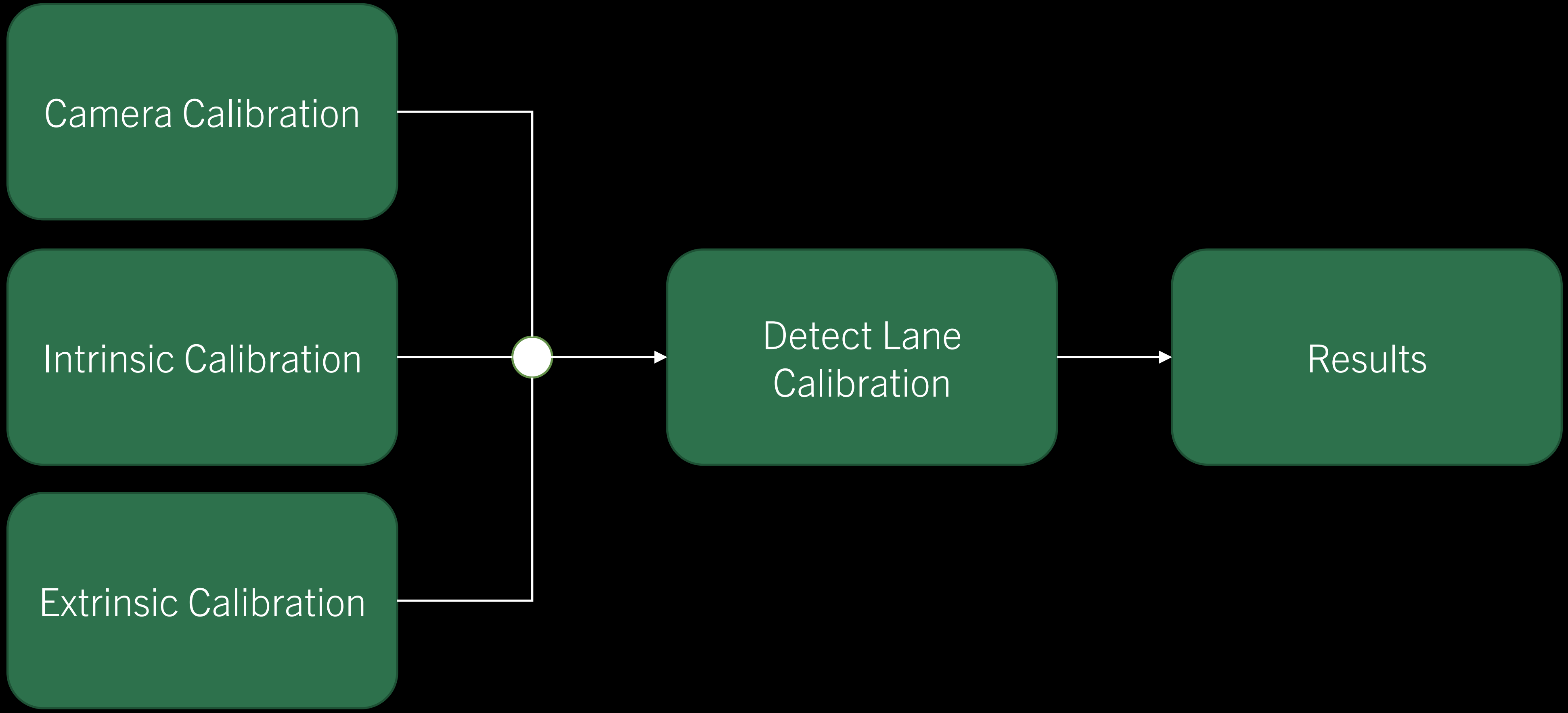
OUTLINE

- Overview
- Project Structure
- Calibration
- Algorithm
- Demo -Videos
- Limitations

OVERVIEW

- This project used ROS to demonstrate a simple line following Turtlebot in both Gazebo and real environment using Ubuntu 20.04, ROS Melodic and OpenCV 3
- The Turtlebot3 is used as the vehicle to demonstrate the concept which uses simple Image Processing Techniques
- The main decider of the robots trajectories is the camera which is integrated on the robot

PROJECT STRUCTURE



Camera Calibration

Intrinsic Calibration

Extrinsic Calibration

Detect Lane Calibration

Results

CAMERA CALIBRATION

- It is an optional step
- Only if the image is not clear
- Parameters able for modifications:
  - Contrast
  - Brightness
  - Sharpness
  - Saturation
  - Zoom
  - Etc.

INTRINSIC CALIBRATION

- In this phase, we tried to fix the distortion of the acquired images
- The camera_calibration package [1] contains the method to undistort the frames
- We used a checkerboard for the intrinsic calibration

$$x_{distorted} = x + [2p_1xy + p_2(r^2 + 2x^2)]$$

$$y_{distorted} = y + [p_1(r^2 + 2x^2) + 2p_2xy]$$

$$cameramatrix = \begin{bmatrix} f_x & 0 & C_x \\ 0 & f_y & C_y \\ 0 & 0 & 1 \end{bmatrix}$$

$$DistortionCoefficients = (k_1,\ k_2,\ p_1,\ p_2,\ k_3)$$

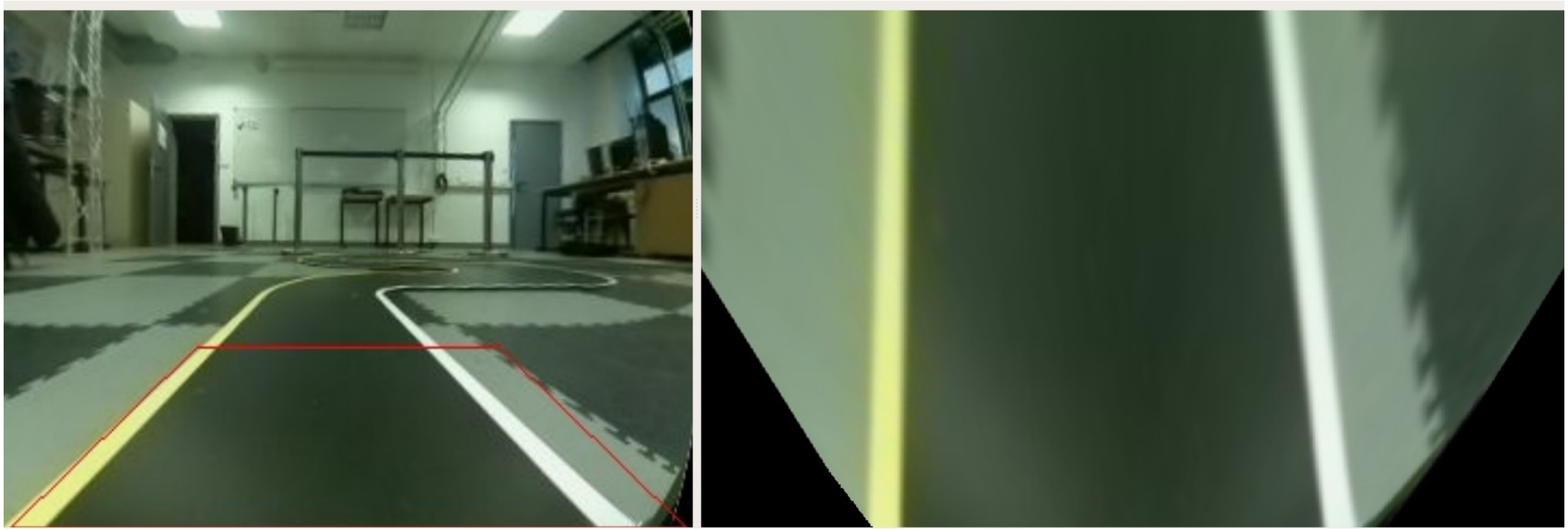
IMAGE PROJECTION - EXTRINSIC CALIBRATION

- Get 4 coordinates which corresponds to the projected frame
- Find the homography according to the projected frame and the image that will be transformed
- The image transformation established using homography transformation
- The functions that used for this are in OpenCV:
  - cv2.findHomography()
  - cv2.warpPerspective()

IMAGE COMPENSATION - EXTRINSIC CALIBRATION

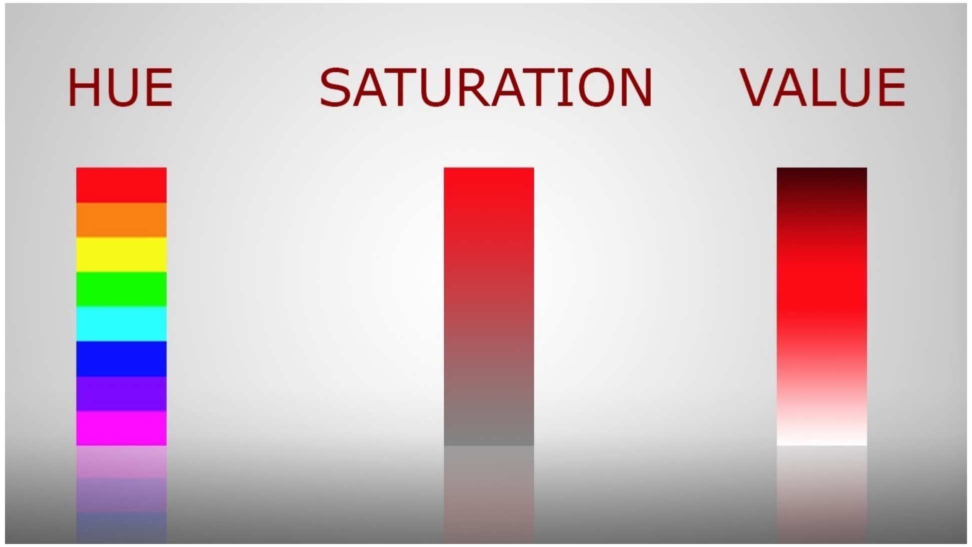
- Image Compensation used for the ground-projected image
- A grayscaled version of the projected frame used
- Histogram Equalization performed according to the grayscaled frame

EXTRINSIC CALIBRATION

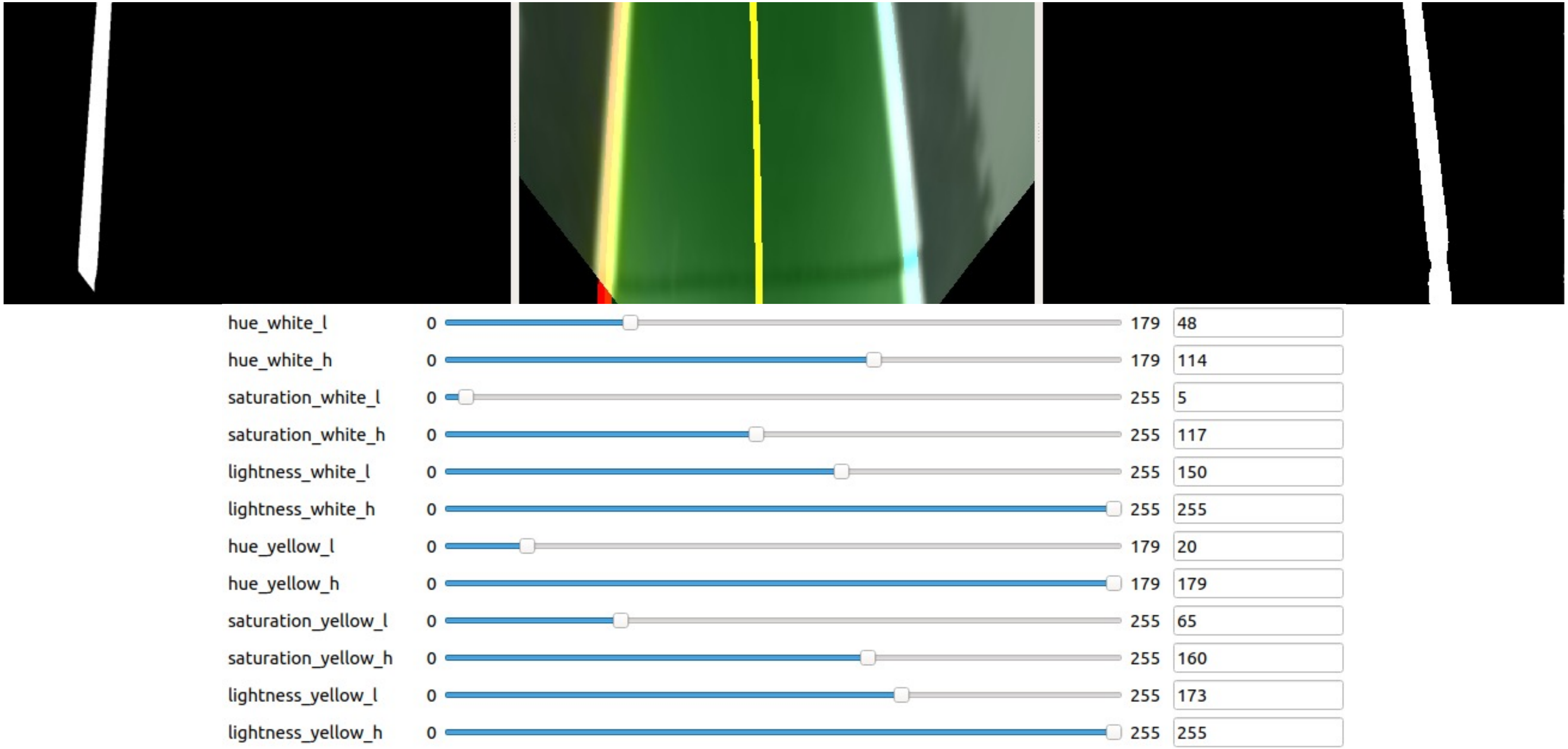


DETECT LANE CALIBRATION

- After the image compensation, two identical images used for yellow and white lines
- We convert the image to HSV
- Hue is the colour region
- Saturation is the ration of colourfulness to brightness
- Value (or Lightness) is the average of the largest and smallest colour components



DETECT LANE CALIBRATION



hue_white_l	0	179	48
hue_white_h	0	179	114
saturation_white_l	0	255	5
saturation_white_h	0	255	117
lightness_white_l	0	255	150
lightness_white_h	0	255	255
hue_yellow_l	0	179	20
hue_yellow_h	0	179	179
saturation_yellow_l	0	255	65
saturation_yellow_h	0	255	160
lightness_yellow_l	0	255	173
lightness_yellow_h	0	255	255

ALGORITHM – DETECT LANES

1. Get the current image
2. Find the white and yellow lines
3. Auto-adjust lightness according to the fraction number of the lines (the sum of the non zero pixels)

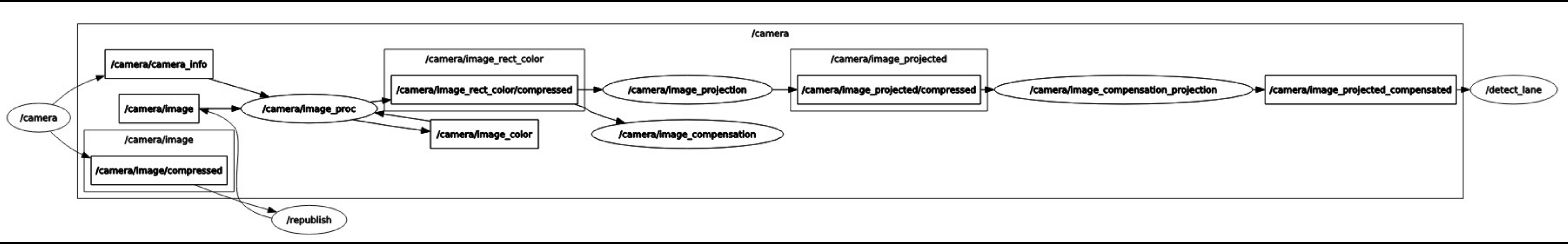
   ```
   If the fraction number > 35000 then:
           Increase lightness
   Else If the franction number < 35000 then:
           Decrease lightness
   ```

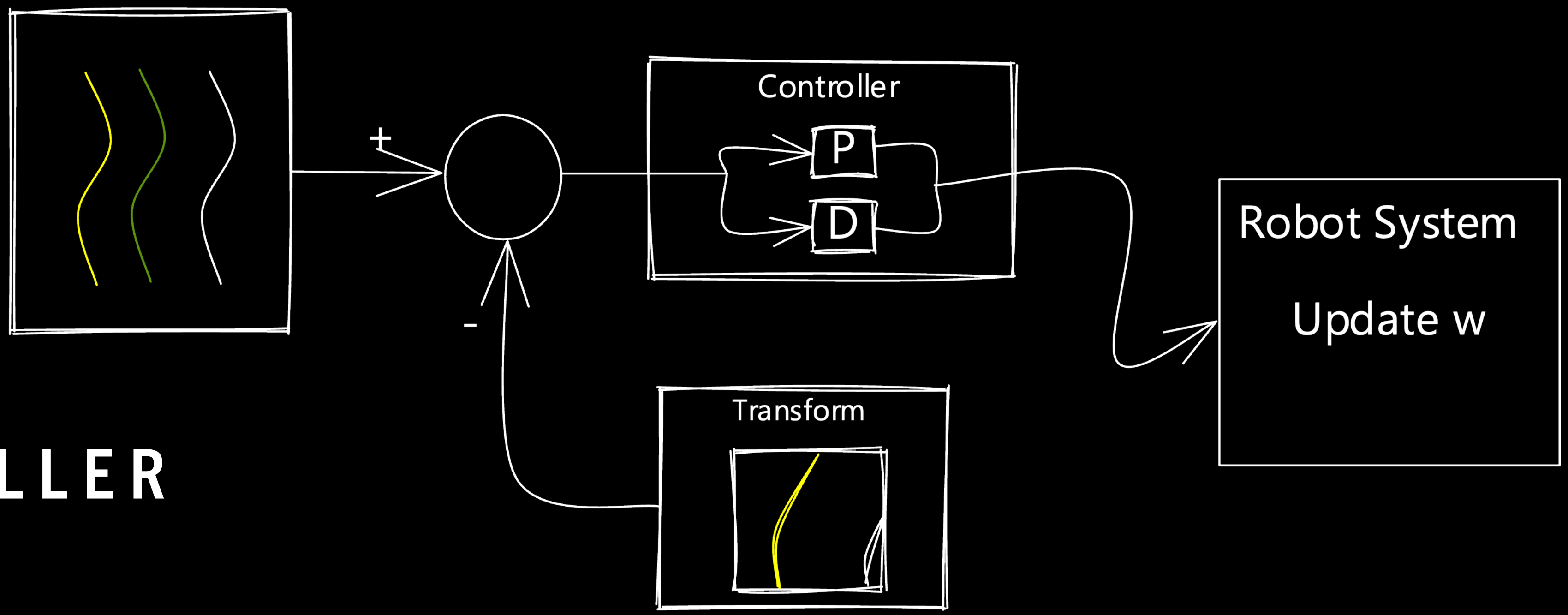
4. Get the franction number and the Mask of the lane

This procedure is produced for both yellow and white line

5. Return the centered lane according to the yellow and white lines

GRAPH OF CAMERA





LANE CONTROLLER

- It is based on PD (Proportional & Derivative) Controller
  - Stabilizes the oscilation
  - Reduces the influence of delay in vision processing
- Set the PD coefficients
- It works only for the Z- axis in order to manipulate the angular velocity
- The procedure is update every time when it gets response from the corresponding topic (/detect/lane)

DEMO

https://drive.google.com/file/d/18MEHNkZ5sAiVhV4xFWQW4ufk0lJ2UDF5/view?usp=sharing

LIMITATIONS

- Lighness and Noise:

  There is automatic method for lightness-adjustment which provokes wrong colour recognition

- Tunnel Problem

  Lack of lightness – Missing lines

THANK YOU

https://github.com/manoskout/robotics_project