# Predicting Physical World Destinations for Commands Given to Self-Driving Cars

**Dusan Grujicic**[*2], **Thierry Deruyttere**[*1], **Marie-Francine Moens**[1] and **Matthew Blaschko**[2]

[1] Department of Computer Science, KU Leuven, Belgium
[2] Department of Electrical Engineering, KU Leuven, Belgium
{dusan.grujicic, thierry.deruyttere, sien.moens, matthew.blaschko}@kuleuven.be

**Abstract**

In recent years, we have seen significant steps taken in the development of self-driving cars. Multiple companies are starting to roll out impressive systems that work in a variety of settings. These systems can sometimes give the impression that full self-driving is just around the corner and that we would soon build cars without even a steering wheel. The increase in the level of autonomy and control given to an AI provides an opportunity for new modes of human-vehicle interaction. However, surveys have shown that giving more control to an AI in self-driving cars is accompanied by a degree of uneasiness by passengers. In an attempt to alleviate this issue, recent works have taken a natural language-oriented approach by allowing the passenger to give commands that refer to specific objects in the visual scene. Nevertheless, this is only half the task as the car should also understand the physical destination of the command, which is what we focus on in this paper. We propose an extension in which we annotate the 3D destination that the car needs to reach after executing the given command and evaluate multiple different baselines on predicting this destination location. Additionally, we introduce a model that outperforms the prior works adapted for this particular setting.

## 1 Introduction

Many companies are in the race to be the first to develop fully self-driving cars. It is expected that once this technology matures, manufacturers might entirely remove the steering wheel. While some enthusiasts are eagerly waiting for this day, surveys (Othman 2021; Deruyttere, Milewski, and Moens 2021; Schoettle and Sivak 2014) have indicated that an average person is wary of relinquishing physical control of the car. However, it was found that the notion of being able to give spoken commands that can change the behavior of the vehicle tends to make people much more at ease (Deruyttere, Milewski, and Moens 2021).

In recent years, researchers have investigated systems where passengers can give commands to self-driving cars. For instance, (Vasudevan, Dai, and Van Gool 2021; Chen et al. 2019) consider navigational commands such as "Take the first left and at the red building turn right. Afterwards, drive to the white building". On the other hand, Talk2Car (Deruyttere et al. 2019) considers commands that abruptly change the driving dynamics in the imminent future, such as "Let me out near my friend with the red shirt on the left". The two former datasets feature graph-based navigation at a city level, where each node represents an individual street scene. The latter dataset takes a lower level approach and focuses on predicting objects referred to by the command within an individual street scene, without addressing the navigational aspects of executing the command.

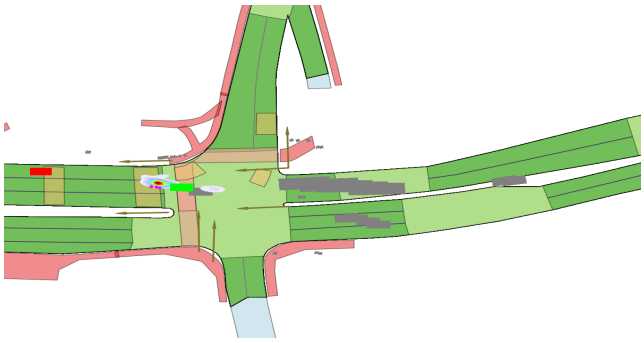
Taking the same low-level approach, we focus on the environment within the observable vicinity of the current location and extend the Talk2Car dataset with the 3D physical destination for each given command. This extension is interesting as it requires models to interpret spatial language in a 3D visual context. The works of (Dendorfer, Osep, and Leal-Taixé 2020; Mangalam et al. 2020) suggest that it is also beneficial to first predict the end goal of a path before starting to navigate. Therefore, we focus on predicting the end destination given the passenger command (an example is given in Figure 1), after which, in a practical setting, one of the many already existing systems could be tasked with safely navigating to it within the dynamic environment (Messaoud et al. 2020).

Additionally, indicating the destination as understood from the command offers additional visual insight to the passenger in their interaction with the car. It may also alleviate a degree of uneasiness that comes with the absence of direct physical control. To the best of our knowledge, our dataset is the first of its kind where a model needs to predict the physical, 3D destination of the self-driving car of the passenger that has given a command. We will refer to the car in which the passenger is as ego car from now on. In this paper, we investigate if existing models are adequate enough to perform well on this task. In addition to this, we propose a new model called **PDPC** for this destination task which outperforms our evaluated baselines. Our proposed model uses a feature pyramid network to predict distributions at different feature levels and finally aggregates them into a distribution mixture. The contributions in this paper are the following:

- We introduce an expansion, **Talk2Car-Destination**, for the Talk2Car dataset (Deruyttere et al. 2019) where we annotate the **possible destinations in the physical world**

[*]These authors contributed equally.
Copyright © 2022, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

(a) Frontal View



(b) Top-down View

Figure 1: Visualization of the predictions made for the task at hand. The issued command here is "Oh no I'm on the wrong lane, change lanes so I drive behind the car in front". On the left, we have visualized the predicted destination of the command as a heatmap in the physical world and the referred object with a green bounding box. On the right, we have the prediction on a top-down view of the scene (under the purple dots). The red rectangle is our location, the green rectangle is the predicted object referred to by the command, and the gray rectangles are the other detected objects. The purple dots are the ground truth destinations. Prediction made with our proposed model from Sect 4.4.

for a given command, i.e., where the ego car should be after it has executed the command.

- We evaluate multiple baselines on our expansion and show that the dataset is challenging.
- We propose a novel method called **Physical Destination Predictor for Commands** that achieves state-of-the-art results compared to our evaluated baselines and outperforms them by as much as 32% in certain measures.

The Talk2Car-Destination dataset is available at https://github.com/ThierryDeruyttere/Talk2Car-Destination. An extended version of the paper is available on arXiv (Grujicic et al. 2021).

## 2 Related Work

In this section, we describe previous works on future prediction relevant for the destination prediction task featured in the Talk2Car-Destination dataset. Moreover, our task requires to detect the object referred to by the command and the 3D locations of objects in the physical world, and hence, we also list the relevant works on query understanding and 3D object detection.

### 2.1 Future Prediction

Future prediction features a broad range of different tasks, such as future image segmentation prediction (Luc et al. 2017), future action prediction (Rodriguez, Fernando, and Li 2018), to future frame prediction (Liu et al. 2018). The task of predicting the future location of a moving object in the physical world featured in the works of (Lee et al. 2017; Yagi et al. 2018; Makansi et al. 2019) is conceptually related to our task of predicting the car's end location after executing the command. As the future is often uncertain, it is beneficial that the model predicts multimodal distributions of the destination location. Makansi et al. (2019) tackle this task with an approach based on Mixture Density Networks (MDN). However, they observe that the Mixture Density Networks are prone to experiencing mode collapse. To overcome this, they propose an adaptation to the Winner-Takes-All loss (Guzman-Rivera, Batra, and Kohli 2012) where they only update the top-$k$ hypotheses while decreasing the value of $k$ over time. Subsequently, they fit the mixture distribution to the estimated hypotheses through soft assignments. A problem with the aforementioned solution is the need for defining the number of mixture components a priori. To overcome this limitation, RegFlow (Zieba et al. 2020) avoids making any assumptions on the underlying distribution by using Continuous Normal Flows (CNF) (Chen et al. 2018), thus decreasing the number of distribution parameters.

Another relevant future prediction task is trajectory prediction, where the models are tasked with jointly predicting the future location of the object and the path to it (Narayanan et al. 2021; Liang et al. 2020; Mohamed et al. 2020). Dendorfer, Osep, and Leal-Taixé (2020) propose GoalGAN, which first encodes the trajectory of pedestrians with an LSTM before passing the encoded trajectory to an encoder-decoder architecture that takes the top-down view of the scene as input. The output of the encoder-decoder is a probability map of the pedestrian's future locations.This information is passed to the next module that predicts the possible path towards this goal. Their model uses a GAN to predict if the generated trajectories come from the same distribution as the ground truth trajectories. Another example is PECNet (Mangalam et al. 2020) where the future destination is modeled using a CVAE (Lee et al. 2017), by first encoding the past trajectory of the object and passing it to the CVAE, without the use of visual features of the image. Then, along with the predicted destination, the model predicts the trajectory by applying social pooling (Alahi et al. 2016). This paper utilizes the components from GoalGAN, PECNet, and RegFlow as the baselines on our new dataset. We opt for RegFlow over (Makansi et al. 2019) as the training code of their model was not available.

### 2.2 Query Understanding

Our task requires the model to find the referred object of the command (Visual Grounding) and predict the car's end location after executing the command (Spatial Language Interpretation). In Visual Grounding (Hu et al. 2016; Deng et al. 2018; Akbari et al. 2019), the model receives a query, an image, and potentially a set of objects. The model is asked to predict the object in the image referred to by the query. A popular approach for joint reasoning over the query and vision is the use of Transformers (Dai et al. 2020; Kamath et al. 2021; Du et al. 2021). Multi-step reasoning models (Hudson and Manning 2018; Deng et al. 2018) and modular models (Yu et al. 2018) that reason over image and/or objects have also been popular. In this paper we use a similar model to (Rufus et al. 2020).

With regards to grounding text to physical locations, Lourentzou, Morales, and Zhai (2017) focus on predicting physical geographic origins of Twitter posts, while Grujicic et al. (2020) localize medical text referring to anatomical concepts to their corresponding physical locations in the human body. Finally, (Collell, Deruyttere, and Moens 2021) have also worked on understanding the implicit spatial relationships of queries and objects.

Focusing on the domain of autonomous-driving, the work of Sriram et al. (2019) presents a dataset of natural language instructions and trajectories in a synthetic environment. Our dataset, on the other hand, features command and destination annotation in the natural, non-simulated environments. In a work concurrent to ours, Rufus et al. (2021) introduce a dataset built on top of the Talk2Car which features destination annotations in the frontal camera view. In contrast, our work focuses on destinations in the car's 3D environment, in which the physical layout and distances can be considered.

### 2.3 3D Object Detection

The locations of other objects in the scene give important cues for predicting the final destination in the physical world. Therefore, aside from predicting the referred object, an essential step in our task is the accurate prediction of 3D bounding boxes of the objects visible in the frontal view. A popular approach to this task is to use LIDAR/RADAR point clouds (Zhou and Tuzel 2018; Lang et al. 2019; Engelcke et al. 2017; Zheng et al. 2021). For instance, Yin, Zhou, and Krähenbühl (2021) use LIDAR to predict 3D objects by first predicting the center of each object in a top-down point cloud view. Afterwards, the model regresses the centers to the object's 3D size, orientation, and velocity. In the second stage, it refines the predicted attributes. Lang et al. (2019) perform 3D object detection by aggregating voxels into vertical columns, also called pillars. Afterwards, 2D convolutions are applied on said pillars to predict 3D objects.

In addition to using point clouds to predict 3D bounding boxes, there are approaches that rely on image data alone. This is known as monocular 3D object detection (Chen et al. 2015; Roddick, Kendall, and Cipolla 2018). An example of such a method is FCOS3D (Wang et al. 2021), which is very similar to the CenterNet (Zhou, Wang, and Krähenbühl 2019) architecture. It predicts 3D bounding boxes by creating a feature pyramid network of an image, and then, at each level and cell of the feature map, it predicts the 3D attributes of objects. In this work, we use FCOS3D to predict 3D object locations.

## 3 Dataset

In this paper, we extend the Talk2Car dataset (Deruyttere et al. 2019) which is built on top of the nuScenes dataset (Caesar et al. 2020) and features driving scenes from Boston (right-hand traffic) and Singapore (left-hand traffic) in different lighting (night vs day) or weather conditions (sun vs rain). In addition to the 360 degrees camera view, it also contains LIDAR/RADAR point clouds. Talk2Car adds unrestricted natural language commands that refer to a specific object in a street setting, and consists of 8349, 1163, and 2447 samples in the training, validation, and test sets. We extend the Talk2Car dataset by having three annotators per each command indicate the physical end position of the car after executing the command in the top-down view (Figure 3) together with the true intent of the command i.e., the intent of the command "park next to the tree" is "park". For the true intent of the command, the annotators received a set of predefined options to choose from and the final annotation was determined in a majority vote among three annotators. In case of no majority, additional annotators were asked to indicate the intent until a majority was reached. The annotation process was performed using a custom annotation interface created with EasyTurk (Krishna 2019), hosted on Amazon Mechanical Turk.

In addition to the annotation, we also normalized the data for easier training. We first rotated all top-down views (as seen in Figure 3(b)) in such a way that the ego car is always facing directly right. The resolution of the frontal camera view images is $1600 \times 900$. We normalize the sizes of the top-down views such that they correspond to physical map regions of $120 \times 80$ meters in size, with the ego car located seven meters from the left boundary and halfway along the height of the top-down view. The resulting resolution of the top-down views is $1200 \times 800$, i.e., distance of ten pixels in the top-down view corresponds to the distance of one meter.

We removed the commands that referred to objects outside such a map patch, which was the case in 0.51% of the samples. This process resulted in 8301, 1159, and 2439 samples in the train, validation, and test set. The average distance from the center of the ego car to the destination is 26.54 meter. In Figure 2 we display a heatmap of the destinations. Each sample consists of a command, frontal and top-down view, ground-truth 3D object annotations from the NuScenes dataset, three different destination annotations, as well as the annotated intent of the command.

We dub this extension **Talk2Car-Destination**. The statistics from the original Talk2Car dataset (Deruyttere et al. 2019) regarding the driving locations, weather conditions, times of day etc. apply to Talk2Car-Destination as well.

## 4 Methods

We formally define the destination prediction task as follows: given an image of the frontal view of the car $\mathbf{I}$, a command $q$ and the representation of the top-down view con-

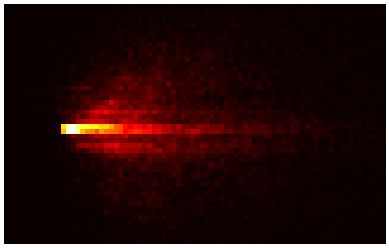


Figure 2: Destination distribution of the Talk2Car-Destination dataset. This heatmap represents the destinations 50 meters in front of the ego car and 20 meters to the right and left of the ego car.

taining the road layout $\mathbf{L}$, the model is asked to predict a distribution $\mathbf{P}$ of the destination the car needs to reach in order to execute the command. We first extract the locations of the objects around the ego car using a 3D object detector. As we later demonstrate in Sect. 6.3, the location of the object indicated by the command is highly indicative of the the final destination. Therefore, we also formulate the task of referred object detection, where given the command and the set of object proposals obtained from a 3D object detector, we predict the object referred by the command. Figure 4 presents the flowchart of our approach. Focusing on the destination prediction, we decouple it from the already established task of referred object detection on Talk2Car (Deruyttere et al. 2019). The predicted referred object and the 3D object proposals are subsequently provided as input to the destination prediction model during the training and evaluation phases. We do not utilize ground-truth annotation from either NuScenes or Talk2Car during training, as they would be unavailable in a deployed setting. In the following subsections, we describe our main building blocks. We use the following notation: vectors are represented as $\boldsymbol{a}$ and their size $\mathbb{R}^{d1}$ where $d1$ is the size. Matrices are denoted as $\boldsymbol{M}$ and their size $\mathbb{R}^{d1 \times d2}$ where $d1$ and $d2$ are the size of the first and second dimensions. The layers of multi-layer perceptrons (MLP) are indicated between square brackets, e.g. $[\boldsymbol{W}^{d1 \times d2}, \boldsymbol{W}^{d2 \times d3}]$ denotes a two layer MLP.

### 4.1 3D Object Detector

The availability of LIDAR in the Talk2Car-Destination points towards the use of LIDAR-based 3D object detectors, given the effectiveness such an approach has demonstrated in previous works. However, in our case, we require accurate bounding boxes for the visible objects in the frontal view of the car, which are the focus of the given commands. From preliminary experiments, we found the use of LIDAR-based methods to be inadequate for our use case. The reason is twofold. First, we observed that when projecting the predictions of the LIDAR-based method to the frontal image, there was often a substantial shift between the projected bounding boxes and the objects. This is likely due to the different speeds at which the camera and LIDAR sensors capture data, as well as the positioning offset between the camera and the LIDAR sensor, causing both temporal and spatial parallax. Second, the objects referred to by the commands tend to be at greater distances from the ego car. We find that after 70 meters, the LIDAR-based object detector abruptly becomes unable to detect objects accurately. In contrast, the performance drop of monocular-based detectors at larger distances tends to be less dramatic.

We performed a comparison between FCOS3D (Wang et al. 2021), and CenterPoint (Yin, Zhou, and Krähenbühl 2021), where we evaluated the average distance between the ground-truth location of the referred object and the closest 3D object proposal. We found that FCOS3D achieves an average discrepancy of 1.50 meters, which is a significant improvement over the 2.64 meters of CenterPoint as the latter does not output predictions at distances greater 70 meters.

We therefore opt for a monocular 3D object detector, which we found to be more suitable for our task in the light of aforementioned limitations of LIDAR-based approaches. More specifically, we use FCOS3D as, at the time of writing, it is one of the highest performing vision-only 3D object detectors on nuScenes with publicly available code (Contributors 2020).

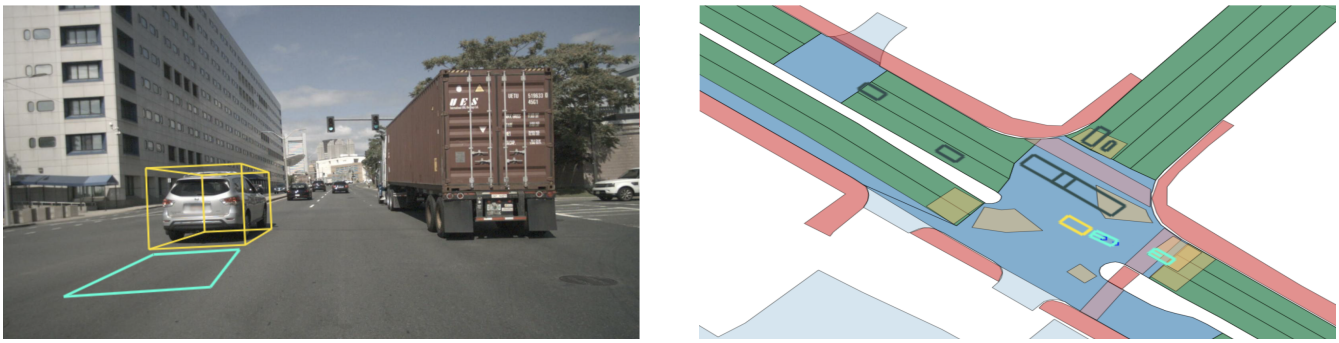
### 4.2 Referred Object Detector

From our data and preliminary experiments, we observe that the object referred to by the command is often in relation to the destination of the car, and that the information on the referred object location is central for predicting the destination of the command. For instance, to execute the command "Park next to the man on the bench", the system ought to know where the "man on the bench" is before predicting the parking location. We train a referred object detector to indicate which object the command relates to and feed the predicted referred object and other detected objects as the input to the destination prediction model.

We notice that the use of the features from the object detector was left unexplored by the previous works on Talk2Car, all of which relied on using pre-trained image feature extractors and then optionally fine-tuning them. In the vein of (Anderson et al. 2018), we evaluate the benefits of directly using the object features $\mathbb{R}^{1536}$ of FCOS3D as local object representations in the referred object detection task by feeding these features as input to the model used in the work of (Rufus et al. 2020). The model uses a Sentence-BERT (Reimers and Gurevych 2019) to create a $\mathbb{R}^{768}$ command embedding, which is then passed through the following MLP: $[\boldsymbol{W}^{768 \times 1024}$, ReLU, $\boldsymbol{W}^{1024 \times 1024}]$. The extracted visual features from FCOS3D of size $\mathbb{R}^{1538}$, which contain the concatenated outputs of all prediction heads, extracted at the activation map location corresponding to the predicted bounding box, are passed through the following MLP: $[\boldsymbol{W}^{1538 \times 1024}$, ReLU, $\boldsymbol{W}^{1024 \times 1024}]$. Finally, we compute the dot product between command and the visual features of each object and construct a probability distribution over object proposals by performing the softmax operation on the products.

### 4.3 Layout Encoding

We construct a top-down layout representation ($L^*$), leveraging the road information, the ego car position, the top-down view projections of all FCOS3D detected objects in



(a) Frontal View (b) Interactive Top-down View

Figure 3: An example from our annotation tool that was made with EasyTurk (Krishna 2019), given the command *"get behind this silver SUV"* and the Talk2Car object that it refers to (indicated with a yellow bounding box), the annotators were asked to indicate the destination in the interactive top-down view. The annotated destination is the cyan rectangle visible in the frontal view and the cyan rectangle behind the yellow rectangle in the top-down view.

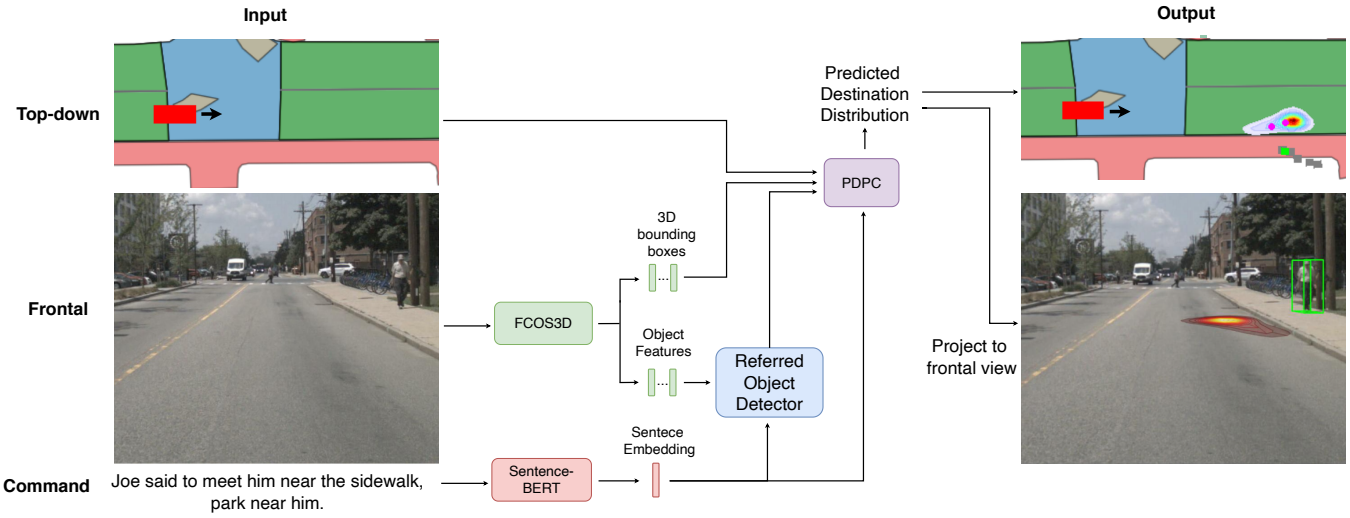



Figure 4: The above image depicts the flow of the architecture used in this paper. As input, we require a top-down image of the road, a car's frontal image, and a (written) command. The frontal image is passed through a monocular 3D object detector (i.e., FCOS3D) to extract 3D bounding boxes and object features. The command is passed through a sentence encoder (i.e., Sentence-BERT). The embedded sentence and object features are then passed through a Referred Object Predictor (Sect. 4.2) to predict the object referred by the command. Afterward, the predicted referred object, together with the sentence embedding, the 3D bounding boxes, and the top-down image are passed through our proposed destination predictor (Sect. 4.4) to finally predict a destination distribution on the top-down image. This distribution can also be projected to the frontal view as seen in the image. The red car on the top-down image represents the ego car. The black arrow indicates the driving direction. The green rectangle is the predicted referred object. Gray objects are other detected objects.

3D, as well as the prediction of the object referred to by the command. The layout representation takes a form of a 3D tensor, with 3 channels reserved for the road information (an RGB image representing the lanes and the road layout), 1 channel reserved for the ego car (a binary grayscale image with a white polygon representing the car at the position of the ego car), 1 channel reserved for the referred object bounding box, and one channel per each object class, as detected by the 3D object detector, where each channel contains a binary grayscale image with white polygons at the positions of the top-down view projections of the detected 3D bounding boxes of objects of it corresponding class.

### 4.4 Destination Predictor

In this subsection, we describe our proposed destination predictor which we dub **Physical Destination Predictor for Commands (PDPC)**. We opt for a fully convolutional and spatial-based approach, where the destination distribution is modeled as a mixture distribution, where a distribution mixture component is predicted at every location in the extracted spatial feature maps.

Our method receives a command and a top-down layout augmented with the detected object locations and the referred object (Sect. 4.3) in the current visual scene as input. We pass the layout through a Feature Pyramid Network (FPN) to extract feature maps $\mathbb{R}^{C\times W_i\times H_i}$ at four different scales with $C$, $W_i$, and $H_i$ being the number of channels (256 in our case), width and height of a feature map at scale $i$, respectively. The different scales share a series of fully convolutional blocks that process the feature maps into a set of spatial features at each grid location. We use stride of 1, kernel size of 3 and padding of 1 in order to preserve the size of the original feature map. Then, a multi-layer perceptron $[\boldsymbol{W}^{768\times 512}, ReLU, \boldsymbol{W}^{512\times C}]$ is used to project the command embedding to the same size as the channel dimension of the convolutional block. Afterwards, a dot product between the projected command and the channel dimension of the feature maps is performed to produce a $\mathbb{R}^{1\times W_s\times H_s}$ tensor. A softmax operation is then used to get a probability distribution over the spatial cells to produce a weights tensor that indicates which cells align more with the command. Finally, we multiply the feature map $\mathbb{R}^{C\times W_i\times H_i}$ with this weights tensor. Another shared series of fully convolutional blocks is applied on the attended feature maps at each scale, after which the task specific convolutional heads, also shared across different scales, are used to predict the offset between the grid location and the distribution mean, as well as the standard deviation and mixture weight for each Gaussian mixture component. We aggregate the mixture components from each location at each scale level into one large density mixture with $\sum_{i=1}^{S} W_i \times H_i$ components, which we fit to the destination samples. The distribution of the destination $\boldsymbol{y}$, given the layout encoding $\boldsymbol{L}^*$ (Sect 4.3) and command $\boldsymbol{c}$ is therefore represented with a Gaussian mixture:

$$p(\boldsymbol{y}|\boldsymbol{L}^*, \boldsymbol{c}) = \sum_{i=1}^{S} \sum_{w=1,h=1}^{W_i,H_i} \pi_{iwh}\phi\left(\boldsymbol{y}|\boldsymbol{\mu}_{iwh}, s_i\boldsymbol{\sigma}_{iwh}\right) \quad (1)$$

where $S$ is the number of resolution scales, while $H_i$ and $W_i$ represent the height and the width of the feature map at the $i$-th scale. The Gaussian mixture component $\phi$ at the location $h, w$ in the $i$-th scale is parametrized by its mean $\boldsymbol{\mu}_{iwh} \in \mathbb{R}^2$ and standard deviation $\boldsymbol{\sigma}_{iwh} \in \mathbb{R}^2$, while $\pi_{ihw}$ represents its mixture weight. We use softmax to normalize the mixture weights such that $\sum_{i=1}^{S}\sum_{w=1,h=1}^{W_i,H_i} \pi_{iwh} = 1$. For each location in low resolution scale grids $[w, h]$, we keep track of the corresponding spatial location in the input layout, $\boldsymbol{l}_{ihw} = [w', h']$, where $w'$ and $h'$ are computed as $w' = wk_i + \left\lfloor \frac{k_i}{2} \right\rfloor$ and $h' = hk_i + \left\lfloor \frac{k_i}{2} \right\rfloor$ and $k_i$ represents the down-sampling rate at scale $i$. At each location, the network directly predicts the mixture weights and the standard deviation, as well as the offset $\boldsymbol{o}_{ihw}$, after which the mean is computed as $\boldsymbol{\mu}_{ihw} = \boldsymbol{o}_{ihw} + \boldsymbol{l}_{ihw}$. Predicting the offset from the corresponding physical location, as opposed to directly predicting the mean, allows for a regularized model to predict small offsets within the immediate neighborhood of each grid cell. At each resolution scale, the predicted standard deviation is multiplied by a learnable parameter $s_i$, allowing the network to adjust the variance of the Gaussian components for each resolution scale.

Our model uses Negative Log Likelihood (NLL) to minimize the log probability of the ground truth destinations under the predicted Gaussian Mixture, or in other words we minimize:

$$\mathcal{L}_j = -\log\left(p(\boldsymbol{y}_j|\boldsymbol{L}, \boldsymbol{c})\right) \quad (2)$$

where for $N$ different destinations provided by different annotators we minimize $\frac{1}{N}\sum_{j=1}^{N}\mathcal{L}_j$. The benefit of our model over (Makansi et al. 2019) is that we do not require to a priori define how many components we want. Our model predicts a component in each cell in the spatial grid and can learn to give certain components very low weights, and effectively ignore them, if needed.

## 5 Experiments

In this section we explain the performed experiments, the evaluation measures and the baselines.

### 5.1 3D Object Detector

We train FCOS3D (Wang et al. 2021) only on the Talk2Car training scenes to predict bounding boxes as they do not overlap with the scenes from the Talk2Car validation or test sets. We use the default parameters provided by the authors.

### 5.2 Referred Object Detector

Our referred object detector uses a Sentence-BERT (Reimers and Gurevych 2019) to compute the command embedding, whose parameters remain fixed during training.

We evaluate this model by measuring the Intersection over Union (IoU) between between the boxes of the predicted and ground truth objects. If IoU $> 0.5$, we consider the predicted object to be correct. We refer to this as $IoU_{0.5}$. The IoU is defined as:

$$IoU = \frac{\text{Area of Overlap between two boxes}}{\text{Area of Union of the two boxes}}. \quad (3)$$

| Model | $\text{IoU}_{0.5}$ | # Params (M) |
|---|---|---|
| (Dai et al. 2020) | **0.710** | 683.80 |
| (Luo et al. 2020) | 0.691 | 194.97 |
| (Rufus et al. 2020) | 0.686 | 366.50 |
| Sentence-BERT+FCOS3D | 0.701 | **166.29** |

Table 1: The $\text{IoU}_{0.5}$ of different models on the Talk2Car test set. The first three models are the previous state-of-the-art models for reference. The "Sentence-BERT+FCOS3D" model is the referred object predictor described in Sect. 4.2 and is based on (Rufus et al. 2020) but uses the extracted object features from FCOS3D.

### 5.3 Destination Prediction Measures

To evaluate the performance of the destination prediction models on our dataset, we use the following three measures:

- Average Displacement Error (**ADE**): Sample average of the average distance between the samples of the predicted destination distribution from the closest ground truth destination. Measured in meters [m].
- Median Displacement Error (**MDE**): Sample median of the average distance between the samples of the predicted destination distribution from the closest ground truth destination. Measured in meters [m].
- Prediction Accuracy with threshold $k$ meters ($\mathbf{PA}_k$): measures the rate of predicting the destination within the radius of $k$ meters around the ground truth destination.

### 5.4 Command Encoding

The command is processed by using the pre-trained Sentence-BERT (Reimers and Gurevych 2019), where the input command is tokenized and fed through the model, and the command representation is obtained by computing the mean of the output token representations. The Sentence-BERT is trained on the combination of SNLI (Bowman et al. 2015) and MultiNLI datasets (Williams, Nangia, and Bowman 2017) to perform the Natural Language Inference (NLI) task of predicting semantic labels for annotated sentence pairs, and has been shown to perform well on a variety of downstream tasks that require high quality sentence representations in a wide range of semantic domains.

### 5.5 Baselines

To evaluate Talk2Car-Destination and our proposed model, we implemented or adapted the following baselines.

**SinglePoint** The layout tensor encoded using ResNet18 (He et al. 2016) to a vector of $\mathbb{R}^{1024}$, concatenated with the sentence-BERT command embedding of $\mathbb{R}^{768}$ is fed through a multi-layer perceptron (MLP) that regresses the input to two-dimensional destinations.

**UnimodalNormal** Predicts the destination distribution given the layout tensor and the command after encoding them as in SinglePoint. We model the distribution as a bivariate Gaussian, by predicting the distribution mean and covariance matrix. We minimize the negative log-likelihood (NLL) of the target destinations under the predicted distribution.

**MDN** Predicts destination distribution similarly to UnimodalNormal, except that we model distribution as a mixture of bi-variate Gaussian distributions. For each component, we predict the mean, covariance matrix and the mixture weight. We minimize the NLL of the target destinations under the predicted Gaussian mixture.

**NonParametric** The distribution of the destinations is modeled as a histogram over possible locations in a grid, where each grid cell corresponds to a physical location in the top-down view. We minimize the cross-entropy between the predicted distribution over grid cells. The locations in the grid where the cells correspond to ground truth destination locations have the value 1.0 and 0.0 otherwise. We treat the rows and columns of the grid independently to keep the output dimensionality low.

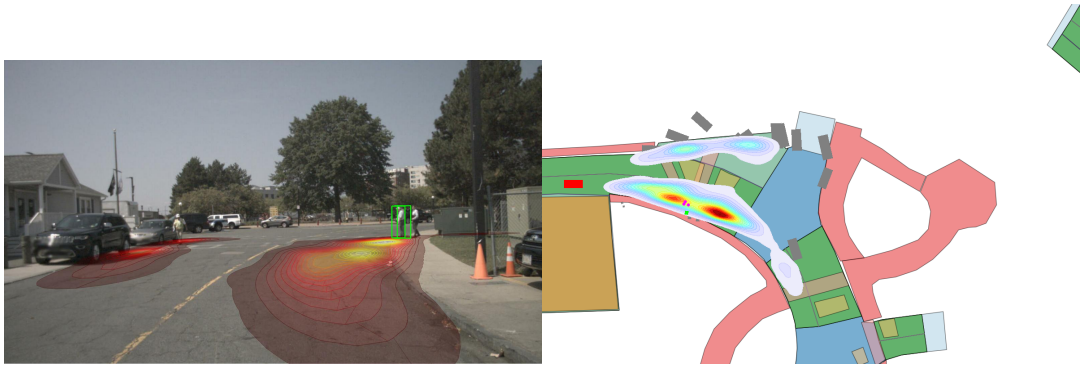
**Adapted RegFlow (Zieba et al. 2020)** FlowNet (Dosovitskiy et al. 2015) is used to encode the top-down view in $\mathbb{R}^{1024}$. Next, we pass the sentence-BERT command embedding through a linear layer of $\boldsymbol{W}^{768\times768}$ and then concatenate it to the layout encoding. The remainder of the model is unchanged and a hypernetwork is used to train a continuous normalising flow (CNF) model to output a distribution. This model is trained by minimizing the NLL of the CNF.

**Endpoint VAE (Mangalam et al. 2020)** We adapt the endpoint prediction components of PECNet, which is a trajectory prediction model that takes a goal-conditioned approach, where the model first predicts the trajectory endpoint and, subsequently, the trajectory. We utilize the Endpoint VAE, which is the model component that performs the endpoint prediction, and adapt it for our task of destination prediction. In PECNet, the Endpoint VAE encodes the past trajectory and the ground truth endpoint into a latent destination distribution during training. The decoder is then fed a sample drawn from the latent distribution, and the past trajectory representation outputs predicted endpoints. During inference, the latent samples are drawn from a zero-mean isotropic Gaussian. We adapt this approach to our setting by replacing past trajectory encoding with the representation of the spatial layout from the ResNet-18 and the command embedding obtained from the Sentence-BERT.

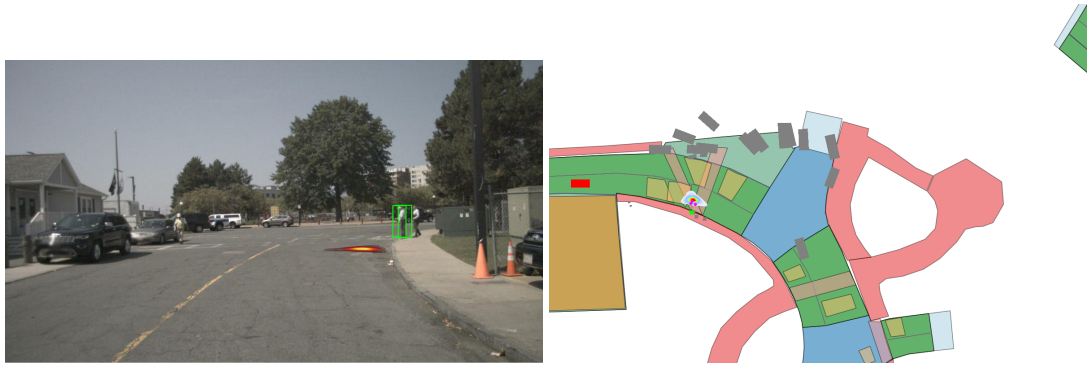
**Adapted GoalGAN (Dendorfer, Osep, and Leal-Taixé 2020)** GoalGAN has a RoutingModule to predict the path, a GoalModule to predict the destination and a MotionEncoder to predict the last locations of the objects. We remove the MotionEncoder as we do not require it. Then, our top-down layout tensor is processed in an encoder-decoder architecture with as output a probability map over the image. After the encoder stage, we introduce the command to the encoded feature map by concatenating it along the channel dimension, after which a convolutional layer is used to project it to a lower dimension. The RoutingModule is kept as-is but we set the length of the path to one instead of $N$ as to predict the destination. We keep the GAN as-is.

| Method | ADE [m] | MDE [m] | $PA_2$[%] | $PA_4$ [%] |
|---|---|---|---|---|
| Random Point | $43.98 \pm 0.12$ | 44.24 | $0.28 \pm 0.00$ | $0.97 \pm 0.00$ |
| Random Road Point | $37.38 \pm 0.17$ | 37.35 | $0.81 \pm 0.01$ | $2.60 \pm 0.03$ |
| Pick Ego Car | $25.62 \pm 0.32$ | 21.61 | $0.00 \pm 0.00$ | $0.00 \pm 0.00$ |
| Random Object | $28.02 \pm 0.23$ | 26.77 | $1.11 \pm 0.06$ | $4.62 \pm 0.14$ |
| Pick Referred Object | $9.04 \pm 0.18$ | 6.07 | $6.60 \pm 0.50$ | $27.96 \pm 0.91$ |
| SinglePoint | $8.15 \pm 0.34$ | 5.52 | $13.41 \pm 1.38$ | $35.88 \pm 1.94$ |
| NonParametric | $9.65 \pm 0.36$ | 6.72 | $12.81 \pm 0.64$ | $31.22 \pm 1.20$ |
| UnimodalNormal | $8.16 \pm 0.32$ | 5.61 | $17.61 \pm 0.92$ | $38.86 \pm 1.44$ |
| MDN | $8.08 \pm 0.32$ | 5.30 | $16.82 \pm 1.34$ | $38.42 \pm 1.48$ |
| Adapted GoalGAN | $8.65 \pm 0.40$ | 5.31 | $22.60 \pm 1.20$ | $43.09 \pm 1.56$ |
| Endpoint VAE | $\mathbf{7.84 \pm 0.34}$ | 5.22 | $16.38 \pm 1.34$ | $38.83 \pm 1.86$ |
| Adapted RegFlow | $17.79 \pm 0.36$ | 15.89 | $13.01 \pm 0.70$ | $32.23 \pm 0.70$ |
| PDPC - Base (Ours) | $8.21 \pm 0.36$ | 5.05 | $28.45 \pm 1.28$ | $47.56 \pm 1.52$ |
| PDPC - Top-64 Components (Ours) | $7.96 \pm 0.36$ | 4.61 | $28.97 \pm 1.30$ | $48.42 \pm 1.56$ |
| PDPC - Top-32 Components (Ours) | $\mathbf{7.82 \pm 0.38}$ | **4.39** | $\mathbf{29.88 \pm 1.36}$ | $\mathbf{49.51 \pm 1.64}$ |
| PDPC Base - NoRef (Ours) | $13.95 \pm 0.40$ | 11.15 | $11.25 \pm 0.52$ | $25.19 \pm 0.94$ |

Table 2: ADE, MDE, PA computed for 2m and 4m thresholds ($PA_2$ and $PA_4$, respectively). The results of our model are denoted with PDPC. The error bars represent 95% confidence intervals.



(a) PDPC - No referred object information - Frontal (b) PDPC - No referred object information - Top-down



(c) PDPC - With referred object - Frontal (d) PDPC - With referred object - Top-down

Figure 5: The issued command was: "The man standing closest to the road is coming with us. Park next to him". This image shows the heatmaps of the model that does not use the information of the referred object prediction (top row) versus the model that uses this information (bottom row). The red car on the top-down view is the ego car. The purple dots are the ground truth destination. The gray boxes represent other detected objects. The green rectangle in the top-down and in the frontal view indicates the referred object. In the top row, the model outputs heatmaps at different locations with increasing probability near objects of the same type as the referred object. In contrast, the model in the bottom row can pinpoint where it needs to output its heatmap as it has information about the referred object.

# 6 Results

## 6.1 3D Object Detector

FCOS3D achieves a mean Average Precision (mAP) of 30.73 on the Talk2Car test set scenes. Additionally, we evaluate the quality of predictions on the referred object detection task. We project the top-32 confidence scoring predictions to the frontal view and evaluate the rate at which at least one sample achieves an IoU $\geq 0.5$ with the ground truth referred object in the Talk2Car test set. We find that such a bounding box exists in 92% of the cases which is the same as with the bounding boxes of (Deruyttere et al. 2020), and therefore, no substantial increase in IoU is expected solely from improved bounding box predictions. The average distance between the 3D bounding box of the ground truth referred object and their 3D predicted bounding is 1.7m on the Talk2Car test set.

## 6.2 Referred Object Detectors

Table 1 showcases that our model outperforms two out of three state-of-the-art models on the referred object detection task in Talk2Car. Comparing to (Rufus et al. 2020), on which we based our detector, we observe a relative improvement of 2% in $\text{IoU}_{0.5}$ while having less than half the parameters.

## 6.3 Destination Prediction

In Table 2, we find that our model for destination prediction significantly outperforms all baselines in terms of $PA_2$, $PA_4$, and MDE. We also find that our model manages to generate 47.56% of its samples in a radius of 4m (roughly the length of a small passenger vehicle) from the nearest ground truth destination. We also see that the median distance of our base model is 5.05m which is 3.4% lower than our closest competitor. As evident from the discrepancy between the ADE and MDE scores, our model, as well as other baseline models such as the Endpoint VAE and GoalGAN, tend to produce large outliers more often than simpler baselines like UnimodalNormal and MDN. However, in an inference setting, our model can be easily adapted by using the top-K components in terms of their mixture weight magnitude, and re-normalizing the mixture weights. This significantly reduces the number of outliers, and when the number of mixture components is thus set to 32 during inference, our model outperforms all baselines in terms of the ADE as well. This is a very desireable feature of our model, which gives it a significant advantage over other competing models.

**Influence of Referred Object for Destination Prediction** From the last entry in Table 2, it can be seen that the referred object information is crucial for the model's ability to predict the destination accurately. The model PDPC (Base) - NoRef, which is the same as the PDPC (Base), with the distinction of not being provided with the predicted referred object as an additional input (normally provided to all baselines), achieves considerably inferior results. Figure 5 shows the difference between using or not using the referred object information during training. As the command refers to a person, the model without the referred object information, and for which the referred object was encoded as just another object of its particular class in the input layout tensor, tends to predict high probabilities near all persons on the top-down view. The model that does have the referred object information available, on the other hand, manages to pinpoint the correct destination in front of the man on the right, standing closest to the road. In general, we see that the model learns to adhere to very general traffic rules, keep to the road, and perform rudimentary commands without the referred object information. However, with the referred object information, it manages to pinpoint the correct destination.

# 7 Conclusion

In this paper, we propose a new challenging spatial language understanding extension to Talk2Car called Talk2Car-Destination, where a model needs to predict the ego car's destination in the physical 3D world after receiving a passenger command, i.e., "Park next to the man on the bench". To handle these types of commands, a referred object detector is also required. Our proposed referred object detector outperforms some of the state-of-the-art models on Talk2Car while using considerably fewer parameters. We also show that our Talk2Car-Destination task is not trivial by evaluating multiple baselines and (modified) existing state-of-the-art models. Hence, we believe this new dataset can be used as a benchmark for exciting future research. Additionally, we propose the Physical Destination Predictor for Commands (PDPC), which predicts distribution parameters at each feature map location at different resolution scales, and aggregates them into a distribution mixture. PDPC achieves a relative increase in performance of 32% for $PA_2$ and 15% on $PA_4$ and 16% in MDE. In this work, we focused on the top-down view alone. However, in future work, one could investigate incorporating the visual information from the camera view. Furthermore, one could also examine whether jointly predicting the referred object and destination in a multi-task end-to-end fashion can yield benefits to the individual tasks.


# Acknowledgements

This project was supported by Flanders AI, the MACCHINA project from KU Leuven (grant number C14/18/065) and ERC (grant H2020-ERC-2017-ADG 788506). We also wish to thank NVIDIA for providing us with two RTX Titan XPs.


# References


Akbari, H.; Karaman, S.; Bhargava, S.; Chen, B.; Vondrick, C.; and Chang, S.-F. 2019. Multi-level multimodal common semantic space for image-phrase grounding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 12476–12486.

Alahi, A.; Goel, K.; Ramanathan, V.; Robicquet, A.; Fei-Fei, L.; and Savarese, S. 2016. Social lstm: Human trajectory prediction in crowded spaces. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 961–971.

Anderson, P.; He, X.; Buehler, C.; Teney, D.; Johnson, M.; Gould, S.; and Zhang, L. 2018. Bottom-up and top-down attention for image captioning and visual question answering.

In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 6077–6086.

Bowman, S. R.; Angeli, G.; Potts, C.; and Manning, C. D. 2015. A large annotated corpus for learning natural language inference. *arXiv preprint arXiv:1508.05326*.

Caesar, H.; Bankiti, V.; Lang, A. H.; Vora, S.; Liong, V. E.; Xu, Q.; Krishnan, A.; Pan, Y.; Baldan, G.; and Beijbom, O. 2020. nuscenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 11621–11631.

Chen, H.; Suhr, A.; Misra, D.; Snavely, N.; and Artzi, Y. 2019. Touchdown: Natural language navigation and spatial reasoning in visual street environments. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 12538–12547.

Chen, R. T.; Rubanova, Y.; Bettencourt, J.; and Duvenaud, D. 2018. Neural ordinary differential equations. *arXiv preprint arXiv:1806.07366*.

Chen, X.; Kundu, K.; Zhu, Y.; Berneshawi, A. G.; Ma, H.; Fidler, S.; and Urtasun, R. 2015. 3d object proposals for accurate object class detection. In *Advances in Neural Information Processing Systems*, 424–432. Citeseer.

Collell, G.; Deruyttere, T.; and Moens, M.-F. 2021. Probing Spatial Clues: Canonical Spatial Templates for Object Relationship Understanding. *Ieee Access*, 9: 134298–134318.

Contributors, M. 2020. MMDetection3D: OpenMMLab next-generation platform for general 3D object detection. https://github.com/open-mmlab/mmdetection3d. Accessed: 2022-03-14.

Dai, H.; Luo, S.; Ding, Y.; and Shao, L. 2020. Commands for Autonomous Vehicles by Progressively Stacking Visual-Linguistic Representations. In *Proceedings of the 16th European Conference on Computer Vision, 2020. Commands for Autonomous Vehicles (C4AV) ECCV Workshop*.

Dendorfer, P.; Osep, A.; and Leal-Taixé, L. 2020. Goal-GAN: Multimodal Trajectory Prediction Based on Goal Position Estimation. In *Proceedings of the Asian Conference on Computer Vision*.

Deng, C.; Wu, Q.; Wu, Q.; Hu, F.; Lyu, F.; and Tan, M. 2018. Visual grounding via accumulated attention. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 7746–7755.

Deruyttere, T.; Milewski, V.; and Moens, M.-F. 2021. Giving commands to a self-driving car: How to deal with uncertain situations? *Engineering Applications of Artificial Intelligence*, 103: 104257.

Deruyttere, T.; Vandenhende, S.; Grujicic, D.; Liu, Y.; Van Gool, L.; Blaschko, M.; Tuytelaars, T.; and Moens, M.-F. 2020. Commands 4 autonomous vehicles (c4av) workshop summary. In *European Conference on Computer Vision*, 3–26. Springer.

Deruyttere, T.; Vandenhende, S.; Grujicic, D.; Van Gool, L.; and Moens, M. F. 2019. Talk2Car: Taking Control of Your Self-Driving Car. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2088–2098.

Dosovitskiy, A.; Fischer, P.; Ilg, E.; Hausser, P.; Hazirbas, C.; Golkov, V.; Van Der Smagt, P.; Cremers, D.; and Brox, T. 2015. Flownet: Learning optical flow with convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision*, 2758–2766.

Du, Y.; Fu, Z.; Liu, Q.; and Wang, Y. 2021. Visual Grounding with Transformers. *arXiv preprint arXiv:2105.04281*.

Engelcke, M.; Rao, D.; Wang, D. Z.; Tong, C. H.; and Posner, I. 2017. Vote3deep: Fast object detection in 3d point clouds using efficient convolutional neural networks. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 1355–1361. IEEE.

Grujicic, D.; Deruyttere, T.; Moens, M.; and Blaschko, M. B. 2021. Predicting Physical World Destinations for Commands Given to Self-Driving Cars. *CoRR*, abs/2112.05419.

Grujicic, D.; Radevski, G.; Tuytelaars, T.; and Blaschko, M. 2020. Learning to ground medical text in a 3D human atlas. In *Proceedings of the 24th Conference on Computational Natural Language Learning*, 302–312.

Guzman-Rivera, A.; Batra, D.; and Kohli, P. 2012. Multiple Choice Learning: Learning to Produce Multiple Structured Outputs. In *NIPS*, volume 1, 3. Citeseer.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 770–778.

Hu, R.; Xu, H.; Rohrbach, M.; Feng, J.; Saenko, K.; and Darrell, T. 2016. Natural language object retrieval. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 4555–4564.

Hudson, D. A.; and Manning, C. D. 2018. Compositional Attention Networks for Machine Reasoning. *CoRR*, abs/1803.03067.

Kamath, A.; Singh, M.; LeCun, Y.; Misra, I.; Synnaeve, G.; and Carion, N. 2021. MDETR–Modulated Detection for End-to-End Multi-Modal Understanding. *arXiv preprint arXiv:2104.12763*.

Krishna, R. 2019. EasyTurk: A Wrapper for Custom AMT Tasks. https://github.com/ranjaykrishna/easyturk. Accessed: 2022-03-14.

Lang, A. H.; Vora, S.; Caesar, H.; Zhou, L.; Yang, J.; and Beijbom, O. 2019. Pointpillars: Fast encoders for object detection from point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 12697–12705.

Lee, N.; Choi, W.; Vernaza, P.; Choy, C. B.; Torr, P. H.; and Chandraker, M. 2017. Desire: Distant future prediction in dynamic scenes with interacting agents. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 336–345.

Liang, J.; Jiang, L.; Murphy, K.; Yu, T.; and Hauptmann, A. 2020. The garden of forking paths: Towards multi-future trajectory prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 10508–10518.

Liu, W.; Sharma, A.; Camps, O.; and Sznaier, M. 2018. Dyan: A dynamical atoms-based network for video prediction. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 170–185.

Lourentzou, I.; Morales, A.; and Zhai, C. 2017. Text-based geolocation prediction of social media users with neural networks. In *2017 IEEE International Conference on Big Data (Big Data)*, 696–705. IEEE.

Luc, P.; Neverova, N.; Couprie, C.; Verbeek, J.; and LeCun, Y. 2017. Predicting deeper into the future of semantic segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, 648–657.

Luo, S.; Dai, H.; Shao, L.; and Ding, Y. 2020. C4AV: Learning Cross-Modal Representations from Transformers. In *European Conference on Computer Vision*, 33–38. Springer.

Makansi, O.; Ilg, E.; Cicek, O.; and Brox, T. 2019. Overcoming limitations of mixture density networks: A sampling and fitting framework for multimodal future prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 7144–7153.

Mangalam, K.; Girase, H.; Agarwal, S.; Lee, K.-H.; Adeli, E.; Malik, J.; and Gaidon, A. 2020. It is not the journey but the destination: Endpoint conditioned trajectory prediction. In *European Conference on Computer Vision*, 759–776. Springer.

Messaoud, K.; Deo, N.; Trivedi, M. M.; and Nashashibi, F. 2020. Trajectory prediction for autonomous driving based on multi-head attention with joint agent-map representation. *arXiv preprint arXiv:2005.02545*.

Mohamed, A.; Qian, K.; Elhoseiny, M.; and Claudel, C. 2020. Social-stgcnn: A social spatio-temporal graph convolutional neural network for human trajectory prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 14424–14432.

Narayanan, S.; Moslemi, R.; Pittaluga, F.; Liu, B.; and Chandraker, M. 2021. Divide-and-Conquer for Lane-Aware Diverse Trajectory Prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 15799–15808.

Othman, K. 2021. Public acceptance and perception of autonomous vehicles: a comprehensive review. *AI and Ethics*, 1–33.

Reimers, N.; and Gurevych, I. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.

Roddick, T.; Kendall, A.; and Cipolla, R. 2018. Orthographic feature transform for monocular 3d object detection. *arXiv preprint arXiv:1811.08188*.

Rodriguez, C.; Fernando, B.; and Li, H. 2018. Action anticipation by predicting future dynamic images. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, 0–0.

Rufus, N.; Jain, K.; Nair, U. K. R.; Gandhi, V.; and Krishna, K. M. 2021. Grounding Linguistic Commands to Navigable Regions. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

Rufus, N.; Nair, U. K. R.; Krishna, K. M.; and Gandhi, V. 2020. Cosine meets softmax: a tough-to-beat baseline for visual grounding. In *European Conference on Computer Vision*, 39–50. Springer.

Schoettle, B.; and Sivak, M. 2014. A survey of public opinion about autonomous and self-driving vehicles in the US, the UK, and Australia. Technical report, University of Michigan, Ann Arbor, Transportation Research Institute.

Sriram, N.; Maniar, T.; Kalyanasundaram, J.; Gandhi, V.; Bhowmick, B.; and Krishna, K. M. 2019. Talk to the Vehicle: Language Conditioned Autonomous Navigation of Self Driving Cars. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 5284–5290.

Vasudevan, A. B.; Dai, D.; and Van Gool, L. 2021. Talk2nav: Long-range vision-and-language navigation with dual attention and spatial memory. *International Journal of Computer Vision*, 129(1): 246–266.

Wang, T.; Zhu, X.; Pang, J.; and Lin, D. 2021. FCOS3D: Fully Convolutional One-Stage Monocular 3D Object Detection. *arXiv preprint arXiv:2104.10956*.

Williams, A.; Nangia, N.; and Bowman, S. R. 2017. A broad-coverage challenge corpus for sentence understanding through inference. *arXiv preprint arXiv:1704.05426*.

Yagi, T.; Mangalam, K.; Yonetani, R.; and Sato, Y. 2018. Future person localization in first-person videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 7593–7602.

Yin, T.; Zhou, X.; and Krähenbühl, P. 2021. Center-based 3D Object Detection and Tracking. *Conference on Computer Vision and Pattern Recognition*.

Yu, L.; Lin, Z.; Shen, X.; Yang, J.; Lu, X.; Bansal, M.; and Berg, T. L. 2018. Mattnet: Modular attention network for referring expression comprehension. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1307–1315.

Zheng, W.; Tang, W.; Jiang, L.; and Fu, C.-W. 2021. SE-SSD: Self-Ensembling Single-Stage Object Detector From Point Cloud. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 14494–14503.

Zhou, X.; Wang, D.; and Krähenbühl, P. 2019. Objects as points. *arXiv preprint arXiv:1904.07850*.

Zhou, Y.; and Tuzel, O. 2018. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 4490–4499.

Zieba, M.; Przewiezlikowski, M.; Smieja, M.; Tabor, J.; Trzcinski, T.; and Spurek, P. 2020. RegFlow: Probabilistic Flow-based Regression for Future Prediction. *arXiv preprint arXiv:2011.14620*.