ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Computer Vision
and Geometry Lab

MSc. Thesis Title

Subtitle

Master's Thesis

Amalie Kjaer
Department of Computer Science D-INFK

Advisors: Dr. Zuria Bauer, Dr. Mihai Dusmanu
Supervisor: Prof. Dr. Marc Pollefeys
Computer Vision and Geometry Group
Department of Computer Science D-INFK

August 13, 2024

Abstract

The abstract gives a concise overview of the work you have done. The reader should be able to decide whether the work that has been done is interesting for him by reading the abstract. Provide a brief account of the following questions:

- What is the problem you worked on? (Introduction)
- What are the shortcomings of existing solutions? (Technical Gap)
- How did you tackle the problem? (Method)
- What were your results and findings? (Experiments)
- Why are your findings significant? (Conclusion)

The abstract should approximately cover half of a page, and does generally not contain citations.

Acknowledgements

Contents

Chapter 1

Introduction

Give an introduction to the topic you have worked on:

- *What is the rationale for your work?* Motivate the problem, *e.g.*with a general description of the problem setting, narrowing down to the particular problem you have been working on in your thesis. Allow the reader to understand the problem setting.

- *What is the technical gap in existing work?* Briefly outline how this problem has been tackled before, and what the shortcomings of the existing solutions are.

- *What is your work doing to fix it?* Given the above background, state briefly the focus of the work.

Chapter 2

Related Work

The related work section has the following purposes:

- *Is the overview concise?* Give an overview of the most relevant work to the needed extent. Make sure the reader can understand your work without referring to other literature.
- *Does the compilation of work help to define the "niche" you are working in?* Another purpose of this section is to lay the groundwork for showing that you did significant work. The selection and presentation of the related work should enable you to name the implications, differences, and similarities sufficiently in the discussion section.

description of task

2.1 Visual Question Answering Task (indoor + other)

VQA vs 3D-QA

indoor is a subtask but it is interesting because we can include some 3D element to it - also this lays the foundation for answering harder questions later on, which cannot be answered simply with 2D methods: eg. about distances and paths, which would be interesting for navigation tasks.

- Task description
- Expected inputs / outputs
- Question ontology
- Classical methods (non-SG), strenghts and limitations
- Scene Graph methods (Graph Question Answering), strengths and limitations (most of these for indoor scenes say they use graphs but they never utilise edge info)

description of data

2.2 Scene Graphs

- Scene graph formal definition + advantages
- Scene graph generation methods
- Graph neural networks
- Applications and limitations

description of the ”building blocks” used to create the model

2.3 Image-text pretrained models

2.4 Transformer encoder / decoders

GNN can be formulated as a

2.5 Existing datasets

Chapter 3

Datasets

ScanNet The ScanNet dataset consists of 1,513 annotated point clouds of 3D-indoor environments, generated from 2,492,518 annotated RGB-D frames. The dataset includes a wide range of spaces such as offices, apartments, bathrooms, classrooms, libraries, and more. It spans both small spaces (e.g., bathrooms, closets) and large spaces (e.g., studio apartments, classrooms). The dataset is annotated with instance-level and semantic-level (NYU40) labels.

Note: In most existing SSG methods, the scene must first be segmented. However, to get a more accurate scene graph dataset, we opt to use the segmentation already provided by ScanNet. we use oracle segmentation (we assume perfect segmentation is available).

Note: we remove scenes with segmentation errors (some segmented objects do not appear in any of the images – specifically scenes 180, 279, 305, 530, 597) [Plot scene / label distribution]

ScanQA The ScanQA dataset consists of xxx questions about the ScanNet scenes. We prepare x different question datasets to experiment on: 1) easy 2) hard 3) complete/mixed 4) constant scenes [Plot question / answer distribution]

Limitations: although ScanQA is designed for 3D-VQA, most of the questions do not require 3D information to answer (ie could be answered with 2D images only). This is because the questions are generated from the 2D images of the scenes (check this fact?). Ie. no questions about distances, paths, dimensions, geometry, etc.

Chapter 4

Method

This project addresses the task of Visual Question Answering within 3D indoor environments (3D-VQA). 3D-VQA consists in answering open-vocabulary, free-form questions about an environment. For example, given segmented RGB-D images of a hallway and the question ”Where did I put my keys?”, the 3D-VQA model should be able to locate the keys in the scene and output an accurate response to the question, such as ”On top of the drawer at position (x, y, z)”.

The two main challenges of the 3D-VQA task are 1) to create a rich scene representation, often starting from RGB-D frames and 2) to combine this scene representation with a text query to produce an accurate output answer. The 3D-VQA task therefore requires merging different data modalities in a multimodal model to condition the output answer on the given question-scene input pairs.

While current research often claims to utilize scene graphs as scene representations, it solely relies on node information, and does not utilize edge information in the VQA model. This means that current methods do not leverage the full potential of scene graphs. Furthermore, existing VQA models often rely solely on pre-trained language-vision models, such as CLIP, to bring the question and scene representation into the same embedding space, lacking a learning mechanism.

To mitigate these limitations, we propose:

1. a dataset of scene graphs for the ScanNet dataset (ScanNet-SG), paired with questions about the graphs from the ScanQA dataset. This is, to our knowledge, the first dataset of question-scene graph pairs for 3D indoor environments.

2. an end-to-end 3D-VQA classification model which combines graph and text modalities to answer questions about the scene.

We emphasize that our proposed VQA model is a classificaton model, rather than a generation model. This means that the model does not aim to output free-form answers, but rather to solve the task of selecting the object in the scene which is most relevant to the question posed. For example, the question ”What color is the chair by the desk” should return the instance of ”chair”, rather than the answer ”the chair is black”. In other words, the proposed model returns the object in the scene which is most relevant to the question (where someone should look to answer the question), rather than the actual answer to the question. This classification task is easier to evaluate and interpret than the generation task, and a good starting point for evaluating the model's ability to understand question-scene pairs. Furthermore, solving the classification task first ensures that we can train a model that does not rely on textual priors to answer questions, since the model must focus on instance accuracy rather than semantic accuracy. This forces the model to learn to understand the scene and the question. Our model can be easily extended to a generation model by adding a decoder component to the model.

The following sections detail the scene graph generation method used to obtain the ScanNet-SG dataset, as well as the proposed 3D-VQA classification models.

4.1 Scene Graph Generation

[Add that we use scannet labels to embed rather than nyu40 because its more descriptive / larger more precise vocabulary than nyu40] [specify that for clip we are using the huggingface implementation]

[Add figure here] (each step summarized)

We aim produce semantically meaningful graph representations of the ScanNet indoor scenes. [best possible preserve scene structure / translate all the important pointcloud information into a scene graph] To achieve this, we let graph nodes represent segmented objects in the scene, and graph edges represent proximity, such that nearby nodes are connected by an edge. We note that there is no such ”ground truth” representation, since scene graph generation requires some design choices. More specifically, to construct a graph representation $G = (V, A, E)$, the following components must be defined:

1. **Node features,** $V \in \mathbb{R}^{n \times D_V}$**:** the graph contains n nodes with D-dimensional embedding, which encodes information about the object it represents.

2. **Adjacency,** $A \in \mathbb{R}^{n \times n}$**:** the graph edges are defined in an $n \times n$ adjacency matrix, which encodes information about which nodes are connected. Node connectivity can represent any type of node similarity, such as proximity, visual similarity, semantic similarity, functional similarity, etc.

3. **Edge features,** $E \in \mathbb{R}^{|A| \times D_E}$**:** Optionally, edge embeddings E (which represent edge features), can also be defined. An example of edge labels could be the description of the relation between nodes, such as ”next to”, ”inside”, ”above”, ”same color”, ”same function”, ...

The following sections detail the scene graph generation method used in this paper.

4.1.1 Node features

We initialize the node embeddings with CLIP-embedded images of the object of interest. However, as each object may appear in many RGB-D frames under different views, we must select which views to embed, and how to combine these into a single node embedding.

View Selection Given the high frame rate of the image capture, we discard every other RGB-D image in each scene to speed up the data processing. The best views for each object are then selected by calculating a ”view score”: firstly, we use the 3D pointcloud segmentation to obtain the 3D-coordinates of a tight, axis-aligned bounding box for each object in the scene. The coordinates of the bounding box corners are then projected into each RGB-frame. Projecting the bounding box corners into each frame allows us to determine whether the object is fully visible in the frame, or only partly visible. Then, for each object in the scene, we rank views according the following score s:

$$s = \frac{n}{w \times h} \times \frac{b}{8},$$

where n is the number of pixels occupied by the object in the frame, w and h are the width and height of the frame and b is the number of bounding box corners visible in the frame (out of 8 possible corners). The first term in this score is the ”size term”, which represents how much space the object takes up in the frame (the larger the object, the better). The second term is the ”visibility term”, which represents how much of the total object is visible in the frame (the entire object is better than only part of the object). Hence a score close to 1 signals a close-up view of a fully visible object, whereas a score close to 0 signals either a distant view, or a partial view, or both. This method therefore attributes low scores to zoomed-in partial views and distant views, and large scores to large, complete views of an object. We note that weights could be added to both the ”size” and ”visibility” components of the score to weigh their relative importance.

+ Using the 3D bb corner projections assumes that the entire object is visible in the frame dataset (if only half an object was photographed, the bb will obviously be smaller than GT since the 3D segmentation is obtained from the 2D segmentation)

Cropping For each object, we then select the top-k views and crop these around the object of interest. Three different cropping methods were tested: 1) a tight, non-square crop around the object with all pixels not belonging to the object blacked out, 2) a tight crop, non-square around the object with no further changes, 3) a square crop around the object, resized to 224×224.

Figure XX illustrates these cropping methods:

Embedding Since CLIP was trained on square images and can only embed square images, for cropping methods 1) and 2) the standard CLIP pre-processing was used: resizing the shortest edge to 224 and randomly selecting a 224×224 crop within the image. Then we CLIP embed the top-k views and aggregate them with a score-weighted average, yielding a single CLIP-embedding for each object that captures view variability and gives a more rich embedding to the object node.

Section xx quantitatively compares the weighted CLIP embeddings obtained using each of the cropping methods described above.

+ justify why we use CLIP (brings text + images into the same embedding space, hopefully in semantically meaningful, should be a good initialization for a model).

4.1.2 Adjacency

Adjacency matrix is the component of the graph which allows us to encode/transfer 3D information of the scene into the graph. The adjacency matrix in the provided dataset was chosen to represent spatial proximity of nodes. Therefore, the shortest distance between each object should be calculated, and edges should exist only between objects with a distance less than some predetermined heuristic threshold. Several methods were explored to calculate or approximate the shortest distance between objects. Figure XX illustrates these methods.

Naive approach: Pointcloud-based shortest distance First, a naive approach was implemented: the shortest distance between two objects was calculated by computing the distance between all 3D points of 2 object pointclouds. Although this is the most accurate measure of shortest distance, it is too computationally expensive and also very sensitive to outliers. We therefore decided to approximate the shortest distance between two objects rather than accurately computing it.

Approximation approach: Axis-aligned bounding box shortest distance To approximate the shortest distance between two pointclouds, we obtain the axis-aligned bounding box for each object from the 3D scene segmentation. The shortest distance between two axis-aligned boundign boxes can then be calculated as:

A threshold of 0.2 was selected heuristically. We note that this measure is not very precise. Results can be changed by varying the threshold.

The adjacency matrix is symmetrized to ensure bidirectional relationships between nodes. Additionally, self-loops are added to represent self-relationships.

4.1.3 Edge features

We compare two learning-based approaches to annotate/predict the edge label of the edges determined above.

Random Forest

MLP

4.2 VQA Models

Figure XX illustrates the general 3D-VQA classification method. VQA models take as inputs a question and a scene graph (generated as described above), and outputs the most relevant object in the scene to the question. Regardless of architecture, any VQA model must have the following components: 1) a scene encoder, 2) a question encoder, 3) object selection head, which combine scene and question encodings to select the output object.

We explore solutions for each of these components, and compare the performance of three distinct families of model architectures: a baseline model (no learning), a GNN-only (learning only for scene encoding) model, and a transformer-based model (learning for scene encoding, question encoding and selection head). Some of the components are shared across models, this helps us identify which components of the final proposed model are actually helpful.

The following sections detail the architecture and design choices of each model.

4.2.1 Baseline Model

The baseline model is used to establish a lower bound on the performance of the VQA task. The baseline model is illustrated in Figure XXX. It does not require any learning, and relies solely on the pre-computed CLIP embeddings of the objects in the scene and the question. The model consists of the following components:

Scene Encoder: The scenes are encoded as the set of its ScanNet-SG scene graph nodes (without edges, obtained as described in section XXX).

Question Encoder: The question encoding is the sentence-level CLIP embedding of the question.

Object Selection Head: The model computes the cosine similarity between the question embedding and the node embeddings of the scene graph. The node with the highest similarity is selected as the output object.

While this baseline model is a simple and common approach to the VQA task [ref], it has some important limitations. Firslty, the scene encoder encurs a large information loss: by discarding edges, it does not encode any 3D or structural information about the scene. It therefore does not capture the overall room layout, and lacks contextual awareness. This implies that objects with similar appearance may have indifferentiable encodings, regardless of their context and neighbouring objects, and that the overall scene encoding does not change if even if objects are moved around. For example, the baseline model may not be able to differentiate between a chair in the middle of the room and a chair in the corner of the room, as the scene representation would be the same. This means that while semantic accuracy might be high, instance accuracy is expected to be low. Furthermore, the model has no learning mechanism, meaning that the scene representation cannot adapt to new questions.

4.2.2 GNN-only Model

To address these limitations, we explore the GNN-only model, which uses a graph neural network to learn better node embeddings. The model is illustrated in Figure XXX. The model consists of the following components:

Scene Encoder: The ScanNet-SG scene graphs are passed through a dimension-preserving GNN. The GNN updates node embeddings iteratively by aggregating information from neighbouring nodes. This ensures each node becomes aware of its context.

- What type of GNN architecture? How many layers? How many hidden units? What activation function? What aggregation function? What loss function?

Question Encoder: Same as the baseline model.

Object Selection Head: Same as the baseline model.

The GNN-only model is expected to perform better than the baseline model, as structural information is added to the scene encoding. This model should therefore better differentiate between objects with similar appearance but different context. Furthermore, the scene encoder is learned, meaning that the model can learn to adapt scene encodings to better match the question encoding. In other words, the model can learn to align the question and scene embeddings in a learned way, rather than a fixed way as in the baseline model. However, the GNN-only model still has some limitations. Firstly, the model is still limited by the question encoding and object selection, which are not learned. This means that the model may struggle with questions that are not well-aligned with the learned scene graph.

4.2.3 Transformer-based Models

The transformer-based model incorporates learning for the question encoding and object selection head. It is illustrated in Figure XXX. The idea was to create a model with more degrees of freedom than the GNN-only model: instead of trying to align the fixed question embeddings and learned scene embeddings using cosine similarity, the model learns to predict a score for each object instance in the scene based on the learned scene and question embeddings. High score indicates high relevance to the question.

The model is composed of four main components: 1) a question encoder, 2) a scene graph encoder, 3) a cross-attention mechanism, and 4) a classifier.

The question encoder is a transformer-based model which encodes the question into a fixed-size vector. The scene graph encoder is a graph neural network which encodes the scene graph into a fixed-size vector. The classifier is a multi-layer perceptron which takes the question and scene graph embeddings as input and outputs the most relevant object in the scene to the question.

How many layers? How many hidden units? What activation function? What aggregation function? What edge update function? What loss function?

4.2.4 Loss function

Cross-entropy loss Multi-object Cross-entropy loss Focal loss

Chapter 5

Experiments

In this section, we detail experimental methods and results.

5.1 Scene Graph Generation

In order to train a successful 3D-VQA model, we must ensure that its input data, namely generated scene graphs and questions, are of high quality. This section empirically evaluates the impact of different design choices on the quality of the generated scene graphs. We evaluate the design choices for scene graph nodes and edges separately, and choose the best performing parameters for the proposed scene graph dataset. For scene graph nodes, we evaluate the effects of different cropping methods, k-values, and visibility scores on the quality of the node embeddings. For scene graph edges, we evaluate the impact of different threshold values on the quality of the edges in the scene graphs.

All experiments in this section are performed on a smaller subset of the dataset, consisting of XXX scenes, with a total of XXX objects across scenes.

5.1.1 Node quality evaluation

We evaluate the impact of different cropping methods, k-values and visibility scores on the quality of the generated node CLIP-embeddings. The following paragraphs detail the methods and parameters considered for each of these design choices.

Cropping method: **Add the information below in a table, rather than a list?**

We compare the impact of the following cropping methods on node embedding quality (illustrated in Figure XXX):

1. a tight, rectangular crop around the object with all background pixels masked out. With this method, the object is centered in the image, and the background is removed. However, CLIP was trained on square images with no background removal. To adapt rectangular crops to CLIP, we use the resizing method described in the original CLIP paper [XXX]: the crop is resized to 224 in its minimum dimension, and then randomly cropped to a 224 × 224 square.

2. A tight, rectangular crop around the object with no further changes. With this method, the object is centered in the image. However, the background is not removed, meaning other surrounding objects could contribute to the embedding, and the image is not square. The rectangular crops are resized as in the original CLIP paper [XXX].

3. A square crop centred around the object, resized to 224×224. With this method, the crop is square and correctly sized, and no further processing is needed. However, the background is not removed and the non-tight crop means other surrounding objects might contribute to the node embedding.

4. A tight, rectangular crop, resized to 224×224. With this method, the crop is tight around the object (fewer surrounding objects included in the crop). However, the resized crops distort the image, and CLIP was not trained on distorted images.

Add cropping methods figure

k-value: We also evaluate the impact of different k-values on the quality of the scene graph nodes. The k-value determines how many images of the object will be considered when generating the scene graph node. A higher value of k could result in a more accurate representation of the object, but could also introduce noise as different views may have different lighting conditions, occlusions, etc, leading to very different embeddings, and averaging them out could result in a less accurate representation. Increasing k-value also highly increases the computational cost, since computing the CLIP-embedding over more images is more expensive. It is therefore important to find the optimal k-value that balances these trade-offs.

Visibility score:

TODO: viz score experiment - visibility score with equal weights, multiplied - visibility score with equal weights, summed - visibility score with higher weight for object size, summed - visibility score with higher weight for object visibility, summed - similarity score

We need to correlate text and image so the image should be as close to its relevant text description as possible so we can query the relevant object.

We would like to have node embeddings that are as ”pure” as possible. This means that the embedding of the object should be as close as possible to the embedding of the object label, and not be influenced by surrounding objects. Context awareness can be integrated later but the input graph should be as ”clean” as possble.

To evaluate the comparative quality of these cropping methods and k-values, we perform the following experiment: first, we generate scene graphs using each of the three cropping methods, with $k = 1$, $k = 3$ and $k = 10$ respectively. We therefore have a total of 9 different scene graph node generation methods. Then, we CLIP-embed all the object labels in the scene, and calculate the similarity between the embeddings of the object and the embeddings of the object labels. For each object, we select the label with the highest similarity, and compare it to the ground truth label. We calculate the accuracy of the scene graph nodes for each of the 9 methods, and choose the best performing method for the final model. Note that we only use semantic accuracy (since there is no context-awareness mechanism or learning when we consider nodes only with the baseline model) to evaluate the quality of the scene graph nodes. We therefore ignore whether the correct instance of the object was most similar, but rather whether the correct object label was selected.

This experiment allows us to determine the best cropping method and k-value for the final model since we can quantify the percentage of nodes that are embedded close to their label in the CLIP embedding space (and hence truly representative of the object). For example, if the node CLIP-embedding for the object ”sink” is actually semantically closer to the word ”mirror”, this would be a poor quality node, and would likely lead to incorrect answers in the 3D-VQA model. Table XX shows the results of this experiment.

Table 5.1: with multiplicative similarity score

	k-value		
Cropping method	1	3	10
Tight masked crop	49.5	50.1	52.1
Tight crop	57.6	58.9	60.3
Square crop	57.1	58.3	**62.2**
Tight resized crop	56.2		

complete table

add results for method 3, k=25

All methods benefit from a higher k-value, but the tight crop method with $k = 10$ performs best. This is likely because...

The best score achieved (Table XXX) is fairly low, which indicates that the CLIP-embedding of the object is not very close to the CLIP-embedding of the object label. The experimental examples in Figure XXX provide some insight into why this might be the case.

Interpretation **Include examples of bad embeddings vs. good embeddings**

There are some inherent limitations to the CLIP method, but this is still a good starting point since it allows us to already compare the images of the object with text.

The experiments also highlight the necessity of including some learning into the model, because the CLIP-embedding of the object is not very close to the CLIP-embedding of the object label. And this label task is easier than the question task, as shown in Figure XXX.

can add experiment to show that CLIP(answer) accuracy is lower when CLIP(question) is lower. To do so, do an exp with the worse nodes+edges and one with the best node+edeges and show that the accuracy is lower for the worse nodes+edges. Ie. this experiment is a good way to evaluate whether the node embedding is good and to select the best node embedding

5.1.2 Edges

threshold value experiment Simple GNN model -¿ show that chosen threshold is better than random edges. Also show that edges can have a huge impact on the final answer –¿ must be chosen carefully.

5.2 Visual Question Answering

WRITE PLAN FOR EXPERIMENTS - WHICH ONES NEED TO BE REDONE / INCLUDED / MODIFIED

Object selection block: CLIP(question) vs CLIP(answer) –¿ similarity with static embeddings does not work well - we show that CLIP(question) does not really capture the same meaning as CLIP(answer)

Hyperparameter tuning gives: LR Batch size Adam optimizer Loss function Number of epochs

Dataset split:

Evaluation metrics:

5.2.1 Results

compare performance of all 3 models with the best node embeddings determined above + the best threshold determined above. + can include GraphVQA results (with no training)

analyse results + why each method is better / worse

Overall, there is a overfitting problem. to attempt to overcome this, we report the results of the best performing model with the following changes: normalization dropout L2 regularization changing loss function, ...

downsizing embedding word-level embedding (rather than last layer)

Chapter 6

Discussion

The discussion section gives an interpretation of what you have done [1]:

- *What do your results mean?* Here you discuss, but you do not recapitulate results. Describe principles, relationships and generalizations shown. Also, mention inconsistencies or exceptions you found.
- *How do your results relate to other's work?* Show how your work agrees or disagrees with other's work. You can rely on the information you presented in the "related work" section.
- *What are the implications and applications of your work?* State how your methods may be applied and what implications might be.

Make sure that the introduction/related work and the discussion section act as a pair, i.e. "be sure the discussion section answers what the introduction section asked" [1].

Dataset size: The dataset is relatively small, which might have an impact on the performance of the model - might not be able to generalize well to unseen data, overfitting to the training. For reference/comparison, the VQAv2 dataset has 1.1M questions, and the GQA dataset has 22M questions (2D VQA datasets). Yet we only have 20K

Dataset contents: most quesitons can be answered with 2d information only...

Chapter 7

Conclusion

List the conclusions of your work and give evidence for these. Often, the discussion and the conclusion sections are fused.

Appendix A

The First Appendix

In the appendix, list the following material:

- Data (evaluation tables, graphs etc.)
- Program code
- Further material

Bibliography

[1] R.A. Day and B. Gastel. *How to Write and Publish a Scientific Paper*. Cambridge University Press, 2006.