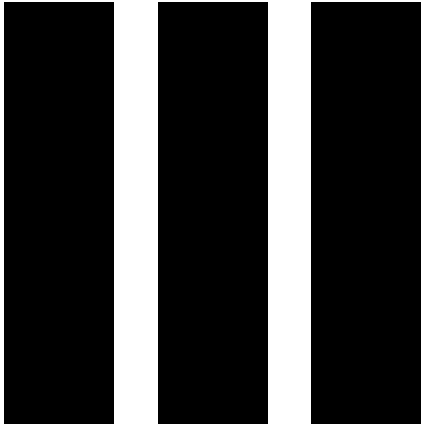
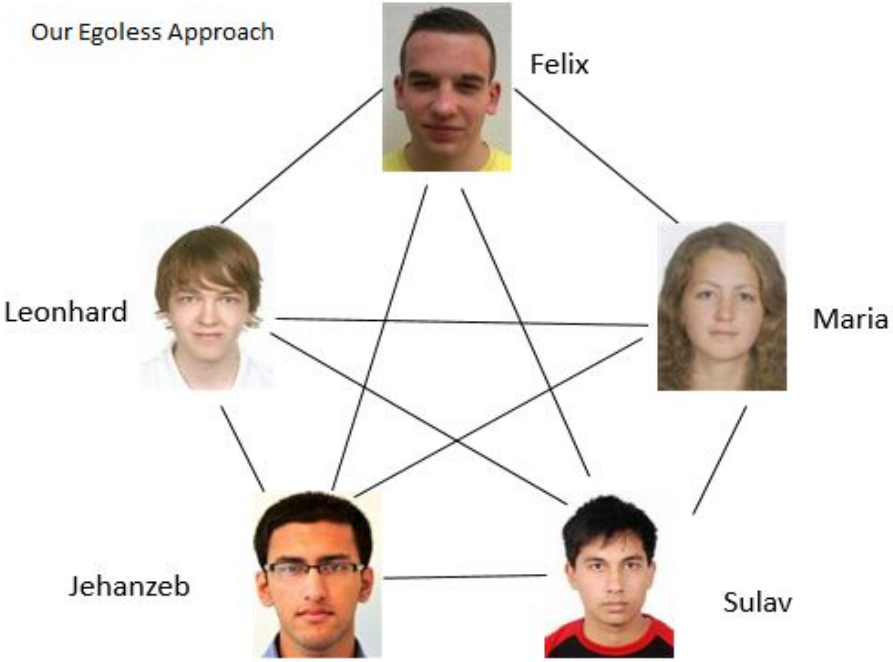


Outline

- Team
  - Team Organization
  - Different Roles
- Approach
  - Schedule
  - Algorithms
  - Organization
- Conclusion
  - Next Steps

Team Organization

Who are we?



Team management and roles

- Scrum
  - Felix is Scrum master and the product owner.
  - Weekly Scrum meetings on Tuesdays
  - Additional meetings whenever necessary
  - Pair programming

- How do we manage?
  - Everyone is responsible
  - Regular discussion about project on Facebook
  - Shared Google docs

Scrum(continued…)

- Product backlog
  - Turning, moving
  - Half circle detection
  - Random walk
  - Wall detection
  - Wall avoidance
  - Wall following
  - Integration
  - Calibration
  - SLAM
  - Integration

1st Sprint

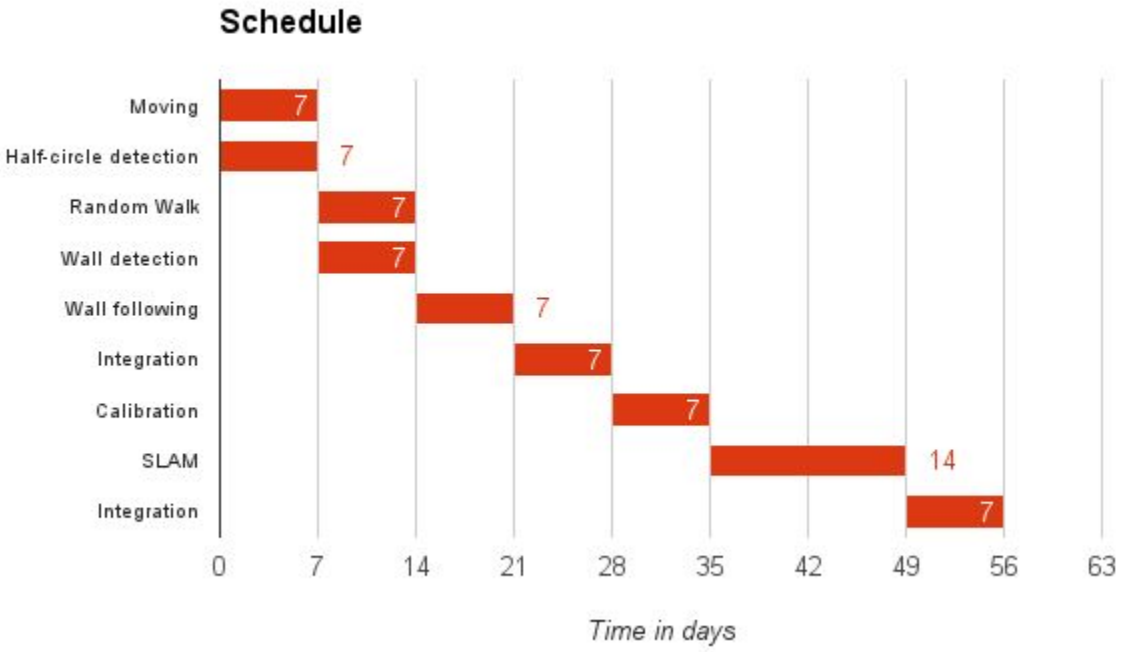
Milestones

- Random Walk
- Wall-following
- SLAM

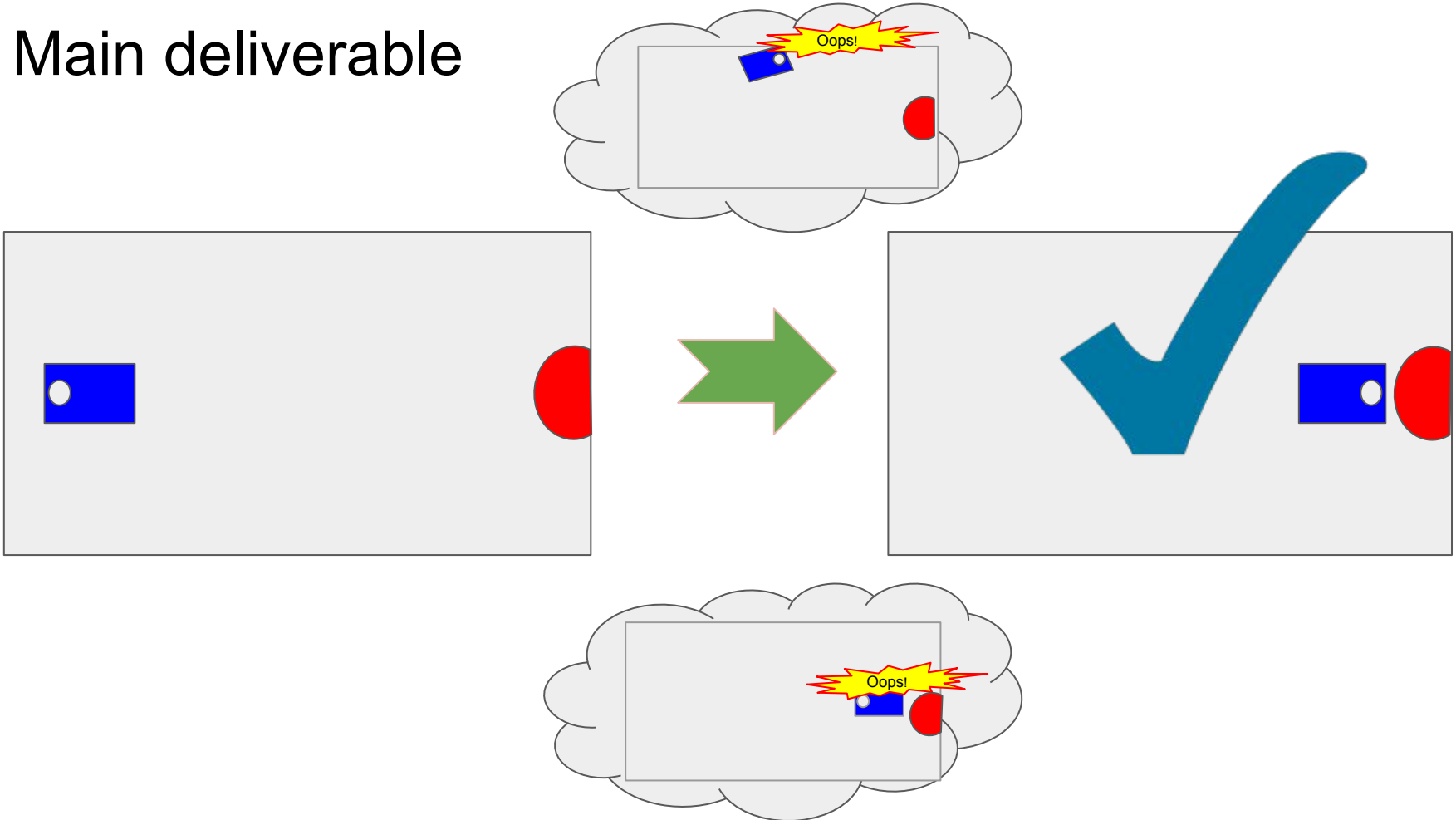
Work Packages

- Half-circle detection
- Moving/calibration(especially for real robot)
- Random Walk
  - decide on which direction to take
- Wall-following
- Wall detection
  - Collision avoidance

GANTT

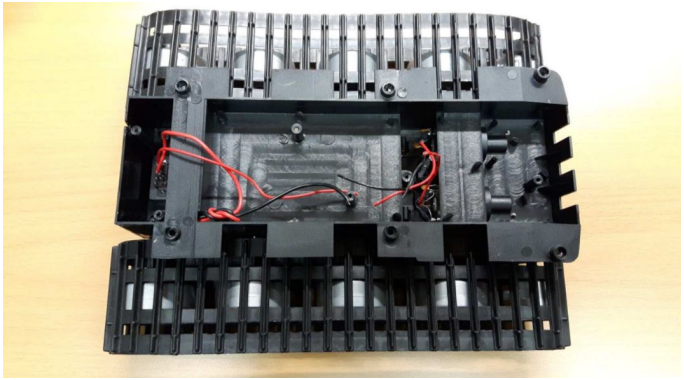

Schedule
Moving
Half-circle detection
Random Walk
Wall detection
Wall following
Integration
Calibration
SLAM
Integration
7
7
7
7
7
7
7
14
7
0
7
14
21
28
35
42
49
56
63
Time in days

Main deliverable



Driving Robot

Controller



Model

Sensors

Source: Professor Andreas Birk(2016).Software Engineering Lab [SoftwareEngineering16-P01-Intro, slide 10-11]

Approaches

Instead of building one complicated controller - use of behaviors (divide-and-conquer)

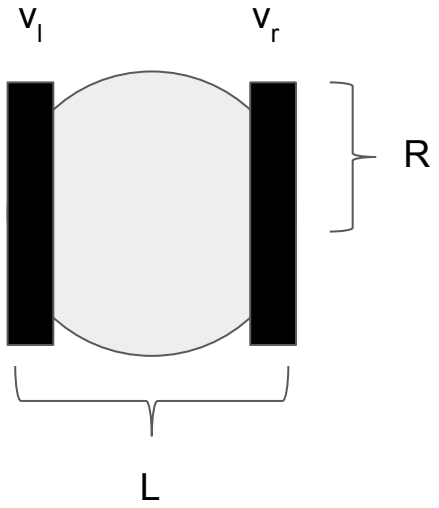
- Go-to-goal
- Avoid-obstacles
- Follow-wall
  - allows to cover ground that may be missed by wandering around a room (aka random walk)

Behavior-based approach - switch among controllers in response to environmental changes

- avoiding an obstacle or driving to the target?

Source: https://roachnet.wordpress.com/2009/06/19/8/

A bit of Maths and Physics in the basic motion of the robot



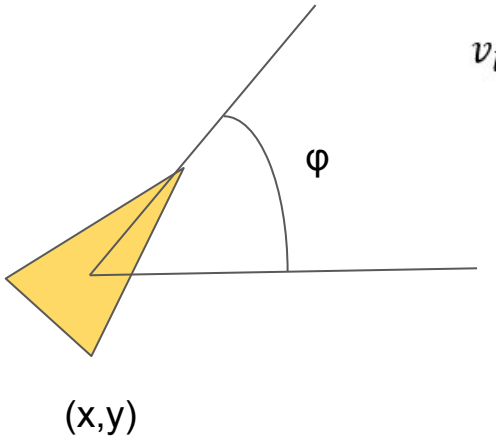
$$\dot{x} = \frac{R}{2}(v_r + v_l)\cos\varphi$$

$$\dot{y} = \frac{R}{2}(v_r + v_l)\sin\varphi$$

$$\dot{\varphi} = \frac{R}{L}(v_r - v_l)$$

$$v_r = \frac{2v + wL}{2R}$$

$$v_l = \frac{2v - wL}{2R}$$



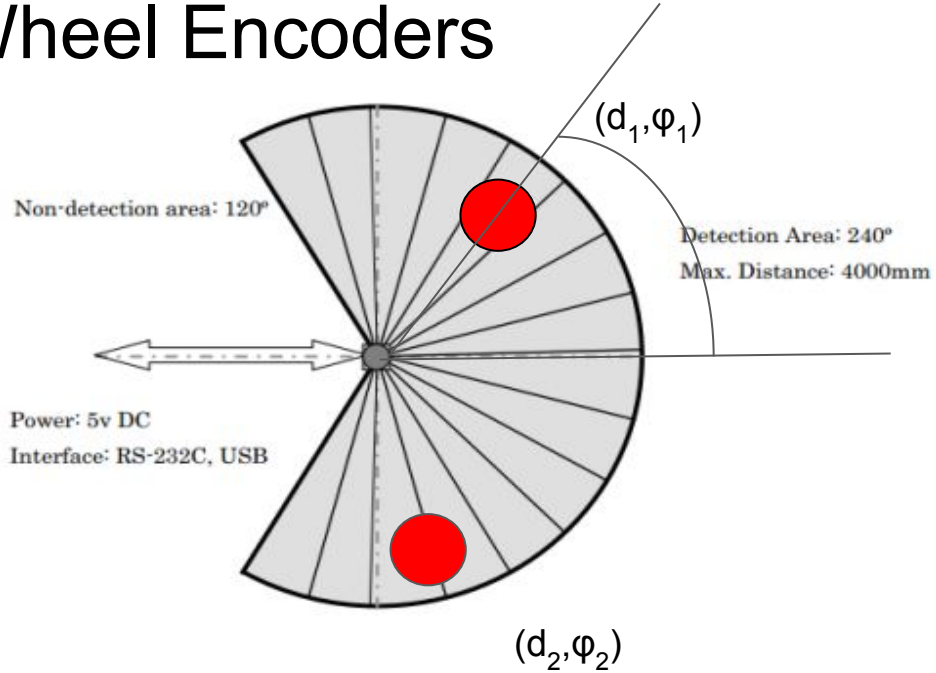
Initialization: $(x_0, y_0) = (0,0)$
v,w - designed values
v_r, v_l - actual inputs

Source: Magnus Egerstedt, Control of Mobile Robots, Georgia Institute of Technology

Odometry - Sensors and Wheel Encoders

$$x_1 = x + d_1 \cos(\varphi_1 + \varphi)$$

$$y_1 = y + d_1 \sin(\varphi_1 + \varphi)$$



- Wheel encoders (calculations of distance, new position of the robot)
  - tick count
  - problem - slipping => not 100% reliable

Source: http://www.hizook.com/projects/hokuyo-urg-04lx-laser-rangefinder-lidar
http://www.generationrobots.com/fr/content/52-se-localiser-avec-un-telemetre-laser-hokuyo

Algorithms and packages

Sensors:

- **laser_filters** package to operate on 2D planar laser scanner
  - processing *sensors_msgs/LaserScan* messages
  - LaserScanRangeFilter, LaserArrayFilter
- **pluginlib** package to writing and dynamically loading plugins using the ROS build infrastructure

Source: http://wiki.ros.org

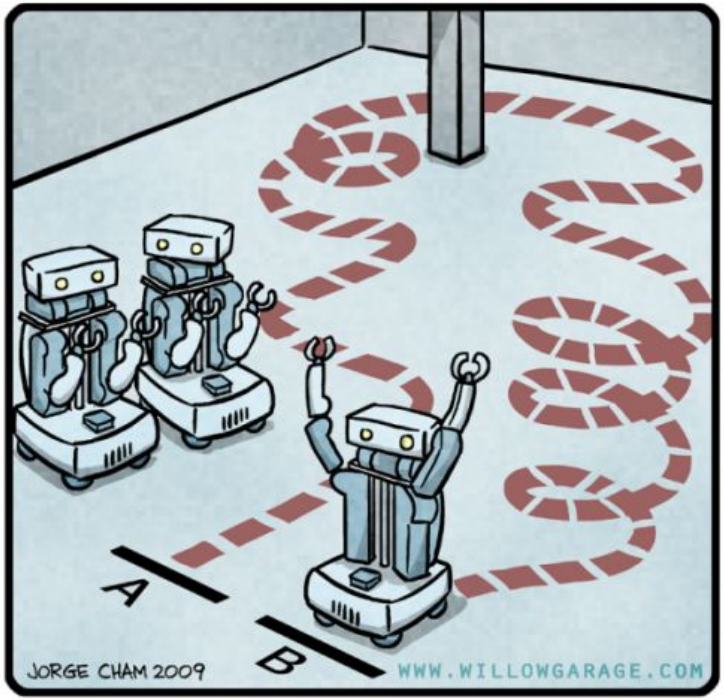
Algorithms and packages

Detection of features in environment using OpenCV:

- `void Canny(InputArray, OutputArray, double, double, int, bool);` - detection of edges
  - Too small threshold —gets lots of noise, fat edges
  - Too big threshold —loss of sections of edge
- `void cornerSubPix(InputArray, InputOutputArray, Size, Size, TermCriteria);` - detection of corners
- `void HoughCircles(InputArray, OutputArray, int, double, double, double double, int, int);` - detection of circles
- `void HoughLines(InputArray, OutputArray, double, double, int, double, double);`- detection of lines (aka walls)

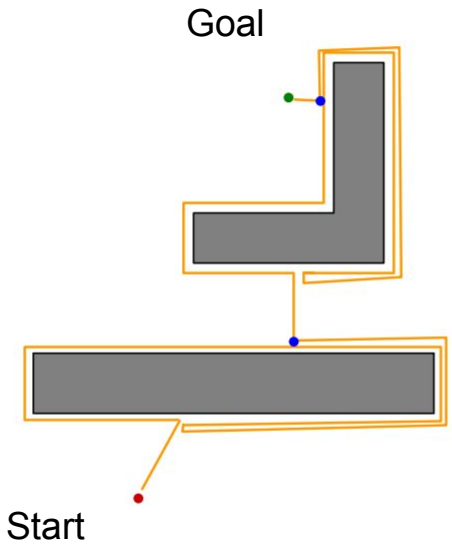
MIT(2005). Mobile Autonomous Systems Laboratory[Lecture slides]. Retrieved from: http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-186-mobile-autonomous-systems-laboratory-january-iap-2005/
http://docs.opencv.org/2.4/modules/imgproc/doc/feature_detection.html?highlight=houghcircles

R.O.B.O.T. Comics



"HIS PATH-PLANNING MAY BE SUB-OPTIMAL, BUT IT'S GOT FLAIR."

Algorithms and packages



Motion planning

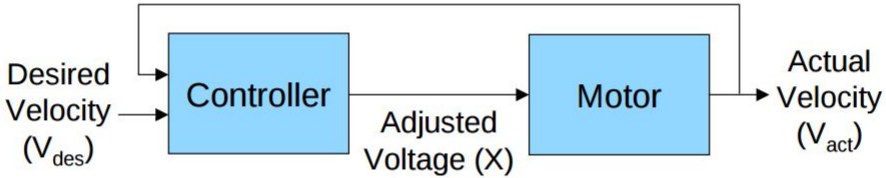
- “Bug”-approach
  - Wall-,curve-tracing
  - Advantage: faster than random walk approach
  - Problem: unknown goal

Approach(corrected for our project):

1. ~~head towards goal~~ detect the nearest obstacle, head towards it
2. if an obstacle is encountered, circumnavigate it and remember how close you get to the goal
3. return to that closest point (by wall-following) and continue

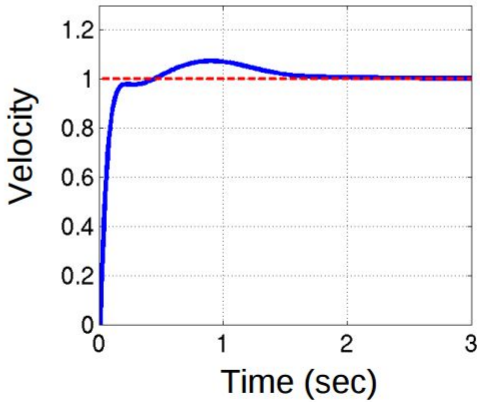
Choset, H. (2010). Robotic Motion Planning [Lecture slides]. Retrieved from http://www.cs.cmu.edu/~motionplanning/lecture/lecture.html
Plaku,E.(2014). Introduction to Robotics[Lecture slides].Retrieved from http://faculty.cua.edu/plaku

PID controller to set motor velocity



- Situation of detecting a goal while moving => change the direction + driving towards it
  - problem with error term and angles range
  - problem with overshooting the goal, circling around it, drifting

$$X = V_{des} + K_P e(t) + K_I \int e(t)\, dt - K_D \frac{de(t)}{dt}$$



MIT(2005). Mobile Autonomous Systems Laboratory[Lecture slides]. Retrieved from: http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-186-mobile-autonomous-systems-laboratory-january-iap-2005/

Organization

UML Diagram

Packaging

- Pre-built packages
  - hokuyo-node or urg_node
  - teleop_twist_keyboard
  - rviz
- ROS packages we will create
  - wall-finder
  - circle-finder
  - random-walk-strategy
  - wall-follow-strategy
  - optionally: mapping-strategy

Testing

- Strategies
  - Unit testing
  - Regression testing
  - (possibly) automated simulator
  - (possibly) CI with GitHub
- Implementation
  - CppUnit or Google Test
  - scripts for automated simulator testing

APERTURE LABORATORIES

Documentation

Doxygen:

- Wide range of Programming Languages including C++
- Extract documentation from source code
- HTML, PDF and LaTex output
- Documentation within code
- Cross reference between code and documentation

Conclusion

Next

- Incorporate feedback into strategy
- Set up environment
  - testing
  - formatter
- First Scrum-Sprint next week
  - Distribution of tasks
  - Form teams