Natural Language Grounding and Grammar Induction for Robotic Manipulation Commands

M Alomari, P Duckworth, M Hawasly, D C Hogg, A G Cohn
{scmara, scpd, m.hawasly, d.c.hogg, a.g.cohn}@leeds.ac.uk,
School of Computing, University of Leeds, Leeds LS2 9JT, England

Abstract

We present a cognitively plausible system capable of acquiring knowledge in language and vision from pairs of short video clips and linguistic descriptions. The aim of this work is to teach a robot manipulator how to execute natural language commands by demonstration. This is achieved by first learning a set of visual 'concepts' that abstract the visual feature spaces into concepts that have human-level meaning. Second, learning the mapping/grounding between words and the extracted visual concepts. Third, inducing grammar rules via a semantic representation known as Robot Control Language (RCL). We evaluate our approach against state-of-the-art supervised and unsupervised grounding and grammar induction systems, and show that a robot can learn to execute never seen-before commands from pairs of unlabelled linguistic and visual inputs.

1 Introduction

Understanding natural language commands is essential for robotic systems to naturally and effectively interact with humans. In this paper, we present a framework for learning the linguistic and visual components needed to enable a robot manipulator of executing new natural language commands in a table-top environment. The learning is divided into three steps: (i) learning of visual concepts, (ii) mapping the words to the extracted visual concepts (i.e. language grounding), and (iii) inducing grammar rules to model the natural language sentences. Our system updates its knowledge in language and vision incrementally, by processing a pair of inputs at a time. The input to our system consists of a short video clip of a robot performing a single action, e.g. a *pick up* or a *move* action, paired with a natural language command corresponding to the action in the video. The natural language commands were collected from volunteers and online crowd-sourcing tools such as Amazon Mechanical Turk with minimal amount of supervision or constraints on the language structure which annotators could use.

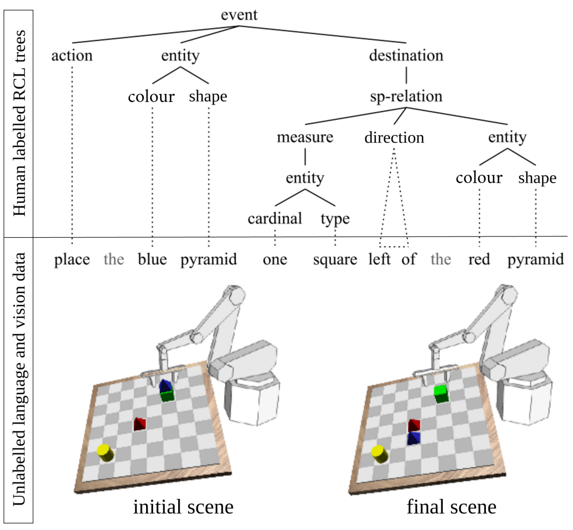

Figure 1: Human expert annotation of a natural language command. The annotation includes grounding for each word and an RCL tree.

Generally, *supervised* language grounding and grammar induction systems learn from sentences that have been *manually* annotated by a human expert. As shown in Fig. 1, each word gets annotated with a semantic category (e.g. colour, shape, etc.), and the grammar structure gets annotated using a tree that connects the different words together (e.g. RCL trees) as presented by Dukes (2014) and Matuszek (2013). The manual annotation of data is a labour intensive task that hinders learning from large corpora, and such labels are not necessarily available for all languages. Therefore, *unsupervised* grounding and grammar induction systems learn language models from unlabelled/raw linguistic data by exploiting co-occurrences of words in a corpus, which generally performs poorly. Therefore, in this work we take a different approach to learn words meanings and grammar rules by connecting natural language to extracted visual features from video clips.

2 Related Work

Language acquisition has been a long standing objective of AI and cognition research. Siskind (1996) was one of the earliest researchers to try to understand in a computational setting how children learn their native language and map it to their vision. Following his research, in the field of developmental robotics, researchers have connected language and vision to teach their robots different concepts; one of the earliest works in the field is a system by Roy *et al.* (1999) where a robot capable of learning audio-visual associations (e.g. objects' names) using mutual information criterion was presented. Several robotic applications were developed subsequently, such as Steels (2001) where language games for autonomous robots are used to teach the meaning of words in a simple static world. Further, researchers developed systems capable of learning objects' names and spatial relations by interacting with a human or robot teacher, as by Steels (2002), Bleys (2009) and Spranger (2015). Providing machines with the ability to understand natural language commands is a key component for a natural human-robot interaction. For example, "Back to the blocks world" (She et al., 2014) and "Tell me Dave" (Misra et al., 2015) focused on learning the natural language commands for simple manipulation tasks. This is similar to our work, but we improve on their work in three different aspects. First, their works use a pre-trained language parser to extract relevant words from sentences for learning, while we learn from raw (unprocessed) linguistic inputs. Second, they assume the robot knows the visual representations of shapes, spatial-relations and actions beforehand, while we learn these automatically from videos. Finally, we learn the grammar rules along with word groundings.

Language grounding systems in robotic applications are usually trained in a supervised setting on a corpus of labelled/tagged text as in Tellex *et al.* (2011), Bollegal *et al.* (2015), and Cui *et al.* (2016). The manual annotation of text is a labour intensive task. Therefore, researchers developed unsupervised techniques that learn the semantic categories of words from unlabelled data by exploiting regularities in natural language as in Schütze (1998), Biemann (2009), Socher *et al.* (2012), and Houthooft *et al.* (2016). Similarly, in **grammar induction**, parsers are commonly trained in a supervised setting on a corpus of annotated grammar trees as presented by Matuszek *et al.* (2013), and Dukes (2014). Other researchers have tackled unsupervised grammar induction from unlabelled sentences as presented by Klein *et al.* (2002), Smith and Eisner (2005), Barzilay *et al.* (2009), Chen *et al.* (2011), Ponvert *et al.* (2011), and Søgaard (2012). While unsupervised grounding and grammar induction techniques enable learning from unlabelled data, their performance is usually significantly worse than those of the supervised techniques. In this work, we present a novel technique capable of acquiring grounding and grammar knowledge comparable to supervised techniques from unlabelled data by mapping words to automatically extracted visual concepts from video clips.

3 Learning Visual Concepts ($\mathcal{C}$)

In this section, we describe how we represent the visual input data: we first extract a set of visual features from each video clip; then, we show how we abstract values from these features to form a set of clusters (or visual concepts). These clusters are used to learn the visual representation of words in the language grounding section.

We start by processing the video clips to detect and track the objects in each frame. The objects are detected using a table-top object detector (Muja and Ciocarlie, 2013), where each object in a video is assigned a unique id (a number), and its location is tracked using a particle filter (Klank et al., 2009). Next, we obtain three sets of observations from each video clip; (i) object features: $\{colour, shape, location\}$ of each object, (ii) relational features: $\{direction, distance\}$ of each pair of objects in the scene, and (iii) the $\{atomic\ actions\}$ that the robot applies on each object during the video. The features and atomic actions are presented in Fig. 2. These features and actions are obtained at every frame in each video. It is worth noting that these features are not intended to be exhaustive, but rather to demonstrate our approach; more features can be added as an extension without changing the learning framework.

Object

- F_{colour} : $object \rightarrow ([0,360) \times [0,1] \times [0,1])$; $F_{colour}(o)$ gives HSV colour values for an object.
- F_{shape} : $object \rightarrow \mathbb{R}^{32}$; $F_{shape}(o)$ gives a 32 bin fast point feature histogram (FPFH) per object.
- $F_{location}$: $object \rightarrow \mathbb{R} \times \mathbb{R} \times \mathbb{R}$; $F_{location}(o)$ gives an x, y, z location wrt the bottom left corner of the table.

Relational

- $F_{distance}$: $object \times object \rightarrow \mathbb{R}$; $F_{distance}(o_1,o_2)$ gives the distance between o_1 and o_2 centroids.
- $F_{direction}$: $object \times object \rightarrow [0,360) \times [0,360)$; $F_{direction}(o_1,o_2)$ gives the azimuth and altitude angles from o_1 to o_2.

Atomic Actions

- F_{action} : $object \rightarrow [approach, depart, grasp, discard, lift, lower, move, rotate, null]$; $F_{action}(o)$ gives the atomic action, as shown below.

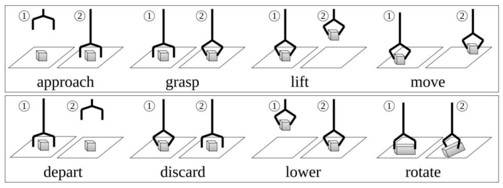

Figure 2: Predefined features and atomic actions.

Once the observations (objects, relations, actions) have been obtained for all objects in all video clips, we process them to extract the unique concepts, e.g. distinguishing the shape *cube* from the shape *prism*, and the action *pick up* from *put down*, etc. This is achieved by clustering the values of each feature space separately to obtain multiple clusters in the dataset. The extracted clusters are used to construct a visual concept vector ($\mathcal{C}$) with length equal to the total number of clusters. This forms the list of possible visual representations of words. For instance, Dukes (2013) dataset (intended to train semantic taggers in a supervised setting), contains four unique shapes: prism, cube, ball, and cylinder. We cluster the shape values of all objects from all video clips and are thus able to extract these clusters/shapes, e.g. $shape_1$ = cube, $shape_2$ = prism, $shape_3$ = ball, and $shape_4$ = cylinder. The clustering is performed using a combination of Gaussian Mixture Models and Bayesian Information Criterion to find the optimal number of clusters representing the data in each feature space. The same clustering method is used on all observations (*colours, locations, directions, distances, and atomic actions*) each of which is done separately, then the outputs (or the clusters) are combined into a single vector $\mathcal{C} = \{shape_1, shape_2, shape_3, shape_4, colour_1, \ldots, location_1, \ldots, direction_1, \ldots, distance_1, \ldots, action_1, \ldots\}$ to give the visual representations of words. This is used in the next section for language grounding. After generating the vector $\mathcal{C}$, we go through each of the video clips and represent the observed contents of each clip (objects, relations, actions) as a collection of entries or predicates. For example, an object with $id = 3$ and shape $shape_1$ ('cube') is represented as the entry $shape_1(3)$. An example from the dataset is shown in Fig. 3 where the entries are shown on the right.

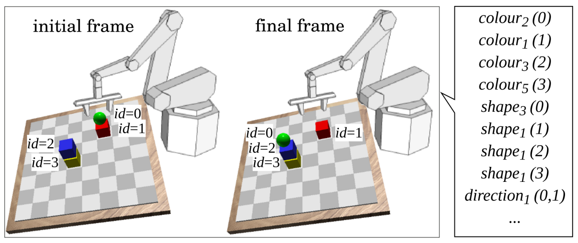

Figure 3: The initial and final frames of a scene from the Dukes (2013) dataset represented in predicates using the learned clusters in $\mathcal{C}$.

4 Language Grounding (Φ)

Assigning a word to its correct visual representation is an essential preprocessing step for understanding and executing natural language commands in robotics. In this section, we show how we connect words to their visual representations that we extracted in the previous section. The problem statement of this section is: given (1) a corpus of n sentences $S = \{s_1, \ldots, s_n\}$ that contains m unique words $W = \{w_1, \ldots, w_m\}$, and (2) corresponding video clips $V = \{v_1, \ldots, v_n\}$ that contain k extracted visual concepts $\mathcal{C} = \{c_1, \ldots, c_k\}$, find a partial function Φ that maps words from language to their representations in vision, $\Phi : W \rightarrow \mathcal{C}$. This language grounding learning problem is formulated as an assignment problem where words $w_i \in W$ should be assigned to clusters $c_j \in \mathcal{C}$ subject to a cost function $\mathcal{F} : W \times \mathcal{C}$ that needs to be minimised. We define the cost function as $\mathcal{F}_{w,c} = (1 - (N_{w,c}/N_w))$, where $N_{w,c}$ is the total number of times a word w and a cluster c appear together, and N_w is the total number of times the word w appears in the entire dataset. This cost function is equal to zero if w and c always appear together, and equal to one if they are never seen together. This provides a clear indication of whether a word w should be mapped to a cluster c or not.

Once the cost function is computed for all word-cluster pairs, we create a cost matrix with words W as rows and clusters $\mathcal{C}$ as columns, as shown in Fig. 4 (left). We then use the Hungarian algorithm

(Kuhn, 1955) to find the grounding for each word by assigning it to its most likely visual concept, i.e. a word can only have one meaning (though multiple words could have the same meaning), as shown in Fig. 4 (right).

To simplify the learning of language grounding in NLP applications, it is common to use a stop word list to remove function words, such as 'the' and 'as', from all sentences. However, since we learn from unlabelled data (i.e. avoiding human annotation including stop word lists), we remove such words by setting a threshold on the Hungarian algorithm. This has the same effect as using term frequency-inverse document frequency (tf-idf) weighting to remove function words (Jones 1972).

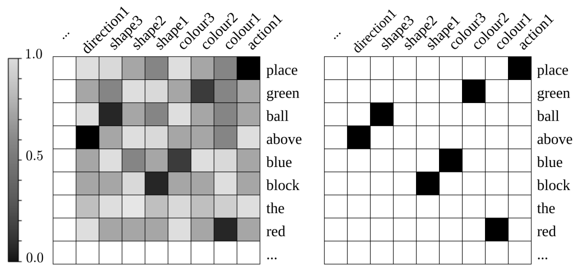

Figure 4: (left:) The cost matrix. (right:) The output of the Hungarian algorithm.

5 Generation of RCL Trees (Ω)

Robot Control Language (RCL) is a tree semantic representation for natural language. As shown in Fig. 1, a sentence is represented as an RCL tree where leaf nodes align to the words in the sentence, and non-leaves are tagged using a predefined set of categories that a robot can understand/execute as presented by Matuszek (2013) and Dukes (2013). Although the RCL used in this work is designed to operate within the context of robot manipulation only, it can be extended to other domains such as robot navigation, learning from YouTube how-to videos (Alayrac et al., 2016), or learning cooking instructions (Malmaud et al., 2015). Table 1 lists the different types of RCL elements that are used to compose natural language commands.

In the literature, the problem of transforming sentences into RCL trees has been formulated as a grammar induction one. A parser is trained on pairs of sentences and corresponding human annotated RCL trees, as shown in Fig. 1. The parser is then used to parse new (previously unseen) sentences into RCL trees. The human annotation of RCL trees is labour-intensive, and it would prevent the

RCL element	*Description*
event	Specification of a single command. Takes (action, entity, destination) as children.
action	Aligned to a verbal group in NL, e.g. 'place'.
entity	Specification of a single entity. Takes (colour, shape, location, sp-relation) as children.
sp-relation	Used to specify a spatial relation between two entities or to describe a location. Takes (direction, distance, entity) elements as children.
destination	A spatial destination. Takes (sp-relation, location) as children.
colour	Colour attribute of an entity, e.g. 'red'.
shape	Shape attribute of an entity, e.g. 'pyramid'.
location	Location attribute of an entity, e.g. 'centre'.
direction	Direction relation between two entities.
distance	Distance relation between two entities.

Table 1: Universal semantic elements in RCL.

robot from learning without constant supervision.

In this paper, we automatically generate a *vision tree* Ω_i from each video clip $v_i \in V$. We define a *vision tree* as an event tree with only three elements ($action$, $entity$, $destination$), as shown in the example in Fig. 5. The $action$-element holds the action feature extracted from the video v_i, the $entity$-element has the *id* of the object that is manipulated by the robot in the video, and the $destination$-element has the final (x, y, z) location feature of that object. In the next section, we show how to use the vision trees $\Omega = \{\Omega_1, \ldots, \Omega_n\}$ to automatically generate language RCL trees analogous to the one in Fig. 1.

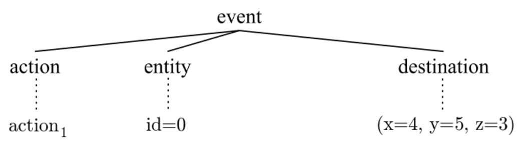

Figure 5: An example of a *vision tree* from the video clip shown in Fig. 3.

6 Grammar Induction (G)

Grammar induction refers to the process of learning a formal grammar (usually as a collection of re-write rules or productions) from a set of observations. In this work, we show how we learn such rules by mapping natural language commands to visual features. The main contribution of our grammar induction approach is that we *automatically* generate training examples similar to those annotated by a human expert shown in Fig. 1. This is achieved by exploiting the learned groundings Φ (shown in Fig 4) and the extracted vision trees Ω (shown in Fig. 5) to successfully replace the human annotator. We formulate the automatic generation of language RCL trees into a search problem as follows. Given (1) vision trees $\Omega = \{\Omega_1, \ldots, \Omega_n\}$, (2) the learned grounding $\Phi : W \rightarrow \mathcal{C}$, and (3) input sentences $S = \{s_1, \ldots, s_n\}$, we want to search the space of all possible language RCL trees from a sentence $s_i \in S$ for one that matches the extracted vision tree $\Omega_i \in \Omega$. Given a match, we use that language tree to learn grammar G. We say a language RCL tree matches a vision tree if the values of all corresponding elements are equal. The procedure to perform the search is divided into five steps (*substitute, connect, query, match,* and *learn*) shown in Algorithm 1. The following sections walk through an example of the entire search process.

Algorithm 1 Automatic generation of language RCL trees

```
1:  procedure SEARCH ALGORITHM
2:  Inputs $\Phi, \Omega, S$
3:  Output $G$
4:  for each sentence $s_i \in S$ do
5:      Substitute each word in $s_i$ with its visual
        concepts in $\Phi$
6:      Connect vision concepts to create language
        RCL elements.
7:      Query the RCL elements with video $v_i$.
8:      Match RCL elements with the vision RCL
        tree $\Omega_i$.
9:      if all RCL elements match with $\Omega_i$ then
10:         Use RCL elements to learn grammar $G$
```

6.1 Substituting words with visual concepts

For each sentence $s_i \in S$ consisting of t words $s_i = \{w_1, \ldots, w_t\}$, we substitute each word with its visual concept using the mapping function Φ learned in the language grounding section. For instance, a sentence $s_i = \langle place, the, green, ball, above, the, blue, block\rangle$, is transformed using the mapping function Φ into $s'_i = \langle action_1, None, colour_2, shape_3, direction_1, None, colour_3, shape_1\rangle$, as shown in Fig. 6 (*substitute*).

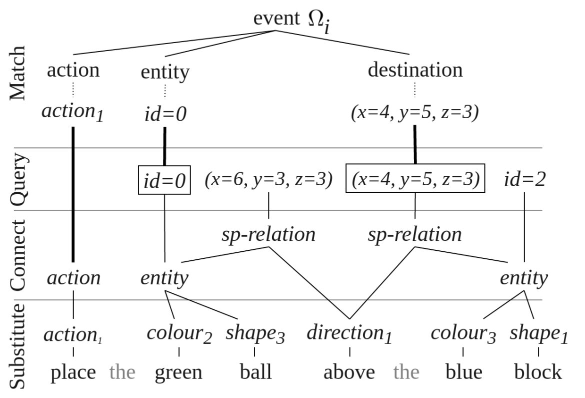

Figure 6: Example for the generation of language RCL tree.

6.2 Connecting concepts to generate RCL elements

We group the visual concepts in s'_i to create all possible *entity, action*, or *sp-relation* RCL elements (specified in Table. 1). Particularly, consecutive $colour$, $shape$, and $location$ concepts are grouped to form *entity* RCL elements, consecutive $action$ concepts form *action* RCL elements, and consecutive $direction$ and $distance$ concepts with $entity$ elements are grouped to form *sp-relation* RCL elements. For example, in the sentence $s'_i = \langle action_1, colour_2, shape_3, direction_1, colour_3, shape_1\rangle$ the visual concept $colour_2$ and $shape_3$ are grouped together to generate an *entity* element $entity(colour_2, shape_3)$. The same procedure applies to *action* and *sp-relation* elements, as shown in Fig. 6 (*Connect*).

6.3 Querying RCL elements

For each connected *entity* and *sp-relation* RCL element from the previous step, we query the observations in the scene to retrieve a corresponding object id or an (x, y, z) location. For instance, the element $entity(colour_2, shape_3)$ matches from the extracted observations shown in Fig. 3 (right) a single object with $id = 0$, as it is the only object that satisfies both query properties $colour_2$ and $shape_3$ ('green ball' in this case). Similarly, querying the $sp-relation(direction_1, entity(colour_3, shape_1))$ returns $(x = 4, y = 5, z = 3)$ referring to 'above blue block'. This is repeated for all connected entities and spatial relations, as shown in Fig. 6 (*Query*).

If multiple objects in the scene satisfy a query, a list of ids is returned, while if there are none the query returns an empty list (this might happen due to noise in vision and/or language). In the results section we show that our system is capable of learning using these connections, even in the presence of noise from real-world data.

6.4 Matching RCL elements with vision RCL trees Ω

Given the results of the previous process, we match the returned query results to elements from the vision RCL trees in Ω. For example, the vision RCL tree Ω_i in Fig. 5 has an *entity* element with $id = 0$, matching the query output in Fig. 6, thus matching it with 'green ball'. This is repeated for *action* and *destination* elements in Ω_i. This process grounds linguistic descriptions to their visual counterparts in RCL without human supervision as shown in Fig. 6 (*Match*).

6.5 Learning grammar G

To provide a robot with the ability of understanding natural language commands, we learn grammar G from the automatically generated language RCL trees. The grammar induction is performed using probabilistic context free grammar (Charniak, 1997), by training a semantic parser on the automatically generated examples. The parser is then used in the experiment section to parse new natural language commands into Robot Control Language (RCL) trees. This concludes the search process.

7 Experimental Procedure

We evaluate the performance of our system using two datasets; a synthetic-world dataset, and a new, simplified real-world dataset of table-top environment.

For the **synthetic-world**, we use the *Train Robots* dataset[1] which was designed to develop systems capable of understanding verbal spatial commands described in a natural way (Dukes, 2013). Non-expert users from Amazon Mechanical Turk were asked to annotate appropriate natural language commands to 1000 different scenes. A total of 4850 commands were collected and later annotated by human experts with appropriate RCL trees. Examples from this synthetic dataset are shown in Fig. 1 and 3.

For the **real-world** setup, we use a Baxter robot as our test platform and attach a Microsoft Kinect2 sensor to its chest as shown in Fig. 7. The Kinect2 device is used to collect RGBD videos as volunteers controlled the robot arm to perform various manipulation tasks with real objects from the robot's point of view. The dataset consists of 204 videos with $17,373$ frames in total. The videos are annotated with 1024 natural language commands (5 per video in average) by a separate group of volunteers[2]. A total of 51 different objects are manipulated in the videos such as basic block shapes, fruit, cutlery, and office supplies. A detailed description of both datasets is presented in Table. 2.

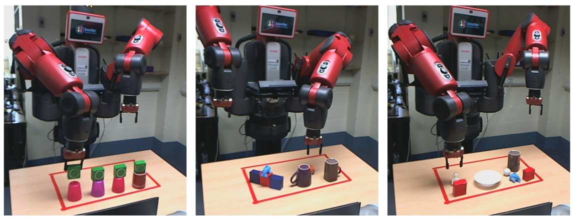
Figure 7: Example scenes from our robotic dataset.

Dataset contents							
features	A	B	C	D	E	F	G
Synthetic	9	4	4	3	NA	4	24.8
Real-world	11	13	3	5	2	3	5.3

Table 2: Number of concepts in A-colour, B-shape, C-location, D-direction, E-distance, and F-action features in both datasets, and G-average number of objects present in each scene.

7.1 Implementation Details

For the real-world dataset, **objects** are detected using a tabletop object detector on the first frame in each video. These objects are then tracked throughout the video using a six dimensional (x, y, z, r, g, b) particle filter, as shown in Fig. 8.

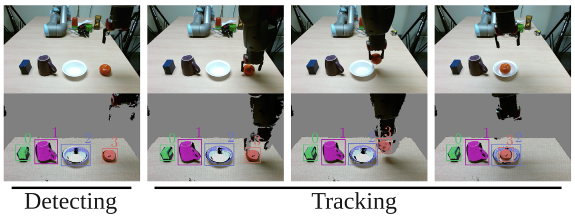

Figure 8: Example of a video sequence "place the orange in the bowl" when the objects are tracked using a particle filter. (Best viewed in colour.)

During the learning process, we use *atomic actions* (Fig. 2) to represent more **complex actions** in videos. For example a 'pick up' action is

[1]*Train Robots*: http://doi.org/10.5518/32

[2]Baxter dataset: http://doi.org/10.5518/110

represented with the sequence (*approach*, *grasp*, *lift*) as the robot approaches, grasps and lifts the object, while a 'drop' action is represented with just (*discard*) as the robot lets go of the object to fall down on the table.

We automatically detect **function words** by setting a threshold of $\sigma = .6$ on the Hungarian algorithm. Thus a word w is considered a function word if it is not consistent with any cluster $f_j \in F$ by more than 60% in the entire dataset. This threshold detects all function words.

In our **experiments**, we divide each dataset randomly into four equal parts, and perform four-fold cross validation, where we train on three folds and test on the fourth.

7.2 Evaluation

We evaluated the performance of our technique using two metrics: (i) the ability to correctly ground words to the learned visual concepts using Φ, and (ii) the ability to correctly parse previously unseen natural language commands to produce correct RCL trees using the learned grammar G.

To better demonstrate our results in language grounding and grammar induction, we compare our technique with (1) a supervised system that learns from labelled data, and with (2) an unsupervised system that learns from unlabelled linguistic data. We consider our **baseline** as the performance of the unsupervised system, i.e. our joint language and vision technique should outperform the unsupervised system that learns from unlabelled linguistic inputs, otherwise there is no benefit of the additional vision component. On the other hand, an **upper bound** on performance is the results of the supervised system trained on human labelled (ground-truth) data.

7.3 Language Grounding Experiment

In this section, we evaluate the system's ability to acquire correct groundings for words from parallel pairs of short video clips and linguistic descriptions. The given task is to learn the partial function $\Phi : W \rightarrow \mathcal{C}$ that maps words $w_i \in W$ to their corresponding clusters $c_j \in \mathcal{C}$, e.g. the word 'red' should be mapped to the cluster colour-*red*.

The results for our language grounding experiment are shown in Fig. 9. Here, 'our-system' is compared against (1) the supervised semantic tagger (Fonseca and Rosa, 2013) that is trained on human labelled data, and (2) the unsupervised semantic tagger (Biemann, 2009) that is trained on unlabelled linguistic data. The results are calculated based on the total number of correct tags/groundings assigned to each word in the test fold (four fold cross validation). Note that for the unsupervised system, the results are calculated based on its ability to cluster words that belong to the same category together, i.e. words that describe colours should be given a unique tag different to those that describe shapes, directions, etc. Also, we assign new words in the test fold (words that only exist in the test fold) with a *function word* tag.

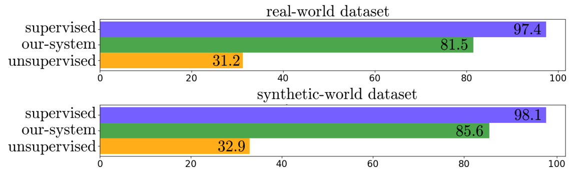

Figure 9: The grounding results of (a) supervised, (b) our system, and (c) unsupervised semantic taggers, on both datasets.

Our system is able to correctly ground (85.6%) of the total words in the synthetic, and (81.5%) in the real-world datasets, compared to only (32.9% and 31.2% respectively) using the unsupervised system. This clearly shows that adding vision inputs produces more correct semantic representations for words, even though both systems use unlabelled data for learning. Detailed analysis of how the different techniques performed in each feature space is shown in Fig. 10. Note that *distance* is not a feature in the synthetic dataset and therefore the corresponding row/column are left empty.

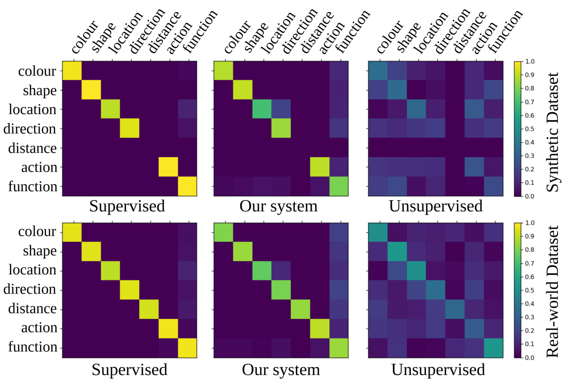

Figure 10: The grounding/tagging performance in each feature space for the three systems on both real-world and synthetic datasets.

7.4 Grammar Induction Experiment

In this experiment we test our system's ability to acquire correct grammar rules G from pairs of video clips and unlabelled sentences (with no human-annotated RCL trees). The learned grammar G is then used to parse new (previously unseen) natural language commands. We compare our technique with (1) a supervised parser (Abney, 1996) trained on labelled data, i.e. pairs of sentences and human-annotated RCL trees, and (2) an unsupervised parser (Ponvert *et al.* 2011) trained on unlabelled sentences, i.e. a corpus of sentences without RCL trees or semantic tags.

The results for (a) our approach, (b) the supervised parser, and (c) the unsupervised grammar induction systems on both datasets are shown in Fig. 11. The results were calculated based on the number of correctly parsed RCL trees from sentences in the test fold (in the four-fold cross validation). A score of 1 is given if the parsed sentence completely matches the human annotation, while a partial score in $(0, 1)$ is given if it partially matches the human annotation. The partial matching is computed by matching subtrees in the both trees divided by the total number of subtrees. For example, if a tree contains 10 subtrees and only 8 of which has a complete match in labels and links, then we give a score of 0.8 to this tree.

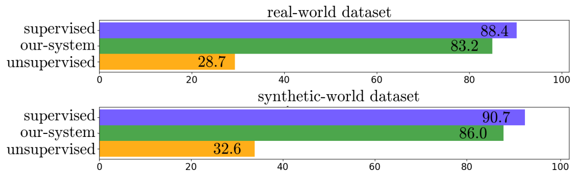

Figure 11: The grammar induction results for (a) supervised, (b) our system, and (c) unsupervised parsers on both real-world and synthetic datasets.

The results in Fig. 11 clearly show that our approach outperforms the unsupervised grammar induction system and achieves comparable results to the supervised system by learning from both language and vision as opposed to learning from language alone. The number of grammar rules generated differs between techniques: our approach generated (139 and 87) grammar rules from the synthetic and real-world datasets respectively, while the supervised system generated (182 and 114) and the unsupervised system generated (45 and 38) grammar rules, respectively.

8 Conclusion and Discussion

We present a novel technique to simultaneously learn the groundings of words and simple grammar rules of natural language. Our learning framework connects words from sentences to automatically-extracted visual clusters from videos to enable automatic generation of RCL trees, which is a key contribution of this paper. These trees act as an intermediary representation between the continuous perceptual space and the purely symbolic linguistic structures, thus provide robots with the ability to automatically learn about language, actions and perception. Our approach outperforms unsupervised techniques in both semantic tagging and grammar induction in learning from unlabelled data, and provides comparable performance to language-only supervised approaches.

Our approach suffers from two main limitations that hinder learning from longer videos (such as *YouTube* videos). First, it requires the videos and sentences to be temporally aligned beforehand, and second, it requires the feature spaces (e.g. $colours$, $shapes$, etc.) to be specified beforehand (though not their discretisation, which is learned). In order to allow learning from such data, our system should be able to learn from continuous, unaligned videos and documents, and it should be able to generate new feature spaces to cope with new emerging concepts. We aim to address these limitations in future work.

Acknowledgments

We acknowledge the financial support provided by EU FP7 project 600623 (STRANDS), and the the anonymous referees for their useful comments.

References

Steven Abney. 1996. Partial parsing via finite-state cascades. *Natural Language Engineering* 2(04).

Jean-Baptiste Alayrac, Piotr Bojanowski, Nishant Agrawal, Josef Sivic, Ivan Laptev, and Simon Lacoste-Julien. 2016. Unsupervised learning from narrated instruction videos .

Chris Biemann. 2009. Unsupervised part-of-speech tagging in the large. *Research on Language and Computation* 7(2-4):101–135.

Joris Bleys, Martin Loetzsch, Michael Spranger, and Luc Steels. 2009. The grounded colour naming game. *Proceedings of Spoken Dialogue and Human-Robot Interaction Workshop at the RoMan 2009 Conference* .

Danushka Bollegala, Alsuhaibani Mohammed, Takanori Maehara, and Ken-ichi Kawarabayashi. 2015. Joint word representation learning using a corpus and a semantic lexicon. *arXiv preprint arXiv:1511.06438* .

Eugene Charniak. 1997. Statistical parsing with a context-free grammar and word statistics. *AAAI/IAAI* 2005(598-603):18.

David L Chen and Raymond J Mooney. 2011. Learning to interpret natural language navigation instructions from observations. In *AAAI*. volume 2.

Wanyun Cui, Xiyou Zhou, Hangyu Lin, Yanghua Xiao, Haixun Wang, Seung-won Hwang, and Wei Wang. 2016. Verb pattern: A probabilistic semantic representation on verbs .

Kais Dukes. 2013. Semantic annotation of robotic spatial commands. In *Language and Technology Conference (LTC)*.

Kais Dukes. 2014. Semeval-2014 task 6: Supervised semantic parsing of robotic spatial commands. *SemEval 2014* page 45.

Erick R Fonseca and Joao Luis G Rosa. 2013. A two-step convolutional neural network approach for semantic role labeling. In *Neural Networks (IJCNN), The 2013 International Joint Conference on*. IEEE.

Rein Houthooft, Cedric De Boom, Stijn Verstichel, Femke Ongenae, and Filip De Turck. 2016. Structured output prediction for semantic perception in autonomous vehicles. In *AAAI*. AAAI Press.

Eric H Huang, Richard Socher, Christopher D Manning, and Andrew Y Ng. 2012. Improving word representations via global context and multiple word prototypes. In *ACL*. ACL, pages 873–882.

Ulrich Klank, Dejan Pangercic, Radu Bogdan Rusu, and Michael Beetz. 2009. Real-time CAD Model Matching for Mobile Manipulation and Grasping. In *9th IEEE-RAS International Conference on Humanoid Robots*. Paris, France, pages 290–296.

Dan Klein and Christopher D Manning. 2002. A generative constituent-context model for improved grammar induction. In *ACL*. ACL, pages 128–135.

Harold W Kuhn. 1955. The Hungarian method for the assignment problem. *Naval research logistics quarterly* 2(1-2):83–97.

Jonathan Malmaud, Jonathan Huang, Vivek Rathod, Nick Johnston, Andrew Rabinovich, and Kevin Murphy. 2015. What's cookin'? interpreting cooking videos using text, speech and vision. *arXiv preprint arXiv:1503.01558* .

Cynthia Matuszek, Evan Herbst, Luke Zettlemoyer, and Dieter Fox. 2013. Learning to parse natural language commands to a robot control system. In *Experimental Robotics*. Springer, pages 403–415.

Dipendra K Misra, Jaeyong Sung, Kevin Lee, and Ashutosh Saxena. 2015. Tell me Dave: Context-sensitive grounding of natural language to manipulation instructions. *JAIR* page 0278364915602060.

Marius Muja and Matei Ciocarlie. 2013. tabletop object detector - ROS Wiki. http://www.ros.org/wiki/tabletopobjectdetector.

Elias Ponvert, Jason Baldridge, and Katrin Erk. 2011. Simple unsupervised grammar induction from raw text with cascaded finite state models. In *Proceedings of the 49th Annual Meeting of the ACL: Human Language Technologies*. ACL, Oregon, USA.

Deb Roy, Bernt Schiele, and Alex Pentland. 1999. Learning Audio-Visual Associations using Mutual Information. In *Integration of Speech and Image Understanding, 1999. Proceedings*. IEEE.

Hinrich Schütze. 1998. Automatic word sense discrimination. *Computational linguistics* 24(1):97–123.

Lanbo She, Shaohua Yang, Yu Cheng, Yunyi Jia, Joyce Y Chai, and Ning Xi. 2014. Back to the blocks world: Learning new actions through situated human-robot dialogue. In *15th Annual Meeting of the Special Interest Group on Discourse and Dialogue*. volume 89.

Jeffrey Mark Siskind. 1996. A Computational Study of Cross-Situational Techniques for Learning Word-to-Meaning Mappings. *Cognition* 61(1):39–91.

Noah A Smith and Jason Eisner. 2005. Contrastive estimation: Training log-linear models on unlabeled data. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*. ACL.

Benjamin Snyder, Tahira Naseem, and Regina Barzilay. 2009. Unsupervised multilingual grammar induction. In *Proceedings of the Joint Conference of the 47th Annual Meeting of ACL*. ACL, pages 73–81.

Anders Søgaard. 2012. Unsupervised dependency parsing without training. *Natural Language Engineering* 18(02):187–203.

Karen Sparck Jones. 1972. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation* 28(1):11–21.

Michael Spranger and Luc Steels. 2015. Co-acquisition of syntax and semantics - an investigation in spatial language. In Qiang Yang and Michael Wooldridge, editors, *IJCAI'15*, AAAI Press, Palo Alto, US, pages 1909–1905.

Luc Steels. 2001. Language Games for Autonomous Robots. *Intelligent Systems, IEEE* 16(5):16–22.

Luc Steels and Frederic Kaplan. 2002. Aibo's First Words: The Social Learning of Language and Meaning. *Evolution of Communication* 4(1):3–32.

Stefanie A Tellex, Thomas Fleming Kollar, Steven R Dickerson, Matthew R Walter, Ashis Banerjee, Seth Teller, and Nicholas Roy. 2011. Understanding natural language commands for robotic navigation and mobile manipulation .