

# Active strategies for object discovery

Phil Bradfield      Jan Fabian Schmid

February 28, 2017

## 1 Introduction

Our project is to develop an autonomous mobile system for object detection. It is desirable for an autonomous robot that it is able to interact with and use objects in its environment. However, in order to do this it has to be aware of the precise locations of objects in order to manipulate or classify them. For this task object detection can be used. An object detection algorithm provides a list of object proposals to be analyzed in further detail, meaning that we can focus our object classification systems on promising areas in an image rather than blindly searching. Such restrictions are necessary, because every mobile system bears certain limitations to its capabilities, particularly regarding processing power, energy and response time.

Usually object detection algorithms only work on single 2D images [1]. A mobile system, however, can produce a sequence of images of a scene during its exploration of the environment. This produces a number of additional challenges to the traditional 2D problem that we have to deal with. One is that at each new viewpoint a new set of object proposals is calculated, and these have to be fused with previous object proposals. The second big problem is that the robot has to decide where to move to next in order to best improve its knowledge of the environment. This is the problem of finding the next best view (NBV). For our project the goal is to analyze and evaluate multiple possible NBV-algorithms in their capability of finding good viewpoints to examine the environment.

This intermediate report is structured as follows: The next subsection of this introductory chapter presents the three application scenarios that we want to test our system in. Then some related work is described. The presented publications solved similar problems to the ones we encounter.

In Chapter 2 we will go into detail of some of the used approaches in our system.

Following, Chapter 3 will provide an overview of the information flow in our system, and some steps will be described in more detail. We also describe the current progress and speculate about possible extensions to the system.

Finally, in Chapter 4 we will indicate how we will proceed to work on the system and our anticipated timeline.



## 1.1 Application scenarios

Currently three scenarios of application with different complexity are planned.

- **Table scenario:** Objects of different size, shape and color are assembled on a table. Some objects are partially or completely occluded by other objects, therefore a single viewpoint is not sufficient to detect all objects. The mobile system has an RGB-D camera mounted at an appropriate height to look on the table. It can move freely around the table to reach different viewpoints.
- **Pre-recorded images scenario:** This scenario is only an intermediate application for testing during the development of the system. We utilize the same setting as in the table scenario, but without an autonomous system. Using an RGB-D camera on a tripod, we take pictures from different viewpoints of the table. The system can use these images for object detection and computation of an NBV.
- **Scattered objects scenario:** In this scenario objects lie scattered on the floor. The mobile system can move in between the objects to take views on different areas of the scene. This scenario is more demanding on the abilities of computing the NBV and moving the robot to the desired viewpoints, which should allow us to examine the pros and cons of different NBV-algorithms.


## 1.2 Related work

In this chapter we introduce some other approaches in which similar problems to those that we address with our system have been solved.

García and Frintrop build a framework for 3D object detection by utilizing a visual attention system [5]. They work with a recorded video stream from an RGB-D camera. The color and depth stream are separately processed. The color stream is used to create so called proto-objects, which are areas with high probability of being part of an object. These proto-objects correspond to the peaks in the calculated saliency map. Simultaneously, the image is segmented into areas of pixels with similar color. Where proto-objects overlap, they are considered to be a single proto-object and are united to a single object candidate.

At the same time, the depth stream is used to build a map of the scene. By projecting the object candidates into the 3D map they are able to label voxels as associated to a certain object in the scene. An inhibition of return

(IOR) mechanism for three dimensional scenes is used to allow the algorithm to focus on one salient region after the other.

 We can use this framework as a starting point for our project. As the framework is used for pre-recorded videos, the active sensing part that we need is missing. Apart from that, we can use the described methods for object candidate creation and the fusion of data from multiple views. For our project it may not be necessary to implement a 3D IOR map. This is because we don't have to use a continuously moving video stream to find object candidates. In our case it is possible to stop movement at our desired viewpoints and take a picture from which object candidates are determined.

Meger et al. created Curious George, a mobile system that has been successful in the past in object recognition tasks [8]. The robot is given a set of objects, and computes a visual classifier for each in a training phase. Afterwards the robot drives autonomously through an environment, simultaneously exploring and searching for the known objects. To build this map and navigate in it they have to solve the Simultaneous Localization and Mapping (SLAM) problem. This is done with GMapping, which produces a 2D occupancy grid map from laser scan and odometry data (see Section 3.2).

Curious George analyzes its environment in two steps. At first, the robot is guided by a frontier-based exploration algorithm, until a map of the whole scene is built. Then, as a second phase, the scene is analyzed in more detail. They use an attention system with a saliency map (comparable to the one in the approach of García and Frintrop) to find interesting places in the already discovered environment. Saliency is used to focus the attention of the system on relevant parts of the environment. This is necessary as many possible views are redundant or promise only a very low information gain. Therefore, visual saliency is used to evaluate the potential information gain of an area; additionally, the environmental structure is analyzed to find areas like desks where a high amount of objects is to be expected.

Even though the system of Meger et al. does object recognition instead of object detection, some parts of the system can be adopted by us. We have to perform SLAM as well, and Meger et al.'s system shows that GMapping is an applicable solution for this type of problem. Our robot has no laser range-finder to generate laser scan data, but with the depth data from Kinect we can simulate such data. The two steps of exploring is also an interesting idea that we can keep in mind for our system. And as a first approach for a simple NBV-planner we will use a frontier-based exploration algorithm.

Li et al. presented an online system for incremental scene modeling [7]. They combine the capabilities of a SLAM-based scene understanding framework with semantic segmentation and object pose estimation. Their goal was to build a multi-view recognition system that works fast and with

the necessary accuracy required for robotic application scenarios. As in our scenario, they take advantage of having a mobile camera to take multiple observations of the same scene, using them to improve recognition abilities. They train object models on a large set of partial views, and also construct a background model through a provided sequence of scene views without any objects in them. During application, a set of object hypotheses is updated with every new frame from an RGB-D camera sequence. To find corresponding spatial areas at the different viewpoints they use the estimated current camera pose from a dense SLAM method. In their results they reach improved performance in respect to single-view methods for semantic segmentation and object pose estimation.

☞ In our use case the robot doesn't get object models to learn classifiers that can be used during application. Therefore, this approach is not feasible for us. However, their work does clearly show the advantage of multi-view object recognition over single-view methods. The method of fusing multiple views through knowledge of the current camera position seems to be the standard approach, which we can be apply to solve this problem in our system.

## ☞ 2 Theoretical background

In this section we will introduce in more detail the methods that we will utilize for our system.

### 2.1 Saliency-based object discovery

We perform will 2D object discovery on RGB images with a saliency-based approach. Our method closely follows the approach of García et al. [4], which is summarized below.

In their paper they introduce a saliency-based object discovery method using RGB-D data with a late-fusion approach, which means that color and depth object candidates are computed separately and then combined. This method stands in contrast to early-fusion approaches in which the computation of object candidates is directly performed on the RGB-D data.

☞ An example of such an approach is Voxel Cloud Connectivity Segmentation [9].

The method of García et al. consists of four steps:

1. First, salient blobs in the image are calculated. A salient region is a region in the image with high contrast to the surrounding. The contrast of a specific region can be calculated in terms of several different features (e.g. the intensities of different color channels) in parallel. To find salient blobs, their first step is to calculate a saliency map with the VOCUS2 system [3], which considers red/green, blue/yellow and

intensity center-surround contrasts. Then the salient blobs are computed with seeded region growing on the local maxima of the saliency map.

2. The second step, which happens simultaneously with the salient region extraction, is to segment the image. They use four different segmentation methods:
  - Felzenszwalb and Huttenlocher’s segmentation algorithm [2], which only considers color.
  - Surface clustering searches for surface patches in the depth data.
  - The previously mentioned Voxel Cloud Connectivity Segmentation for an early-fusion approach.
  - And the late-fusion approach of separately computing candidates from color and surface segments and then combining them.
3. For candidate generation the information of salient regions and segments is brought together. An object candidate consists of multiple segments, which results in precise boundaries of the candidates. For each salient region the set of overlapping segments is selected as an object candidate.
4. The last step is to rank the computed candidates. They compare three different ranking methods:
  - A score calculated as average saliency multiplied by the square of the area of the candidate.
  - 3D convexity.
  - A range of features that are learned by a SVM.

They conclude from the comparison of the different methods that the late-fusion approach performs best. Using the SVM for candidate ranking has a much higher precision when only considering a few object candidates; however, the performance of the three different approaches is more similar for recall and it converges with a higher number of object candidates.

## 2.2 Sampling-based exploration

Surmann et al. developed a system for autonomous digitalization of 3D indoor environments [10]. This system consists of three core modules: a mapping algorithm, a view planner and a navigator to move the robot to a desired goal. Especially interesting for us is how they compute a NBV. A view planner is necessary in this scenario because many possible views promise only a very low information gain and the amount of views that can

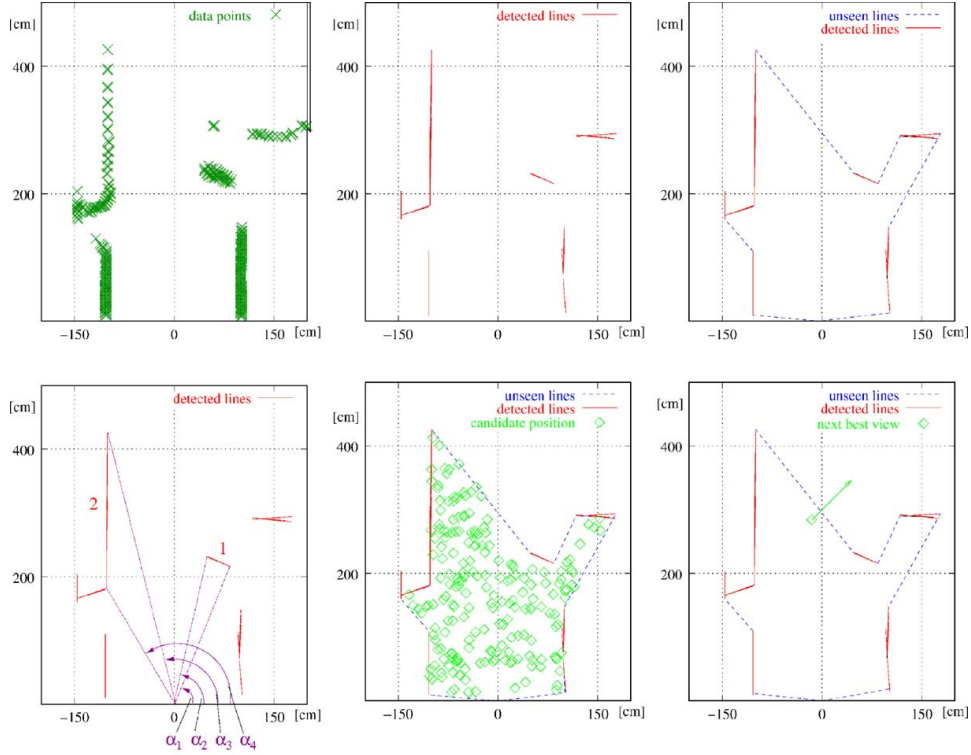


Figure 1: Next best view planning as it is done in the approach of Surmann et al. [10]

be used is limited by time and energy constraints. A visualization of the steps of their method to find the NBV can be seen in Figure 1.

To compute the next best view they generate multiple horizontal 2D layers of their 3D map of the environment. An example of one of these layers is pictured in the first frame in Figure 1. The green crosses represent single data points from the laser range finder.

They use Hough transform to find lines in the layer which represent the obstacles (e.g. walls) they already detected (frame 2).

The lines are then connected by “assumed lines” to form an estimate of the layout of the room (frame 3).

In frame 4 it is shown how the assumed lines are calculated, using the example of the assumed line between the actual lines numbered 1 and 2. The endpoints of the actual lines are sorted by their increasing polar angle counterclockwise. Actual lines with neighboring endpoints in that sorting are then connected by assumed lines.

The next step is to randomly generate possible viewpoints in the already discovered space, which are pictured as green squares in frame 5. For each



viewpoint  $p$  the information gain  $V(p)$  is calculated as a measure of how many assumed lines can be seen from it. A compromise between traveling distance and information gain is then used to decide on the next viewpoint. The score  $S(p)$  for the viewpoint  $p$  is calculated as  $S(p) = V(p) \exp(-c_1 \|r - p\|) \exp(-c_2 \|\Theta_r - \Theta_p\|)$ , where  $r$  is the position of the robot and  $\Theta_r$  its current orientation,  $p$  the position of the viewpoint and  $\Theta_p$  the orientation of the viewpoint. The terms can be weighted with the constants  $c_1$  and  $c_2$ .



The NBV is chosen as the sample with highest score.

### 3 System description and progress

A general overview of our system is given in Figure 2.

Before we start explaining the graph in detail, it is useful to have a look at the available input and the desired output of our system.

- **Input:**

- We will use a mobile platform like a Pioneer or Turtlebot, which provides odometry data from the motion sensors.
- Onto the mobile platform we mount a Kinect camera, which provides RGB-D images of the scene.

- **Output:**

- During operation of the system it is moving around the scene. Therefore, motor control can be seen as intermediate output of the system.
- The goal is to retrieve a 3D map with incorporated labels of all found object proposals.

Independent of the used application scenario the robot starts with no pre-knowledge of the scene. Therefore, we are completely dependent on the system input to infer information.

At the beginning only an initial RGB-D image of the robot viewpoint at the start is available. One processing line uses the RGB image to find any possible objects in the view; this is described in detail in Section 3.1.

Simultaneously the depth image of the Kinect camera is processed. From the depth data a point cloud is computed; this is used to simulate laser-scan data, which consists of depth data of one horizontal line in the robot view. This laser scan data is used together with the odometry data to perform simultaneous localization and mapping. This part is described in more detail in Section 3.2.

The SLAM algorithm provides a 2D occupancy grid map, which is a kind of floor plan containing information about obstacles and free space in the already discovered environment. A second output of SLAM is an

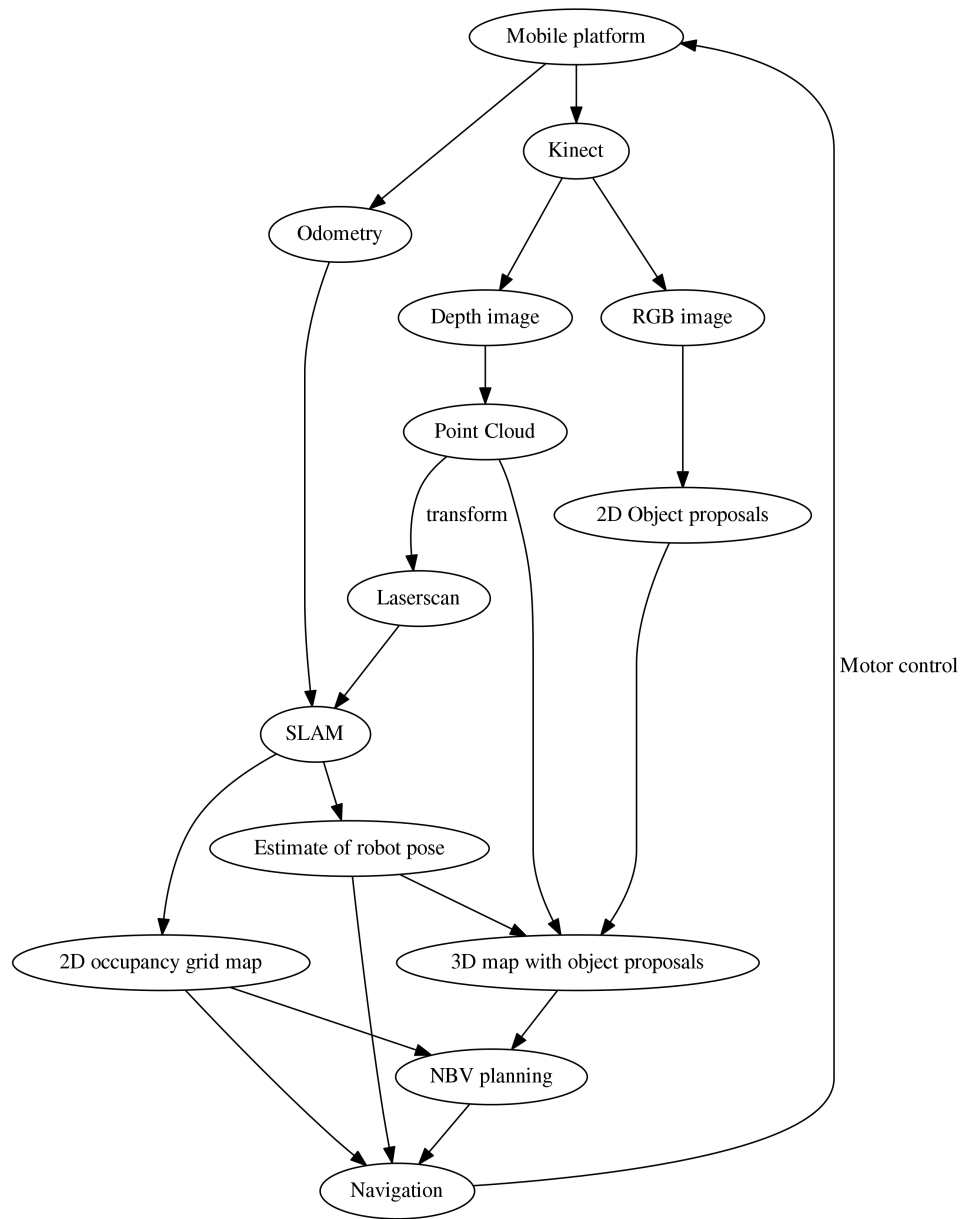


Figure 2: Overview of the system



estimate of the current robot pose. This pose is used together with the newly found object proposals from the current view and the current depth image to update the 3D map that contains a labeling of voxels to object proposals. This procedure is specified in Section 3.3.

The next step is to compute an NBV for the robot that should maximize the information gain about the environment. For this computation all data that has been gathered to this point can be used; in particular the positions of previously found object proposals and the current floor map will be used for this step. Details can be found in Section 3.4.

Once a desired view point is calculated, it is used together with the current occupancy grid map and pose estimation to plan movement of the robot to this point in space. This part is described in Section 3.5.

The output of the navigation module is the motor commands to move the system. During movement the SLAM algorithm is constantly processing the camera data to update the map and keep track of the robot position. Once the viewpoint is reached another image of the scene is taken, and the next iteration of the object detection process starts.

After describing the individual parts of our system in more detail, the last subsection of this chapter provides a short list of possible extensions to the system, that we will work on if time permits.

### 3.1 Object discovery in two dimensions

To find object candidates, we use a part of the method from García et al. described in Section 2.1. For simplicity, we will only use color candidates; the use of depth candidates that are then fused with the color candidates is a possible extension for our project. Our object discovery works as follows: First we calculate a saliency map with VOCUS2 [3], then we iterate over local maxima in the map to calculate salient blobs with seeded region growing. That means that all pixels neighboring the currently considered local maximum are taken as part of the salient region if their saliency value is at least  $x\%$  or more of the saliency value of the maximum. This process is done once with 60% and once with 70% similarity requirement. A salient region is only kept if it has at least a mean saliency value of 120 (from a maximum of 255) in order to only keep actually interesting regions. Some of the salient regions will overlap each other, if the overlap is greater than 50% only the one with higher saliency score is kept. The saliency score is calculated as mean saliency times square root of the area of the corresponding region. The segmentation is done using a C++ implementation of the Felzenszwalb and Huttenlocher algorithm [2] by Christoffer Holmstedt <sup>1</sup>. For each remaining salient region an object candidate is now calculated by finding all segments that overlap with at least 30% with the region.



<sup>1</sup><https://github.com/christofferholmstedt/opencv-wrapper-egbis>

This part of our system is already implemented as a C++ class and is working. To use it with the mobile platform, we will have to write a wrapper for ROS for it.

### 3.2 SLAM

To perform simultaneous localization and mapping we use a ROS wrapper<sup>2</sup> for OpenSlam’s GMapping algorithm [6], which is an implementation of Fast SLAM. It uses laser scan and odometry data to compute a 2D occupancy grid map of the discovered environment. The algorithm requires laser scan data from a horizontally-mounted laser range-finder. We don’t have such a laser range-finder; however, we can transform the depth image provided by the Kinect camera to laser scan data. Therefore we have to define the height for which the horizontal laser range data is computed. Because the data is used for navigation the height should be at ground level.

The ROS wrapper provides topics to retrieve the built 2D occupancy map and the current pose estimation of the robot in this map.

### 3.3 Object discovery in three dimensions

Once object proposals have been generated from the RGB image using the saliency system, the pixel-based object proposals will be translated into 3D voxel-based proposals. This will be done by using the Kinect’s depth map to project the 2D object proposals into a 3D space; each voxel in the 3D space which corresponds to part of a proposed object surface will be assigned the label of that proposed object, and a score representing the confidence with which we are labeling that voxel with each different label (i.e. a single voxel can have multiple labels, each with their own confidence score).

For each new viewpoint that the robot reaches, new 2D object proposals will be generated and these will be used to update the existing voxel labels. First, the 2D object proposals will be projected into the existing 3D space. The robot pose estimate from the SLAM system will give us the initial point and direction from which to perform the projection; combining this information with the depth map from the Kinect will give us the full transformation into the 3D space, telling us which voxels each new proposal occupies. We will then use the newly generated proposals to update the existing ones, taking our basic strategy from [5]:

- If less than 5% of the voxels in the new proposal are already labeled, then the voxels in the new proposal will be assigned to a new label and confidence score, as before.
- Otherwise, we will search over the voxels in the new proposal and find the existing label  $l$  which occurs most often (i.e. the existing proposal

---

<sup>2</sup><http://wiki.ros.org/gmapping>

which most significantly overlaps the new one). Any voxels in the new proposal which are not labeled as  $l$  will be assigned label  $l$  with an initial confidence level. For each voxel in the new proposal, all labels which are not  $l$  will have their confidence levels decreased. If a label's confidence score falls below a threshold then it is deleted.

The initial confidence score of a newly-added label will be low, as the idea of the fusion process is that proposals which truly correspond to objects will be repeated often and thus build up confidence quickly, while false proposals which are not replicated should be quickly flushed from the system in order to keep the computation manageable. The exact initial value of the confidence score, and the magnitude of the increase or decrease in confidence at each update, will need to be empirically determined.

### 3.4 Determining the next best view

After processing the image data from one viewpoint the robot has to decide where to go next. This step is the core of our system and our main research focus. The goal is to find object candidates with a high level of both precision and recall, i.e. we create the minimum possible number of candidates which do not match to objects in the scene, and we create closely matching candidates for all objects actually present in the scene. The computation of the NBV determines the strategy how to reach this goal.

Desirable properties of the NBV algorithm are:

- The computed NBVs should maximize the information gain on the scene. It is not necessary to process images from the same or similar viewpoints multiple times. Also the view should focus on regions containing objects and on regions with high uncertainty about the object candidates. If the remaining energy of the robot has a value associated with it, at some point the predicted information gain has less value than the required energy to reach the NBV. At this point our system can terminate.
- The computational cost should be reasonable. Because of the constraints of energy and time in mobile systems, the value of a very good NBV against a mediocre NBV has to be considered against the additional effort required to compute the better NBV. It is probably not appropriate if the energy or time consumption for calculating a very good NBV and moving to it is as high as calculating and moving to two mediocre NBVs. This is especially important in our use case, because exploring the whole scene is already a very important aspect of a good next viewpoint.

We want to implement and evaluate multiple NBV algorithms during

our project. We will then evaluate their ability to find objects in the scene and how long and how much energy each one took.

In the following subsections we will explain in detail the different NBV algorithms that we plan to implement.

#### 3.4.1 Frontier-based exploration

This approach is very simple, without much computation going into the NBV. The goal of this method is to find viewpoints that are suitable in order to explore the whole scene, which is something an autonomous robot has to do in a new environment anyhow. No further effort is taken to focus on objects in the environment, objects are discovered as a side-product of the exploration process. The performance of this simple approach will be used as baseline to evaluate the gain of more sophisticated approaches.

In frontier-based exploration the robot is always moving to the nearest reachable frontier. A frontier is defined as a border region between explored free space and unexplored space, and a frontier is reachable if a path from the current robot position through the known free space to the frontier position exists.

Frontier-based exploration has been implemented in a ROS-package <sup>3</sup> that we plan to use.

#### 3.4.2 Information gain approach

This NBV method is based on the approach described in Section 2.2. We plan to implement two variations of this approach: one directly using Surmann et al.’s method of calculating information gain based on the number of assumed lines a potential next viewpoint can see; another which calculates information gain based on attempting to reduce the uncertainty in our labeling. A combination of the two approaches will also be attempted.

**Unseen lines** The basic intention of Surmann et al.’s approach is to explore an indoor environment. In applying the approach to NBV, our system will sample the already discovered environment and find the viewpoints which allow a view on as many assumed lines as possible. The system will then calculate a trade-off between the potential information gain offered by a potential next viewpoint and the amount of time it would take the robot to get there. Exploration is an important aspect that has to be considered by our NBV planner, because the robot can only find all objects when it has at least looked in all general areas of the room once. In our system an occupancy grid map is computed at ground level for navigation purposes; to use the Surmann et al. approach, we compute another 2D grid map

---

<sup>3</sup>[http://wiki.ros.org/frontier\\_exploration](http://wiki.ros.org/frontier_exploration)

for the height at which the objects are located. The whole method is then performed on this 2D grid map.

**Label uncertainty** For the purposes of NBV calculation, an alternative formulation of information gain would be to look at the uncertainty of the labels in our voxel map. Areas of the map which have high uncertainty would be areas which have either not been viewed at all (which would have maximum uncertainty) or areas in which our system has previously generated contradictory labels, denoting an area which needs further investigation. As with the assumed lines approach, the system would calculate the information gain for each sampled viewpoint, and then select the next viewpoint by means of a trade-off between the viewpoint's potential information gain and its distance from the robot's current location.

**Combination** The two approaches above could also be complementary, with information gain being calculated using the sum of both measures, possibly with a weighting factor.

This part is not implemented yet.

### 3.5 Navigation

Navigation will be handled using the ROS Navigation Stack<sup>4</sup>. Once the next best view processor has decided where the robot should next observe the scene from, this target pose will be sent to the navigation stack along with the applicable co-ordinate frame transformations. The navigation stack will then plan a path to the target position and formulate the motor commands to move the robot along the path.

In the table scenario, successfully moving the robot to the target position should be relatively straightforward because we will be able to keep the robot's area of navigation clear, meaning there are no obstacles for the robot to avoid. The scattered objects scenario represents a greater challenge to the navigation system as the objects which the system is trying to detect also automatically become obstacles that it has to avoid, and thus the robot's path planning becomes more complex. Additionally, in the scattered objects scenario the robot may potentially have to search for objects in a full 360° radius around its starting position, whereas in the table scenario the robot only needs to focus on a single area of interest.

This part of the system is not implemented yet. It is expected that the effort required to implement this is low, as we are simply using existing ROS functionality and do not need to customize the navigation system ourselves in any way.

---

<sup>4</sup><http://wiki.ros.org/navigation>

### 3.6 Extensions

In the following list we present possible extensions to the core system that has been described so far. The extensions could improve the system and may be implemented if time permits, but they are not essential for its abilities.

- Depth information for the computation of object proposals. As described in Section 3.1 we only use color candidates, but in the work of García et al. we summarized in Section 2.1 they showed that the additional usage of depth information which is integrated in a late-fusion approach can significantly improve the accuracy of saliency-based object discovery.
- An extension to the assumed lines approach which uses heuristics to improve on the estimation of unseen surfaces, e.g. by assuming that a flat surface which disappears into unknown space continues to be flat and that a curved surface continues with the same curvature.
- An additional NBV planning algorithm that exploits POMDPs.

## 4 Timeline

The Gantt chart in figure 3 presents our time plan for the project. The chart contains the different tasks we still have to work on and a rough time schedule for them. The two black diamonds represent significant milestones for the project.

Our timeline starts on 27<sup>th</sup> March; this start date reflects the fact that the project is likely to be largely dormant during March due to other commitments. As the final deadline for the project is not concretely fixed, we have chosen to work towards what we judge to be the earliest likely deadline: 16<sup>th</sup> July, the final day of the teaching period of summer semester 2017. Planning to be finished by this date will allow us some contingency time, should the final agreed deadline be later than this.

**Simulation environment** In addition to implementing the system as described in section 3, for the purposes of increasing the speed of testing we will also be setting up a simulation environment using Gazebo<sup>5</sup>. Gazebo already contains modules for simulating small mobile platforms like we shall be using and also the input from a Kinect camera; we shall create application scenarios in Gazebo using the available models of the objects in our dataset. An advantage of using ROS for our system is that aside from the work of actually creating the application scenarios in Gazebo, there is very little overhead involved in creating this simulation environment. Any code,

---

<sup>5</sup><http://gazebosim.org/>

launch files etc. which we create which work in the simulation should not need any major structural changes to work on the real robot, although obviously the extra noise and uncertainty present in real world environments means significant real world testing will still be required. We plan to perform some brief confidence testing on the real world system in mid to late May, while the full transition from simulation to real world testing is scheduled for late June (see below).

The work of creating this simulation environment is currently in progress.  
Planned completion date: week ending 23<sup>rd</sup> April.

**Navigation and map building** Once the simulation environment is created, we will implement the abilities of building a 2D occupancy grid map and navigating through the space. At this point we reach our first milestone, which is to have a working simulation environment in which the robot can freely roam around and explore.

Planned completion date: week ending 7<sup>th</sup> May.

**Object discovery** The next step is to create a wrapper to integrate our existing saliency-based object detection system into ROS. The wrapper should subscribe to a topic at which RGB images are provided and then publish a topic which provides the 2D object candidates. Alongside this we can implement the fusion of multiple views which maps the 2D object proposals into a 3D data structure that is then updated from viewpoint to viewpoint. We plan to work on these two modules concurrently as changes in either one may well affect the messaging required between them.

Planned completion date: week ending 21<sup>st</sup> May.

**Frontier-based next best view planning** With the object discovery system in place, we can implement frontier-based exploration as a baseline NBV planning approach. This marks our second milestone, which is to have a fully functional system using our baseline NBV planner. At this point, we may perform some brief confidence testing on the real robot system in order to make sure that there are no large, unexpected differences between the simulation environment and the real world which we need to be aware of in implementing our planners.

Planned completion date: week ending 28<sup>th</sup> May.

**Information gain-based next best view planning** The final implementation stage is to implement our information gain-based planners. As it is likely that one of the planners will be ready before the other, this phase will in reality probably overlap with the following testing and evaluation phases, as one of us can be testing one NBV planner while the other person is still developing the second planner.

Planned completion date: week ending 18<sup>th</sup> June.

**Testing** In this phase, we will test our system on the real robot platform and make any adjustments necessary to successfully make the transition from simulation to real world operation, such as exploring different possible parameterizations.

Planned completion date: week ending 25<sup>th</sup> June.

**Evaluation** Finally, we shall perform comparative evaluations in the real world, measuring the performance of the different NBV planners, and prepare the results for presentation.

Planned completion date: week ending 16<sup>th</sup> July.



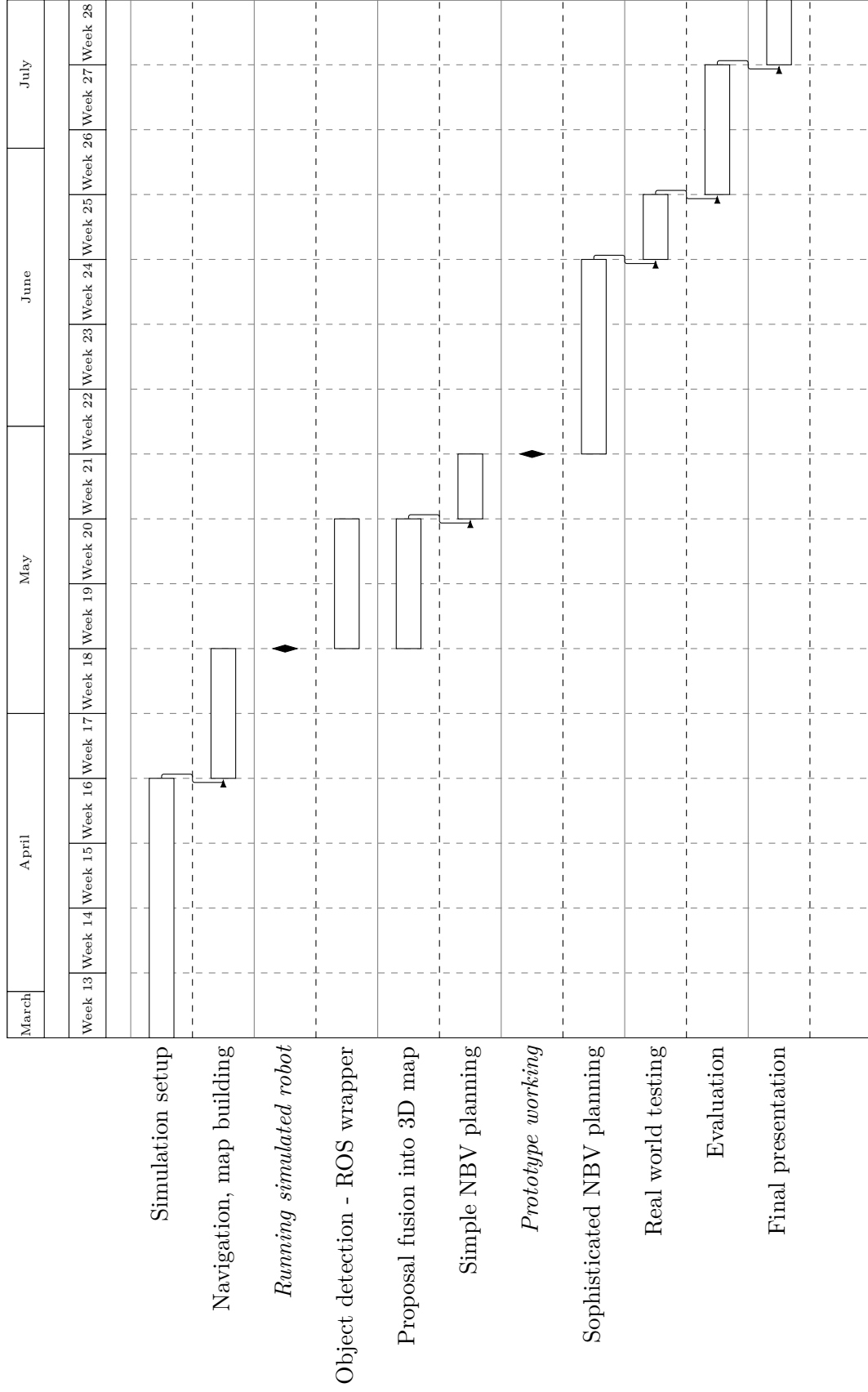


Figure 3: Planned timeline of the remainder of the project

## References

- [1] Nikolay Atanasov, Bharath Sankaran, Jerome Le Ny, George J Pappas, and Kostas Daniilidis. Nonmyopic view planning for active object classification and pose estimation. *IEEE Transactions on Robotics*, 30(5):1078–1090, 2014.
- [2] Pedro F Felzenszwalb and Daniel P Huttenlocher. Efficient graph-based image segmentation. *International journal of computer vision*, 59(2):167–181, 2004.
- [3] Simone Frintrop, Thomas Werner, and German Martin Garcia. Traditional saliency reloaded: A good old model in new shape. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 82–90, 2015.
- [4] Germán M García, Ekaterina Potapova, Thomas Werner, Michael Zillich, Markus Vincze, and Simone Frintrop. Saliency-based object discovery on **rgb-d** data with a late-fusion approach. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pages 1866–1873. IEEE, 2015.
- [5] Germán Martín García and Simone Frintrop. A computational framework for attentional **3d** object detection. In *Proc. of the Annual Conf. of the Cognitive Science Society*. Citeseer, 2013.
- [6] G. Grisetti, C. Stachniss, and W. Burgard. Improved techniques for grid mapping with **rao**-blackwellized particle filters. *IEEE Transactions on Robotics*, 23(1):34–46, Feb 2007.
- [7] Chi Li, Han Xiao, Keisuke Tateno, Federico Tombari, Nassir Navab, and Gregory D Hager. Incremental scene understanding on dense **slam**. In *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, pages 574–581. IEEE, 2016.
- [8] David Meger, Marius Muja, Scott Helmer, Ankur Gupta, Catherine Gamroth, Tomas Hoffman, Matthew Baumann, Tristram Southey, Pooyan Fazli, Walter Wohlkinger, et al. **Curious george**: An integrated visual search platform. In *Computer and Robot Vision (CRV), 2010 Canadian Conference on*, pages 107–114. IEEE, 2010.
- [9] Jeremie Papon, Alexey Abramov, Markus Schoeler, and Florentin Worgotter. Voxel cloud connectivity segmentation-supervoxels for point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2027–2034, 2013.

- [10] Hartmut Surmann, Andreas Nüchter, and Joachim Hertzberg. An autonomous mobile robot with a 3d laser range finder for 3d exploration and digitalization of indoor environments. *Robotics and Autonomous Systems*, 45(3):181–198, 2003.