Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences

b-it Bonn-Aachen International Center for Information Technology

Software Development Project (SDP)

Summer 2022

Learning Objectives

- Develop (create & update) requirement specifications with coaches
- Create and assess software designs for a given application problem
- Depending on the specific project:
  - Gain practical understanding of scientific literature
  - Implement and/or improve existing implementations of advanced methods
- Present advanced concepts in a meaningful way
- Demonstrate software on physical robot platform
- Collaborate with fellow students and coaches

Overall Process

Communication & Collaboration!

General rules

- Each group is composed of max 4 members
  - Group selection under “Project Booking”, first person to book get to choose teammates
  - GitHub repository will be provided after groups are decided, please fill the “SDP Projects: Team Members” Wiki
- Each group has a “coach”, depending on the project
- Group needs to establish **frequent communication** with their coach

Communication means

- Dedicated slack workspace: https://massdp.slack.com/
  - Sign-up link: https://join.slack.com/t/massdp/signup
- Default online meeting location: https://h-brs.webex.com/meet/minh.nguyen

Grading of deliverables

- D1-D3, D4.3: coach’s evaluation (70%)
- D4.1, D4.2: evaluation by coach + “external jury” (30%)

Deliverables

D1 Initial Presentation

- Project introduction, e.g. problem definition, project goals, required libraries,…
- Requirements specifications
  - Format of requirement specs up to coach, a common one is user story (example last slide)
  - Coach may specify non-functional requirements, e.g. documentation, code quality, as necessary
- Collaboration plans, e.g. scheduling, version control, communication tool,…

D2 Revised Presentation

- Revise and update requirements
- Steps towards Minimum Viable Prototype (MVP)
- Incorporate any other feedback from first presentation as appropriate

D3 Midterm Demonstration

- Present:
  - Recap of goals & requirements
  - Features/results included in MVP
  - Any *notable* design decisions
  - Any necessary revisions to requirements specs
  - Next steps towards the final software
- Demonstrate MVP

D4 Final Demonstration, Presentation & SW

- (D4.1) Present:
  - Project recap for external audiences
  - Final features/results
- (D4.2) Demonstrate final SW on physical platform
- (D4.3) Software with documentation

Timeline

Date	Deliverables
25.04.	D1 Initial Presentation
09.05.	D2 Revised Presentation
06.06.	D3 Midterm Demonstration
12.09. (2nd examination period)	D4 Final Demonstration and Presentation, Final Software

Projects

People Detection for the Kelo ROBILE

Overview

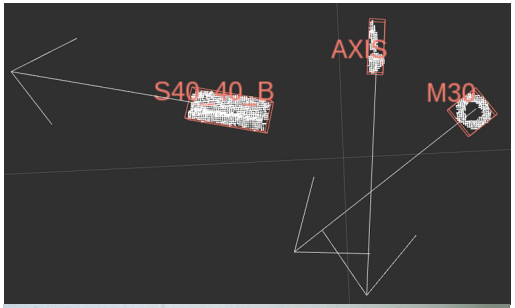
- Detecting people in human-populated environments is crucial for collision avoidance, human-robot-interaction, path planning and so forth
- Several approaches exist which employ different modalities [1-3] (e.g. Lidar, images, etc.) as input

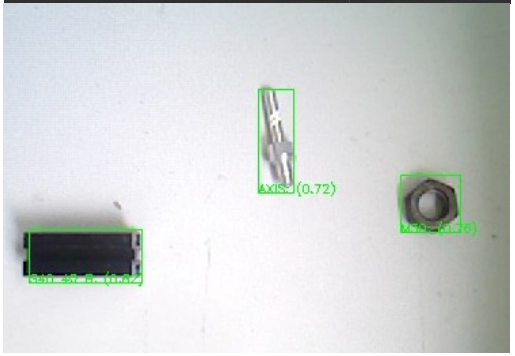
Objective

- Implement (integrate and optimize) a lightweight, fast (>15Hz) people detection approach on the ROBILE based on laser scan data
- Test the people detection system in various scenarios

[1] http://wiki.ros.org/cob_people_detection
[2] https://github.com/Thordreck/people_detector
[3] https://github.com/spencer-project/spencer_people_tracking

Object Perception in ROS2





Overview

- Object detection/recognition for @Work is currently implemented in ROS melodic
- Migration of @Work code to ROS2 is ongoing

Objective

- Get familiar with ROS2 concepts such as composition [1] and managed nodes [2]
- Rewrite existing ROS1 nodes for object detection and recognition using ROS2
- Show running system with a 3D camera (Realsense D435)

Languages/Frameworks

- ROS2
- C++, Python
- PCL (required), Tensorflow 2 (optional)

[1] https://docs.ros.org/en/foxy/Concepts/About-Composition.html
[2] https://github.com/ros2/demos/blob/foxy/lifecycle/README.rst

SLAM with Factor Graphs

- Implement factor graph using with GSL
  - Or understand factor graph building using gtsam library
- Use factor graphs to do slam with aruco marker based map building and \ localization
- Implement the toy problem from [1]
- Deploy on robot or oak-d camera.
- Additional: can be made a ROS2 node

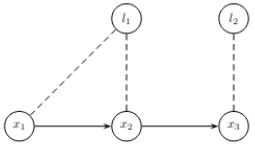


Figure 1.1: A toy SLAM (simultaneous localization and mapping) example with three robot poses and two landmarks. Above we schematically indicate the robot motion with arrows, while the dotted lines indicate bearing measurements.

[1] https://www.cs.cmu.edu/~kaess/pub/Dellaert17fnt.pdf
[2] GTSAM: GeorgiaTech Smoothing and Mapping [Dellaert et al.] https://github.com/borglab/gtsam
[3] http://ingmec.ual.es/~jlblanco/papers/2020-introduction-factor-graphs_JLBlanco.pdf
[4] https://berndpfrommer.github.io/tagslam_web/

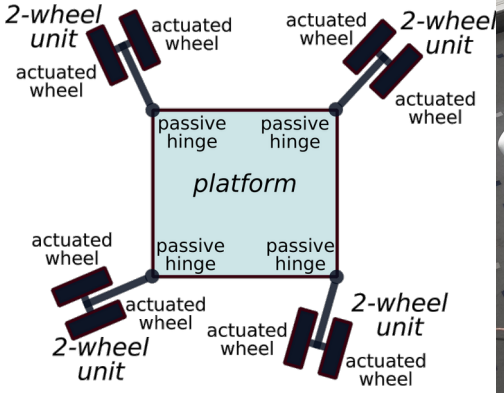
Concurrency in robot remote control

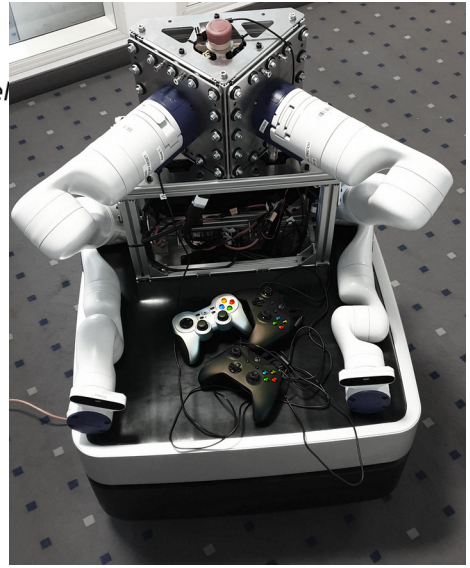
Overview

- At times we need to move our robots under remote control by a human
- Simple example of *concurrent activities*: (i) controlling robot; (ii) reading device input

Objectives

- Interface with gamepad [1]
- Base line: non-blocking I/O & polling
- Concurrent I/O:
  - Event loop (coroutines)
  - Threads (lock-free ring buffer [2])
- Command robot with existing interface [3]





Tools

- Linux, C

1) https://www.kernel.org/doc/html/latest/input/joydev/index.html
2) https://robmosys.pages.gitlab.kuleuven.be/composable-and-explainable-systems-of-systems.pdf (Sections 11.3 & 11.4)
3) https://github.com/rosym-project/robif2b/

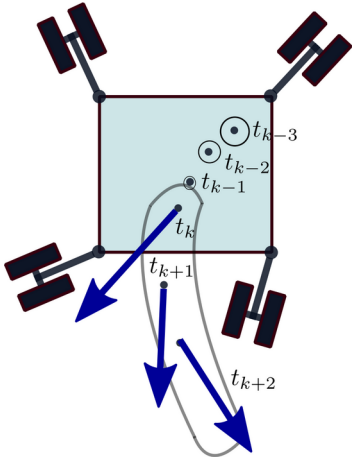
Motion primitives for Freddy

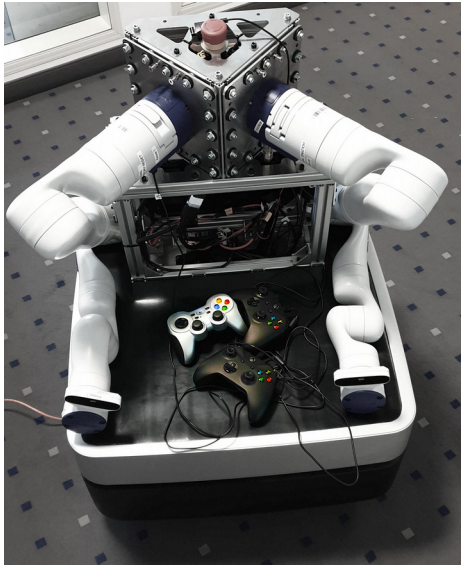
Overview

- From previous SDP class: instantaneous *force distribution solver* available
- *Constraints* describe desired platform forces, *objective* is force minimization
- Missing: non-instantaneous *control* to modify constraint and objectives

Objectives

- Realize controllers for various situations [1] e.g.
- … different wheel configurations
- … physical platform alignment
- … driving up a ramp



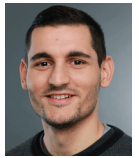


Tools

- C, robif2b [2]

1) https://robmosys.pages.gitlab.kuleuven.be/composable-and-explainable-systems-of-systems.pdf (Section 13.7.10)
2) https://github.com/rosym-project/robif2b/

Hardware Communication Layer for Freddy




Overview

- New robot Freddy has hardware from previous projects and different manufacturers
- Need a unified way to communicate with HW components for control, monitor, log, present

Objectives

- Performant and decoupled access to data from HW components (for control, logging, visualization) with Apache Arrow [1].
- Presentation of data in a web interface, e.g. with Perspective [2]
- Example applications with battery monitor and wheel sensors

Tools

- Black-box [3]: similar solution with different goals
- KELO wheel monitor with qt-based GUI [4]
- Battery monitor documentation [5]
- C API of Arrow [1] as communication layer
- Perspective [2] or alternatives to present data

[1] https://arrow.apache.org/docs/format/CDataInterface.html
[2] https://perspective.finos.org/
[3] https://github.com/ropod-project/black-box
[4] https://github.com/mas-group/kddv
[5] http://68.168.132.244/KG-F_EN_manual.pdf

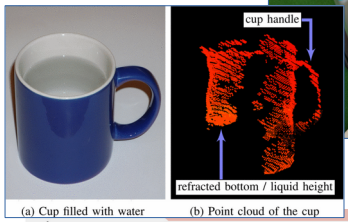
Feedback for Pouring Behaviour


Overview

- Some tasks in RoboCup require the pouring action (liquid, cereal,...)
- Normally terminate with a timer, but some feedback would be better

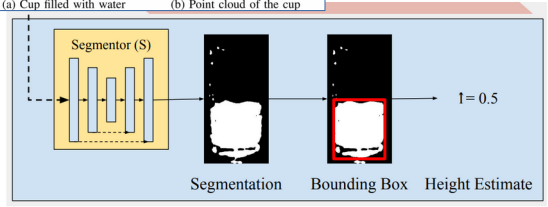
Project Goals

- Learn to work with the Toyota HSR
- Explore feedback methods using available sensors **on the robot** (wrist force sensor, RGB/-D camera(s))
- Experiment setup for testing different feedbacks
- Feedback must be fast enough for controlling the pouring motion (minimum ~5Hz)



Tools, Resources

- [1] weight estimation example using wrist force sensor (special permission required)
- [2] PR2 solution with (external?) RGB-D camera
- [3] liquid level segmentation with external RGB camera



[1] https://docs.hsr.io/hsr_develop_manual_en/ros_interface/force_sensor_sample.html
[2] Do et al. - 2016 - A probabilistic approach to liquid level detection in cups using an RGB-D camera
[2.1] Extension paper: https://arxiv.org/abs/1810.03303
[3] Narasimhan et al. – 2022 – Transparent Liquid Segmentation for Robotic Pouring

General Behaviour to Find Objects


Overview

- Some RoboCup@Home tasks (e.g. Find My Disk) [1] involve looking for a particular object (e.g. specific cup, table,…) in the environment
- Current solution [2] handles knowledge query, but lacks locomotion

Project Goals

- Implement a general strategy to find object, given "usual storage locations", e.g.:
  - Navigate through "storage locations"
  - Perceive scenes to look for specified object(s)
  - Move to next location if object not found
- Potential extension: "fetch" object and bring back to original location

Tools, Resources

- Image embedding calculation and comparison for recognizing faces [2, 3]
- Current implementation with knowledge query [1]
- Rule book [4]
- ROS, Python

[1] `mdr_find_object_action` package
[2] Embedding calculation for faces
[3] Embedding comparison for faces
[4] 2022 Rulebook Draft
https://github.com/alex-mitrevski/RuleBook/tree/2022/draft

Questions

Example User Story

Individual wheel unit control
Wheel units to a disered configuration wrt based of robot.

Priority:	Unique Identifier:	Estimate:
Task description: Wheel units to a desired configuration wrt base of robot (orientation)	Acceptance Criteria: • Bring the wheel to 0°, 270°, 180°, 90° • No overshoot by _ degrees	
Risk:	Real Effort:	

Task description: Alignment (

- Assumption: In front of the ramp

- Bring robot to alignment with the ramp.

Acceptance: Criteria

- 3 trial runs with

  platform alights

Example User Story

Priority: High	**Unique Identifier:** D3	**Estimate:** *6W*
As a *Software God* **I want to** *decouple the generation, storage, and reaction to events* **so that I can** *enhance the maintainability and composability of the software application*	**Acceptance Criteria:** • *The persistent database storage of monitoring events should be optional and decoupled in the form of a separate storage component* • *The component monitor should only be responsable for the generation of events* • *FTSM is only responsible for reacting to events*	
Risk: Minimum	**Real Effort:**	

⓪ — 25 - 9th may

① — 9 - 23 (Alignment)

② - (3 weeks) (23 - 13 June)

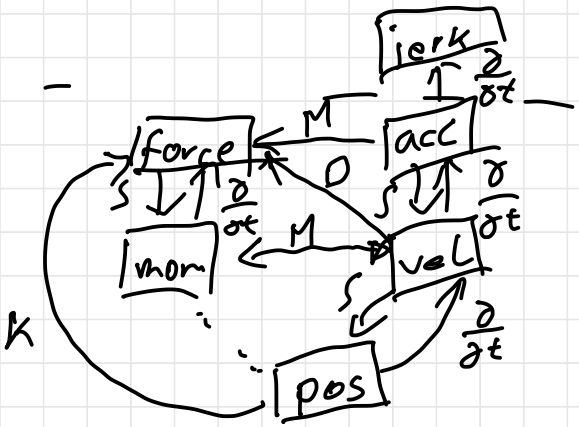
— Task description — Ramp

③ 4 weeks driving up the ramp.

④ 13 - 11 July

⑤ 2 weeks.

11 - 25 July



29 Aug

12 sept (final)

Road map -

Task description: Termination.

Ramp demonstration

- Type of ramp.

Task description: Integration by state machine.

demonstration