Learning spatial referential words with mobile robots

Simon Dobnik
Computational Linguistics
Clarendon Institute, Oxford University
Walton Street, Oxford, OX1 2HG, UK
simon.dobnik@clg.ox.ac.uk

Abstract

Natural language contains a number of word categories that are referential in nature. Their full semantics can only be evaluated by examining the context in which the words are used. An important and challenging group of words are those that describe space. A system has been developed where the meanings of spatial words such as *near*, *left* and *behind* are learnt by a mobile robot from its experience of environment: by abstracting over the properties of its knowledge of environment and the descriptions that a human commentator used to describe it. Learning is performed offline using statistical and symbolic learning techniques. With the knowledge that it acquired the robot is able to generate new descriptions of new environments. Users can query the robot through a simple dialogue interface using spoken natural language.

1 Introduction

A semantic theory of spatial expressions is of interest of any computational treatment of the meaning of natural language. This is especially the case in automatic generation of descriptions for virtual environments such as computer games, car navigation systems that alert and inform the driver of the route to be taken, and for assistive aids for visually impaired. The area is tightly connected to mobile robotics that deals with localisation and mapping (SLAM) (Dissanayake et al., 2001).

The meaning of spatial words can only be evaluated by establishing a reference to the properties of the environment in which the word is used. For example, in order to evaluate how *near* is *near* or how *fast* is *fast* in a given context, we need to evaluate properties such as the size of the scene, the perspective at which the scene is viewed, typical behaviour and properties of objects in the scene and the configuration and position of other objects or 'distractors' (Herskovits, 1986).

There are two main difficulties in interfacing the information about the physical world and natural language descriptions. Physical world can be evaluated by measures that are *continuous* in nature. The location of two points can be determined by introducing a coordinate system with a fixed origin and by determining the points' x and y coordinates in the system using the scale of real numbers. On the contrary, natural language descriptions use discrete reference to refer to events or objects. They partition space into regions such as *near*, *back* and *left*, and degrees of motion to *slowly*, *moderately* and *fast*. While non-linguistic reference can be made with a high degree of accuracy, the reference of spatial descriptions is often ambiguous and vague.

As elsewhere in natural language and knowledge representation, the approaches to the semantics of the spatial expressions divide between symbolic and non-symbolic. Symbolic approaches (Di Tomaso and Lombardo, 1998) attempt to design rules that encode domain specific knowledge manually. Non-symbolic techniques, such as (Gapp, 1994) and

(Regier and Carlson, 2001), identify abstract parameters that model some properties of physical environment (supported by psychological evidence) and subsequently train their values using machine learning techniques to match the descriptions.

2 Our approach

Our approach follows the second line of research. However, instead of introducing complex (already abstracted) parameters and training their values, we train our classifiers on simple primitives that are available to us through the sensory data of a mobile robot. If a robot can be 'taught' to understand and use such expressions in a manner that would seem natural to a human observer, then we can be reasonably sure that we have captured at least something important about their semantics. Of course, as we already know, this may not be sufficient to be able to model the correct usage of natural language spatial expressions, but it gives us a platform on which we can superimpose successively more abstract properties in order to approach this more nearly.

Our learning task closely resembles the task of *grounding* of word meanings described in (Roy, 2002). Here lexical semantics, syntax and word categories are learnt from a bag of words that represent human commentary of computer generated scenes that include rectangles of various sizes, colours and shapes. The features of these scenes and the words describing them are paired using probabilistic machine learning techniques. Our approach transposes the learning task to the area of mobile robotics where the 'training' corpus consists of observations generated by the robot using its sensors and human commentary. The commentary consists of short phrases describing what the robot is doing or what are the relations between the objects in its environment.

Our system is able to use this domain specific knowledge to generate new descriptions in new environments. A robot that can both be instructed to move via natural language, and which can report what is doing when moving autonomously or under control of another in a burning building or hazardous environment is of great practical utility. Furthermore, linguistic descriptions could be used alongside sensory data in SLAM to disambiguate the location of the robot. For example, (Stachniss et al., 2005) describe a method where parts of the map are automatically labelled as 'office', 'corridor' and 'kitchen'. The reverse should also be possible: by being told that its location is 'in the corridor by the kitchen' or 'to the left of the table' the robot could correct its position on the map where sensory data were insufficient. Our language system is already integrated with the system that is used to drive, navigate and localise a mobile robot MOOS (Newman, 2001). It is one of our long term goals to be able to exchange sensory and other already abstracted information across both systems in a way that would improve the operation of both.

3 MOOS

MOOS (Newman, 2001) stands for Mission Oriented Operating Suite and its core represents a set of libraries and executables that run a mobile robot. Its main advantages are that it is platform independent (Unix, Windows) and that it is conceived as a modular system with a star-like topology (Figure 1). Each application within a MOOS community has a connection to the MOOS database (`MOOSDB`). This acts as a blackboard: applications do not communicate directly with each other but can only publish and retrieve information to and from `MOOSDB`.

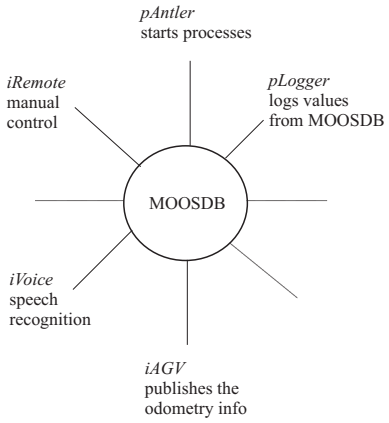

Figure 1: *The simple topology that was used during collection of motion data. iRemote was used to move the robot around the room. A human describer described the scene to speech recognition software that passed the sentences to iVoice, and iAgv provided odometry information such as robot's heading and speed. All data was passed to MOOSDB where it was retrieved from and logged by pLogger.*

4 Data collection

Spatial descriptions that describe a mobile robot belong to two different contexts. In the first context, the robot is moving in an enclosed space and the descriptions refer to its motion (*You're going forward slowly. Now you're turning right.*). In the second context, the robot is static in an enclosed space which contains real-size objects such as desks, chairs and walls. Here we are interested in prepositional phrases that are used to describe relationships between objects (*The chair is to the left of you. The table is further away than the chair*). The perspective can be varied by changing the location of the robot. However, descriptions are always made from the robot's point of view.

Four different describers, two native and two non-native speakers of English, were asked to describe each context. Each describer first trained the speech recognition software to get accustomed to their voice. Before beginning describing, in the first context, the operator of the robot demonstrated to the describers various types of motion that the robot is capable of performing (without using any linguistic descriptions). In the second context, the operator explained to the describers the names of the objects. The describers were then asked to describe the scenes with any descriptions that they considered appropriate. Figure 2 shows the robot and its environment in the second context.

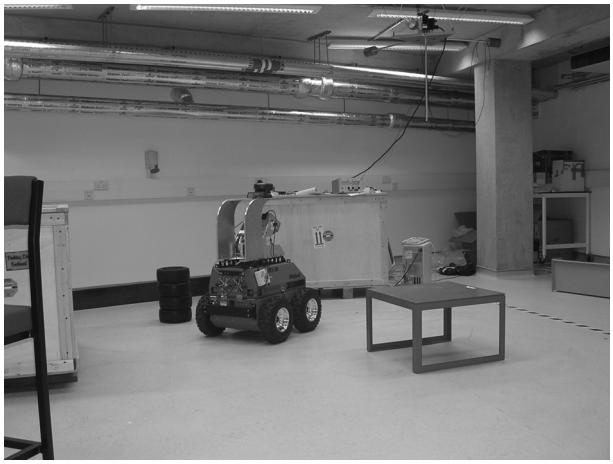

Figure 2: *The robot in the second context: the tyres are to the left of the robot, the table is to its right and the chest is behind it.*

The first context included dynamic scenes where the robot operator guided the robot in an enclosed space attempting to perform all kinds of motion that the robot was capable of. As a guideline the operator had a list of possible motions (but which was not disclosed to the describers) to make sure that during each data collection session all kinds of motion were exhausted. The motion types included forward and backward motion with various velocities and turning left or right under acute or obtuse angles. The robot can be controlled using a keyboard by incrementing or decrementing the desired thrust and heading values which are expressed as percentages of their maximum values. The changes in values are made in steps of 2.

One difficulty that became evident during the data collection was a possible delay between the time of the motion and the time when the description reached `MOOSDB`. There are three reasons for this. The data was collected in an enclosed space in the lab and thus the length of motion was quite limited. Secondly, it was observed that it takes a considerable time for the describer to decide upon a suitable description once the motion has started. Finally, a time lag was also introduced by the speech recogniser during which it recognised the string and published it to the system. It was also noticed that some describers experienced difficulties describing from the robots perspective, especially for backward left and right. In same cases they switched to their own perspective. This means that a considerable error was introduced in the dataset and methods had to be devised to evaluate and minimise this error in the learning procedure.

In the second context, real-size objects (chest, box, table, pillar, stack of tyres, chair, desk, shelves, etc.) were introduced to the environment and the task of the describer was to describe the location of these objects which also included the robot. The operator changed the location of the robot from time to time to vary the perspective. However, while the describer was describing, the robot was stationary and hence much of the errors from the previous context could be avoided. Nonetheless, describers may have introduced some errors by giving wrong descriptions.

Before the data collection a SLAM map of the environment was built using the MOOS SLAM ap-

plication pSMSLAM (Newman, 2001). Initially, the robot has no information about the environment or its location. pSMSLAM (in SLAM mode) builds its map incrementally through the observations using a scanning laser while the robot explores the environment.[1] The SLAM algorithm uses online probabilistic methods ($Pr(location|observation)$) to fuse consecutive scans into a coherent map of the environment relative to the robot's origin.

The map contains no abstraction of objects or environment. It is a bag of $\langle x, y\rangle$ coordinates[2] relative to the robot's origin. For this reason, objects had to be grounded manually. Each object was defined as a vector $\langle x, y, z,$ object-name$\rangle$, where x, y and z are the coordinates of the central axis of each object relative to the robot's origin and *object-name* is the name of the object. Since the central axis was taken, the approach ignores the shape of the object and hence will have difficulties dealing with objects that are long and thin rather than square. Consider a situation where such an object (e.g. *the barrier* in our case) is parallel with the robot but where the robot aligns with it only alongside one half of the object's length. A describer may describe it to be *to the right of the robot*, yet the object's y coordinate will be atypically large (and thus perhaps better corresponding to the description *in front of*) in comparison to objects whose sides are of equal length. We plan to include information about object shape in our future work.

When the robot was placed in a scene and the system was ready for the describers to start, PSMSLAM (now in the localisation mode) was used to localise the robot on the map that was build in the previous step ($Pr(location|observation, map)$). Thus, the system had the information about the location of the objects and the current position of the robot. An application iCommentary was written which outputs the Euclidean relationships between the vehicle and the labelled objects. The data that was logged to pLogger represented the coordinates of objects in the current reference frame of the robot and the descriptions created by the describers made from the robot's perspective.

[1] While it is possible to include navigation, for the purposes of data collection, the robot was navigated manually.

[2] The z coordinate can also be measured if 3D rather than 2D laser is used.

5 Creating instances for machine learning

For machine learning we chose Weka (Witten and Frank, 2000) a deservedly popular implementation of many common machine learning algorithms and associated tools. One considerable advantage is that it uses a unified file structure for input and output data for all learning algorithms that it implements. Thus, different algorithms can be tested on the same dataset with no extra processing, and their performance can be compared. Weka algorithms can handle datasets that consist of independent, unordered instances. Instances are vectors of attribute values, the properties that we want to include in the learning. To learn one attribute (which is also known as the target concept) means to find relationships between the values of other attributes that predict the values of the target concept.

The choice and representation of attributes has a highly significant impact on learning. Our method of machine learning is a supervised method, which means that the input data represented as attributes is already structured and abstracted. As stated previously, pLogger logs each attribute independently of other attributes. To create a coherent instance, a set of log entries, one per each attribute, should be chosen and their values extracted. These values should subsequently be rewritten as vectors $\langle$Attr1, Attr2, Attr3...$\rangle$ in a structured text file that the Weka can open. All numeric data should be normalised so that the theories that will be learnt will be applicable to new environments and situations. Linguistic data, which is represented as random sentences, should be tokenized to words, and their categories (or attribute membership) should be determined. Instances were created automatically, while the tagging procedure was performed manually.

A straightforward algorithm to create instances from log entries is as follows: (i) find a VOICE_INPUT entry, (ii) find ODOMETRY entry (or COMMENTARY_RELATIONS entry, depending on what kind of instances we are extracting) such that its log time is lesser than the log time of the VOICE_INPUT entry, and such that the log time of the next ODOMETRY (or COMMENTARY_RELATIONS) entry is greater than the log time of that VOICE_INPUT entry. This assures that only the immediately preceding odometry

or commentary relations entry is extracted. This is necessary because such entries are numerous as they are published at less than a second intervals.

Unfortunately, there is no assurance that the immediately preceding entry will be the best one. This is especially true for motion instances from the first context where delays are quite frequent. In order to try to minimise the error, an alignment procedure was devised. The point where the algorithm can check whether the data is correctly aligned is when a describer said 'stopped'. The data is segmented into segments bound by these descriptions. If the immediately preceding odometry information reports zero speed, the alignment is correct and all intermediate descriptions can be handled as under first algorithm, if not, the segment must be realigned. Currently, a method where all description times in a given segment are linearly shifted by the delay of the description 'stopped' has been implemented. We proceed as follows: (i) find odometry time in the same segment where speed is zero, (ii) determine *Delay* between the 'stopped' description and that odometry time, (iii) subtract *Delay* from all description times in that segment, (iv) create instances as under first algorithm. The disadvantage of this approach is that it assumes that the delay is linear (which may be not) and that for descriptions at the beginning of the segment we may not find any odometry information, because this has been cut off by the segmentation. On the other hand, the idea of segmentation is attractive, since it allows us to separate segments that contain clean data and those that do not.

Matching entries from log files is not sufficient to create instances. Information from these entries must be further processed. The `ODOMETRY` variable published by the robot gives us the x and y coordinates and the heading of the robot relative to the origin where the vehicle started, its current velocity in the x and y directions (in m/s) and its angular velocity (rad/s). The two attributes that we require are the robot's current heading and speed $\langle$delta_heading, speed$\rangle$. The angular velocity is a measure of the robot's current heading. Its speed can be calculated by Pythagoras from the x and y velocities. The value of the speed is normalised to the value of a pre-specified maximum speed that the vehicle can take. For the second context, the value of the MOOS entry `COMMENTARY_RELATIONS` will contain the coordinates of all objects relative to the robot at any given time $\langle$object, x, $y\rangle$. The coordinates x and y of each entry are normalised to the maximum x and y value found in the SLAM map.

To process linguistic information automatic POS tagger could be used to tag words, yet their number was relatively small. The total number of words that all four describers used in the context of motion expressions amounted to 267. Most of them, however, were not spatial expressions but words from non-related statements that may have been logged while testing the speech recogniser and words that have been recognised incorrectly. By examining input sentences, it became evident that the categories that are required to describe a motion scene are *Verb* and various kinds of adverbs: those that describe *Direction* of movement (forward, backward), *Heading* (left, right), and *Manner* (slowly, fast). For the second context containing descriptions of static scenes, the categories that are required are prepositional *Relation* and *Object*. During the tagging procedure, a set of log files was opened, all words were extracted from them and a human tagger assigned one of the above categories to each word. Words that did not belong to any of these categories were ignored. The spelling of words that were incorrectly recognised could also be adjusted. The tagged words were saved as entries of a very simple unification grammar. Each entry was represented as a predicate `lexical_entry/1` whose argument is a list of `feature=value` pairs. Three features that are required for our grammar are `form` which specifies the word form of the entry, `sem_type` that defines its category, and `arg_list` which is a list of arguments that this entry takes and whose names correspond to the semantic types of other words. When extracting arguments from a sentence, the algorithm first looks for a verb and then tries to match words whose categories correspond to its arguments. If no word of the required category is found, the entry is rewritten as *none*. Thus, when parsing a sentence, the algorithm returns a vector of words $\langle$verb, direction, heading, manner$\rangle$.[3]

When both linguistic and non-linguistic data was combined, two sets of instances, one set per context, were created. The instances for the first con-

[3] In this step, spelling correction is also applied.

text are represented by vectors of attribute values ⟨delta_heading, speed, verb, direction, heading, manner⟩ and those for the second context are represented by vectors of values ⟨lo_x, lo_y, refo_x, refo_y, relation⟩. Note that the latter vector does not contain object names that were used in the description but their coordinates. LO refers to the object that is located by its relation to the reference object REFO ('the table (LO) is in front of you (REFO)'.

6 The learning algorithms

Two popular algorithms from the Weka toolkit were chosen to train on data: J48 and NaiveBayes. Each uses a different method of forming hypotheses about the data.

J48 is Weka's implementation of the C.45 decision tree learner. A new instance can be classified by following branches of a tree until reaching leaf nodes which represent the target concept. (Witten and Frank, 2000) describe the method of building a decision tree as 'divide and conquer'. First, an attribute is selected as the root node for the tree and then branches are created for each of its possible values. The process is repeated recursively for each branch but using only a subset of instances that fall under that mother branch. If all instances at a node have the same classification, the development of that part of the tree can be stopped. The most important step is to decide which attribute to split on first. Small trees are preferred and thus an attribute that would result in reaching the leaf node as soon as possible is the winning candidate. The measure that J48 uses is information gain (which is based on information and entropy). The attribute that gains the most information is the preferred candidate to create a branch for. This is the attribute for which creating branches for its values will introduce the smallest variation of the target attribute values alongside each branch.

NaiveBayes learns probabilistic knowledge. For example, to learn the meaning of a word it means to determine probability distributions on the right hand side of the Equation 1. E are the values of other attributes included in the learning.

$$\Pr(\text{desc}|E_{1\ldots n}) = \frac{\Pr(E_1|\text{desc})\ldots\Pr(E_n|\text{desc})\Pr(\text{desc})}{\Pr(E)} \quad (1)$$

It follows from Equation 1 that all attributes are making a contribution to the decision about the class. Importantly, their probabilities will be different and some will have a greater weight than the others. Although in real datasets attributes are not equally important and independent of one another, NaiveBayes nonetheless works surprisingly well in practice.

7 Results

Both algorithms were run on various subsets of our data and their results are presented in the following sub-sections. All classifiers were evaluated by 10-fold cross-validation.

7.1 Context I: motion scenes

Let us consider the results from two subsets of data. Subset I (192 instances) contains data that was manually estimated during collection to be the most error-free and no alignment of instances was performed. Subset II (338 instances) contains data that may contain errors and on which alignment was performed. The attributes that were included in each training run were the two numerical attributes {delta_heading, speed} and one of the nominal attributes representing the categories of words {direction, heading, manner, verb}, thus for example ⟨delta_heading, speed, verb⟩. Including all nominal attributes in each run significantly increased the classifiers' accuracies (not shown here). This is expected, since nominal attributes are not independent and knowing the class of verb makes it much easier to predict the class of heading, for example. If the vehicle is 'turning' then the class of heading is most likely to be 'left' or 'right'. Tables 1 and 2 show that the accuracies of classifiers trained and tested on Subset II are lower than the accuracies of classifiers based on Subset I. A full comparison between the effects of alignment *vs* no-alignment is yet to be made. Note that in both tables the classifiers for *Manner* and *Verb* have the lowest accuracy. This confirms our intuition that these attributes are the most ambiguous and thus hardest to learn. Finally, Table 3 shows the values of nominal attributes in Subset I. Since each attribute has 4 or 5 values, the probability of randomly guessing a word for each class is 25% or 20% which serves as the base case for com-

parison.

Classifier	Direction	Heading	Manner	Verb
J48	74.0%	74.0%	79.2%	75.5%
NaiveBayes	67.7%	75.5%	78.1%	64.1%

Table 1: *Subset I: nominal instances*

Classifier	Direction	Heading	Manner	Verb
J48	73.4%	73.1%	65.4%	65.4%
NaiveBayes	70.1%	67.8%	57.4%	62.1%

Table 2: *Subset II: nominal instances*

Attribute	Values
Direction	backward, forward, none, spot, straight
Heading	anticlockwise, clockwise, left, none, right
Manner	fast, moderately, none, slowly
Verb	creeping, going, moving, turning, stopped

Table 3: *Nominal attribute values for Subset I. Other subsets may have different values depending on what words describers used.*

Learning continuous numeric attributes {delta_heading, speed} is not as straightforward as learning nominal ones since classifiers cannot deal with target concepts that are not nominal. Before learning, numeric attributes must be turned into nominal ones. This can be done by splitting the scale of all possible real numbers that define the numeric attribute into a predefined number of intervals or bins of fixed size.[4] Contrary to the naïve belief, increasing the number of bins does not improve the accuracy of the classifier but decreases it, since the greater the number of bins, the harder to predict the instance class as demonstrated in Table 4. On the other hand, to have a useful classifier, an increased number of bins is desirable. A choice on the optimal number of bins must thus be made. For our dataset, 5 appears to be a reasonable choice since this corresponds to the number of conceptual distinctions contained in our nominal classes, and the classifier accuracies are also comparable.[5]

[4]Weka already contains such filter (`weka.filters.unsupervised.attribute.Discretize`).

[5]The optimal number of bins could be determined formally by starting with a large number of bins and then computing mutual information between each neighbouring pair. Bins with high mutual information could be fused into one until an optimal number would be reached.

Bins	Delta heading	Speed
3	81.8%	80.2%
5	72.4%	69.8%
10	49.5%	64.0%
20	35.9%	52.6%
30	30.7%	52.1%
40	26.0%	49.5%

Table 4: *The accuracy of the J48 classifier trained on Subset I for numeric attributes binned to various number of bins.*

7.2 Context II: prepositional relations

251 instances from the second context were chosen where each instance was represented as a vector of values ⟨lo_x, lo_y, refo_x, refo_y, relation⟩. *Relation* was the target concept to be learnt and its possible values were {'in front of', 'behind', to the left of', 'to the right of'}. The performance of classifiers is comparable to the ones from the first context: J48 correctly classified 74.9% of instances and NaiveBayes 77.3% of instances.

8 Using the acquired knowledge

The aim of the learning experiment is to be able to show that the robot can use the knowledge that it learnt in practice. It is planned for our future work to have human evaluators that would judge the linguistic response of the robot and compare the figures with those given by the classifiers. To this end, two programmes were designed that can be integrated with the MOOS community which we called *pDescriber* and *pDialogue*.

pDescriber is a commentator: if the robot is moving it describes its actions, else if the robot is stationary, it provides comments about position of objects. In order to do so, *pDescriber* uses the classifiers for nominal attributes discussed in Sections 7.1 and 7.2. The system has a similar setup as when collecting the data, except that now a speech synthesiser rather than a recogniser is used which pronounces the sentences that are generated from the words predicted by the classifiers.

pDialogue answers user's questions about the position of objects, performs motion commands or just chats with users. It contains three modes of operation: the first one is a simple pattern matching dialogue interface that tries to match the words in the user's question with a predefined pattern of words

and then responds with the reply of the winning pattern. The second mode performs commands such as 'Go forward slowly' and 'Go forward right fast'. For this mode, the classifiers for *Delta heading* and *Speed* are used. Furthermore, their output values (which represent robot's states) must be turned to instructions that move the robot. These are *desired rudder* and *desired thrust* which are percentages of the maximum rudder and the maximum speed. In the third mode, the robot answers questions about the position of objects, for example 'Where is the chair?'. For this, the classifier from Section 7.2 is used. At the moment, the reference object is chosen manually. This is not sufficient, since human intuition shows that depending on the configuration of the scene, some objects are more likely to be REFOs than the others. *pDialogue* determines its mode by parsing a tokenized user input with the grammar discussed in Section 5 and by checking the semantic type of the verb.

Both *pDescriber* and *pDialogue* normalise all numeric values to current maximum values and thus can be used in environments different from the ones in which their classifiers were trained.

9 Conclusion and further work

In this paper we describe a complete cycle how descriptions are learnt from the properties of the environment internalised by the robot and later used by it to interact with users and its environment. There are many ways in which the system could be improved. Machine learning described in this paper is a type of supervised learning which means that many steps depended on human input and judgment. An important achievement would be to show how such human 'interference' can be minimised. Secondly, the learning is performed on very simple representations of robot's environment. We have seen that such representations are not sufficient to model the meanings of spatial expressions closely. The question of representations is also related to the modelling of perspective (of the robot and its interlocutors) which our system presently does not implement. A significant effort has been devoted to minimise and evaluate errors in the input data and further improvements are underway. Finally, our longterm research goal is to examine how linguistic and localisation systems can be closely integrated to exchange information that is potentially useful to both.

References

Vittorio Di Tomaso and Vincenzo Lombardo. 1998. A computational model for the interpretation of static locative expressions. In Patrick Olivier and Klaus-Peter Gapp, editors, *Representation and processing of spatial expressions*, chapter 4, pages 73–90. Lawrence Erlbaum Associates.

M. W. M. G Dissanayake, P. M. Newman, H. F. Durrant-Whyte, S. Clark, and M. Csorba. 2001. A solution to the simultaneous localization and map building (SLAM) problem. *IEEE Transactions on Robotic and Automation*, 17(3):229–241.

Klaus-Peter Gapp. 1994. A computational model of the basic meanings of graded composite spatial relations in 3-d space. In *Proceedings of the AGDM'94 (Advanced Geographic Data Modelling) Workshop*, pages 66–79, Delft, The Netherlands. Netherlands Geodetic Commission.

Annette Herskovits. 1986. *Language and spatial cognition: an inderdisciplinary study of the prepositions in English*. Studies in natural language processing. Cambridge: Cambridge University Press.

Paul Newman, 2001. *MOOS - Mission Orientated Operating Suite*. Oxford University Robotics Research Group, Department of Engineering Science, Oxford University, http://www.robots.ox.ac.uk/~pnewman/MOOS/.

Terry Regier and Laura A. Carlson. 2001. Grounding spatial language in perception: an emprical and computational investigation. *Journal of Experimental Psychology*, 130(2):273–298.

Deb K. Roy. 2002. Learning visually-grounded words and syntax for a scene description task. *Computer speech and language*, 16(3):353–385.

Cyrill Stachniss, Oscar Martínez-Mozos, Axel Rottmann, and Wolfram Burgard. 2005. Semantic labelling of places. In *Proceedings of the International Symposium of Robotics Research (ISRR)*, San Francisco, CA, USA.

Ian H. Witten and Eibe Frank. 2000. *Data Mining: Practical machine learning tools with Java implementations*. Morgan Kaufmann, San Francisco.