Assignment

Submission Deadline: 4 weeks after receipt

1 Introduction

This placement assignment is designed to provide you with an introduction to basic math, algorithms, and information about aerial robotic perception. Your performance in this evaluation assignment will enable us to admit you to our group and give you an appropriate task within our group. In addition this assignment will introduce you to a lot of the basics, thereby setting you up for a productive visit in our group.

This assignment is composed of two sections: common and field-dependent. Problem 2 is meant to be solved by all applicants. You can choose between problem 3 (Planning), problem 4 (3D Perception), and problem 5 (Semantic Perception) based on your interests and strengths. This would determine the direction of your work during the internship, however, it will not guarantee a particular placement or task since that will depend on the current projects that are relevant at the time of your visit. The solution should be sent in a PDF format, explaining your approach in solving the problem and displaying your results clearly by the deadline. In case of any additional photos or videos, send a zip folder containing everything.

Please take this assignment seriously as it will determine if we can offer you a place in our group. The overall assignment is due 4 weeks after start. Send your assignment solution and resume to airlab0internship.bwes9@zapiermail.com. The solutions should be included in a link in the email. Do not send an attachment.

1.1 Setup Tutorial

To make your internship experience productive you need to familiarize yourself with some of the mathematical and software tools we are using. Please follow the following steps to get setup for the assignment and to prepare for your visit.

Before getting setup make you sure you have installed Ubuntu 14.04 LTS (64-bit) (`http://goo.gl/mccYb7`) on your computer to be able to complete the assignments. Familiarize yourself with Linux after installation if necessary by following this tutorial `http://goo.gl/jVL6q`.

After you have installed Linux, install the Robot Operating System (ROS),indigo version, by following the instructions at this url: `http://goo.gl/U69KBq`. After you are

set up it is important to understand the basics of ROS. Follow at least the beginner tutorials at http://goo.gl/WZ24h to get a good understanding of the development infrastructure.

2 Common Assignment

2.1 Rotations (10 points/ Recommended completion week 1)

There are several conventions to represent 3D rotations, such as rotation matrices, unit quaternions, Euler angles, etc. Each has its own advantages and disadvantages. For a useful summary of these methods see the section "Geometric primitives and transformations" in the second chapter of the book "Computer Vision: Algorithms and Applications" by R. Szeliski [1].

2.1.1 Problem

We will ask some questions and do a few practical exercises to get you more familiar with various ways to represent rotations. You don't have to implement the computation and transformation of rotations. We encourage you to use the Eigen library (see http://goo.gl/cV5LY), which is used extensively in ROS and PCL.

2.1.2 Hints

There is a vast mathematical theory built around quaternions, but don't be intimidated. Most of it is unnecessary for our purposes. The introduction in the aforementioned book should be enough for most robotics applications.

Quaternions can be represented with four numbers: a scalar w and a three-vector (x, y, z). In code, there is no agreement on whether to represent this as (w, x, y, z) or (x, y, z, w), so make sure to check.

2.1.3 Task

1. Name one advantage and one disadvantage of using Euler angles (e.g. Roll-Pitch-Yaw), unit quaternions, rotation matrices and axis/angle representations of rotations.

2. Let $R^a = (\theta^a_{roll}, \theta^a_{pitch}, \theta^a_{yaw})$ and $R^b = (\theta^b_{roll}, \theta^b_{pitch}, \theta^b_{yaw})$ be a rotation corresponding to the following roll-pitch-yaw angles:

$$\begin{aligned} \theta^a_{roll} &= 0 & \theta^b_{roll} &= -\pi/3 \\ \theta^a_{pitch} &= \pi/4 & \theta^b_{pitch} &= 0 \\ \theta^a_{yaw} &= \pi/3 & \theta^b_{yaw} &= 0 \end{aligned}$$

[1] A free pdf is available at http://szeliski.org/Book

Compute the 3×3 rotation matrices corresponding to R^a and R^b.

3. Compute the quaternions q^a and q^b equivalent to the matrices R^a and R^b. Are these quaternions a unique representation of these rotations?

4. Compute two compositions of the rotations using quaternions [2], $q^c = q^a q^b$ and $q^d = q^b q^a$. Are q^c and q^d the same?

5. Compute the *relative* rotation between q^a and q^b as $q^e = q^a (q^b)^{-1}$. Then compute the composition $q^f = q^e q^b$. Verify that q^f and q^a are the same.

Please send us your answers to the questions as a PDF document.

3 Planning

3.1 Planning Concepts (10 points/ Recommended completion week 1)

3.1.1 Problem

There are several theoretical conecpts that are important to know for planning research and this question will enable you to explore several of them. There is a vast array of planning literature, however, in our lab we mostly focus on kinodynamic motion planning with various mission objectives.

3.1.2 Hints

There are several classes of motion planning algorithms that uses different techniques to solve motion planning problems. A lot of these techniques are described in the "Planning Algorithms" book by Steven M. LaValle. http://goo.gl/yXVD7I This book gives a good introduction to the relevant background in motion planning. For these questions the most relevant chapters are 4, 13, and 14.

3.1.3 Task

Each of these questions should be answered in about 2-3 paragraphs.

1. Explain the difference between an objective and a constraint. Can one be converted to the other?

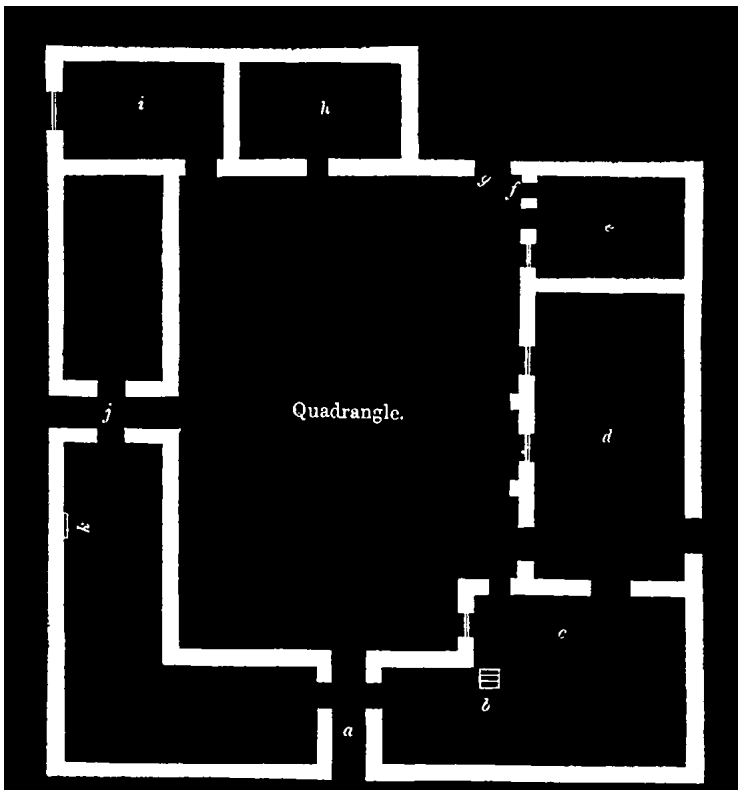
2. Explain what homotopic classes are with regards to motion planning.

3. Explain information gain and its relevance for motion planning.

4. Explain the difference between RRT* vs D* lite.

Please send us your answers to the questions as a PDF document.

[2] Note that this is the quaternion product!



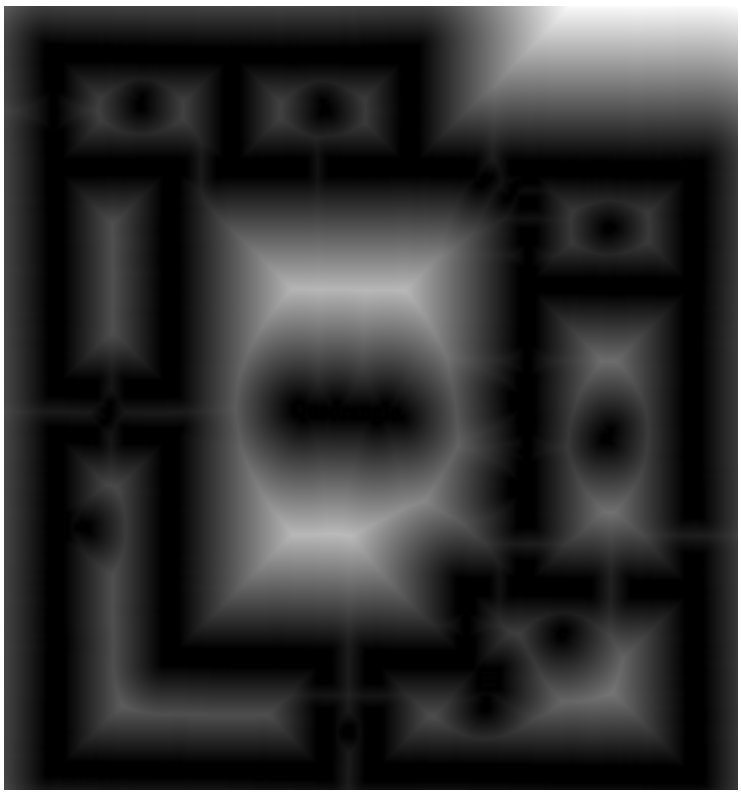
3.2 Distance Transform (20 points / Recommended completion week 2)

3.2.1 Problem

To ensure the safety of robot and the nearby objects, it is important to plan paths which maintain a safe distance from any obstacle. Therefore distance transform is a critical concept required in motion planning. A distance transform is used as a cost corresponding to safety of the vehicle while finding a path.

3.2.2 Task

The preknown map in this task holds binary information i.e. black or white. White regions are obstacles while black ones are traversable. Generate a distance transform for this map. For the output, include your fully commented source code and the output distance transform as a png image in the PDF.
NOTE: The images for this task and the next can be downloaded from http://goo.gl/1SXNQv



3.3 Motion planning (30 points / Recommended completion week 3)

3.3.1 Problem

Motion planning starts from an idea of free space and obstacles in and finds a path which the robot can safely follow in this space to go from point A to point B. Finding an optimized path requires a cost function like distance transform, time taken to reach goal on that path etc. Here, you have to find the least cost path using a given distance transform.

3.3.2 Task

The image for this task contains a distance transform depicting the distance to obstacles, where each cell contains the distance to the nearest obstacle. Darker/lower values signify closer proximity to obstacles and brighter/higher values signify that obstacles are farther. Write a program that finds the best path going from point A(144,144) to point B(910,1035) while optimizing the following:

1. **Distance to Obstacles** The planner should try to keep the robot away from obstacles to ensure safety. Minimum allowed distance to any obstacle is 10 pixels.
2. **Distance to GOAL** The planner should try to optimise the total distance travelled by the robot i.e. find shorter path to the goal.

Include your source code as well as an overlay of the generated path on the image in the assignment solution.
NOTE: pixel(x,y) implies pixel(along width,along height)

3.4 Exploration Planning (30 points / Recommended completion week 4)

3.4.1 Problem

Robots operating in uncertain environments not only have to reason about known obstacles and free space but also about the unknown space. This problem is apparent when the robot is trying to explore unknown regions to create a map. The robot has to select its actions such that the total cost of exploring the environment is minimum. A naive algorithm used to extract exploratory behavior from robots is frontier exploration (http://goo.gl/hZr53A). We have provided an implementation of the algorithm here (http://goo.gl/OIuwuf). Your task is to improve the explorative behavior of the robot.

3.4.2 Task

The implementation of the alogrithm can be found at
`demos/informationgain/information_gain_planning.m`

When you run the demo, you will notice that the robot jumps from one unexplored frontier to the other, oscillating back and forth. You should change the way frontier nodes are selected to remove this oscillatory behavior.

The setup instructions for the code are available at the repository wiki here (http://goo.gl/OIuwuf).

3.5 Extra Credit (20 points)

After you have completed all the basic steps in the assignment please show us something interesting with the implementations you have done. You can be creative here and show your strengths. Please only send a document, pictures or videos.

4 3D Perception

4.1 Plane Segmentation (30 points/ Recommended completion week 2)

4.1.1 Problem

To familiarize yourself with the ROS visualization tool (`rviz`), please follow the given tutorial http://goo.gl/eh0kt. We use Plane Segmentation of a 3D point cloud as a

motivational example to get you familiar with how ROS infrastructure might be used in real world applications.

4.1.2 Hints

Please find and download the plane_segmentation_tutorial package at http://goo.gl/vWbAK5. Download the package in a new workspace's `src` directory. To run the demo:

1. Run `catkin_make` in a terminal, with your new workspace as your working directory, and make sure the package compiles without any errors. Run `source ./devel/setup.sh`
2. Type `roscore &` to run a ROS master in the background.
3. Now type `roslaunch plane_segmentation_tutorial sample.launch` to launch the rviz and planesegmentation binary.
4. In another terminal, run `rosbag play -l -r 0.01 2013-02-21-17-59-01.bag` after changing the working directory to `[yourpath]/plane_segmentation_tutorial/data`. This plays a pre-recorded data file ("bag file") which provides the point cloud to the plane segmentation.

The demo should show a grayscale pointcloud and overlaid colored points (red, green or blue) indicating which points belong to which of the detected planes. This is displayed inside inside the `rviz` window. You may need to adjust the viewpoint.

The source file you will be updating is in `plane_segmentation_tutorial/src` directory. The code is very similar to http://goo.gl/Yu1sp. We use RANSAC for plane segmentation. Some good readings about the subject can be found in `plane_segmentation_tutorial/readings`.

4.1.3 Task

Your task is to display the normal vector arrows of the planes in the same colors as points belonging to that plane. Please send us a screenshot and your updated source file.

4.2 3D reconstruction with stereo (60 points/ Recommended completion week 3 and 4)

4.2.1 Probem

To navigate effectively, we need to give our robots an idea of the 3D structure of the world around them. A common way to achieve this is stereo vision. Stereo extracts depth information from pairs of images (usually arranged horizontally, like human eyes) with known calibration and position by triangulating the depth of each pixel in the image. Once the images are appropriately preprocessed, this simply corresponds to computing the vertical displacement for each pixel in the right image relative to the left image.

See the Szeliski textbook on Stereo for an extensive overview of this topic. "Learning OpenCV" by G. Bradski also has an accessible introduction.

4.2.2 Task

In this assignment we will give you five pairs of stereo images taken from our own flight logs, along with calibration information and the 3D poses from which each pair was taken. Your task will be to generate a point cloud from the stereo imagery using OpenCV, Eigen and PCL, similar to the one in fig. 1.

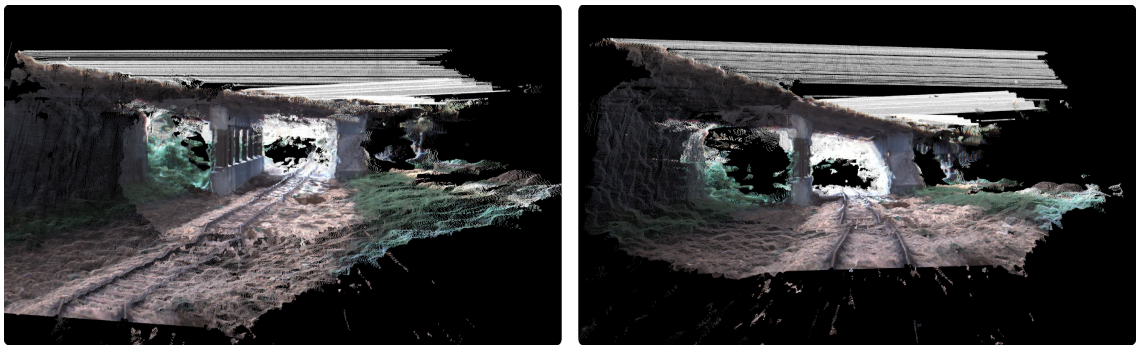


Figure 1: Example output of stereo reconstruction assignment.

To accomplish this follow these steps:

1. Browse the following documentation pages:
   - http://docs.opencv.org/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html
   - http://wiki.ros.org/image_pipeline/CameraInfo
   - http://eigen.tuxfamily.org/dox/group__TutorialGeometry.html

   This will help you understand the format and conventions of our data.

2. Clone the git repository in https://bitbucket.org/castacks/stereo_assignment. This repository contains a package with example code to load the assignment data and additional instructions. You can use this package as a starting point for your assignment.

3. Load the data into your program (This is mostly done for you in the package).

4. For each image pair, compute a disparity map. For this you may use the stereo algorithms implemented in OpenCV (e.g. `StereoSGBM`), but feel free to try other open source code (e.g. http://www.cvlibs.net/software/libelas/) or your own implementations. Note that the images have been undistorted and rectified, considerably simplifying the problem. Hint: examine the output of this step as a depth image. The 3D structure should already be visible.

5. From the calibration, disparity map and pixel positions of each pair, compute the (x, y, z) position for each pixel in the disparity map. Create a PCL PointCloud object with the XYZ points for each pair. For many pixels you might not have

disparity; simply ignore these pixels. Make sure to save the RGB information for each point in the point cloud. This will make the output easier to understand (and debug).

6. You will now have five point clouds. Save each PointCloud to a different `.pcd` file, named as `pair00.pcd`, `pair01.pcd`, etc. Then use `pcl_viewer` to visualize the point clouds. You should be able to see a reasonable 3D reconstruction of the scene. However, there might be several artifacts, and noise, specially in regions with low or ambiguous texture, such as the sky and ground. This is expected.

7. The point clouds from last step are not in the same coordinate system; they are in the coordinate system of the left camera, which is in motion. To combine these clouds into a single cloud you will have to transform them into a common coordinate system. We have given you the pose (in some global coordinate system) of the camera for each pair. Transform each point cloud into the global coordinate system using this pose and then concatenate them into a single cloud, and save it as `aligned.pcd`.

   Submit all the `.pcd` generated for this assignment. Make sure to compress them with `gzip`, `zip` or similar.

4.3 Extra Credit (20 points)

You will probably notice the 3D point clouds can be quite noisy. There are many different methods to improve the 3D reconstruction, some of them very simple, some of them very sophisticated. See e.g. http://www.cvlibs.net/datasets/kitti/eval_stereo_flow.php?benchmark=stereo for a benchmark featuring the latest advances in stereo vision. For extra credit, make the best-looking reconstruction you can. Submit the pointcloud as `extra.pcd` and briefly explain what you did. A few ideas (of varying difficulty):

- Preprocess the images so stereo matching works better, e.g. apply a gradient operator, equalize illumination, etc.
- Implement a more sophisticated disparity computation method, using e.g. the census transform or DAISY descriptors. It is OK to use a library for this.
- Postprocess the disparity images to improve the output, e.g. by detecting and discarding noisy regions in the sky.
- Fix regions without disparity by using some kind of interpolation.
- Integrate information across pairs so their 3D clouds clouds are consistent with each other.
- Use RGB information to guess which areas are smooth and which are discontinuous (edges), and use this to improve disparities.
- Any combination of the above, or anything else you can think of!

Even if whatever you tried didn't work, let us know what you tried and why you think it failed, illustrated with screenshots.

5 Semantic Perception

5.1 Machine Learning Concepts (20 points)

5.1.1 Problem

Helping robots understand the world in terms of semantic categories (such as "person", "car", "tree", etc.) is a active research topic in machine learning and computer vision.

Here we will ask a few questions focusing on basic concepts you should be familiar with for the kind of work we do in our lab.

5.1.2 Resources

There are many freely available sources of information on this topic in the web. Here we list a few, but feel free to read and research on your own.

- Computer Vision http://szeliski.org/Book/, in particular chapter 14, "Recognition".
- Elements of Statistical Learning http://statweb.stanford.edu/~tibs/ElemStatLearn/, chapter 1.
- Deep Learning http://www.iro.umontreal.ca/~bengioy/dlbook/, Part 1.

5.1.3 Questions

- What is the difference between classification and regression?
- It is common practice to divide a dataset in three subsets, usually called *training*, *testing* and *validation*. Why? What is the role of each subset?
- Suppose a researcher develops an algorithm that detects a rare type of bird in images. The researcher claims her algorithm is "99% accurate". Does this mean the problem is solved? Why or why not? What questions would you ask the researcher?
- What is the difference between unsupervised and supervised learning? What are the (dis)advantages of each?
- Suppose a researcher given the task of creating an algorithm to detects the head, hands and legs and of a person in an image. His first idea is to create a classifier for head, one for hands, and one for legs, and apply it to each pixel in the image. What problems do you see with this approach? Can you suggest some improvements?

- In the context of neural networks, what is the difference between *parameters* and *hyperparameters*?
- Can standard gradient descent be used to learn the hyperparameters of a neural network? Why or why not?

5.2 Semantic Labeling (70 points)

5.2.1 Problem

The problem in this task is *semantic segmentation*: Given an image, predict one of K predefined labels for each pixel. To see a recent notable work in this area, see http://www.cs.berkeley.edu/~jonlong/long_shelhamer_fcn.pdf.

Here we will approach the problem using supervised machine learning. We provide a small set of images with manually created pixelwise labels. The objective is to create a pipeline to assign a label for each pixel.

5.2.2 Data

Please download the dataset from Google Drive: https://drive.google.com/file/d/0B3JLY7es_MEXc3VYanp6SDdST3c/view?usp=sharing. Please do not distribute.

- `left/` contains RGB images.
- `disp/` contains disparity images (see stereo assignment task). They are pretty noisy. Format is `png` 16 bit.
- `label/` contains label images. The labels were manually drawn and are far from perfect.

There is `readme.md` in the tarball – it is out of date, you can safely ignore it.

For the labelled image, each pixel has an integer value indicating its label. The mapping is listed in listing 1.

```
{ 'building': 1,
    'dirt': 2,
    'foliage': 3,
    'grass': 4,
    'human': 5,
    'pole': 6,
    'rails': 7,
    'road': 8,
    'sign': 9,
    'sky': 10 }
```

Listing 1: Mapping of labels to numbers.

A value of 0 means unlabelled and can be ignored. Note that since these values are so small, if you open the images in an image viewer you will only see a very dim image.

5.2.3 Task

The task is to create and evaluate a pipeline for semantic segmentation. We will leave most of the implementation details up to you, but your solution must meet the following constraints:

It must be implemented in Python, C++, or both. These are the primary languages we use for development in our lab.

There must be an evaluation of pixelwise accuracy using four-fold cross-validation. That is, for four randomized partitions of the images, use 3/4 of the data for training and 1/4 for testing. Report the training accuracy in each fold. This is the bare minimum – you are encouraged to perform more exhaustive evaluations, and report different evaluation metrics (e.g. F1, precision/recall, confusion matrices, etc.).

A report. Deliver a report briefly describing your approach and the results of your evaluation. The exact length is up to you; 3-4 pages is sufficient.

5.2.4 Notes

Desirable attributes of your solution(s) include:

- robustness
- originality
- efficiency
- accuracy
- thoroughness in evaluation

The balance between these attributes is up to you. High accuracy, by itself is not the goal – we would rather see a creative approach with a good report, rather than a very accurate classifier that is exclusively using off-the-shelf code with a bad report.

Note that there is disparity (depth) data, which is relatively unusual compared to the most standard setting in which only RGB data is available. You are encouraged to use this data to improve accuracy.

It's ok to use third party code for stuff like extracting superpixels, learning neural nets, or HOG features. On the other hand, if your solution is just third party code then that is not very impressive.