Embedded Motion Control Group 1

Members:	Wouter van Buul
	Marc Meijs
	Richard Treuren
	Sander Hoen
	Joep van Putten
Coach:	Sjoerd van den Dries

Technische Universiteit
Eindhoven
University of Technology

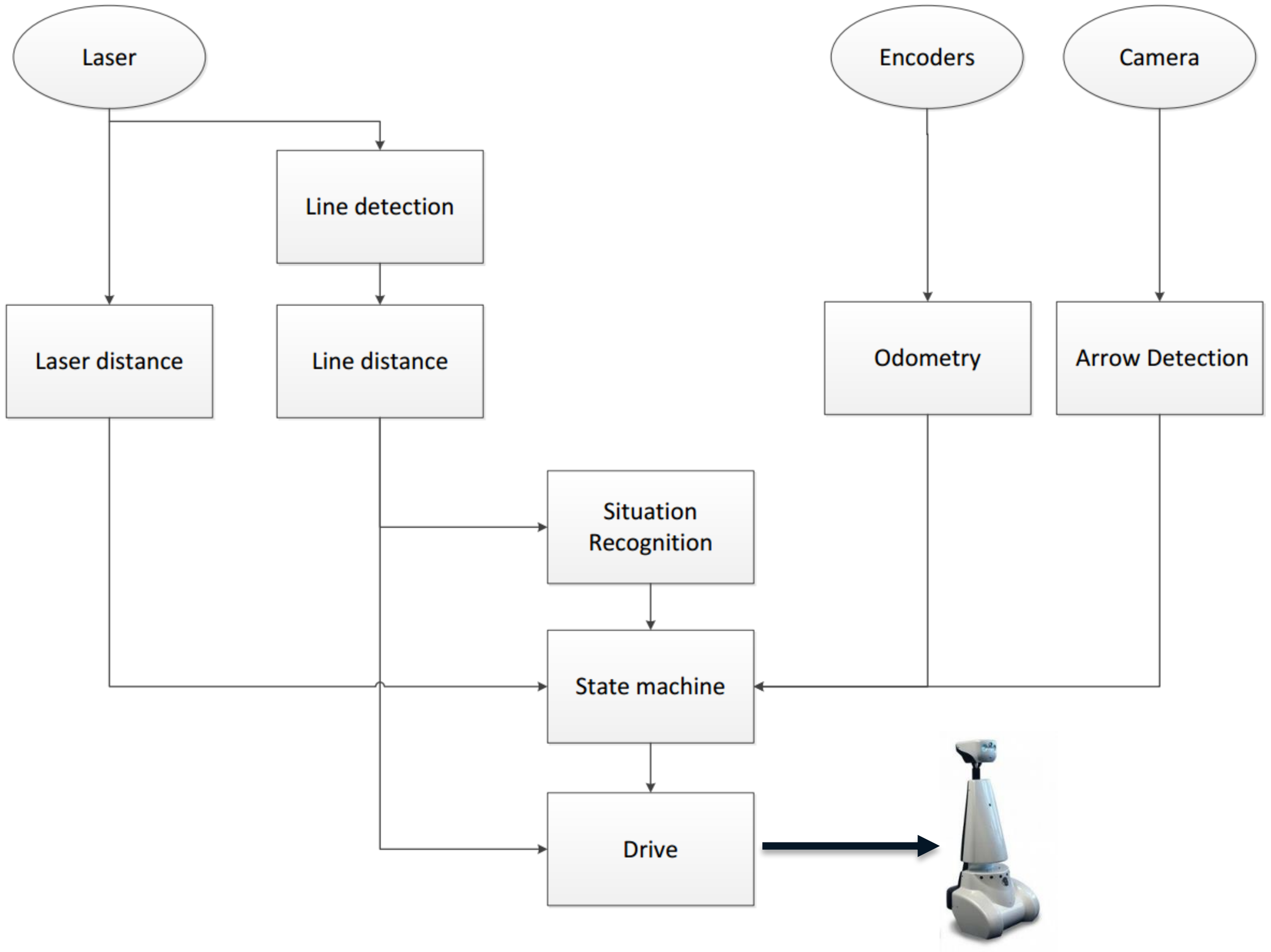
Where innovation starts

Strategy

- Maze solving algorithm: the wall follower
  - Situation recognition
  - Arrow detection

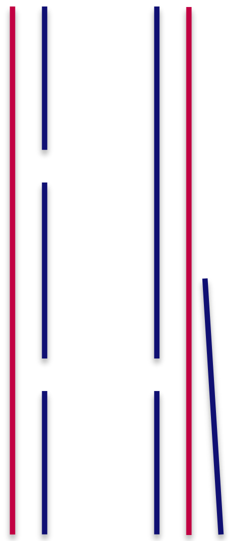
- Modular software design
  - Effective and easy to tweak
  - Start with simple, functioning software, then add more sophisticated ‘blocks’ to improve performance

Software architecture



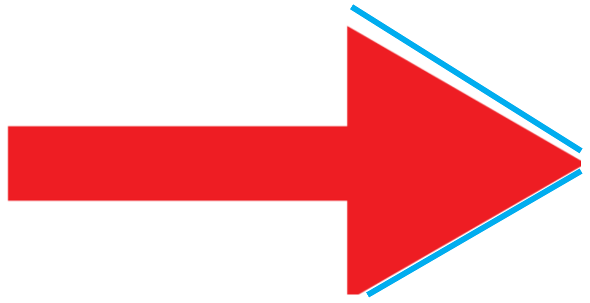
Line detection

- **Goal**
  - Detect walls
- **Input**
  - Laser data
- **Approach**
  - Hough transform
  - Custom line filter
    - Merges dublicate lines
- **Output**
  - Matrix with detected lines (x1,y1) – (x2,y2)



Arrow detection

- **Goal**
  - Detect left/right arrows in the maze
- **Input**
  - Image of camera pico
- **Approach**
  - Edge detection
    - Hough transform, custom filter
    - Detect if \ is above / or vice versa
    - Arrow has to be detected 3 times
    - GUI to tune the color red
    - Feature detection
  - Template matching
- **Output**
  - Boolean 'arrow left/right'



Technische Universiteit Eindhoven University of Technology

Situation recognition

- **Goal**
  - Determine maze ‘situation’
- **Input**
  - Detected lines
- **Approach**
  - KISS (Keep It Simple, Stupid)
    - Detect only what you need, when you need it
  - Cluster lines
- **Output**
  - Available exit left or right
    - If exit is a dead end? → No exit detected

State machine (decision making)

- **Goal**
  - Determine PICO behavior
- **Input**
  - Situation
  - Arrow detection
- **Approach**
  - Modular design
  - Plug&play
  - Custom-written **FSM** class
- **Output**
  - Drive left, drive right, drive straightforward, etc

Conclusion

- **Fast maze solving PICO**
  - Modular design
  - Plug&play state machine
  - Robust wall/arrow detection
  - Dead end recognition