Text2Pos: Text-to-Point-Cloud Cross-Modal Localization

Manuel Kolmet Qunjie Zhou Aljoša Ošep Laura Leal-Taixé
Technical University of Munich, Germany
{manuel.kolmet, qunjie.zhou, aljosa.osep, leal.taixe}@tum.de
text2pos.github.io

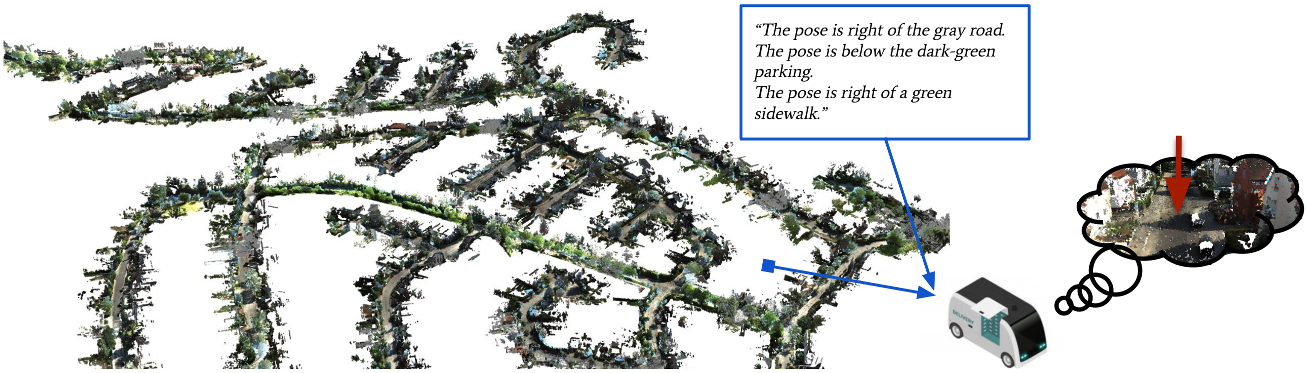

Figure 1. We propose Text2Pos for city-scale position localization based on textual descriptions. Given a point cloud that represents our surroundings and a query position description, Text2Pos provides the most-likely estimate of the described position within that map.

Abstract

Natural language-based communication with mobile devices and home appliances is becoming increasingly popular and has the potential to become natural for communicating with mobile robots in the future. Towards this goal, we investigate cross-modal text-to-point-cloud localization that will allow us to specify, for example, a vehicle pick-up or goods delivery location. In particular, we propose Text2Pos, a cross-modal localization module that learns to align textual descriptions with localization cues in a coarse-to-fine manner. Given a point cloud of the environment, Text2Pos locates a position that is specified via a natural language-based description of the immediate surroundings. To train Text2Pos and study its performance, we construct KITTI360Pose, the first dataset for this task based on the recently introduced KITTI360 dataset. Our experiments show that we can localize 65% *of textual queries within* 15*m distance to query locations for top-10 retrieved locations. This is a starting point that we hope will spark future developments towards language-based navigation.*

"Alexa, hand me over my special delivery at the sidewalk in front of the yellow building next to the blue bus stop."

Authors of this paper, the future. Hopefully.

1. Introduction

Future mobile robots, such as autonomous vehicles and delivery drones, will need to cooperate with humans to coordinate actions and plan their trajectories. In this paper we tackle large scale position localization of the target position based on natural-language-based position descriptions, as needed for, *e.g*., for goods delivery or for vehicle pickup.

For self-localization within a map, mobile agents rely on visual localization methods [4, 22, 35, 41, 56]. These methods match observed images either to a database of geo-tagged images [4, 16, 45] or point-cloud-based maps [33, 35, 39], often obtained using structure-from-motion techniques [38, 40]. By contrast, in this paper, we study *language-based localization of any location*, which, importantly, *does not require the user to be physically present at the target location.* This would, for example, allow us to explain the pick-up position or delivery location through text/voice to a robo-taxi via natural language based communication, that is preferable to humans. Our method can also be seen as complementary to GPS localization methods, *e.g*., when a GPS tag is too coarse, unavailable, or language-based communication is more convenient.

As the main contribution of this paper, we formalize the task of language-based localization and provide the first dataset and methods for this task. In this problem setting, we assume an intelligent agent is given access to the map of the environment that comes in the form of a 3D point cloud and object instance labels. While there are several ways of acquiring point clouds, we rely on LiDAR point clouds, readily available in modern automotive [8, 43, 52] and urban [26] datasets, On the query side, we assume a textual description of the query position surroundings, such as the one shown in Fig. 1. The task is then to provide the most likely position estimate based on this query.

To study this challenging problem, we need a dataset that (i) provides a point-cloud-based representation of the environment and (ii) provides labels in the form of query positions and their corresponding textual descriptions, extracted from their immediate surroundings. We build on the recently proposed KITTI360 dataset [52], which provides nine scenes (city districts), covering 80 km of driving data. Importantly, this dataset provides semantic and instance-level annotations of the point cloud, which we use to automate the generation of query position descriptions. We obtain the *KITTI360Pose* dataset by randomly sampling query positions and by generating corresponding textual descriptions. For each query position we automatically generate multiple descriptions based on a natural language template that specifies spatial relations of surrounding instances to the query position, together with their semantic classes and appearance. We generate $43,381$ such descriptions for $14,934$ sampled positions, which we split by scene (representing a city district) to obtain our train/test splits.

We use this dataset to train and evaluate our proposed Text2Pos model that performs coarse-to-fine localization. In the coarse localization step, we retrieve sub-regions of the map that likely contain our target position. To this end, our network learns to align the encoded query with the point clouds, representing these sub-regions. We finally refine the position estimate within retrieved candidate regions using our matching-based fine localization module. Our experiments show that we can localize such randomly generated positions within KITTI360 scenes with 65% recall for top-10 queries, demonstrating that localizing positions based on textual descriptions is feasible.

In summary, our **main contributions** are: we (i) introduce and formalize the task of 3D point-cloud based localization based on textual descriptions. To this end, we (ii) provide *KITTI360Pose*, the first public dataset for this task, based on the KITTI360 dataset, together with our method for automated mining of positions and corresponding textual descriptions. We (iii) provide a coarse-to-fine baseline model for the localization task, that learns to align objects which are mentioned in the text with object instances in the point cloud and thoroughly evaluate and ablate the performance on this challenging new task. We believe this work is the first step towards natural language-based communication with future mobile robots, such as delivery drones and self-driving taxis.

2. Related work

Vision-based localization. Related to our problem is the task of visual localization [4, 6, 16, 22, 32, 33, 35, 41, 45, 46, 56], which means estimating a precise pose based on an observed image or image sequence. Existing methods commonly adopt a two-stage coarse-to-fine localization pipeline [33, 35, 56]. Given the query image, a *coarse* step firstly finds a subset of images with aligning views using image retrieval techniques [4, 16, 45]. Then, the *fine* step establishes 2D-2D correspondences between pixels of the query and the retrieved images based on the visual descriptors. These can further be used to obtain 2D-3D correspondences between the query and a 3D map, usually obtained using structure-from-motion techniques. Finally, the camera poses can be computed using either a set of 2D-2D correspondences [56] or 2D-3D correspondences [33, 35]. Our method follows the coarse-to-fine localization scheme by first localizing a coarse cell, containing the objects described by the query text, followed by a more precise pose estimate within the coarse cell. Compared to matching between visual features, our method needs to implicitly learn to align two different modalities: text and 3D point clouds. By contrast to visual localization, commonly used for robot *self-localization*, we tackle linguistic localization, intended to specify an arbitrary *target location*.

2D vision and language. Vision and language understanding has been widely investigated in tasks such as image captioning [20, 24, 47, 53], visual question answering (VQA) [3, 50] and visual grounding [18, 21, 25, 54], the task of localizing visual elements in the images that are linguistically described by the query text. Visual and linguistic perceptions are combined to assist the task of robot navigation from room to room under building-scale environments [2]. The ALFRED benchmark [42] was later released to encourage research on connecting language to a series of human daily tasks in an interactive visual 3D environment. Closer to our task is text-to-image retrieval [19, 27, 28, 48, 49], where text descriptors are learned to match corresponding image descriptors. This usually requires the model to reason about the relationship between a set of words and image regions and match a word to its corresponding image region [28]. The main difference between our approach and previous work is that we match from text to point clouds instead of images. In the coarse localization stage, our method first matches a sequence of texts to a cell which represents a region in the scene and contains a set of objects that are then matched to individual textual object hints.

3D vision and language. Motivated by the 3D world we live in, recent work explores the potential of 3D vision and language understanding on the tasks of 3D shape generation [11] and language grounding of 3D objects [1, 10, 15, 30, 55]. The method by [30] embodies language grounding implicitly on 3D visual features and predicts 3D bounding boxes for target objects of primitive shapes of different colors. ScanRefer [10] localizes 3D objects referred by the query descriptions in real-life indoor scenes. ReferIt3D [1] tackles a similar task, but assumes to be given segmented object instances in a room and focuses on identifying the referred object among instances of the same fine-grained category. InstanceRefer [55] improves their performance by using a 3D panoptic segmentation backbone, guiding the model to capture multi-level visual context. Recent work by [15] proposes several graph modules that aid learning of the contextual information for both visual and language domains.

Similar to our work, these methods localize regions in 3D point clouds based on textual queries. However, this is different to our proposed city-scale text-based position localization task in several aspects: For 3D object reference, a model needs to learn to align a natural-language descriptor to one of the objects in the scene. Different to that, our model needs to learn to interpret a composition of objects as a location and distinguish it from other possible locations, since there is no explicit visual notion of a position. Additional challenges stem from the fact that we are targeting large (city) scale outdoor scene localization. This is challenging due to memory constraints of modern GPUs, which can fit only a very small portion of a city scale point cloud. Furthermore, outdoor regions are less diverse in terms of semantics compared to cluttered man-made indoor environments [29], making it difficult to leverage semantic instances to obtain a unique position signature. In summary, our work serves as the first attempt to tackle this challenging task and opens the door to natural language based localization for the 3D vision and language community.

3. The *KITTI360Pose* dataset

To tackle language-based position localization in large-scale environments such as urban cities, we (i) need a large-scale dataset that provides point clouds representing real-world cities and (ii) a large set of position-text pairs to train and evaluate our models. To this date, no such dataset exists, and hand-annotating textual queries would be prohibitively expensive.

The recently introduced KITTI360 [52][1] dataset provides nine static scenes that represent different districts of the city of Karlsruhe, covering in total over $80km$ of driving distance. These scenes were obtained by registering LiDAR scene scans using LiDAR SLAM methods (*e.g.*, [5]). These point clouds would be suitable for studying this problem; however, the dataset does not provide textual descriptions of positions. Luckily, KITTI360 provides object instance labels for static (*e.g.*, building, traffic light, garage) and dynamic (*e.g.*, person, car, bicycle) object instances and semantic labels for the *stuff* classes (*e.g.*, road, vegetation, wall). In the following, we utilize these object instance and semantic labels to automatize the generation of position-description query pairs. We use these to train our models and to benchmark large-scale cross-modal localization without manual annotation work.

In this study, we focus on point clouds, recorded by LiDAR sensors, readily available in modern automotive [8, 43, 52] and robotics [26] datasets. Our approach would also be applicable to point clouds obtained using structure-from-motion methods [38, 40] available in existing visual localization datasets [22, 36]. However, such datasets currently do not contain appropriate instance annotations that we could utilize to automatize the generation of query positions. Finally, we note that indoor RGB-D datasets, such as [9, 12] do contain such object instance labels. However, in this paper, we explicitly aim to study large-scale localization, hence, focus on outdoor scenarios.

3.1. Dataset Generation

Object instances. Contrary to the majority of existing automotive datasets [8, 43], that focus on instance segmentation of dynamic objects such as *cars* and *pedestrians*, KITTI360 additionally provides object instance labels for several static classes, such as *buildings* and *traffic lights*, that provide a reliable cue for localization. In this work, we leverage static object instances to generate position queries and as cues for the position localization.

In addition to labeled instances, we also further split certain *stuff* classes and use the obtained clusters to generate descriptions. For example, the class *vegetation* aggregates a large set of separate trees and bushes which could be specified as localization cues into a single object that spreads across the entire scene. In order to instead recover a set of separate and therefore localizable instances, we cluster all *stuff* classes, such as *vegetation*, *fence* and *wall* using the DBSCAN [14] algorithm. We provide further details on the clustering procedure in the supplementary material.

Query generation. The next step is position-query pair generation. The aim here is to obtain a set of positions and corresponding texts that describe each position qualitatively based on the surrounding objects and their spatial relations in an automated fashion. We start by sampling equidistant locations along recorded vehicle trajectories, readily available with maps. In the vicinity of each sampled location, we sample a fixed number of random locations (in practice, 4 or 8) to increase the number of positions. We describe adjacent objects to each position based on their relative position, color and semantic class by generating textual descriptions based on a simple sentence template, describing the position

[1] Available under Creative Commons Attribution-NonCommercial-ShareAlike 3.0 license. This dataset contains no personally identifiable information or offensive content.

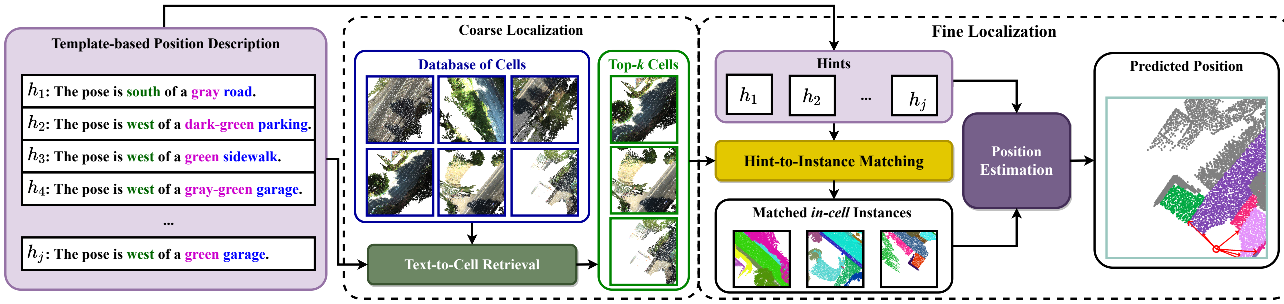

Figure 2. **Text2Pos**. *Coarse localization.* Given a template-based query position description, we first identify a set of coarse candidate locations (*i.e.*, cells) that potentially contain the target position, which serves as the coarse localization of the query. This is achieved by retrieving *top-k* nearest cells from our constructed database of cells using our text-to-cell retrieval model. *Fine localization.* we then refine the pose within retrieved cells via our position refinement module.

in relation to adjacent objects. We extract the relative position and object color tag in an automated fashion directly from the point cloud. We detail the sentence generation in the supplementary.

In the following, we refer to one such generated sentence as a *hint*. A position description $\mathcal{T}$ is defined by a set of hints $\{h_i\}$ describing a given position, *i.e.*, $\mathcal{T} := \{h_i\}_{i=1}^{N_h}$ where N_h is the number of hints per position description. The set of objects to describe a position with is obtained by selecting a set of k objects close to the sampled query position. We retain only positions with at least N_h objects in their vicinity. The reason for such well-structured queries is two-fold: (i) this allows us to investigate the problem without costly human annotations, and (ii) enables us to rigorously study cross-modal localization in a well-controlled setting based on explicit hint-to-object matching. We are confident that more complex language queries can be encoded and understood by building on recent developments in the field of natural language processing [7, 13].

KITTI360Pose dataset. Using the procedure described above, we generate the *KITTI360Pose* dataset. In particular, we sample $14,934$ positions and generate up to three descriptions for each, totaling in $43,381$ position-query pairs. We use five scenes (districts) for training (covering in total $11.59km^2$), one for model validation, and three for testing (covering in total $2.14km^2$). An average district covers an area of $1.78km^2$. In contrast, the Cambridge dataset [22] covers an area of $0.063km^2$, Oxford RobotCar and CMU Seasons [36] cover $10km$ and $8.5km$ of driving distance (respectively), and Tokyo 24/7 [45] covers the area of $2.56km^2$. Descriptions of these positions are generated based on objects that fall within a $15m$ radius for a sampled position. We provide additional details in the supplementary.

4. Tex2Pos: A Baseline for Language-based Localization

Given a textual position query, our goal is to localize an agent within a given point-cloud-based (Sec. 3.1) map via 2D planar coordinates of its position w.r.t. the scene coordinate system. To this end, we propose the first text-based coarse-to-fine localization method, that we outline in Fig. 2. Due to the large-scale nature of the problem, we follow a coarse-to-fine framework, well studied and proven successful in the field of large-scale visual localization [33, 56]. We first perform a coarse localization of the query where we discretize the search region into rectangular cells and retrieve the *top-k* cells matched to the description from the database (Fig. 2, *left*). To refine this estimate, we match the visible 3D instances within a retrieved cell, to their corresponding referring hints in the textual description (Fig. 2, *right*). Finally, we obtain the position estimate from the set of instances identified by the text (Sec. 4.2). We note that the coarse retrieval could in the future be replaced by hints such as street name or coarse address. This would require establishing an additional alignment between the 3D point cloud and city map, and remains our future work.

4.1. Coarse Localization

Image retrieval techniques [4, 16, 45] are commonly used within visual localization pipelines to efficiently narrow down the search space [33, 44, 56] or even provide a direct coarse position estimate [37] of the query image [36]. Given a query image, its learned global descriptor is matched against the global descriptors extracted from a database of reference images to obtain its *top-k* nearest reference images based on their descriptor distances. We follow this general approach in our *text-to-cell* cross-modal retrieval method. With this step, we aim to efficiently locate candidate regions within the map that could potentially contain our target position.

Database construction. As a pre-processing step, we divide point clouds, representing city districts, into rectangular cells. We sample cells by sliding a $\mathcal{W} \times \mathcal{W}$ window with a stride of $\mathcal{S}$ horizontally and vertically over the scene, where the cell size $\mathcal{W}$ should be large enough to contain a certain number of instances, that can be used to describe a position. The stride size $\mathcal{S}$ is picked to be smaller than the

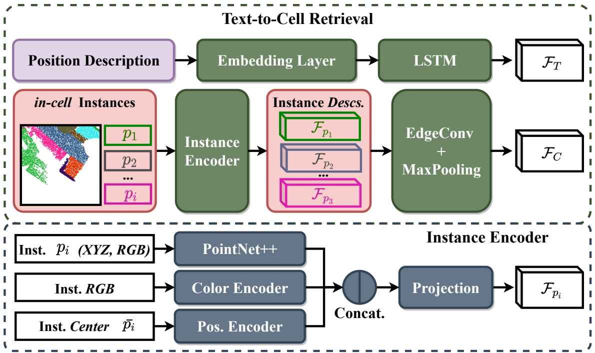

Figure 3. *(top)* Query text encoder and cell encoder architecture, *(bottom)* instance encoder architecture.

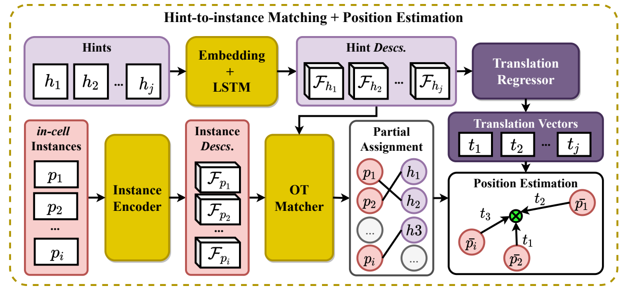

Figure 4. **Fine localization**. In the fine localization, for each candidate cell, we first establish the correspondences between the query hints and its *in-cell* instances, which allows us to filter out noisy signals that are not useful for position estimation from both domains. We then predict a vector for each matched instance that translates its instance center to a position estimate via a translation regressor. The final position is the average of the position estimates by all matched instances.

cell size to cover the whole scene area and to allow partially overlapping cells. We consider an instance p_i to be inside a cell $\mathcal{C}$ if at least a third of its points lie within the cell, or if a minimal number of points (250 in practice) overlap with the cell – this criterion is important for *stuff* classes, such as *tree* or *building*. We name those instances *in-cell* instances of that cell, *i.e.*, $\mathcal{C} := \{p_i\}_{i=1}^{N_p}$ where N_p is the number of *in-cell* instances per cell and varies for different cells.

Text-to-cell retrieval. Given a position description, the task of our retrieval model is to identify its *top-k* candidate cells that are likely to contain the described position location. Compared to image-to-image retrieval, the model needs to learn to extract descriptors for inputs from two different modalities, *i.e.*, text and point clouds, such that the two can be directly compared using Euclidean distance in the embedding space.

As shown in Fig. 3 (*top*), our retrieval network has two encoding branches to process a query position description $\mathcal{T}$ and a candidate cell $\mathcal{C}$. The complete position description $\mathcal{T}$ is encoded into a global text descriptor $\mathcal{F}_T$ using a bidirectional LSTM cell [17]. On the cell encoding side, we first extract a descriptor $\mathcal{F}_{p_i}$ for each *in-cell* instance $p_i \in \mathcal{C}$. We aggregate *in-cell* instance descriptors $\{\mathcal{F}_{p_i}\}_{i=1}^{N_p}$ into a global cell descriptor $\mathcal{F}_C$ using an EdgeConv layer [51] followed by a max pooling operation.

Instance encoder. Each instance P_i is represented by a point cloud where each point contains three spatial and three color (RGB) coordinates, yielding 6D input features (Fig. 3 *bottom*). We encode such a point cloud using a PointNet++ [31] backbone, which gives us a semantic embedding. In addition, we explicitly obtain a color embedding of it by encoding its RGB coordinates using our color encoder and a positional embedding of it by encoding its instance center $\bar{P}_i$, *i.e.*, the mean value of its coordinates, using our positional encoder. Each of the color encoder and positional encoder takes form of a 3-layer multi-layer perceptron (MLP), whose output dimension is the same as the semantic embedding dimension. The semantic, color and positional embeddings are fused by concatenation and fed into a projection layer (another 3-layer MLP) which outputs a final instance embedding $\mathcal{F}_{p_i} \in \mathcal{R}^{D_p}$.

4.2. Fine Localization

Given a set of retrieved candidate cells, we now find cues that allow us to refine the position among those cells based on the query description. In classical visual localization, such a fine localization step normally relies on establishing 2D pixel correspondences between the query image and retrieved *top-k* database images [33, 56]. These images have geo-information attached and provide cues to compute an accurate camera position for the query. Inspired by this idea, we propose to compute the refined position by establishing hint-to-instance correspondences between a position description and its *top-k* retrieved cells.

Hint-to-instance matching. Given a position description $\mathcal{T}$ and a candidate cell $\mathcal{C}$, we first establish hint-to-instance correspondences with our cross-modal matching module (Fig. 4). As explained in Sec. 3, a position description consists of a series of hints $\{h_j\}_{j=1}^{N_h}$, where each hint describes a single instance about its attributes and relation to the position. We use our *instance encoder* to extract a descriptor $\mathcal{F}_{p_i}$ for each *in-cell* instance $p_i \in \mathcal{C}$ and our text encoder to extract a descriptor $\mathcal{F}_{h_j}$ per hint.

We then use a matching module (inspired by SuperGlue [34]) to perform a partial matching between the set of instances and the set of sentences. The matching module first uses several blocks of self-attention layers and cross-attention layers to propagate contextual information. The pair-wise similarity scores are computed from the two sets of aggregated descriptors as the cost matrix to an optimal transport (OT) matching layer. We adopt the same OT setup as in [34]. Finally, we obtain the partial assignment from hints to instances by picking matches with confidence scores above the certain threshold (0.2 in practice).

Translation-based refinement. For each identified match

Stride	# Cells	Localization Recall ($\epsilon < 5/10/15m$) $k=1$	$k=5$	$k=10$
$\mathcal{S}=10m$	1434	**0.14/0.25/0.31**	**0.36/0.55/0.61**	**0.48/0.68/0.74**
$\mathcal{S}=15m$	629	0.10/0.19/0.25	0.26/0.47/0.56	0.35/0.61/0.70
$\mathcal{S}=20m$	362	0.07/0.15/0.19	0.18/0.36/0.45	0.25/0.50/0.60

Table 1. Ablation on varying sampling stride for cell database construction.

(p_i, h_j) of the instance p_i and the hint h_j, we additionally learn a translation vector t_i that transforms the instance center $\bar{p_i}$ to the target position. Each translation prediction t_i then leads to a position estimate that is defined as $\tilde{y_i} = \bar{p_i} + t_i$. To predict such a translation vector, we train an additional 3-layer MLP translation regressor that takes a hint descriptor $\mathcal{F}_{h_j}$ as the input, and outputs a translation vector t_i. The final position of the query text is computed by taking the average of the set of estimates $\{\tilde{y_i}\}$ predicted from all hint-to-instance matches. We show in our ablation (Sec. 5.1) that our learned translation leads to a more accurate position prediction compared to simple averaging of the matched instance positions.

4.3. Losses

Coarse loss. Given an input batch of cell descriptors $\{\mathcal{F}_C^i\}_{i=1}^{N_b}$ and matching text descriptors $\{\mathcal{F}_T^i\}_{i=1}^{N_b}$ where N_b is the batch size, we train the network for cross-domain retrieval with the pairwise ranking loss [23]:

$$\begin{aligned}\mathcal{L}_{coarse} = &\sum_{i}^{N_b}\sum_{j\neq i}^{N_b}[\alpha - \langle \mathcal{F}_C^i, \mathcal{F}_T^i\rangle + \langle \mathcal{F}_C^i, \mathcal{F}_T^j\rangle]_+ \\ &+ \sum_{i}^{N_b}\sum_{j\neq i}^{N_b}[\alpha - \langle \mathcal{F}_T^i, \mathcal{F}_C^i\rangle + \langle \mathcal{F}_T^i, \mathcal{F}_C^j\rangle]_+ \,,\end{aligned} \tag{1}$$

where $[\bullet]_+$ equals $max(0, \bullet)$ and α is the margin hyper parameter. This loss enforces that each cell descriptor $\mathcal{F}_C^i$ is closer to its matching text descriptor $\mathcal{F}_T^i$ than it is to non-matching text descriptors $\mathcal{F}_T^j$ in the batch by a margin. The same is also enforced when matching from a text descriptor $\mathcal{F}_T^i$ to a cell descriptor.

Fine loss. To train our matching module, we adopt the matching loss used in [34], which maximizes the scores at ground truth (GT) match locations in the predicted assignment matrix. We train this model separately from the coarse module. To train the translation regressor, we minimize the mean square error between the predicted and the GT translation. Matching module and the regressor are jointly trained with the sum of the matching loss and regression loss. We provide implementation details of model training in our supplementary material.

Train	Infer	Localization Recall ($\epsilon < 5/10/15m$) $k=1$	$k=5$	$k=10$
4	6	0.12/0.21/0.27	0.30/0.48/0.55	0.41/0.61/0.67
6	6	0.13/**0.25/0.30**	**0.36/0.54/0.60**	**0.47/0.67/0.73**
10	6	**0.14/0.25/0.30**	**0.36**/0.51/0.57	**0.47**/0.64/0.70
12	6	0.11/0.21/0.25	0.30/0.45/0.51	0.41/0.58/0.64
6	4	0.09/0.17/0.22	0.26/0.42/0.48	0.35/0.55/0.61
6	6	0.13/0.25/0.30	0.36/0.54/0.60	0.47/0.67/0.73
6	10	0.20/0.34/0.40	0.47/0.68/0.73	0.61/0.80/0.84
6	12	**0.23/0.36/0.41**	**0.50/0.69/0.74**	**0.63/0.80/0.84**

Table 2. Ablation on the number of hints in a query description.

5. Experimental Evaluation

In this section, we discuss the performance of our model (Sec. 4) on the proposed *KITTI360Pose* dataset. We report results on our validation split for ablation studies (Sec. 5.1) and on our test split for a final evaluation of our best performing models (Sec. 5.2). For details on the dataset and splits, we refer to our supplementary.

Evaluation metrics. We perform the evaluation w.r.t.the top k retrieved candidates ($k \in \{1, 5, 10\}$) and report localization recall, *i.e.*, the ratio of successfully localized queries if its error is below specific error thresholds, *i.e.*, $\epsilon < 5/10/15m$ by default.

5.1. Model Ablations

Database construction. The *coarse localization* module (as explained in Sec. 4.1) retrieves the *top-k* candidate cells. To study its performance w.r.t. the localization task, we use the center of a cell as a coarse position estimate and measure its accuracy. As our first ablation, we study the impact of the cell sampling stride $\mathcal{S}$ on localization performance.

As shown in Tab. 1, the retrieval performance on cells sampled with $\mathcal{S} = 10m$ performs better than other stride settings across different k values. As the smaller stride implies more overlap between consecutively sampled cells, it helps our model to learn more discriminative descriptors that can be used to distinguish the content of close-by cells. While decreasing the stride allows for more accurate localization, it increases the computation demand in terms of memory and runtime in quadratic order. We present results for denser sampling using $\mathcal{S} = 1/3/5m$ in the supplementary. Considering the trade-off between accuracy and computational efficiency, we use $\mathcal{S} = 10m$ to train and evaluate our models in the following experiments.

Ablation on number of localization cues. To evaluate how the number of specified object cues influences localization recall, we vary the number of hints N_h from four to 12 either during training or inference, as shown in Tab. 2. The results show that alternating the number of hints below or above 6 during training decreases the recall in most metrics. Furthermore, we show that our model trained on

Model	Localization Recall ($\epsilon < 2/5/10m$)
Center of Cell:	0.14/**0.77**/**0.99**
Mean of Matched Instance:	0.15/0.62/0.97
Matched Instance + Translation (Text2Pos):	**0.24**/0.76/**0.99**
Matching Oracle:	0.34/0.91/1.00
Translation Oracle:	0.55/0.90/0.99
Both Oracles:	1.00/1.00/1.00

Table 3. Ablation on fine localization components. This ablation study on the fine localization module requires narrower localization thresholds compared to Tab. 1 to reveal differences in localization precision.

$N_h = 6$ hints is robust to varying N_h during inference and that performance even rises by up to 16 points for $N_h = 12$, which is to be expected as additional hints help to resolve some of the ambiguity in cross-modal localization.

Fine localization module. In this ablation, we provide insights into design decisions for the *fine localization* module (Sec. 4.2). To make the evaluation independent of the retrieval module performance, we use a retrieval oracle in the following experiments to provide the model with the database cell that is closest to the GT position.

Firstly, we study the performance of the hints-to-instance matching module, by measuring precision and recall of the predicted matches given the GT matches. Our matching model can achieve 78% and 76% for both recall and precision, respectively.

Next, we show the benefit of our translation regressor by comparing the following three different variants for position estimation: (i) we take the cell-center as the estimated position (no refinements); (ii) we compute the mean of the instances matched by query hints, and (iii) we use the mean of the position estimates computed by using the predicted translation vectors. Due to the use of a retrieval oracle, we report their localization recall with stricter error thresholds $\epsilon < 2/5/10m$. As shown in Tab. 3 (*top*), all three variants can localize queries within $10m$ errors with almost 100% recall. The difference between model performance becomes more evident when using the smaller error threshold $\epsilon = 2m$, where we see the simple cell-center baseline performs poorly to localize only 14% of queries. This only improves marginally by 1% if we take the mean center of matched instances. Additionally, we even see the opposite results for a moderate error of $\epsilon = 5m$ where the cell-center baseline outperforms the instance-mean baseline by 15%. The results thus suggest that naively assuming either the position to be at the center of the cell or between the described instances is not sufficient. Instead of making such manual assumptions, we learn where to expect the position given an instance of a specific object class in a data-driven manner. We show that our learned translation leads to 24% localization recall on the smallest threshold, outperforming the other two variants by up to 10% and confirming its potential efficacy for fine-grained localization.

Finally, we replace parts of the *fine model* by oracles to understand the limitations of individual components of our method. As shown in Tab. 3 (*bottom*), using GT associations (*matching oracle*) instead of performing hint-to-instance matching using our model, we localize $(10\%/15\%)$ more queries within $2/5m$ errors. Purely replacing predicted translation with GT translations (*translation oracle*) leads to 31% more queries localized within $2m$, suggesting room for improvements by predicting more accurate translation vectors. Both oracles combined yield perfect localization as expected.

Pipeline-level oracle ablation. While the previous experiments used a coarse oracle to evaluate fine localization in isolation, we now replace first our coarse and then our fine module by oracles, in order to understand their limitations of the full system. The results are shown in Tab. 4. With *coarse oracle*, we replace the retrieval component with a retrieval oracle (as in the previous fine module ablation). As can be seen, identifying cells reliably localizes all queries within $15m$ distance to the query position, and with 98% accuracy within $10m$ threshold. This clearly shows significant potential in improving the retrieval module, which can be achieved potentially with additional information (such as nearby landmarks or street names) to reduce the inherent cell ambiguity. This remains our future work. With *fine oracle*, we then measure the localization performance of our learned retrieval model + GT matching association + GT translation prediction. We show that perfect fine localization improves the baseline by 11%/29%/31% at the smallest thresholds for $k = 1/5/10$ candidate cells, which indicates further potential in improving the refinement module. Finally, using both oracles, we can localize all queries within $5m$ distance from the query position. For the lower part of Tab. 4, we also replace our coarse and fine modules with random choice to highlight their importance. Compared to our baseline, a random coarse retrieval leads to near-zero localization recall, while a random matching in the fine module leads to localizing up to 34 % less queries.

Comparison to visual localization While Text2Pos is not intended as a competitor against visual localization, we do provide a comparison in localization recall through a two-step experiment (Tab. 5). Visual localization has to be performed on rendered images in order to be applicable on our dataset, so we first show a comparison between real and rendered images from identical locations, confirming that rendered images do not significantly reduce retrieval performance. Then, we compare visual retrieval (NetVLAD [4][2]) to our cross-modal retrieval by rendering images at each cell and query pose of our validation set, resulting in 5736 and 3187 images on the database and query side, respectively.

The results indicate that visual retrieval shows superior

[2]We use the pretrained NetVLAD obtained from here

Model	Localization Recall ($\epsilon < 5/10/15m$) $k=1$	$k=5$	$k=10$
Text2Pos	0.14/0.25/0.31	0.36/0.55/0.61	0.48/0.68/0.74
Coarse Oracle	0.68/0.98/1.00	-	-
Fine Oracle	0.35/0.37/0.38	0.65/0.67/0.68	0.77/0.78/0.80
Both Oracles	1.00/1.00/1.00	1.00/1.00/1.00	1.00/1.00/1.00
Coarse random	0.00/0.01/0.01	0.01/0.02/0.03	0.02/0.04/0.06
Fine random	0.03/0.10/0.20	0.09/0.30/0.49	0.14/0.40/0.62

Table 4. Ablation on localization oracle. Note that the line 2 results differ from Tab. 3 line 3 due to different experimental setups.

Model	Localization Recall ($\epsilon < 5/10/15m$) $k=1$	$k=5$	$k=10$
NetVLAD (real)	0.11/0.51/0.79	0.11/0.59/0.95	0.11/0.60/0.96
NetVLAD (rendered)	0.10/0.49/0.74	0.11/0.59/0.93	0.11/0.59/0.95
Text2Pos (w/o fine loc.)	0.10/0.23/0.30	0.27/**0.52**/0.60	**0.37/0.65/0.72**
NetVLAD [4]	**0.18/0.33/0.43**	**0.29**/0.50/**0.61**	0.34/0.59/0.69

Table 5. Comparison with visual localization. A pre-trained NetVLAD model for visual retrieval is compared between real and rendered images (top rows) and against our novel cross-modal retrieval.

Model	Localization Recall ($\epsilon < 5/10/15m$) $k=1$	$k=5$	$k=10$
Text2Pos (full)	**0.13/0.21/0.25**	**0.33/0.48/0.52**	**0.43/0.61/0.65**
Text2Pos (w/o trans. offs.)	0.10/0.20/**0.25**	0.26/0.46/**0.52**	0.35/0.58/**0.65**
Text2Pos (w/o fine loc.)	0.10/0.20/**0.25**	0.27/0.46/**0.52**	0.35/0.59/**0.65**

Table 6. Model evaluation on the test set.

accuracy for smaller *top-k* values with up to 13 percentage points at $k = 1$ and $\epsilon = 15m$, but also lags behind cross-modal retrieval with up to 6 percentage points in $k = 10$ and $\epsilon = 10m$. We note that for simplicity, this comparison considers retrieval only without refinement and that more advanced visual localization pipelines are available. Nevertheless, we take this as a first indication that equally strong advancements in cross-modal localization might approach the accuracy of state-of-the-art visual pipelines.

5.2. Evaluation on the Test Set

As can be seen in Tab. 6, for top-1 retrieval, we can successfully localize up to 25% of queries up to $15m$ distance threshold and 20% of queries up to a distance of $10m$. When considering top-5 queries, we can already successfully localize 52% of all queries and go up to 65% successfully localized queries when considering top-10 cell candidates. As can be seen, based on purely textual descriptions, we can already localize the target position quite accurately (*i.e.*, $10m$ radius). However, to get to a good localization recall, we need to consider top-10 cell candidates. This suggests that our model is rather uncertain which cell contains the query position. This is not surprising, as many of the query descriptions could refer to several locations in the city. Additional hints such as a nearby address, nearby street names, or landmarks should help in the future to further narrow down the search space and resolve this ambiguity and provide a good direction for future research.

Finally, as can be seen in Tab. 6, the full localization variant using translation prediction outperforms the variant using naive matched-instances position averaging by localizing 3% more queries within $5m$ errors considering top-1 retrieved cell. This improvement increases to 8% when considering top-10 candidates.

6. Broader Impact and Limitations

Our work on text-based localization opens a new front of research on natural-language-based action coordination with future mobile systems, to which we may need to specify our current or a target location. Usage scenarios include autonomous goods delivery (*e.g.*, food or packages), ordering autonomous vehicle pick-up, or sending vehicles to remote locations in case of emergency. These use-cases will play an important role in the automation of tasks that can be considered unsafe (due to a high number of traffic accidents) and currently require a human operator. Automation of these tasks can also improve the weight and utilization of delivery vehicles and, consequentially, the carbon footprint.

However, to devise a first feasible solution, we rely on the following assumptions: (i) we assume our maps contain labeled instances of objects used as anchor points for localization; (ii) we rely on simplistic, template-based localization instructions for training and evaluation. Generalization to arbitrary, unlabeled point clouds and more realistic, human-generated textual queries remains our future work.

7. Conclusion

We presented Text2Pos and KITTI360Pose, the first method and dataset for text-based position localization within a 3D environment. As such language-based communication is natural to human beings, we foresee it will be an integral part of future mobile agents that will require location instructions, such as goods delivery or vehicle pick-up positions. We demonstrated that our coarse-to-fine approach can localize 65% of textual queries within $15m$ distance to query locations when considering top-10 retrieved locations. We believe we can further improve localization precision by using street names and visual landmarks as cues for coarse localization, which we leave for future work. This work is the first step in the direction of language-based localization, showing its great potential, and hopefully inspiring researchers to further make this technology a reality.

Acknowledgments. This project was partially funded by the Sofja Kovalevskaja Award of the Humboldt Foundation. The authors of this work take full responsibility for its content.

References

[1] Panos Achlioptas, Ahmed Abdelreheem, Fei Xia, Mohamed Elhoseiny, and Leonidas Guibas. Referit3d: Neural listeners for fine-grained 3d object identification in real-world scenes. In *ECCV*, 2020. 2, 3

[2] Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton Van Den Hengel. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *CVPR*, 2018. 2

[3] Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. Vqa: Visual question answering. In *ICCV*, 2015. 2

[4] Relja Arandjelovic, Petr Gronat, Akihiko Torii, Tomas Pajdla, and Josef Sivic. Netvlad: Cnn architecture for weakly supervised place recognition. In *CVPR*, 2016. 1, 2, 4, 7, 8

[5] Jens Behley and Cyrill Stachniss. Efficient Surfel-Based SLAM using 3D Laser Range Data in Urban Environments. In *RSS*, 2018. 3

[6] Eric Brachmann, Alexander Krull, Sebastian Nowozin, Jamie Shotton, Frank Michel, Stefan Gumhold, and Carsten Rother. Dsac-differentiable ransac for camera localization. In *CVPR*, 2017. 2

[7] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020. 4

[8] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuScenes: A multimodal dataset for autonomous driving. *arXiv preprint arXiv:1903.11027*, 2019. 2, 3

[9] Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niessner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3d: Learning from rgb-d data in indoor environments. *arXiv preprint arXiv:1709.06158*, 2017. 3

[10] Dave Zhenyu Chen, Angel X Chang, and Matthias Nießner. Scanrefer: 3d object localization in rgb-d scans using natural language. 2020. 2, 3

[11] Kevin Chen, Christopher B Choy, Manolis Savva, Angel X Chang, Thomas Funkhouser, and Silvio Savarese. Text2shape: Generating shapes from natural language by learning joint embeddings. 2018. 2

[12] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *CVPR*, 2017. 3

[13] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018. 4

[14] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *KDD*, 1996. 3

[15] Mingtao Feng, Zhen Li, Qi Li, Liang Zhang, XiangDong Zhang, Guangming Zhu, Hui Zhang, Yaonan Wang, and Ajmal Mian. Free-form description guided 3d visual graph network for object grounding in point cloud. *arXiv preprint arXiv:2103.16381*, 2021. 2, 3

[16] Stephen Hausler, Sourav Garg, Ming Xu, Michael Milford, and Tobias Fischer. Patch-netvlad: Multi-scale fusion of locally-global descriptors for place recognition. In *CVPR*, pages 14141–14152, 2021. 1, 2, 4

[17] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 5

[18] Ronghang Hu, Marcus Rohrbach, and Trevor Darrell. Segmentation from natural language expressions. In *ECCV*, 2016. 2

[19] Yan Huang, Qi Wu, Chunfeng Song, and Liang Wang. Learning semantic concepts and order for image and sentence matching. In *CVPR*, 2018. 2

[20] Andrej Karpathy and Li Fei-Fei. Deep visual-semantic alignments for generating image descriptions. In *CVPR*, 2015. 2

[21] Sahar Kazemzadeh, Vicente Ordonez, Mark Matten, and Tamara Berg. Referitgame: Referring to objects in photographs of natural scenes. In *EMNLP*, 2014. 2

[22] Alex Kendall, Matthew Grimes, and Roberto Cipolla. Posenet: A convolutional network for real-time 6-dof camera relocalization. In *ICCV*, 2015. 1, 2, 3, 4

[23] Ryan Kiros, Ruslan Salakhutdinov, and Richard S Zemel. Unifying visual-semantic embeddings with multimodal neural language models. *arXiv preprint arXiv:1411.2539*, 2014. 6

[24] Jiasen Lu, Caiming Xiong, Devi Parikh, and Richard Socher. Knowing when to look: Adaptive attention via a visual sentinel for image captioning. In *CVPR*, 2017. 2

[25] Junhua Mao, Jonathan Huang, Alexander Toshev, Oana Camburu, Alan L Yuille, and Kevin Murphy. Generation and comprehension of unambiguous object descriptions. In *CVPR*, 2016. 2

[26] Roberto Martin-Martin*, Mihir Patel*, Hamid Rezatofighi*, Abhijeet Shenoi, JunYoung Gwak, Eric Frankel, Amir Sadeghian, and Silvio Savarese. JRDB: A dataset and benchmark of egocentric robot visual perception of humans in built environments. *TPAMI*, 2021. 2, 3

[27] Nicola Messina, Giuseppe Amato, Andrea Esuli, Fabrizio Falchi, Claudio Gennaro, and Stéphane Marchand-Maillet. Fine-grained visual textual alignment for cross-modal retrieval using transformer encoders. *TOMM*, 2020. 2

[28] Nicola Messina, Fabrizio Falchi, Andrea Esuli, and Giuseppe Amato. Transformer reasoning network for image-text matching and retrieval, 2021. 2

[29] Shervin Minaee, Yuri Y Boykov, Fatih Porikli, Antonio J Plaza, Nasser Kehtarnavaz, and Demetri Terzopoulos. Image segmentation using deep learning: A survey. *TPAMI*, 2021. 3

[30] Mihir Prabhudesai, Hsiao-Yu Fish Tung, Syed Ashar Javed, Maximilian Sieb, Adam W Harley, and Katerina Fragkiadaki. Embodied language grounding with implicit 3d visual feature representations. *arXiv preprint arXiv:1910.01210*, 2019. 2

[31] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *NIPS*, 2017. 5

[32] Noha Radwan, Abhinav Valada, and Wolfram Burgard. Vlocnet++: Deep multitask learning for semantic visual localization and odometry. *RAL*, 3(4):4407–4414, 2018. 2

[33] Paul-Edouard Sarlin, Cesar Cadena, Roland Siegwart, and Marcin Dymczyk. From coarse to fine: Robust hierarchical localization at large scale. In *CVPR*, 2019. 1, 2, 4, 5

[34] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superglue: Learning feature matching with graph neural networks. In *CVPR*, 2020. 5, 6

[35] Torsten Sattler, Bastian Leibe, and Leif Kobbelt. Efficient & effective prioritized matching for large-scale image-based localization. *TPAMI*, 39(9):1744–1756, 2016. 1, 2

[36] Torsten Sattler, Will Maddern, Carl Toft, Akihiko Torii, Lars Hammarstrand, Erik Stenborg, Daniel Safari, Masatoshi Okutomi, Marc Pollefeys, Josef Sivic, et al. Benchmarking 6dof outdoor visual localization in changing conditions. In *CVPR*, 2018. 3, 4

[37] Torsten Sattler, Qunjie Zhou, Marc Pollefeys, and Laura Leal-Taixe. Understanding the limitations of cnn-based absolute camera pose regression. In *CVPR*, 2019. 4

[38] Johannes L Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *CVPR*, 2016. 1, 3

[39] Johannes L Schönberger, Marc Pollefeys, Andreas Geiger, and Torsten Sattler. Semantic visual localization. In *CVPR*, 2018. 1

[40] Johannes L Schönberger, Enliang Zheng, Jan-Michael Frahm, and Marc Pollefeys. Pixelwise view selection for unstructured multi-view stereo. In *ECCV*, 2016. 1, 3

[41] Jamie Shotton, Ben Glocker, Christopher Zach, Shahram Izadi, Antonio Criminisi, and Andrew Fitzgibbon. Scene coordinate regression forests for camera relocalization in rgb-d images. In *CVPR*, 2013. 1, 2

[42] Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. Alfred: A benchmark for interpreting grounded instructions for everyday tasks. In *CVPR*, 2020. 2

[43] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. In *CVPR*, 2020. 2, 3

[44] Hajime Taira, Masatoshi Okutomi, Torsten Sattler, Mircea Cimpoi, Marc Pollefeys, Josef Sivic, Tomas Pajdla, and Akihiko Torii. Inloc: Indoor visual localization with dense matching and view synthesis. In *CVPR*, 2018. 4

[45] Akihiko Torii, Relja Arandjelovic, Josef Sivic, Masatoshi Okutomi, and Tomas Pajdla. 24/7 place recognition by view synthesis. In *CVPR*, 2015. 1, 2, 4

[46] Julien Valentin, Matthias Nießner, Jamie Shotton, Andrew Fitzgibbon, Shahram Izadi, and Philip HS Torr. Exploiting uncertainty in regression forests for accurate camera relocalization. In *CVPR*, 2015. 2

[47] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. In *CVPR*, 2015. 2

[48] Haoran Wang, Ying Zhang, Zhong Ji, Yanwei Pang, and Lin Ma. Consensus-aware visual-semantic embedding for image-text matching. In *ECCV*, 2020. 2

[49] Liwei Wang, Yin Li, and Svetlana Lazebnik. Learning deep structure-preserving image-text embeddings. In *CVPR*, 2016. 2

[50] Peng Wang, Qi Wu, Chunhua Shen, Anthony Dick, and Anton Van Den Hengel. Fvqa: Fact-based visual question answering. *TPAMI*, 40(10):2413–2427, 2017. 2

[51] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *TOG*, 38(5):1–12, 2019. 5

[52] Jun Xie, Martin Kiefel, Ming-Ting Sun, and Andreas Geiger. Semantic instance annotation of street scenes by 3d to 2d label transfer. In *CVPR*, 2016. 2, 3

[53] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*, pages 2048–2057, 2015. 2

[54] Licheng Yu, Zhe Lin, Xiaohui Shen, Jimei Yang, Xin Lu, Mohit Bansal, and Tamara L Berg. Mattnet: Modular attention network for referring expression comprehension. In *CVPR*, 2018. 2

[55] Zhihao Yuan, Xu Yan, Yinghong Liao, Ruimao Zhang, Zhen Li, and Shuguang Cui. Instancerefer: Cooperative holistic understanding for visual grounding on point clouds through instance multi-level contextual referring. *arXiv preprint arXiv:2103.01128*, 2021. 2, 3

[56] Qunjie Zhou, Torsten Sattler, Marc Pollefeys, and Laura Leal-Taixe. To learn or not to learn: Visual localization from essential matrices. In *ICRA*, 2020. 1, 2, 4, 5