ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

3D Scene Understanding through CLIP Embedding and Instance Segmentation Fusion

Semester Thesis CVG Lab
SS 2023

Guangda Ji
guanji@student.ethz.ch
ID: 21-951-249

Advisors: Silvan Weder
Professor: Prof. Marc Pollefeys

Abstract

This project presents a novel 3D instance segmentation and detection pipeline by fusing 2D instance segmentations and features into 3D. The pipeline involves 3D scene reconstruction, 2D instance mask extraction and CLIP feature extraction. A novel graph-connecting algorithm is proposed to address the 3D instance label assignment problem based on 2D instance segmentation. Remarkably, our algorithm achieves finer-grained and more intact segmentation results compared to an existing method. For semantic class detection, we employ a CLIP captioning module. In essence, our pipeline serves as a bridge connecting the realms of 2D and 3D.

Contents

List of Figures

List of Tables

1. Introduction

Significant improvements have been made in the field of computer vision in recent years, particularly in 3D scene understanding. The ability to accurately segment and detect objects hold profound implications for many applications, such as robotics, autonomous driving and augmented reality. One particular challenge in this domain is to fuse information from 2D into 3D. In this project, we try to apply high-performance 2D computer vision models to 3D scene understanding tasks, without any training of neural network. Our main contributions of this project are:

1. We propose a pipeline of 3D instance segmentation and detection by fusing 2D instance segmentation and dense CLIP features. Instance segmentations are obtained by clustering voxels into patches and merging them together by our graph-connecting algorithm. Instance detection is accomplished by Grounded SAM and CLIP embedding captioning.

2. Our graph-connecting algorithm for fusing 2D instances overcomes the weakness of over-segmentation by PanopticFusion [1], and utilizes 2D instances more efficiently than PanopticFusion.

3. We provide an open-source realization of TSDF volume with feature fusion.

4. We provide the first open-source implementation for PanopticFusion.

The code is available at `https://github.com/quantaji/feature-and-instance-fusion`.

2. Related Works

OpenAI's CLIP [2] is the inspiration of this project. It employs contrastive learning to generate a paired feature vector $f_{\text{img}}, f_{\text{txt}}$ for images and text. Images that correspond closely with the textual description yield higher cosine similarity scores. This is a huge step forward because of the flexible compositional ability of language.

Our project starts with an aim of building a dense CLIP feature at the per-voxel level for 3D scenes. Although there are existing work [3] that learns a per-scene 3D dense CLIP representation, there is currently scarce methods capable of accomplishing this task in an on-line manner, specifically, the integration of dense 2D CLIP features into 3D scenes.

Substantial studies [4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20] have been conducted to address open-vocabulary semantic segmentation using CLIP features. LSeg [20] employs a straightforward approach, fine-tuning CLIP on pixel-level text-image alignment, resulting in good results. [4, 13] try to modify the original CLIP architecture to produce dense output. GroupViT [8] designs a new attention module that cluster pixels into different semantic classes. [10, 6] also apply a similar idea. [21, 18, 17, 14] try to combine mask proposal and CLIP together to give a dense feature. [5, 15] goes a step further to add mask module bypath on CLIP network. [19, 12, 9, 7] utilize multi-modal transformers to tackle region-text grounding. Nonetheless, these approaches are impractical for us as the textual input is integrated at an early stage, before the dense feature is generated, so that we cannot extract a text-independent image feature. Additionally, compatibility with the original CLIP text encoder is a prerequisite for our pipeline. Having taken all these considerations into account, we choose LSeg as our dense CLIP feature model.

During the course of this project, a new approach known as ConceptFusion [22] was introduced. Like the aforementioned related works, it leverages a mask generator and the CLIP model to obtain dense 2D CLIP features. Then this pipeline fuse the features into 3D by gradslam [23]. We will give an in-depth introduction, analysis on ConceptFusion and compare it with our approach in Chapter 6.

PanopticFusion [1] is another work that is heavily studied in this project. It aligns closely with our objective of integrating 2D instance segmentation into 3D. We will give a thorough introduction to PanopticFusion in Chapter 8 along with a detailed comparison with our graph-connecting algorithm.

Our pipeline relies heavily on instance segmentation models. Specifically, we use the Segment-Anything Model (SAM) [24], a highly effective model developed by Meta, which provides precise and detailed instance segmentation masks. We also use Grounded SAM, an instance segmentation model that combines of SAM with Grounding DINO [25]. The latter is a model that generates both the bounding box and tag name for each instance. These two models play a crucial role in our pipeline.

3. Main Results

Our pipeline, depicted in Figure 3.1, is designed for 3D instance segmentation and detection using 2D data. The comprehensive procedure is outlined as follows:

1. **3D Scene Reconstruction:** We employ Truncated Signed Distance Function (TSDF) fusion to build a 3D voxel volume for the scene using depth images.

2. **Instance Mask Extraction:** For each color image, we use Grounded SAM [25] to extract instance segmentation masks along with their categories.

3. **CLIP Feature Extraction:** From each 2D color image, We extract dense CLIP feature using LSeg [20]. The features are then fused onto TSDF volume in the same way as fusing RGB channels. This results in a 512-dimensional feature vector for each voxel, denoted as $f_{\text{clip}(x,y,z)}$ for voxel (x, y, z).

4. **Random Feature Fusion:** Additionally, We fuse a mask-consistent 128-dimensional random feature onto the TSDF volume. The 2D mask-consistent random feature is defined as

$$f_{\text{rand2d}}(x, y) = \sum_{i \in M} \mathbb{1}\{(x, y) \in \text{Mask}_i\} \times \vec{e}_i, \text{ where } \vec{e}_i \sim \mathcal{N}(0, I) \in \mathbb{R}^{128}. \tag{3.1}$$

   The resulting 3D fused random feature is denoted as $f_{\text{rand}(x,y,z)}$.

5. **K-Means Clustering**: We apply the K-Means algorithm to all random feature $f_{\text{rand}(x,y,z)}$ with $K = 1024$. This clusters the voxel into patches, which are labeled as P_i.

6. **Graph Construction for Instance Segmentation:** We construct a graph in which the nodes are K-means patches. The edges are linked based on the outcomes of graph-connecting algorithm (See Algorithm 1). Intuitively, this algorithm use previous instance segmentation in step 2 to performs a Bayesian inference to determine whether two patches are from the same instance. The connected component of the patch graph is then employed as the 3D instance segmentation solution, with the instance ID defined as $\text{Id}(x, y, z)$.

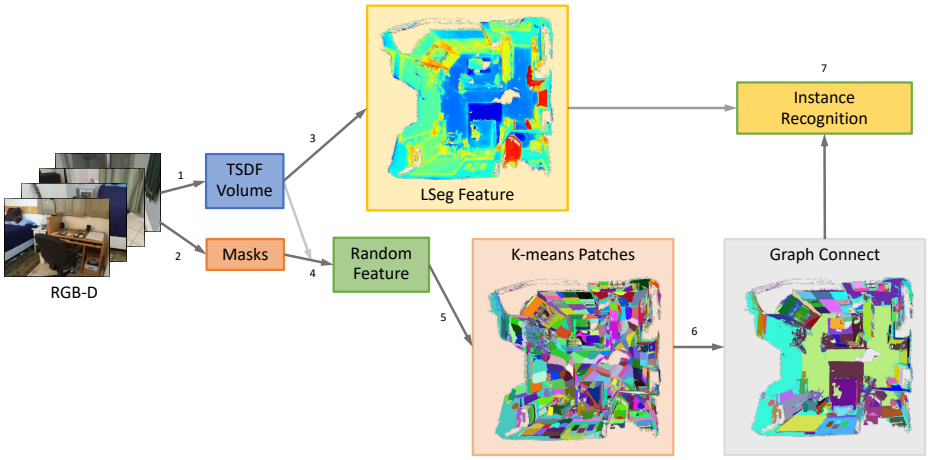
7. **Semantic Information Acquisition:** For each 3D instance, we obtain its semantic information by either (1) decoding a caption from its mean LSeg feature, or (2) counting the mode category in Grounded SAM masks and use it as the instance's category.

Figure 3.1(b) provides a partial list of the instances and their semantic information obtained through our pipeline. Figure 8.1(d) illustrates instance segmentation on additional scenes. Our pipeline has an open-vocabulary detection capability, allowing detection of instances beyond a fixed set. Furthermore, it delivers more granular instance segmentation compared to PanopticFusion [1] (refer to Figure 8.1 for detailed comparisons).

The subsequent chapters provide detailed discussion on each component within our pipeline. Chapter 4 analyzes the semantic segmentation ability of LSeg feature. Chapter 5 discusses decoding category information from LSeg feature using captioning models. In Chapter 6, we conduct a detailed analysis of ConceptFusion [22], exploring the reasons of its unsuccessful integration into our pipeline. Chapter 7 discusses our efforts in instance segmentation and extracting relationships between instances. Chapter 8 provides a comprehensive introduction to PanopticFusion and how it influenced the development of our graph-connecting algorithm. Chapter 9 concludes with a discussion.

Algorithm 1 Graph-connecting algorithm for K-Means patches

Initialize positive and negative confidence matrix $w_+(i,j) \leftarrow 0, w_-(i,j) \leftarrow 0$
for each frame, access previous generated Grounded SAM instance map $\mathrm{Id}(x,y)$, **do**
  Find pixel correspondence to the voxel $\mathrm{VoxID}(x,y)$.
  Compute the confidence score for each patch:
  $c_1(P) := \mathrm{Clip}(\frac{\mathrm{NumVoxel}(P)}{100}, 0, 1)$, voxel size criterion,
  $c_2(P) := \mathrm{Clip}(\frac{\mathrm{NumVoxelInFrame}(P)}{\mathrm{NumVoxel}(P)/3}, 0, 1)$, voxel percentage criterion,
  $c_3(P) := \mathrm{Clip}(\frac{\mathrm{NumPixelInMostFreqMask}(P)}{0.0025 \times \mathrm{FrameArea}}, 0, 1)$, pixel percentage criterion,
  then the overall confidence is $c := c_1 \times c_2 \times c_3$.
  for Patch pair (P_i, P_j) appears in this frame, and are of the same instance, **do**
    $w_+(i,j) \leftarrow w_+(i,j) + c(P_i) \times c(P_j)$.
  end for
  for Patch pair (P_i, P_j) appears in this frame, and are of different instance, **do**
    $w_-(i,j) \leftarrow w_-(i,j) + c(P_i) \times c(P_j)$.
  end for
end for
if Patch P_i and P_j are neighbor and $\log(\frac{w_+(i,j)}{w_-(i,j)}) > \theta_1 = 2.0$ and $w_+(i,j) > \theta_2 = 1.0$, **then**
  Build an edge between patch P_i and P_j, otherwise leave blank.
end if
return the connectivity graph.



(a) Pipeline

ID	Size	DeCap Detection	Sim	1st freq (%)	Sim	2en freq (%)	Sim
20	4e5	a wall with the word	0.89	curtain (33.5)	0.92	shower curtain (13.8)	0.78
14	2e5	a floor of a floor of a room	0.90	floor (81.3)	0.89	bag (4.2)	0.72
0	1e5	a couch	0.93	carpet (24.1)	0.88	pillow (16.5)	0.86
65	5e4	a bed	0.96	bed (56.3)	0.99	blanket (7.1)	0.78
80	5e4	the wall of a wall with a lot of other	0.91	tile wall (49.5)	0.84	kitchen (26.7)	0.75
26	5e4	the wall of a wall with a single	0.90	wall (35.0)	0.86	remote (27.2)	0.73
27	4e4	a bicycle	0.97	bicycle (81.1)	0.98	stool (4.7)	0.78
4	4e4	some kind of door	0.93	glass door (25.8)	0.86	shower (25.6)	0.81
40	3e4	a cabinet of the	0.93	shelf (66.1)	0.94	closet entertainment center (6.2)	0.82
128	3e4	a wall with some kind of door	0.89	doorway (19.7)	0.85	bathroom accessory (15.2)	0.85
23	3e4	a curtain	0.97	curtain (63.1)	0.99	ceiling (23.6)	0.76
				.....			

(b) Instance Detection

Figure 3.1.: **fig. 3.1(a) shows the whole Pipeline.** 1. Fusing depth map into TSDF volume. 2. Extracting 2D instance segmentation by Grounded SAM. 3. Fusing 2D dense clip feature (LSeg) into 3D. 4. Fusing 2D random mask feature into 3D. 5. Obtaining sub-instance level patches from K-Means over random mask feature. 6. Building connectivity graph over patches and connect same instance patches. 7. Extracting instances' semantic class using Grounded SAM masks or decoding from mean LSeg feature. **fig. 3.1(b) shows the semantic information of each instance.** Label ID and voxel size are listed as basic information. We use two methods to get the semantic information of the instance: Decoding averaged LSeg feature of instance with DeCap [26], or using top frequent words in Grounded SAM masks. The quality of recognition are evaluated by the cosine similarity between the CLIP text embedding of instance information and the averaged LSeg feature. For the second methods, the percentage of most frequent word is another evaluation of recognition quality as it represents the purnity of recognition.

4. TSDF Fusion of Dense CLIP Feature

In this chapter, we present a comprehensive examination of the 3D fused Dense CLIP feature.

When fusing features to TSDF volume, we perform an online update, $f_{t+1} = \frac{W_t \times f_t + w_{t+1} \times f_{\text{new}}}{W_t + w_{t+1}}$. This operation basically computes the weighted average of all feature vectors projected onto the given voxel, $f_{3\text{d}} = \frac{\sum_i w_i f_i}{\sum_i w_i}$. In our implementation, we use the simplest case where $w_i = 1$. A natural question arises: do these 3D fused CLIP features retain their responsiveness to CLIP text feature queries as they do in the 2D scenario? We conduct three types of text queries to address this:

- **Multi text query**: We compute CLIP text features for multiple text queries, typically category names. Each queried CLIP feature is then assigned to the most similar text query. This technique is originally employed in LSeg [20] for generating 2D semantic segmentation.
- **Single text and "other" class query**: This involves computing two CLIP text features. One corresponds to the text to which we desire the CLIP feature to respond, while the other encodes the text "other". We use the softmax of cosine product of image and text feautre to denote similarity score.
- **Single text similarity score query**: In this case, only one CLIP text feature is considered. We compute its cosine similarity with the queried CLIP features, and visualize this score as a heat map. This query method is used by ConceptFusion [22].

Multi text query. This query type is particularly suited for semantic segmentation within a predefined set. Figure 4.1(d) illustrates the segmentation based on the NYU40 scheme along with its Intersection over Union (IoU) ratio. The obtained results demonstrate the efficacy of the 3D fused features, as they provide segmentation boundaries that are reasonably accurate. However, it is worth noting that they still fall short in comparison to state-of-the-art 3D-based learned segmentation models.

Single text and "other" class query. In Appendix A, a comprehensive examination of various query types is presented, encompassing noun phrases, verb phrases, adjectives, logical combinations, differing scopes (e.g., broader: "living room", narrower: "specific object component"), relative positions, and implicit inferential text. The findings demonstrate that our CLIP features accurately respond to nouns, verbs, and adjectives, in which cases the semantic interpretation is most straightforward. However, when subjected to logical combinations, scopes, or relative positions, CLIP features yield responses that are entirely inaccurate. The heat map visualization shows that CLIP features exclusively response to nouns, verbs, and adjectives within the queried sentences, but remain ignorant of the overall semantic combination. This outcome is in expectation, given that CLIP is not trained with object grounding, and is also revealed in previous work [27]. Nonetheless, the absence of awareness of relative position impedes us from progressing beyond the knowledge of object existence to comprehending their interrelationships. As we can see in Chapter 7, even for 2D images this is a challenging task in the present state of research.

(d) Semantic Segmentation (NYU40)

cls	IoU_{3d}	IoU_{2d}
unknown	7.68	0.00
wall	62.62	66.55
chair	0.00	0.00
books	0.00	0.00
floor	57.10	61.54
door	28.73	22.67
otherprop	0.00	0.00
window	54.77	46.20
table	17.77	16.36
otherfurniture	0.00	0.00
pillow	0.00	0.00
picture	0.00	0.00
ceiling	73.86	86.85
box	0.00	0.00
cabinet	45.35	40.47
desk	0.00	0.00
shelves	2.01	3.31
towel	0.00	0.00
sofa	79.51	76.32
sink	35.96	25.97
lamp	0.00	0.00
bed	61.72	72.00
bookshelf	0.00	0.00
mirror	15.85	8.73
curtain	59.19	60.90
whiteboard	0.00	0.00
toilet	39.16	35.93
bag	0.00	0.00
clothes	0.00	0.00
night stand	0.00	0.00
television	37.14	41.13
dresser	0.00	0.00
refridgerator	69.22	79.79
shower curtain	0.00	0.00
bathtub	0.00	0.00
counter	25.51	31.92
otherstructure	0.00	0.00
floor mat	0.00	0.00
paper	0.00	0.00
person	0.00	0.00
blinds	0.00	0.00

(e) IoU

Figure 4.1.: **Left:** Semantic segmentation using LSeg dense feature. Voxels are assigned to semantic class with highest text-image similarity. **Right** Intersection over Union (IoU) ratio with respect to (1) ground truth 3D segmentation using meshes provided by ScanNet, (2) Voxel segmentation obtained by fusing 2D ground truth semantic segmentation.

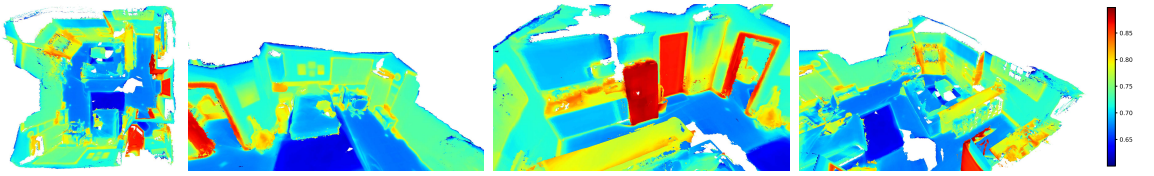


Figure 4.2.: Cosine similarity between LSeg features and embedding of query text, "*The door is near the refrigerator.*" This figure shows single vector query gives non localized heatmap, and therefore, cannot be used as reliable localization tools.

Single text similarity score query. The efficacy of a single text and "other" class query in yielding localized and reasonable responses rise the question whether a single vector is sufficient to achieve comparable outcomes. The answer is negative. As depicted in Figure 4.2, although the desired objects are visually delineated in red within the scene, the process of thresholding for localizing these objects is contingent on specific cases. The similarity score of a single vector shows continuous variations across voxels and fails to yield an obvious localization boundary unless subjected to manual thresholding.

CLIP features enable image features to response to arbitrary text query, making open-vocabulary detection promising. However, the querying methods outlined above still require a pre-defined set of text, making the query pseudo-open-vocabulary. Hence, in the next chapter, we will employ CLIP feature captioning as a strategy to circumvent the need for a fixed predefined text query, in order to achieve true open-vocabulary detection.

5. CLIP Feature Captioning

In this chapter, we aim to validate the idea of extracting open-vocabulary class information using CLIP feature captioning model.

While there are many captioning models conditioning on CLIP feature as input, we need two additional requirements to integrate them into our pipeline: (1) The model must be trained on normalized CLIP feature. Our feature fusion pipeline require the features to be normalized, in order to prevent unpredictable weight factor. (2) The model must be compatible with the `ViT-B-32` version of OpenAI's CLIP, which gives a 512 dimensional feature vector and also forms the basis of LSeg model. Alternative CLIP-like models such as [28] may not align effectively with LSeg features and caption models trained with these clip models should therefore be avoided.

We find DeCap [26] and ClipCap [29] meet our requirements. It is noteworthy that ClipCap performs well in generating captions on the original CLIP features, whereas DeCap works better in captioning LSeg features. Consequently, we choose to ClipCap for original CLIP features, and DeCap for LSeg features.

Figure 5.1.: An example of captioning from dense CLIP feature (LSeg). The words *a table* and *a blanket* are output of DeCap, given masked-averaged or single pixel feature as input.

5.1. Captions of Linear Combination of Features

Figure 5.1 shows an example of captioning LSeg feature. DeCap successfully decode the semantic class *table* and *blanket* (albeit with a slight deviation). However, we need to further investigate whether captioning ability still remains after line combination, since our 3D fused feature is also a linear combination.

In fig. 5.2, we present examples of linearly combining three vectors, each representing distinct semantic meanings. Although the decoded text does not perfectly indicate the original semantic meaning, at least we observe elements of the semantic class in the text. In practical

A B C

Decoding:
A: A wooden table with a wooden table cloth on top of it.
B: A black coffee mug sitting on top of a table.
C: A laptop computer with a blue screen on it.

(A+B)/2 : A wooden table with a cupboard and a black and white clock. (✗)
(B+C)/2: A black coffee cup sitting on top of a desk. (✓)
(C+A)/2 : A laptop computer sitting on top of a wooden table. (✓)

(A+B+C)/3: A computer mouse sitting on top of a wooden table. (?)
(A+B+2C)/4: A computer mouse sitting on top of a desk. (?)

Figure 5.2.: This figure shows the captions generated by ClipCap. This is a proof of concept, since the 3D CLIP feature are linear combinations of various 2D clip features.

case, we expect the fusion of LSeg features within a voxel to be more consistent than this test case, as we shall see in the next section.

5.2. Caption on 3D Fused LSeg Feature

Table 5.1 illustrates generated for LSeg features using DeCap. For the majority of semantic classes, the decoded text matches or provides synonyms for the ground truth class. This is a demonstration of the model's the open-vocabulary detection ability. Interestingly, we encountered instances such as *the ceiling of a lot of ceiling* or *the wall of a wall with a lot of other*, where the text, although grammatically incorrect, effectively reflects its origin from a combination of CLIP features. Moreover, in cases where the ground truth class lacks specificity, for instance, *otherstructure*, the decoded text gives *some kind of door*, which is also ambiguous. This could be attributed to the fusion of feature vectors from various semantic classes.

The table also includes caption of ConceptFusion features, which completely fails to convey the semantic information. We will give an in depth discussion on why it fails in the next chapter.

5.3. Attempts to Extract Positional Relationships

Even though CLIP features may not accurately respond to relative position queries, as we discussed in Chapter 4, we still want to try if captioning can provide some insights. In Figure 5.3,, we manually establish semantic relationships, such as *apple on the chair* or *mark cup on/under the table*. However, the generated captions either neglect the objects or fail to convey the correct positional relationships. Therefore, captioning from CLIP feature is also not the approach to acquiring relationship information.

Class	Size	LSeg Avg Cap	LSeg Random Cap	ConceptFusion Random Cap	ConceptFusion Random Cap
wall	2e5	the wall of a wall with a lot of other	wall with the side of a wall , one of the other	picture of a dark - haired woman 's lap top and door in the corner	blurry photograph of small cat in dark room with tie and remote
floor	2e5	a floor of a room with the floor	a floor of a room of the floor	blurry photograph of a persons dog 's lap top and remote	blurry photograph of a person 's beard and his baby cat in dark jacket
cabinet	2e5	a cabinet with the top	a television	picture of a dark - haired woman 's lap top and tie at home	picture of dark - haired person 's lap top , in dark room
curtain	9e4	a curtain	a curtain	blurry photograph of man 's hair in dark room	blurry photograph of man in dark suit and tie laying in television
otherprop	7e4	a back of a	a wall of a person	picture of a dark haired person 's hair with glass collar in corner	blurry photograph of person 's hair with his lap top and beard lying in the dark sky
sofa	7e4	a couch	a couch	blurry photograph of a person 's lap top and baby dog in the dark room	a dark haired cat 's couch in living room area
shelves	5e4	a cabinet with the top	a shelf that has been used	picture of woman in dark suit and tan striped tie at home	picture of woman in suit and tie at dark room
table	5e4	a wall	a table	blurry photograph of a person 's lap top and tan dog in window	black and white photograph of person 's hair with his head
unknown	5e4	a back of a	a table	picture of a dark haired person 's hair in dark suit and tie at home	blurry photograph of a person 's lap top and brown dog 's lap top
bed	4e4	a bed	a bed	blurry photograph of a person 's lap top and baby dog in bed	this black cat 's hand 's bed with her hair and suitcase in the corner
ceiling	3e4	the ceiling of a lot of ceiling	ceiling that is hanging in the ceiling	picture of dark colored cat in ocean 's hand at night	picture of
desk	3e4	a table	a floor of a table	picture of a woman with dark hair in hand and striped tie at night	picture of person in dark colored tie at night with
otherstructure	2e4	some kind of door	some kind of door	picture of a dark colored cat in ocean with his arm out at night	picture of small tan cat with eyes closed in dark room
refridgerator	2e4	a refrigerator	a refrigerator	picture of a woman 's lap top and tan - - fashioned door at night	blurry photograph of person in dark colored suit and tie in window
door	2e4	some kind of door	some kind of door that is door	picture of a dark - haired woman 's lap top and door in the corner	blurry photograph of a person in dark suit and tie
television	2e4	a tv	a tv	picture of a dark - haired woman 's tie at night with the door of the sky	picture of a dark - haired woman 's tie at night with the word [messy text]
otherfurniture	8e3	a toilet	a toilet	picture of a female male with dark hair dryer in corner of the window	picture of a female 's hair with a white tank top new
window	8e3	window of open	window of open	blurry photograph of person in dark blue sky at home	blurry photograph of person in dark blue sky at home
toilet	6e3	a toilet	a toilet	picture of a female male dog 's lap top in dark room	picture of male cat in small bathroom with leather tie and seat [messy text]
sink	5e3	a toilet	a sink	picture of a woman with dark hair in hand and striped sky	picture of a woman in dark colored suit and gold long
night stand	5e3	a table with the bottom	table with the top	picture of a male cat with dark hair in hand and tie at night	picture of a woman with dark hair and tie at small [messy text]
counter	4e3	a counter	a sink	picture of a woman 's hair with baby eyes in dark colored cabinet	blurry photograph of person 's dog in dark blue and leather suit case
pillow	3e3	a bed	a person with it	blurry photograph of a person 's hair with his lap top and black dog in the window	blurry photograph of person 's hair with his neck tie in dark sky
mirror	2e3	a few mirror of some sort	a few mirror that is by	picture of male cat in dark suit and tie at tennis ball in corner of room	blurry photograph of person in dark suit and tie at television

Table 5.1.: This table shows the caption generated by DeCap given a CLIP feature from a specific semantic class. **Avg** means the feature is averaged over all features of this semantic class, whereas **random** means a random samples of the given class.

(1) (2) (3) (4)

Decoding:
(1): `A chair sitting on top of a wooden floor.`
(2): `A wooden table with a cupboard and a wooden tablecloth.`
(3): `A wooden table with a wooden table cloth on top of it.`
(4): `A wooden table with a laptop computer on top of it.`

Figure 5.3.: These examples shows caption results of ClipCap given same set of objects but different relative positions. CLIP features do not reveal much about relative position information.

6. ConceptFusion Feature Analysis

In this chapter, we address ConceptFusion, a work closely related to our project. We aim to assess its compatibility with our findings and its potential integration into our pipeline. Our evaluation includes an introduction to ConceptFusion, followed by a section of our replicated results on the 3D fused ConceptFusion feature. However, our findings show a disparity with the claimed effectiveness. Finally, we provide a detailed analysis of the reasons for ConceptFusion's shortcomings.

6.1. Introduction to ConceptFusion Feature

The ConceptFusion pipeline first prduce a 2D dense CLIP feature. It aims to avoid the disaggregation in semantics. This means that the feature of each pixel should contain both its individual semantic meaning and the global semantics (i.e., the objects surrounding the pixels). Therefore, ConceptFusion not only extract the global CLIP feature vector f^G but also local CLIP features $\{f_i^L\}_{i=1}^N$.

Segment-Anything-Model (SAM) [24] is used to obtain fine-grained instance segmentation masks and bounding box. Then the local features are obtained by passing cropped image to CLIP, $f_i^L = \text{CLIP}(\text{Crop}(\text{Image}, \text{BBox}_i))$.

For each SAM mask, the regional feature f_i^R is a weighted combination,

$$f_i^R = w_i f^G + (1 - w_i) f_i^L, \tag{6.1}$$

where the weight is the softmax of cosine similarity,

$$w_i = \frac{\exp\left(\text{CosSim}(f_i^L, f^G)/\tau\right)}{\sum_j \exp\left(\text{CosSim}(f_j^L, f^G)/\tau\right)}, \text{ where } \tau = 1. \tag{6.2}$$

Then the final dense CLIP feature is a linear combination of all the mask feature

$$f_{\text{ConceptFusion2D}}(x, y) = \sum_{i=1}^{N} \mathbb{1}\{(x, y) \in \text{Mask}_i\} \times f_i^R. \tag{6.3}$$

After obtaining the 2D feature, ConceptFusion employs gradslam [23] to fuse the features into 3D. Unlike TSDF fusion, gradslam use point clouds as 3D representation, dynamically updating the position, normals, and features of points, or adding new points to the point clouds.

We will show the results of fusing real scene in next section.

6.2. ConceptFusion on 3D Scene

During our implementation, we observed that gradslam has a high demand for GPU memory. Even with a Tesla A100 80G GPU, it couldn't accommodate a resized 2D feature of 240×320

This is because the point cloud density in gradslam increases with the feature's resolution, which is not an issue in TSDF fusion. After some GPU-memory-saving adjustments, we successfully run gradslam on a Tesla V100 32G GPU with resolution of 120×160.

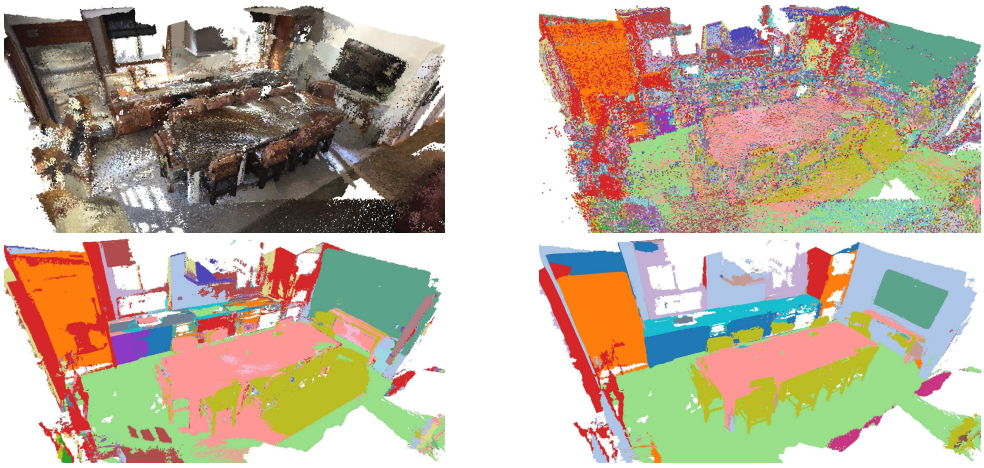


Figure 6.1.: Initial Results of ConceptFusion. Top left: Point clouds with RGB colors obtained from gradslam. Scannet `scene0011_00` is used for fair comparison with original paper. **Top right:** 3D ConceptFusion feature fused by gradslam, queried by NYU40 class labels, and colored according to NYU40 rules. The messy scattering points are caused by gradslam's instability when input feature is near zero. **Bottom left:** 3D ConceptFusion feature fused by TSDF integration, queried by NYU40 class labels, and colored according to NYU40 rules. The whole wall is recognized as television. This figure proves the semantic leakage weakness of ConceptFusion. **Bottom right:** 3D LSeg feature fused by TSDF integration, queried by NYU40 class labels, and colored according to NYU40 rules.

Figure 6.1 illustrates the semantic segmentation results of ConceptFusion features fused by both gradslam and TSDF fusion. The point cloud generated by gradslam exhibits notable noise, leading to apparent contamination in its semantic segmentation. Further, we found the features in these points usually have near infinity norm. After switching to TSDF fusion, the result gets better, although the segmentation defects are still very obvious. These defects can be summarized as "semantic leak". For instance, an entire wall is misclassified as television, or the chairs near the table are misclassified as table. Overall, the semantic segmentation performance for ConceptFusion is far worse than LSeg feature.

We then experiment with single vector similarity score query on ConceptFusion features (Figure 6.2), same as Fig. 7 in the ConceptFusion paper. Following the paper's methodology, we applied a threshold on similarity scores, highlighting only points with scores above this threshold. However, we did not get a well-localized heatmap. This is in expectation because the optimal threshold varies depending on the specific case.

Additionally, we attempted captioning using ConceptFusion features. However, caption results, as presented in Table 5.1, are completely unrelated. Keywords such as "*blurry*" or "*picture of...*" shows up frequently, suggesting a lack of meaningful captions.

In conclusion, ConceptFusion's performance fell below our initial expectations. As a result, integrating it into our pipeline does not appear to be a viable option. The following section

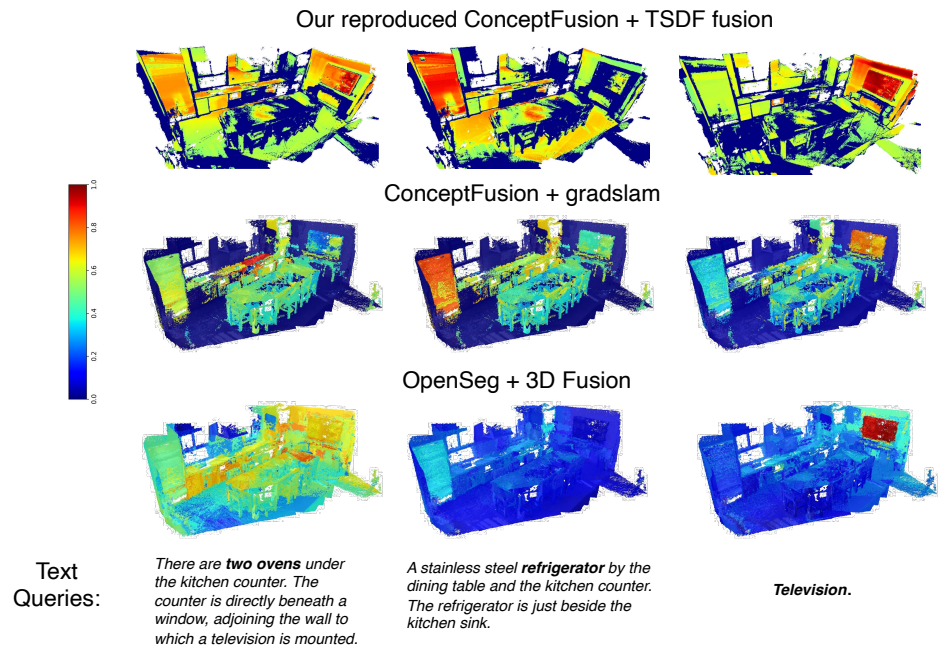


Figure 6.2.: Comparison of single vector query of ConceptFusion feature from original paper and our reproduced results. The highlighted area in this heatmap depends heavily on thresholding of similarity score. Therefore, localization of heatmap is not instinct in ConceptFusion. The figures from second and third rows are from the original ConceptFusion paper.

will provide a detailed discussion of the aforementioned problems.

6.3. Analysis of Failure of ConceptFusion

In our analysis, we refer to the image on the left of Figure 6.3 as a representative example.

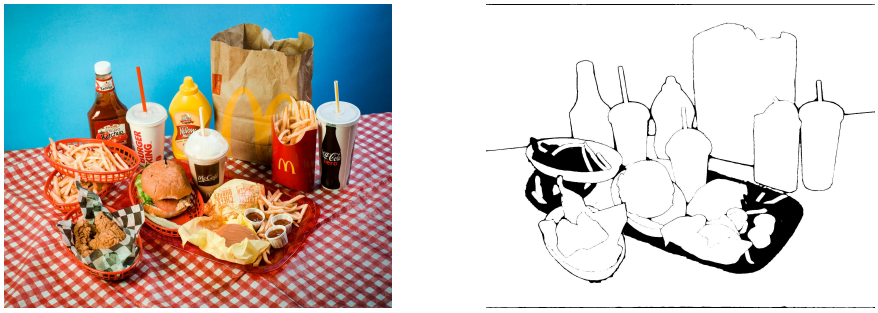


Figure 6.3.: This **right** figure shows the hollow area (**black**) of 2D ConceptFusion feature. The zero norm feature in these areas leads to instability in gradslam point fusion.

We observed that SAM fails to cover the entire image. The right of Figure 6.3 shows the hollow region overlooked by SAM, leading to feature vectors with zero norms in these pixels. We attribute the presence of noisy feature points in gradslam to these zero-norm features, as they are likely to cause division-by-zero during fusion. After substituting them with random Gaussian vector, the noisy points disappear.

Furthermore, we identified that "semantic leak" phenomenon arise from the semantic impurity of local feature. For instance, while SAM accurately detects the background in the sample image, the cropped background image inevitably includes objects such as the sauce bottle, the paper bag, and the cola cup, due to the non-convex shape of the background. Consequently, the local CLIP feature also contains semantics of nearby objects. Therefore, when queried with the leaked objects, the background is highlighted (Figure 6.4 upper left provides an example).

Another potential source of "semantic leak" originates from the global CLIP feature. It is worth noting that the cosine similarity between CLIP image features typically falls within the range of $[0.3, 1.0]$. If a high temperature is used, for instance $\tau = 1$ in our case, the global weight w_i may not exhibit significant variability. In such cases, $w_i \approx \frac{1}{N}$, where N denotes the number of instance masks. Consequently, each non-zero ConceptFusion feature will have a roughly equal amount of global feature, making them similarly responsive to any text query.

We come up with an enhanced version of ConceptFusion that addresses the above two issue: (1) discarding global feature, and (2) weakening or blocking the background during local feature computation. The latter step aims to avoid the leak of neighboring objects' semantics into local feature. Nevertheless, in some extreme cases, some degree of semantic leakage may still occur based on the silhouette of the background. The bottom two figures of Figure 6.4 illustrate that our improved iteration of ConceptFusion gives precise and localized response to target text query. As our pipeline ultimately relies on fused CLIP for captioning, an aspect in which ConceptFusion is not good at, we choose not to present the fusion results for the improved version of ConceptFusion.

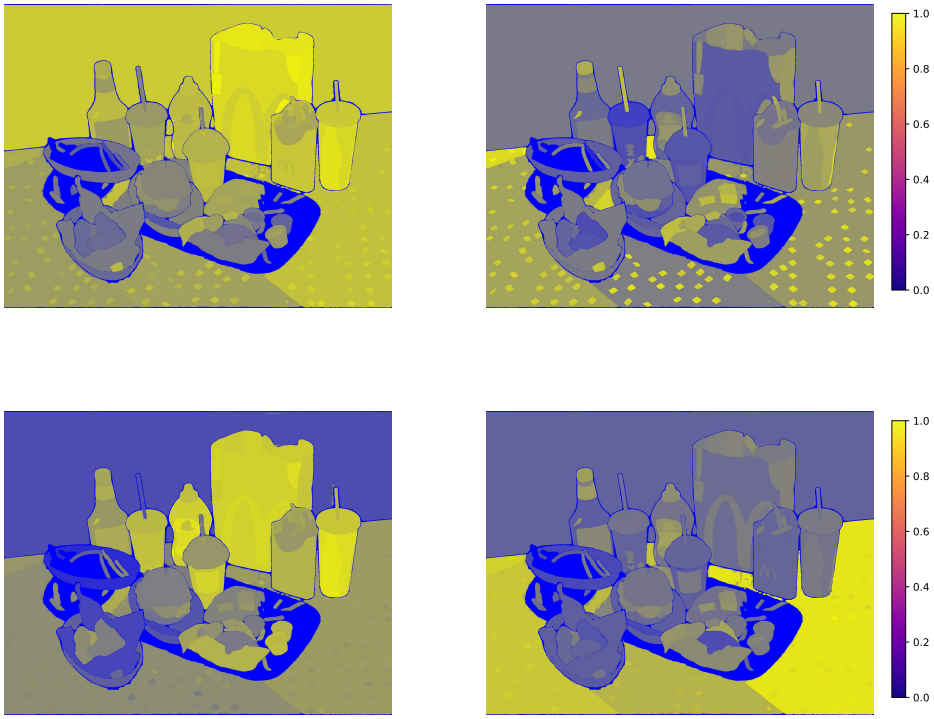


Figure 6.4.: 2D Analysis of ConceptFusion. The two figures on the **left** are query with *there is a paper bag left to a cup of cocacola and it is right to a yellow bottle of sause*, and the **right** two are queried with *tablecloth*. The **upper** two figures use original ConceptFusion feature, where semantic leak leads to non-localized highlighted area, whereas our enhanced version of ConceptFusion does not suffer semantic leak (the **lower** two figures).

A potential explanation for ConceptFusion's lower classification performance could lie in the fact that LSeg undergoes fine-tuning on image segmentation data utilizing a contrastive loss. Despite potential issues with catastrophic forgetting, LSeg demonstrates enhanced discrimination capabilities, particularly for objects that are more likely to coexist within a given scene.

Figure 6.5.: We use caption as a probe of CLIP feature quality. Low resolution figures' CLIP feature gives inconsistent captions with the high resolution ones. ConceptFusion may suffer from low quality CLIP features as it use small segmentations from SAM.

The diminished captioning quality of ConceptFusion features may originate from SAM's tendency to segment the image into the smallest possible parts. This results in low-resolution cropped images when computing local features. Experiment illustrated in Figure 6.5 shows that down-sampling image leads to degradation in captioning quality and introduce keywords such as *blurry image* and *photograph of*. This explains the unrelated text appear in ConceptFusion captions. Intriguingly, we observed that stretching the image to an extreme aspect ratio does not lead to a decline in caption quality.

We have concluded our finding on ineffectiveness of ConceptFusion, and provided analysis on the reasons. In the next chapter, we will state our initial idea on building the detection pipeline.

7. Attempts on Information Extraction

This chapter functions as an intermediary, dedicated to documenting shifts in ideas.

7.1. Attempt to Extract Relations with Visual Language Model

In the middle stage of our project, we set an ambitious goal: not only to detect instances in an open-vocabulary manner, but also to construct a scene graph, which involves extracting potential relationships between these instances. Initially, we considered querying 2D images using a fixed set of relationships. However, after researching state-of-the-art models in visual question answering (VQA), we discovered that, at the present time, this remains a highly challenging task. In Figure 7.1, we provide an example of BLIP-2, a well-known visual language model in VQA. It consistently provides incorrect answers to every question. It's possible that the model requires fine-tuning, or that we didn't use the appropriate prompt. Nevertheless, these findings led us to redirect our focus from relationship extraction to instance segmentation.

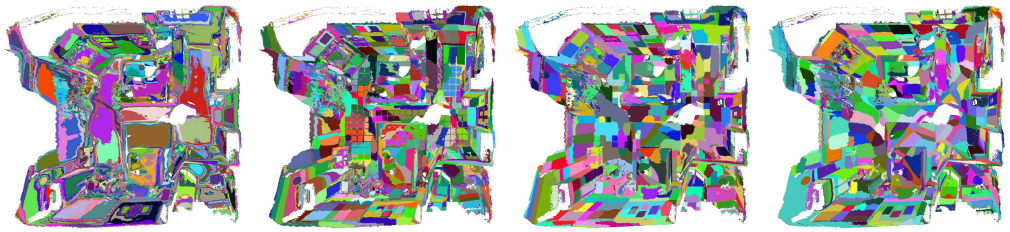
```
- [Question] proposition: "the apple is on the chair",
answer yes if this statement is true, no if it is false
or unreasonable.
- [Answer] yes
- [Question] proposition: "the apple is under the chair",
answer yes if this statement is true, no if it is false
or unreasonable.
- [Answer] yes.
- [Question] proposition: "there are no apple nor chair
in this image.", answer yes if this statement is true,
no if it is false or unreasonable.
- [Answer] yes..
```

Figure 7.1.: An example of question answering using BLIP-2. It shows that current visual language model is not capable of accurate relation extraction.

7.2. Attempt of Instance Segmentation

Performing instance segmentation based on LSeg or ConceptFusion features is a challenging, even impossible task. This is due to the possible occurrence of distant objects sharing similar semantic classes. Consequently, clustering on semantics related features, makes it impossible to distinguish between features of these objects. Figure 7.2 shows the K-Means clusters ($K = 1024$) on LSeg and ConceptFusion features. We can see that the paintings on the wall share same labels even if they are spatially separated.

Therefore, we need a semantics agnostic but instance sensitive feature. Our solution is mask-consitent random feature. For each 2D frame, we extract its instance segmentation



(a) LSeg (b) ConceptFusion (c) Random SAM (d) Random Grounded SAM

Figure 7.2.: K-Means cluster coloring of different features (LSeg, ConceptFusion, Random SAM, Random Grounded SAM).

masks $\{\text{Mask}_i\}_{i=1}^{M}$ using SAM. Then we use a random gaussian vector $\vec{e}_i \sim \mathcal{N}(0, I) \in \mathbb{R}^{128}$ as the unique "identity vector" for this mask, and "color" it to all the pixels belonging to this mask. This gives a dense random feature, defined as,

$$f_{\text{rand2d}}(x, y) = \sum_{i \in M} \mathbb{1}\{(x, y) \in \text{Mask}_i\} \times \vec{e}_i, \text{ where } \vec{e}_i \sim \mathcal{N}(0, I) \in \mathbb{R}^{128}. \tag{7.1}$$

The choice of a 128-dimensional vector balances vector uniqueness and computational efficiency.

After fusing this random features into 3D and applying K-Means clustering, we found that the voxels are clustered into sub-instance level patches (Figure 7.2). Notably, the boundaries of these patches align well with actual instance segmentation boundaries. While these patchs may not be perfectly instance level, it provides a good starting point.

During our project, a new model called Grounded SAM [25] was introduced. It not only provides fewer, better and more efficient instance segmentation compared to SAM, but also assigns open-vocabulary tags to each object. We also provide the K-Means of Grounded SAM's random feature in Figure 7.2. As we will see in the next chapter, Grounded SAM is heavily integrated into our pipeline.

8. PanopticFusion and Graph-Connecting Algorithm

Up to this point, we have demonstrated the ability of our pipeline in extracting semantic information. The current challenge lies in extracting instances. This chapter begins with an introduction to an influential work to our pipeline: PanopticFusion. This work is about integrating 2D instance segmentation into 3D. Then we will present our replicated results of PanopticFusion and analyze its limitations. Finally, we will introduce how we solve these limitations with our graph-connecting algorithm.

8.1. Introduction to PanopticFusion

Fusing instance segmentation into 3D involves finding an optimal labeling for all voxels, that minimizes conflicts with 2D segmentations. This is computationally challenging due to the large number of voxels in a scene (often in the millions), each potentially having plenty possible labels (up to the number of instances in all 2D frames).

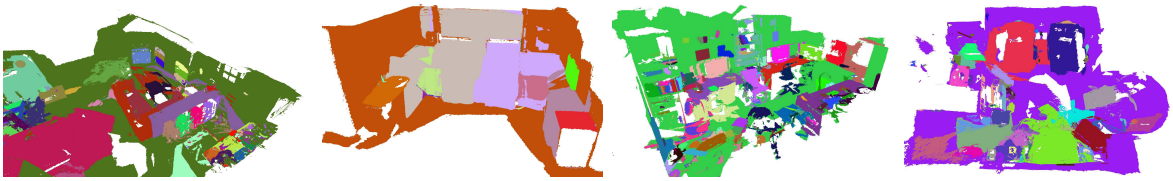
PanopticFusion tackles this problem with a greedy approach. Here we will introduce the general idea of PanopticFusion, omitting the cumbersome mathematical derivations. PanopticFusion maintains a list of 3D instance IDs from observed 2D frames and assigns each voxel an ID from this list. When a new frame is introduced, its 2D mask is either matched to an existing 3D instance ID or recognized as a new instance, which is then added to the 3D instance list. To save computation, PanopticFusion retains the most probable instance ID for each voxel, along with a confidence weight. Once this 2D-3D instance matching for a new frame is completed, we increase confidence weight if the new instance ID matches the old one, or decrease it if not. The procedures above represent the basic version of PanopticFusion pipeline. The authors also propose adding Conditional Random Field (CRF) layer to improve accuracy. However, this requires neural network training, and this aspect falls beyond the scope of our current discussion.

Figure 8.1 displays the instance segmentation results of PanopticFusion, employing Mask-RCNN, Grounded SAM, or ground truth as the mask generator. PanopticFusion with Grounded SAM or ground truth masks gives very impressive results with clear segmentation boundary. For a fair comparison, we also try to replicate the results from original paper, where they use a Mask-RCNN model fine-tuned on ScanNet. However, due to unavailability of their source code and a lack of comprehensive training methodology, reproducing their results is unfeasible. The Mask-RCNN used to produce results in Figure 8.1 is based on ResNet101 and FPN, it is pretrained on MS COCO dataset without fine-tuning. This disparity explains the relatively inferior segmentation performance observed in the figure.

Despite the impressive results of PanopticFusion, we find some evident problems:

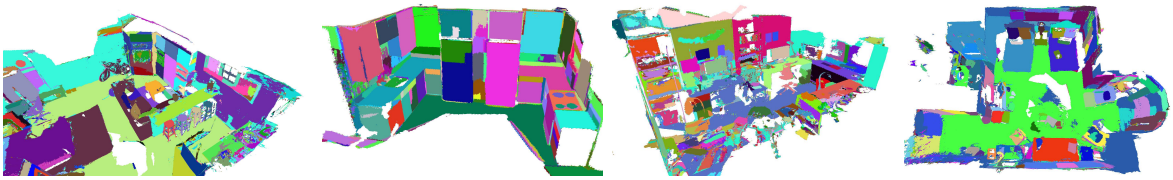
- **Inefficient data usage due to duplicate IDs:** PanopticFusion encounters inefficiencies in instance assignment due to its use of a fixed IoU threshold for 2D-3D matching.

(a) *Color*

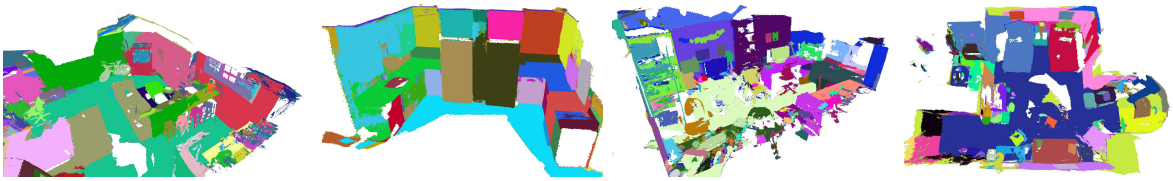


(b) *PanopticFusion with MS-COCO pretrained Mask-RCNN instance segmentation.*

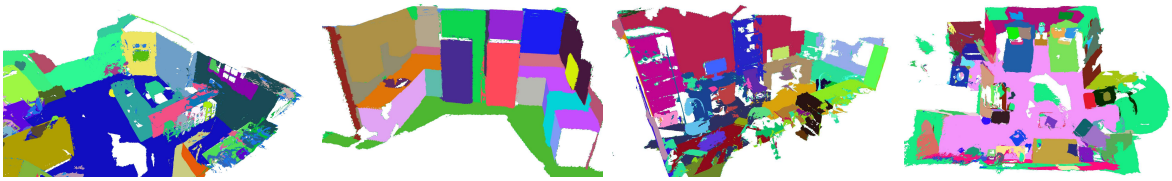
(c) *PanopticFusion with Grounded SAM instance segmentation.*



(d) ***Graph Connecting Instance segmentation using Grounded SAM masks (Ours)***



(e) ***PanopticFusion guided with graph-connect instance segmentation (Ours)***



(f) *PanopticFusion with 2D ground truth instance segmentation.*

Figure 8.1.: Instance segmentation results using PanopticFusion with various 2D masks and our graph-connect algorithm. Scannet scenes `scene0000_00`, `scene0488_01`, `scene0643_00` and `scene0645_01` are used for fail Comparison with the original PanopticFusion paper. See here for more views. Guided PanopticFusion is a post-processing step after graph-connecting algorithm. It uses graph-connecting algorithm's 3D segmentation label as guidance in 2D-3D matching. However, it does not lead to better results.

This one-size-fits-all rule results in the assignment of duplicate instance IDs to objects that should be represented as a single entity. Consequently, unnecessary competition arises among these duplicates, leading to suboptimal data utilization. For instance, in `scene0000_00`, PanopticFusion erroneously identifies around 10,000 3D instances, while the accurate count should be closer to a hundred.

- **Path-dependent Nature:** The 3D instance ID list used in PanopticFusion depends on the history frames, and this may lead to extra instance competition. An illustrative example is observed in `scene0000_00` with ground truth segmentations (bottom left of Figure 8.1). Here, the curtain is bifurcated into red and green segments. This division arises from the scanner's trajectory, which captures the left part in the initial frames and the right part in the latter frames. Consequently, PanopticFusion assigns separate instance IDs to these segments and there is no chance these parts can merge by the algorithm. The division depicted in the figure is a consequence of this type of instance competition.

- **Preference for Larger Segmentations:** In certain scenarios, mask generator produces multiple levels of segmentation (e.g., an entire bookshelf versus individual books on the shelf). In the 2D-3D matching stage of PanopticFusion, matching of 2D masks are ordered descending by their area. Additionally, large 3D instance tend to have higher IoU, making them more likely to be selected during matching. Therefore, PanopticFusion can potentially downgrade segmentation details. This becomes apparent when compared to our graph-connecting algorithm.

8.2. Graph-connecting Algorithm

PanopticFusion addresses computational complexity by a greedy assignment method. We achieve this by narrowing the optimization domain from voxels to K-Means patches. In Section 7.2, we observe that K-Means patches is at the sub-instance level and align their boundaries well with actual segmentation boundaries. This reduces the task to merging patches belonging to the same instance. A natural step next is to build a K nodes graph denoting each patches and establish edges between patches likely to be in the same segment.

This idea leads us to make a statistics about whether two patches are consistently recognized in same segments across all frames. However, there remain practical challenges in algorithm design, stated as follows:

The first problem is that the mask generator may produce multiple levels of segmentation, causing two patches to belong to the same segment in certain frames but different segments in others. Thus, a good solution is to keep both positive evidence counts $w_+(i,j) = \sum_{f \in \text{all frames}} c_{+,f}(i,j)$ where patches pair (P_i, P_j) are of the same segment, and negative evidence counts $w_-(i,j) = \sum_{f \in \text{all frames}} c_{-,f}(i,j)$ denoting that patches are of different segments. In the end, we preserve edges where positive evidence surpasses negative evidence, i.e., $\log\left(\frac{w_+(i,j)}{w_-(i,j)}\right) > \theta_1$. We additionally put a constraint on the absolute value of positive evidence$w_+(i,j) > \theta_2$, to prevent cases where there are no negative evidence for pair (P_i, P_j) but an accidental positive evidence.

Another concern arises from the potential misassignment of unrelated patches to a specific 2D instance in some frames. To mitigate this, we apply a confidence score $c_{+/-,f}(i,j)$ to per-frame evidence. For each voxel patch, we define three types of confidence $c_1(P_i), c_2(P_i), c_3(P_i)$

based on (1) patch's absolute voxel size, (2) the proportion of patch's voxel that appears in current frame, and (3) number pixels occupied by this patch in current frame. $c_1(P_i), c_2(P_i), c_3(P_i)$ are thresholded functions of these three factors. Finally, the per-frame edge evidence is $c_{+/-,f}(i,j) = c_1(P_i) \times c_2(P_i) \times c_3(P_i) \times c_1(P_j) \times c_2(P_j) \times c_3(P_j)$.

We summarize the above discussion into our graph-connecting algorithm, as depicted in Algorithm 1. The final instance IDs are determined by the connected components of the graph.

This algorithm addresses all three issues in PanopticFusion. Firstly, it inherently avoids generating duplicate instances. Secondly, the evidence are the accumulation of per-frame evidence, making the algorithm path-independent. Third, our edge criterion $\log\left(\frac{w_+(i,j)}{w_-(i,j)}\right) > \theta_1$ ($\theta_1 = 2$ in our implementation) imposes a strict condition on merging two patches. This effectively prevents the merging of instances into higher abstraction levels. Therefore, as shown in Figure 8.1, our algorithm produces finely detailed yet intact segmentations compared with PanopticFusion.

9. Discussion

In this section, we'll address the potential limitations of our pipeline.

Firstly, our pipeline exhibits slower processing times (refer to Table 9.1) and lacks the capability for online execution due to the requirement of complete frame data for K-Means processing. Consequently, real-time execution is not possible. This stands as a notable drawback when compared to the PanopticFusion.

Stage	**Seconds per Frame**
TSDF Fusion	0.05
LSeg Feature Fusion	0.16
Grounded SAM Mask Extraction	0.62
Random Feature Fusion	0.18
Graph Building	0.33
total	1.34
ConceptFusion (unused)	5.27
GradSLAM (unused)	0.40
PanopticFusion (unused)	0.23

Table 9.1.: Processing time for each stage of our pipeline

Secondly, there are potential improvements to be made in our pipeline. While K-Means provides localized and mask-sensitive patches, localization is probabilistic and not guaranteed. Patches can sometimes contain distance outlier. Consequently, we may accidentally merge two patches that are distant in space (see Figure 9.1 for an example). Furthermore, in the graph-connecting algorithm, we employ a fixed threshold for edge pruning. This may not be suitable for cases where the observation over each object in a scene is highly imbalanced. Therefore, a better solution approach would be to implement a graph neural network (GNN) to handle the graph connecting task at greater precision.

Finally, we posit that our pipeline serves as a proficient bridge from 2D to 3D. We harness high-performing models in 2D to fulfill tasks in 3D. However, we hold the belief that the optimal solution for these tasks will ultimately be 3D-centric.

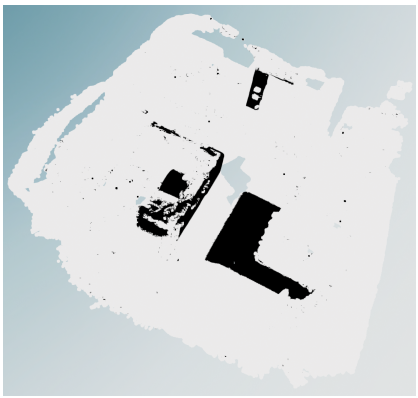


Figure 9.1.: A failure case where our graph-connecting algorithm merge distance patches together.

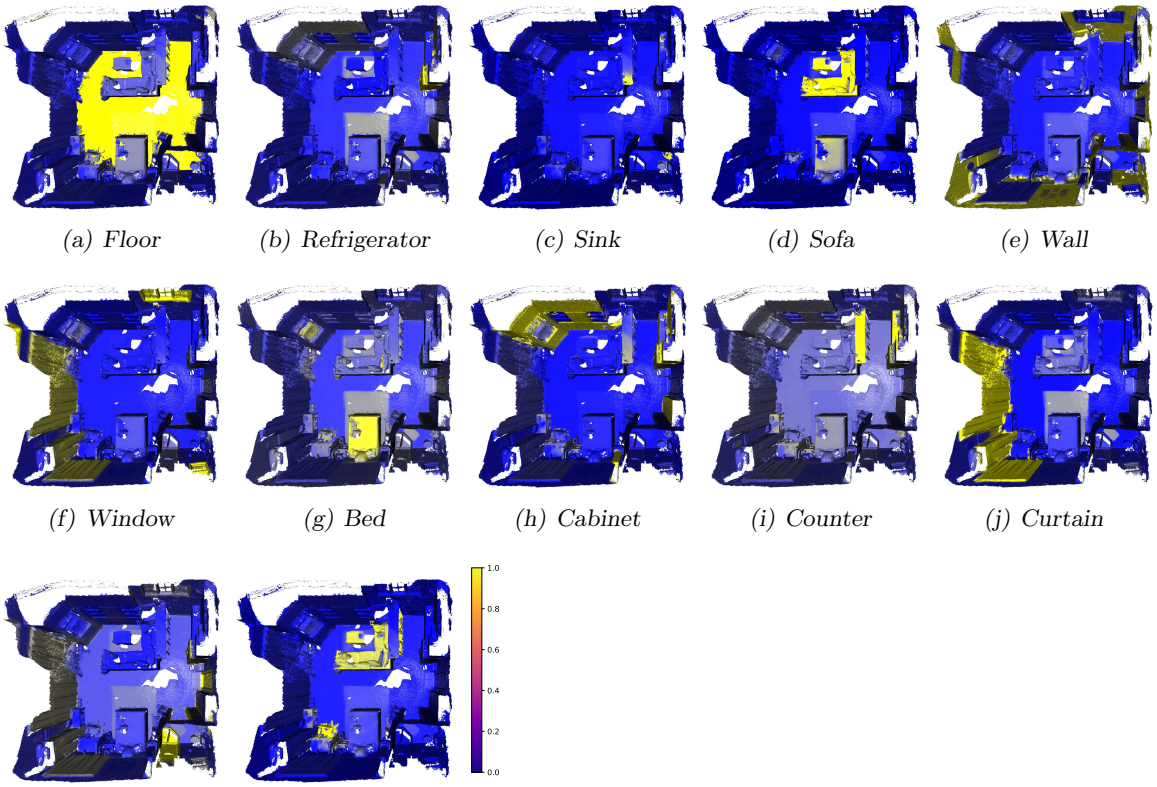
10. Conclusion

In this project, we propose a novel 3D detection pipeline. It capitalizes on high-performing 2D models for instance segmentation and employs dense CLIP feature extraction, seamlessly integrating them into the 3D domain. The graph-connecting algorithm in our pipeline yields more precise and comprehensive instance segmentation compared to prior approaches. Additionally, the incorporation of a captioning module enables us to perform open-vocabulary object detection. We believe our work will offer valuable insights for future research in 3D scene understanding.

Appendices

A. LSeg feature single word and “other” class query

The following figures show how well aligned the LSeg figure is with respect to the given text query, or to the “other” text query. The score is processed with softmax and a temperature of 0.03.

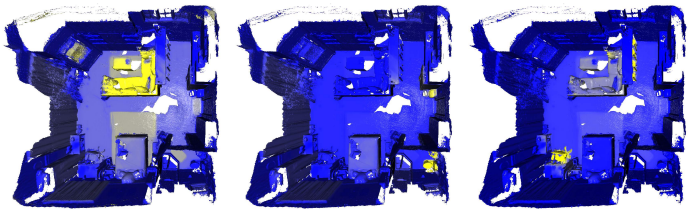


(a) Floor *(b) Refrigerator* *(c) Sink* *(d) Sofa* *(e) Wall*

(f) Window *(g) Bed* *(h) Cabinet* *(i) Counter* *(j) Curtain*

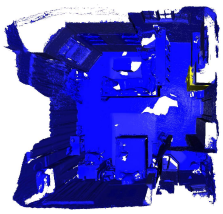
(k) Door *(l) Chair*

Figure A.1.: LSeg single class and 'other' class query: Noun.



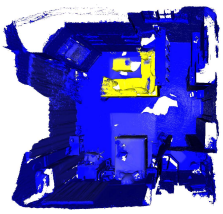
(a) Living room *(b) Bathroom* *(c) Legs of chair*

Figure A.2.: LSeg single class and 'other' class query: Bigger or smaller scope.

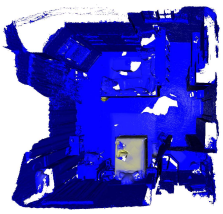


(a) *Front panel of fridge*

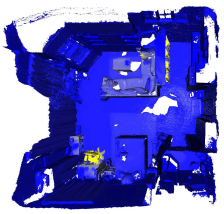
(b) *Middle of the bed*



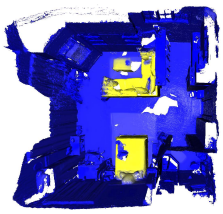
(c) *Edge of sofa*



(d) *Backpack on the bed*

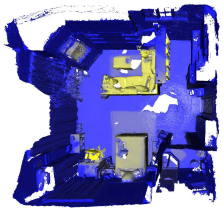


(e) *Left of chairs*

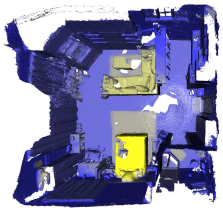


(f) *Middle between bed and sofa*

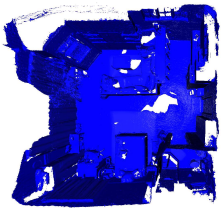
Figure A.3.: LSeg single class and 'other' class query: Relative Position.



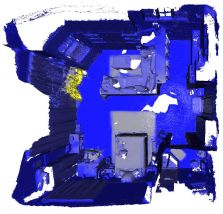
(a) *Sit*



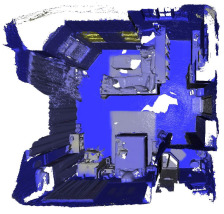
(b) *Sleep*



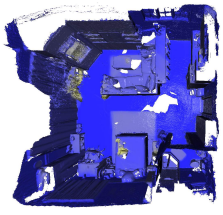
(c) *Brush teeth*



(d) *Ride*

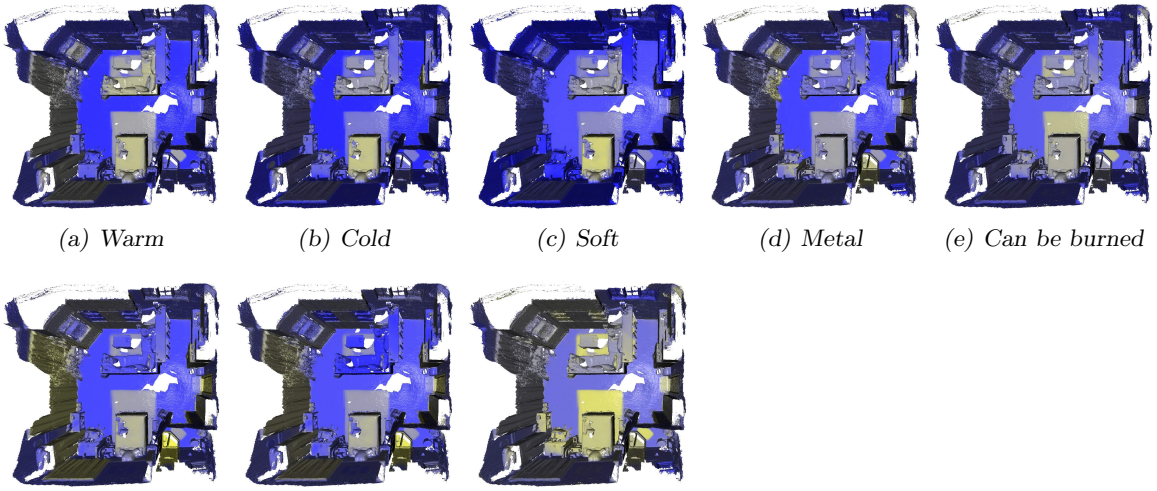


(e) *Read*



(f) *Travel*

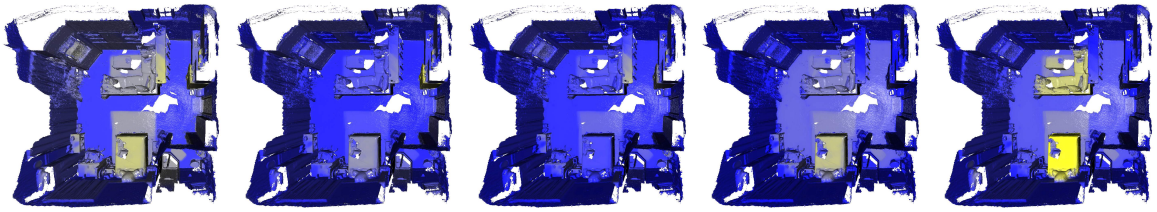
Figure A.4.: LSeg single class and 'other' class query: Verb.



(a) Warm *(b) Cold* *(c) Soft* *(d) Metal* *(e) Can be burned*

(f) Can be closed *(g) Can be closed or opened* *(h) Can be used*

Figure A.5.: LSeg single class and 'other' class query: Adjective.



(a) Food *(b) Cold food inside* *(c) Place for cooking* *(d) Place for sleeping* *(e) Sleeping*

Figure A.6.: LSeg single class and 'other' class query: Indirect words, implicit inference.

(a) Bed and sofa *(b) Toilet and curtain* *(c) Not on the floor* *(d) Toilet but not sink*

Figure A.7.: LSeg single class and 'other' class query: Logical combination.

Bibliography

[1] G. Narita, T. Seno, T. Ishikawa, and Y. Kaji, "Panopticfusion: Online volumetric semantic mapping at the level of stuff and things," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 4205–4212.

[2] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, *et al.*, "Learning transferable visual models from natural language supervision," in *International conference on machine learning*. PMLR, 2021, pp. 8748–8763.

[3] S. Peng, K. Genova, C. M. Jiang, A. Tagliasacchi, M. Pollefeys, and T. Funkhouser, "Openscene: 3d scene understanding with open vocabularies," 2023.

[4] C. Zhou, C. C. Loy, and B. Dai, "Extract free dense labels from clip," in *European Conference on Computer Vision*. Springer, 2022, pp. 696–712.

[5] M. Xu, Z. Zhang, F. Wei, H. Hu, and X. Bai, "Side adapter network for open-vocabulary semantic segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 2945–2954.

[6] J. Xu, J. Hou, Y. Zhang, R. Feng, Y. Wang, Y. Qiao, and W. Xie, "Learning open-vocabulary semantic segmentation models from natural language supervision," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 2935–2944.

[7] J. Mukhoti, T.-Y. Lin, O. Poursaeed, R. Wang, A. Shah, P. H. Torr, and S.-N. Lim, "Open vocabulary semantic segmentation with patch aligned contrastive learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 19 413–19 423.

[8] J. Xu, S. De Mello, S. Liu, W. Byeon, T. Breuel, J. Kautz, and X. Wang, "Groupvit: Semantic segmentation emerges from text supervision," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 18 134–18 144.

[9] J. Cha, J. Mun, and B. Roh, "Learning to generate text-grounded mask for open-world semantic segmentation from only image-text pairs," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 11 165–11 174.

[10] H. Luo, J. Bao, Y. Wu, X. He, and T. Li, "Segclip: Patch aggregation with learnable centers for open-vocabulary semantic segmentation," in *International Conference on Machine Learning*. PMLR, 2023, pp. 23 033–23 044.

[11] Q. Liu, Y. Wen, J. Han, C. Xu, H. Xu, and X. Liang, "Open-world semantic segmentation via contrasting and clustering vision-language embedding," in *European Conference on Computer Vision*. Springer, 2022, pp. 275–292.

[12] C. Ma, Y. Yang, Y. Wang, Y. Zhang, and W. Xie, "Open-vocabulary semantic segmentation with frozen vision-language models," *arXiv preprint arXiv:2210.15138*, 2022.

[13] K. Ranasinghe, B. McKinzie, S. Ravi, Y. Yang, A. Toshev, and J. Shlens, "Perceptual grouping in vision-language models," *arXiv preprint arXiv:2210.09996*, 2022.

[14] F. Liang, B. Wu, X. Dai, K. Li, Y. Zhao, H. Zhang, P. Zhang, P. Vajda, and D. Marculescu, "Open-vocabulary semantic segmentation with mask-adapted clip," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 7061–7070.

[15] Z. Ding, J. Wang, and Z. Tu, "Open-vocabulary panoptic segmentation with maskclip," *arXiv preprint arXiv:2208.08984*, 2022.

[16] G. Shin, W. Xie, and S. Albanie, "Reco: Retrieve and co-segment for zero-shot transfer," *Advances in Neural Information Processing Systems*, vol. 35, pp. 33 754–33 767, 2022.

[17] D. Huynh, J. Kuen, Z. Lin, J. Gu, and E. Elhamifar, "Open-vocabulary instance segmentation via robust cross-modal pseudo-labeling," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 7020–7031.

[18] M. Xu, Z. Zhang, F. Wei, Y. Lin, Y. Cao, H. Hu, and X. Bai, "A simple baseline for open-vocabulary semantic segmentation with pre-trained vision-language model," in *European Conference on Computer Vision*. Springer, 2022, pp. 736–753.

[19] T. Lüddecke and A. Ecker, "Image segmentation using text and image prompts," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 7086–7096.

[20] B. Li, K. Q. Weinberger, S. Belongie, V. Koltun, and R. Ranftl, "Language-driven semantic segmentation," in *International Conference on Learning Representations*, 2022. [Online]. Available: https://openreview.net/forum?id=RriDjddCLN

[21] G. Ghiasi, X. Gu, Y. Cui, and T.-Y. Lin, "Scaling open-vocabulary image segmentation with image-level labels," in *European Conference on Computer Vision*. Springer, 2022, pp. 540–557.

[22] K. Jatavallabhula, A. Kuwajerwala, Q. Gu, M. Omama, T. Chen, S. Li, G. Iyer, S. Saryazdi, N. Keetha, A. Tewari, J. Tenenbaum, C. de Melo, M. Krishna, L. Paull, F. Shkurti, and A. Torralba, "Conceptfusion: Open-set multimodal 3d mapping," *arXiv*, 2023.

[23] J. Krishna Murthy, S. Saryazdi, G. Iyer, and L. Paull, "gradslam: Dense slam meets automatic differentiation," *arXiv*, 2020.

[24] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, P. Dollár, and R. Girshick, "Segment anything," *arXiv:2304.02643*, 2023.

[25] S. Liu, Z. Zeng, T. Ren, F. Li, H. Zhang, J. Yang, C. Li, J. Yang, H. Su, J. Zhu, *et al.*, "Grounding dino: Marrying dino with grounded pre-training for open-set object detection," *arXiv preprint arXiv:2303.05499*, 2023.

[26] W. Li, L. Zhu, L. Wen, and Y. Yang, "Decap: Decoding clip latents for zero-shot captioning via text-only training," *arXiv preprint arXiv:2303.03032*, 2023.

[27] P. J. Rösch and J. Libovickỳ, "Probing the role of positional information in vision-language models," *arXiv preprint arXiv:2305.10046*, 2023.

[28] G. Ilharco, M. Wortsman, R. Wightman, C. Gordon, N. Carlini, R. Taori, A. Dave, V. Shankar, H. Namkoong, J. Miller, H. Hajishirzi, A. Farhadi, and L. Schmidt, "Openclip," July 2021, if you use this software, please cite it as below. [Online]. Available: https://doi.org/10.5281/zenodo.5143773

[29] R. Mokady, A. Hertz, and A. H. Bermano, "Clipcap: Clip prefix for image captioning," *arXiv preprint arXiv:2111.09734*, 2021.