ROBOTIC MAPPING AND GUIDING USERS THROUGH NATURAL LANGUAGE INSTRUCTIONS

Likhitha Surapaneni

Master of Technology Thesis
June 2019

International Institute of Information Technology, Bangalore

ROBOTIC MAPPING AND GUIDING USERS THROUGH NATURAL LANGUAGE INSTRUCTIONS

Submitted to International Institute of Information Technology,
Bangalore
in Partial Fulfillment of
the Requirements for the Award of
Master of Technology

by

Likhitha Surapaneni
IMT2014030

International Institute of Information Technology, Bangalore
June 2019

Dedicated to

the spirit of perseverance

Thesis Certificate

This is to certify that the thesis titled **Robotic mapping and Guiding Users through Natural Language Instructions** submitted to the International Institute of Information Technology, Bangalore, for the award of the degree of **Master of Technology** is a bona fide record of the research work done by **Likhitha Surapaneni**, **IMT2014030**, under my supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Prof.G Srinivasaraghavan

Bengaluru,

The 10th of June, 2019.

ROBOTIC MAPPING AND GUIDING USERS THROUGH NATURAL LANGUAGE INSTRUCTIONS

Abstract

Artificial intelligence has made quite an entrance into our daily lives in the form of mobile applications, smart appliances, voice assistants giving both an easier and personalized approach to dealing with technology. In our work, we attempt to establish smart assistance in indoor spaces by focusing on mapping and navigation in indoor environments. Although there has been a lot of prior work in simultaneous localization and mapping, popularly called SLAM, in order to map an environment using sensor data extending it to navigation is still a novel field of research. Our work contains two parts, the first part emphasizes on devising an architecture for representing an indoor environment, houses in our case. We use images from several 2D locations that span the entire house. Object detection and other feature detection algorithms are used to construct a map. The second part uses this map to generate natural language instructions to navigate from one place to another in the environment. The instructions are generated by first constructing a path between two queried locations within the house, extracting key landmarks (as objects seen during the navigation) and then using a natural language generator to construct sentences from the landmarks and navigation actions. Given a starting point and a destination, the system therefore produces a sequence of simple natural language instructions to take someone from the starting point to the destination.

Acknowledgements

Firstly, I would like to express my gratitude to my thesis advisor Prof. G.Srinivasaraghavan for his constant guidance and patience. At times, when the task seemed impossible, he was always there helping me come out with a way to make it possible. This work would not have been the way it is without his mentorship.

I would also like to thank Prof. Dinesh Babu Jayagopi and Prof. Shrisha Rao for inspiring me to take up research. I would like to specially thank Vineet and others responsible for setting up GPU resources.

I would like to thank my friends for cheering me up during my stressful times. Lastly, I cannot thank my family enough for encouraging me and supporting me throughout the endeavour of writing thesis. This journey had been much easier because of them.

Contents

List of Figures

List of Tables

List of Abbreviations

AMT	Amazon Mechanical Turk
BLEU	Bilingual evaluation understudy
CNN	Convolutional Neural Network
CVPR	Conference on Computer Vision and Pattern Recognition
DoG	Difference of Gaussians
DTAM	Dense Tracking and Mapping
EM	Expectation Maximization
IoU	Intersection over Union
LSD	Large-scale direct
NLG	Natural Language Generation
NMT	Neural Machine Translation
PTAM	Parallel Tracking and Mapping
R2R	Room-to-Room
RCNN	Region Convolutional Neural Network
RNN	Recurrent Neural Network
RoI	Region of Interest
RPN	Region-Proposal Network
SIFT	Scale-Invariant Feature Transform
SLAM	Simultaneous Localization and Mapping

US	United States
vSLAM	Visual Simultaneous Localization and Mapping

CHAPTER 1

INTRODUCTION

1.1 Multimodal Learning

Perception, communication and understanding are some of the long-held challenges of artificial Intelligence in robotics. Humans have the natural ability to communicate with the environment and interpret information using different sensory organs. Each sensory input when reasoned gives a different yet complimentary understanding of the environment.

Modality is defined as a phenomenon of experience. Having more than one such mode in the process of learning is termed as *Multimodal Learning*. Historically, audio-visual speech recognition has gained immense importance due to their supplementary support in noisy scenarios. With the onset of personal computers and the internet, multimedia content has proliferated and there was a need to index the digital content giving rise to newer research problems like video summarization. It was not until recently that the research was focused in replicating the human multimodal behaviour in social interactions. There are however several challenges when dealing with multimodal data like dealing with the heterogeneity of the data by exploiting complementarity and redundancy, getting multiple modalities into a common space suitable for executing a particular task. [1]

1.2 Vision and Language

This sphere of research combines extracted visual data like images with aspects of language and social cognition to understand the environment.The emergence of new benchmark datasets for several tasks has resulted in substantial progress in vision and language understanding.With the advent of deep learning architectures, building an end-to-end architecture for incorporating both vision and language has become seamless.

Logically, there are two modules to solve this problem - image-based module which captures the features and nuances in an image and a language-based module which translates the former features into a natural sentence. The contribution of attention mechanisms has greatly helped in combining heterogeneous data. The following are a few such tasks in the intersection of vision and language.

1.2.1 Image Captioning

The task of describing an image in the form of a sentence is precisely the goal of image captioning. Such a problem consists of extracting features from the image as well as grounding them to language. This takes images as input and captions as outputs.

1.2.2 Visual Question Answering

A visual question answering system takes an image and a free-form, open-ended, natural language question about the image as an input and outputs a natural language answer.

1.2.3 Vision and Language Navigation

This task has been introduced recently in the CVPR conference 2018. Combining vision and language with action is the motivation behind this task as an attempt to create a resemblance with how humans interact with the real world. Creating an environment with real world simulator like Matterport3D Simulator, given a starting point along with an instruction describing a destination, the goal is to generate a path using artificial intelligence. [2]

1.3 Our Problem Statement

Outdoor navigation systems have completely changed the way humans commute. In a tech-savvy world we live in today, with applications like Google Maps, travelling to unknown places is a quotidian phenomenon. However, the same is not true for indoor navigation. Indoor spaces like malls, museums, houses and corporate spaces still largely rely on humans memorizing the knowledge about space. A person who is new to such spaces has to either access the map about the space or consult another human who has already consumed that knowledge. Vision and language navigation is one such attempt to enable indoor navigation. While understanding indoor spaces can help humans to navigate better, it also creates a potential for personal robotics.

We, in our work, have focused on mapping indoor spaces, specifically houses and generated natural language instructions which help in navigation from one location to other.

1.4 Main contribution

Our two main contributions are

1. **Mapping**

   All the nodes present in the house are reasoned and connected to each other using the algorithms in the project, hence generating a map that represents the entire house.

2. **Generating instructions**

   Given a start and end location within the house, descriptive natural language instructions are generated by accessing the previously developed map.

The reason for designing the two modules separately is to allow mapping to be used with other related tasks like navigation, augmented reality and so on. This work aims at enhancing the vision and language navigation by providing automated instructions in contrast to the prior used manual generation of instructions.

1.5 Organisation of remaining chapters

In Chapter 2, we look at the relevance of AI in robotics and some related work for the thesis

Next, in Chapter 3, we discuss the first module of the project, robotic mapping and how it is tackled in the current project.

Chapter 4 talks about a novel direction in the field of robotics, indoor navigation assistance in the form of natural language instructions.

The final results are shown in the Chapter 5 followed by conclusions and future work in Chapter 6.

CHAPTER 2

BACKGROUND

This chapter gives a brief introduction to robotics and how artificial intelligence helped in enhancing the field of robotics.

2.1 Robotics and AI

AI and robotics had come into existence almost at the same time and hence share a common history. Initially, although there was no distinction between the two, a clear separation was visible in the 70s when robotics was confined to industrial applications while AI has seen its presence in the machines of everyday lives.

However, robotics has come a long way from developing sensorimotor functions that work in a carefully simulated environment to a real world environment. The introduction of computer vision, natural language processing, speech recognition and expert systems has strengthened the lost relationship between the two. According to [3], the idea behind combining robotics with AI is to enhance its interaction with the dynamic world, ultimately optimizing its autonomy. The level of autonomy is described as its ability to predict future either in planning a task or in interacting with the environment. For the purpose of the thesis, let us limit our discussion to how artificial intelligence has changed the subfield of indoor robot mapping and navigation.

2.2 Robot Mapping

Mapping is defined as the process of connecting several nodes within the environment with an aim to provide a complete representation of the indoor environment

There are several representations for the data while storing as given in [4]:

1. Metric maps: This category of maps are built by keeping geometry of the environment in mind. It uses numeric values like distance, angle to represent the environment. Due to its verbose nature, it is mostly accurate and tackles the data association problem. However, it consumes large memory and is not in a form that can be easily communicated to humans.
2. Topological maps: In this representation, the primary focus is on the connections between different spaces in the environment. While it consumes lesser memory, it is not detailed enough to convey descriptive information about the environment.
3. Semantic maps: This representation gives semantic representation of the environment, hence allowing higher level tasks.
4. Hybrid maps: A combination of two or more maps is more effective and stores less space.

Another categorization of maps is based on the coordinate space. Primarily, there are two - world-centric(allocentric) vs robot-centric(egocentric) and they are defined in [5] as given below.

1. Allocentric: This system of spatial representation encodes information about an object with respect to its neighbouring object.
2. Egocentric: This system of spatial representation encodes information about an object with respect to itself.

To build a map, robots make use of sensors to capture information about the environment. One of the main challenges in robotic mapping as discussed in [6] is the nature of measurement noise. In robotic mapping, measurement noise is statistically dependent i.e noise gets accumulated and affects the measurements in future. A deviation from the base direction while capturing images can affect the direction of movement tremendously after a point.

The second concern with robotic mapping is with the dimensionality of data required to represent the environment. While the dimensionality should be sufficient to differentiate different scenes in the environment, it should also not compromise with the memory and processing power.

The third and major concern in a typical robotic mapping problem is the problem of data association. Upon hitting an already seen location, the robot has to realize and associate the current scene with the already existing map and hence take care of the loops.

Environments are dynamic in nature. One challenge in robotic mapping is to stay updated with the changes in the environment and modify the map accordingly.

2.2.1 Traditional Methods

Most of the approaches trying to solve the problem of robotic mapping look at a broader problem of SLAM. SLAM is the technique of learning a map and locating the position of the robot simultaneously as calculation of each component is dependent on the other. Using probabilistic techniques, both mapping and localization would be estimated together.

[6] gives a brief description of how to use probabilistic methods for robotic mapping. Bayes rule solves temporal estimation problems like map building by its extension to

Bayes filters. The unknown state is referred to as x_t. Assuming that data is collected in the sequence of sensor measurements and motion commands, z_t and u_t represent the sensor measurement taken at time t and robot motion in the interval of [t-1,t). z^t and u^t are refer to data leading up to time t. Applying Bayes rule to this, we get

$$p(x_t|z^t,u^t) = \eta p(z_t|x_t)\int p(x_t|u_t,x_{t-1})p(x_{t-1}|z_{t-1},u_{t-1})dx_{t-1} \qquad \text{(Eqn 2.1)}$$

The time per update is constant, allowing to extend it to Bayes filters. The unknown quantities with respect to robotic mapping consist two quantities - robot pose and map.

$$p(s_t,m_t|z^t,u^t) = \eta p(z_t|s_t,m_t)\int p(s_t,m_t|u_t,s_{t-1},m_{t-1})p(s_{t-1},m_{t-1}|z_{t-1},u_{t-1})ds_{t-1}dm_{t-1} \qquad \text{(Eqn 2.2)}$$

Here, an assumption is made that the environment is static hence omitting t in m_t. Now, the equation becomes

$$p(s_t,m|z^t,u^t) = \eta p(z_t|s_t,m)\int p(s_t,m_t|u_t,s_{t-1})p(s_{t-1},m|z_{t-1},u_{t-1})ds_{t-1} \qquad \text{(Eqn 2.3)}$$

Here, $p(z|s,m)$ is referred to as the perceptual model and $p(s|u,s_t)$ is referred to as the motion model.

Kalman Filter

Kalman filter refers to an algorithm that estimates unknown parameter variables of a system based on indirect, uncertain or noisy data observed. Their applications include navigation, guiding and control, time series analysis and robotic motion planning.

Kalman filters represent posteriors in Bayes filters using Gaussians. Gaussians are unimodal distributions represented by parameter values. The Kalman filter relies on three basic assumptions:

1. Motion model must be linear with added Gaussian noise.
2. Perceptual model also must be linear with added Gaussian noise.
3. Initial uncertainty is Gaussian

In motion model, the current robot pose s_t is non-linearly dependent on u_t and s_{t-1} through non-linear trigonometric functions. However, applying Taylor series expansion to the motion module can accommodate the non-linearities and such an approximation works well for most robotic vehicles. The result of this is The extended Kalman filter in which the first assumption is satisfied giving us a linear equation with added Gaussian noise.

$$p(x|u^t, x^{t-1}) = Ax^{t-1} + Bu + \varepsilon_{control} \qquad \text{(Eqn 2.4)}$$

A and B are linear mappings from states x_{t-1} and u and $\varepsilon_{control}$ is a normal distributed measurement noise with zero mean and $\sum_{control}$‘ co-variance.

Similarly, Taylor series expansion is applied to the perceptual model and we get

$$p(z|x) = Cx + \varepsilon_{measure} \qquad \text{(Eqn 2.5)}$$

Here C is a linear mapping matrix and $\varepsilon_{measure}$ is a normal distributed measurement noise with zero mean and $\sum_{measure}$ covariance Now, these equations transformed from the Bayesian filters represent the standard Kalman filter equations and hence could be solved.

One of the most important limitations with Kalman filters is the assumption that measurement noise $\varepsilon_{measure}$ is Gaussian. An environment with two indistinguishable landmarks induces multimodal distribution over possible robot poses, hence suffering from data association problem. However, Kalman filters calculate a full posterior over maps m in an online fashion.

Expectation Maximization

Expectation maximization is the technique of finding parameter values in the presence of latent variables. EM algorithms works in two steps - first, expectation step where posterior over robot poses is calculated followed by a maximization step used to calculate the most likely map given the poses. In order to find the map at i+1 iteration, we maximize the expectation over the joint log likelihood of data d^t and robot's path till time t, s^t and can write the equation as

$$m^{[i+1]} = argmax_m E_s^t[logp(d^t, s^t | m^{[i]}, d^t)] \qquad \text{(Eqn 2.6)}$$

Since logarithm is a monotonic function, maximising logarithm is equivalent to maximizing likelihood. Part of the likelihood is the robot's path s^t which is unknown in mapping. Hence, the above equation can be understood as expectation of likelihoods over all possible paths the robot might have taken.

$$m^{[i+1]} = argmax_m \sum_{\tau} \int p(s_\tau | m^{[i]}, d^t) logp(z^t | s_\tau, m) ds_\tau \qquad \text{(Eqn 2.7)}$$

Here, the probability, $p(s_\tau | m^{[i]}, d^t)$ is the expectation step. Then by fixing them, we maximize the log-likelihood of sensor measurements, $logp(z^t | s_\tau, m)$. As there is no closed form solution, it comes down representing the map using finite number of locations.

Unlike Kalman filters, EM technique involves calculating the most likely map, they cannot generate maps incrementally. However, they are able to solve the data association problem [6]

vSLAM

Recently, there has been a lot of discussion in visual SLAM , using cameras as the sensor with their simple configurations yet higher technical difficulties. vSLAM consists of five main tasks as given in [7]

- Initialization : The global coordinate system is defined and part of the environment is initialized to use it as the initial map
- Tracking: The map constructed so far is used to estimate the current robot's pose
- Mapping: The map is expanded as the camera explores unknown regions
- Re-localization: When the tracking fails due to fast motion or disturbances, camera's pose is recomputed with respect to the pose.
- Map optimization: This is done to correct the accumulative estimation error.

The following are several algorithms discussed in [7] depending on the methodologies applied:

1. Feature-based: Methods which track and map deature points present in the images.
2. Direct methods: In a feature-less and texture-less images, the whole image is used for tracking and mapping
3. RGB-D camera based methods: With introduction of low-cost RGB-D sensors, algorithms that take both image and depth as inputs are proposed.

Some of the prior works in vSLAM include Mono-SLAM, PTAM, DTAM, LSD-SLAM and SLAM++.

2.3 Natural Language Generation

NLG is defined by [8] as 'the sub-field of artificial intelligence and computational linguistics that is concerned with the construction of computer systems than can produce understandable texts in English or other human languages from some underlying non-linguistic representation of information' .

With the recent interest in Neural Networks and advancement in hardware, Neural Networks have gained considerable importance in most of the artificial intelligence tasks. Neural networks are trained to learn representations at different abstraction levels by exploiting backpropagation hence making them dense, low-level and distributed [9] and this makes them potentially significant for grammatical and semantic generalisations. [10, 11, 12] . They have gained success in sequential modelling tasks using feed-forward networks[13, 14], log-bilinear [15] and recurrent neural networks[16]. Their main advantage over traditional language models is their handing of varying sequence lengths, keeping it in low-dimensional representation and avoiding data sparseness.

CHAPTER 3

ROBOTIC MAPPING - OUR APPROACH

While robotic mapping has been a traditional research problem, mapping and generating instructions is a novel attempt in the field of robotics. We have devised the solution in an intuitive manner, keeping in mind how humans figure out the same problem.

We have used a metric-topological-semantic hybrid map and an egocentric spatial view for data collection. Unlike the traditional methods, we do not bother about localization as the images are collected from fixed discrete locations along with their coordinates. This assumption eliminates the problem of looping, however stitching of images is still an interesting problem.

3.1 Dataset Description

Mapping an environment requires some kind of sensor data representing the environment. With the advancement in research in vision, we are able to use cameras as sensors to produce comprehensive image data hence reducing the possibilty of noise and uncertainty unlike lasers used in traditional approaches.

3.1.1 Matterport3D Dataset

Recently, there has been a lot of research in understanding indoor scenes in order to achieve several computer vision tasks like augmented reality and personal robotics. This has led to a demand in RGB datasets that could capture indoor scenes. However, most of the datasets have fewer images or have limited scene coverage and often the images are accompanied with noise. Datasets like NYUv2 , SUN RGB-D, ScanNet although provide a reasonable input for training, they lack in providing comprehensive set of viewpoints and multiple paths in a single scene.

In contrast, Matterport3D dataset [17] encompasses various types of buildings like houses, apartments, hotels, offices and churches of varying sizes and complexities. The dataset contains a total of 1,94,400 RGB-D images from 90 different building scans. 2D locations are spanned across the entire floor of the scan with a separation of 2.25 m. From each 2D location, data is acquired by rotating a three-colour three-depth camera point slightly up, horizontal and slightly down. Figure FC3.1 shows the images for one 2D location.

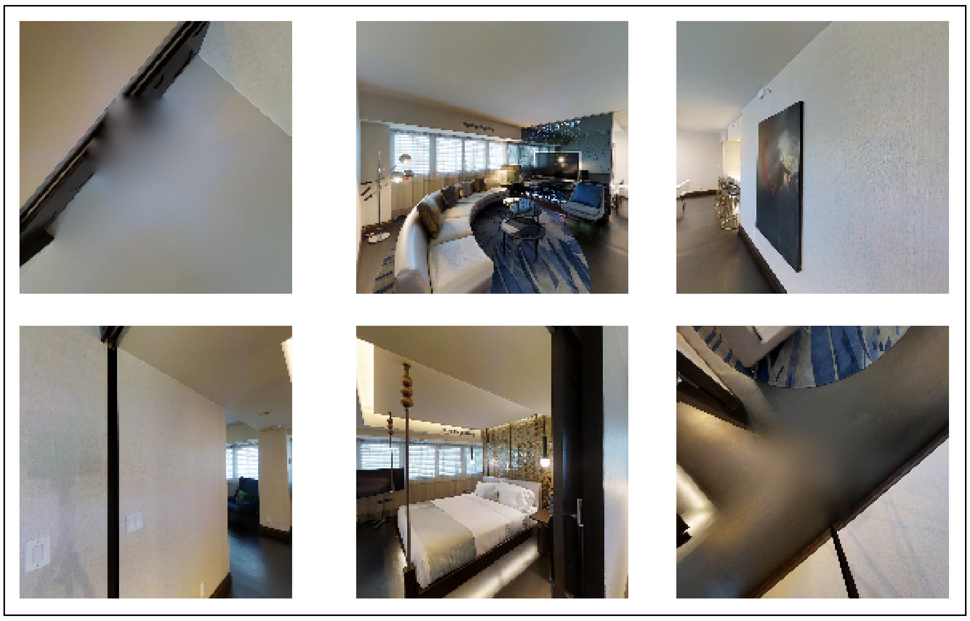


Figure FC3.1: Images in the dataset from a 2D location

3.2 Terms and Definitions

3.2.1 Scan

All the data acquired pertaining to a single house/office is called a **scan** .

3.2.2 Viewpoint

A 2D location in an indoor scan which acts as a reference point for capturing images is defined as a **viewpoint**.

Each viewpoint in the image consists of 6 non-overlapping panoramic images in the dataset. We generate 8 images as shown in FC3.3 from these panoramic images starting from a global direction, say north, and rotate 360 degrees to complete a panoramic view from a viewpoint. Figure FC3.2 gives a clearer picture.

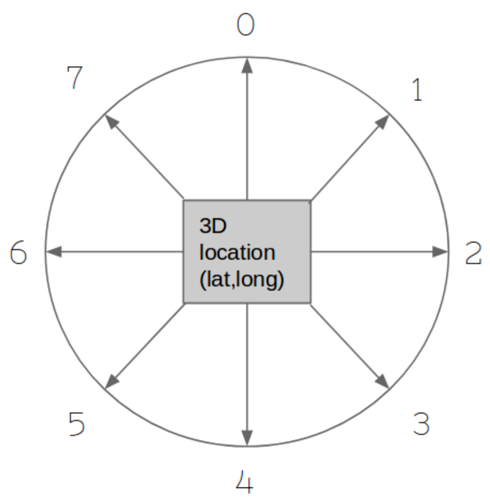


Figure FC3.2: Viewpoint diagram

3.2.3 Navigable Locations

Two viewpoints are said to be **navigable** if in real world, one viewpoint is the immediate traversable viewpoint from the other. This assumption is derived from the fact that walls and kitchen counters are impenetrable.They are also at a reasonable distance far away from each other.

3.2.4 Reachability graph

A graph consisting of all the viewpoints in the house scan along with their the immediate navigable viewpoint locations is called a **reachability graph**.

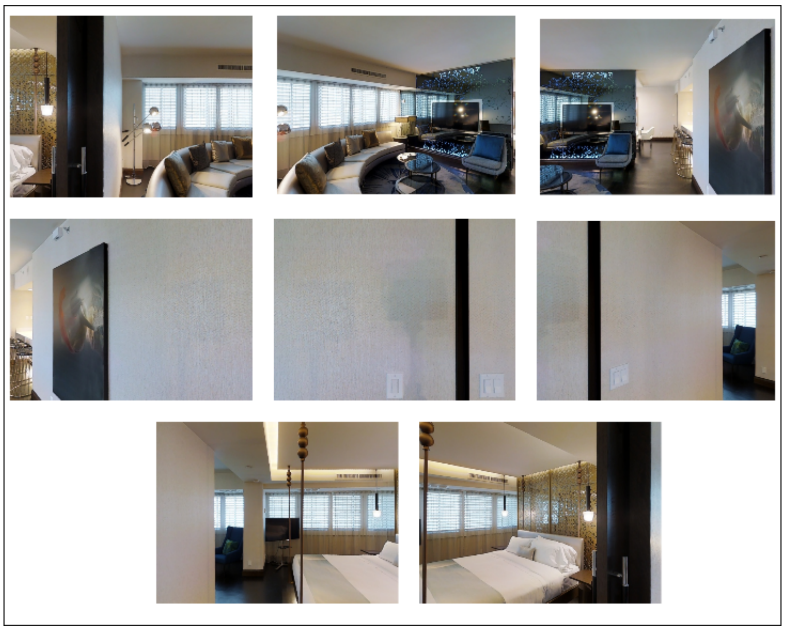


Figure FC3.3: Images in the dataset from a viewpoint after pre-processing

Figure FC3.3 is a preprocessing step on images from Figure FC3.1. The top and bottom views aare discarded and the remaining images are sent through the simulator to get images at angle 45 degress apart from each other, hence arranging them as in FC3.2

3.3 Solution Architecture

The following figure FC3.4 describes the approach for this module in brief.

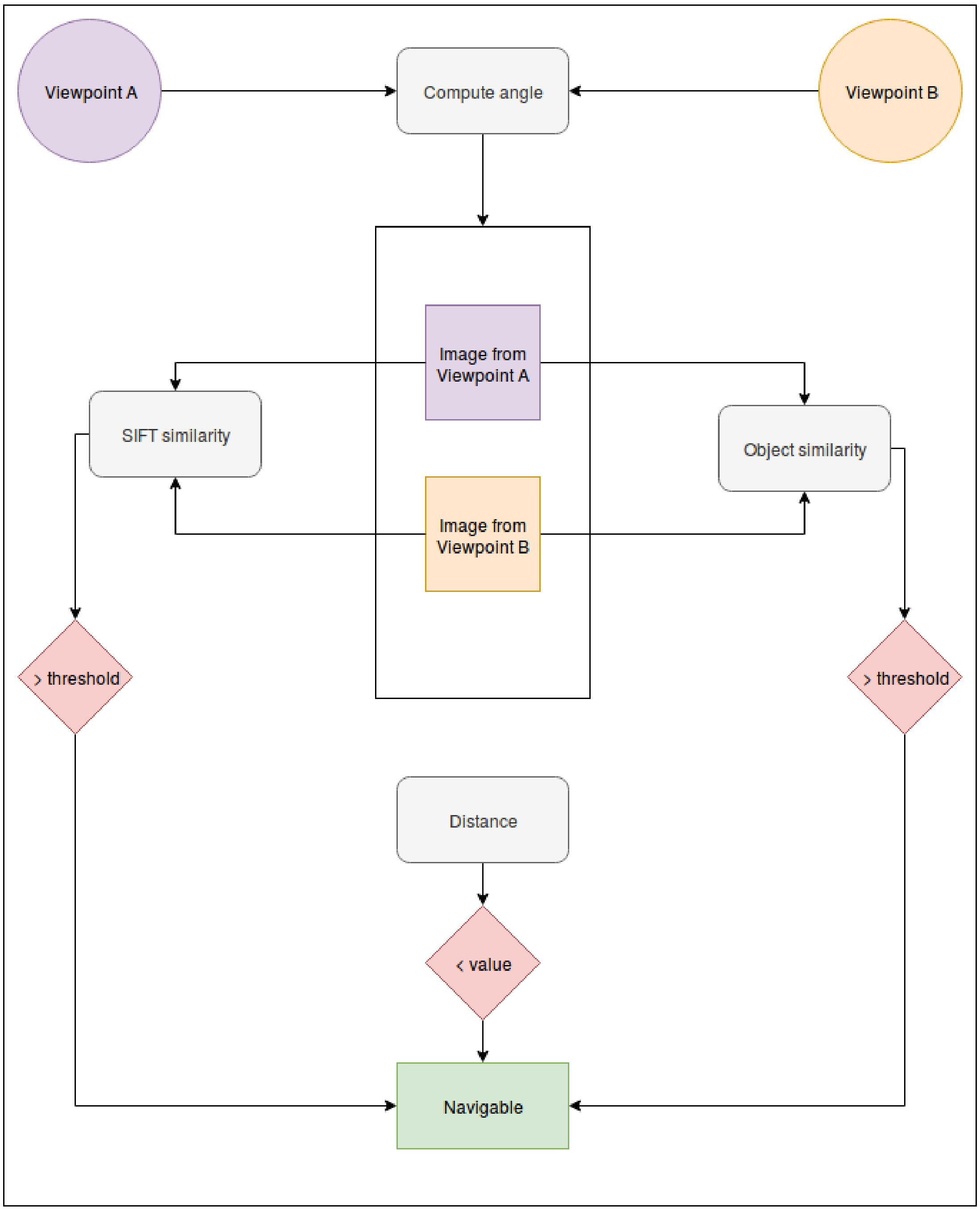


Figure FC3.4: Architecture of Robotic Mapping

3.4 Modules

3.4.1 Feature extraction

It is important to convert the given images into a form that the machine can understand. Feature extraction is the transformation of original data to values that are informative and non-redundant.

Feature extraction has two significant uses in this work. Firstly, it helps in obtaining a similarity between two images within a viewpoint as well as across viewpoints. Finding a common set of features between images within an image ensures that the images around a viewpoint create a panoramic understanding of objects present in the image. On the other hand, finding intersection between images across viewpoints ultimately helps us in achieving the actual task of mapping. Secondly, feature extraction helps in producing descriptive instructions, primarily defined by objects and directions in the image. In this module, we adopt two ways of extracting features.

Object detection using MaskRCNN

A framework for instance segmentation, MaskRCNN has an end-to-end architecture to detect semantic objects and give a high-level understanding of an image.

MS COCO 2017 [18] stuff and objects datasets are used in training the model. Once the model is trained with the subclasses mentioned in the Appendix A, it is used to predict classes and bounding boxes of the objects in an image. This process is repeated for all the 8 images of a viewpoint and likewise for all the viewpoints present in the scan.

Mask RCNN [19] is an extension of Faster RCNN [20] for instance segmentation.

Instance segmentation is the process of detecting the objects while precisely segmenting each instance. Faster RCNN has two stages - the first stage is a Region Proposal Network which proposes candidate object bounding boxes. The second stage which is the ROoPool extracts features from each candidate bounding box and performs bounding box regression and object detection. Mask RCNN adopts a similar two-stage approach. In the second stage, it additionally outputs masks for RoI. Also, the RoIPool operation is replaced by RoIAlign in the second stage. Now, let us discuss each module in FC3.5 in detail.

- **CNN:** The role of Convolutional Neural Network is to extract features from the image data useful for region proposals. Features are calculated by forward propagation through several convolutional layers followed by fully connected layers.

- **Region Proposal Network:** An RPN takes an image as input and outputs a set of rectangular object proposals. To generate these, a network is passed over the feature map of the last shared convolutional layer in the previous module. This network is fully connected to an n x n spatial window of the feature map. Each sliding window is converted into a low dimensional vector and each vector is separately connected to two fully connected layers, regression and classification layers. At each sliding window, k region proposals are parameterized corresponding to k reference boxes called anchors depending on size and aspect ratio. The network is trained with a binary loss for an anchor being an object. An anchor which has the highest intersection over union with the ground truth box and anchors with IoU greater than 0.7 are given a positive label. The anchors with IoU less than 0.3 are given a negative label and the rest are discarded for training. Now, a multi-task loss is used to train the region proposal network. The loss is given as

$$L(p_i, t_i) = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_{i^*}) + \frac{\lambda}{N_{reg}} \sum_i p_{i^*} L_{reg}(t_i, t_{i^*}) \qquad \text{(Eqn 3.1)}$$

Here, i denotes the index of the anchor in the mini-batch. p_{i^*} and t_{i^*} are the ground-truths of the i^{th} anchor being an object (0 or 1) and its box coordinates. p_i is the predicted probability and t_i is the predicted bounding box.

- **RoI Align:** A smaller and fixed-size feature map is computed from region proposals. Unlike RoI Pooling in Faster RCNN, RoI align uses bi-linear interpolation to compute the exact value of input features of four regularly sampled locations in the RoI bin and then aggregate the values to find the value in the reduced feature map.

The rest of the architecture focuses on getting a mask layer, classification layer and a regression layer and the total loss is given by

$$L = L_{cls} + L_{reg} + L_{mask} \qquad \text{(Eqn 3.2)}$$

where L_{cls} is classification loss for object detection and L_{reg} is the loss for bounding box prediction where a multi-task loss is adopted, L_{mask} is defined as average binary cross entropy loss between the predicted and the ground-truth mask.

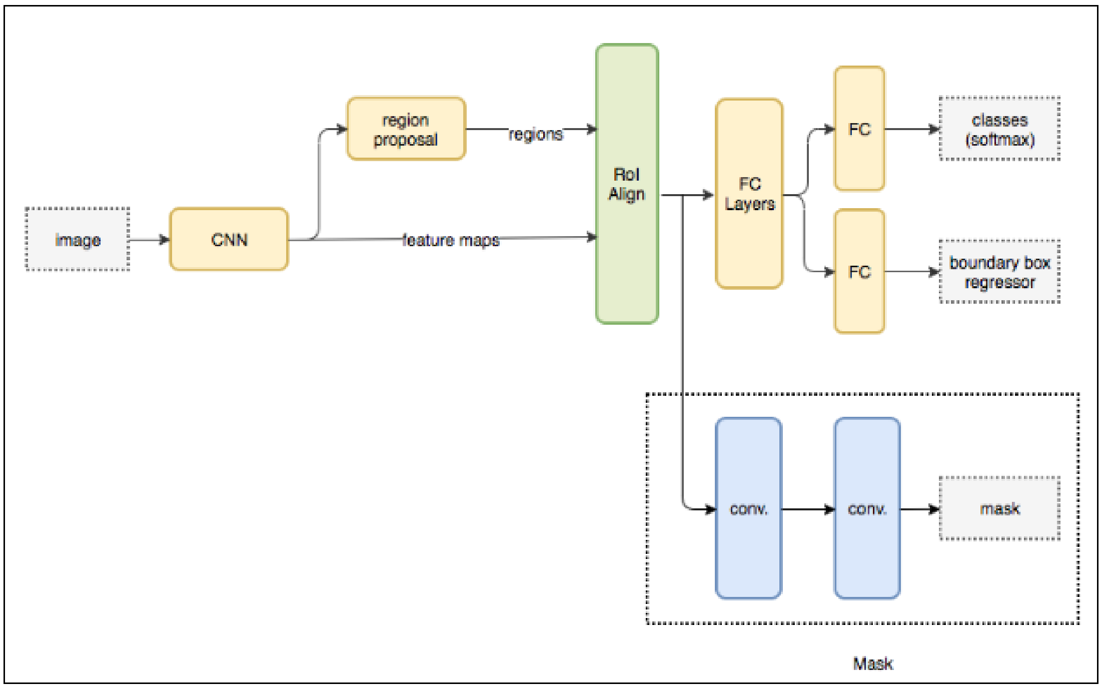


Figure FC3.5: MaskRCNN Architecture

Once we obtain different objects present in the image, the objects are sorted from left to right based on their bounding boxes to ensure that the 8 images corresponding to a viewpoint form a continuous sphere. The object detection is not sufficient to establish uniqueness of an image and hence we need other low-level features to uniquely represent an image in the form of features.

SIFT features

SIFT [21] is developed by Lowe. Low level features that are invariant to image-translation, scaling, rotation and partially invariant to illumination, affine and 3D projections. These low-level features are captured for a set of images which serve as a metric to decide similarity between images. There are four main modules in SIFT:

- Space-scale extrema detection: Identifying keypoints with different scale might require filters of different windows. In order to tackle this, SIFT uses Difference of Gaussians for blob detection with σ as the scaling factor Difference of Gaussian is obtained by Gaussian blurring an image with two different σ and taking the difference between them. Once these DoG are calculated, each pixel value is compared with its eight neighboring pixels and the nine pixels corresponding to it in the previous and next scales.

- Keypoint localization: Here, two things are taken care. One, it eliminates low-contrast keypoint and two, it discards edges and leaves us with strong keypoints. A Taylor series expansion of space scale at extrema is calculated and intensity lower than a threshold value is eliminated. For the edges, a 2X2 Heissan matrix is used to detect and remove the corners from our keypoints.

- Orientation assignment: It makes keypoints invariant to rotation by assigning orientation. A neighbourhood around the keypoint is taken and gradient magnitude and direction are calculated. An orientation histogram of 36 bins covering 360 degrees is set up . It is filled by taking the gradient magnitude for sample points around a keypoint. As peaks correspond to dominant directions in local gradients, the highest peak in the histogram is taken and any peak greater than 80% is taken to calculate the orientation.

- Keypoint descriptor: In this, a descriptor for a local image region is computed which is highly distinctive yet invariant to illumination, scale etc. A 16 X 16 neighbourhood around the keypoint is taken and is divided into 16 sub-blocks of 4 X 4 size. For each sub-block, an 8 bin orientation histogram is created.

3.4.2 Generating Reachability Graph

To generate a reachability graph, we connect different viewpoints within the scan using the conditions for navigability.

In order to do this, we first collect the metric data between two viewpoints by calculating the distance and direction in terms of angle by fixing a base direction.

We define a boolean variable called object similarity by calculating the intersection of objects between the two viewpoints and this is explained in the following algorithm. The images within the viewpoint are selected based on their proximity to the direction considered. The first set of images are selected by computing the angle between the first viewpoint and the second viewpoint in the direction of the second viewpoint. An a similarity is calculated between the selected images. This process is repeated for a direction from the second viewpoint in the direction of the first viewpoint. If only when both the similarities are greater than a threshold, 0.5 in this case, the object similarity outputs True. Algorithm 1 gives a pseudo-code for computing object similarity.

Algorithm 1 Object Similarity Measure

Input: $viewpoint_1, viewpoint_2$

```
1:  procedure RESOLVEOBJECTS($image$,angle)
2:      return $objects, rois$
3:  procedure OBJECTSIMILARITY($viewpoint_1$,$viewpoint_2$)
4:      angle = COMPUTEANGLE($viewpoint_1, viewpoint_2$)
5:      $image_1, image_2$ = GETIMAGES($viewpoint_1, viewpoint_2$,angle)
6:      for a in [angle,reverse-angle] do
7:          $o_1$,$roi_1$ = RESOLVEOBJECTS($image_1$,a)
8:          $o_2$,$roi_2$ = RESOLVEOBJECTS($image_2$,a)
9:          similarity = intersection($o_1$,$o_2$)/min(len($o_1$),len($o_2$))
10:         if similarity $<$ threshold then
11:             return False
12:     return True
```

Similarly, a boolean variable, SIFT similarity is computed. Images are selected in the direction of the second viewpoint from the first viewpoint and the opposite direction.

SIFT features are computed and are sent to Brute Force Matcher [22] which takes the descriptor from one and matches with all the features in the second descriptor using a distance metric. We get the k best matches and finally select matches lesser than a value, 0.7 in our case. The length of those matches. The following algorithm 2 details the approach.

Algorithm 2 SIFT Similarity Measure

Input: $image_1$, $image_2$

```
1: procedure SIFTFEATURES(image)
2:     return keypoints, descriptors
3: procedure SIFTSIMILARITY(image,image_2)
4:     k1,d1 = SIFTFEATURES(image_1)
5:     k2,d2 = SIFTFEATURES(image_2)
6:     matches = BruteForceMatcher(d1,d2)
7:     list = []
8:     for m,n in matches do
9:         if m < 0.7*n then
10:            list.add(m)
11:    if len(list) > 20 then
12:        return True
13:    return False
```

Conditions for Navigability

There are 3 conditions for deciding the navigability of two viewpoints A and B

1. The distance between the 2D locations of the viewpoints A and B should be less than or equal to the average minimum distance d between two 2D locations during data collection. In this case, d is 2.25 m
2. The object similarity between A and B is True

3. The sift similarity between A and B is True

All the three conditions must be satisfied in order to qualify a viewpoint as a navigable location. The viewpoints satisfy a symmetric relation of navigability, i.e if B is a navigable location of A, then A will be a navigable location of B.

CHAPTER 4

INDOOR NAVIGATION ASSISTANT - OUR APPROACH

Most of the approaches in this field have addressed robotic mapping. Extending this module to a novel task like building an indoor navigataion assistant is detailed in this chapter.

4.1 Dataset Description

4.1.1 R2R dataset

This dataset [2] is aimed at computing navigation paths for simulators using Matterport3D dataset. It consists of instructions, start and goal locations along with other useful data which are considered as input in vision and language navigation task. Altogether, there are 21,567 instructions with an average length of 29.

Matterport3D region annotations are used to sample a start pose. For a pair of start point and goal location, a shortest path is calculated from a navigation graph. This navigation graph is constructed by ray-tracing between viewpoints in Matterport3D scene meshes and then manually verified. The path so constructed is discarded if it is shorter than 5m or has less than 4 or more than 6 edges. The average length of the path is 10m. In total, 7189 paths are sampled. For generating instructions, AMT workers in US who were screened on their previous performances were consulted. They were provided with an interactive 3D WebGL environment and for each start and goal locations in the path, they were asked to write instructions by following a certain path, not necessarily the shortest.

4.2 Terms and Definitions

4.2.1 Path

Given two viewpoints A and B, a **path** is defined as the series of viewpoints starting from A and followed by intermediate nodes encountered in the process of reaching B in the reachability graph. If B is a navigable location of A, then path length of A and B is 2, having only A and B in its path.

4.2.2 Keyword

From the objects extracted in the images of viewpoints, significant descriptive objects and their directions are selected as **keywords** which are later used in guiding.

4.2.3 Instruction

A natural language sentence describing the path descriptively and which is unambiguous and grammatically correct is an **instruction**.

4.3 Architecture of Indoor Navigation Assistant

The following figure FC4.1 briefly describes the solution approach.

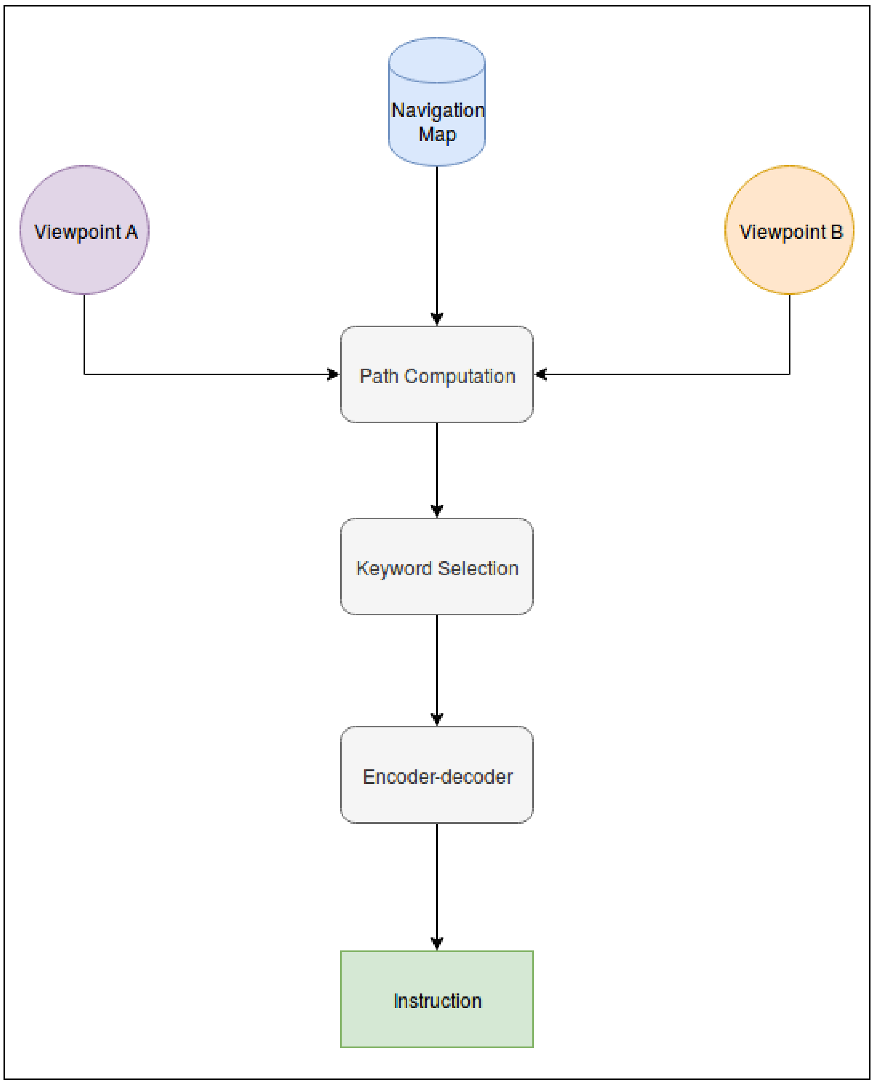


Figure FC4.1: Architecture for Indoor Navigation Assistant

4.4 Modules

4.4.1 Path Computation

Dijkstra's algorithm

This is an algorithm proposed by Dijkstra for finding the shortest path between two nodes, often used in road networks. We used this algorithm in our reachability graph for path computation as we would like to create an efficient navigation assistant.

The viewpoints in our reachability graph are equivalent to the nodes in the graph. There is an undirected edge between two navigable locations with edge weight equal to 1. Now, let us look at the algorithm 3.

Algorithm 3 Path Computation

Input: $viewpoints, viewpoint_1, viewpoint_2$

```
1: procedure DIJKSTRA($viewpoints, viewpoint_1$):
2:     for v in viewpoints do
3:         dist[v] = infinity
4:         previous[v] = undefined
5:     dist[$viewpoint_1$] = 0
6:     Q = set of all nodes in the graph
7:     while Q is not empty do
8:         u = node in Q with the smallest dist
9:         remove u from Q
10:        for each neighbour v of u do
11:            temp = dist[u] + distBetween(u, v)
12:            if t thenemp < dist[v]
13:                dist[v] = temp
14:                previous[v] = u
15:    return previous
16: procedure COMPUTEPATH($viewpoint_1, viewpoint_2$)
17:    graph = DIJKSTRA($viewpoints, viewpoint_1$)
18:    node = $viewpoint_1$
19:    path = [node]
20:    while node is not $viewpoint_2$ do
21:        node = prev[node]
22:        path.append(node)
23:    return path
```

Keyword Selection

For keyword selection, we look into objects that are significant and distinctive. Hence, we compute two components - **prominence** and **relevance**. While prominence captures the significance of an object in an image, relevance ensures it has a higher importance in describing the path computed.

We relate prominence to area of the bounding box of an object and relevance to how close the objects are located in the direction of the path. For each viewpoint. We then find the dominating set by running a pareto-front algorithm on the objects based on their area and angle from the path. Pareto-front algorithm in 4 gives all the objects o_i in the set such that the area is greater and the angle is lesser than the values from the remaining set. Out of the selected objects, we discard the extremes and randomly output a candidate from the remaining set along with its direction. We repeat this process for all the viewpoints in the path. The idea behind taking a random choice is from the assumption that instructions are non-deterministic.

Algorithm 4 Keyword selection

Input: *path*

```
1: procedure GETKEYWORDS($path$)
2:     candidate = []
3:     for viewpoint in $path$ do
4:         objects,roi = RESOLVEOBJECTS(VIEWPOINT)
5:         areaOfObjects = COMPUTEAREA(objects,roi)
6:         angleOfObjects = COMPUTEANGLE(objects,roi)
7:         dominatingSet = PARETO-FRONT(areaOfObjects,angleOfObjects)
8:         keyObject = random(dominatingSet)
9:         candidate.add(keyObject)
10:    return candidate
```

4.4.2 Translating Keywords to Instructions

Once we have a pair of key object and its direction for each viewpoint, we join them to form a phrase of words that could potentially become an instruction. However, it is still not grammatically correct and in order to make it a proper sentence, we adopt the concept of NLG as a Neural Machine Translation problem using an encoder-decoder architecture.

Neural Machine Translation

NMT attempts to build and train a single, large neural network that reads a sentence and outputs a correct translation [23]. In this, keyword phrases are interpreted as the source language whereas their corresponding grammatically correct sentences compose the target language.

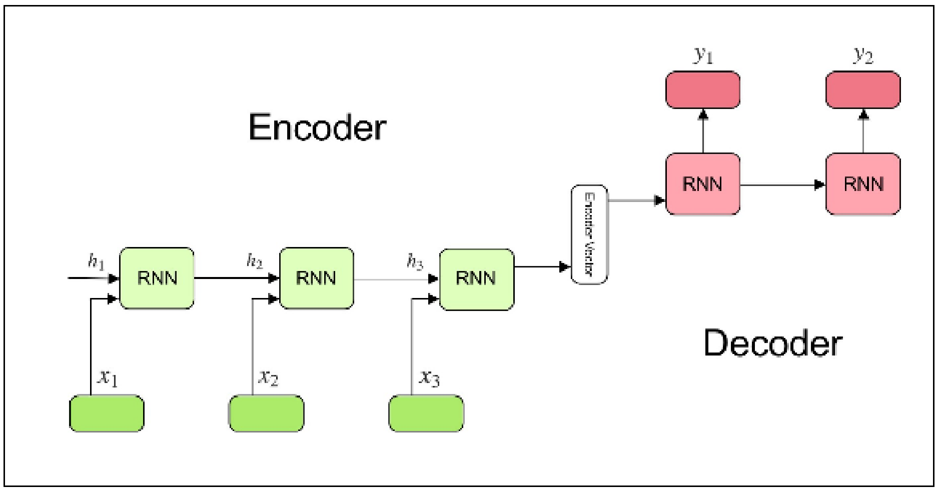


Figure FC4.2: Encoder-Decoder Architecture

The Encoder-Decoder framework [24] uses an RNN to encode a variable-length sentence which goes as one of the inputs of the decoder. Figure FC4.2 gives a complete picture of this network.

- **Encoder:** The encoder is an RNN that starts reading the first word of the input sentence and results in a hidden vector each time that goes as input to the next cell in the network along with the next word in the sentence. By the end of the sentence, we obtain a hidden representation summarizing the entire sentence.
- **Decoder:** Decoder takes the output of the encoder as the initial input and predicts the target sentence with the help of an RNN.
- **Loss function:** Both the encoder and decoder are trained on perplexity loss. Perplexity is a measurement of how well the probability models predict a target sentence. It is an exponential of negative log-likelihood.
- **Evaluation metric:** A unigram BLEU score is calculated for the predicted and target sentences.

We use R2R dataset to train our module. We first take the natural language instructions corresponding to each path of the selected scan. We send these sentences through a preprocessor that retains only nouns and prepositions in the sentences and discards the words corresponding to the remaining parts of speech. We consider the original sentences as the target language and the transformed sentences as the source language. We then train the machine translator using this data and with the settings given in Appendix B. The output thus obtained is our desired instruction.

CHAPTER 5

RESULTS

In this chapter, we state and analyze the results for the discussed modules.

5.1 Robotic mapping

We adopt a qualitative approach to evaluate the results in robotic mapping. While the map obtained is comparable with the ground-truth navigation graph, they are not exactly similar. Since the motive is not to find its similarity with an existing map but to give a path that has navigable nodes from source to destination, we adopt a qualitative approach.

We start from an initial viewpoint and follow the nodes in the predicted map and see if there is a similarity between the nodes from start to destination. We give a label of 1 to the path if the path leading to the destination has nodes which are naturally traversable from the previous viewpoint and 0 otherwise.

Out of 30 paths, it is found that 22 had scored a label of 1 and 8 have got 0.

- It is found that object similarity and SIFT similarity are sometimes not sufficient to establish similarity between two images.

- Some images are captured using a camera of height very low or very high of the camera. This does not give room for analyzing objects present in the surroundings.

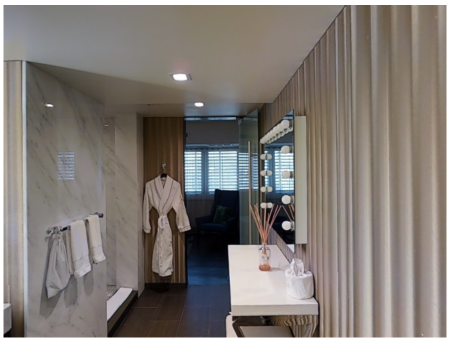


Figure FC5.1: Error Image - 1

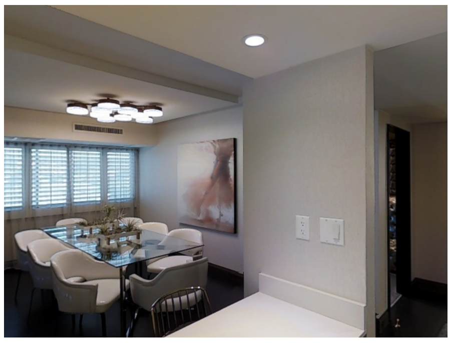


Figure FC5.2: Error Image - 2

As we can see in the FC5.1, we can spot a flowerpot and a platform that looks like a table, hence confusing the predictor while comparing this image with any image in the living room. In the figure, FC5.2, we can see that the counter is not visible in the first half of the image, hence making the predictor assume that there is no obstruction.

5.2 Indoor Navigation Assistant

We analyze the module by comparing the output of this module, i.e the instructions with how close they are to the target instructions. The following table gives the perplexity error and accuracy for different epochs with both the ground-truth map and our map.

Table TC5.1: Results from Neural Machine Translation

Epoch-Number	Training Loss	Validation loss	Accuracy
21	4.3181	18.5284	0.3871
53	1.2978	163.5220	0.3830
72	1.1461	163.5220	0.3691

From the table TC5.1, it is clear that while training loss deceases with increase in the number of epochs, validation loss also increases which means the model is trying to overfit on the data. However, we notice that accuracy is less irrespective of the number of epochs. This suggests that we could improve our loss function with evaluation techniques.

CHAPTER 6

CONCLUSIONS AND FUTURE WORK

Leveraging the knowledge of existing discrete locations, we have attempted to generate a map for one house and produced instructions to traverse within the house. This work is an attempt to investigate the possibility of creating a framework that could provide us with an indoor assistant and the results suggest that such a system is feasible with refined training and evaluation methods. The datasets used in this work are not directly sent as input but had to undergo a lot of modifications to suit the purpose of the task, this work hence emphasizes the need for creating a dataset that is aimed to make this task simpler. Some of the future works include

- Improving the mapping techniques and evaluating the framework for multiple houses.
- Creating datasets that aim at achieving this task for supervised learning.
- Extending this work to simultaneous localization and mapping
- Expanding the scope to other indoor spaces like museums, malls and offices.

The trajectory of the research in this field promises a future where we would be using robots not just like machines but more like assistants.

Bibliography

[1] Tadas Baltrusaitis, Chaitanya Ahuja, and Louis-Philippe Morency. Multimodal machine learning: A survey and taxonomy. *IEEE Trans. Pattern Anal. Mach. Intell.*, 41(2):423–443, 2019.

[2] Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian D. Reid, Stephen Gould, and Anton van den Hengel. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. *CoRR*, abs/1711.07280, 2017.

[3] Javier Andréu Pérez, Fani Deligianni, Daniele Ravì, and Guang-Zhong Yang. Artificial intelligence and robotics. *CoRR*, abs/1803.10813, 2018.

[4] Chuho Yi, Seungdo Jeong, and Jungwon Cho. Map representation for robots. *Smart CR*, 2(1):18–27, 2012.

[5] Mental Imagery and Human-Computer Interaction Lab. Allocentric vs. Egocentric Spatial Processing. `$http://www.nmr.mgh.harvard.edu/mkozhevnlab/?page_id=308$`. [Online; accessed 8-June-2019].

[6] Sebastian Thrun. Exploring artificial intelligence in the new millennium. chapter Robotic Mapping: A Survey, pages 1–35. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2003.

[7] Takafumi Taketomi, Hideaki Uchiyama, and Sei Ikeda. Visual SLAM algorithms: a survey from 2010 to 2016. *IPSJ Trans. Computer Vision and Applications*, 9:16, 2017.

[8] Helmut Horacek. Building natural language generation systems - ehud reiter and robert dale (eds.), university of aberdeen and macquarie university, cambridge university press, 2000, ISBN 0-521-62036-8. *Artificial Intelligence in Medicine*, 22(3):277–280, 2001.

[9] Yann LeCun, Yoshua Bengio, and Geoffrey E. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[10] Christopher J. C. Burges, Léon Bottou, Zoubin Ghahramani, and Kilian Q. Weinberger, editors. *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, 2013.

[11] Julia Hockenmaier and Sebastian Riedel, editors. *Proceedings of the Seventeenth Conference on Computational Natural Language Learning, CoNLL 2013, Sofia, Bulgaria, August 8-9, 2013*. ACL, 2013.

[12] Alessandro Moschitti, Bo Pang, and Walter Daelemans, editors. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*. ACL, 2014.

[13] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C. Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. *CoRR*, abs/1502.03044, 2015.

[14] Holger Schwenk and Jean-Luc Gauvain. Training neural network language models on very large corpora. In *HLT/EMNLP 2005, Human Language Technology Con-*

ference and Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference, 6-8 October 2005, Vancouver, British Columbia, Canada, pages 201–208, 2005.

[15] Andriy Mnih and Geoffrey E. Hinton. Three new graphical models for statistical language modelling. In *Machine Learning, Proceedings of the Twenty-Fourth International Conference (ICML 2007), Corvallis, Oregon, USA, June 20-24, 2007*, pages 641–648, 2007.

[16] Tomas Mikolov, Martin Karafiát, Lukás Burget, Jan Cernocký, and Sanjeev Khudanpur. Recurrent neural network based language model. In *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September 26-30, 2010*, pages 1045–1048, 2010.

[17] Angel X. Chang, Angela Dai, Thomas A. Funkhouser, Maciej Halber, Matthias Nießner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3d: Learning from RGB-D data in indoor environments. In *2017 International Conference on 3D Vision, 3DV 2017, Qingdao, China, October 10-12, 2017*, pages 667–676, 2017.

[18] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014.

[19] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask R-CNN. *CoRR*, abs/1703.06870, 2017.

[20] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR*, abs/1506.01497, 2015.

[21] OpenCV-Python Tutorials. Introduction to SIFT (Scale-Invariant Feature Transform). `$https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_feature2d/py_sift_intro/py_sift_intro.html$`. [Online; accessed 08-June-2019].

[22] OpenCV-Python Tutorials. Feature Matching. `$https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_feature2d/py_matcher/py_matcher.html$`. [Online; accessed 08-June-2019].

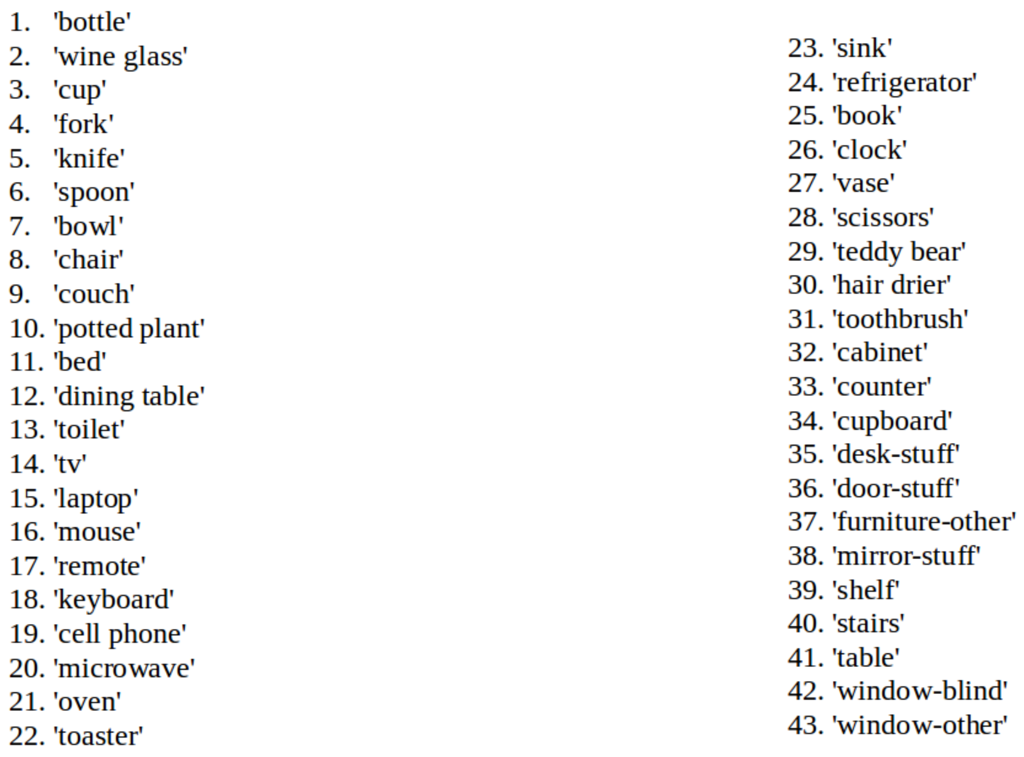
[23] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[24] Ilya Sutskever, James Martens, and Geoffrey E. Hinton. Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA, June 28 - July 2, 2011*, pages 1017–1024, 2011.

APPENDIX A

APPENDIX: SETTINGS IN MASKRCNN MODEL

The classes used for training the model are the following figure FA1.1



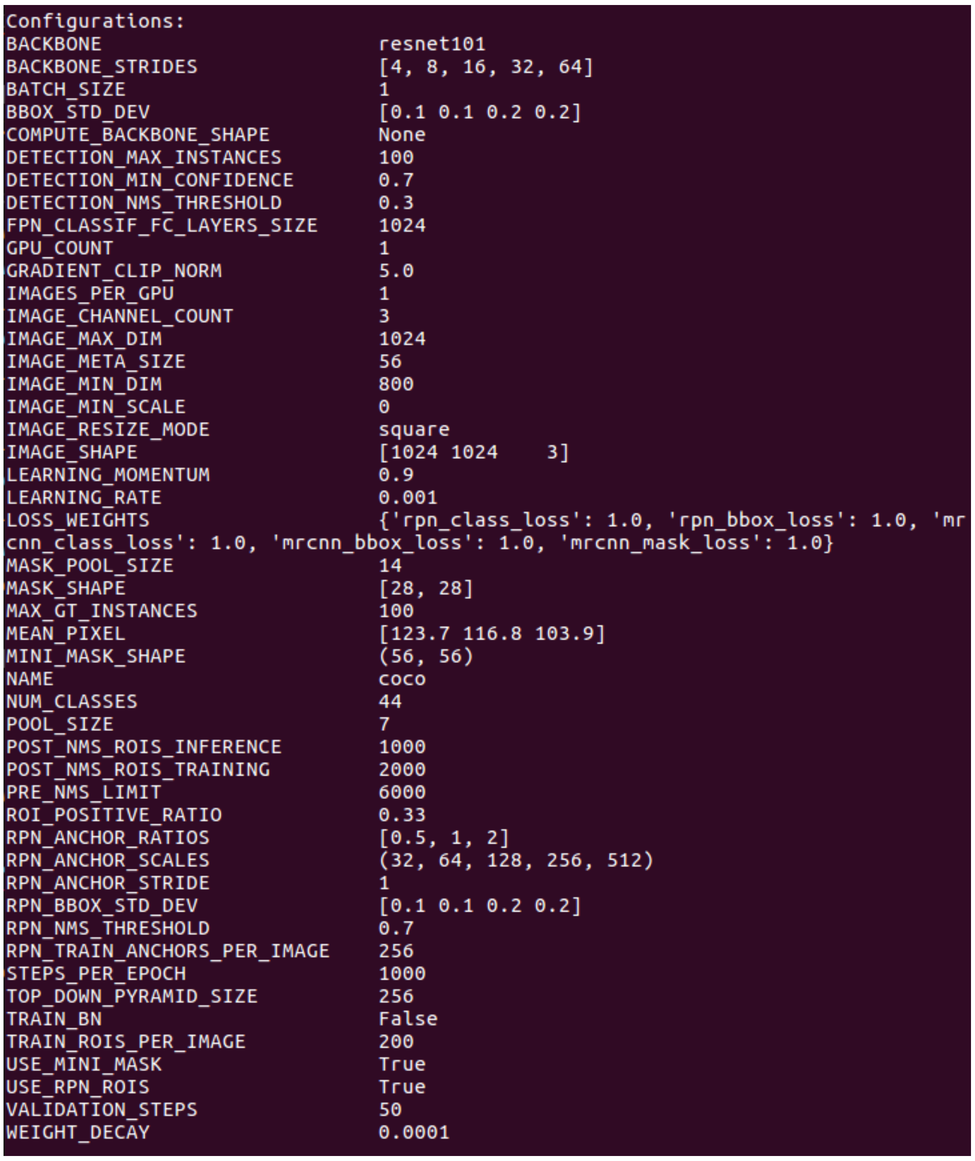
1. 'bottle'
2. 'wine glass'
3. 'cup'
4. 'fork'
5. 'knife'
6. 'spoon'
7. 'bowl'
8. 'chair'
9. 'couch'
10. 'potted plant'
11. 'bed'
12. 'dining table'
13. 'toilet'
14. 'tv'
15. 'laptop'
16. 'mouse'
17. 'remote'
18. 'keyboard'
19. 'cell phone'
20. 'microwave'
21. 'oven'
22. 'toaster'
23. 'sink'
24. 'refrigerator'
25. 'book'
26. 'clock'
27. 'vase'
28. 'scissors'
29. 'teddy bear'
30. 'hair drier'
31. 'toothbrush'
32. 'cabinet'
33. 'counter'
34. 'cupboard'
35. 'desk-stuff'
36. 'door-stuff'
37. 'furniture-other'
38. 'mirror-stuff'
39. 'shelf'
40. 'stairs'
41. 'table'
42. 'window-blind'
43. 'window-other'

Figure FA1.1: Class objects used for training

Input

Train size and validation size are 11827 and 5000.

In the figure FA1.2, the configuration details for training are given.



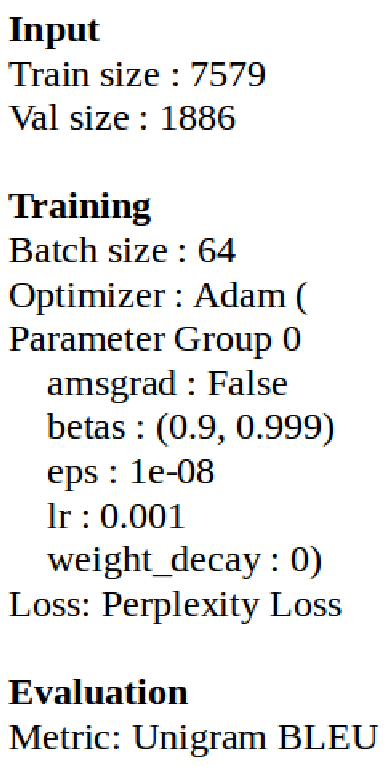
```
Configurations:
BACKBONE                       resnet101
BACKBONE_STRIDES               [4, 8, 16, 32, 64]
BATCH_SIZE                     1
BBOX_STD_DEV                   [0.1 0.1 0.2 0.2]
COMPUTE_BACKBONE_SHAPE         None
DETECTION_MAX_INSTANCES        100
DETECTION_MIN_CONFIDENCE       0.7
DETECTION_NMS_THRESHOLD        0.3
FPN_CLASSIF_FC_LAYERS_SIZE     1024
GPU_COUNT                      1
GRADIENT_CLIP_NORM             5.0
IMAGES_PER_GPU                 1
IMAGE_CHANNEL_COUNT            3
IMAGE_MAX_DIM                  1024
IMAGE_META_SIZE                56
IMAGE_MIN_DIM                  800
IMAGE_MIN_SCALE                0
IMAGE_RESIZE_MODE              square
IMAGE_SHAPE                    [1024 1024    3]
LEARNING_MOMENTUM              0.9
LEARNING_RATE                  0.001
LOSS_WEIGHTS                   {'rpn_class_loss': 1.0, 'rpn_bbox_loss': 1.0, 'mr
cnn_class_loss': 1.0, 'mrcnn_bbox_loss': 1.0, 'mrcnn_mask_loss': 1.0}
MASK_POOL_SIZE                 14
MASK_SHAPE                     [28, 28]
MAX_GT_INSTANCES               100
MEAN_PIXEL                     [123.7 116.8 103.9]
MINI_MASK_SHAPE                (56, 56)
NAME                           coco
NUM_CLASSES                    44
POOL_SIZE                      7
POST_NMS_ROIS_INFERENCE        1000
POST_NMS_ROIS_TRAINING         2000
PRE_NMS_LIMIT                  6000
ROI_POSITIVE_RATIO             0.33
RPN_ANCHOR_RATIOS              [0.5, 1, 2]
RPN_ANCHOR_SCALES              (32, 64, 128, 256, 512)
RPN_ANCHOR_STRIDE              1
RPN_BBOX_STD_DEV               [0.1 0.1 0.2 0.2]
RPN_NMS_THRESHOLD              0.7
RPN_TRAIN_ANCHORS_PER_IMAGE    256
STEPS_PER_EPOCH                1000
TOP_DOWN_PYRAMID_SIZE          256
TRAIN_BN                       False
TRAIN_ROIS_PER_IMAGE           200
USE_MINI_MASK                  True
USE_RPN_ROIS                   True
VALIDATION_STEPS               50
WEIGHT_DECAY                   0.0001
```

Figure FA1.2: Training configuration details for MaskRCNN model

APPENDIX B

APPENDIX: SETTINGS IN MACHINE TRANSLATION MODEL

In the following figure FA2.1, the implementation details are mentioned.



Input
Train size : 7579
Val size : 1886

Training
Batch size : 64
Optimizer : Adam (
Parameter Group 0
    amsgrad : False
    betas : (0.9, 0.999)
    eps : 1e-08
    lr : 0.001
    weight_decay : 0)
Loss: Perplexity Loss

Evaluation
Metric: Unigram BLEU

Figure FA2.1: Implementation details for NMT