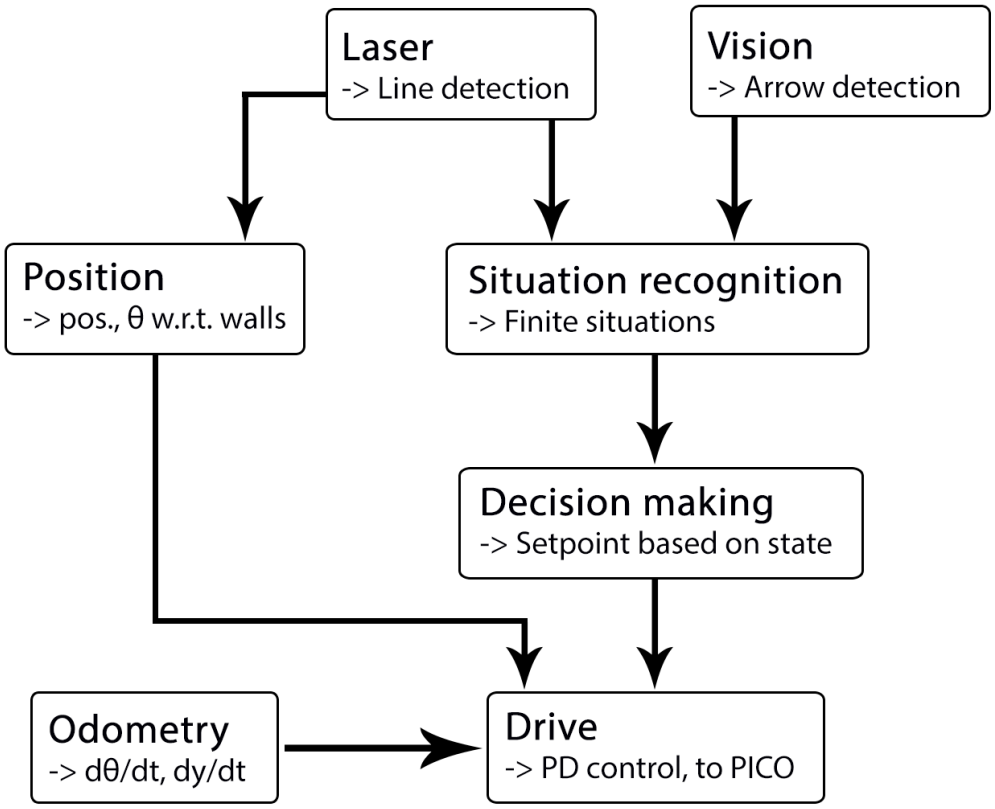
Progress EMC week 4

4K450 Embedded Motion Control
Groep 1

Technische Universiteit
Eindhoven
University of Technology

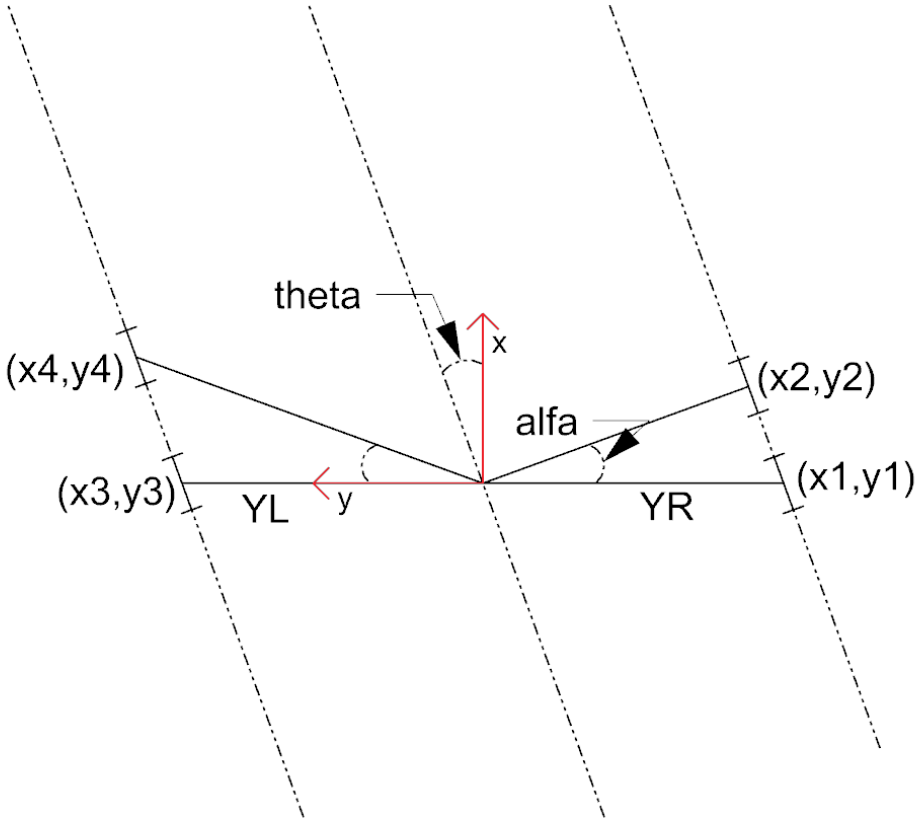
Where innovation starts

Software architectuur



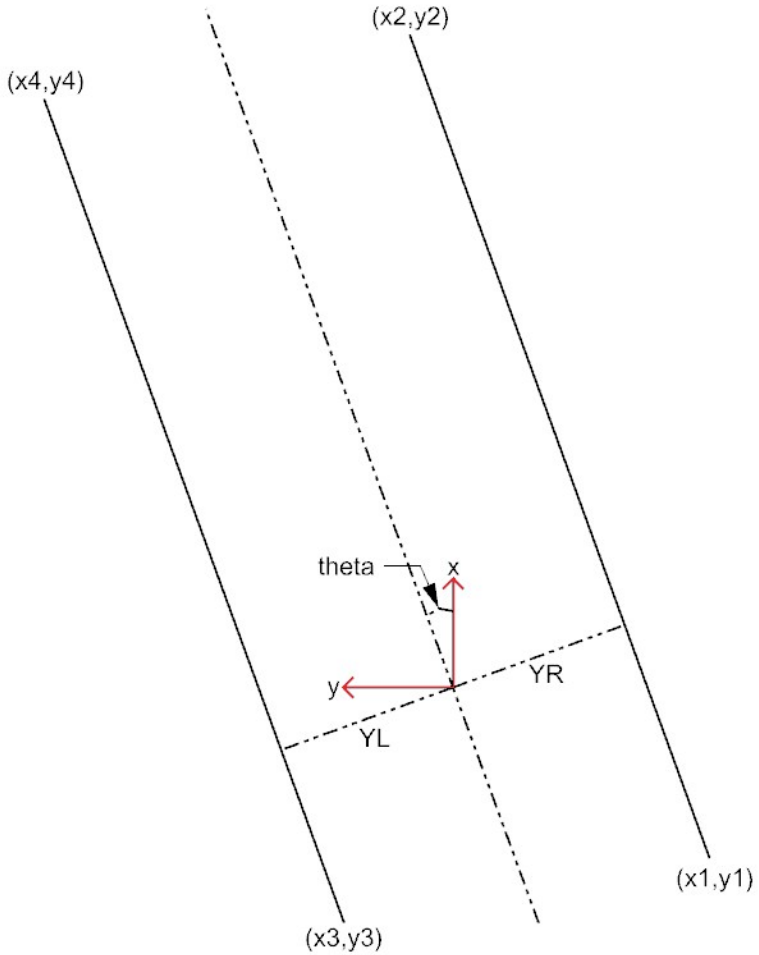
Relative distance (position)

- **Corridor challenge:**
  - **Parts of laser data used**



Relative distance (position)

- **Use lines: all laser data**



Relative distance (position)

- Theta on 1 or 2 walls
- YR and YL can be large (at junction)
- To do:
  - Publish and subscribe (message)
  - test

Line detection

- **Approach**
  - **Convert laserdata to pointCloud**
  - **Output (x,y,z) point**
    - **sensor_msgs/PointCloud Message**
      - Points are relative to base PICO
  - **Use Hough Transform**
  - **Input pointCloud**
  - **Output (x1,y1) (x2,y2)**
  - **Create a line**
  - **Input (x1,y1) (x2,y2)**
  - **Output matrix of lines**
    - **Use of visualization_msgs/Marker Message: Line Strips**
      - It will draw a line between every two consecutive points, so 0-1, 1-2, 2-3, 3-4, 4-5...
  - **Filter detected lines**

Line detection

Problems

- Hough transform is not working properly
- Error during rosrun

TODO

- Filter detected lines

Situation block

- Inputs: Linedata + Visual input
- Two approaches:
- Simple approach: Detect if Pico does not have a corridor to the left / right or in front using line data.

- Linedata is categorized into longitudinal lines and lateral lines by comparing the begin and en points of every line.
- Determine if corridors are outside range of pico.
- Output: Booleans: leftfree,rightfree,frontfree.

Situation block

- **“complicated” approach: Determine by analyzing the different lines on what kind of situation pico is.**

- **Inbetween two walls: two longitudinal lines detected**
- **Junction: two longitudinal lines, one lateral line detected. Compare x position of lateral line with x positions of longitudinal lines (see if connected or not) determine direction of junction.**
- **Etc…**

Decision making

- FSM package ‘decision_making’ available, but…
  - For ROS “hydro” & catkin
- Instead: custom-built lightweight FSM
- In FSM: event, state and transition declarations
- FSM runs as a node
  - Listens to events published by other nodes
  - Publishes state when a transition is made
- Controller runs as another node
  - Listens to state topic, calls a control function
  - Listens to sensor input, publishes bool signals (events)
  - Control functions can trigger an event upon completion

Drive

Input

- **Position:** $y_r, y_l,$ and θ
- **Odometry:** Quaternion
- **State:** $\dot{x}, \dot{\theta}$

- **Lasercallback - Safety**
- **Odometry – Quaternion to** roll**,** pitch**,** **yaw**
- **Drive**
  - $\dot{x}$ **- Based on state machine**
  - $\dot{y}$ **-** y_r y_l **with possibly a gain. In certain states 0.**
  - $\dot{\theta}$ **-** $\dot{\theta}$ **=/ 0 else** θ **with a gain**