Understanding Natural Language Parking Instructions and Grounding for Self-driving Cars

Nana Otawara
Ochanomizu University
g1420511@is.ocha.ac.jp

Hiroshi Tsukahara
Denso IT Laboratory, Inc.
htsukahara@d-itlab.co.jp

Atsushi Keyaki
Denso IT Laboratory, Inc.
akeyaki@d-itlab.co.jp

Ichiro Kobayashi
Ochanomizu University
koba@is.ocha.ac.jp

Abstract

There has been a rapid development of practical applications of automatic driving, and interactive operation using natural language will soon be necessary to easily operate autonomous cars. In this study, we attempt to realize a correspondence relationship, (i.e., symbol grounding) between driving instructions expressed in natural language and objects in the real world recognized by the sensors in a car. We then propose a method to solve the grounding problem by means of belief propagation method and show an example of the solution with the observed real sensor data. In particular, we focus on the parking operation of a car in this study.

1 Introduction

There has been growing research on the practical use of autonomous cars, and some models have already been sold on the market. However, there would be many complex and irregular driving situations in real road scenes and expected to be cases that the system cannot recognize the situation correctly or react it appropriately. Safe and comfortable driving can be achieved by enabling verbal human-car communications through which passengers can temporarily provide driving instructions to cars, in facing difficult situations for the system to handle. Moreover, it is especially convenient for elderly individuals or individuals unskilled in driving. We believe that verbal communication between a car and a human is an important factor for the operability of a car. To achieve communication between humans and cars, cars must first correctly understand the information about events, objects, and places contained in human instructions. Then, cars perform the instructions using the given information. To understand the content of verbal instructions, it is important that cars have functions to analyze natural language and extract spatial semantic relations among events, objects, and places from the instructions such as "*park (event) in front of (place) the white car (object)*".

To perform the instructions, there are two technical challenges; 1. extracting spatial semantic relations and 2. correctly correlating the spatial information of the instructions. In general, driving is extremely complex because many factors must be considered, such as different traffic situations and control targets. Therefore, in this study, we focus on the case in which we control a car for parking. This alleviates technical difficulties but is still practically significant. We propose a method to extract spatial semantics from verbal instructions for parking and correlate the extracted spatial semantics with objects or places observed around the car. Figure 1 presents an overview of our proposed method. Our proposed method infers a parking space or objects, given a driver's instruction such as "*Park next to the red bus*". A driver's instruction is converted into generalized grounding graphs (G^3) [1] through CCG [2] and SDC [3] so that we can apply belief propagation [4], before

Machine Learning for Autonomous Driving Workshop at the 33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

the inference is conducted. First, an instruction is converted to SDC using [5]. Next, G^3 is generated based on the SDC and the environment information a car observed, and then the likelihood of the corresponding relation between the SDC and the environment information is calculated so as it gets maximum. Finally, most likely parking place is inferred.

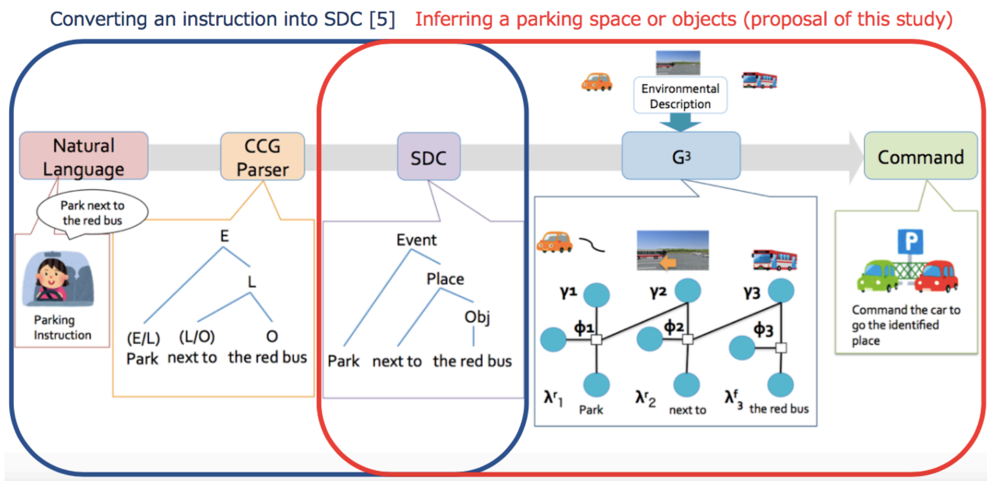

Figure 1: Overview of proposed: first, a driver's natural language instruction is converted into G^3 by the method of Inago et al. [5] (the part enclosed in blue), and then inference is conducted (the part enclosed in red).

2 Related Work

The spatial description clause (SDC) is proposed by Koller et al. [3]. Tellex et al. [1] developed generalized grounding graphs (G^3), which are probabilistic graphical models (factor graphs) that follow SDC structures, and constructed a system for correlating language information with observed data and understanding natural language commands containing spatial information. Howard et al. [6] proposed a distributed correspondence graph (DCG), which efficiently infers a set of planning constraints, such as goal regions and accessible or inaccessible regions in the environment, and solves the grounding problem with a higher accuracy rate than G^3. In addition, Paul et al.[7] proposed an adaptive distributed correspondence graph (ADCG), which extends the DCG so that it can reason about abstract or hierarchical concepts, such as rows or columns of objects. G^3, DCG, and ADCG are all graphical models and infer the most probable set of events or objects by maximizing their likelihood for given instructions and environments.

Some studies have tackled the grounding problem using deep learning methods with unsupervised learning. Rohrbach et al. [8] proposed an unsupervised learning method to correlate words and objects in an image by combining two procedures. The first is a grounding procedure that infers the region relevant to the given words, while the second is a reconstruction procedure that reproduces the original words from the inferred region.

Inago et al. [5] proposed a method for converting instructions for controlling a car into spatial description clauses through the analysis of the instructions with a combinatory categorial grammar (CCG)-based shift-reduce parser. To build a syntactic analyzer, they first defined a grammar, which regards spatial semantic categories as its syntactic rules, based on the CCG [2]. The grammar makes it possible to flexibly represent the relationship between different spatial semantic categories in the form of a parse tree. As a base syntactic analysis method, Inago et al. selected the shift-reduce parsing method [9] because it is simple to apply a grammar into the shift-reduce parser. A SDC is generated by converting the syntactic analysis result using heuristic conversion rules.

3 Inference of Grounding Graphs

The overview of the process is illustrated in Figure 1. We adopted the method of Inago et al [5] to generate SDCs from natural language instructions, and then employed a probabilistic graphical model called grounding graphs to take correspondence between the SDCs and the sensory information observed by a car. Grounding graphs are used to infer a parking space or objects that are referred to by SDC components and observed environmental information. Hereinafter, we focus on inference process.

For the calculation to get the G^3 with maximum likelihood, we consider factor graph models in which the joint probability distribution $p(\lambda, \gamma, \phi|\theta)$ is expressed as the product of the local weight factors, e.g., $\Psi_\alpha(\lambda_\alpha, \gamma_\alpha, \phi_\alpha|\theta)$. A graph is represented as a bipartite graph consisting of a set of variable nodes $x = (\lambda, \gamma, \phi)$ and a set of factor nodes, e.g., α:

$$p(\lambda, \gamma, \phi|\theta) = \frac{1}{Z(\theta)} \prod_{\alpha} \Psi_\alpha(\lambda_\alpha, \gamma_\alpha, \phi_\alpha|\theta_\alpha), \tag{1}$$

where, λ is a set of language variable nodes that correspond to the elemental units of SDC, γ is a set of environment variable nodes that express manipulation of the car or observed information of objects in the real world, ϕ is a set of grounding variable nodes that calculate the truth value for the Boolean state of correspondence, $x_\alpha = (\lambda_\alpha, \gamma_\alpha, \phi_\alpha)$ is a set of variable nodes that connect to factor node α. Grounding variable node ϕ_α connects to every factor node α. It returns True if the correspondence of a set of environment variable nodes γ_α and a set of language variable nodes λ_α is correct and False if incorrect. An index i is used for recognizing variable nodes, and x_i shows the variable node with the index i. The weight factor is an exponential type given by

$$\Psi_\alpha(x_\alpha|\theta_\alpha) = \exp\left(\sum_s \theta_{\alpha,s} t_s(x_\alpha)\right), \tag{2}$$

where $\theta_{\alpha,s}$ is a weight parameter for features $t_s(x_\alpha)$ with binary values, the suffix s is an index that represents features, $\theta_\alpha = \{\theta_{\alpha,s}\}$, $\theta = \{\theta_\alpha\}$, and $Z(\theta)$ is normalization factor,

$$Z(\theta) = \sum_{\{x_\alpha\}} \prod_{\alpha} \Psi_\alpha(\lambda_\alpha, \gamma_\alpha, \phi_\alpha|\theta_\alpha), \tag{3}$$

where $\sum_{\{x_\alpha\}}$ is the sum of all combinations of available values of variables in all nodes.

Inference using grounding graphs has the following challenges: i) First, we must generate grounding graphs that reflect the structures of SDCs. We can construct grounding graphs by recursively following the tree structure of the SDC; this procedure is the same as in a previous study [10]. In Section 3.1, we discuss the method of generating grounding graphs from given SDC structures. ii) In terms of weight factor (2), we must design appropriate features for the understanding of parking instructions. We define the following two feature types—language features and spatial features—and prepare quantitative and qualitative features as the feature characterizing spatial information. In Section 3.2, we describe the design of features in terms of weight factors. iii) To generate the grounding graph, we determine the values of the environment variable node γ so that its marginal probability, which fixes the values of language variable node λ and grounding variable node ϕ in a joint probability, is maximized. It should be noted that the range of the environment variable node γ is given by the environment expression obtained from the observed data. In this maximization, the cost of calculating the marginal probability of grounding graphs is problematic. In a prior study [3], the probability of the entire graph was not always maximized because the prior study addressed every factor independently and approximately calculated by multiplying them, then its marginal probability was maximized greedily. In this study, we calculate the marginal probability considering the connection of the entire graph by means of belief propagation [4] and maximize the marginal probability of the graph. In Section 3.3, we discuss calculating the marginal probability distribution of the grounding graphs by means of belief propagation. iv) We also determine parameter values θ for features. For this end, we build a dataset from observed data annotating the true parking places specified by the parking instructions and make use of this dataset to estimate the value of the parameters with log-likelihood maximization. In log-likelihood maximization, we adopt the iterative proportional fitting (IPF) algorithm [11]. However, due to the annotation cost, the amount of available data is limited, and in many cases, the empirical distribution estimated from the learning data becomes zero. For this reason, instability, such as in the parameters, occurs if IPF is adopted as is. Thus, we propose an update equation that contains a normalization parameter. We then enable the learning of parameters for features by avoiding instability with sparse data. In Section 3.4, we show an algorithm of learning weight parameters for the features.

3.1 Grounding graph generation

In this section, we explain how to generate a grounding graph from a SDC obtained by parsing parking instructions. Because a SDC has a nested structure, we can use this structure to generate grounding graphs. An example of a SDC and grounding graph is provided in Figure 2. The parking instruction, “Park in front of the space where the white car is parked” is converted to a SDC and expressed as “EVENT(r=Park l=PLACE(r=in front of l=PLACE(f=space s=STATE(r=is parked l=OBJ(f=the white car)))))”. In this SDC, EVENT contains the relation field “Park”, while the landmark field is included in PLACE. That is, EVENT SDC has PLACE SDC as a nested element. Grounding is expressed as the factor node that corresponds to "Park” and connects to the node γ_1. Grounding graphs are generated by a recursive procedure, as demonstrated in Algorithm 1. ψ represents the factor nodes in this algorithm.

```
EVENT(r=駐車して  Park
    l=PLACE(r=手前に   in front of
         l=PLACE(f=スペース  the space
              c=STATE(r=が停まっている  is parked
                   l=OBJ(f=白い車)))))  the white car
```

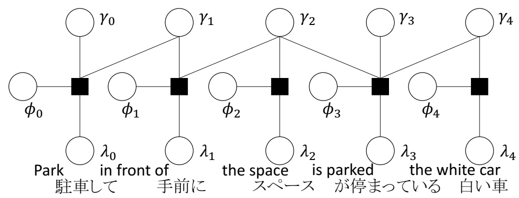

Figure 2: SDC and its generated grounding graph

Algorithm 1 Algorithm of generating G^3

1 generate the first γ, ψ, and ϕ nodes, as γ_0, ψ_0, and ϕ_0
2 check elements of SDC:figure, relation, landmark, view, condition
  2-1 if the element is first phrase,
    generate new λ, λ_0, and connect to ψ, ψ_0
  2-2 if the element is second phrase,
    generate new ψ, ϕ, and λ, and connect to γ
  2-3 if the element is EVENT or STATE SDC,
    generate new γ, ψ, and ϕ, and connect to γ, then return to 2 and check the next element
  2-4 if the element is other SDC,
    generate new γ, ψ, and ϕ, and connect to ψ, then return to 2, and check the next element

3.2 Features

We use a combination of two feature types about languages and spatial relations from the environment. The configuration of each feature is as follows.

Features for natural language expressions We prepared a dictionary of features for natural language expressions in advance. This dictionary was manually made from parking instructions used in the experiments. This dictionary is used for normalizing fluctuations of phrases that express the same event or object in different ways (i.e., synonyms).

Features for spatial relations As illustrated in Figure 3, we classified factor nodes into three types based on the number and SDC category of variable nodes γ (in this paragraph, we omit the factor suffix). The three types are as follows: a factor node with only one γ node whose SDC type is OBJECT, a factor node with two γ nodes whose SDC types are PLACE and OBJECT, and a factor node with two γ nodes whose SDC types are PLACE and EVENT. We consider the distance from the self (car), angle,

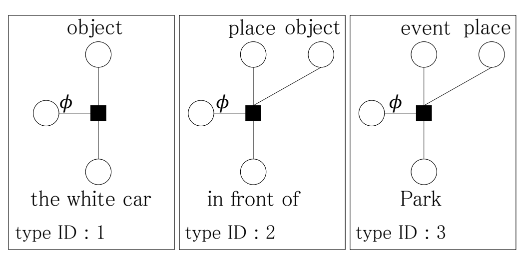

Figure 3: Factor graph types and their type ID.

and so on as spatial relation features. The spatial relation features are divided into two types: qualitative (e.g., order of distance) and quantitative (e.g., actual numerical distance).

3.3 Calculating marginal probability by message passing

The purpose of calculating grounding graphs is to estimate the values $\gamma_{\max}$ of environment variables that maximizes the marginal probability of environment variables γ to the given language variables λ and the grounding variables ϕ. In the prior study [3], the authors expressed a grounding graph as a product of logistic regression models with grounding variable ϕ_α under the condition that language variable nodes λ_α and environment variable nodes γ_α are independently given, but not the marginal probability of the entire graph. However, they considered the individual maximization of the conditional probability distribution that the logistic regression model of each factor express, that is,

$$\gamma_{\max} \approx \arg\max_{\gamma_\alpha} p_\alpha(\phi_\alpha = \text{True}|\lambda_\alpha, \gamma_\alpha) \tag{4}$$

where $p_\alpha(\phi_\alpha = \text{True}|\lambda_\alpha, \gamma_\alpha)$ is the probability that the logistic regression model outputs in factor node α, given only instructions for which all values of the grounding variables are True. However, the value of (4) does not agree with the conditional probability given λ and ϕ in (1) because (4) does not consider relations that some γ_α are shared between factor nodes. For this reason, the result differs depending on the order of maximizing factor nodes. In this study, we fixed λ and ϕ, and calculated the marginal probability of γ in (1) considering the relation of the entire graph. We grounded language information onto environmental information more effectively by calculating γ that maximizes the marginal probability. However, calculating marginal probability is generally difficult because it is difficult to calculate the sum of partition function (3). Therefore, we calculated the marginal probability that considers the relation of the entire graph by means of belief propagation [4, 12], although this is an approximate method. If the grounding graph has a tree structure, then calculating the exact marginal probability is possible [4]. The belief propagation algorithm is described in Appendix A.

3.4 Learning of weight parameters

The calculation of grounding graphs described above is on the assumption that parameters $\theta_{\alpha,s}$ are provided. Because we cannot decide these parameters ourselves, they must be decided inductively from the actual data. In this section, we describe the method for learning parameters. When the data $D_N = \{x^{(n)}\}_{n=1}^N$, true labels annotated in the actual data, are given as training data, we define a log-likelihood function for the data based on (1) and apply the IPF algorithm [11], which maximizes the defined log-likelihood function for each $\theta_{\alpha,s}$. The update equation of parameters $\theta_{\alpha,s}$ is given in the following algorithm based on IPF:

$$\theta_{\alpha,s}^{(q+1)} \leftarrow \theta_{\alpha,s}^{(q)} + \eta \log \frac{\hat{p}_\alpha(t_s(x_\alpha))}{p_\alpha(t_s(x_\alpha)|\theta_\alpha^{(t)})}, \tag{5}$$

where q represents the number of updates, $\hat{p}_\alpha(t_s(x_\alpha))$ is the probability of the empirical distribution that the feature takes the value of $t_s(x_\alpha) = 1$ calculated from the learning data $p_\alpha(t_s(x_\alpha)|\theta_\alpha^{(t)})$ is the same probability as the probability model in (1), and η is the learning rate. However, in updating with (5), if there is insufficient training data, in most cases the empirical distribution becomes 0 or very small, and the calculation of the log-likelihood of (5) occasionally falls into instability and overflows the memory. Therefore, we added the regularization term in (5) for stable calculation.

$$L(\theta) = \frac{1}{N}\sum_{n=1}^{N} \log p(x^{(n)}|\theta) - \frac{\mu}{2}\sum_{\alpha,s}\theta_{\alpha,s}^2, \tag{6}$$

Equation (6) is a convex function due to the convexity of the exponential family. Therefore, we can calculate the maximum value by optimization using a gradient method. The derivative of $\theta_{\alpha,s}$ is

$$\frac{\partial L(\theta)}{\partial \theta_{\alpha,s}} = \hat{E}[t_s(x_\alpha)] - \mu\theta_{\alpha,s} - E_\theta[t_s(x_\alpha)], \tag{7}$$

where

$$\hat{E}[t_s(x_\alpha)] = \frac{1}{N}\sum_{n=1}^{N} t_s(x_\alpha), \tag{8}$$

$$E_\theta[t_s(x_\alpha)] = \frac{1}{Z(\theta)}\frac{\partial Z(\theta)}{\partial \theta_{\alpha,s}} = \sum_{\{x_\beta\}} p(x_\beta|\theta) t_s(x_\alpha), \tag{9}$$

therefore, from the condition that $L(\theta)$ is maximized with $\theta_{\alpha,s}$, the following equation is given:

$$\hat{E}[t_s(x_\alpha)] = E_\theta[t_s(x_\alpha)] + \mu\theta_{\alpha,s}, \tag{10}$$

This conditional equation is satisfied when the expectation values from the empirical distribution about features $t_s(x_\alpha)$ match the expectation values of the model in the case of not considering the regularization term $\mu = 0$. Now, assuming that the condition of equilibrium (10) does not satisfy slightly, we consider satisfying the condition by adjusting $\theta_{\alpha,s}$ with the negligible quantity $\delta\theta_{\alpha,s}$ as follows:

$$\hat{E}[t_s(x_\alpha)] = E_{\theta^{(q)}+\delta\theta_{\alpha,s}}[t_s(x_\alpha)] + \mu(\theta_{\alpha,s}^{(q)} + \delta\theta_{\alpha,s}), \tag{11}$$

Because $t_s(x_\alpha)$ represents binary features, the left-hand side of (11) is given below:

$$\hat{E}[t_s(x_\alpha)] = \hat{p}(t_s(x_\alpha)), \tag{12}$$

$\hat{p}(t_s(x_\alpha)) = \hat{p}(t_s(x_\alpha) = 1)$ represents the appearance probability of features $t_s(x_\alpha)$ in the empirical distribution. The first term of the right-hand side of (11) can be replaced as in (13):

$$\begin{aligned} E_{\theta^{(q)}+\delta\theta_{\alpha,s}}[t_s(x_\alpha)] &= \sum_{\{x_\beta\}} t_s(x_\alpha) e^{\delta\theta_{\alpha,s} t_s(x_\alpha) - C\delta\theta_{\alpha,s}} p(x_\beta|\theta_\beta^{(q)}), \\ &= e^{(1-C)\delta\theta_{\alpha,s}} \sum_{\{x_\alpha : t_s(x_\alpha)=1\}} p(x_\alpha|\theta_\alpha^{(q)}), \end{aligned} \tag{13}$$

where, C is the constant not depending on $\delta\theta_{\alpha,s}$ and is defined by the first approximation of $\delta\theta_{\alpha,s}$:

$$e^{-C\delta\theta_{\alpha,s}} = \frac{Z(\theta^{(q)})}{Z(\theta^{(q)} + \delta\theta_{\alpha,s})}, \tag{14}$$

However, if the scale of the weight factor $\Psi_\alpha(x_\alpha|\theta_\alpha)$ regains $e^{C\prod_s \theta_{\alpha,s}}$ times, constant C can be 0. If we denote $e^{\delta\theta_{\alpha,s}} \simeq 1 + \delta\theta_{\alpha,s}$ by the first order approximation with $\delta\theta_{\alpha,s}$ and substitute this into (13), the update equation including the regularization term is given from (11)~~.~~:

$$\theta_{\alpha,s}^{(q+1)} \leftarrow \theta_{\alpha,s}^{(q)} + \eta\frac{\hat{p}_\alpha(t_s(x_\alpha)) - p_\alpha(t_s(x_\alpha)|\theta_\alpha^{(q)}) - \mu\theta_\alpha^{(q)}}{p_\alpha(t_s(x_\alpha)|\theta_\alpha^{(q)}) + \mu}, \tag{15}$$

where $p_\alpha(t_s(x_\alpha)|\theta_\alpha^{(q)}) = \sum_{\{x_\alpha : t_s(x_\alpha)=1\}} p(x_\alpha|\theta_\alpha^{(q)})$. It is noted that (15) can be expressed as (16) when the second term of its right-hand side is replaced with the first order approximation:

$$\theta_{\alpha,s}^{(q+1)} \leftarrow \theta_{\alpha,s}^{(q)} + \eta\log\frac{\hat{p}_\alpha(t_s(x_\alpha)) - \mu(\theta_{\alpha,s}^{(q)} - 1)}{p_\alpha(t_s(x_\alpha)|\theta_\alpha^{(q)}) + \mu}, \tag{16}$$

Equation (16) results in (5) when $\mu = 0$. If we apply (15) to all factor nodes α, $\theta_{\alpha,s}$ converges to the optimal solution $\theta_{\alpha,s}^*$ that maximizes the log-likelihood function. In this case, we use the marginal probability distribution obtained by (22) as the marginal probability distribution of the model. The learning rate η is defined based on sigmoid function as follows:

$$\eta = \frac{2}{1 + \exp[-Q/(\tau q)]} - 1 \tag{17}$$

where τ is a parameter for adjusting the decay speed. Q is the maximum number of update parameters. We set the parameter τ to 0.25 and schedule the learning rate η so that it begins at 1.0 and decays as learning progresses. The regularization hyperparameter μ is set to 0.1 in the experiments.

4 Experiments

4.1 Experiment settings

The target dataset is provided by pairs of parking instructions collected through crowdsourcing and sensor information observed in the real world for parking. The parking instructions were input by crowdworkers who attempted to park their own cars in the parking spaces specified in presented images by a red arrow as indicated in Figure 4. There were 100 crowdworkers, and we collected 1,000 parking instructions with five different target parking spaces. Crowdworkers were given images of parking scenes from two viewpoints: the car (self) is located either at the left side or at the in front of the parking spaces . There were eight sets of environmental data; four were recorded at slightly different positions and directions of the car (self) where it was not far from the parking space, while the remaining four were recorded where the car was at the front of the parking space , similarly. In this study, we limited parking instructions to ones made from the viewpoint of the car (self) for simplicity. That is, the instructions "park at the right side of the white car (from the viewpoint of the car (self))" can be used; however, "park at the left of the electric car (from the viewpoint of the electric car)" was excluded. Instructions that were divided into two or more sentences were also excluded. We manually annotated ground truth labels for the data. The environment data consist of geographic environment map data with spatial location and extent of parking lots and bounding boxes of vehicles and other obstacles on occupancy grid maps converted from the measured LiDAR point clouds. In this study, we hand-labeled bounding boxes of objects as well as their object classes, i.e., vehicles or others. The ego vehicle is located at the center cell of the occupancy grid maps and ego location was estimated with not only RTK-GNSS positioning but also LiDAR-based scan matching with precise 3D environment map. The overlay of parking lots and object bounding boxes was synthesized by transforming the coordinates of bounding boxes on the occupancy grid to the ones on the geographic environment map relative to the location and the heading direction of the ego vehicle.

4.2 Evaluation

To confirm the effectiveness of the estimation of parking spaces specified by instructions, we used five-fold cross validation as the evaluation method. The results are summarized in Figure 5. We calculated the precision that is a ratio of correct groundings, i.e., correspondence between SDCs and the observed environmental information. In these figures, the "all" graph depicts the precision among all groundings "place" depicts the precision for the grounding of the parking space, and "object" depicts the precision for the grounding of the object The vertical axis of the figures represents the precision, while the horizontal axis represents the number of learning steps. In addition, the colored region shows the range in which the precision deviates from the mean value within -2σ to $+2\sigma$ where σ is the standard deviation calculated from five validation datasets. We conducted three experiments with different combinations of features: i) with quantitative features alone, ii) with qualitative features alone, and iii) with both, respectively. The precision is presented in Table 1. From Figures 5, it is found that the extent of the colored regions are relatively large, about 10% of the mean precision value. The results can be explained as follows. Because the initial parameters were randomly determined, they influenced subsequent learning. In addition, the data was too sparse because the only data from two viewpoints are used and it is not enough to learn spatial relation sufficiently.

After 5,000 steps of learning, the precision in the case of using both qualitative and quantitative features was the highest: 79.2% in "all". Using quantitative features alone resulted in the second highest precision: 77.2% in "all", while using qualitative features alone resulted in the lowest precision: 60.5%. The precision decreased by over 15% in learning with qualitative features because there were fewer features. We could distinguish between detailed places because distances and angles were punctuated by 1 meter and 5 degrees, respectively, in quantitative features. However, when using qualitative features, we estimated only order relations; therefore, it is not informative enough to capture spatial features of the real world. Quantitative features make it possible to distinguish between detailed places or objects, while qualitative features make it possible to capture the relation of objects if the place of the car (self) is tilted a little. For this two reasons, the precision in the case of using both qualitative and quantitative features was the highest. The update equation with the IPF algorithm fell into instability and did not converge when the empirical distribution became 0 due to the sparsity of the learning data. The experiments without regularization were also conducted, however they could not be calculated because of weight parameter overflow. Regularization parameter μ made it possible to learn stably.

Figure 4: View of parking scenes

	all	place	object
combination	79.2	81.6	83.0
qualitative	60.5	66.1	72.7
quantitative	77.2	79.4	82.0

Table 1: Precision of final step

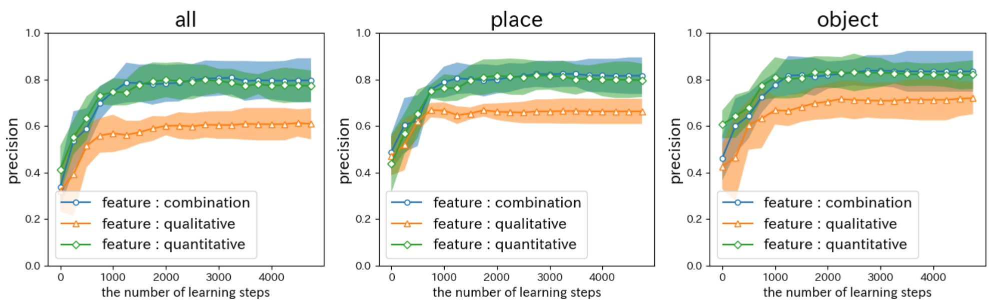

Figure 5: Variation of each precision with changing the number of learning steps

5 Conclusion

In this study, with the aim of parking an autonomous car using speech dialog, we proposed the method for estimating an appropriate parking space by properly understanding the content of users' verbal instructions. The method includes estimating the parking space by achieving grounding between the location specified in an instruction and a location in the real world. We then conducted the experiment with the proposed method. In terms of grounding parking instructions onto objects in the real world, we have thus far focused on simple SDCs and estimated the parameters of the grounding graphs and parking spaces by means of the belief propagation method. In parameter estimation, we proposed a stable learning method by introducing a regularization term into the IPF algorithm [11]. Furthermore, we obtained the accuracy of cases in which features were expressed as qualitative features, quantitative features, and both, and confirmed that the accuracy increased when both features were used.

We calculated the marginal probabilities of the grounding graphs G^3 with belief propagation algorithm. Belief propagation enables to more correctly estimate groundings considering the entire structure of graphs rather than the conventional approximation method , i.e., dividing a grounding graph into a product of graphs with only one factor node and the inference is made independently for each grounding variable of those factored graphs often used in the calculation for groundings. In future work, we plan to expand the size of training data for improving the inference accuracy of grounding graphs and extend the types of instructions, such as "Turn left at the next traffic light", so that we can treat the various driving instructions in addition to parking instructions. In addition, we plan to extend the SDC so that it can handle dynamic driving instructions that depend on temporal information, such as "Before the car behind us approaches, change to the next lane."

Acknowledgement

The authors thank Prof. Manabu Oomae of Keio University for kindly providing us their autonomous driving cars and experiment environment. We would also like to express our gratitude to Waymo's sponsorship and the literary fusion AI-Data science (AI-DS) center of Ochanomizu University for their financial support to attend the workshop.

References

[1] Stefanie Tellex, Thomas Kollar, Steven Dickerson, Matthew R Walter, Ashis Gopal Banerjee, Seth Teller, and Nicholas Roy. Understanding natural language commands for robotic navigation and mobile manipulation. In Twenty-Fifth AAAI Conference on Artificial Intelligence, 2011.

[2] Mark Steedman. The syntactic process, volume 24. MIT press Cambridge, MA, 2000.

[3] Thomas Kollar, Stefanie Tellex, Deb Roy, and Nicholas Roy. Toward understanding natural language directions. In Proceedings of the 5th ACM/IEEE international conference on Human-robot interaction, pages 259–266. IEEE Press, 2010.

[4] Judea Pearl. Reverend Bayes on inference engines: A distributed hierarchical approach. Proceeding of the Second AAAI Conference on Artificial Intelligence (AAAI'82), 1982.

[5] Akari Inago, Hiroshi Tsukahara, and Ichiro Kobayashi. Parsing parking instructions for self-driving cars into spatial semantic descriptions. ICICA, 2019.

[6] Thomas M Howard, Stefanie Tellex, and Nicholas Roy. A natural language planner interface for mobile manipulators. In 2014 IEEE International Conference on Robotics and Automation (ICRA), pages 6652–6659. IEEE, 2014.

[7] Rohan Paul, Jacob Arkin, Nicholas Roy, and Thomas M Howard. Grounding abstract spatial concepts for language interaction with robots. In IJCAI, pages 4929–4933, 2017.

[8] Anna Rohrbach, Marcus Rohrbach, Ronghang Hu, Trevor Darrell, and Bernt Schiele. Grounding of textual phrases in images by reconstruction. In European Conference on Computer Vision, pages 817–834. Springer, 2016.

[9] Yue Zhang and Stephen Clark. Shift-reduce ccg parsing. In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1, pages 683–692. Association for Computational Linguistics, 2011.

[10] Stefanie Tellex. Natural language and spatial reasoning. PhD thesis, Massachusetts Institute of Technology, 2010.

[11] Stephen E Fienberg and Michael M Meyer. Iterative proportional fitting. Technical report, CARNEGIE-MELLON UNIV PITTSBURGH PA DEPT OF STATISTICS, 1981.

[12] Jonathan S Yedidia, William T Freeman, and Yair Weiss. Generalized belief propagation. In Advances in neural information processing systems, pages 689–695, 2001.

A Belief propagation algorithm

Belief propagation makes it possible to calculate the marginal probability distribution of unobserved nodes based on information of the observed nodes. Initially, messages from all factor nodes α to variable nodes i, and messages from all variable nodes i to factor nodes α are initialized as 1 in the calculation of the marginal probability distribution of the grounding graphs. Next, alternate message $m_{\alpha\to i}$ propagates from variable node $j \in x_\alpha \backslash i$ that connects to factor node α except i and message $m_{\beta\to i}$ propagates from other factor nodes $\beta \in \partial i \backslash \alpha$ connecting through α. This is repeated until the value converges to a fixed point or the number of updates reaches a predetermined maximum. The convergence decision is made when the absolute value of the difference between a message updated t times and a message updated $t+1$ times is less than the predetermined residual error, ϵ. The marginal probability of factor nodes $p(x_\alpha)$ and variable nodes $p(x_i)$ can be calculated with the converged messages. The marginal probability distribution of the factor nodes is used when learning, while that of the variable nodes is used when inferring the type of car manipulation or objects. The above belief propagation algorithm and equations are presented in Algorithm 2 [12].

Algorithm 2 Belief propagation algorithm

1. Initialize messages

$$m^{(0)}_{\alpha\to i} = 1, m^{(0)}_{i\to\alpha} = 1 \tag{18}$$

2. Propagate messages until they converge

$$m^{(t+1)}_{\alpha\to i}(x_i) \propto \sum_{x_{\alpha\setminus i}} \Psi_\alpha(x_\alpha) \prod_{j\in x_\alpha\setminus i} m^{(t)}_{j\to\alpha}(x_j) \tag{19}$$

$$m^{(t+1)}_{i\to\alpha}(x_i) \propto \prod_{\beta\in\partial i\setminus\alpha} m^{(t)}_{\beta\to i}(x_i) \tag{20}$$

3. Calculate marginal probabilities by the converged message values $m^*_{\alpha\to i}$ and $m^*_{j\to\alpha}$

$$p(x_i) \propto \prod_{\alpha\ni i} m^*_{\alpha\to i}(x_i) \tag{21}$$

$$p(x_\alpha) \propto \Psi_\alpha(x_\alpha) \prod_{j\in x_\alpha} m^*_{j\to\alpha}(x_j) \tag{22}$$