Lab Cleanup Robot using Mirte Master Platform

AUTHOR:

21st Apr, 2026

Abstract

In this demo, we demonstrate how Jupyter Book can be used to create and publish a content rich paper that includes interactive elements such as code cells, visualizations, and multimedia. We will walk through the process of setting up a Jupyter Book, adding content, and deploying the final product online.

Keywords Navigation2, ROS2, SLAM, Mobile$_{M}anipulation$

Contents

1 Introduction

Jupyter Book has been rebuild from ground up using the MyST engine (Jupyter et al., 2025a). This allows to export content in multiple output formats including HTML, PDF and docx. In this paper we present an overview of the possibilities and demonstrate its working.

In an introduction. you often cite. Than can be done in various ways, either using a .bib file or directly using the doi.

cite with doi

- `[@doi:10.25080/hwcj9957]` resulting in (Jupyter et al., 2025b)
- `@doi:10.25080/hwcj9957` resulting in Jupyter et al. (2025b)

cite from bib-file

- `{cite:t}jupyter2025“` resulting in **?**
- `{cite:p}jupyter2025“` resulting in (**?**)

1.1 Background

Jupyter Book has been rebuild with the intend to export content in multiple output formats including HTML, PDF and docx. Figure 1 provides this idea.

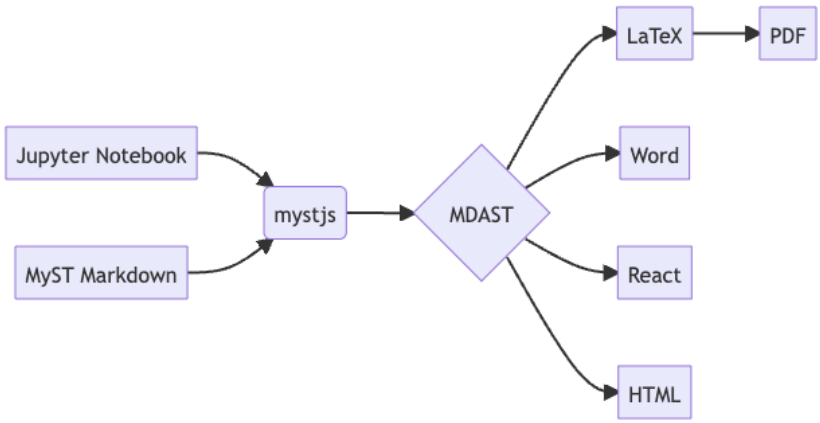


Figure 1: The myst engine allows Jupyter Notebook, markdown and even tex files to be converted to multiple output formats.

As exporting to different formats is possible, it is not always desired. Some content should only be visible in the HTML version, and some content only need to be included in the PDF version. You can use blocks like `+++{"no-pdf":true}` to enable this, as shown below where the figure is seen in the HTML version but not in the PDF version.

2 Theory

Autonomous mobile manipulation requires the integration of navigation, mapping, perception, and localization into a unified system. In this work, these components are orchestrated using a behavior-based control framework.

2.1 Navigation

Navigation refers to the process of planning and executing collision-free motion from a start to a goal position. It is typically decomposed into:

- **Global planning**: computing an optimal path on a map.
- **Local planning**: generating feasible velocity commands.

Modern navigation systems use costmaps and search-based planners such as A* or Dijkstra. In Nav2, navigation is modular, allowing different planners and controllers to be interchanged. Since Local planning can mostly be left to Nav2 for this section only some theory is discussed on how coverage path planning is done in this case.

2.1.1 Global Planning Methods

Global Path planning includes two major steps. Namely Decomposition and Path planning

---

2.2 Mapping

Mapping is the process of constructing a representation of the environment. In unknown environments, this is often addressed using **Simultaneous Localization and Mapping (SLAM)**.

A common representation is the **occupancy grid**, where each cell encodes the probability of being occupied. Mapping can be expressed probabilistically as:

$$p(m \mid z_{1:t}, x_{1:t}) \tag{1}$$

where:

- m is the map
- $z_{1:t}$ are sensor measurements
- $x_{1:t}$ are robot poses

Different SLAM approaches (e.g., grid-based vs feature-based) trade off accuracy, computational cost, and robustness.

2.3 Localization

Localization estimates the robot's pose within a known map. A widely used approach is **probabilistic localization**, such as particle filters.

The recursive Bayesian update is given by:

$$p(x_t \mid z_{1:t}, u_{1:t})p(z_t \mid x_t), p(x_t \mid u_t, x_{t-1}) \tag{2}$$

where:

- x_t is the robot pose
- z_t is the observation
- u_t is the control input

In practice, methods like Adaptive Monte Carlo Localization (AMCL) are commonly used within Nav2.

2.4 Perception

Perception enables the robot to interpret sensor data to detect and classify objects. This includes:

- **Object detection**: identifying candidate objects in the scene
- **Classification**: assigning semantic labels (e.g., electronics vs other)

Libraries such as OpenCV and Open3D are commonly used for image and point cloud processing.

Perception outputs are used by higher-level decision systems to guide actions such as navigation and manipulation.

---

equation example:

$$c\varphi \tag{3}$$

$$i\frac{\psi}{t} = -\frac{^2}{2m}{}^2\psi + V\psi \tag{4}$$

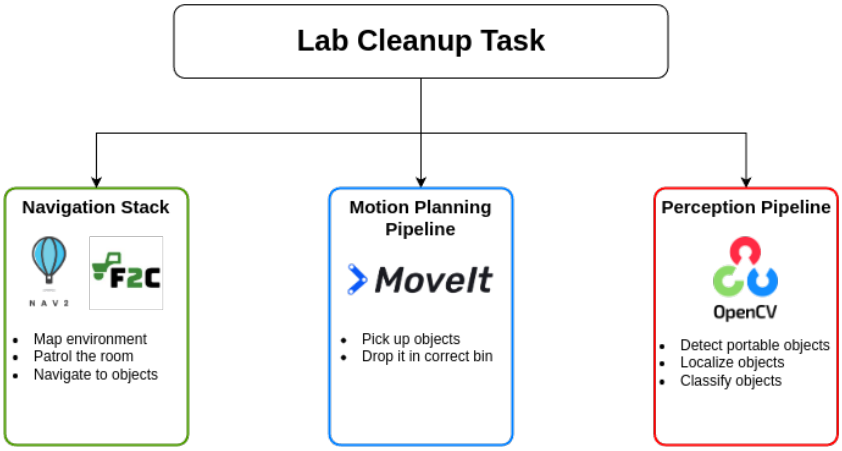
We can link to equations using their labels, like equation (4) or with more emphasis: (4). See the documentation for more options with using formulas. You might be interested in specific ways of numbering.

3 System Overview

3.1 System level strategy

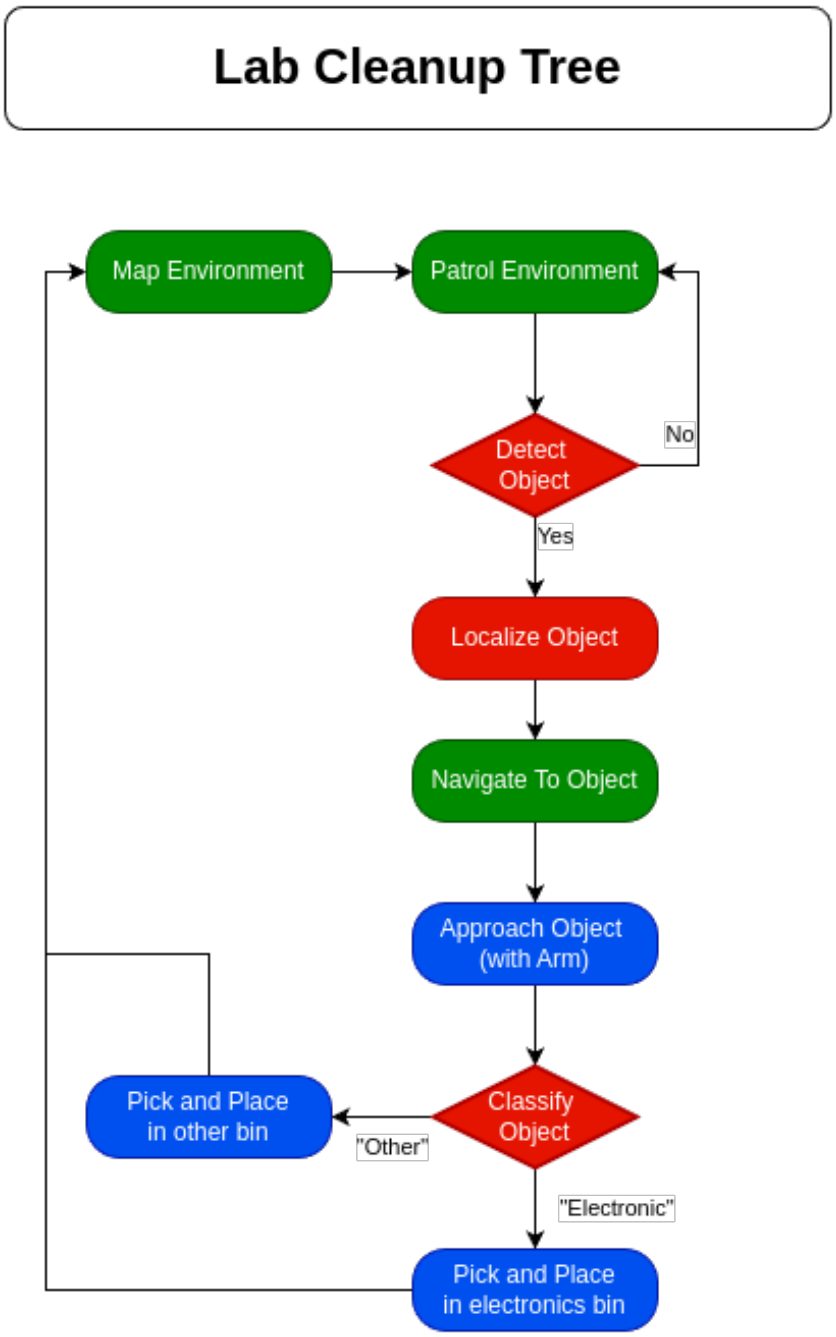
The setup consists of a mobile manipulator (Mirte Master) tasked with autonomously exploring an indoor environment (the robotics lab), identifying and localizing objects, distinguishing between electronics and other objects, and sorting these objects accordingly. The task can be broken down as follows.

The main task of the robot can be broken down into sub-tasks which then fit into specific niches of the entire system architecture. **Motion planning**, **perception** and **navigation**. Figure **??** showcases this idea.



The scope of our investigation includes how different perception(classification) and navigation(mapping and patrolling) approaches influence the performance of the whole system. The highest performing combination of approaches is then chosen for use in the respective subsystems.

The system level strategy consists of how the robot behaves in each scenario that it will find itself. The tasks described above are used as a guideline to construct the full system level strategy. The standard in robotics for such navigation and perception heavy tasks is to use a global behavior tree that describes how the robot ought to behave in certain situations. This system level architecture as shown in Figure **??** is also used here to describe and execute the cleaning strategy.


Lab Cleanup Tree
Map Environment
Patrol Environment
Detect Object
No
Yes
Localize Object
Navigate To Object
Approach Object (with Arm)
Classify Object
Pick and Place in other bin
"Other"
"Electronic"
Pick and Place in electronics bin

3.2 Navigation

The navigation stack of this robot, powered by nav2 and fields2cover, executes three main tasks:

- Mapping
- Patrolling
- Object Approach

3.2.1 Mapping

Before the robot can navigate properly through the environment and ensure proper coverage, this environment must first be known. Thats where mapping approaches come in handy. Several advanced and specialized mapping approaches exist already, but in this paper only three are considered. Frontier based mapping, {2}, {3}. These were chosen through a combination of novelty and frequent application in similar contexts. Once the environment is mapped {sufficiently} the robot is allowed to navigate the space and propagate its behavior further down the tree.

Add ros2 bag playback

3.2.2 Patrolling

Two categories of coverage strategies are implemented:

Systematic Navigation: The environment is traversed using a structured pattern to ensure complete coverage. These approaches are often called coverage methods and paired with decomposition methods discussed in the theory section

Reactive Navigation: The robot dynamically traverses and selects exploration targets based on frontier regions or detected objects, prioritizing areas of high information gain.

iets meer over coverage planning.

Free space: The ROS framework allows for integration with a lot of modular software packages. throughout the years this has given rise to many universally applied packages, Nav2 is such a case. Using Nav2 the Global Costmap can be used to extrapolate the free space for the robot to move in.

3.3 Perception

Perception is responsible for object detection, classification and spatial localization. For this study we limited the object detection and localization approaches to two. A **2D detection and depth projection** approach, and 3D segmentation with **point cloud clustering**. As for classification methods, this study evaluates a classical cv approach, aswell as a classification model based approach.....

objects are categorized into:

1. Graspable vs non-graspable
2. Electronics vs non-electronics

   iets meer over classification.

4 Methodology

This study follows factorial experimental design, analyzing the interplay between perception, coverage and system level decision making.

Three independent variables are defined:

- perception quality
  - Classical cv based classification
  - Model based classification
- detection method
  - point cloud clustering
  - depth projection
- coverage strategy
  - Systematic Coverage following a predefined path
  - Reactive Coverage by acting in real time
- System-level strategy
  - Clean-as-you-go
  - Map-then-optimize

this results in a total of 16 experimental configurations, however some of these setups are fundamentally incompatible, such as Systematic coverage with a clean as you go approach. The full setup is shown in table x.

4.1 Evaluation Metrics

4.1.1 Task level metrics

percentage of electronics, total completion time, failed attempts, distance traveled

4.1.2 Component level metrics

detection percision and recall, localization error, coverage percentage.

5 Results

With JupyterBook it is possible to make your narrative and data-analysis available in a single file, through Jupyter Notebooks. It is even possible to create interactive materials using widgets on the website, but leave them out in the PDF version (using tags). To run the code below, click the icon in the top right corner. This will connect with a kernel. If the kernel is loaded, click to run the code.

Figure 2 shows the results of fitting our linear model to the data. Parameter a is optimal when the chi-squared curve is at its minimum, which corresponds to the smallest residuals and the best match between model and data.

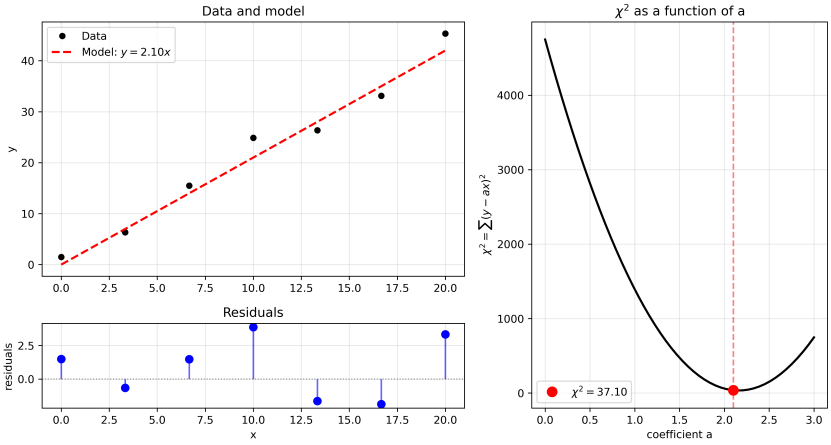


Figure 2: Fitting our data to a linear model with a single parameter a. The optimal value of a is found when the chi-squared curve (bottom) is at its minimum. The residuals (top right) are smallest at this point, and the model (top left) best matches the data.

6 Conclusion

In this study we investigated whether and how scientific publishing is feasible through the use of Jupyter Book. Using a starterkit template repository which can be easily accessed and used, a head start is provided. The manual provides detailed information to make more technical elements accessible. Hence, we conclude that Jupyter Book provides a feasible way of publishing scientific content - at least for bachelor and master thesis.

7 Appendix

7.1 Cheatsheet

7.2 README

This is the TU Delft starterkit for open publishing with Jupyter Book. It provides a skeleton with basic settings to have a head start with your thesis / project.

It includes:

- a github deploy file taking care of deploying the website and building a pdf
- basic content files that can be altered to fit your project
- yml files in the root folder to configure the book and the pdf build

7.2.1 Where to start

Already know about Jupyter Book, familiar with GitHub or GitLab, Markdown... start with cloning the starterkit and start writing working on your project. New to the ecosystem? Start with the manual, and then move on to the starterkit.

7.2.2 Purpose

The purpose of this project and this manual is to enable others to use Jupyter Book for writing and publishing their own scientific and educational content. We hope to lower the technical barrier for researchers and educators to create and share their own resources, and to promote open science and open education practices.

7.2.3 License

7.2.4 Errors

If you find any errors in the content, please report them by opening an issue on the GitHub repository.

7.2.5 Funding

This project has received funding from the Delft University of Technology Open Science Fund (2025-2026).

7.2.6 Identifier

TUDJBOSSTARTERKIT

References

P. Jupyter, E. Bolyen, J. G. Caporaso, R. Cockett, D. Garside, C. Holdgraf, A. Hollands, J. Kent, F. Koch, J. Madge, M. McKay, M. Morrison, F. Pérez, S. Purves, M. R. Kelley, B. E. J. Rose, M. Sharan, B. M. Sipőcz, S. J. Walt, and K. J. Whitaker. Jupyter Book 2 and the MyST Document Stack. In *Proceedings of the Python in Science Conference*, pages 173–193. SciPy, jul 10 2025a. doi:10.25080/hwcj9957. URL http://dx.doi.org/10.25080/hwcj9957.

P. Jupyter, E. Bolyen, J. G. Caporaso, R. Cockett, D. Garside, C. Holdgraf, A. Hollands, J. Kent, F. Koch, J. Madge, M. McKay, M. Morrison, F. Pérez, S. Purves, M. R. Kelley, B. E. J. Rose, M. Sharan, B. M. Sipőcz, S. J. Walt, and K. J. Whitaker. Jupyter Book 2 and the MyST Document Stack: A modular, extensible, web-native stack for authoring and publishing computational narratives. In *Proceedings of the 24th Python in Science Conference*, SciPy, pages 173–193. SciPy, 2025b. doi:10.25080/hwcj9957. URL http://dx.doi.org/10.25080/hwcj9957.