Final Design & Considerations

Embedded Motion Control 2012

Group 7:

Siddhi Imming

Bart Moris

Roger Pouls

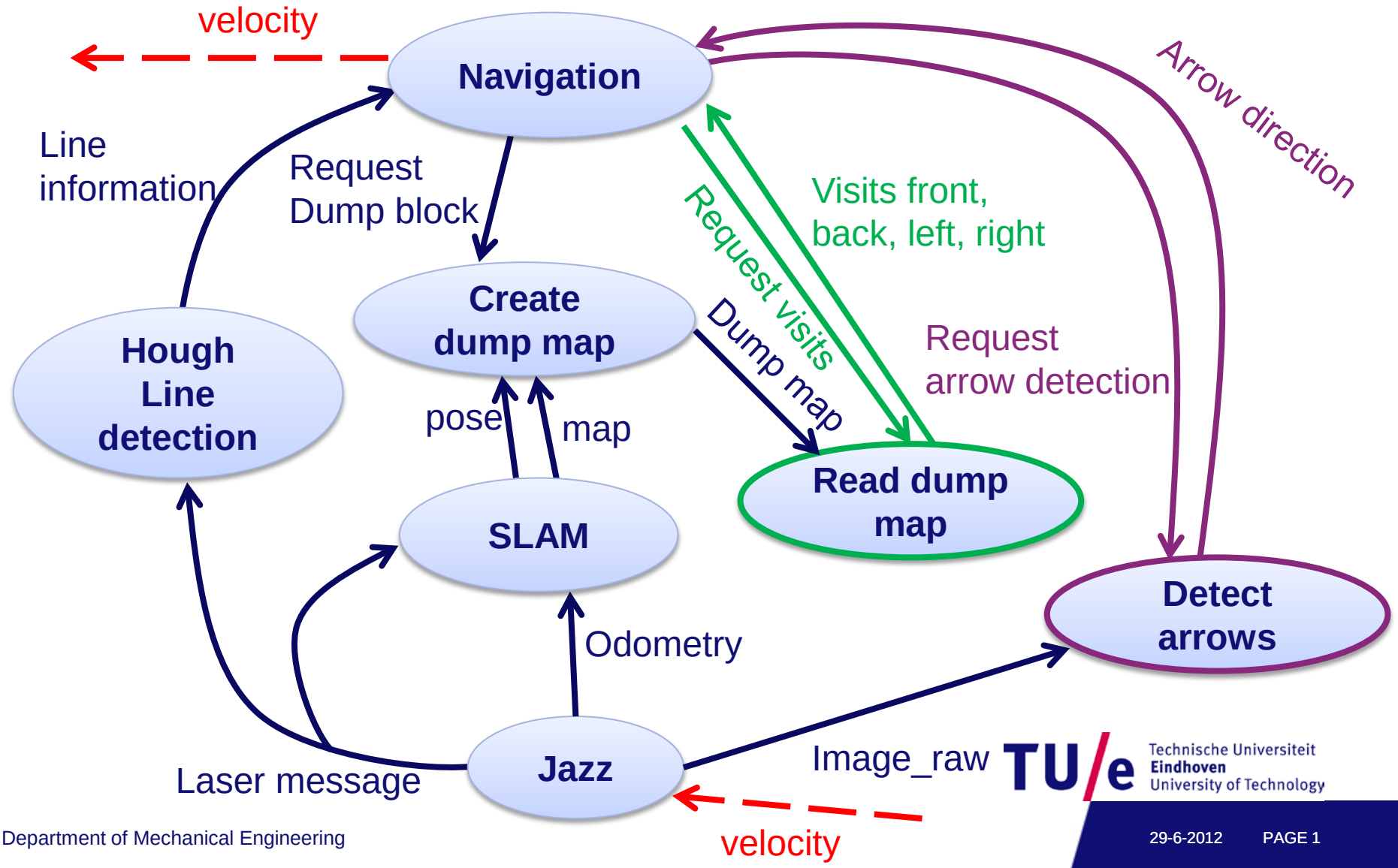
Patrick Vaes

TU/e Technische Universiteit Eindhoven University of Technology

Where innovation starts

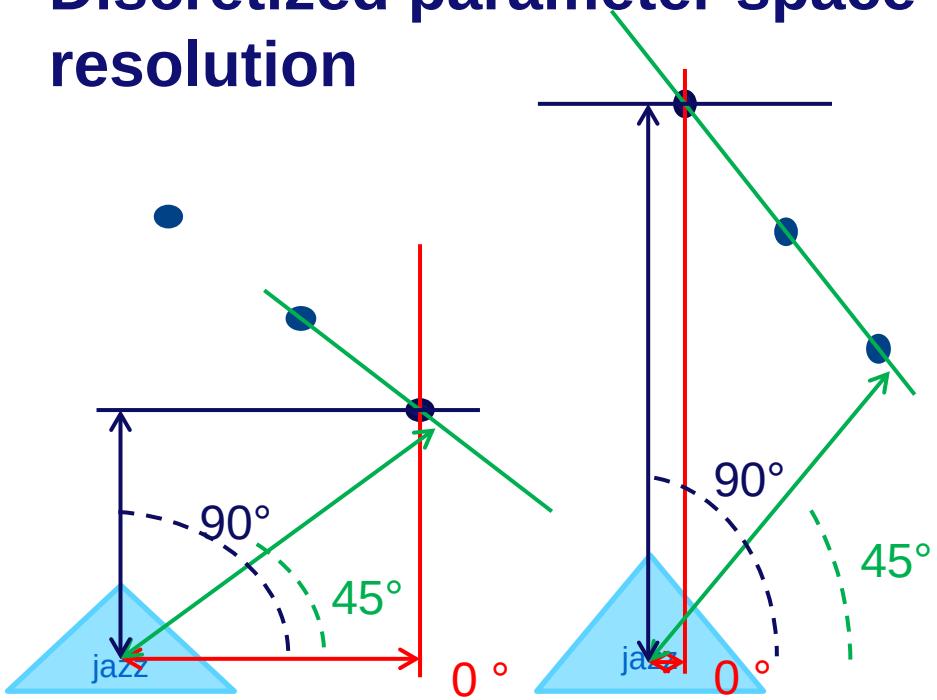
Eindhoven, June 29, 2012

Overview of the program structure for Tremaux's algorithm with arrow detection



Line detection

- **Line detection based on laser data and Hough Transform**
- **Discretized parameter space with appropriate resolution**



Angle	Point 1	Point 2	Point 3
0 °	5	3	1
45°	7	7	7
90°	5	7	10

Line detection

- Line detection based on laser data and Hough Transform
- Discretized parameter space with appropriate resolution
- Filter to find lines as local maxima in parameter space
- Appoint points to corresponding line

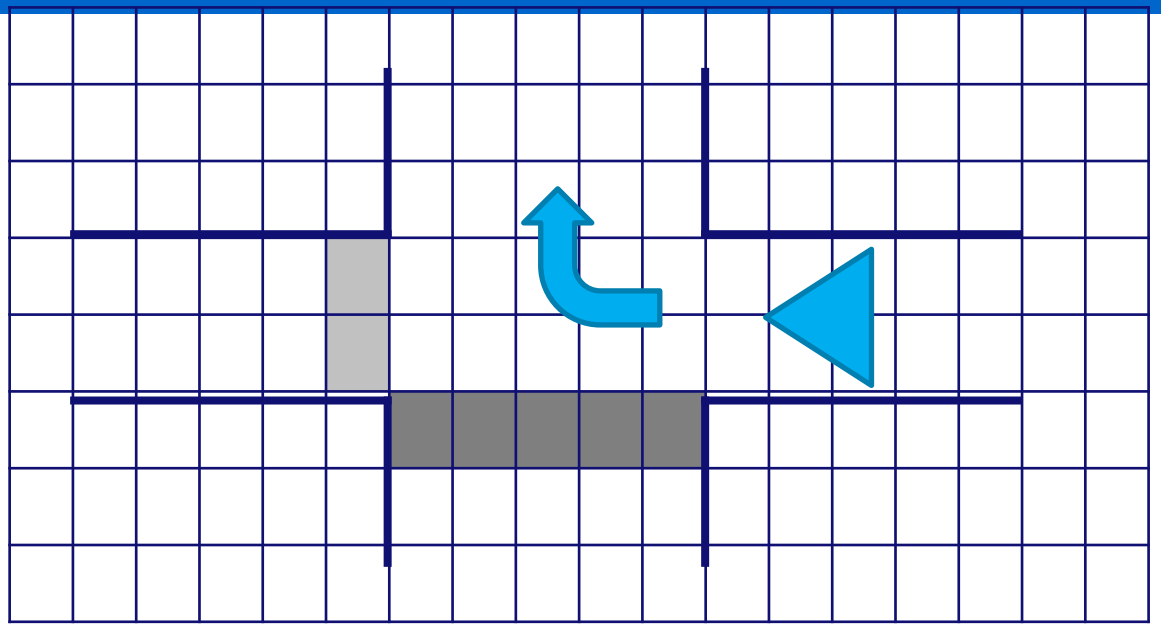
SLAM / Mapping

- Create map based on SLAM Gmapping
- Mapping based on re-observing landmarks and estimated positions based on laser and odometry data, weighted with Kalmann filter
- Proven that it works, and reuse of what is invented is the idea of ROS

Dump Map

- Overlay of original map
- Containing blocks with an integer indicating number of visits
- Based on synchronized original map and pose

SLAM / Mapping

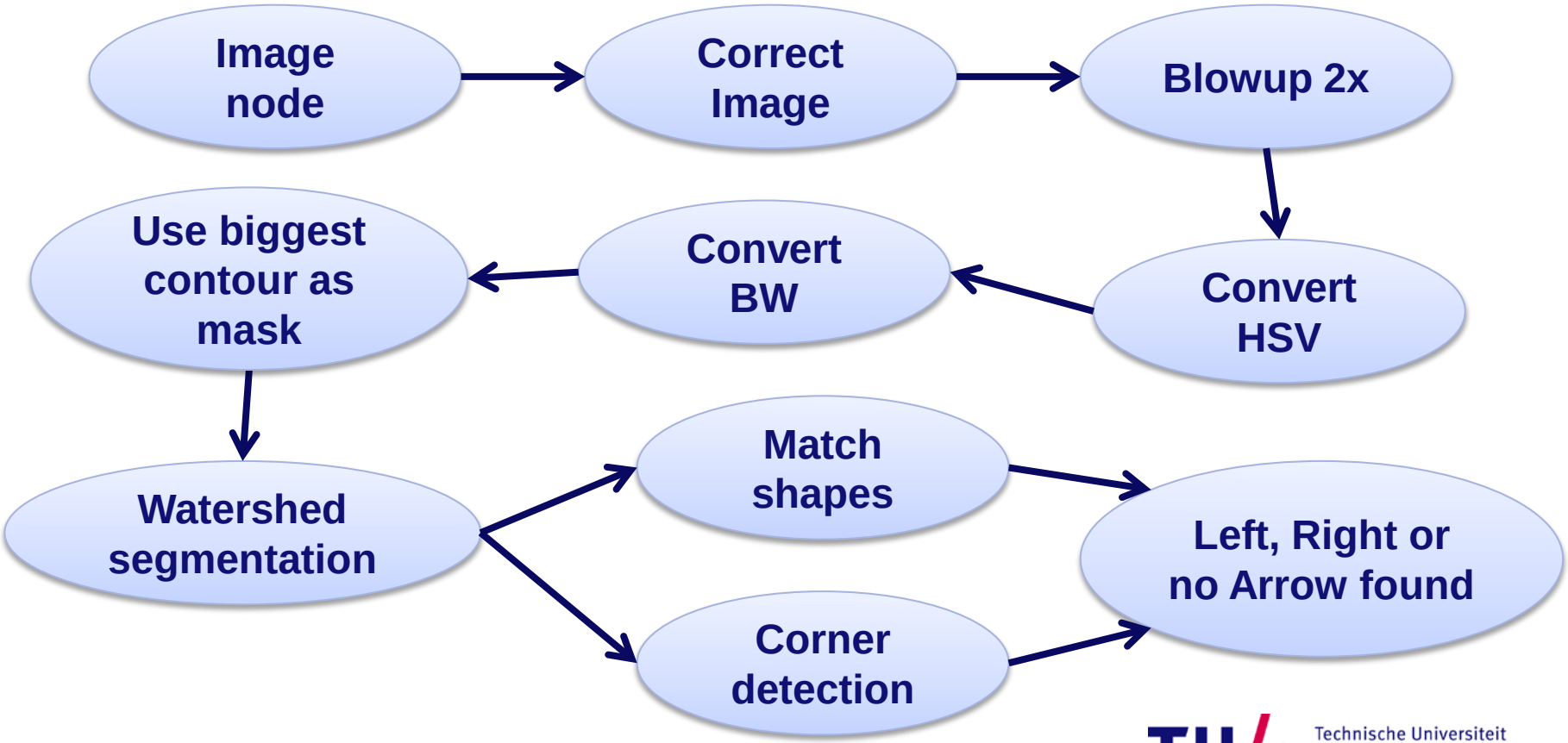


Dump Map

- **Overlay of original map**
- **Containing blocks with an integer indicating number of visits**
- **Based on synchronized original map and pose**

Arrow detection

- Overview



Arrow detection

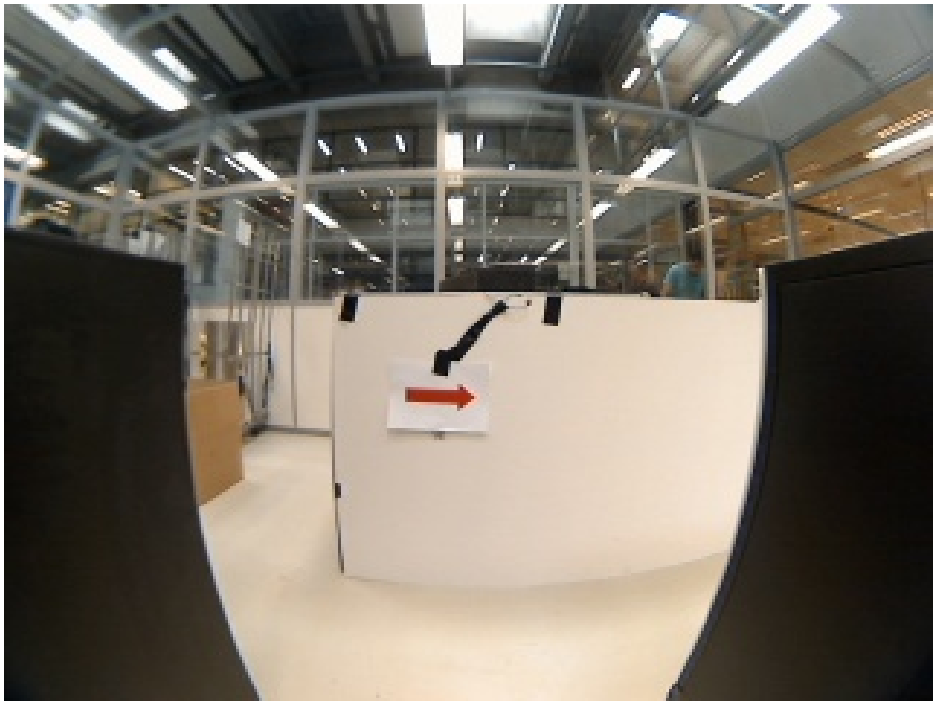
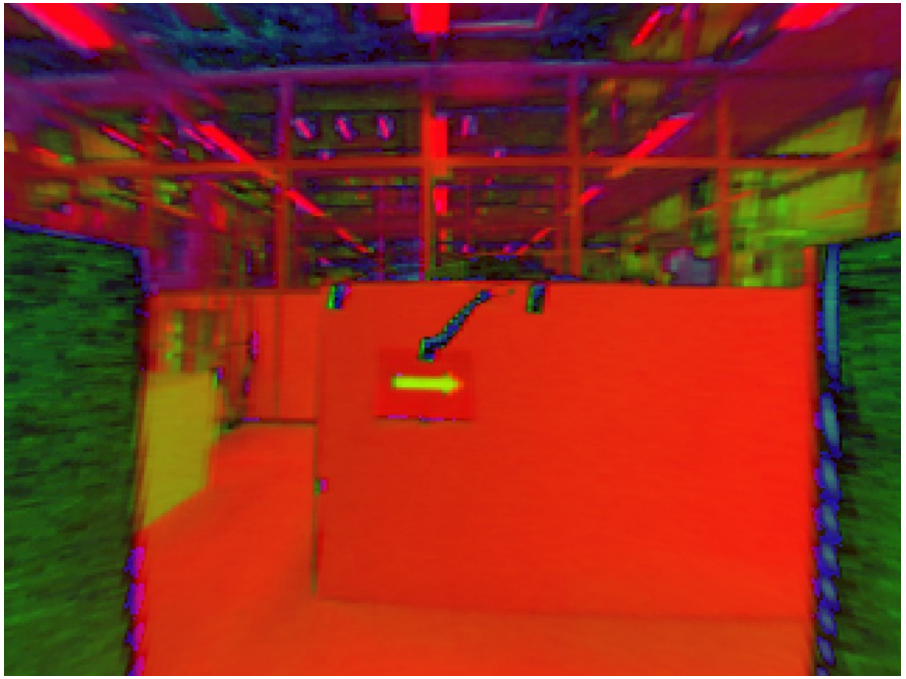


Image pico

Correct Image

Arrow detection

Blowup 2x



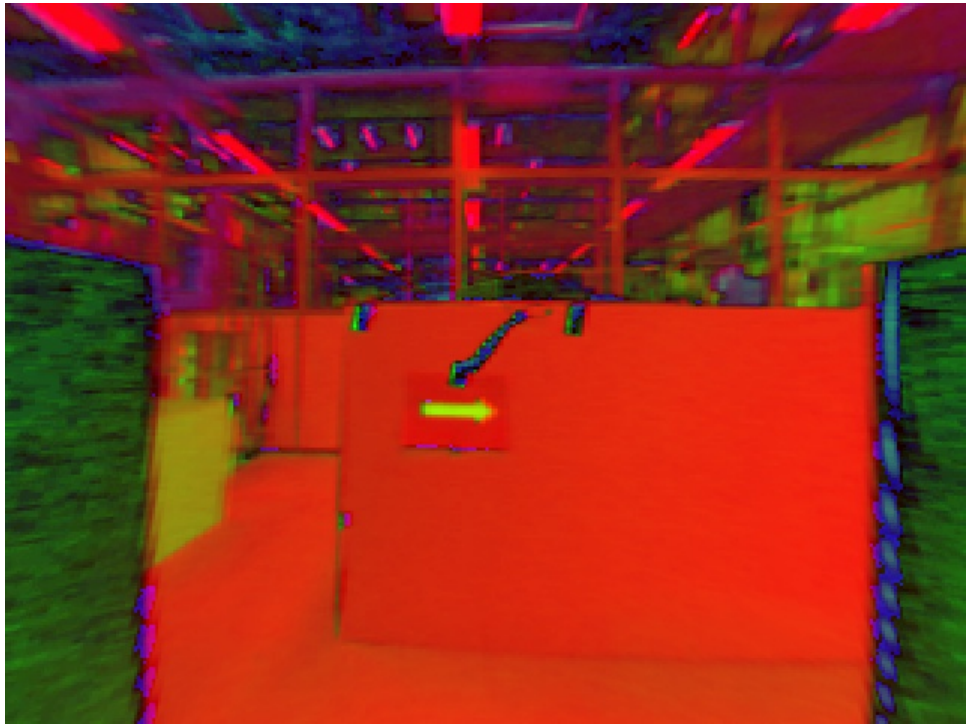
Convert HSV

Arrow detection

Correct Image

Blowup 2x

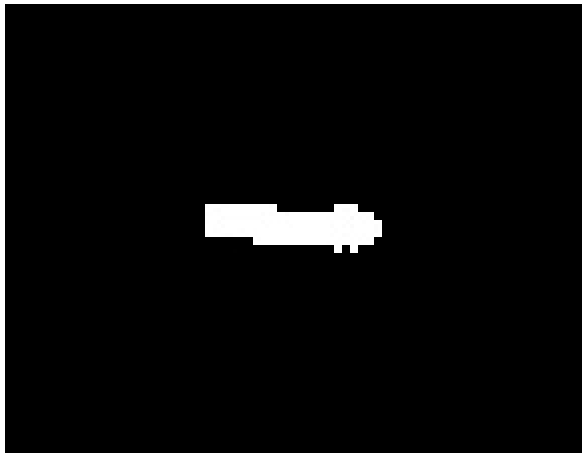
Arrow detection



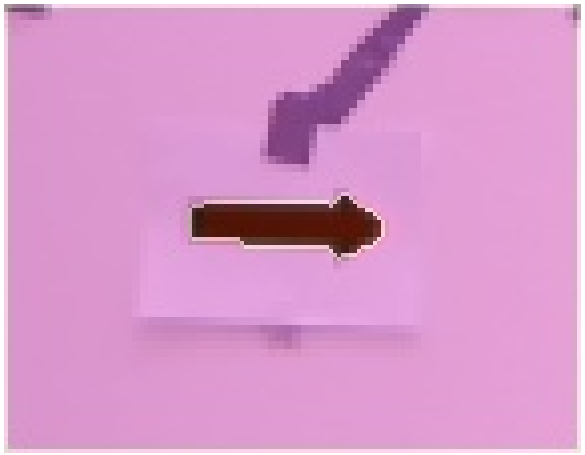
Convert HSV

Convert BW

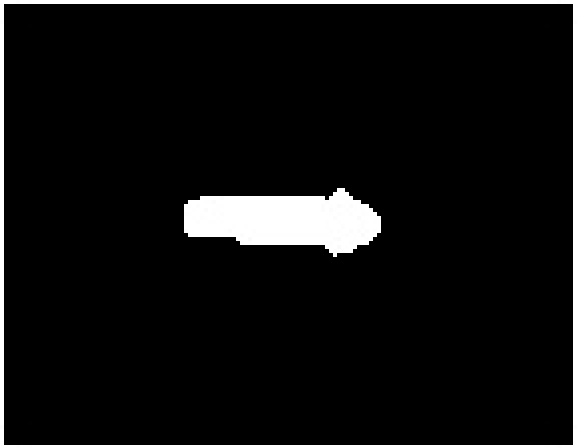
Arrow detection



Convert BW

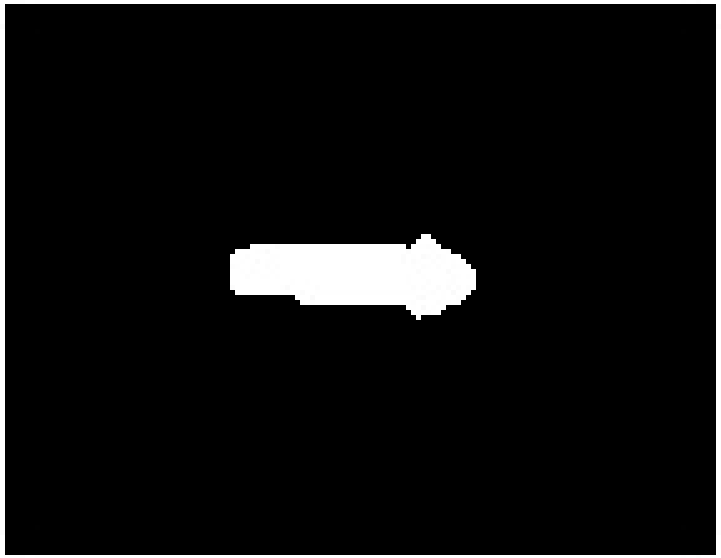


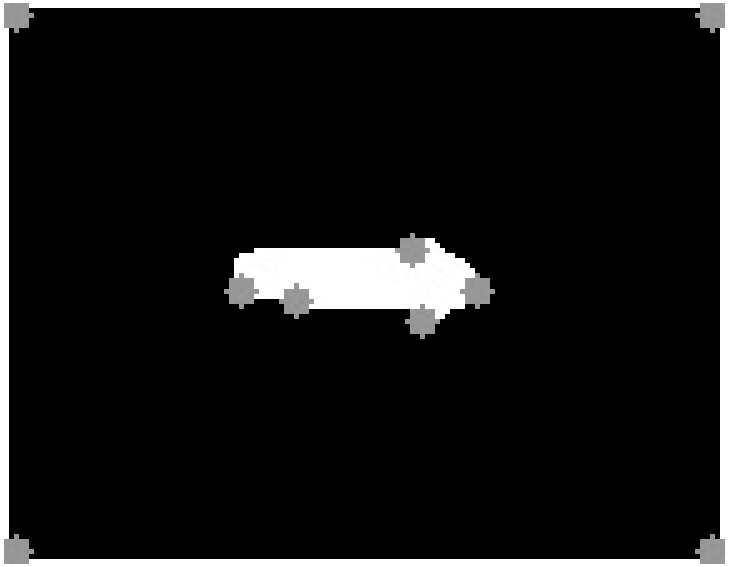
Watershed Segmentation color

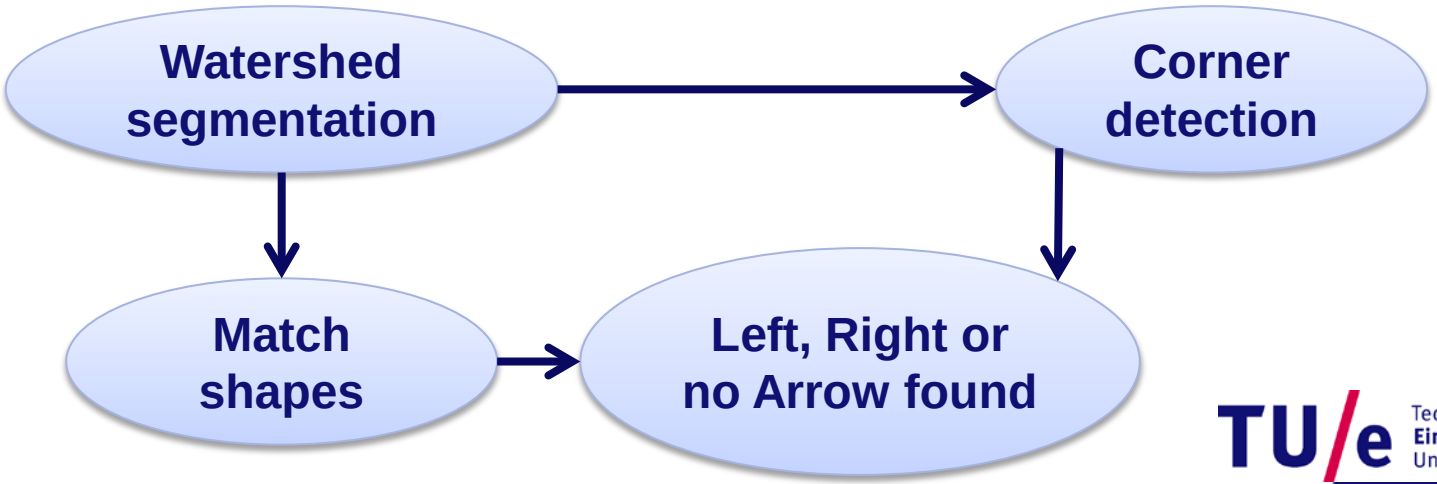


Watershed Segmentation BW

Arrow detection



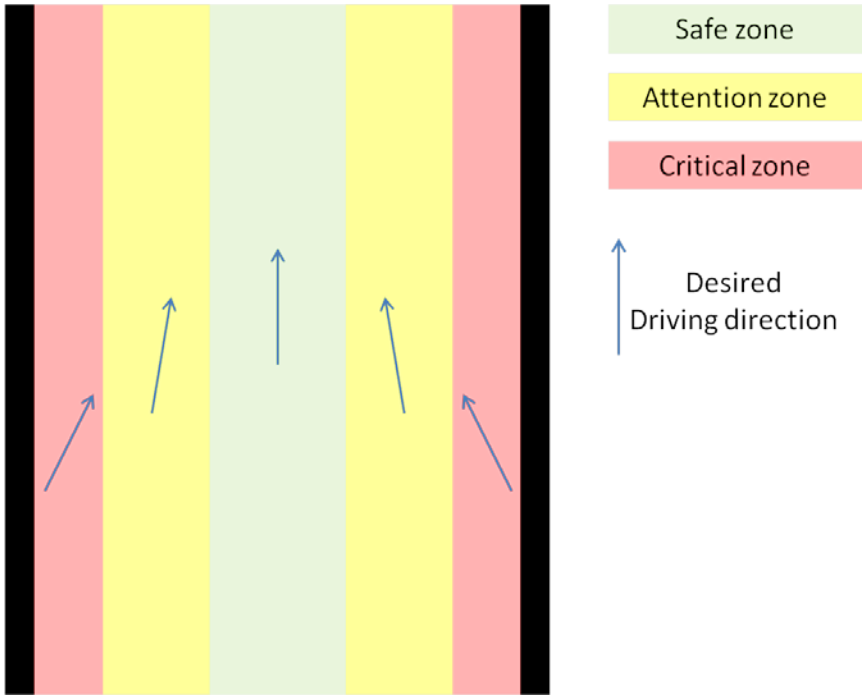




Navigation node

- Check first distance to parallel lines
- Check for line at 90 degrees to parallel lines and if we are close enough, half forward speed, determine exit 3 times, put dump, else drive parallel between lines
- Exit detection and handling
  - Detection of type via distance between points on line and position of the line relative to Jazz
  - If necessary, check for arrow
  - Request number of visits at each exit
  - Take decision, according to algorithm

Navigation node



- **Parallel driving**
  - **Corridor divided in several zones, with different correction speeds to enlarge safety**

Robustness measures

- **Several drive strategies implemented:**
  - **Left hand rule, right hand rule, Tremaux's rule with random and fixed order of preference, influenced by detected arrow**
- **Forward speed is maximal at the center of the corridor and is lower near a wall**
- **Angular corrections in a corridor are with a low angular speed to prevent 'overshoot'**
- **Initialization after startup and when no lines are detect for some time**
- **Entrance is blocked, using the dump map**