Verbalization of Service Robot Experience as Explanations in Language Including Vision-Based Learned Elements

Sai Prabhakar Pandi Selvaraj

CMU-RI-TR-17-53

8th August 2017

Robotics Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Manuela Veloso, Co-chair
Stephanie Rosenthal, Co-chair
Jean Hyaejin Oh
Vittorio Perera

Submitted in partial fulfillment of the requirements for the degree of Master of Science in Robotics.

Keywords: Explainable AI, Human Robot Interaction, Mobile Robots, Vision, Deep Learning, Deep Visualization, Metrics

Abstract

In this thesis, we focus on making robots more trustable by making them describe and explain their actions. First, to tackle the problem of making robots describe their experience, we introduce the concept of *verbalization*, a parallel to visualization. Our verbalization algorithm can analyze log files as well as the robot's live execution data to produce narratives describing its actions while taking user's preference about into consideration using the verbalization space. We introduce the verbalization space to cover the variability in utterances that the robot may use to narrate its experience as we realized that different people might be interested in a different type of description from the robot. We demonstrate verbalization at multiple levels of verbalization space to describe CoBot's path while performing multi-floor navigation tasks. Our initial introduction of verbalization requires manual grounding of the log data to natural language phrases which makes the algorithm unscalable. To tackle this problem, we propose using classifiers and similar techniques to act on robot's data to automatically annotate or ground the data. We then discuss and analyze the classifier we use to ground the log data to natural language automatically. We create DNN based classifier to find the floor CoBot has entered via elevator using input from the camera mounted on CoBot. To analyze the classification, we use different techniques to find important regions in the images for the classification. We have also developed metrics to analyze the relative importance of different regions in the image for a classification. Finally, using the important regions in an image, we produce an explanation in terms of natural language for its classification. We evaluate each algorithm and technique we have developed in this work and compare them with similar state-of-the-art techniques. Although our work focuses on CoBot, we contribute techniques to generalize the techniques we develop here beyond it.

Acknowledgments

First, I would like to thank my advisors Stephanie Rosenthal and Manuela Veloso for providing me advice and teaching me. I'm deeply thankful for your patience and support along the way, and I enjoyed working with you. You gave me the resources and the freedom to do independent research. I would like to thank my thesis committee for all of their help, guidance, and feedback through this process.

I appreciate the valuable discussions with many intelligent students here, especially Sandeep Konam, Devin Schwab, Rui Silva, Avinash Siravaru, Guan-Horng Liu, Ashwin Khadke, and Lekha Walajapet Mohan, as they have knowingly or unknowingly inspired my research. I also thank all my friends both close and afar who had motivate me when I needed it.

Finally, I would like to thank my parents for providing guidance, care and the at times annoying motivations without which none of this would have been possible.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Service robots can autonomously generate and execute plans to successfully perform tasks for humans, appropriately handling the uncertainty of their surroundings. Service robotics is a field where the state-of-the-art techniques in robotics and artificial intelligence merge. For example, our service robot CoBot [46] can localize and autonomously navigate in our buildings to complete assigned tasks like escorting people, as well as participate in a dialogue with humans. CoBot intelligently uses symbiotic autonomy, i.e., CoBot uses help from people around it to overcome some of its limitations, like pressing the elevator buttons or making coffee.

With mobile robots performing more autonomous behaviors without human intervention, people in the environment may wonder what exactly the robot was perceiving, predicting, planning, and doing. Since service robots work in an environment where humans interact with them, they need to make people understand their capabilities. Robotics researchers have developed logging approaches to enable the recording of the robot experience. For debugging purposes, such developers must dig through the accumulated robot logs to find out about the robot experience in great detail. In addition to researchers, an office worker may want the robot to identify why it was late in completing its task. And a person accompanying the robot may want the robot to summarize its speed and distance traveled.

As exemplified above, making robots describe their actions makes them more transparent, but for the robots to be trustable, they need to be able to explain their actions. For example, an autonomous car when questioned why it is taking the longer route should be able to justify its action by informing the user that the shorter route has heavy traffic. We can achieve explanations for robot's actions by making the robot's decision and inference making modules explainable. Such explanations not only increase the trust in the robots but also helps users better understand its capabilities. Hence, we say that making robots explain themselves is a necessary step for them to be able to play a bigger role in our society.

To tackle the problem of making robots describe their experience, we introduce the concept of *verbalization*. Our verbalization algorithm can analyze log files as well as the robot's live execution data to produce narratives describing the robot's actions. Verbalization also takes user's preference about the description into consideration using verbalization space; we introduce verbalization space as we realized that different people might be interested in a different type of description from the robot. We demonstrate verbalization with multiple levels of verbalization space to describe CoBot's path while performing multi-floor navigation tasks. Our initial intro-

duction of verbalization requires manual grounding of the log data to natural language phrases which makes the algorithm unscalable. We propose using classifiers and similar techniques to act on robot's data to automatically annotate or ground the data. In the remainder of the thesis, we discuss and analyze the classifier we use to ground the log data to natural language automatically. We create DNN based classifier to find the which floor the CoBot has currently entered via elevator using a camera. To analyze the classification, we use different techniques to find important regions in the images for the classification. We have also developed metrics to analyze the relative importance of the different regions in the image for a classification. Finally, using the important regions in the image for the classification we produce an explanation for the same. In this thesis, we briefly describe each of our contributions in turn.

First, we start by describing our work on verbalization for describing robot's actions.

1.1 Verbalization

Verbalization converts the robot's experience data into natural language and is parallel to visualization. During the process of verbalization robot's execution data is converted to natural language. We achieve this conversion to natural language by first grounding the data with the natural language phrases. Then, we generate a description by combining the groundings using a template based natural language generation.

Different people's interaction with autonomous robots might focus on different specific information or, different specific parts of the robot's experience. A one-size-fits-all verbalization will not satisfy all users. To vary the explanations according to the user's preference, we introduce the *verbalization space*. The verbalization space has a set of predefined orthogonal axis or parameters to modify the robot's experience data according to the needs of the user. We demonstrate verbalization for describing CoBot's task execution path. Our algorithm modifies the robot's data and the corresponding groundings according to the user's preference specified through the verbalization parameters and thus allowing each person to receive a personalized narrative based on their priorities and interests.

For describing CoBot's path the verbalization algorithm takes as input the route plan which is a set of nodes, user preference in terms of verbalization space parameters, and an annotated map of the environment which is annotated with landmarks. First, the algorithm decides on what language to use depending on the user's preference, for example, locations in terms of coordinates or landmarks. Then, the algorithm labels each of the nodes in the route plan with information about the point, for example, the distance traveled or the landmark nearby. The algorithm uses language corpus to label the nodes. The language corpus contains phrases to use to describe the path, for example, it contains names and locations of landmarks, and phrases like 'right turn' and 'left turn' to describe the geometry of the route plan. Then, based on the user's preference the algorithm selects the portion of the path or combines nodes in the path into segments. Finally, we generate natural language descriptions by combining the labels in each segment.

We demonstrate verbalization to describe CoBot's route in our building for twelve multi-floor and single floor navigation tasks. We also compare the variation in the explanation for the routes as the user preference varies. Verbalization, as we have described so far, can be difficult to

implement for a new and large environment because of having to annotate the data. In the next part of the work, we make verbalization scalable by using automatic annotation, as we describe below.

1.2 Automatic Annotation for Verbalization

As we described above our verbalization algorithm requires grounding robot's data to natural language phrases. We initially introduced verbalization using manual groundings. But, manually grounding the data is clearly not scalable. For example, the verbalization on CoBot depends on the annotated map of its environment to relates its travel route with landmarks, creating an annotated map for a large environment every time a robot changes its environment is not scalable and practical.

Recent applications of automatic annotation are in map building and particularly semantic maps in robotics. Most works in robotics [15, 32] involve creation of a framework to incorporate a learned classifier to enhance a map with semantic information using robot's perception. Some works also incorporate probabilistic modeling [36] and human-in-the-loop methods [14, 33] to eliminate errors due to imperfect perception and inference. Non-robotics application of automated perception includes geographic map building and video annotations [1, 16].

Automatic annotation takes the robot's execution data as input and tags them with natural language phrases automatically. We can do automatic annotation using techniques like statistical modeling and pattern recognition. For example, to tag the speed of the robot at each instance during its task with a descriptive phrase like fast or slow, we can use simple thresholding to tag the speeds into one the categories fast or slow, and thereby automatically annotating the speed. The threshold can be found modeling the speed of the robot using its previous task loggings.

In our work, we use automatic annotation for generating explanations. We use a deterministic approach to automatically find which floor the robot has entered without a human's help. We demonstrate automatic annotation by using a deep neural network (DNN) to find which floor the CoBot is in after reaching a new floor via an elevator. We achieve floor identification using an RGB camera mounted on the robot and DNN based scene recognition module we developed. We have evaluated our identification module and obtained perfect results in a newly collected testing dataset. The testing dataset was obtained while the CoBot is performing multi-floor navigation tasks. With this specific example, we demonstrate that we can leverage similar techniques to make verbalization scalable. CoBot can use the label of the floor generated from the identification module to generate explanations or descriptions of its actions.

When robots explain their actions, they are trusted more by people around and using them. As a first step toward making robots explainable, we make the floor identification module we used for demonstrating automatic annotation explainable, as we describe below.

1.3 Explaining Robot's Actions

The field of explainable artificial intelligence is quite new, so there is a limited work on generating natural language explanations for classifiers. One common approach to generating model-

agnostic explanations for a classifier is learning an interpretable model based on the predictions of the original model [3, 39, 42]. Some of these work try to explain the classifier in terms of a subset of input features [39] and do not focus on the interpretability of these features or generating natural language explanation like us. The work in Hendricks et al. [18] uses description of the classes to train a network to produce explanations for classification of different species of birds. This process is not practical when there are no descriptions for the classes like in our work.

In our work on verbalization, we have described a robot's actions like narrating what path it took to reach its current place. Explanations are different from descriptions in that in explanations we describe the reason why the robot took that path. For example, for a robot's path, a description can be 'I reached here via the Fifth Avenue' while an explanation can be 'I arrived here via the Fifth Avenue because it was the shortest path.'

We generate an explanation for the module's classification of images by grounding the important regions in the image to natural language phrases. First, to explain a decision by any module, we need to find what part of the input has influenced or most influenced the decision. We identify important features for the DNN's classification using deep visualization techniques or importance functions.

Next, similar to the step in verbalization, we need to ground the important regions with natural language tags. We generate groundings for the important regions in the image by using a separate image based object detection network, once again leveraging the concept of automatic annotation. We demonstrate our explanation generating technique by generating an explanation for all the classes the floor identification module is trained on.

In the last part of the work, we contribute metrics for comparing different deep visualization techniques.

1.4 Evaluating Importance Functions

Finding important features for a classification task is a well-researched concept, but its extension to DNNs has only recently received attention. Deep visualization techniques [20] specialize in finding relative importance of each pixel in the image for a convolutional neural network's classification. Although several techniques have recently been [20, 37, 56] have been developed for deep visualization, the evaluation of these methods is generally qualitative [52, 56] or use a human study [37]. The qualitative and human assessments are not only subjective but also do not provide a way to measure how relevant an important region really is for the classifier.

The output of the importance functions or a deep visualization technique, acting on a classifier, taking an image as input is generally a *heat map* containing relative importance of each pixel in the image for the particular classification. We developed two metrics Simple Confidence Gain (SCG), and Concise Confidence Gain (CCG) for comparing the visualization techniques. Strictly speaking, our metrics compare the relative importance of different regions in the image for the classification. We use the metric to compare deep visualization techniques by first creating the regions of high importance by segmenting the heat map outputted by each of the visualization techniques. Then, we compare between the regions of high importance outputted from different visualization techniques using our metrics.

Both of our metrics work on the principle of comparing the increase in accuracy brought by the important regions to a baseline image. Baseline image is a transformed version of the original image in which the classifier is not confident in its prediction. By adding the important pixel regions to this baseline image, in our first metric SCG, we measure the effective increase in confidence of only the important pixels and normalize the increase by the confidence of the original image. CCG - aims to ensure that the new image formed by adding the important pixels to the baseline image is classified correctly. Additionally, CCG takes into account the size of the region to reward the smallest region of most informative pixels.

We discuss more on the comparison between the metrics and present extensive evaluation and show consistent results in comparing three deep visualization techniques on two different datasets.

1.5 Contributions

- We introduce *verbalization* as the process of converting or narrating robot experiences via natural language. To vary the narrations according to the user's preference, we introduce the *verbalization space*, which has a set of predefined orthogonal axis or parameters to modify the robot's experience data according to the needs of the user. We demonstrate verbalization for describing CoBot's [46], our mobile service robot, task execution path. We also demonstrate the ability of verbalization to personalize the explanation according to user's preference by comparing the explanations generated as we vary the parameters in verbalization space.
- Developed a deep neural network based scene recognition module using an RGB camera mounted on CoBot to identify which floor CoBot has entered after exiting the elevator. We have evaluated our identification module and obtained perfect results in testing.
- We propose using automatic annotation to solve unscalability of our verbalization, as it requires manual annotation of robot's data. We demonstrate the use of automatic annotation for verbalization using the floor identification module to find and label which floor the CoBot has entered.
- We have developed two metrics Simple Confidence Gain (SCG), and Concise Confidence Gain (CCG) for comparing the different image region's importance for a classification. SCG measures the increase in network's confidence brought by the various parts of the image for the classification, while CCG looks at the density of information in the parts according to the classifier. Using the metrics, we show consistent results in comparing three deep visualization techniques on two different datasets.
- Proposed a method to generate an explanation for an image based scene classification network. The technique we developed uses both the important regions for the classification found using a deep visualization technique and the objects in the important regions. We find the labels of the object by using another classifier again using the concept of automatic annotation. We demonstrate the method by generating explanations for the floor-identification classifier.

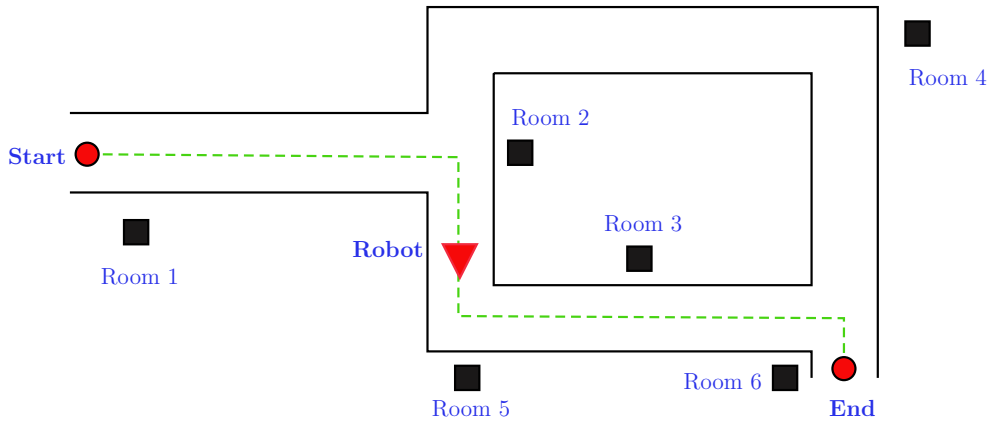


Figure 1.1: Example of a mobile service robot's navigation task

1.6 Illustrative Example

To better understand our contributions, let us consider the example shown in Figure 1.1. In the example, the robot– red triangle, is completing a navigation task of reaching end point from the start point. For describe the path, the verbalization algorithm takes as input the robot's map annotated with landmarks as shown in the Figure 1.1, robot's navigation route plan shown as green dotted line and robot's execution data.

The verbalization algorithm for the example task without taking user's preference into consideration might produce the following description after completing the task.

> 'I started near Room 1, took a right turn near Room 2, took a left turn near Room 5, went by Room 3, and reached end point near Room 6.'

The verbalization, when combined with verbalization space to describe the robot's route for a user who wants a summary, will change the narration to

> 'I started near Room 1 and reached end point near Room 6.'

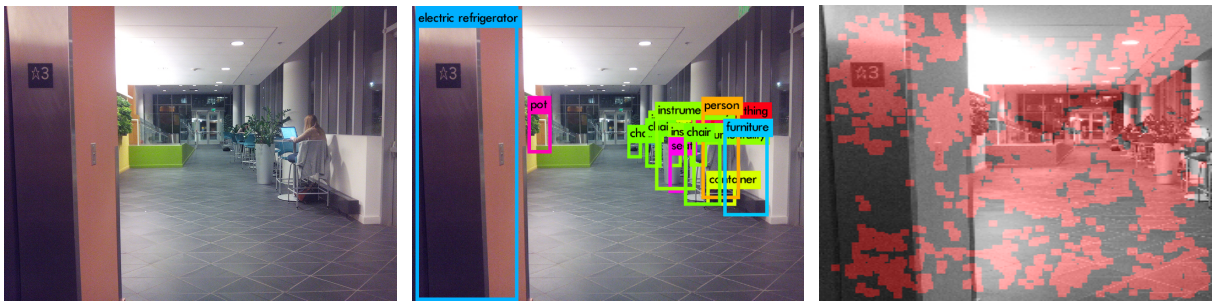
We will go into details of how the verbalization algorithm works in their respective chapter.

Let us assume that the robot incorporates an image based scene recognition module like our floor-identification module. We can use the recognition module to annotate the map with landmarks labels like Room 1, as shown in the Figure 1.1 automatically.

As an example for our technique to explaining the scene recognition classifier, assume that at the robot's position shown in the Figure 1.1, the robot sees the image as shown in Figure 1.2(a) in its camera. Using the recognition module let us also assume that the robot can classify the image correctly as the scene near Room 5. Our explanation generation module takes into consideration the importance heat map Figure 1.2(c), and the objects in the image Figure 1.2(b), as groundings to produce the following output as explanation

> I am near Room 5 because I see furnishing at right bottom, I see chair at right bottom, I see chair at center, I see pot at left top,

for why it thinks it is near Room 5.



(a) (b) (c)

Figure 1.2: Example for explanation generation module for scene recognition classifier (a) original image captured by the robot near Room 5, (b) object predictions or language groundings on the image, (c) importance heat mask for the image

Chapter 2

Verbalization

In this chapter, we introduce *verbalization* as the process of converting or narrating robot experiences via natural language. A robot that verbalizes its experiences could help each of the above example users resolve questions they have about autonomous robot behavior.

We also contribute the concept of the *verbalization space* to represent ways in which verbalizations may vary for different reasons, including user preferences and needs. We define our verbalization space across three orthogonal parameters that prior research has indicated per-user needs or preferences over [9, 13, 44]. The first parameter, *abstraction*, varies the vocabulary and concepts used in the narrative from concrete robot concepts, such as distances, speed, and time to abstract concepts, such as hallways, rooms, landmarks. Second, *specificity* varies the total number of concepts or words used in the summaries, allowing the robot to generate single-sentence general, or multi-sentence detailed, narratives. Finally, *locality* varies the particular parts of the experience that the narration focuses on, from the global path to a local region or landmark of interest. Our verbalization space is general and can be extended to many other parameters.

We first formalize the concept of verbalizing experiences, as well as each of the parameters of our verbalization space with a focus on navigation tasks. We contribute our algorithm for generating narratives using the three verbalization space parameters, and we provide examples of how to combine these parameters. Our algorithm can be adapted to use other natural language generation techniques or verbalization space parameters. Finally, we demonstrate the use of our verbalization space to narrate our mobile robot's experiences through our building, and validate that it generates narratives of different abstraction, specificity, and locality.

2.1 Related Work

Prior work in automatically generating explanations or summaries of planned behavior can be roughly divided into three categories: 1) intelligibility or explanation of machine learning algorithms, 2) summarizing perceived behavior, and 3) generating directions for humans to follow.

As machine learning gains popularity in many different applications, much human-computer interaction research has focused on ways machine learning applications can *intelligibly* explain their reasoning algorithms to users (e.g., for context-aware systems [13]). HCI intelligibility studies have focused on ways that users can query applications for information or explanations

(e.g., [27]) as well as how those explanations can affect users' mental models of how the applications work (e.g., [24, 25]). The studies find that explanations increase trust of machine learning applications [12] as well as improve users' mental models. Due to the success of intelligibility across many applications, intelligibility toolkits have been implemented for consistency of explanation across different machine learning algorithms [26]. While prior work shows that varying the focus of explanations is important and useful to users, no one implements it.

Another growing area of research is in summarizing or generating narratives of perceived behavior. For example, RoboCup soccer commentators aim to use the input of simulated RoboCup games [48] or live RoboCup games [45] to generate realtime summaries of the actions in the games. Activity recognition algorithms and natural language generation have also been used to produce annotated accounts of wartime exercises [29], video conferencing sessions [49], and sports games [2]. While some work generates a variety of summaries to maintain human interest (e.g., [45]), the work does not vary the length or depth of summaries as we do.

Finally, and perhaps most closely related to our work, GPS applications (e.g., [4]) and robot applications (e.g., [9, 21, 44]) are automatically generating navigation instructions and dialog for people to follow and understand. In the prior work, a path is converted into language and ideally presented in an easy-to-understand yet accurate way for the person to follow it seamlessly every time. While these navigation directions do not vary in the language used, recently [9] found that navigation directions should 1) provide differing levels of specificity at different locations in the route and 2) use abstract landmarks in addition to more concrete details. Similarly, prior work on human direction givers shows that humans do not generate the same directions for every person [30].

We note that none of the prior work focuses on summarizing both perception and plans of a robot or other autonomous vehicle. And while the prior work extensively documents the need for parameterized summaries, none of the prior work, to our knowledge, measures those parameters and contributes an algorithm for actually varying them. In this work, we first contribute verbalization as a method of summarizing what robots actually experience. Based on the findings from prior work as well as the needs of our robots' users, we then propose and formalize our verbalization space that represents the variability in narratives, and we provide an algorithm for generating variable verbalizations of route plans.

2.2 Route Verbalization

We define *verbalization* as the process by which an autonomous robot converts its own experience into language. In this work, we consider mobile navigation experience in the physical world, and verbalize what the robot experienced while traversing its route. We define *route verbalization* as the process by which an autonomous robot converts its own route experience into language. A robot can generate route verbalizations mentioning the planned route that will be traversed or the route that has been traversed (*i.e.,* a narrative in the future tense is equivalent to GPS driving directions, while a narrative of the past traversed route describes the actual experience). At this time, we do not distinguish between the future and past tenses, exemplifying the applicability across language generation domains.

We first define simple route verbalizations over common robot map and route representa-

tions. Then, we describe our annotations to the map and route to accommodate the variation in verbalization that humans require.

Specificity, S		Abstraction, A			
		Level 1	Level 2	Level 3	Level 4
	General Picture	Start and finish point of the complete route	Total distance and time taken for the complete route	Total distance and time taken for the complete route	Starting and ending landmark of complete route
	Summary	Start and finish point for subroute on each floor of each building	Total distance and time taken for subroute on each floor of each building	Total distance and angles for subroute on each floor of each building	Starting and ending landmark for subroute on each floor of each building
	Detailed Narrative	Start and finish points of complete route plus time taken for each edge of route	Angle turned at each point plus the total distance and time taken for each edge of route	Turn direction at each point plus total distance for each edge of route	All landmarks encountered on the route

Table 2.1: Narrated information depends on preferred Verbalization Space parameters. Information for Abstraction A and Specificity S are shown assuming Locality L is Global. For a different Locality, a subset of the route is generated, and the information provided is computed in terms of the subset.

2.2.1 Robot Map and Route Plan

We define an indoor mobile robot's map $M = \langle P, E \rangle$ as set of points $p = (x, y, b, z) \in P$ representing unique locations (x, y) in our buildings b for each floor z and edges $e = \langle p_1, p_2, d, t \rangle \in E$ that connect that connect two points taking time t to traverse distance d.

The points on the map are annotated with semantic *landmarks* represented as *room numbers* (e.g., 7412, 3201) and *room type* (office, kitchen, bathroom, elevator, stairs, other). Points could

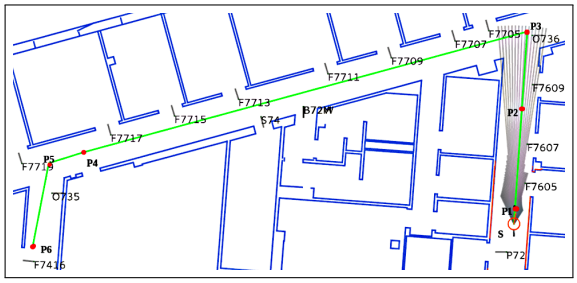


Figure 2.1: Robot route plan (green lines), nodes $\{S$,P_1,...$P_6\}$, Starting node S, and finish node P_6.

be annotated with additional information, including the occupants of the office or the names of laboratory spaces (e.g., as in [40]). We also maintain lists of *corridors* and *bridges* as points that reside within them (e.g., "7400 corridor" contains office 7401, office 7402, office 7404, etc. and the "7th floor bridge" contains other 71, other 72, etc.). Some points may not appear in any corridor or bridge list if they are in open areas, and some points may reside in two hallways if they occur at hall intersections.

Using our map, our route planner produces plans as trajectories through the environment composed of:

- a starting point S,
- a finish point F,
- an ordered list of intermediate waypoints $W \subset P$, and
- a subset of straight line edges in E that connect S to F through W.

Our planner labels waypoints as *turning points* representing the only places the robot turns after traversing straight edges. Figure 2.1 shows a route plan, the starting point S, and finish point $F = P_6$, as the destination of a task requested by a user. The figure shows turning points $W = \{P_1, P_2, P_3, P_4, P_5\}$, connected by straight line edges (as pictured in green).

2.2.2 Simple Route Verbalization

Using the map and route plan described above, a simple route verbalization algorithm could interleave turn angles at each point p and distances traversed for each edge e between waypoints. For the route depicted in Figure 2.1, this simple route verbalization algorithm would produce:

> *I went straight for 8.5 meters and turned left, then straight for 24.9 meters and turned left, then straight for 3.62 meters to reach the destination.*

While this verbalization successfully describes the robot's route, different people in the environment may be expecting more or different information to be provided. For example, we as robotics researchers could be interested in the exact (x, y, b, z) coordinates of the points where the robot turns. Other people in the environment may find landmarks such as room numbers to be useful. We next describe the use of our semantic annotations within our verbalization space.

2.3 Verbalization Space

We represent the variations in possible narratives of the same route as the *verbalization space*. Each region of the verbalization space represents a different way to generate text to describe the route plan. A user may specify their personalized preferences for verbalization within this space, or the preferences may be inferred from some other source. Our verbalization space contains three orthogonal parameters – abstraction, locality, and specificity – that are well-documented as personal preferences in the literature (e.g., [9, 13, 44]). Our verbalization space is general and could be extended to include more parameters as needed.

2.3.1 Verbalization Space Definitions

Table 2.1 details the way we instantiate verbalizations for specified parameters $(a, l, s) \in (A, L, S)$.

Abstraction A: Our abstraction parameter represents the vocabulary or corpus used in the text generation. In the most concrete form (Level 1), we generate text in terms of the robot's world representation, directly using points (x, y, b, z) from the route plan. Our Level 2 derives turn angles and uses expected or actual traversal time and distances from the points and edges in the plan. Level 3 abstracts the angles and distances into right/left turns and straight segments. And finally, in the highest level of abstraction, Level 4 contains the semantic annotations described above.

Locality L: Locality describes the segment(s) of the route the user is interested in. In the most general case, the user is interested in the route through the entire Global Environment including all buildings and floors. However, an office occupant may only be interested in a particular predefined Region of the route composed of multiple points in the maps (e.g., we limit our regions by building b or building floor b, z). Finally, the occupant may specify a single particular point or landmark for the robot to summarize its route around (e.g., a constant distance around the 8th floor kitchen or Office 4002).

Specificity S: Specificity indicates the number of concepts or details to discuss in the text: the General Picture, the Summary, and the Detailed Narrative. The General Picture contains the most general description of the robot's route, namely the start and finish points (or landmarks), the total distance covered, and/or the time taken (see Table 1). Our Summaries contain this same information for the subroute on each floor of each building. The Detailed Narrative contains a description of each edge of the robot's route.

Next we describe how these verbalization space parameters are used to generate verbalization text.

2.3.2 Variable Verbalization Algorithm

The Variable Verbalization (VV) algorithm pseudocode is presented in Algorithm 1. The algorithm directly translates the robot's route plan into plain English given the map and the incorporated annotations described above. It takes as input a *route*, a verbalization space preference *verb_pref* = $(a, l, s) \in (A, L, S)$, and a *map* of the environment with locations labeled as above. It starts by choosing what corpus (Level 1-4) to use when generating utterances depending on abstraction preference a (Line 2). Then, the VV algorithm annotates the given route by labeling each point with landmarks and corridor/bridge names using the map (Line 3).

Once the route is annotated with relevant locations, the algorithm extracts the subset of the route that is designated as relevant by the locality preference l (Line 4). We subset Regions by building and floor and Landmarks by a threshold distance around a given point. Both of these subset types can be directly computed from our point representation - Regions using b, z and Landmarks using a distance function around x, y for the given building/floor. The output of this step is another annotated route that is a copy of the route if l=Global Environment. Otherwise, the output is a subset of the route with a new start and finish point.

Using the *subset_route*, the VV algorithm then computes route segments to narrate with respect to the specificity preference s (Line 5). If the specificity preference is a General Picture,

Algorithm 1 Variable Verbalization Algorithm

Input: *route, verb_pref, map* **Output:** *narrative*

//The verbalization space preferences

1: $(a, l, s) \leftarrow$ *verb_pref*

//Choose which abstraction vocabulary to use

2: *corpus* $\leftarrow$ ChooseAbstractionCorpus(a)

//Annotate the route with relevant map landmarks

3: *annotated_route* $\leftarrow$ AnnotateRoute(*route, map*, a)

//Subset the route based on preferred locality

4: *subset_route* $\leftarrow$ SubsetRoute(*annotated_route,* l)

//Divide the route into segments, one per utterance

5: *route_segs* $\leftarrow$ SegmentRoute(*subset_route, s*)

//Generate utterances for each segment

6: *utterances* $\leftarrow$ NarrateRoute(*route_segs, corpus, a, l, s*)

//Combine utterances into full narrative

7: *narrative* $\leftarrow$ FormSentences(*utterances*)

our algorithm computes the required abstraction information for a single route segment from S to F. For Summaries, it computes one route segment for each floor of each building and then computes the relevant abstraction information for those segments. In Detailed Narratives, all edges are included in the narrative.

The Algorithm then translates the route segments from Line 5 into plain English using the corpus vocabulary from the annotated map and template sentences (Line 6, examples described next). Finally, after the sentences have been generated for each route segment, the VV algorithm stitches them together (Line 7). The final narrative is returned as the output of the function.

In the next section, we describe our implementation of our algorithm on our mobile robot and its routes.

2.4 Mobile Robot Route Verbalizations

Our mobile service robot plans and executes tasks autonomously in our buildings [5, 6], such as accompanying visitors to their meetings and carrying objects to offices [46]. It regularly interacts with humans in the environment through dialog and symbiotic interactions to ask for help [34, 35, 40]. We found many different people in our environment are interested in what our robot is doing and experiencing as it acts. We as researchers tend to be interested in high specificity, detailed narratives about the global environment. Other people may be interested in narratives about their own office locations at a general picture level. The Variable Verbalization algorithm is implemented on our robot and allows each person to receive a personalized narrative based on their priorities and interests.

We first describe our annotated map and corpus for verbalizations that are input into our Variable Verbalization algorithm. Then, we describe two narratives based on different verbalization space preferences for the same route. Finally, we test our algorithm on different routes

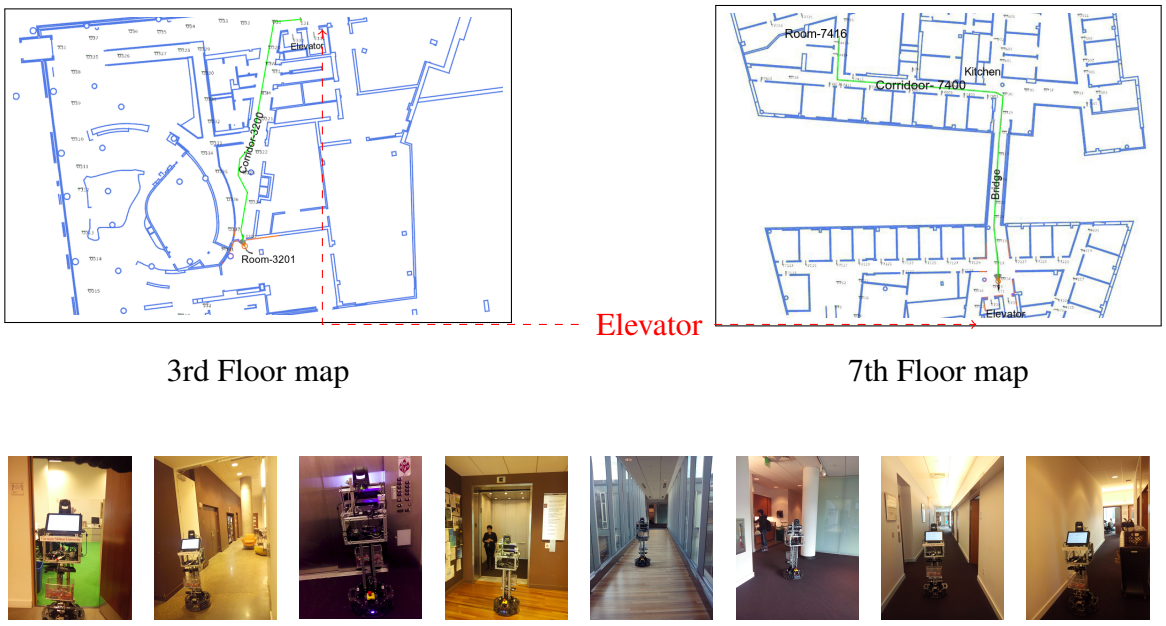


Figure 2.2: Top: Example of our mobile robot's multi-floor plan in our building (blue walls, green route, red connects elevator between floors). Bottom: Images of our robot navigating the route. The robot (1) starts at Office 3201, (2) travels down the 3200 corridor, and turns right to (3) reach the elevator. Once it (4) reaches the 7th floor, it (5) travels straight across the bridge, (6) turns left at the kitchen, (7) travels down the 7400 corridor, and then (8) makes its first right to Office 7416.

through our building to demonstrate how the number of words and numbers changes with each instantiation of our verbalization space.

2.4.1 Robot Map and Language Corpus

Our robot's environment includes three buildings connected by bridges. Each floor of each building has its own coordinate system. The individual floor maps are linked to each other via the elevators and bridges, so that the robot can use multiple floors while planning and executing. The set of all floors and all buildings is defined as our map M. Our map contains points p representing any arbitrary location on the map. Points can be labeled as landmarks representing specific room numbers and room types including office, lab, kitchen, bathroom, elevator, stairs, printers, and other. We also maintain lists of corridors and bridges as outlined above. Given any two points, start S and finish F, our route planner computes a set of edges and waypoints to travel from S to F.

Our corpus of landmarks on the map (exerpt below) is used for Level 4 of our Abstraction parameter. Our other corpora for our other levels of abstraction are much smaller and include

(x, y) "points", "angle" degrees, distance in "meters", "left turns", "right turns", and "u-turns".

$$\begin{cases} \ldots & \ldots \\ \text{Office-3201}(x, y, Gates, 3^{rd}\ floor) \\ \text{Bathroom-3}(x, y, Gates, 3^{rd}\ floor) \\ \text{Stairs-34} \quad (x, y, Gates, 3^{rd}\ floor) \\ \text{Kitchen-71}\ (x, y, Gates, 7^{th}\ floor) \\ \text{Office-7401}(x, y, Gates, 7^{th}\ floor) \\ \text{Office-7412}(x, y, Gates, 7^{th}\ floor) \\ \ldots & \ldots \end{cases}$$

2.4.2 Route Experience Variable Verbalization

Using our map, our mobile robot plans routes between points in our building. Figure 2.2 Top shows one example route (in green) from the 3rd Floor Office 3201 to the 7th Floor Office 7416 in our Gates building. We have labeled in black our annotations over the map including the corridors, the elevators, a bridge, and a kitchen. Figure 2.2 Bottom shows a visual depiction of the robot traveling along this route. We demonstrate two variations of verbalizations for the route.

Example 1: Long, Detailed Verbalization

With our map and corpus, we consider the preference:

(Level 4, Global Environment, Detailed Narrative)

that represents a researcher in our lab who wants a detailed description of what happens on each edge of the robot's route. We will review our algorithm's analysis of the route plan to generate a verbalization fitting this preference.

Choose Abstraction Corpus: Because the abstraction level preference is Level 4, the VV algorithm chooses the large corpus of room numbers, room types, and corridors and bridges for its language model.

Annotate Route: Next, the input route is annotated with these landmarks from the corpus. In this case, the VV algorithm labels starting point Office-3201; the points leading to the elevator are Corridor-3200; the elevator on the 3rd floor is labeled Elevator-31 and similarly the 7th floor is labeled Elevator-71; points on the bridge are Bridge-7; the Kitchen-71 is labeled; the hallway points are labeled Corridor-7400; and finally the finish point is Office-7416.

Subset Route: The researcher is interested in the Global Environment Locality, and thus the route is not subsetted.

Segment Route: The researcher would like s =Detailed Narrative. Our algorithm merges all same-labels, resulting in seven route segments. We write segments in terms of their meaning here because there are too many points to enumerate; the robot maintains the list of points on the route.

$\{$s1: Office-3201, s2: Corridor-3200, s3: Elevator,
s4: Bridge-7, s5: Kitchen-71,
s6: Corridor-7400, s7: Office-7416$\}$

Narrate Route: Our algorithm's ability to narrate a route depends on filling in templates matching different route segments. We manually created the following templates for Level 4 abstractions. We note next to the D whether the type of landmark is specific (e.g., the template must be filled in by a corridor, bridge, etc.), and we note with a slash that the choice of verb is random to prevent repetition by replacing the verbs with a synonym (e.g., [45]). We have similar templates for other abstraction levels that include distances and time to complete the route segments.

- "[I]$_N$ [visited/passed]$_V$ the [___]$_{D:room}$"
- "[I]$_N$ [took]$_V$ the elevator and went to the [___]$_{D:floor}$"
- "[I]$_N$ [went through/took]$_V$ the [___]$_{D:corridor/bridge}$"
- "[I]$_N$ [started from]$_V$ the [___]$_{D:start}$"
- "[I]$_N$ [reached]$_V$ [___]$_{D:finish}$"

Using the templates, the VV Algorithm generates utterances for each of the segments.

s1: "I started from Office 3201",
s2: "I went through the 3200 corridor",
s3: "I took the elevator to the seventh floor",
s4: "I took the 7th floor bridge",
s5: "I passed the kitchen",
s6: "I went through the 7400 corridor",
s7: "I reached Office 7416",

Form Sentences: Finally, the algorithm combines the sentences with "then"s (more complex concatenation could be used):

> *I started from office 3201, then I went through the 3200 corridor, then I took the elevator and went to the seventh floor, then I took the 7th floor bridge, then I passed the kitchen, then I went through the 7400 corridor, then I reached office 7416.*

Example 2: Short Overview Verbalization

To contrast the long detailed landmark-based narrative, a short verbalization can be achieved with preference

(Level 2, Gates 7th Floor Region, General Picture)

Here, a person accompanying the robot wants to know how far they traveled only on the 7th floor. The VV algorithm first annotates our entire route with abstraction Level 2, adding distances to the edges in the route between each pair of points. Since the required locality is Region, the algorithm subsets the route containing only the required Gates 7th floor points. As the specificity is General Picture, a single route segment is generated as the combination of all edges from the new 7th floor start node S to the finish node F. The route is annotated with the total distance and time taken for the route. Next the algorithm narrates the route using the template "$[I]_N$ [traveled] [x] meters in [t] seconds on the [__]$_{D:floor}$". Finally these utterances could be combined (not necessary here) to form the final narrative:

I traveled 56.18 meters and took 75 seconds on the 7th floor.

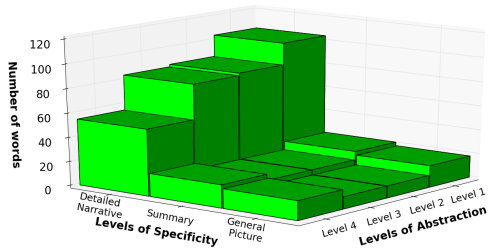


Figure 2.3: Average number of words generated.

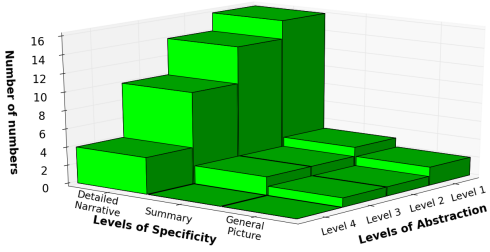


Figure 2.4: Average number of numbers generated.

2.4.3 Validation

Given the well-documented need for verbalizations, we focus our experiment on whether we succeed at varying our verbalizations based on those needs. We randomly generated 12 multi-floor routes in our Gates building and 12 single-floor routes, ran the VV algorithm over the route plans, and analyzed the content of the 36×24 verbalizations that were generated.

Figure 2.3 shows the average number of words for two of our parameters: abstraction and specificity. There are many more words in Detailed Narratives (55-104 words) compared to Summaries (14-21) or General Pictures (10-18). We note that the number of words is nearly the same for Summaries and General Pictures. Because our VV in CoBot creates one phrase per floor of the building for Summaries, it generates the same narrative as the General Picture for single-floor navigation routes. Given that half of our routes are single-floor, the average number of words for Summaries is similar to that of General Picture rather than Detailed Narratives.

Additionally, there are more words generated for Summary/General Picture Level 4 Abstraction than Level 3 or 2. This is due to the landmark descriptions that are more verbose than the time and distances reported. In contrast, for Abstraction Level 4, there are no numbers in most of our narratives as the landmarks are entirely made up of words (Figure 2.4). The exception is Level 4 Abstractions with Detailed Narratives, which do include office numbers.

The addition of the locality parameter reduces the overall number of words and numbers but shows the same patterns. As the narratives become more focused around a region and then a landmark, there are fewer route segments to describe. We conclude that overall we do successfully vary narratives within our verbalization space.

2.5 Summary and Conclusion

It is hard, if not impossible, for humans to understand the experience of an autonomous mobile robot. In this chapter, we have contributed a novel approach to capture verbalization by a robot as a way for the robot to narrate its experience in natural language. Our mobile robot translates its route experiences in into verbalization utterances. We contribute the verbalization space as a formalization of multiple levels of detail in which narrations can be generated. We introduce different axes of the space to represent different dimensions of verbalization, namely abstraction, locality, and specificity, though the space can be extended.

The verbalization algorithm takes as input the route plan which is a set of nodes, user preference regarding verbalization space parameters, and an annotated map of the environment. First, the algorithm decides on what language corpus to use depending on the user's preference. Then, the algorithm labels each of the nodes in the route plan with information about the point and converts them to natural language phrases using the language corpus. Finally, based on the user's preference the algorithm selects a portion of the path or combines certain nodes in the path into segments, and generate a natural language description for the user by combining the labels in each of the segment.

The approach we present aims at being applicable beyond mobile robots to other planning algorithms, allowing language to be adjusted to the desired levels of detail. For autonomous vehicles, we can imagine using a new map and semantic landmark labels with our same verbalization space and the same verbalization algorithm to produce narrations of driven routes. Autonomous vehicles would reason over points in GPS space, and use landmarks such as buildings, roads, and street signs to create a variety of narrations. Other intelligible machine learning applications could also produce new formalisms for the verbalization space to produce variable narrations.

We demonstrate the use of verbalizations on our mobile service robot. We present two examples of narrations corresponding to different points in the verbalization space for one multi-floor route through our building environment. Then, we validate on 24 routes that a variety of narrations that can be generated from any single plan. Future work will focus on studying techniques for the personalization of verbalization preferences among our building occupants.

Chapter 3

Automatic Annotation for Verbalization and Classifier-Based Evaluation of Image Feature Importance

In the previous chapter, we generated descriptions for robot's path by grounding the locations in the map manually. It is clearly inefficient and not realistic to ground the map or data from the robot completely by hand. For example, the verbalization on CoBot depends on the annotated map of its environment to relates its travel route with landmarks, creating an annotated map for a large environment every time a robot changes its environment is not scalable and practical.

In this chapter, we discuss automatic annotation for verbalization to avoid manual labeling. We demonstrate automatic annotation by automatically finding and labeling which floor the CoBot has entered after exiting the elevator using the data that CoBot potentially captures along its path traversal. We predict the identity of the floor using the camera mounted on CoBot, with a deep learning based scene recognition classifier and use that floor label in the verbalization algorithm to generate descriptions. Currently, the CoBot relies on symbiotic autonomy [46], i.e., it uses help from people near by to find if it has arrived on the right floor. We would also need to ask for floor information during verbalization. However with this specific example, we demonstrate that we can leverage automatic techniques to make verbalization scalable and reduce the need for human help.

Another interesting question to ask is what part of an explanation is most important, as we can use the important parts to avoid excessively long verbalizations. Because we focus on explaining image classification, we also find features in the image that made the scene recognition classify the image into a particular class. In the next chapter, we will describe our approach to explaining the classification. In this chapter, we first explain the first step - to identify the important regions or features in the image responsible for the classification. Since we use deep learning-based scene recognition for automatic annotation, we will focus on finding important features of images within deep learning algorithms. We have developed metrics to compare between different important regions found using different methods.

This chapter is organized into two parts. In the first, we describe the out floor identification module implemented in the CoBot and thus demonstrating the automatic annotation. Then we discuss the importance function and the metric we developed to evaluate the function. Related

work, experiments and the results for each of the parts are discussed in their corresponding section.

3.1 Automatic Annotation for Verbalization

Automatic annotation is a well-researched concept which uses different modeling techniques to classify or label data. In this work, we look at the application of automatic annotation to tackle the problem of unscalability of verbalization due to manually-generated groundings between robot data and English words. With execution data already collected from the robots current perception systems, we can utilize statistical modeling or pattern recognition to annotate that data with natural language labels automatically. And, we can generate groundings for different types of data from the robot like camera, laser scanner or odometry. For example, to tag the speed of the robot at each instance during its task with a descriptive phrase like fast or slow, we can use a simple threshold to label the speeds into one the categories fast or slow, and thereby annotating the speed. We can find the threshold by modeling the speed of the robot using its previous task loggings.

In this work, to demonstrate the use of automatic annotation in verbalization, we have used image based scene recognition using the camera mounted on the CoBot. We use a deterministic approach to automatically find which floor the robot has entered without a human's help, i.e., we do not use probabilistic modeling of labelings. The annotation module finds which floor CoBot is currently in after it exits the elevator using a deep learning based scene recognition network.

There has been a tremendous advancement in the performance of image recognition using deep convolution networks (CNN) [23]. CNNs also lead to better performance in both indoor and outdoor scene recognition tasks. In this work, we frame the problem of finding which floor the robot is on as a scene recognition problem. After exiting the elevator, CoBot takes a picture of the scene outside the elevator to find which floor it is currently on and classifies the image into one of the floors its floor detector CNN was trained on.

The scenes outside the elevator on all the floors are similar regarding the objects present in them. And since the scene is static except for people in them, the classifier during training is prone to overfitting. To avoid overfitting we chose to use a CNN based on Siamese architecture [11], because it has been shown to perform well in one-shot learning problems [22] and so can make the network explore the input space more.

We have evaluated our floor-identification module and obtained perfect results in a newly collected testing dataset. The testing dataset was obtained while the CoBot is performing multi-floor navigation tasks. With this specific example, we demonstrate that we can leverage similar techniques to make verbalization scalable. CoBot can use the label of the floor generated from the identification module to generate explanations or descriptions of its actions. We have discussed the related work, network architecture and results for automatically annotating the robot's floor in this section.

3.1.1 Related Work

Some of the recent applications of automatic annotation in artificial intelligence are in building robot's semantic map [15, 32, 36] and to annotate the perception data [1, 16]. Using a 2D laser range sensor and a mobile robot [15] builds semantically annotated maps. The work compares a hand-crafted and a learned classifier to generate indoor maps and enhance them with semantic information. Thus the work generates semantically annotated maps. Pronobis and Jensfelt [36] proposes a system to build a probabilistic representation to estimate room labels along with the room's properties like room size and the objects in them. Pronobis and Jensfelt [36] mainly focuses on automated perception and inference.

Hansen et al. [16] deals with automatic annotation of humans in the surveillance videos. The work annotates each human in the video with their color of the clothing, the height, and focus of attention. The height and the focus of attention are estimated using 3D geometry and changes in the intensity of the person. Alexanderson et al. [1] has developed a method to segment and label gestural units automatically from a stream of 3D motion capture data obtained from a Microsoft Kinect. The models use a 2-level Hierarchical Hidden Markov Model to model gestural flows.

Relying on fully automated annotation is error prone due to the various reasons like limitations of sensors and inaccurate inferencing module, these limitations can be overcome by using the human in the loop methods [14, 33]. Diosi et al. [14] makes the user and the system to build contextual topological maps interactively. In Nieto-Granda et al. [33] a user guide supports the robot in the process of associating spatial regions to semantic labels using a probabilistic model, by instructing the robot in selecting the labels. In our work, we use a deterministic approach to automatically find which floor the robot has entered without a human's help.

3.1.2 Annotating CoBots Paths with Floor Labels

When a robot navigates across many floors of a building or multiple buildings, one major challenge that it has when exiting the elevator is localizing itself to determine which floor it is currently on [46, 47]. To train a scene recognition classifier for finding which floor the CoBot has reached via elevator, we have collected the Building-Floor dataset.

Building-Floor

We collected a Building-Floor dataset in one of our Gates-Hillman buildings. Each image contains the scene just outside the elevator from six different floors of the building, ref Figure 3.1. The goal of the classifier trained on this dataset will be to find which floor the image belongs is taken from. Since the robot cannot change floors other than by taking the elevator, the elevators are the only locations where we need to classify which floor the robot is on.

For each of the floors in the building, five images were taken at a particular location that our robot stops at after exiting the elevator. To simplify the analysis all the images were taken at the same time of the day, and the effects of people moving around in the building are not considered. The training data consists of three images, and the remaining two images form the verification dataset. As the scene does not change a lot in theory one image is enough to be able to train and obtain good performance, but in practice this is not true, so we use three images

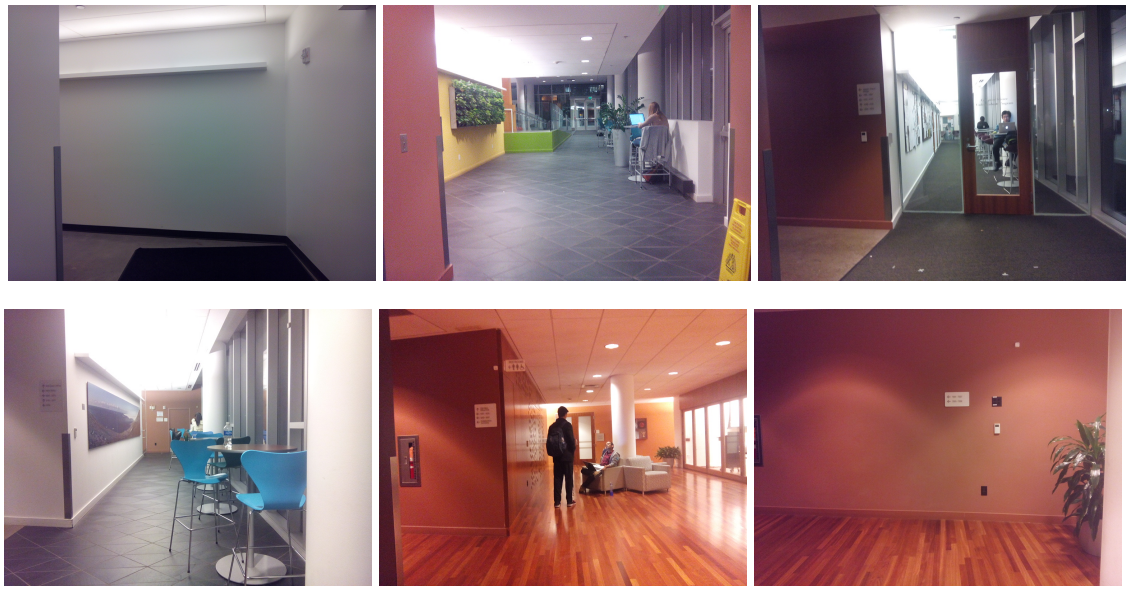


Figure 3.1: Sample of images from the Floor detection dataset. Each image belongs to a different floor.

to avoid overfitting and capture some variation in the scene. Testing data collected separately consists of five images from each of the floor taken at similar day time and setting as the training dataset. The newly collected testing dataset was obtained while the CoBot is performing multi-floor navigation tasks.

Network Architecture

Our training network of nine layers followed the AlexNet [23] in a modified Siamese architecture proposed in Sun et al. [43], Zheng et al. [53], which combined the identification– Softmax, and the verification loss– Contrastive, for better performance, refer Figure 3.2. Our main reason for combining identification and verification loss with a pre-trained network is to reduce overfitting which could happen when the complexity of network is higher than the data. During training, the first seven layers of our network were initialized from Places205-AlexNet which was trained in the Places205-Standard dataset and provided by the authors [54]. The remaining two layers were trained from scratch. During training contrastive loss was utilized in the eighth layer which is a dense layer of 1000 units, while the softmax loss was employed in the ninth layer.

3.1.3 Results and discussion

Our network can classify all the images in the testing dataset correctly. The approach we have taken to classify each of the floors using a DNN based network is currently a popular method and can be easily scaled to accommodate more classes. Increasing the number of classes the network can classify can be done by just increasing the number of units in the final layer of the network, but the network will have to be trained from scratch following the procedure we have outlined in this document.

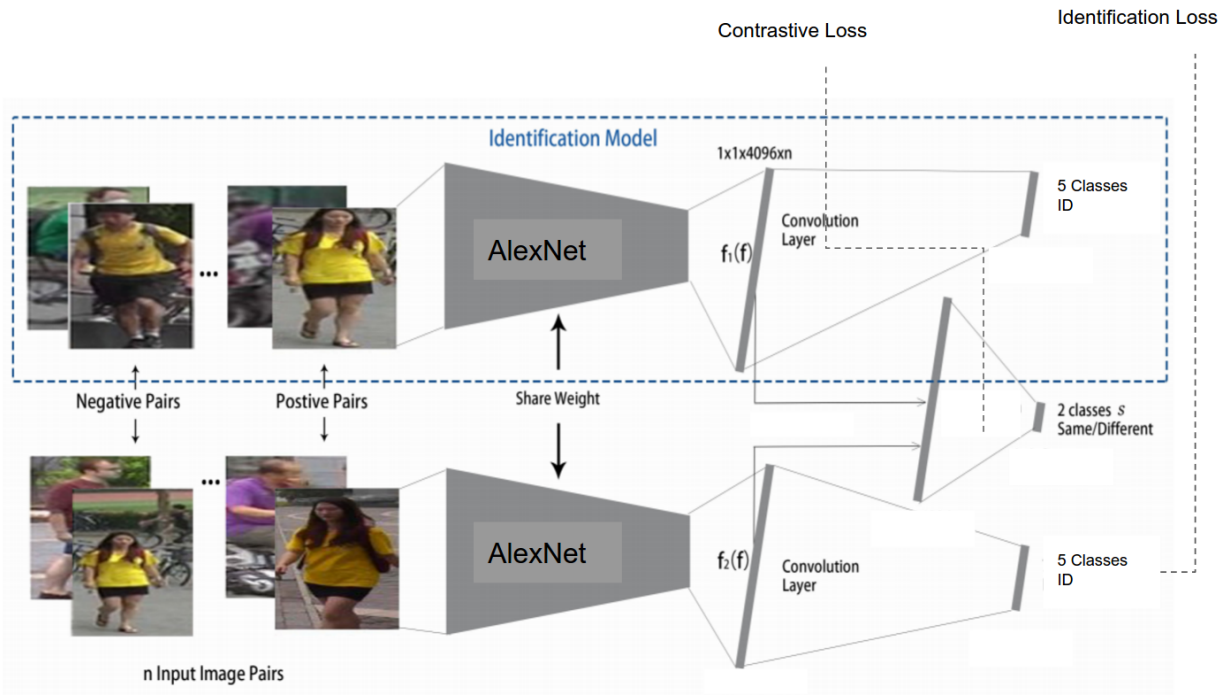


Figure 3.2: Architecture of modified Siamese network used for training floor identification classifier

We believe the high performance of our network is because the training and testing images were taken in similar setting and that the inter-class variation is quite high in the dataset. Another important reason for the good performance of our floor identification module is the use of the Siamese architecture during training. Siamese architecture by its nature increases the network's ability to maximizes inter-class variability while minimizing intra-class variability.

From the demonstrations of floor recognition, we can see that it is possible to use automatic annotation to label many other data from CoBot.

For example, if we want the CoBot to describe the crowdedness of its path, we can count the people it sees and then based on the number of people in a segment of the path we can classify it as crowded or not. CoBot can use this tagging while verbalization to describe the task. The CoBot using our floor-identification the CoBot, after finding which floor it is currently on, can decide on next stage of its task plan. If the CoBot has reached the correct floor, then it proceeds to execute the remaining part of its task plan after updating its map with the one identified using our floor-identification module, or it uses the elevator again to try and reach the correct floor.

3.1.4 Summary and Conclusion

In the previous chapter, we introduced verbalization with manual groundings of data with natural language phrases, this made it difficult to scale. To solve this problem, we propose automatic annotation of the data using techniques like machine learning. We demonstrate automatic annotation with a particular problem of annotating which floor our mobile service robot CoBot, is

currently in after exiting the elevator. We have created and trained a DNN based classifier to identify which floor CoBot has reached after exiting the elevator. The network uses modified Siamese based architecture during training and uses both the contrastive and the identity loss.

We have described our training methodology and the dataset in this work. With this specific example, we provide a demonstration of how to create an automated annotation system and thus scale verbalization. We have evaluated our identification module in a newly collected testing dataset obtained while the CoBot is performing multi-floor navigation tasks. Our network was able to classify all the images in the testing dataset correctly.

3.2 Classifier-Based Evaluation of Image Feature Importance

There has been a tremendous advancement in the performance of deep neural networks (DNNs), specifically in the task of image recognition using deep convolution networks (CNNs) [23]. Because of this, there is much interest in understanding how these complex networks work. A variety of visualization techniques have been proposed to indicate which pixel features are most important for CNNs to determine their classification prediction on a given image (e.g., K. et al. [20], R. et al. [37], Zintgraf et al. [56]). For example, K. et al. [20] uses the gradients to determine which pixels are important. Zintgraf et al. [56] proposes an occluding the image systematically with a Gaussian square patch and observing the confidence scores to determine the important pixels in the visualization.

With such different algorithms to determine pixel importance, we are interested in comparing the regions that each one finds and evaluating which are best. Most current techniques to evaluate visualizations are qualitative [52, 56] or use a human study [37] and determine which most appeal to people. While these studies successfully determine the best methods for displaying important pixels, they do not compare which algorithm finds more important pixels with respect to the predicting CNN. Recently, a quantitative method was proposed to perturb a random subset of important pixels and then reevaluate the perturbed images to observe how the prediction confidence changes [41]. However, the non-determinism in the perturbations could introduce new artifacts in the image which might confuse the classifier rather than measure confidence.

In this work, we propose two new quantitative deterministic metrics for evaluating the relevance of an important region. Like Samek et al. [41], our metrics also measure the classifier confidence of the important region. However, we compare the increase in accuracy to a baseline image rather than the decrease in accuracy by perturbing the pixel values. Our baseline image is not confident in its prediction. By adding the important pixel region to this baseline image, in our first metric Simple Confidence Gain (SCG), we measure the effective increase in confidence of only the important pixels and normalize the increase by the confidence of the original image. Our second metric - Concise Confidence Gain (CCG) - aims to ensure that the new image formed by adding the important pixels to the baseline image is classified correctly. Additionally, CCG takes into account the size of the region to reward the smallest region of most informative pixels.

Using these new metrics, we contribute comparisons of three different algorithms for finding important regions of images - occluding patches [51], gradients of the class score with respect to the image [20], and Contrastive-Marginal Winning Probability (C-MWP) [52] - on two different datasets - Place365 [55] and our own dataset containing images of various floors in our build-

ing. In addition to contributing a metric to compare importance functions, we also contribute a technique and evaluation procedure to find most concise pixel regions by combining multiple important regions together. Our results indicate that the most important pixels according to our metrics are those that different algorithms can all agree on. We conclude that our measures of important pixels can be used in conjunction with subjective measures of human preferences to evaluate new importance measures and visualization techniques.

3.2.1 Related Work

A variety of deep visualization techniques have been developed to understand CNNs (e.g., K. et al. [20], R. et al. [37], Zintgraf et al. [56]). We roughly divide these techniques into two categories - class model visualization and image specific visualization. Class model visualizations such as K. et al. [20], Yosinski et al. [50] aim to understand how the neurons in the network contribute to the classification. Image specific visualization techniques aim to find what features the CNNs have learned to look for in each image [20, 37, 51, 52, 56].

In our work, we focus on evaluating image specific visualizations for the important features that they highlight. We refer to their feature-finding algorithms as importance functions. For example, K. et al. [20], Zhang et al. [52] have developed error backpropagation-based techniques to find the importance of different regions of an image for a prediction by computing gradients with respect to the image. The work has been extended to evaluate activations of particular neurons rather than pixels in images. Zeiler and Fergus [51] have developed a technique for sensitivity analysis by occluding patches in the image. [56] has also created a procedure for finding importance function using occluding patches. While we evaluate our new metrics on three importance functions K. et al. [20], Zhang et al. [52], Zintgraf et al. [56], it can be applied to any techniques that find features that are important to classifiers.

There are relatively few evaluation techniques for comparing these importance functions. R. et al. [37] use human studies to compare different importance function's ability to discriminate between classes. Human studies only evaluate the quality of the function's visualization from a human's point of view, and do not give any measure of how well the function has captured what the network has learned. Samek et al. [41] proposes an algorithm to objectively evaluate importance functions by randomly perturbing a small region around the important pixels of the image and observing the confidence scores from the classifier. They do this random perturbation sequentially in order of importance for 100 relevant pixels. The confidence scores during the process are then used for comparing different importance functions. However, since the work randomly perturbs the pixels of the images it could introduce new artifacts in the image which might confuse the classifier. We also propose objective metrics to compare importance functions by observing the confidence changes on a classifier. Our proposed metrics are comparatively faster than the prior approach 1:100 times - and are deterministic.

3.2.2 Importance Functions

We assume that a CNN classifier C outputs $p(I = y|w)$, the probability of an image $I \in [0, 1]^{c*N}$ with c channels (i.e., 3 for R,G,B) and N pixels having classification y given the trained weights w. For clarity, we will refer to the ith pixel in the image as $I[i]$. Given C and I, an importance

function determines the pixels that have the largest impact on the classifier for predicting it as y. The output of these importance functions is a *heat map* $H \in [0,1]^N$ containing relative importance of each pixels in the image. A variety of importance functions have been proposed for explaining the classification predictions of CNNs. In this section we introduce the three visualization techniques we use as a feature importance functions in our work - occluding patches *occ* [51], gradients of the class score with respect to the image *grad* [20], and Contrastive Marginal Winning Probability *C-MWP* [52].

Occluding Patches (*occ*)

A gray square patch of a fixed size called an occlusion mask is used to systematically occlude parts of the input image and the confidence scores in these images are used to find the heat map. The idea behind the approach is if a key feature in an image gets occluded, then the classifier's confidence will fall. The algorithm first creates a visibility mask V as the inverse of the occluded patch:

$$V[i] = \begin{cases} 0 & \text{if } I[i] \text{ is occluded} \\ 1 & \text{otherwise.} \end{cases}$$

By weighing the non-occluded regions of the image by their classifier accuracy, the highest confidence regions are those that are most important in the heat map H:

$$H = 1 - \frac{\sum_{j=1}^{J} p(I_{o,j} = y|w) * V_j}{J} \tag{3.1}$$

where $I_{o,j}$ is the j^{th} occluded image and J is the number of images generated by systematic occlusion. Note that the heat map H is a function of the size of the occluding patch, so we can evaluate different sized patches to understand how their resulting heat maps change our importance measures.

Gradients (*grad*)

The heat map H, for the gradient visualization technique represents the magnitude m of derivative of the classification confidence with respect to the image. The magnitude of ith pixel m_i represents the sensitivity of the network's prediction to the change in that pixel's value.

$$m_i = \frac{\partial p(I = y|w)}{\partial I[i]}$$

We expect that the probability scores are more sensitive to the change in values of the important features than others. In the case of a multichannel image– $c > 1$, only the maximum absolute value of the gradients for each pixel across all color channels are considered:

$$H[i] = \begin{cases} |m_i| & \text{if c = 1} \\ max_c|m_i| & \text{otherwise.} \end{cases}$$

Note that since the gradients are pixel-wise importance values for the image, the heat map is generally of high entropy thus lacks continuous important image regions.

Contrastive Marginal Winning Probability (*C-MWP*)

The work models the top-down attention (importance function) of a neural network classifier, using probabilistic WTA formulation [19]. The WTA identifies the neurons that are relevant to the task in a particular layer using the backpropagation technique *Excitation Backprop* that computes the Marginal Winning Probabilities (MWP). After identifying the relevant neurons, we compute the most relevant neurons' corresponding receptive field and the generate the heat map for the image.

The MWP map's discriminative ability can be improved by backpropagating contrastive signals to produce Contrastive-MWP maps. Contrasting signals for the image belonging to class A, is the difference in the gradients of A classifier and $not\ A$ classifier. C-MWP is able to achieve better discriminative visualizations than MWP. The heat maps generated at higher-layers generally have lower spatial accuracy than the ones from lower-layers, so our implementation (described in Section 5) selects a lower-layer to form the heat map.

3.2.3 Analyzing Important Features

Given an image and classifier that determines what the image contains, our goal is to understand which pixels of the image are most important to its classification. Because different algorithms may determine that different pixels are important, we are interested in finding a measure of goodness to compare different classifier's important regions. Prior work has focused on allowing users to rate visualizations overlayed on the image as a measure of goodness. In contrast, we propose to reduce variability in subjective preferences by instead utilizing the classifier itself as our measure. This proposal also captures the relevance of the important regions to the classifier which is not captured in the human studies. We compare the goodness of different important regions based on how they affect the classifier accuracy as well as the size of the important region.

Problem Formulation

In the previous section, we provided some examples of importance functions importance(I, C) that take as input an image I and a classifier C and output a *heat map* $H \in [0,1]^N$ that contains a measure of the relative importance of each pixel. The map will have higher values for those pixels, which are considered important and lower values otherwise. Note that different importance functions may output different heat maps. The heat map can be visualized on top of the image to allow a user to see the relative importance of each pixel. For example, Figure 3.3(c) shows the visualizes the heat map for image in Figure 3.3(a).

While the heat map is useful for visualization, we propose the use of a binary mask as shown in Figure 3.3(d), that signifies whether each pixel is important or not. A mask $M \in \{0,1\}^N$ is created such that each pixel i takes value:

$$M[i] = \begin{cases} 1 & \text{if important} \\ 0 & \text{otherwise.} \end{cases}$$

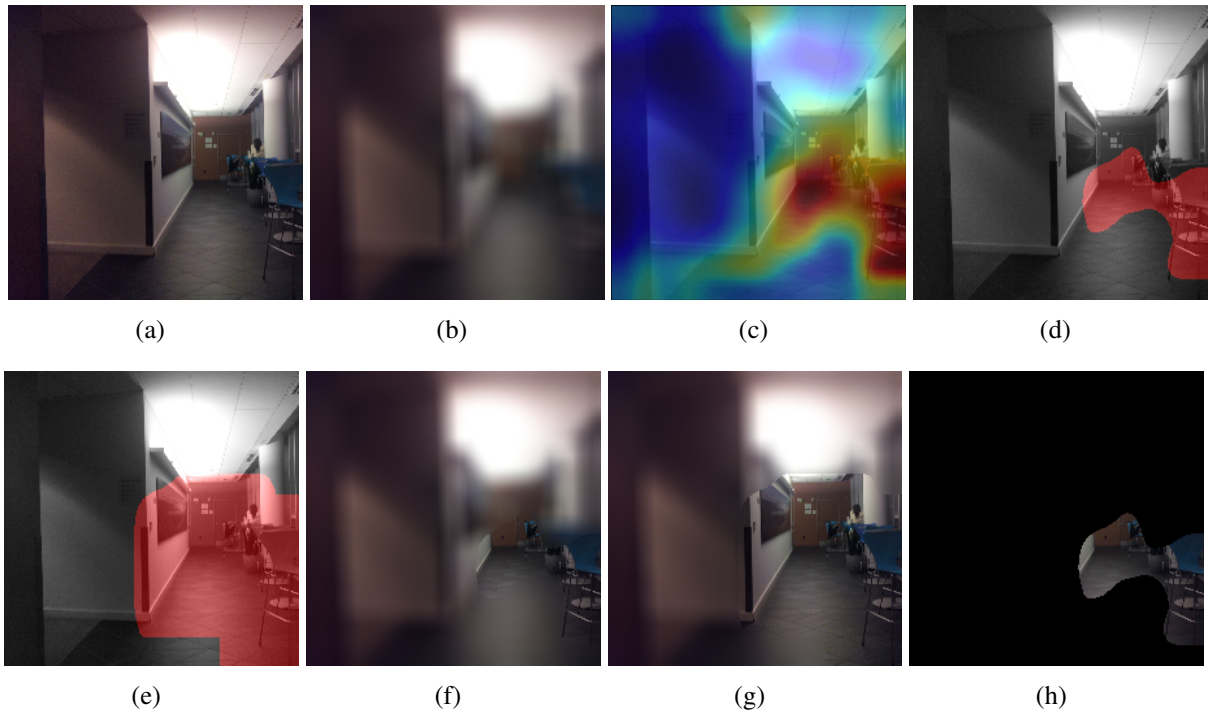


Figure 3.3: (a) is the original image, (b) is the base image obtained from using Gaussian kernel G_k, and (c) is the heat map obtained using *C-MWP*, where red and blue represents the most and the least important pixels. (d) is the mask obtained after thresholding the heat map (c) for top ρ=5% pixels, and (e) is the mask obtained after growing the regions of the mask in (d). (f) and (g) are the hybrid images created using mask in (d) and (e) respectively using the base image (b). (h) is the hybrid image obtained using mask in (d) and a base image obtained using zeros kernel Z_k.

In this work, we use thresholding over $H-$ segmenting top ρ % elements, to determine importance but other techniques such as graph cut [10] could also be used.

We propose the creation of a *hybrid image* $I_{M,K}$ as shown in Figures 3.3(f) and 3.3(h) that contains the important pixels indicated by M and less informative remaining pixels. The classification accuracy of the hybrid image reflects only the contribution of the important pixels. In order to create the hybrid image, we first create a base image I_K which represents the image I altered using a kernel function K. The kernel is chosen such that it renders the pixels comparatively less informative for classification. In this work, we have explored two such kernels: a Gaussian kernel G_k, which blurs the image refer Figure 3.3(b), and a zeros kernel Z_k, with zeros in its all element values. The hybrid image $I_{M,K}$ is then created using the original image I, the mask M and the base image I_K such that:

$$I_{M,K}[i] = \begin{cases} I[i] & \text{if } M[i] = 1 \\ I_K[i] & \text{otherwise.} \end{cases}$$

Next, we propose two measures of confidence gain to reflect the improvement in the classifier's confidence when the base image $p(I_K|w)$ is converted to the hybrid image $p(I_{M,K}|w)$.

Metric: Simple Confidence Gain (SCG)

The Simple Confidence Gain (SCG) measures the ratio of the improvement in accuracy of the base image to the hybrid image containing only important features compared to the improvement in accuracy of the base image to the original image. Note that we assume that the kernel is predefined and the same for all compared masks M.

$$SCG(I, I_K, I_{M,K}) = \frac{p(I_{M,K} = y|w) - p(I_K = y|w)}{p(I = y|w) - p(I_K = y|w)} \tag{3.2}$$

A value of SCG close to 1 indicate that the masked pixels contribute highly to the classifier accuracy. Values close to 0 indicates that the mask has little contribution.

Metric: Concise Confidence Gain (CCG)

The Concise Confidence Gain CCG is a modified version of SCG. It improves upon the SGC in two ways. First, it requires the important region to produce an accurate classification. Second, it measures the conciseness of the important region necessary to classify the image correctly.

The idea is to increase the region under the mask M to form a new accurate mask AM as shown in Figure 3.3(e), such that the classifier predicts the class y of the hybrid image $I_{AM,K}$ correctly as shown in Figure 3.3(g). It should be noted that there are many ways of increasing the size of the mask. For example, we can simply construct a new mask from the heat map with an increased threshold ρ. In this work, we have chosen to grow the mask using the dilate operation which enlarges the boundary regions of the foreground pixels. With the new hybrid image the CCG metric is calculated as: $CCG(I, I_K, I_{AM,K}) =$

$$\frac{\big(p(I_{AM,K} = y|w) - p(I_K = y|w)\big) * N}{\big(p(I = y|w) - p(I_K = y|w)\big) * A_{AM}} \tag{3.3}$$

where $A_{AM} - \sum_i AM[i]$ is area of the image masked by M_f. We note that two different masks that are originally the same size need not be the same after dilation. The dilated area depends on the geometry of the mask, so we divide the relative confidence by the ratio of the area of the new accurate mask A_{AM} to the size of the image I (N).

High values of CCG reflect more important masked regions compared to the those with lower values. Additionally, because CCG divides the relative confidences by their areas, CCG also values the conciseness of the mask. The complete algorithm for finding CCG is shown in Algorithm 2.

Combining Important Features

We hypothesize that combining the features from each individual importance functions can improve the conciseness of the most important region. In particular, we propose that the most important region is located at the intersection of the individual important regions. Pixels that several different functions can agree are important are likely to be the most discriminative and will have higher values of our CCG metric compared to the regions found by individual importance functions.

Algorithm 2 Procedure for calculating CCG

Input: Heat map H, Image I, Kernel K, Mask percent ρ, classifier parameters w, Image class y

/* Creating the mask */

$M \leftarrow$ GetBitMask(H, ρ)

/* Creating the base image */

$I_K \leftarrow$ TransformImage(I, K)

/* Loop until prediction matches the correct class */

repeat

$I_{M,K} \leftarrow$ CreateHybrid(M, I)

/* Prediction for the hybrid image */

predicted class = Classify($I_{M,k}$, w)

/* Break if the predicted the correct class */

if predicted class == y $I_{AM,k} \leftarrow I_{M,k}$

/* finding Area of the mask */

$A_{AM} \leftarrow$ TotalNoElements(M)

break

end if

/* Grow the mask */

M $\leftarrow$ DilateGrow(M)

until

CCG $\leftarrow$ Compute with Equation 3.3

There are many different ways to find the intersection of important regions. For example, it is possible to directly add two heat maps and then segment it to generate a new mask. In this work, we take the intersection of the masked regions. These small sets of important features can also be used as an explanation to people when the individual importance functions are complex for example when the classification task is scene recognition.

3.2.4 Experiments

In order to evaluate our metrics and our algorithm for combining important features together, we performed experiments on several different datasets and the three importance functions described in Section 3. We will first describe our datasets and the corresponding CNNs that we have used in our experiments. Then, we present our experiments for evaluating the important features using our proposed measures. Finally, we demonstrate our metrics on combinations of important features.

We have considered two datasets - Building-Floor refer Section 3.1.2, and Places365-Standard [55] for our evaluations. We chose to test our algorithms on scene recognition data because the images are complex yet, there are few classification labels.

			SGC				CCG			
			Floor		Places365		Floor		Places365	
ρ	Config		G_k	Z_k	G_k	Z_k	G_k	Z_k	G_k	Z_k
25%	occ	10	43	34	31	23	114	98	93	77
		50	28	38	22	21	103	96	85	82
		100	36	27	18	18	105	94	81	80
	grad	0	46	24	39	19	113	70	110	82
		2	61	31	43	20	107	74	112	88
		5	57	30	44	25	116	88	116	90
	C-MWP		**71**	**39**	**50**	**37**	**120**	**113**	**137**	**115**
5%	occ	10	25	22	20	16	161	132	103	88
		50	16	19	14	13	135	136	96	86
		100	20	22	11	12	119	113	91	84
	grad	0	18	22	26	14	163	70	128	83
		2	35	**28**	**29**	15	**237**	98	128	94
		5	31	22	28	18	189	122	130	96
	C-MWP		**40**	22	27	**21**	208	**162**	**162**	**125**

Table 3.1: Average SCG values (*100) for individual masks with 25% and 5% top pixels. C-MWP performs best (bold) in almost all datasets, base image kernels, and values of ρ.

Places365-Standard

The Places365-Standard dataset [55] contains images from 365 categories for scene recognition. It contains scenes from both indoor and outdoor locations. We used the Places365-AlexNet model provided by the authors in our work on this dataset, which was trained using $\sim$1.8 million images. For our testing, we selected 200 random images from the dataset.

Experimental Procedure

We evaluated the three importance functions described in Section 3.2.2. For occ, the heat map generated will be a function of the size of the occluding patches. Hence we evaluated the algorithm as we also varied the size of the occluding patches $\in \{10, 50, 100\}$ pixels. In the case of $grad$, since the heat map generated is of high entropy we dilate the heat map 0, 2, and 5 times with a 3x3 kernel. Dilating smoothens the heat map and improves the continuity of important regions as shown in Figure 3.4. For *C-MWP*, we use the output from the $pool2$ layer for both of the networks as we lose the spatial accuracy at higher layers. For our *C-MWP* implementation, we used the source code provided by the authors.

Given the heat maps generated by each importance function, we then created the masks using simple thresholding to ensure that a certain percentage (ρ = 5% and 25%) of the top features are consistent across tests. To create the base images, we used two different techniques - a Gaussian kernel G_k of size 17*17 for creating the blurred base images and a zero kernel Z_k to substitute black for the non-important pixels. While finding the accurate hybrid image between each iteration, we grow the regions of the mask using a 3x3 dilate operation. When testing

Figure 3.4: The image masks for the (top) *occ* importance function and (bottom) *grad* importance function generated with the parameters ρ=25% and with dilation = {0, 2, 5} respectively for one image from the Building-Floor dataset (left three images) and one image from the Places365 dataset belonging to the class $amusement\ station$ (right three images).

Figure 3.5: A side by side comparison of occ (patch size = 10), *grad* (dilation = 5), and *C-MWP* respectively on an image from the Building-Floor dataset and the Place365 dataset (ρ=25%).

the combinations of the importance functions, we evaluated all pairs on both datasets, namely *grad+occ*, *C-MWP+grad*, and *C-MWP+occ*. For the experiments combined with *occ*, we have fixed the size of the patch to be 10, and for $grad$, the number of dilating operations is 5.

During the experiments, the cases where $p(I = y|w) - p(I_K = y|w)$, and $p(I_{M,k} = y|w) - p(I_k = y|w)$ are less than zero are not considered, as it violates our assumption that the kernel function renders the pixels less informative for the classifier. An example for such a case could be a dark image belonging to the class $night$, using a zero kernel Z_k will make it the best image for the class. During CCG calculation, we also fix the maximum number of times a region can be grown to 50, and we ignore the cases when it exceeds this margin. This is to avoid the growing mask from using other important features that were not captured by their importance function.

3.2.5 Results and Discussion

Comparing Importance Functions

The quantitative results of the evaluation of the individual importance function's masks are shown in Table 3.1. The masks obtained from the variation of occ and $grad$ on one image from each dataset are shown in Figure 3.4(top) and 3.4(bottom). A side by side comparison of the mask

Figure 3.6: A side by side comparison of the three pairs of importance functions (*grad+occ*, *C-MWP+grad*, and *C-MWP+occ* respectively) on an image from the Building-Floor dataset and the Place365 dataset (ρ=25%).

		SCG				CCG			
		Floor		Places365		Floor		Places365	
ρ	Importance	G_k	Z_k	G_k	Z_k	G_k	Z_k	G_k	Z_k
25	*best single*	**71**	**39**	**50**	**37**	120	113	137	115
	grad+occ	25	30	20	15	223	139	122	89
	C-MWP+grad	43	32	28	16	**239**	169	**155**	**122**
	C-MWP+occ	29	30	17	13	225	**171**	154	114
5	*best single*	**40**	**28**	**29**	**21**	237	162	162	125
	grad+occ	6	18	7	9	209	166	133	103
	C-MWP+grad	16	22	11	9	**330**	230	**186**	**137**
	C-MWP+occ	7	15	4	5	249	**269**	179	133

Table 3.2: Average SCG and CCG values (*100) for all combinations of importance functions with 25% and 5% top pixels. The best single importance value is also included (the bold value from each column of Table 1). Patch size for *occ* is 10, and number of dilate operations for *grad* was 5.

obtained for occ with patch size = 10, $grad$ with number of dilations=5, and *C-MWP* are shown in Figure 3.5.

Among the variation of *occ*, we find that the patch size of 10 on an average performs better than 50 which is better that 100. With the patch size of 10, the mask tends to be spread throughout the image compared to 100, refer Figure 3.4(top). This is because the importance function rates all pixels covered by large occlusion patches as important when only a small area under the occlusion is actually important. A patch size of 10 occludes the smaller important features "are more concise" and hence better capture what the network has learned.

Similarly for *grad*, dilating the region 2 and 5 times performs as good or better than not dilating the heat map. As can be seen in Figure 3.4(bottom), when the important features are too small (as with 0 dilation), the mask has high entropy and this makes the hybrid image not informative enough for the classifier.

Comparing the three importance functions, *C-MWP* outperforms the other two by a significant margin, followed by *grad* and then *occ*. The result matches the conclusion in Zhang et al. [52], which compares *C-MWP* and *grad* qualitatively and shows that the former is able to produce better localization of objects. However, our technique does not require human evaluation for similar results. We conclude that our metric compares importance functions objectively and

indicates high values when the important regions lead to higher classifier accuracy.

Effect of varying threshold

We varied the percentage of the top pixels considered important while generating the mask, as 5% and 25%. We find that SCG values for 25% are higher that for 5%, and CCG values for 5% are greater that 25%. This is expected, as when the percentage decreases the size of important regions decreases but since only the top pixels are retained, their quality increases. Selecting values of ρ for future evaluations of importance functions should take this tradeoff into account.

Analyzing combination of important functions

The quantitative results for the combination of importance function's masks are shown in Table 3.2 and the masks are visualized in Figure 3.6. Considering only the average CCG values, the combination of two importance functions performs much better than the individual ones, as predicted. However, their average SCG values are much less than the individual ones, the reason being the SCG measure considers only the amount of information gained and not the density or conciseness of the features like CCG.

There are far fewer discriminative features in the combined masks (Figure 3.6) compared to the individual ones (Figure 3.5). This result is most apparent in the Building-Floor dataset, where the combined masks have captured the glass door and the hallway behind it while the individual masks have captured other features as well. Upon human inspection, we agree that the glass door is the most discriminative feature of that image and of the floor because there are other floors with the similar wall, carpet, and hallways, but not a glass door (see Figure 3.1 for examples of each floor). Similarly, in the image from the Places365 dataset, we can see that the individual visualizations capture additional features that are not relevant to the class of the image– $bus\ station$, but the combinations have captured only the features that are on the bus.

Among the combinations of the three importance functions considered *C-MWP+grad* performs better than *C-MWP+occ* by a small margin. And, both *C-MWP+grad* and *C-MWP+occ* outperforms *grad+occ* by a large margin. This reflects our result in the previous subsection which shows that *C-MWP* performs the best followed by *grad*. We can conclude that combining different importance functions results in a more concise region of important features, which can be beneficial for preventing information overload to humans for visualization as well as finding the most discriminative pixels for classification.

3.2.6 Summary and Conclusion

There has been much interest in how image recognition especially using deep convolution networks (CNNs) works. Prior work to visualize important regions of images that are most discriminative for classification have been largely subjective, depending on humans to rate the visualizations. In this work, we contribute two objective metrics– SCG and CCG, for finding how important a region of an image is for a classifier. Both metrics compare the accuracy improvement of the important region over a baseline image. Our CCG metric also takes into account the conciseness of the region as well as requiring the classifier to classify the image accurately.

We have used these metrics to compare three different visualization techniques on two scene recognition datasets. Our results on the relative performance of these techniques correlate with the qualitative results provided in [52]. However, our results do not require a human to rate the visualizations. We also show that the new metrics can be used to find the best hyperparameter (ρ) for the visualization techniques. Finally, we motivate the need for combining different visualization techniques by demonstrating using our metric that taking an intersection of their important regions results in a more concise set of better discriminative features than the individual ones.

Chapter 4

Explaining the Robot's Actions

In the previous part of our work, we use verbalization to describe a robot's actions like narrating what path it took to reach its current place. In this chapter, we look at explaining a robot's actions. Explanations are different from descriptions in the sense that in explanations, we describe the reason why the robot took that path. For example, for a robot's path, a description can be 'I reached here via the Fifth Avenue' while an explanation can be 'I arrived here via the Fifth Avenue because it was the shortest path.'

When robots explain their actions, they are trusted more by people around and using them. As a first step towards making robots explainable, we make the floor identification module that we used for demonstrating automatic annotation explainable, as we describe below.

We propose a new method to explain an image based classifier used for scene recognition to the people in natural language. The algorithm we develop in this chapter creates groundings for the explanation automatically. Our approach to generating an explanation to the classification consist of the following steps:

- Identify the important regions in the input space for the classification
- Extract *explainable features* from the identified important regions
- Generate an explanation using the features

First, to explain a decision by any module, we need to find what part of the input has influenced or most influenced the decision. We identify important features for the DNN's classification using deep visualization techniques or importance functions. Next, similar to the step in verbalization, we need to ground the important regions with natural language tags. We generate groundings for the important regions in the image by using a separate image based object detection network, once again leveraging the concept of automatic annotation. Finally, similar to our variable verbalization algorithm, using all the explainable features we generate natural language sentences. We demonstrate our explanation generating technique by generating an explanation for all the classes the floor identification module is trained on.

We start by describing some of the related works and a brief background for the proposed algorithm. In Algorithms section, we describe the algorithm for generating the explanation for a scene recognition classification followed by the details of our implementation for generating an explanation for CoBot's floor recognition network. Finally, we analyze the results and discuss the future work.

4.1 Related Work

The field of explainable artificial intelligence has only recently started getting attention from the research community. In this work, since we only look at explaining a machine learning based classifier we will limit our discussion to the same. Finding the important features or feature size reduction [7, 8] in the context of machine learning are well-researched areas. But these work does not focus on generating an explanation or making them interpretable, as our work focuses on generating explanations we will not discuss these topics either.

A common approach to generating model-agnostic explanations for a classifier is learning an interpretable model based on the predictions of the original model [3, 39, 42]. In Ribeiro et al. [39] the authors explain the classification of any model by learning a separate sparse linear model that locally approximates the original model to explain its decisions. The work demonstrates explaining text models. However, it is unclear on how to extend this approach to domains such as images or tabular data. The work also does not focus on generating explainable or making the locally learned model understandable to humans.

Hendricks et al. [18] focuses on generating textual explanations for fine-grained classification of 200 bird species using deep network from their images. The explanations are generated using ResNet [17] features extracted from the entire image, and the explanations are conditioned both on the image and the class predictions. The explanation generation module is trained on the bird's descriptions. While such an approach might be possible for cases where the vocabulary and type of explanations are similar within and across classes, this will not generalize to other scenarios.

An architecture for querying the robot's data and answering the questions from the user has been discussed briefly in Lomas et al. [28]. The demonstration focusses on a simple example as a proof-of-concept, and the generalization of the querying or answering mechanism have not been discussed.

Our work focuses on generating an explanation for image based scene classifiers. Our approach for generating explanations does not require any training descriptions since we use the objects in the scene to describe it.

4.2 Background

To recap on the notations and definitions we introduce again some of the term we describe in previous chapters. Let classifier C output $p(label = y|I, w)$, the probability of an image $I \in [0, 1]^{c*N}$ with c channels (i.e., 3 for R,G,B) and N pixels having classification y given the classifier's weights w. For clarity, we will refer to the ith pixel in the image as $I[i]$. Given C and I, an importance function determines the pixels that have the largest impact on the classifier for predicting it as y. The output of these importance functions is a *heat map* $H \in [0, 1]^N$ containing relative importance of each pixels in the image.

Let H be the importance heat map for the classifier C, classifying a given image I belonging to class y, into class. We find the important features for the classification by segmenting the heat map to get a binary heat mask M. We generate the mask by segmenting top $\rho\%$ of the pixels from H.

In this work, we use the gradient based importance function [20] we discussed in the previous section. To recap, the heat map H, for the gradient visualization technique represents the magnitude m of derivative of the classification confidence with respect to the image. The magnitude of ith pixel m_i represents the sensitivity of the network's prediction to the change in that pixel's value.

$$m_i = \frac{\partial p(I = y|w)}{\partial I[i]}$$

We expect that the probability scores are more sensitive to the change in values of the important features than others. In the case of a multichannel image– $c > 1$, only the maximum absolute value of the gradients for each pixel across all color channels are considered:

$$H[i] = \begin{cases} |m_i| & \text{if c = 1} \\ max_c|m_i| & \text{otherwise.} \end{cases}$$

When a robot navigates across many floors of a building or multiple buildings, one major challenge that it has when exiting the elevator is localizing itself to determine which floor it is currently on [47],[46]. We collected a Building-Floor dataset in one of our university buildings. Each image contains the scene just outside the elevator from six different floors of the building. The goal of the classifier trained on this dataset will be to recognize which floor the image belongs to.

For each of the floors in the building, five images were taken at a specific locations that our robot stops at, with slight variation in position. To simplify the analysis all the images were taken at the same time of the day, and the effects of people moving around in the building are not considered. The training data consists of three images, and the remaining two images form the testing dataset.

In order to classify the floor for each image, we chose to use a CNN based on Siamese architecture [11] C, because it has been shown to perform well in one-shot learning problems [22]. Our training network of nine layers followed the AlexNet [23] in a modified Siamese architecture proposed in [43, 53] , which combined the identification– Softmax, and the verification loss– Contrastive, for better performance. Our main reason for combining identification and verification loss with a pre-trained network is to reduce overfitting which could happen when the complexity of network is higher than the data. During training, the first seven layers of our network were initialized from Places205-AlexNet which was trained in Places205-Standard dataset and provided by the authors [54]. The remaining two layers were trained from scratch. During training contrastive loss was utilized in the eighth layer which is a dense layer of 1000 units, while the softmax loss was employed in the ninth layer. During testing, our network was able to classify all the images in the dataset correctly.

4.3 Algorithm

In this section, we describe the algorithm we propose to generate the explanation for scene recognition task. First, we generate explainable features that are important for the classification using

a separate object detector and an importance function. As described earlier we define explainable features as the features that can be grounded using natural language. Then we create a class-wise explainable feature for each class using the image in training dataset. Finally, we generate an explanation for a given image by using the corresponding class-wise features and the explainable features of the given image. We describe our algorithm in detail in the following subsections.

4.3.1 Generating explainable features

Using an image-based multi-object detector D, we find a set of objects $\{o_0, ...o_l\}$ in the image and their corresponding bounding boxes $\{(xmin_0, xmax_0, ymin_0, ymax_0),\ ...\ (xmin_l, xmax_l, ymin_l, ymax_l)\}$. We then create explainable features for the whole image using the detected object ids o_i and their corresponding location in the image. To allow some spatial invariability in the position of the detected objects, we discretize the image into g grids and use the grid index gId of the detected object's center as a proxy to their location in the image. The explainable features E for an image is of the form $\{\ (gId_0, o_0), ...(gId_l, o_l)\}$.

An image taken by the robot during its tasks may contain objects that are not related to the scene. To avoid the objects that are not related to the scene we filter the explainable features of the objects that are not in the most important region of the scene. For example, a dog might have wandered into kitchen space, since the scene belongs to the kitchen category the dog will not be considered as an important feature by the network. We remove features in E with objects having less than a threshold ratio Ov_r, of their bounding box overlapping with the importance heat mask M.

Another way in which we filter unrelated features in E by removing features containing objects in rejection corpus O_r. The objects in the corpus are generally not present in the area in which the robot operates in, or containing objects that are not preferred in our explanation. We construct O_r by examining the corpus of objects D is capable of labeling. After applying the filters discussed above, we get relevant explainable features E_r, for I. We apply filters to E as:

$$E_b = \{e_i \mid \frac{sum(M[xmin_i : xmax_i, ymin_i : ymax_i])}{(xmax_i - xmin_i) * (ymax_i - ymin_i)} > Ov_r,\ \forall e_i \in E\}$$

$$E_r = E_b \setminus \{e_i \mid o_i\ \in O_r,\ \forall e_i \in E\}$$

where M[x1:x2, y1:y2] is a submatrix of M with (y2-y1) rows and (x2-x1) columns, it containing elements from M in x1 to x2 columns and y1 to y2 rows. Summarizing, we generate E_r for I with the following steps:

- Find the class y, the test image belongs to using C
- Compute heat mask M using the importance function
- Using D, g, O_r and M generate relevant explainable feature for the image E_r

4.3.2 Class-wise explainable features

We then consider a second training set for identifying class-wise explainable feature $E_{c,r}$. This training set can be same as the training set used for training the classifier C. $E_{c,r}$ contains a set

of explainable features for each class, it can be considered as a definition of the classes. Given a set of training images, there are multiple ways to generate $E_{c,r}$. In this work, we construct $E_{c,r}$ by simply taking a class-wise union of E_r for all the training images.

In this work, we assume that the C, importance function and D are good enough to produce unique class-wise explainable features, i.e., all the classes are distinguishable by using only their explainable feature. We use $E_{c,r}$ as a dictionary of features defining the class.

4.3.3 Generating explanations

For a given test image I_t, belonging to class y_t, we generate relevant explainable features $E_{r,t}$ as described above. Then, we create the class specific explainable features $E_{s,r,t}$ by considering both the class-wise explainable features $E_{c,r}[y_t]$ for y_t, and $E_{r,t}$. Again, there are multiple ways of generating $E_{s,r,t}$ given $E_{c,r}[y_t]$ and E_{rt}, in this work we take intersection between them to get $E_{s,r,t}$.

Finally, we use grounding and template based natural language generation to convert $E_{s,r,t}$ to natural language explanations. We convert each of the features to natural language phrases using a set of groundings. The groundings for the object labels o_i are themselves. For the grid id gId_i, the groundings are related to their spatial location, for example for g=3 the grounding for gId=(0,2) can be 'far right' and for gId=(1,1) can be 'center.'

4.4 Experiments

In our experiments, we used Yolo9000 [38] as our multi-object detector D, we used the network weights trained and provided by the authors. Yolo9000 uses a hierarchical tree classification which is built using WordNet [31] concept.

D uses hierarchal classification and the level of hierarchical in the detection is controlled by a threshold $t_{h,D}$ and a separate threshold which controls the threshold for the bounding box $t_{b,D}$. D traverse the tree down, taking the highest confidence path at every split until we reach the threshold th_D and we predict that object class. The hierarchical nature of the classification leads to some objects labeled as instrumentality, things or matter, O_r is constructed using these labels. Our rejection corpus for the building-floor detection dataset is:

O_r = {'home appliance', 'living thing', 'container', 'artifact', 'person', 'conveyance', 'bottle', 'instrumentality', 'whole', 'deep-freeze', 'electric refrigerator', 'machine'}

In our work, we use the building-floor detection dataset to explain the floor classification. The training set for the building the class-wise explainable features are the same as the training set for the scene classifier. For more details on the dataset refer Section 3.1.2. We have used a grid size g=3. This leads us to the groundings like 'top' for (0,1), 'top right' for (0,2), and 'center' for (1,1) gIds.

The heat map generated from the gradient visualization technique is generally of high entropy thus lacks continuous important image regions, hence we dilate the heat map 2 times with a 3x3 kernel. Dilating smoothens the heat map and improves the continuity of important regions.
In our experiments we use the following values for the hyper parameters:

- $t_{h,D}$ = 0.7

Figure 4.1: Example for generating E_r (a) original image belonging to floor 3 of floor-detection dataset, (b) D predictions on the image, (c) heat mask M using gradient visualization technique, and (d) discretization of image into 9 grids.

- $t_{b,D} = 0.1$
- $Ov_r = 0.5$
- $\rho = 0.5$

4.5 Results and Discussion

To consider an example for generating E_r for an given image let us consider an example shown in Figure 4.1 (a). For the original image Figure 4.1 (b) shows the object detection results generated using D, (c) importance mask for the classification of the image to class floor 3, and (d) shows the discretization of image into grids. E produced by using the discretization and output from D is:

['(0,1), electric refrigerator' ,'(1,1), pot', '(2,1), person', '(2,1), living thing', '(2,1), furnishing', '(0,1), pot', '(2,1), chair', '(1,1), chair', '(1,1), seat', '(1,1), instrumentality', '(2,1), container']

In our work, to generate E we group object ids belonging to similar labels for example, 'seat' and 'chair' are grouped to form a single label 'chair', and we discard the repeated features. Applying the filters to E as described in the algorithms section we generate E_r as:

['(1,1), pot', '(2,1), person', '(2,1), furnishing', '(0,1), pot', '(2,1), chair', '(1,1), chair']

As described above we generate E_r for all the images in the training dataset and combining the features class-wise using union operation to generate class-wise features $E_{c,r}$. The class-wise features for the building-floor detection dataset is:

- floor 2 : []
- floor 3 : ['(1,1), pot', '(2,1), person', '(2,1), furnishing', '(0,1), pot', '(2,1), chair', '(1,1), chair']
- floor 4 : ['(1,1), furnishing']
- floor 5 : ['(2,1), chair', '(2,2), furnishing', '(2,2), chair', '(1,1), furnishing', '(1,1), table', '(1,2), chair', '(2,1), furnishing', '(1,1), chair']
- floor 6 : ['(2,1), furnishing', '(2,1), chair', '(2,1), pot', '(1,1), person', '(2,1), sofa']
- floor 7 : ['(2,1), pot']

Figure 4.2: Testing images belonging to (a) floor 3, (b) floor 5, and (c) floor 6

For the test images in Figure 4.2, we generated E_r as described above. We then create $E_{s,r,t}$ for the images using the corresponding class-wise features from $E_{c,r}$ as described in the algorithms section by taking intersection between them. $E_{s,r,t}$ for test images in Figure 4.2:

- floor 3: [‘(2,1), furnishing’, ‘(2,1), chair’, ‘(1,1), chair’, ‘(0,1), pot’]
- floor 5: [‘(2,2), chair’, ‘(2,1), chair’, ‘(2,1), furnishing’]
- floor 6: [‘(2,1), pot’, ‘(2,1), sofa’]

We have used grounding ‘left’, ‘center’ and ‘right’ for x coordinate and ‘top’, ‘center’, and ‘bottom’ for y coordinates in gId. The corresponding explanation using a simple template based natural language generation is:

- floor 3: I am in floor 3 because I see furnishing at right bottom, chair at right bottom, chair at center center and pot at left top.
- floor 5: I am in floor 5 because I see chair at right bottom, chair at right bottom and furnishing at right bottom.
- floor 6: I am in floor 6 because I see pot at right bottom and sofa at right bottom.

In the case of training floor-detection dataset the class-wise features are unique and hence passes our assumption needed for generating the explanations. It is possible that the object detector is not good enough in some situations leading to many empty class-wise features. For floor-detection dataset the $E_{c,r}$ for ‘floor 2’ is empty but since no other class is empty, having no explainable features can be considered unique.

In our experiments in the test dataset in two images from ‘floor 7’ the object detector could not detect objects, and hence contains empty features in E_r and $E_{s,c,r}$. The algorithm is able to explain all other images in the testing dataset.

4.6 Summary and Conclusion

We have developed an algorithm for generating an explanation for scene recognition task in natural language. We demonstrate our algorithm on the classifier which identifies the floor the CoBot is currently on. We first, identify important features for the DNN’s classification using deep visualization techniques or importance functions. Then, we ground the important regions in the image with natural language tags using a separate image based object detection network, once

again leveraging the concept of automatic annotation. We demonstrate our explanation generating technique by generating an explanation for all the classes the floor identification module is trained on.

Chapter 5

Conclusion

We have contributed a novel approach– verbalization, to describe robot's actions and experience in the natural language. We generate explanations while being able to capture all the possible variation in the narration using verbalization space. We demonstrate the use of verbalizations on our mobile service robot CoBot, to describe its route experiences. We validate on 24 routes that a variety of narrations that can be generated from any single plan using our algorithm.

Our verbalization used manual groundings of data with natural language phrases; this made it difficult to scale. To solve this problem, we propose automatic annotation of the data. We demonstrate automatic annotation for verbalization with a particular problem of annotating which floor our mobile service robot CoBot, is currently in after exiting the elevator. We have created, trained and evaluated a DNN based classifier to identify which floor CoBot has reached after exiting the elevator. With this specific example, we provide a demonstration of how to create an automated annotation system and thus scale verbalization.

We contribute two objective metrics– SCG and CCG, for finding how important a region of an image is for a classifier. Both metrics compare the accuracy improvement of the important region over a baseline image. Our CCG metric also takes into account the conciseness of the region as well as requiring the classifier to classify the image accurately. We have used these metrics to compare three different visualization techniques on two scene recognition datasets. We also show that the new metrics can be used to find the best hyperparameter (ρ) for the visualization techniques. Finally, we motivate the need for combining different visualization techniques using our metrics.

In our final contribution, we have developed an algorithm for generating an explanation for scene recognition task in natural language. We demonstrate our algorithm on the classifier which identifies the floor the CoBot is currently on. We first, identify important features for the classification and by grounding the important regions in the image with natural language tags using a separate image based object detection network. We demonstrate our explanation generating technique by generating an explanation for the floor identification task.

Chapter 6

Future Work

A potential avenue for extension of our work on automatic annotation for verbalization is in creating a system for using many annotation modules for annotating different types of robot's data. The system needs to incorporate human-in-the-loop techniques to avoid errors due to limitations of the annotation modules. In the case of floor-identification module, extensions are required to the dataset to accommodate variability due to people and lighting conditions.

Extensions to the work on explaining scene recognition network can be testing the algorithm in other scene recognition dataset and improving the explainable features generation process. The feature generation process can be improved by using a probabilistic features vector for class-wise features, i.e., instead of taking a union of all the features in the training set we can create a distribution over all the possible features. Another easy and intuitive way to improve the generation of explainable features can be using the absence of a feature as a feature. For example, say class A's $E_{c,r}$ is presence of feature 1 and 2, class B's $E_{c,r}$ is presence of feature 1 and absence of 2 and class B's $E_{c,r}$ is presence of feature 2 and absence of 1, differentiating between these classes requires us to use the absence of a feature as a feature.

In our work on explaining robot's actions, we have used the class-wise explainable features as only as a means for an explanation, but as we described earlier, it can act as a dictionary defining the class itself. Hence, another interesting line of work is combining both the confidence of the classifier and the similarity vector between E and $E_{s,c,r}$ to predict the classification. We can also use the discrepancy between the classifier and the similarity vector between E and $E_{s,c,r}$ for an image to raise doubts or uncertainty about the robot's location.

Extending verbalization to describe explainable feature can be another future line of work. We can generate explanations from a given explainable features more flexibly through verbalization. By defining new verbalization space dimension, we can personalize explanations as per the user's preference as we did while verbalizing the path of CoBot.

The metrics we have developed for the deep visualization technique tell us how good the visualization technique is for the classifier. We can also use the metric to ask the reverse question, i.e., use the metric to train the classifier for a particular visualization technique.

Bibliography

[1] Simon Alexanderson, David House, and Jonas Beskow. Automatic annotation of gestural units in spontaneous face-to-face interaction. In *Proceedings of the Workshop on Multimodal Analyses enabling Artificial Agents in Human-Machine Interaction*, pages 15–19. ACM, 2016. 1.2, 3.1.1

[2] Nicholas D Allen, John R Templon, Patrick Summerhays McNally, Larry Birnbaum, and Kristian J Hammond. Statsmonkey: A data-driven sports narrative writer. In *AAAI Fall Symposium: Computational Models of Narrative*, 2010. 2.1

[3] David Baehrens, Timon Schroeter, Stefan Harmeling, Motoaki Kawanabe, Katja Hansen, and Klaus-Robert MÃžller. How to explain individual classification decisions. *Journal of Machine Learning Research*, 11(Jun):1803–1831, 2010. 1.3, 4.1

[4] Robert Belvin, Ron Burns, and Cheryl Hein. Development of the hrl route navigation dialogue system. In *Proceedings of the first international conference on Human language technology research*, pages 1–5. Association for Computational Linguistics, 2001. 2.1

[5] Joydeep Biswas and Manuela Veloso. Localization and Navigation of the CoBots over Long-term Deployments. *International Journal of Robotics Research*, 32(14):1679–1694, 2013. 2.4

[6] Joydeep Biswas and Manuela Veloso. Episodic Non-Markov Localization: Reasoning about Short-Term and Long-Term Features. In *Proceedings of the International Conference on Robotics and Automation*, 2014. 2.4

[7] Avrim L Blum and Pat Langley. Selection of relevant features and examples in machine learning. *Artificial intelligence*, 97(1):245–271, 1997. 4.1

[8] Zvi Boger and Hugo Guterman. Knowledge extraction from artificial neural network models. In *Systems, Man, and Cybernetics, 1997. Computational Cybernetics and Simulation., 1997 IEEE International Conference on*, volume 4, pages 3030–3035. IEEE, 1997. 4.1

[9] Dan Bohus, Chit W Saw, and Eric Horvitz. Directions robot: In-the-wild experiences and lessons learned. In *Proceedings of the International Conference on Autonomous Agents and Multi-Agent Systems*, pages 637–644, 2014. 2, 2.1, 2.3

[10] Yuri Y Boykov and M-P Jolly. Interactive graph cuts for optimal boundary & region segmentation of objects in nd images. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, volume 1, pages 105–112. IEEE, 2001. 3.2.3

[11] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah. Signature verification using

a siamese time delay neural network. In *Proceedings of the 6th International Conference on Neural Information Processing Systems*, pages 737–744. Morgan Kaufmann Publishers Inc., 1993. 3.1, 4.2

[12] Adrian Bussone, Simone Stumpf, and Dympna O'Sullivan. The role of explanations on trust and reliance in clinical decision support systems. *IEEE International Conference on Healthcare Informatics*, 2015. 2.1

[13] Anind K Dey. Explanations in context-aware systems. In *ExaCt*, pages 84–93, 2009. 2, 2.1, 2.3

[14] Albert Diosi, Geoffrey Taylor, and Lindsay Kleeman. Interactive slam using laser and advanced sonar. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 1103–1108. IEEE, 2005. 1.2, 3.1.1

[15] Nils Goerke and Sven Braun. Building semantic annotated maps by mobile robots. In *Proceedings of the Conference Towards Autonomous Robotic Systems*, pages 149–156, 2009. 1.2, 3.1.1

[16] Dennis Mølholm Hansen, Bjarne K Mortensen, PT Duizer, Jens R Andersen, and Thomas B Moeslund. Automatic annotation of humans in surveillance video. In *Computer and Robot Vision, 2007. CRV'07. Fourth Canadian Conference on*, pages 473–480. IEEE, 2007. 1.2, 3.1.1

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 4.1

[18] Lisa Anne Hendricks, Zeynep Akata, Marcus Rohrbach, Jeff Donahue, Bernt Schiele, and Trevor Darrell. Generating visual explanations. In *European Conference on Computer Vision*, pages 3–19. Springer, 2016. 1.3, 4.1

[19] Tsotsos J.K., Culhane S.M., Wai W.Y.K., Lai Y., N. Davis, and Nuflo F. Modeling visual attention via selective tuning. *Artificial intelligence*, 78(1-2):507–545, 1995. 3.2.2

[20] Simonyan K., Vedaldi A., and Zisserman A. Deep inside convolutional networks: Visualising image classification models and saliency maps. In *ICLR Workshop*, 2014. 1.4, 3.2, 3.2.1, 3.2.2, 4.2

[21] Rachel Kirby, Frank Broz, Jodi Forlizzi, Marek Piotr Michalowski, Anne Mundell, Stephanie Rosenthal, Brennan Peter Sellner, Reid Simmons , Kevin Snipes, Alan Schultz, and Jue Wang. Designing robots for long-term social interaction. In *Proceedings of the International Conference on Intelligent Robots and Systems*, pages 2199 – 2204, 2005. 2.1

[22] G. Koch, R. Zemel, and R. Salakhutdinov. Siamese neural networks for one-shot image recognition. In *ICML Deep Learning Workshop*, 2015. 3.1, 4.2

[23] A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. 3.1, 3.1.2, 3.2, 4.2

[24] Todd Kulesza, Simone Stumpf, Margaret Burnett, and Irwin Kwan. Tell me more?: the effects of mental model soundness on personalizing an intelligent agent. In *Proceedings*

of the SIGCHI Conference on Human Factors in Computing Systems, pages 1–10. ACM, 2012. 2.1

[25] Todd Kulesza, Simone Stumpf, Margaret Burnett, Songping Yang, Irwin Kwan, and Weng-Keen Wong. Too much, too little, or just right? ways explanations impact end users' mental models. In *Visual Languages and Human-Centric Computing (VL/HCC), 2013 IEEE Symposium on*, pages 3–10. IEEE, 2013. 2.1

[26] Brian Y Lim and Anind K Dey. Toolkit to support intelligibility in context-aware applications. In *Proceedings of the 12th ACM international conference on Ubiquitous computing*, pages 13–22. ACM, 2010. 2.1

[27] Brian Y Lim, Anind K Dey, and Daniel Avrahami. Why and why not explanations improve the intelligibility of context-aware intelligent systems. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2119–2128. ACM, 2009. 2.1

[28] Meghann Lomas, Robert Chevalier, Ernest Vincent Cross II, Robert Christopher Garrett, John Hoare, and Michael Kopack. Explaining robot actions. In *Proceedings of the seventh annual ACM/IEEE international conference on Human-Robot Interaction*, pages 187–188. ACM, 2012. 4.1

[29] Linus J Luotsinen, Hans Fernlund, and Ladislau Bölöni. Automatic annotation of team actions in observations of embodied agents. In *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*, page 9. ACM, 2007. 2.1

[30] Alastair MacFadden, Lorin Elias, and Deborah Saucier. Males and females scan maps similarly, but give directions differently. *Brain and Cognition*, 53(2):297–300, 2003. 2.1

[31] George A Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine J Miller. Introduction to wordnet: An on-line lexical database. *International journal of lexicography*, 3(4):235–244, 1990. 4.4

[32] Oscar Martinez Mozos, Hitoshi Mizutani, Ryo Kurazume, and Tsutomu Hasegawa. Categorization of indoor places using the kinect sensor. *Sensors*, 12(5):6695–6711, 2012. 1.2, 3.1.1

[33] Carlos Nieto-Granda, John G Rogers, Alexander JB Trevor, and Henrik I Christensen. Semantic map partitioning in indoor environments using regional analysis. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 1451–1456. IEEE, 2010. 1.2, 3.1.1

[34] Vittorio Perera and Manuela Veloso. Handling Complex Commands for Service Robot Task Requests. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 2015. 2.4

[35] Vittorio Perera, Robin Soetens, Thomas Kollar, Mehdi Samadi, Yichao Sun, Daniele Nardi, René van de Molengraft, and Manuela Veloso. Learning task knowledge from dialog and web access. *Robotics*, 4(2):223–252, 2015. 2.4

[36] Andrzej Pronobis and Patric Jensfelt. Large-scale semantic mapping and reasoning with heterogeneous modalities. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 3515–3522. IEEE, 2012. 1.2, 3.1.1

[37] Selvaraju R., Das A., R. Vedantam, M. Cogswell, D. Parikh, and D. Batra. Grad-cam: Why did you say that? visual explanations from deep networks via gradient-based localization. *CoRR*, abs/1610.02391, 2016. 1.4, 3.2, 3.2.1

[38] Joseph Redmon and Ali Farhadi. Yolo9000: Better, faster, stronger. 2016. 4.4

[39] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Why should i trust you?: Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1135–1144. ACM, 2016. 1.3, 4.1

[40] Stephanie Rosenthal, Joydeep Biswas, and Manuela Veloso. An Effective Personal Mobile Robot Agent Through Symbiotic Human-Robot Interaction. In *Proceedings of AAMAS'10, the Ninth International Joint Conference on Autonomous Agents and Multi-Agent Systems*, Toronto, Canada, May 2010. 2.2.1, 2.4

[41] W. Samek, A. Binder, G. Montavon, S. Lapuschkin, and K. Müller. Evaluating the visualization of what a deep neural network has learned. *IEEE Transactions on Neural Networks and Learning Systems*, 2016. 3.2, 3.2.1

[42] Ivan Sanchez, Tim Rocktaschel, Sebastian Riedel, and Sameer Singh. Towards extracting faithful and descriptive representations of latent variable models. *AAAI Spring Syposium on Knowledge Representation and Reasoning (KRR): Integrating Symbolic and Neural Approaches*, 2015. 1.3, 4.1

[43] Y. Sun, Y. Chen, X. Wang, and X. Tang. Deep learning face representation by joint identification-verification. In *Advances in neural information processing systems*, pages 1988–1996, 2014. 3.1.2, 4.2

[44] Jesse Thomason, Shiqi Zhang, Raymond Mooney, and Peter Stone. Learning to interpret natural language commands through human-robot dialog. In *Proceedings of the 2015 International Joint Conference on Artificial Intelligence (IJCAI)*, Buenos Aires, Argentina, July 2015. 2, 2.1, 2.3

[45] Manuela Veloso, Nicholas Armstrong-Crews, Sonia Chernova, Elisabeth Crawford, Colin McMillen, Maayan Roth, Douglas Vail, and Stefan Zickler. A team of humanoid game commentators. *International Journal of Humanoid Robotics*, 5(03):457–480, 2008. 2.1, 2.4.2

[46] Manuela Veloso, Joydeep Biswas, Brian Coltin, and Stephanie Rosenthal. CoBots: Robust Symbiotic Autonomous Mobile Service Robots. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 2015. 1, 1.5, 2.4, 3, 3.1.2, 4.2

[47] M. Veloso and J. Biswas and C. Brian and S. Rosenthal and T. Kollar and C. Mericli and M. Samadi and S. Brandao and R. Ventura. CoBots: Collaborative Robots Servicing Multi-Floor Buildings. In *Proceedings of IROS'12, the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Vilamoura, Portugal, October 2012. Video and Extended Abstract. 3.1.2, 4.2

[48] Dirk Voelz, Elisabeth André, Gerd Herzog, and Thomas Rist. Rocco: A robocup soccer commentator system. In *RoboCup-98*, pages 50–60. Springer, 1999. 2.1

[49] Ameni Yengui and NEJI Mahmoud. Semantic annotation formalism of video-conferencing documents. *Cognition and Exploratory Learning in Digital Age*, 2009. 2.1

[50] J. Yosinski, J. Clune, T. Fuchs, and H. Lipson. Understanding neural networks through deep visualization. In *In ICML Workshop on Deep Learning*. Citeseer, 2015. 3.2.1

[51] M. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014. 3.2, 3.2.1, 3.2.2

[52] J. Zhang, Z. Lin, J. Brandt, X. Shen, and S. Sclaroff. Top-down neural attention by excitation backprop. In *European Conference on Computer Vision*, pages 543–559. Springer, 2016. 1.4, 3.2, 3.2.1, 3.2.2, 3.2.5, 3.2.6

[53] Z. Zheng, L. Zheng, and Y. Yang. A discriminatively learned CNN embedding for person re-identification. *CoRR*, abs/1611.05666, 2016. 3.1.2, 4.2

[54] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva. Learning deep features for scene recognition using places database. In *Advances in neural information processing systems*, pages 487–495, 2014. 3.1.2, 4.2

[55] B. Zhou, A. Khosla, A. Lapedriza, A. Torralba, and A. Oliva. Places: An image database for deep scene understanding. *CoRR*, abs/1610.02055, 2016. 3.2, 3.2.4, 3.2.4

[56] L. Zintgraf, T. Cohen, and M. Welling. A new method to visualize deep neural networks. *CoRR*, abs/1603.02518, 2016. 1.4, 3.2, 3.2.1