Full_coverage_path_planner
&
Tracking_pid

ROSIN: Experiences and outcomes
• Nobleo: consulting & projects in high-tech
  • Robotics, mechatronics, embedded SW, computer vision
  • ASML, NXP, Bosch Rexroth, Ultimaker, Heineken

• Loy van Beek (loy.van.beek@nobleo.nl)
  • At Nobleo since 2015, Nobleo Projects since 2018
  • RoboCup@Home with TechUnited Eindhoven since 2011
Full coverage path planning

• Needed for cleaning, painting, inspection, ...
• No FCPPs available in ROS-Industrial
  • Auburn University’s automow?
    • 7 years since last commit
    • No docs, written for a competition
  • Vector represented maps
• In Python, using Shapely library
Full coverage path planning

• ROSIN, stage 1: Minimal Viable Product
  • Requirements:
    • Map: known OccupancyGrid
    • Dynamic Obstacles: handled by local planner
  • Nobleo developed a global planner & local planner to accurately follow global path
  • Most work done by colleagues:
    • Yury Brodskiy: algorithm implementation
    • Michiel Franke: controller
  • My work: refactoring, MoveBase-plugins & testing
Full coverage path planning

• Backtracking Spiral Algorithm
  • Enrique González et al.
• Map is discretized by coverage tool size
• Core separated from ROS
  • Easier testing
  • Easier port to ROS 2 later
• Wrapped as nav_core::BaseGlobalPlanner
  • Goal is ignored, serves as trigger only
Full coverage path planning
- Local planners take shortcuts on coverage paths
• Accurately follow path, reproducible
• Do not deviate from path!
• Pause for obstacles
• Do not go to end ASAP!
• PID controller on X and Y error
• wrt. carrot moving at fixed speed
• nav_core::BaseLocalPlanner, core ROS-independent
- Tuning via dynamic_reconfigure
- Debug intermediate P, I, D outputs
Applications

- Tracking_pid compensates for wheel shear by rotating robot
Applications

Autonomous Harbor Cleaning: RanMarine WasteShark
- Tracking pid as virtual anchor
• Refactored original code for test-ability
  • ROS-independent parts separated
• Testing
  • Unittests with GTest
  • Coverage algorithm is tested against 1000 random maps
    • Expected result: determined by OpenCV FloodFill
  • Tests run in BitBucket with ROS-Industrial CI
    • PR accepted to make script provided for BitBucket executable: 100644 → 100755
• Great initiative
  • Requires to make in-house SW more generic, thorough
• TODO beyond MVP
  • Adhere to vehicle constraints
  • Different coverage algorithms
  • Property-based testing
  • Generate more test-cases
  • E.g. use CppQuickCheck/QuickCheck++/RapidCheck