Night Time Robot Patrol System

Project Plan

Computer Vision Master Project 2018/2019

Fabian Behrendt, Chris Carstensen,
Christian Reichel, Nicolás Pérez de Olaguer, Caus Danu, Nana Baah

15th of February 2019

1 Introduction

In environments with high cost equipment, patrolling and surveillance is important to lower the risk of thievery. For indoor surveillance, the security staff has to patrol around the floors and establish a higher security by presence, but in the same time this is inefficient and coupled with high costs. Also, many environments, including offices have just one security person per night to patrol indoors. On the other hand, using indoor surveillance with night cameras is not feasible and in offices illegal, due to German Federal Data Protection Act. For this reason, we propose a robot system that can patrol autonomously through corridors and can detect and report any anomalies. The robot will patrol in the building of the University of Hamburg, Department of Informatics. Its main task will be to detect doors of the environment and check their status, particularly if they are closed during the non-working hours. Consequently, the robot will notify if it detects an open door and will trigger a message with the door location through a remotely accessible, web based GUI. This project faces many complexities. For example in the field of vision, one of the most challenging problems is working with low illumination. Ideally, the robot would be able to patrol during night time. Therefore, a night vision enhancement technique must be provided. Moreover, door detection is not a straightforward process. Many objects in a room may have rectangular shapes, resembling a door. Consequently, an advanced vision algorithm is needed to correctly detect doors from different view points. The idea we propose is an attention system based on a "search and inspect" behaviour, i.e. a door should be located such that the robot will drive until close vicinity to it for further inspection. The result of the inspection is a reporting of whether the door is open or closed.

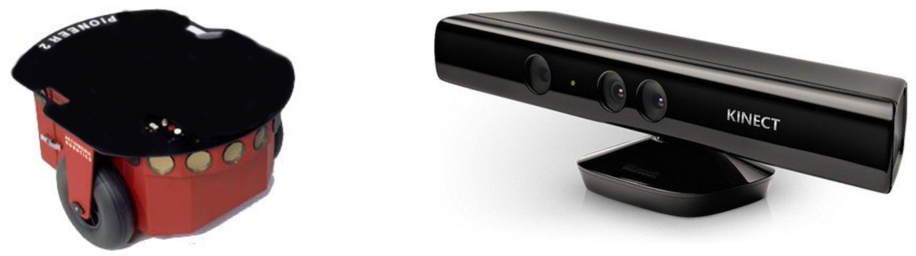
1.1 Preconditions

For this project we will make use of the following components:

- A Pioneer P2-DX Robot as seen in figure 1a
- A Kinect Version 2 camera, based on infrared technology for depth perception (see figure 1b)
- One or multiple RGB cameras for image capturing
- CUDA enabled GPUs with 4GB of memory for training the neural net components

Additionally, we plan to use a reporting environment based on web related python technologies like the "Django" framework for example or other. To host the module, we will need a server machine, on which we will install specialized software like WSGI or NGINX for actually serving the web pages, distributing work load and ensuring the low level communication with our robot components. WiFi access has to be available as well in order for the communication to be established. Otherwise, the robot would have to continue working without having a reporting functionality until it reaches a local network access point.

The other components will be implemented using C++ and Python, wrapped in ROS modules that will subsequently be handled by the ROS Framework.



(a) Pioneer P2-DX Robot

(b) Infrared (version 2) Kinect Camera

Figure 1: Main hardware components

2 Methods and Background

The focus of this project will be the computer vision part of the system. At first we will only inspect doors with our robot and see whether they are opened or closed. This section describes some background and methods we intend to use in the project.

2.1 Door detection

The detection of doors has been done previously by many contributors in the field of indoor navigation for mobile robots. Most of the work is based on using sonar or laser range scanners, having doors as landmarks for the localization of the robot or to traverse them successfully. The papers by Birchfeld et. al. and Monasteiro et. al. on the other hand, describe the usage of 2D images to extract features of the environment and check the mapped and detected lines with predefined constraints [4,11]. In many cases, the developed system had a high detection rate of 97.7%, yet has wrongly detected other parts of the environment as doors. Based on the learnings of previous papers, we consider the following aspects to make a detection of a door challenging:

- The robot does not stand directly in front of the door, so we just can work with a partially, jamb-, surface- or frame-occluded door. The robot might not even stand in front of any door at all.
- The door has its height lintel-occluded. Thus, we have to work with simpler geometries

- The door can be opened to the inside or outside of a room.
- The door jambs might not be in view (in case the robot is too close to the door frame)
- The door itself does not have an even surface or is not rectangular
- The colour can be different or equal to that of the wall
- Colours of doors might lead to lack of contrast
- The surface of the door can be too reflective
- The door can have a kick plate
- The door can have a gap below and may lead to light and shadow side effects
- The environment can have a disturbing pattern
- The floor of the corridor can be too reflective
- The light conditions can differ and produce shadows on the floor or opposite walls or even be very bad for capturing 2D images
- Book cabinets also have doors
- Vendor machines and other miscellaneous objects often have a strong contrast to their environment and may lead to a detected door ("false positives")

Probably combining 2D with 3D data might solve some of the visually distracting effects of the environment. Early papers have tried to combine the 2D information with the distance to the environment using laser range scanning [1], but were assuming similarly coloured doors and environments.

2.2 Door status classification

Doors are meaningful entities in an indoor setting and they help the localization process of a robot. In the approach of Avtos et. al., the location of doors, as well as their status are used to locate the robot with particle filters and provide extra contextual information related to the doors [3]. Avtos et.

al have shown that a robot is able to detect the status of a door, as long as the location of the doors in the map is known [3]. Recent works by Nieuwenhuisen et. al. show that a robot can still localize itself in an environment of outwards opened doors with particle filters, but the status has to be very clear and fully open, partially open or closed [12]. There is a high chance that narrow open doors will be considered closed since the scanners might not be able to send a ray through the slit. Both papers are using outwards opening doors for localization with a sonar or radar scanner.

To summarize, a door can have the following states (regardless if they are inwards or outwards opening doors, left or right side opened) which we have to consider:

- Fully open
- Partially open
- Narrow open
- Closed

2.2.1 Bounding box prediction and classification

One way to classify the door status is to use the 2D image captured by the Kinect camera, to first optimize the color values and then use a machine learning technique to classify the image. Currently, convolutional neural networks like Faster R-CNN by Ren et. al. [15] or the more recent works like RetinaNet by Lin et. al. [10] achieve a high accuracy with short computing time on modern GPUs in comparison to traditional sliding window approaches. RetinaNet is remarkable by introducing a custom loss function called "Focal Loss" function, that lowers the influences of complex backgrounds for object detection, and by combining the feature pyramid method of Mask R-CNN by He et. al [6] for detection of objects in even smaller sizes.

2.2.2 Point clouds and PointNet

Combining depth and 2D image information captured by the Kinect might help making the classification more accurate. Since a door can be opened inwards or outwards, it might be a reliable data source which can be suitable for the number of different factors a door and its environment can have (see beginning of section 2.2). Through the additional geometrical data, we have an extra helper source of classifying the door even with difficult 2D imaging or light conditions. However, 3D information adds complexity because of the undefined order relationship between points, which is expected in CNNs and RNNs. In fact, earlier approaches have tried to project the depth information into plain 2D images and train CNNs to learn the multi-view features of it [19]. Qi et. al. have proposed a PointNet segmentation and classification network that is able to learn and classify point clouds of any dimension without the need of projecting the points into a lower (2D) space [14].

2.2.3 Point combination into surfaces and supervoxels

Because we are interested in simple environment geometry for the detection, and less parameters to be learned or classified by neural nets or potentially support vector machines, we want to find a way to simplify the depth information, to reduce the number of features, as well as noise. One way is to fit geometric elements into the point cloud and filter all residuals away [17]. Another one is to use a seed growing technique to combine groups of points in the same surface, called supervoxels [13]. By using saliency maps, we could get supervoxels that are more aligned to the 3D boundaries [5].

To summarize, there are multiple ways how we can resolve and classify open doors. The exact tasks and potential solutions are described in the following sections.

3 Tasks

This section provides an overview of the different essential tasks within the overall system. The entire application is divided into modules that help the robot navigate, (pre-)process data and report to the user.

Figure 2 depicts an overview of the intended information flow. On the highest abstraction level, there are three sub-modules: "Navigation", "Data Preparation" and "Reporting". Each module has further delegated sub-components for the conceptually important tasks.

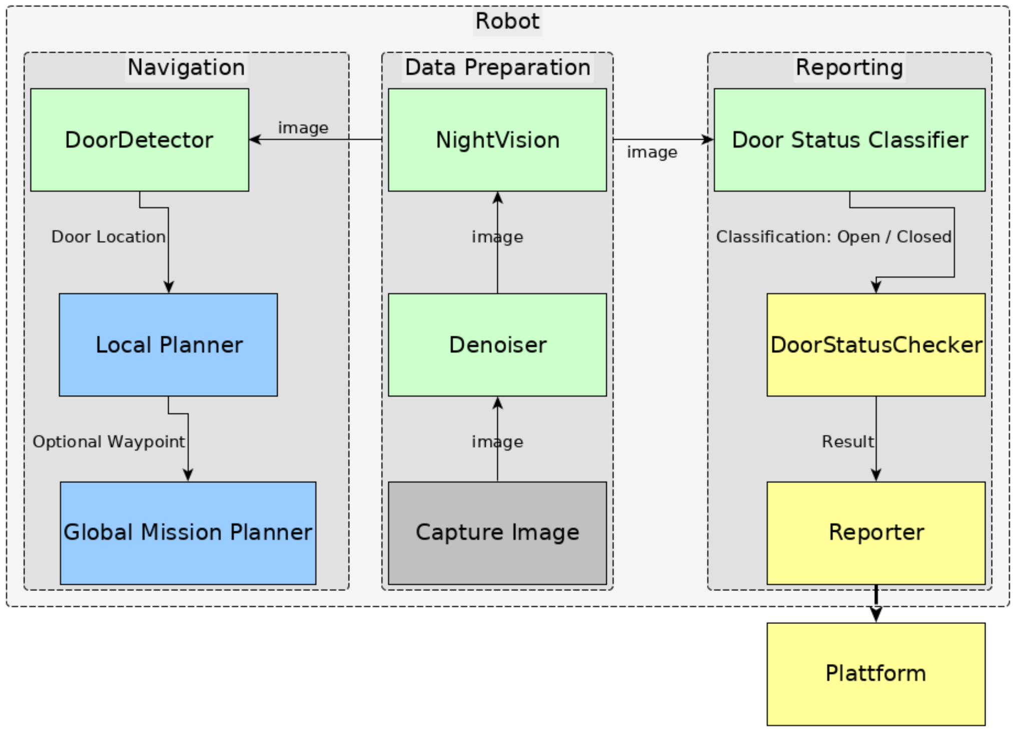


Figure 2: *Information Flow Overview*
This figure shows the general information flow between the modules of our system. The green squares highlight modules with focus on computer vision.

In the following, the main building blocks, as well as their planned submodules and associated tasks are explained in more detail.

3.1 Navigation

The navigation subsystem controls the robot and is in charge of moving it through the university. We make use of some pre-existing ROS nodes, described below, as well as custom made modules implemented specifically for the task at hand.

3.1.1 Move_base Module

Since we will use the Pioneer Robot shown in Figure 1a, we will also use the provided *move_base* ROS node to control it. This component takes care of the movement, as well as path planning tasks. All needed functionality should be already implemented. By using this ROS node, we can send our robot a position in its coordinate frame of reference and the *move_base* node would trigger the relevant motion steps. To get more information about the environment, in path planning purposes for instance, we generate a map of the environment. This is accomplished by the next module.

3.1.2 SLAM

Since our robot should navigate through the university by itself and a GPS tracker is not suitable for indoor applications, we rely on Simultaneous Localization and Mapping (SLAM) principles. As with the previous module, we rely on a provided ROS node package, for example the *rgbdslam*. For this task we might have to compare different implementations available and see which one suits our needs the most. As this is an essential component of our system, we might also need to implement this by ourselves.

3.1.3 Global Mission Planner

The *Global Mission Planner* is responsible for planning the general route the robot should take while patrolling through the building. The path of the robot is defined by several checkpoints it has to pass through.

Our first attempt will be to define static landmarks between those checkpoints, that describe the positions of the doors. If the robot reaches those landmarks, it will turn towards the door to check if the door is open or closed.

The second attempt would be to incorporate the information of the *Local Planner* into the global planning, which will be described in the next section. This allows for dynamically detecting doors while moving around instead of using only static landmarks.

If possible, we would also consider a docking station landmark i.e. a predefined location where the robot goes to when it is running low on power.

3.1.4 Local Planner

The *Local Planner* relies on the *Door Detector*, which detects doors during runtime and will be explained in section 3.3.2. It receives the position of the detected door relative to the robot and calculates a landmark orthogonal to the door. The *Local Planner* then sends this landmark to the *Global Mission Planner*, which will integrate it into the path.

3.2 Data Preparation

This subsystem will provide the input data for all other components. Within it, the raw sensor data will be filtered and enhanced by methods described in the following.

3.2.1 Capture Image

Capturing RGB-D images would be the first step for collecting data and enabling the robot to sense its surroundings. To get the images from the Kinect camera, we are going to use a ROS package already implementing this functionality, namely *usb_cam*, which runs on the ROS node *ros_cam_node*.

3.2.2 Denoiser

The "Denoiser" component is the first subcomponent of the "Data Enhancement" block. It takes the raw RGBD input from the camera and applies various filtering algorithms in order to clean it. It than spits out the clean filtered image to the second component of the "Data Enhancement" block, namely: the "Night Vision" module.

The main filter for the depth images will most likely be an interpolator, because the infrared rays of the Kinect Camera are dispersed radially and hence, the resolution of the further away regions is smaller as compared to the close by regions. Therefore, the interpolator will serve as a resolution enhancement component.

Regarding the RGB components of the input, we can do some gamma correction and lightness/contrast correction in real time such that the input is (more or less) standardized for the neural nets down the pipeline. This is an important auxiliary step, even if neural nets typically are designed to be color/light invariant (since we might not have enough data to make them as "invariant" as we like).

Generally speaking, we can also use some median filter to reduce the "salt and pepper" noise (especially in reference to the RGB data) and smoothing filters to reduce jitter noise (especially in reference to the depth data). The jitter can also be reduced using Kalman filtering ideas of deciding to trust or not trust highly oscillating sensor measurements from the depth camera.

3.2.3 Night Vision Approach

If the robot wants to move during the night, we need an algorithm to enhance the low illumination video frames. Following [9], we propose an algorithm that fuses daylight images with their night counterparts. Ideally, the robot will be able to move in the environment during night time. Moving in the dark poses a clear challenge. Low-illumination images are prone to have a low signal-to-noise ratio (SNR). The objective of this module is to enhance the captured image for the robot to work as well as in a normal illumination scenario. As an input of the algorithm we have the illuminated and the non-illuminated video frames. As an output, ideally, we have an enhanced image after fusing/aligning both data sources with meaningful information extracted. This will then act as input for the "Door Detector" and the "Door Status Classifier" blocks.

3.2.4 Data Collection

To be able to develop our systems without the robot running, we need to collect data for offline usage. Since we are aiming to use machine learning algorithms, which are driven by a lot of data, we will manually move the robot through the corridors of the university and record the necessary information streams like localization and visual data. Subsequently, we can for example split the video material into images to train the necessary neural networks or feed them in other non-neural net blocks/algorithms.

3.3 Computer Vision

As one can see from Figure 2, there is no such thing as a standalone computer vision module or subsystem. Nevertheless, since the success of this project relies heavily on computer vision principles, this section describes in more detail the modules that form the computer vision part of the project and how they contribute to the task of allowing the robot to make sense of its environment.

3.3.1 Door Status Classifier

The aim of the Door Status Classifier block is to mark a door as open or closed. The classification result is then fed to the DoorStatusChecker (see section 3.5) which sends the result with additional meta data via the Reporter module (see section 3.4.2) to the user platform (see section 3.4.1). We assume that this will happen after the robot has detected a door and has positioned itself facing the door with the help of the local planner (see section 3.1.4).

The following tasks are considered necessary to achieve a robust and reliable classification:

- ☐ Pre-capture RGBD images, using the robot, to get a broader view of possibilities and challenges in our specific university environment. We also have to keep in mind the visual and status differences of a door and its environment mentioned in section 2.2.
- ☐ Analyze data features, clarify potential differences based on the captured data, build hypothesis.
- ☐ Research for current solutions, select fitting ones from options like: SVM, neural net based solutions etc.
- ☐ Capture training and test data in different conditions, angles and with various properties; capture training and test data from the environment for a control class (i.e. not a door), especially cabinets, vendor machines etc., and split it evenly per each status and class type.
- ☐ Mark the doors with bounding boxes and label their status.
- ☐ Augment the dataset by flipping, recolouring, shear transformations, pixel shifting etc.
- ☐ Select best features.
- ☐ Select loss and training functions, if needed.
- ☐ Build the model.
- ☐ Train and test the model as long as it isn't overfitting to the data.
- ☐ Drop, merge features, retrain, refine model.
- ☐ Check performance requirements and adjust the model if necessary.
- ☐ Evaluate the model against other test sets.
- ☐ Integrate it into the Door Status Checker pipeline.

3.3.2 Door Detector

As mentioned previously, the overall setup works like an attention system. The robot moves around and searches for regions of interest (ROI). Once it detects those regions, it focuses attention on them and investigates them closer. In our case, matching some regions of interest is the task of the Door Detector, while looking closer and analyzing them is the job of the classifier described previously.

Unlike the classifier, which can potentially spend more time on analyzing a door in order to semantically classify it as open or closed, the Door Detector has to be quick and work in real time. By "quick" it is meant that the robot should classify all objects that look like doors from a distance at a normal speed of the robot wheels. If we revert to ideas from attention theory, the Door Detector should be a component that uses bottom-up methods to "match" some pattern to a real world scene. We can also implement some top-down ideas to actually "search" for those patterns in the scene by using some preexisting knowledge about the nature of the environment in which the robot moves. However, since the requirement is to be fast, we anticipate that bottom-up matching algorithms are more suited, since they are usually more "parallelizable" and hence: work faster.

We plan to try out two main approaches to engineer the Door Detector component. One is more "manual"/using a vanilla computer vision or image processing algorithm and the other is more "deep"/neural net related, using a trained convolutional neural network (CNN). A Hough Transform is the algorithm preferred as an alternative to the deep approach (Paper [7] uses a Hough Transform in conjunction with a Canny Edge Detector resulting in 80% accuracy at an inference time of 7 seconds per door). We will have to define a Hough Transformation that would recognize the rectangular shape of a door as well as the shape of the door seen from the side, in various sizes and from different angles. This is to say that we mainly need to recognize rectangular, as well as trapezoid shapes of different sizes. The neural net alternative is to get a pretrained neural net like the VGG16 network and fine tune it by replacing the classification fully connected layers to recognize doors in our environment specifically. Paper [15] describes how to obtain a fast inference approach using only 5 frames for detection of the relevant object with a mean average precision of 73%.

Once a candidate door is detected, a bounding box is created and the coordinates of that bounding box (or center of mass of the door/box) are sent to the Local Planner so that it can calculate an estimate of where to move and how far in order to reach the door (such that the classifier can take over and start its process of analysis).

3.4 Reporting

Since the goal of our robot is to observe the corridors of the university by night, it has to report its findings to the human guard in a timely fashion. This subsystem identifies and reports open doors to a human via a plattform described below.

3.4.1 Reporting and Monitoring Platform

To monitor the patrolling progress of the mobile robot, a remotely accessible graphical user interface is necessary, that also updates the view by itself (so called adoptable or reactive user interface). We want to build a web-based GUI that would display the SLAM-based indoor map, as well as the detected locations of doors and the location of the robot as in-the-browser-map annotations. Additionally, the door status will be visible in the map, conveying the following information:

1. not checked,
2. checked and closed,
3. checked and open.

The information is forwarded to the robotic platform itself by using the "Reporting" node, which acts as a client. Location based information like the position of the robot or of a certain door will be transferred into the map view taking into consideration the coordinate transfer function between the GUI map and the actual map of the robot.

The tasks for this submodule are the following:

- ☐ Design the concept and the UI by prototyping.
- ☐ Find an adoptable framework for modern web browser applications.
- ☐ Build the RESTful API.
- ☐ Test its connection (via direct API calls).

☐ Develop a SLAM coordinate frame to GUI coordinate frame transfer function.

☐ Deploy the modules on a server.

3.4.2 Reporting Node

To get the information gathered by the robot, we want to establish a client server communication. Similar projects have been successfully created using Representational State Transfer (REST) architecture ([8,16]). For example, ROSTful is an implementation that is enabling the access to ROS-Topics directly via RESTful API calls and is open sourced at `https://github.com/pyros-dev/rostful`. Another possibility is to use a cloud based browser interface to control a robot and monitor its state, like the Robot Web Service by Toris et. al. ([20]).

We want to build a lightweight module in Python that only calls the RESTful API of our web platform to transfer data unidirectionally. A bidirectional communication to e.g. control the robot can be part of a later version of the platform. We call this module ”Reporter” for simplicity. The API is called for the following instances:

- Register a robot on the platform (to make multi-robot monitoring possible in later stages).
- Transfer the gathered indoor map (possibly via Base64 encoding and decoding).
- Send the coordinate space metadata to enable the platform to do coordinate transformations.
- Send the current coordinate of the robot.
- Send the location of any detected door.
- Inform the platform about the status of a door and relevant identifiers (like location, ID or semantic name).

The following tasks are sufficient to achieve the Reporter node:

☐ Setup the Reporter module as ROS node.

☐ Test the consumption of messages in relevant topics.

☐ Setup RESTful API calls.

☐ Test the whole flow.

3.5 Door Status Checker

The Door Status Checker is an intermediary module that gathers all the required information from other components and sends it via the Reporter node to the user platform (as depicted in Figure 2). The input data that is forwarded to the Reporter module is:

- Current status as computed by the Door Status Classifier (i.e. Open/Closed)
- Current (door identification) timestamp
- Current location of the robot

The current location of the robot and the position of the door can be determined from the SLAM map.

If the Door Status Classifier will be extended to provide even more semantic information besides the open/closed status, this module will have to gather it as well.

Simply put, this component is more or less of an interface between the reporting side and the robotic side of the application. If we refer to a Model View Controller paradigm of application architecture, this component can be regarded as part of the Data-Model to be accessed by the View.

4 Timetable

We estimated the construction timing of each submodule via a popular Scrum method called Story Points [18]. These are Fibonacci numbers indicating the difficulty of each subtask. In later steps, we have to split the tasks in smaller, consumable subtasks for which we can track time in a more exact fashion. The list of estimated story points is shown in table 1.

From storypoints, we have estimated the actual time on a week level and determined that we will need roughly one week per storypoint, which leads to a total estimation of 90 weeks of work. Figure 3 shows the roadmap with dependencies and foreseeable constraints factored in. We hope to finish the project until end of August.

Subtask	Estimated story points	Estimation in weeks
SLAM	8	8
Global Mission Planner	8	8
Denoiser	3	3
Night Vision	13	13
Door Status Classifier	21	21
Reporting Node	2	2
Platform	3	3
Capture Image	1	1
Local Planner	13	13
Door Status Checker	3	3
Door Detector	13	13
Data Collection	2	2
TOTAL	90	90

Table 1: Time estimation using story points

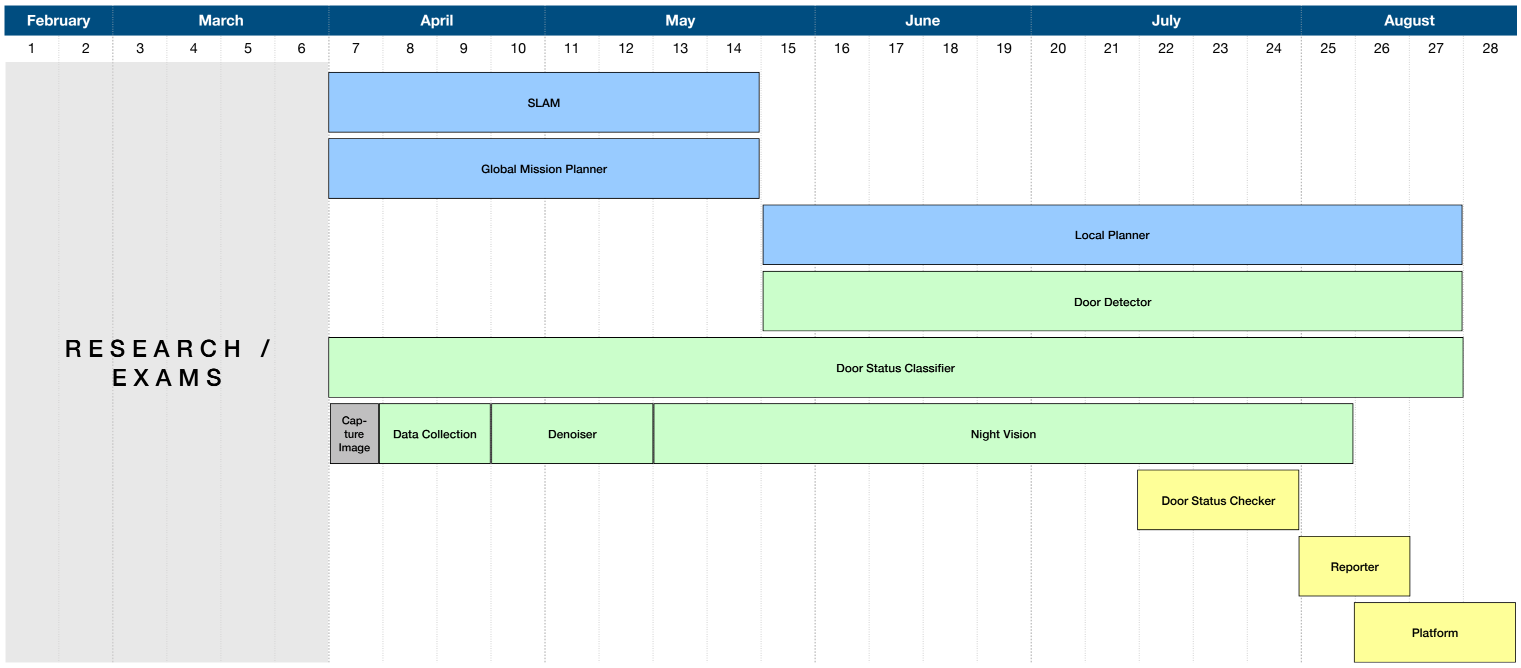


Figure 3: Roadmap (based on dependencies and constraints)

5 Evaluation

5.1 Door Detector

The door detector shall be evaluated in regards with two aspects: how fast it is and how accurate. Detection accuracy will be measured by using ambiguous objects like cupboards, cabinets, vending machines, as well as doors of different colors and shape.

We aim for an accuracy of at least 80% at a speed of at most seven seconds per door detection. This inference speed should suffice at a slow moving rate of the robot wheels.

5.2 Door Status Classifier

The door status classifier should be robust and classify each of the status types with a high accuracy. The narrow open door case will give us interesting insights of how well the system performs. The successful door status classification should be similar to the door detection rate from Birchfeld et. al., of roughly 97% [4]. We will use the same property list as seen in the paper and test the classifier against:

- concavity
- bottom gap
- vanishing point
- kick plate
- texture
- color and texture combined

with the following cases:

- closed door
- narrow open door
- partly open door
- fully open door

We will try to use the Stanford 2D-3D-Semantics Dataset by Armeni et. al. that portrays over $6,000m^2$ of space, using over 70,000 RGB images and corresponding 3D models of indoor rooms and corridors [2]. With this dataset we can extend the test data for unknown environments, as well as run simulations on the 3D models to get a hint if there is a mismatch between the Door Detector and the Door Status Classifier.

6 Risks and Opportunities

6.1 SLAM

If the provided SLAM algorithm cannot locate the robot within its own map properly, the detected doors can also not be assigned to the right place. In addition, the robot will not be able to drive to its designated position, as it will not know it accurately. This might require a custom implementation of SLAM, which can be quite a challenge. Therefore, this is one of the things to be taken care of with high priority.

6.2 Door Detector

An interesting edge case for the door detector worth mentioning here is having 2 doors close by, that the robot can "overlook". We have to ensure that the robot keeps a correct sequence of actions, such that it does not disregard one of them and just picks the closest door to investigate, while loosing sight of the other one. This can easily happen because of the limited view of the camera frustum: when the robot approaches one door, the other door goes out of the view, because it stands in a "blind spot". Therefore, we have to build a queue of seen doors that should act like a memory, so that the robot does not forget what it saw previously. The SLAM map can be also of help in solving this issue and enable the robot to orient itself in space and know what doors to expect.

6.3 Door Status classifier

The door status classifier is potentially the trickiest component to be developed because it would have to be done most likely using a neural network that would process the point cloud coming from the Kinect camera. Such an approach would require a lot of data and a lot of training, which is a stochastic process as well. Consequently, the challenge would be to fine tune the black-box and gather a huge amount of data. If however, we are able to find pretrained weights from other implementations that we can adapt to our scenario, we can even consider merging this block with the door detector, such that both detection and classification happen at once/in one inference cycle of the network.

References

[1] D. Anguelov, D. Koller, E. Parker, and S. Thrun. Detecting and modeling doors with mobile robots. In *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*, volume 4, pages 3777–3784 Vol.4, April 2004.

[2] Iro Armeni, Sasha Sax, Amir Roshan Zamir, and Silvio Savarese. Joint 2d-3d-semantic data for indoor scene understanding. *CoRR*, abs/1702.01105, 2017.

[3] D. Avots, E. Lim, R. Thibaux, and S. Thrun. A probabilistic technique for simultaneous localization and door state estimation with mobile robots in dynamic environments. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 1, pages 521–526 vol.1, Sep. 2002.

[4] Zhichao Chen and S. T. Birchfield. Visual detection of lintel-occluded doors from a single image. In *2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pages 1–8, June 2008.

[5] G. Gao, M. Lauri, J. Zhang, and S. Frintrop. Saliency-guided adaptive seeding for supervoxel segmentation. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4938–4943, Sep. 2017.

[6] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2980–2988, Oct 2017.

[7] Christopher Juenemann, Anthony Corbin, and Jian Li. Robust door detection. *Final Project, Course EE368, Stanford Electrical Engineering Department, Stanford, California*, 2010.

[8] H. Lee, W. Lin, C. Huang, and Y. Huang. Wireless indoor surveillance robot. In *SICE Annual Conference 2011*, pages 2164–2169, Sep. 2011.

[9] Jing Li, S. Z. Li, Quan Pan, and Tao Yang. Illumination and motion-based video enhancement for night surveillance. In *2005 IEEE Interna-*

tional Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance, pages 169–175, Oct 2005.

[10] T. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar. Focal loss for dense object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, 2018.

[11] Iñaki Monasterio, Elena Lazkano, Iñaki Rañó, and Basilo Sierra. Learning to traverse doors using visual information. *Mathematics and Computers in Simulation*, 60(3):347 – 356, 2002. Intelligent Forecasting, Fault Diagnosis, Scheduling, and Control.

[12] M. Nieuwenhuisen, J. Stückler, and S. Behnke. Improving indoor navigation of autonomous robots by an explicit representation of doors. In *2010 IEEE International Conference on Robotics and Automation*, pages 4895–4901, May 2010.

[13] Jeremie Papon, Alexey Abramov, Markus Schoeler, and Florentin Worgotter. Voxel cloud connectivity segmentation - supervoxels for point clouds. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2013.

[14] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[15] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[16] S. Rosa, L. O. Russo, and B. Bona. Towards a ros-based autonomous cloud robotics platform for data center monitoring. In *Proceedings of the 2014 IEEE Emerging Technology and Factory Automation (ETFA)*, pages 1–8, Sep. 2014.

[17] R. B. Rusu, N. Blodow, Z. C. Marton, and M. Beetz. Close-range scene segmentation and reconstruction of 3d point cloud maps for mobile manipulation in domestic environments. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1–6, Oct 2009.

[18] Ken Schwaber and Jeff Sutherland. The scrum guide. *Scrum Alliance*, 21, 2011.

[19] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[20] R. Toris, J. Kammerl, D. V. Lu, J. Lee, O. C. Jenkins, S. Osentoski, M. Wills, and S. Chernova. Robot web tools: Efficient messaging for cloud robotics. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4530–4537, Sep. 2015.