DISS. ETH NO. 29771

A FAREWELL TO SUPERVISION
Towards Self-supervised Autonomous Systems

A thesis submitted to attain the degree of

DOCTOR OF SCIENCES OF ETH ZURICH
(Dr. sc. ETH Zurich)

presented by

KENNETH TOR BLOMQVIST

MSc Aalto University CS

born on 20.09.1993
citizen of Finland

accepted on the recommendation of

Prof. Dr. Roland Siegwart, examiner
Prof. Dr. Andrew Davison, co-examiner
Prof. Dr. Jen Jen Chung, co-examiner

2023

Autonomous Systems Lab
Department of Mechanical and Process Engineering
ETH Zurich
Switzerland

ABSTRACT

In the past decade, computer vision has progressed by leaps and bounds. Deep learning based methods have crushed benchmark after benchmark in a paradigm shift that has converted precision engineered, hand-crafted approaches into neural networks that simply learn from millions and millions of input-output examples. As each neural network is task and application specific, this means that to tackle a new task, the main problem has become how to create the datasets that will be able to teach a neural network to solve the task. Often, these datasets are built up by hand by annotating examples one by one through computer user interfaces, often outsourcing the work to low income countries. This creates additional challenges as the workers might not be domain experts on the data being annotated and they, in turn, have to be taught.

This severely limits the tasks and domains where we can deploy robots with advanced perception skills, as creating these datasets is expensive. The costs are only bearable for applications which are very general and have massive markets. Creating robots for niche industrial use cases or simply adapting existing robots to new domains, is infeasible. The costs of retraining and annotating data also often have to be borne when significant changes are made to the hardware of the robot, as the data distribution has changed.

In this thesis, we develop techniques to tackle this data problem for robot perception tasks. We approach it from multiple different directions, both by making better use of unlabeled data and by constructing ways in which we can better make use of the human teacher's time.

In the first part of this thesis, we develop a method by which we can quickly build up 3D object keypoint datasets to teach robots about semantic points on objects that are relevant for custom tasks. We design a pipeline to make use of proprioceptive sensing built into the robot and 3D geometry to propagate examples from one annotated frame to the next. We then use these examples to bootstrap a keypoint detection system, which can be deployed in minutes instead of days.

In the second part, we leverage neural implicit representations to extract dense segmentation masks from sparse user input, and use the representation

to synthesize novel examples of the scene, to better teach a downstream object detection system.

In the third part of this thesis, we design an interactive 3D volumetric scene annotation system, which is better able to make use of the expert user's time. We do this by leveraging self-supervised learning, techniques designed to learn from unlabeled data, to augment the data collected by the robot, and thus raising the level of abstraction and ending up with a smarter system.

In the last part of the thesis, we attempt to use information learned from large-scale internet image-caption datasets, and grounding them in real world 3D scenes, as a way to learn without any direct human supervision at all.

Finally, we sketch out a path forward for developing robust, continuously improving perception systems for robotic applications.

RÉSUMÉ

Au cours de la dernière décennie, la vision par ordinateur a progressé à pas de géant. Les méthodes basées sur l'apprentissage profond ont écrasé référence après référence dans un changement de paradigme qui a converti des approches artisanales en réseaux neuronaux qui apprennent simplement à partir de millions et de millions d'exemples. Comme chaque réseau neuronal est spécifique à une tâche et à une application, cela signifie que pour s'attaquer à une nouvelle tâche, le principal problème est de savoir comment créer les ensembles de données qui peuvent apprendre un réseau neuronal à résoudre la tâche. Souvent, ces ensembles de données sont constitués à la main en annotant des exemples un par un via des interfaces utilisateur graphique, en externalisant souvent le travail vers des pays à faible revenu. Cela crée des défis supplémentaires dans la mesure où les travailleurs ne sont peut-être pas des experts dans le domaine des données annotées et doivent, à leur tour, être formés.

Cela limite considérablement les tâches et les domaines dans lesquels nous pouvons déployer des robots dotés de compétences de perception avancées, car la création de ces ensembles de données coûte cher. Les coûts ne sont supportables que pour les applications très générales et ayant des marchés massifs. Créer des robots pour des cas d'utilisation industrielle de niche ou simplement adapter des robots existants à de nouveaux domaines est irréalisable. Les coûts de recyclage et d'annotation des données doivent également souvent être supportés lorsque des modifications importantes sont apportées au matériel du robot, car la répartition des données a changé.

Dans cette thèse, nous développons des techniques pour résoudre ce problème de données dans le cadre de tâches de perception robotique. Nous l'abordons sous plusieurs angles différents, à la fois en faisant un meilleur usage des données non étiquetées et en construisant des moyens permettant de mieux utiliser le temps de l'enseignant humain.

Dans la première partie de cette thèse, nous développons une méthode par laquelle nous pouvons rapidement construire des ensembles de données de points clés d'objets 3D pour enseigner aux robots les points sémantiques sur les objets qui sont pertinents pour des tâches personnalisées. Nous créons

un pipeline pour utiliser la détection proprioceptive intégrée au robot et la géométrie 3D pour propager des exemples d'une image annotée à la suivante. Nous utilisons ensuite ces exemples pour amorcer un système de détection de points clés, qui peut être déployé en quelques minutes au lieu de plusieurs jours.

Dans la deuxième partie, nous exploitons des représentations neuronales implicites pour extraire des masques de segmentation denses à partir d'entrées utilisateur clairsemées, et utilisons la représentation pour synthétiser de nouveaux exemples de la scène, afin de mieux enseigner un système de détection d'objets en aval.

Dans la troisième partie de cette thèse, nous concevons un système interactif d'annotation de scènes volumétriques 3D, mieux à même de valoriser le temps de la personne utilisant le système. Nous y parvenons en tirant parti de l'apprentissage auto-supervisé, des techniques conçues pour apprendre à partir de données non étiquetées, pour augmenter les données collectées par le robot, élevant ainsi le niveau d'abstraction et aboutissant à un système plus intelligent.

Dans la dernière partie de la thèse, nous tentons d'utiliser les informations tirées d'ensembles de données de légendes d'images Internet à grande échelle et de les ancrer dans des scènes 3D du monde réel, comme moyen d'apprendre sans aucune supervision humaine directe.

ACKNOWLEDGEMENTS

Firstly, I would like to thank the Max Planck ETH Center for Learning Systems for funding me and for running a fair admissions process, without which I would have never been able to attend ETH.

Thank you Roland, for putting up with me and for the unwavering trust in us staff members. Thank you Lionel and Jen Jen for supervising me and at least sometimes, taking some of my ideas seriously. A special thanks to Andrew Davison for joining the examination committee, whose work over the years has served as and endless source of inspiration.

Thank you to all my lab mates. Especially Francesco Milano for working with me and tag teaming on all those projects. Julian Foerster for making life outside of work more fun. Michel who is always good to cheer you up. Margarita for the inspiring conversations and for reminding us not to work too hard. Daniel, a true mad scientist and always an inspiration. Andrei for helping out with all things localization and mapping. Florian for teaching us the art of calibration. Fadri and Tonci for paving the way.

Thank you to my close friends and family and Charlotta for the support and my life outside of work.

Financial Support

The research conducted in this thesis has received funding from the Max Planck ETH Center for Learning Systems and the European Union's Horizon 2020 research and innovation programme under project PILOTING No H2020-ICT-2019-2 871542.

CONTENTS

INTRODUCTION

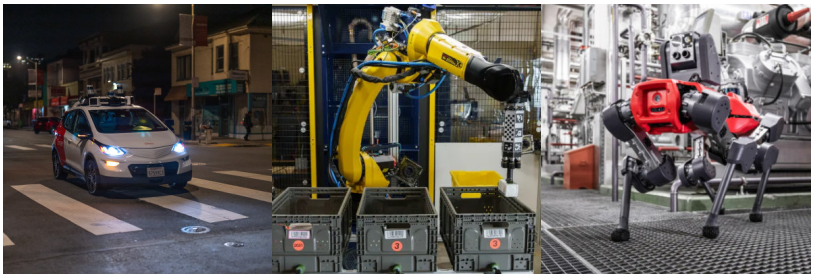
> The most important thing in communication is hearing what isn't said.
>
> *Peter Drucker*

Humans have always dreamed of artificial or magic servants that can perform all their manual labour for them. As far back as ancient Greece [30] people have dreamed of "automata" - machines which can act on their own will. They imagined such machines replacing slaves and even going to war with them.

Since then, especially over the past century, mankind has made great progress on this front, building industrial robots that operate in factories, vacuum cleaning robots, self-driving cars, quadruped robotic dogs and even humanoid robots that can run over obstacles and carry objects.

State-of-the-art legged robots such the quadruped Anybotics Anymal and Boston Dynamics Spot robots and humanoids such as the Boston Dynamics Atlas, have a high level of athletic intelligence, being able to balance, clear rough terrain and even perform backflips [34, 57]. Such robots still mostly follow a script provided by their programmer, or they are teleoperated. They have some geometric perception capabilities, such as localization, mapping and obstacle avoidance, but lack rich semantic perception, high-level reasoning and planning capabilities.

Autonomous vehicles are likely the most advanced robots in use today. Since last year (2022), such robots have been approved for commercial use and transport passengers in good weather conditions in the cities of San Francisco, California; Phoenix, Arizona and Austin, Texas in the United States [14]. Such robots have advanced perception capabilities combining cameras, radar and LiDAR sensor measurements to track other cars, pedestrians and road users, detect traffic signs, segment parts of the road and environment. These robots collaboratively build detailed and accurate maps of the environment and are able to navigate their way through them while following the rules of the road and being safe.



(a) (b) (c)

Figure 1: (a) Cruise autonomous vehicle in San Francisco, (b) industrial robot in Amazon warehouse, (c) Anybotics Anymal quadruped robot

In the case of industrial robots, in addition to entirely repeatable tasks that can be performed without any environment feedback, we have seen autonomous mobile robots moving pallets and shelves around in factories and warehouses. Industrial robots with perception capabilities are currently being used to pick and pack orders in e-commerce fulfillment centers, recycle waste and to tend machines in industrial facilities. Such systems are starting to be very effective, but especially robots making use of visual feedback still struggle with the "Ironies of Automation" [8] where they either perform worse than the manual process, or require some human supervision, reversing some of the gains of automation. Slowly though, these systems are getting there and are starting to replace manual human labor in more complicated tasks.

It is fair to say, that of the currently useful robots, almost all of them are purpose built, in hardware and software, for one specific task. Even in the case of general purpose hardware such as industrial robot arms or humanoid robots which are theoretically and physically able to perform multiple tasks, applying them to a new task or even the same task in a different environment requires reprogramming the robot, re-learning perception algorithms and in some cases, changing the representations used. While the hardware of current robots can always improve in flexibility, dexterity, durability and cost, what constrains current robots from truly being general purpose machines, is their high-level intelligence, planning and perception capabilities.

The high-level perception capabilities of current state-of-the-art robots are mostly built using deep learning techniques. The deep learning models used

are extremely data hungry and require millions of examples to reach good performance. Studies on deep learning models used in vision [119] and natural language processing [49] have shown that the models still improve even after having seen billions of examples. The bottle neck to teaching these systems is very much how quickly the data can be collected and annotated with the required labels. For language modeling, the data can be automatically collected from the internet. For spatial perception tasks such as those of interest in autonomous driving, companies have to employ humans to annotate the data collected by robots operating in the real world.

Currently, teaching robots to perceive things feels like pushing data to the robot through a Soviet era serial communication bus when it should feel more like uploading data over Terabit Ethernet. The bit rate is extremely low.

There are many reasons why communication with robots is slow and painful. Communication all starts with a shared language. In many cases today for perception, that shared language is annotations on individual images, such as individual class labels for pixels in images, vertex coordinates of polygons or bounding boxes. Producing these manually, even with smart software tools is slow. We are communicating very little data, and we are producing it very slowly. Not only do our current methods need very low-level annotations, they also don't make terribly good use of them, so we have to provide millions and millions of them to learn good models. As the supervision targets are quite low-level, such as individual pixels, the models have a tendency to overfit and find shortcuts to produce the correct prediction given the input, instead of building a representation of the scene and understanding the true underlying idea the expert is trying to convey.

We humans communicate with each other using high-level natural language. Often a couple words is enough to tell someone what needs to be done. Language provides shared context. Currently, there is very little shared context between the software run by our robots and us, their operators. Not only do we have shared context in language, but us animals also share the 3-dimensional spaces we occupy in addition to a broad set of societal and cultural norms.

While the robots have sensors, we are still mostly not able to make use of all the information in the data captured by the sensors. Algorithms today typically estimate predefined sparse quantities from sensor data and a lot of subtle information is thrown away. The little information we extract is

mostly not connected to a spatial representation or related concepts nor the prevailing cultural context.

One last point, is that we humans share an embodiment, which can be helpful when learning from each other. Our robots have a different geometry, have less degrees of freedom, use different sensors and operate in a different configuration and state space than we do, making learning from us harder. Any inference originating from the behaviour of another creature with a different embodiment, would have to go through additional reasoning steps to map it to its own embodiment and infer the relevant parts.

Therefore, if we want to have autonomous systems that are as versatile and easy to teach as us humans, we should start by defining a language and building common context between robots and their operators. In this thesis, we explore ways of teaching robots general perception capabilities without relying on large task-specific datasets, by leveraging shared context and prior information.

1 OBJECTIVE

Motivated by the spirit and state-of-the-art of robotic perception presented above, we define the objective of this thesis as *bootstrapping robotic perception*. Bootstrapping, meaning using readily available sources of data, assumptions, constraints and clever programming to equip the robot with an initial level of intelligence such that the robot's operator can more easily and effectively teach the robot the perception skill it needs to perform the tasks that they want the robot to execute. This teaching can be done either online while the robot is running, or offline, after the robot has collected some data and we can teach it something about that data. We seek to do this by minimizing the assumptions made in the design phase on the specific tasks the robot will be expected to perform. The goal is to provide the operator with a specific set of tools that enables them to get the robot to perform a task of their choice, not one of the robot's programmer's choosing. One way to think of this objective, is to provide the robot with *useful* bias that will accelerate its learning and raise the abstraction level in the teaching phase.

Science fiction tends to depict robots of the future as generally intelligent beings that can do and understand everything. That might be the case some day, but we very much hold the view that in the meantime, robots will be

such that they are easy to adapt and teach new tasks through clever tools and user interfaces, instead of simply knowing how to do each task out of the box. For most commercial applications, being able to teach a robot in a couple of days how to perform a new task reliably is more than enough. This vision has served as our guiding light.

2 APPROACH

To tackle this problem, we investigate different approaches, the parts of which can be roughly split into the following three categories.

2.1 *Leveraging multi-view geometry and 3D representations*

The vast majority of computer vision methods in use today, are inherently 2-dimensional. They only operate on individual images. Robots on the other hand operate in 3D environments, requiring us to somehow solve for the third dimension. One way of doing that, is to use multiple views of the scene to build a 3D representation. As data is noisy and algorithms are bound to make mistakes, one way to fix some of those mistakes is to use a 3D representation and disambiguate the predictions from individual views by comparing information across views and solving for the most likely state of things.

In all parts of this thesis, we make use of multiple view geometry and build 3D representations. Not only does 3D consistency help us, we live in a 3-dimensional world and therefore also need to teach the robot about distances and the scale of things.

2.2 *Learning from offline data*

When us humans learn a new task, we don't start from scratch. We have all been shaped by a lifetime of experience that we can lean on. Many skills tend to transfer from one task to another.

Most robot perception systems in use today, are mostly designed and trained from scratch to do a specific task. To build computer vision systems which can be taught with ease by their operators, we somehow need to raise the abstraction level such that the system can learn from few examples. One

approach to this, is *self-supervised learning* [53]. Self-supervised learning is making use of unlabeled data to learn representations which can be used in other, downstream tasks. The learning happens by either leveraging some known structure in the data or otherwise using labels which can be automatically computed from the raw data itself. In the second half of this thesis, we investigate the use of self-supervised learning techniques by leveraging models that have been learned offline without making use of any manual expert annotations. By observing large quantities of real images these self-supervised learning techniques learn representations which describe data that actually occurs in the wild. This can be seen as biasing the system towards real data ignoring implausible configurations that don't occur in the data. We use models learned with self-supervision to build 3D representations that can be used as a starting point for teaching the robot what it needs to learn.

A related technique is the use of *weakly supervised learning*. Weakly supervised learning is often taken to mean using either a large amount of raw data with small amount of labels or abundant labels which are noisy and from an unreliable source. In the first part of this thesis, we make use of semi-supervised learning by taking a few annotations and propagating them to unlabeled data. In the last part of this thesis we make use of vision-language models, which are weakly supervised. They are learned on a large dataset of image-caption pairs that are automatically collected from the internet. These pairs are used by correlating the information in the image with that in the caption through a common representation for both the image and the text caption. This is a form of weak supervision, as there is no way of knowing to which degree the automatically collected caption is actually related to the content in the image, but on average, they tend to have something in common.

2.3 *Making use of offline optimization and user-interfaces*

One strategy we make use of is to take data that is collected online from our robots to process and learn from offline, with the help of the end-user of the robot system. The benefit of processing data offline are many-fold. Firstly, you can spend as much compute as you want and aren't restricted to the computational resources onboard the robot. Secondly, when processing

data offline, you have the benefit of knowing what happens in the future and use that information to inform the representation in the first timesteps of the recording and ultimately end up with a more accurate representation. Thirdly, we can make use of the user of the system who can look at the data and enhance it with whatever information we want to augment our representations with. Some of this information might be application specific, allowing the user to shape the system to their needs.

The resulting representations which are computed offline, can then be used to improve the models running onboard the robot, or distilled into models which are real-time capable and operate on data that is available at run-time.

In all parts of this thesis, we make use of data collection and offline optimization.

CONTRIBUTIONS

> In science if you know what you are doing you should not be doing it. In engineering if you do not know what you are doing you should not be doing it.
>
> *Richard Hamming*

This chapter summarizes the main scientific contributions of the publications included in this thesis. It also lists other contributions in the form of other co-authored papers, workshop participation, supervised student projects and open software developed over the course of the doctoral studies.

1 PART A: KEYPOINTS

PAPER I

Kenneth Blomqvist, Jen Jen Chung, Lionel Ott and Roland Siegwart, 2022, August. Semi-automatic 3d object keypoint annotation and detection for the masses. In 2022 26th International Conference on Pattern Recognition (ICPR) (pp. 3908-3914). IEEE.

Context

3D semantic keypoints have become a popular representation to use within object manipulation. The main reason being that they are easy to interpret. Many objects can easily be described as a set of the most meaningful points relevant for a specific task and most tasks are such that the goal can be defined in terms of the position of the chosen semantic keypoints, relative to each other and the enviroment. Previous works [37, 71, 76] have relied on object instance segmentation masks, making it expensive to apply to new tasks and objects, as thousands of such masks need to be annotated to train

an object instance segmentation system such as Mask-RCNN [47] to reliably segment the objects of interest. These are very expensive to produce, easily a minute of annotation time per frame [75]. Another limitation of previous methods, is that to build a dataset of 3D semantic keypoints, they either rely on 3D reconstructions [37], ruling out transparent or otherwise hard to reconstruct objects, or they require instrumenting the environment with markers [71].

Contribution

In this project, we sought to investigate whether we could create an easy to use object keypoint detection pipeline that would neither require lots of manual data annotation, nor modifying the environment in any way. We wanted to build something where all we had to do, was drive up our mobile manipulation system to the objects we wanted to manipulate, scan them and then provide a couple of clicks to teach the robot about semantic points of interest.

This paper introduces a full 3D semantic keypoint detection pipeline, which can be taught entirely through the camera footage and propriopceptive information collected by the robot itself. It introduces a stereoscopic user interface through which scans of the objects taken by robot can be annotated and then propagated throughout the scan to obtain annotated 3D keypoint and image pairs. To avoid the need for an external object instance segmentation system and ground truth segmentation labels, we propose the concept of a center map, which associates each detected keypoint with its center. This center map is predicted and learned jointly from the little information provided by the user. It can then be used at runtime to associate keypoints with their respective object, enabling the detection and tracking of multiple objects in a frame simultaneously, which is a common requirement for many applications.

The paper shows that the pipeline is able to learn how to predict the 3D semantic keypoint location within roughly centimeter accuracy on held out test data, using only a couple dozen object scans as training input. These couple dozen scans can be annotated using the presented tool in about 15 minutes of expert annotation time, bringing it within reach of one-off industrial applications where a robot needs to be taught to perceive a class of objects.

Interrelations

This paper shows that proprioceptive information and multiple-view geometry can be leveraged within the annotation process, maximizing the value of human annotation time and bootstrapping a custom perception system from scratch. This project led to the idea in Paper II. While in this project, we specifically shied away from segmenting the scene, it did make us think about whether it would be possible to similarly bootstrap a richer representation of the scene than just a set of sparse keypoints on objects. For many manipulation tasks, simply knowing the position of specific points on objects is not enough. Having a dense representation which could tell us more about the geometry and semantics of different parts of the scene, is required for many applications.

The keypoint detection task of this paper is still relevant though. The segmentation, while richer, is not sufficient for all tasks, some tasks being such that the orientation or position of specific points in the scene is important. The user interface from this paper, could be integrated into the interface from Paper III, allowing the estimation of both sparse and dense information. Additionally, combining both keypoints with a surface reconstruction of the scene would allow for estimating the pose and shape of objects, potentially opening up the way for building object databases, an idea we describe in more detail in the conclusions chapter.

2 PART B: SEGMENTATION AND SYNTHESIZING DATA

PAPER II

Kenneth Blomqvist, Jen Jen Chung, Lionel Ott and Roland Siegwart, 2023, May. Nerfing It: Offline Object Segmentation Through Implicit Modeling. In 2023 proceedings of the International Conference on Robotics and Automation. IEEE.

Context

In the previous project, we had shied away from segmenting objects, as it is expensive to do. Neural Radiance Fields [79] had recently become popular

as a scene representation promising both impressive surface reconstruction performance and the ability to synthesize novel viewpoints given a scan of the scene. The view synthesis properties had been explored extensively from a photorealism perspective, but no-one had investigated whether the radiance field could be used to generate training data for downstream machine learning algorithms, such as object detection [47], segmentation [73] or pose estimation [141]. Such machine learning algorithms are very data hungry and have a tendency to overfit and learn shortcuts to produce an answer which is correct for the training data, but does not generalize well to unseen examples. If the knowledge baked into the 3D radiance field representation could be used to diversify the training data, it could result in better performing models which are more robust to changes in viewpoints and which could be learned from much fewer input examples.

Contribution

What we wanted to find out, was (1) whether we could use the radiance field representation to produce object segmentation maps of objects and (2) use the radiance field to generate training data for a downstream learning algorithm. In this paper, we propose a way in which the object segmentation can be extracted given a single 3D bounding box for each object in the scene. The paper also proposes an algorithm to generate novel but plausible viewpoints of the objects in the scene. The segmentation masks are built by extracting the density field within the 3D bounding box and rendering the mask for input viewpoints. Novel viewpoints are synthesized by first constructing a bounding box from the input camera poses and subsequently sampling camera poses within the constructed bounding box, such that the objects of interest in the scene are still visible.

The results show that computing segmentation masks using a radiance field representation yields much higher quality segmentation masks than baseline methods. The paper shows that the synthesized examples are of good quality, both in color and segmentation mask and that complementing real data with synthesized data improves results when learning a downstream object detector.

Interrelations

In Paper I, we explicitly didn't want to segment objects, as producing segmentation masks takes a lot of time from an expert. Here, we wanted to tackle that problem head on by fitting a neural radiance field to the scene and using the inferred geometry to create object instance segmentations. In the previous project, the bottleneck for dataset creation was no longer annotating the data itself, but rather collecting a diverse and representative set of data. As neural radiance fields are able to synthesize novel viewpoints of the scene, we can use the representation to generate color and depth image pairs and their corresponding object segmentation mask, further alleviating the data generation problem for downstream tasks.

The results in this paper did convince us of the promise of radiance fields and convinced us to keep working on neural fields, which led to the ideas of both Paper III and IV. The method for generating synthetic training examples could be applied in the pipeline of Paper III. Future work might also investigate additional randomization of the scene, for example randomizing textures or the object positions.

3 PART C: INTERACTIVE AUTOLABELING

PAPER III

Kenneth Blomqvist, Lionel Ott, Jen Jen Chung and Roland Siegwart, 2023, October. Baking in the Feature: Accelerating Volumetric Segmentation by Rendering Feature Maps. In 2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)

CONTEXT

Recently, systems such as SemanticNeRF [161] and iLabel [162] had been proposed that are able to take sparse semantic labels and propagate them through a scene. We were inspired by these methods' impressive ability to propagate sparse 2D information through a 3D representation and generalize across a scene. However, after trying this type of labeling paradigm, we realized that they still require a fair amount of annotation time per object to

arrive at a high quality segmentation of an object. Our initial tests showed up to a minute of annotation time across many viewpoints to perfectly segment a single object. The thing about SemanticNeRF [161] and iLabel [162], is that they blindly map points in the volume of the scene to radiance, density and sparse semantic labels. They can therefore only reason about spatial information and color information, through the latent representation that the radiance field learns. They make no use of outside data or information from other scenes.

In this project, we set out to investigate whether we could leverage information learned from other scenes and instill *useful bias* into such interactive labeling systems. Reasoning directly on color information has shown itself to be a tricky in computer vision. Using higher-level representations, such as learned features, has shown itself to be a much more effective approach. A dense feature representation of an image is more informative, as it takes into account the local neighborhood of individual pixels and semantic meaning of each individual pixel through the learned representation. We figured, we might as well not only regress the color information, but also the extracted features, which would instill a lot of semantic information into the implicit representation, essentially for free.

CONTRIBUTION

This paper proposes a 3D feature field representation and a user interface through which this representation can be annotated. The feature field is a vector field, mapping 3D points in the volume of the scene to image features. The features correspond to features extracted using conventional feature extractors, in this case DINO [18] features, which are learned in a self-supervised way on very large datasets. The feature field is learned jointly with a radiance field by volumetrically rendering feature maps for each observed viewpoint of the scene and supervising on feature maps extracted from each image. The paper also shows that volumetrically parametrized neural radiance field representations suffer from an overfitting problem when used to infer semantic labels of the scene from sparse annotations, for example as provided through brush strokes drawn through a user interface. To alleviate this problem, the paper introduces a hybrid positional encoding

which allows the neural field to better reason about spatial location within the volume, improving results significantly.

The experiments show that by leveraging the self-supervised features and the hybrid positional encoding, the labeling time is greatly reduced and much higher quality segmentation maps can be achieved with the same level of expert supervision than with previous methods.

INTERRELATIONS

Similarly to Paper II, this method tackles segmenting objects and 3D scenes and generating high quality data for downstream learning algorithms and addresses the problem we were avoiding in Paper I. The goal of this thesis is to bootstrap visual perception algorithms, this work tackles that by combining self-supervised pretraining and 3D consistency through a scene representation. It provides a user interface through which the user can specify their intent, and annotate data for use in downstream learning tasks, ultimately increasing the bitrate at which data is generated for downstream machine learning systems. In this paper, we introduce the feature field representation, which served as the basis for the Vision-Language Feature Fields which we propose in Paper IV. The user interface of this paper could be extended to include keypoint annotation, as proposed in Paper I.

4 PART D: WEAKLY SUPERVISED SEMANTIC LEARNING

PAPER IV

Kenneth Blomqvist, Francesco Milano, Jen Jen Chung, Lionel Ott and Roland Siegwart, 2023, October. Neural Implicit Vision-Language Feature Fields. In 2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE.

CONTEXT

The goal of this thesis is to bootstrap robotic perception. Vision-Language Models (VLM) [90] were recently proposed as a way to learn transferable visual representations by correlating image captions to images. They do so by

mapping both images and their captions into a common vector space. This type of weakly supervised learning is very enticing from a perceptual bootstrapping perspective, as it opens up the door to learning visual and semantic representations from the internet. For such representations to be useful in robotics, we need to be able to connect and ground these representation to the 3D world the robots operate in.

Given that these vision-language representations are just features extracted from images, and that they can be connected directly as such to vector valued language representations, we came to wonder if we could embed them into a feature field, and use the resulting representation to reason about the 3D scene. Some early work on using CLIP features [90] within 3D representations had been very recently unveiled [87, 105], but none of these jointly learned both the geometry with the semantic representation. We figured we could do both of these jointly, and that neural fields would prove to be the right way to approach the problem.

CONTRIBUTION

This paper proposes a neural implicit vision-language feature field scene representation which is learned jointly with a radiance field of the scene. The neural representation has a compact memory footprint, can represent the geometry of the scene with a high level of detail, is differentiable and can be built up incrementally in real-time as the robot explores a new scene. The representation allows for open-vocabulary semantic queries of the scene, allowing the localization of objects within the scene or segmenting different classes from each other. These queries can be run at very high rates of several million queries per second using commodity hardware.

The experiments in the paper show how this method enables zero-shot semantic segmentation of scenes, where labels can be provided at run-time and the scene can instantly be re-segmented into the given labels. To the best of our knowledge at the time, this paper introduced the first real-time vison-language neural field running onboard a robot. The paper includes a series of real-time experiments.

INTERRELATIONS

This paper builds on top of the contributions of Paper III. Papers II through IV all deal with segmenting 3D scenes, although in very different ways. While the method presented in this paper ultimately will not produce as good of a segmentation as the method from Paper III, it is entirely unsupervised. Therefore, this method could be used as a starting point for the interactive segmentation method in Paper III. That is, first fit a feature field using the best vision-language model available to us, then take the list of objects that we are currently trying to teach our robot and infer an initial segmentation of the scene. The robot's teacher can then use the user interface from Paper III to correct any mistakes made by the zero-shot system. Once this has been done for a sufficient amount of scenes, the corrected semantic fields could be distilled into an online 2D vision model that estimates the segmentation at run-time at high rates in real-time.

A limitation of our method, is that while the method is able to semantically segment scenes, it is not able to distinguish instances of objects from each other. This is something we explored in a subsequent master's thesis project, developing a panoptic version of this vision-language feature field algorithm. The method we developed in that project makes use of 2D instance segmentation proposals and uses them to learn an additional instance descriptor field through a contrastive loss function. These instance descriptors can be used at run-time to segment object instances from each other through clustering. We wrote up this work into a paper which is currently under review.

This paper also presents a solid step forward towards the goal of bootstrapping robot perception by learning from one of the largest and richest sources of data known to man, images on the internet. While the method presented does effectively solve the problem of grounding dense vision features in 3D spatial representations, the features used as input to the method are crucial to the performance of the method. The initial results presented here do suggest that this is a very fruitful avenue for future work towards the goal of this thesis.

Vision-language models still stand to improve a lot, especially ones which produce dense features and enable segmenting the resulting feature maps with natural language prompts. Since this work, a number of works have tried to improve on 2D open-vocabulary segmentation [19, 70, 88, 95, 149, 159] by learning on automatically collected datasets, but having tried some of

these, we can say that there is still a lot of room for improvement. Looking forward, given some of the impressive results on foundation models, we expect the day will come when these weakly supervised vision language models will surpass fully supervised segmentation models simply through the fact that they are able to make use of much more data and therefore will be more robust and generalize better.

5 LIST OF PUBLICATIONS

The research conducted during the execution of this doctoral thesis led and contributed to the following publications.

5.1 *Publications included in this thesis*

[P1] Kenneth Blomqvist, Jen Jen Chung, Lionel Ott and Roland Siegwart, Semi-automatic 3d object keypoint annotation and detection for the masses. In 2022 26th International Conference on Pattern Recognition (ICPR) (pp. 3908-3914). IEEE.

[P2] Kenneth Blomqvist, Jen Jen Chung, Lionel Ott and Roland Siegwart, Nerfing It: Offline Object Segmentation Through Implicit Modeling. In 2023 proceedings of the International Conference on Robotics and Automation. IEEE.

[P3] Kenneth Blomqvist, Lionel Ott, Jen Jen Chung and Roland Siegwart, Baking in the Feature: Accelerating Volumetric Segmentation by Rendering Feature Maps. In 2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE.

[P4] Kenneth Blomqvist, Francesco Milano, Jen Jen Chung, Lionel Ott and Roland Siegwart, Neural Implicit Vision-Language Feature Fields. In 2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE.

5.2 *Other publications*

[P1] K. Blomqvist*, M. Breyer*, A. Cramariuc*, J. Förster*, M. Grinvald*, F. Tschopp*, J. J. Chung, L. Ott, J. Nieto, and R. Siegwart. Go Fetch: Mobile manipulation in unstructured environments. *Workshop on Perception, Action, Learning, International Conference on Robotics and Automation (ICRA)*, 2020.

[P2] Kenneth Blomqvist, Francesco Milano, Jen Jen Chung, Lionel Ott and Roland Siegwart. 2023, Grounding Pretrained Features in 3D Representations *ICRA2023 Workshop on Pretraining for Robotics (PT4R)*, 2023.

[P3] Nicolas Gorlo, Kenneth Blomqvist, Francesco Milano and Roland Siegwart 2023, ISAR: A Benchmark for Single- and Few-Shot Object Instance Segmentation And Re-Identification *In 2023 IEEE winter conference on applications of computer vision (WACV)*.

[P4] Maurits Reitsma, Kenneth Blomqvist, Francesco Milano and Roland Siegwart Under pressure: learning based analog gauge reading in the wild *International Conference on Robotics and Automation (ICRA) 2024* Under Review.

[P5] Haoran Chen, Kenneth Blomqvist, Francesco Milano and Roland Siegwart, Panoptic Vision-Language Feature Fields *In IEEE Robotics and Automation Letters*

6 CONFERENCE AND WORKSHOP ATTENDANCE

The research conducted was presented at multiple international conferences, workshops and seminars.

1. Go fetch: Mobile manipulation in unstructured environments *Workshop on Perception, Action, Learning*, International Conference on Robotics and Automation (ICRA), June 2020, virtual.

2. Semi-automatic 3d object keypoint annotation and detection for the masses *2022 26th International Conference on Pattern Recognition (ICPR)*, August 2022, Montreal, Canada

3. Nerfing It: Offline Object Segmentation Through Implicit Modeling *International Conference on Robotics and Automation (ICRA*, June 2023, London, United Kingdom

4. Grounding Pretrained Features in 3D Representations *ICRA2023 Workshop on Pretraining for Robotics (PT4R)*, June 2023, London, United Kingdom

5. Baking in the Feature: Accelerating Volumetric Segmentation by Rendering Feature Maps *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, October 2023, Detroit, Michigan, United States

6. Neural Implicit Vision-Language Feature Fields *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, October 2023, Detroit, Michigan, United States

7 STUDENT SUPERVISION

This section lists all the student projects that have been supervised over the course of the doctoral studies.

7.1 *Master's thesis*

The Master's thesis is a six-month long, full-time project.

1. Joel Bachmann, "Unsupervised Learning of 3D Object Representations, 2020

2. Boyang Sun, "Integrated Perception and Control for Robotic Manipulation", 2022

3. Michael Schmid, "Model Adaption for Predictive Control in Aerial Manipulation", 2022

4. Haoran Chen, "Panoptic Vision-language Feature Fields", 2023

7.2 *Semester thesis*

The semester thesis is a semester-long, part-time project.

1. Pelzmann Nicolas, "Fast Adaptation of Dynamics Models", 2020
2. Jonas Frey, "6D Object Pose Estimation", 2020
3. Cyrill Hedinger, "Mesh Hole Detection and Filling Using Shape Completion", 2020
4. Caroline Sauget, "Typing Support for People with Autism", 2021
5. Paul Fitz, "Teleoperation for Teaching Robots", 2021
6. Matias Turkulainen "Object Pose and Shape Estmation", 2022
7. Silvio Mazzuco, "3-D Aware Superpixel Segmentation", 2022
8. Xiang Liu, "Generalization of Rendering-based Semantic Segmentation, 2022
9. Maurits Reitsma, "Analog Gauge Reading", 2023
10. Nicolas Gorlo, "Single-Shot Object Instance Segmentation and Re-identification", 2023

7.3 *Bachelor's thesis*

The Bachelor's thesis at ETH is a semester-long, part-time project.

1. Tim Reinhard and Nicolas Gorlo, "Object Recognition and Visual Servoing for the VTOL UAV Geranos", 2022

7.4 *Perception and Learning for Robotics course*

The Perception and Learning for Robotics course is a semester long project-based course where students work on a project part-time in pairs.

1. Jonas Frey and Yash Vyash, "6D Object Pose Estimation", 2020
2. Bin Yang and Zongrui Yu, "Scene Understanding and Generalization Through NeRF", 2023

8 LIST OF OPEN-SOURCE SOFTWARE

This section lists the open-source software that has been implemented and released over the course of the doctoral studies.

1. `mpm`: A CUDA accelerated implementation of the material point method `https://github.com/kekeblom/mpm`
2. `HUD`: A package to build OpenGL based user interfaces rapidly `https://github.com/ethz-asl/hud`
3. `object_keypoints`: A package for quickly learning to detect semantic keypoints on objects in 3D [11] `https://github.com/ethz-asl/object_keypoints`
4. `moma`: A software stack for mobile manipulation research `https://github.com/ethz-asl/moma`
5. `autolabel`: A package to semi-automatically annotate 3D scenes [12, 13] `https://github.com/ethz-asl/autolabel`
6. `analog_gauge_reader`: A package for reading analog gauges from images `https://github.com/ethz-asl/analog_gauge_reader/`

CONCLUSION AND OUTLOOK

> Sometimes science is more art than science. A lot of people don't get that.
>
> *Rick Sanchez*

In this chapter, we discuss the key takeaways that resulted from the research carried out, we propose promising future research directions and give a subjective outlook about the field.

1 DISCUSSION

In this thesis, we have presented multiple ways of creating machine learning based perception systems, that do not require thousands of hours of annotation time to build a dataset of input and output examples. We have presented different ways of bypassing this process be that by leveraging automation, constraints, generating additional data, using weak forms of supervision or making use of self-supervised learning. In Paper I, we used robots to collect data and leveraged calibration and multiple view geometry to automate the dataset creation process. In Paper II, we leveraged radiance fields to create object segmentation maps and to augment a dataset of object scans. In Paper III, we showed how we can use self-supervised feature models to accelerate interactive 3D segmentation. In Paper IV, we leveraged weakly supervised semantic feature models and ground these features in 3D maps for scene understanding.

The title of this thesis might be a little bit hyperbolic. Supervised machine-learning is definitely here to stay. Systems such as neural radiance fields and self-supervised learning systems do make use of supervision, they just do this by directly regressing on signal which is already in the data, or they regress quantities which can be automatically computed. This appears to be a very powerful approach. From our results and experience building these projects, it seems quite clear that the days of annotating large datasets

for strongly supervised machine learning are numbered. In none of these works did we have to annotate so much data that it is impossible for a single engineer to annotate. It very much appears that the path forward is to learn rich representations using this type of self-supervision, and then just define the objects, classes, goals or concepts using a few examples and constraints which we can define on the learned representations.

2 OUTLOOK

The goal of this thesis was to investigate how will we be teaching the robots of the future new perception skills. Here, we provide our views on how we believe robots of the future will be taught.

We believe that systems for teaching robots perception skills in the future, will make use of the following ideas:

- Large scale offline optimization
- Interactive tools
- Multiple view geometry and 3D representations
- Using multiple modalities and sensors
- Synthesizing data
- Weak forms of supervision
- Dataset curation

Here, we dig into each of these ideas, provide our perspective on how they tie into teaching robots perception skills and highlight some open problems.

Figure 1 shows how these components might fit together. Our view is that the way forward is to build a positive feedback loop closing state estimation through offline state optimization, a scene database, dataset curation and a machine learning pipeline back into better online state estimators. In the case of actually autonomous systems such as industrial robots, drones or autonomous vehicles, expert users and engineers primarily interact with and improve the system by monitoring and teaching the system through an interactive UI. Large scale foundation models, such as ones learned from videos or pictures on the internet, might be used in the offline state estimation

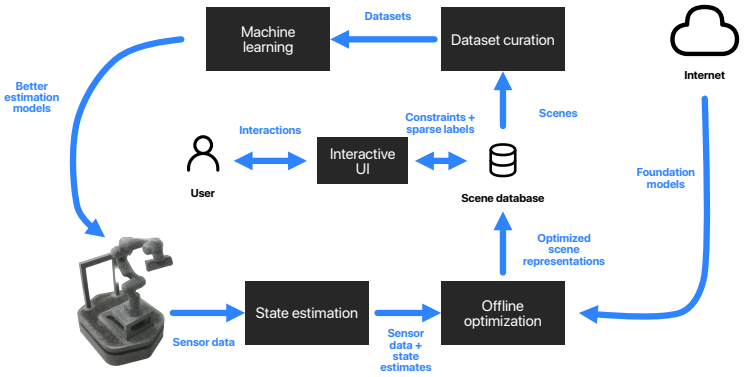


Figure 1: Architectural diagram of envisioned production pipeline.

step to enrich and bring context to the data collected from the robot. Datasets are curated per task for specific onboard machine algorithms by specialized algorithms and finally learned by running the curated datasets through a machine learning pipeline.

Next, we dive into each of these components and wonder what they might look like.

2.1 *Large scale offline optimization*

Mostly, our approach in this thesis has been to collect data offline, process it and then use that data to learn online models. We still very much believe in this approach. Processing data offline has many benefits. First of all, you have access to future information, which gives access to additional constraints. The data can be processed both forwards and backwards, giving an additional source that can be used to disambiguate any errors. Secondly, offline means that we are not constrained to process the data using the limited computational resources found onboard our robot, but we can use however big computers we want and we can run them for as long as we want. This opens up a lot of possibilities. We firmly believe that future systems will keep making use of this.

We posit that in the future, spatial AI systems will evolve in two directions. There will be an offline version of the system that will make use of every trick in the book and large scale compute. This will be complemented with a lightweight online version, which is able to make do with information available online and using compute available onboard the system. The offline systems will be used to curate and generate datasets to be distilled into components of the online system.

This might lead to a reinforcing feedback loop where data is used to improve components onboard the robot, the output of which are used to improve onboard state estimation which are logged and used as starting points for the offline state estimation system which looks at the data from a mission and tries to figure out the most likely true state of the scene at each point in time. This will lead to higher quality data which will lead to even more accurate onboard systems and so forth.

A skeptic might claim that any bias within such a system that feeds back on itself will get amplified to a point where that system is unusable. In this case, we believe this will be mitigated by strong optimization constraints and geometric assumptions, which make sure false information and noise can be optimized out. The remaining errors might be corrected by expert users equipped with interactive tools which we talk about in the next section.

2.2 *Interactive tools*

To check the operation and output of such a system, interactive tools will be needed to go through log recordings, visualize the output of optimization steps and the inferred state of the scene across time. Should there be any errors, smart tools built into the user interface will be used to correct any mistakes and provide further information.

The interactive labeling tool presented in Paper III shows an example of this. Another example of this is the Maplab [23] console, which allows users to configure settings and launch processing steps to optimize maps offline, visualize the results and launch further optimization procedures. The optimized maps can then be exported to be used by an online system. This is an early example of learning offline to improve online performance, but we believe future systems will also learn parametric models which will be used online, not just map representations and sensor parameters.

Our interactive labeling tool from Paper III, could be extended to deal with dynamic scenes and more complex scenarios by adding tools and capabilities to it such as the Segment Anything Model (SAM) [59], Video Object Segmentation (VOS) [146], feature and object tracking. To deal with non-static scenes, a user could initially segment the dynamic objects in the scene using SAM and the segmentation mask produced by SAM could be propagated to future frames using VOS. Once an object has been segmented in all frames, features within the mask could be used to track and reconstruct the object. Observations belonging to the dynamic objects would be ignored by the static scene reconstruction, and possibly all combined and jointly optimized similarly as in [63]. These object reconstructions could be used to build a database of objects relevant to the system, to be identified in future logs collected by the robot and further refined by more optimization across scenes. The constructed object models could be used by the online system to assist with object detection, pose estimation and other state estimation tasks, if relevant to the application.

Also, with the advent of SAM-like [59, 160] segmentation models, objects can be discovered autonomously, enabling completely unsupervised 3D object instance segmentation. Detected object instances can be compared and identified against each other across scenes to ultimately build an object database.

2.3 *Using multiple modalities and sensors*

Each sensor technology and modality has their strengths and weaknesses. Some things are very easy to perceive using one sensor, while they can be very hard to perceive with another. A textureless wall can easily be perceived with a LiDAR sensor but presents a challenge to RGB cameras. Budgets permitting, it makes sense to use all the sensors at your disposal, and combine measurements from all of them to build the most accurate 3D representation possible. Such sensors and modalities include, but are not restricted to, inertial measurement units, LiDAR, radar, sonar, event cameras, tactile and force-torque sensors. Each of these modalities should be connected to the scene representation. Constraints can be defined on and across each of the modalities to disambiguate information and improve the state representation. Should there be a large error in the optimization

problem for certain modalities and time steps that cannot be corrected using the interactive tooling, these sections of logs could be ignored in the dataset curation and generation phase, as there is likely to have been an error at some point. In our systems from Paper III and IV, such errors would occur if a frame is misregistered by the SLAM or structure-from-motion system, in which case it would both deteriorate the quality of the 3D representation and produce garbage output when rendering segmentation masks for those frames.

While in Paper I, we did make use of the positional encoders in the joints of our robot, we mostly dealt with RGB-D camera footage in this thesis. In our NeRF systems from Paper II, III and IV, we made use of depth measurements computed by the camera, learning the representation with an additional metric loss. In the case of neural implicit representations, the system is easy to extend with further optimization loss terms. In the case of a multi-camera rig, frames from all cameras can easily be added to the optimization problem by sampling pixels from those cameras. LiDAR supervision has already been explored by [94, 125].

Works have also explored learning neural implicit representations that are built on top of a signed distance function representation of the scene [67, 137, 138]. Using a signed distance function has many benefits, specifically for robotic applications. As the representation used by these systems is fully differentiable, the surface normal of the surface can be computed by computing the gradient of the distance function with respect to the input coordinates. Incorporating feedback from other modalities such as force torque sensors and position encoders also becomes easier, as you can constrain the zero level-set of the distance function at locations sensed to be a hard surface. The downside of such representations, is that they are slightly more computationally expensive to compute than simply inferring the radiance field of the scene. Especially for a real-time system, this is a bit of a challenge, but one that will likely be solved with smarter algorithms and processors better suited for computing neural fields.

Cross modality optimization has been explored in the context of structure-from-motion and SLAM through bundle adjustment [1], which solves a non-linear optimization problem, optimizing camera poses, camera parameters and landmark locations by minimizing reprojection error of landmarks matched in images. In many cases, IMU measurements are also used jointly as optimization constraints [23]. Systems such as Maplab [23] already make

use of multiple modalities and are able to refine maps and sensor rig trajectories, combining visual, LiDAR, IMU and GPS measurements. Future work might focus on even larger optimization problems, which make use of dense representations. A challenge with cross modality is the weighing of the different modalities and optimization constraints. In most cases, it is impossible to prove an optimal weighting for each measurement and error. Optimization weights might be set by the user through the interactive user interface to optimize qualitative state estimation accuracy. Once good settings have been identified for a specific robot sensor rig, they might not need further adjustments. An art, but with clever tooling and a fast feedback loop, this might not be a big problem.

Multi-modality is one source that can be used to disambiguate information. Another, is using multiple views and constructing a common 3D representation of the scene to correct errors and disambiguate between noisy sources, which we talk about in the next section.

2.4 *Multiple view geometry and 3D representations*

Some people have the perspective that using 2D camera information alone is enough to solve all or most computer vision tasks. They might think that all you need as a powerful model and an active learning loop to iteratively refine that model until the task is solved. We very much believe the opposite.

One issue with any individual predictions, is that they are almost certainly sometimes wrong. When only using a single view, you kind of have to blindly trust whatever inference is done on that one view. One might have several ways or models to estimate the same information, and these predictions might be combined, but any bias in the input view will be reflected in all results computed from that data. Using multiple views allows disambiguating errors made in any individual view.

Even if it was possible to perfectly solve 2D perception on RGB images, robots still operate in 3D environments. Therefore, one would still have to infer the third dimensions and the geometry of the scene. For this reason alone, we believe building persistent 3D representations is crucial for robotics.

One problem with using multiple views, is that often these different views are collected over time. Most 3D reconstruction methods assume that the environment remains static over those observations. This is definitely not

always the case. While it has been shown to be possible to use information learned on static scenes to for example track or estimate the pose of dynamic objects [77], solving the full problem of dealing with movement will eventually be necessary as we move towards more complicated tasks. The assumption of a static scene is a strong one, and one which is hard to let go of. In dynamic scenes, some details might only be visible in a single view, which makes triangulation impossible and some form of prediction or prior might be required to make the problem feasible to solve. In the case of neural radiance fields, some progress has already been made by leveraging object segmentations, tracking and alpha blending [63] to deal with moving objects and reconstructing them over time.

When using offline optimization and interactive user interfaces, one might solve the problem in stages by first building an initial model of the scene, then detecting the moving objects before asking the user to correct any mistakes and finally running further optimization to build a final refined dynamic model of the scene and the objects within.

Once the scene state has been reconstructed in a good model, one can imagine using this model to synthesize further data and simulate new plausible configurations of the scene.

2.5 *Synthesizing data*

In Paper II, we presented a system capable of synthesizing RGB frames and segmentation masks for unseen views of the scene. These examples can be fed into 2D perception algorithms to improve their robustness to changes in viewpoints. Similar results have been observed by [80, 127, 148, 148]. We believe that when used in moderation, such synthetic training data can be a big help. One issue is that the data generated is not quite of the same quality as the real data. There will be some differences in sharpness, unobserved parts of the scene will not be of good quality and there will always be some artifacts, however small. The synthetic data reflects the conditions present when collecting the data it was derived from, meaning many things such as object positions, lighting conditions and so forth cannot be randomized, further biasing the datasets towards the training conditions. Therefore, further randomization of the scene configurations might be required to really make synthetic data work. If the scanned scenes could be automatically

edited into new plausible configurations from which realistic samples can be drawn from, it would make bootstrapping perception systems much easier for new applications.

In the case of NeRFs and other implicit scene representations, one challenge is that they are hard to edit. Therefore generating randomized scene configurations with objects rearranged, randomized textures and lighting or other environmental conditions changed remains challenging. The computer graphics community has studied scene relighting [24, 98, 150, 158], which might be applicable or yield some insights into how this could be achieved. Other work has focused on editing neural radiance fields [22, 46, 54, 62, 151]. Such techniques could be applied to automatically edit and randomize scenes scanned by our robots to diversify the datasets derived from them.

Another benefit of building high fidelity reconstructions of the environment, is that it makes it easier to construct realistic simulations of scenes relevant to the robot. This seems to be used to some extent in the autonomous driving industry, where components are tested in simulations against rare synthetic scenarios that are simulated within detailed maps built from real roads. We expect this to find its way into other robotics applications, as the tools and methods to build such simulations develop.

2.6 *Weak forms of supervision*

In Paper IV, we made use of weak supervision in the form of Vision-Language models trained on image caption datasets. We believe this to be a very rich form of supervision and a promising path forward. Further, connecting the 3D representations with language semantics is powerful and enables a number of applications in planning through such representations.

The image-caption datasets on which vision-language models used today are trained on, are simple still images scraped from the internet. By correlating the captions with the images, one can learn powerful representations, but learning about the dynamics of scenes, causal relationships or about time dependent effects is hard. In the future, we envision that more complex representations and reasoning will be learned from video datasets. Early examples of such datasets include the Ego4D dataset [39].

The largest limitation holding our system back is the features used to construct the vision-language feature field. The LSeg model is not very good

on many classes and it has been shown that the fine-tuning [32, 55] done on the CLIP model leads to catastrophic forgetting and reverses much of the learning done on the massive weakly supervised dataset. Methods such as CLIP [90] predict single vector valued embeddings for entire images, but do not learn dense visual features which could be directly segmented or grounded in a feature field. Some methods have tried to learn such features purely from weak supervision via multi-scale fusion and evaluating the learned CLIP visual encoder over the image at multiple scales [58]. This has proven to work reasonably well, but does come with a high upfront preprocessing cost and doesn't currently lend itself to real-time methods. Others have studied how to learn more fine-grained outputs using a CLIP style contrastive objective [92, 145], by leveraging both self-supervised pretraining and showing that using max-pooling to aggregate dense features in the loss function learns better features that can more easily be segmented. [149] quite similarly made use of sorting the feature maps to learn a better representation. Such methods have yet to be demonstrated for higher output resolutions, but with some further improvements, they might lead to scalable self-supervised methods which yield pixel-aligned features that are robust in the long tail of classes that are not present in curated, closed-set datasets. Combining these methods with information inferred by other foundational models, such as segmentation models [59] could be another interesting direction to explore. So far, vision-language models have made mostly use of image-caption pairs, learning from video and motion, could also be useful specifically for robotics applications.

Another challenge with grounding these types of pretrained features into 3D representations, is that the features chosen have to be viewpoint consistent. This means that if you observe the same 3D surface point from two different directions, the feature corresponding to that point should be the same regardless of the viewing direction. If the features are learned purely from image-caption datasets, this property is hard to enforce. Some early work has explored enforcing motion consistency within self-supervised DINO features [157], with promising results. Such methods would likely also help in learning dense vision-language features and feature fields. Another avenue to make progress on this front, is to design a curriculum which maximizes learning, can make use of several modalities and datasets and avoids catastrophic forgetting, which is what we talk about in the next section.

2.7 *Dataset curation*

With few exceptions, up until this point, machine learning within robotics has been constrained by the amount of data available from real robots performing in real environments. Using the techniques presented in this thesis and leveraging ideas presented in this chapter, the day is likely to come when we will have more data than we might be able to productively process. The question then becomes "which of this data do we use and how?".

Vision-Language models and Large Language Models are already facing this problem. Data is uploaded to the internet at rates far outpacing what can be processed by current computers. A lot of this data is very repetitive and certain issues and themes get a lot more air time than others. Another example is autonomous driving, where there is no shortage of highway driving data, but complicated rare situations are hard to come by. This biases any models learned on this data towards the trivial information, instead of focusing capacity on the subtle, interesting and hard cases, which are often the important ones for systems deployed in the real-world.

Recent work in large language models and vision-language models has shown that big gains can be made by finding hard examples [44, 89, 143] within these image-caption datasets. [89] showed that trivial examples in these datasets, such as product shots are not very helpful in training these models. Such examples are usually simple pictures with a very clear connection between the caption and the image. They also find that CLIP is biased towards text, and filter out examples containing text, as they don't want the model to learn to read, but rather the semantic and spatial context in the images. Therefore, designing effective curricula might be a good path forward to building more capable weakly supervised models. In the case of robots, once we do have a database of scenes which far outpaces what we can use and train our models on, it becomes important to not only filter out cases where not much is happening in the scene, but also mine effective training examples from the remaining data, such that we maximize learning. The field of continual learning might also yield some insights into how to construct an effective curriculum without forgetting previously learned information.

3 GENERAL OUTLOOK

This thesis has been uniquely concerned with building perception components as part of a modular manipulation system with other components dealing with motion planning, control and high-level planning. Recently, impressive results have been published on end-to-end learned systems [16, 108], which tackle and learn all of these components jointly within one machine learned architecture. These systems typically include very minimal assumptions on the intermediate representations used and simply see all of these steps as a single end-to-end function approximation problem.

The rise of these fully machine learned methods has stirred a lot of controversy within the robotics research community. In one camp are the traditionalists which think robotics only works if you build in many assumptions, run simple control loops at very high rates and carefully engineer every single part of your system. On the other side are the learning maximalists who believe that end-to-end learning is the only way to go and thinking about the specifics of individual tasks and problems is futile. All you need is data. As of now, convincing evidence exists on both sides.

The learning maximalists are quick to point out The Bitter Lesson [122], which states that in AI, only methods which leverage computation has prevailed and that when given a choice between two methods, you should pick the one which performs better under the assumption of infinite compute and data. The transformer type architectures indeed do benefit from more data and compute.

On the traditionalist side, the evidence lies in the fact that most impressive results in robotics, such as from legged locomotion [57], mostly use simple models with explicit physics and minimal learning. End-to-end learning has made limited progress in localization and mapping, where the state-of-the-art systems still make lots of assumptions, are carefully tuned and engineered. While the demos are cool on the transformer side of things, the systems have yet to demonstrate that they are capable of performing tasks day in, day out with a very high level of reliability, which the explicitly modeled systems have. A challenge with the end-to-end learned systems, is that they tend to operate at much lower rates than traditional control systems, as they require many more floating point operations to compute a single action. This could change with more powerful hardware and more clever engineering.

We for one, currently stand in the middle. We find it hard to believe that robots of the future will simply have one big machine learning model as a brain. We believe that any practical system will consist of many components which you can inspect and ensure the correct operation of. These individual components will increasingly be, and already are, learned from data. The point about computation from the Bitter Lesson is a good one, and we believe that one of the biggest things holding robotics back at the moment is the lack of structured data to learn from. Every robot is different, datasets are inconsistent, hard to combine and a lot hardware-specific issues need to be considered even when learning from data. Even if individual components would start merging together or some transformer-like learning machine were to take over the scene, it would likely benefit from obtaining state estimates or other information obtained from computer vision systems. We believe the most promising path forward is to design methods which solve this dataset creation problem and benefits all learned systems, whether modular or monolithic.

Part A

KEYPOINTS

PAPER I

SEMI-AUTOMATIC 3D OBJECT KEYPOINT ANNOTATION AND DETECTION FOR THE MASSES

Kenneth Blomqvist, Jen Jen Chung, Lionel Ott, Roland Siegwart

ABSTRACT

Creating computer vision datasets requires careful planning and lots of time and effort. In robotics research, we often have to use standardized objects, such as the YCB object set, for tasks such as object tracking, pose estimation, grasping and manipulation, as there are datasets and pre-learned methods available for these objects. This limits the impact of our research since learning-based computer vision methods can only be used in scenarios that are supported by existing datasets. In this work, we present a full object keypoint tracking toolkit, encompassing the entire process from data collection, labeling, model learning and evaluation. We present a semi-automatic way of collecting and labeling datasets using a wrist mounted camera on a standard robotic arm. Using our toolkit and method, we are able to obtain a working 3D object keypoint detector and go through the whole process of data collection, annotation and learning in just a couple hours of active time.

Published in:
IEEE International Conference on Pattern Recognition (ICPR), 2022

1 INTRODUCTION

Most modern computer vision methods use large datasets to learn to predict features at run time. These have been demonstrated to enable many new capabilities in robotic object manipulation. While the methods are impressive, they are data hungry and require sizeable datasets of ground truth annotations to train. If we could quickly and cheaply create datasets, we could expand to more environments and enable many downstream tasks.

The data requirements force researchers of downstream robotics tasks to either use standard objects, for which trained models and computer vision pipelines have been made available, or a large investment has to be made upfront to collect and label a dataset. Creating a dataset requires either hand-labeling thousands of frames one-by-one, having a data collection setup with environment markers, as done in [71], or a tool such as LabelFusion [77] can be used to partially automate the annotation process. However, LabelFusion requires mesh models of the objects. Creating a known model for objects in turn requires a high-fidelity object scanning setup, which is often unavailable. It also requires the objects to be rigid, or additional parameters need to be estimated to model deformation. Additionally, the objects and environment have to be such that depth sensors are able to accurately measure depth, excluding reflective or transparent objects.

In this paper, our goal is to track category-level semantic points in an object's coordinate frame relative to the camera frame for downstream robotic manipulation tasks. "Category-level" meaning that objects vary, but the intra-category semantic meaning of keypoints are the same. Specifically, we want a system with the following properties:

1. Can estimate 3D object keypoints on arbitrary objects
2. Requires little effort to handle novel objects
3. Can be used in the wild without having to use markers, motion tracking systems or otherwise modify the environment
4. Does not rely on accurate depth sensing
5. Can track multiple objects simultaneously in the image frame

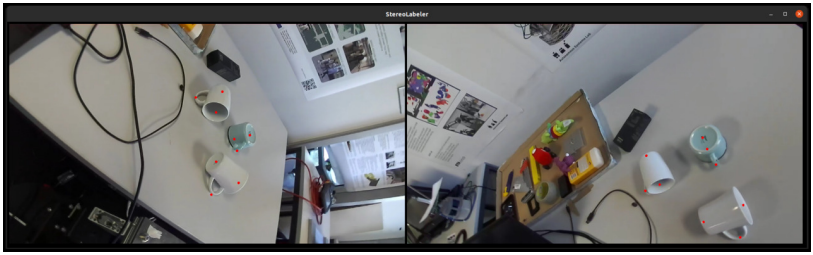


Figure 1: StereoLabel, our keypoint labeling tool. The user is presented with two images of the scene to label. The images are selected to maximize the orthogonality of the views.

Existing methods, such as PVN3D [48] and kPAM [37] require semantic segmentation maps to train or they rely on external object instance segmentation. Semantic segmentation maps are time consuming to annotate, making the systems more expensive to deploy in new scenarios and for new objects.

In contrast, we present a complete 3D object keypoint tracking system, including both a learning-based object keypoint algorithm and a method to very quickly obtain the training labels needed by the algorithm. Our method builds on the insight that we can forgo using semantic segmentation maps to distinguish between objects, if we instead introduce a center keypoint and predict a center map that associates each keypoint with a center keypoint. The amount of objects in the scene is inferred from the amount of detected center keypoints. This makes the labeling task a lot faster, as we can simply label 2D keypoints instead of having to also create dense instance segmentation masks.

We present a way to speed up data collection by capturing many views of the scene and propagating labels from two labeled viewpoints to all the others. We show that by calibrating our robot and making use of calibration and the kinematics of a robot arm, we can forgo using a motion tracking system or environment markers, as done by previous works. Using our system, data can be collected in the wild wherever our robot goes. This means that our tools can be deployed directly on the hardware intended for the downstream robotic task, streamlining the full problem definition and solution by avoiding additional steps. Calibrated robots are now commonly

available and by using one, the data collection can be further automated and enables collecting data autonomously.

We show two different versions of our learning-based algorithm that leverages our data collection pipeline to track keypoints of multiple objects in a scene. The first one uses both views of a stereo camera. The other one is a variation of the algorithm that can work with a monocular RGB camera.

We validate our method and tracking pipeline in experiments on two different object keypoint tracking scenarios. The first one is a single object valve tracking scenario. The second is a multiple object cup tracking task, showcasing that we can handle multiple objects simultaneously in a frame. We show that using only 22.5 minutes of recorded data across 45 sequences, and using less than 15 minutes of labeling time, we can learn a model that can track keypoints on objects of interest. We demonstrate that the resulting tracker is accurate enough to enable manipulation tasks, such as rotating a valve.

Code for our project is made available at `github.com/ethz-asl/object_keypoints`.

2 RELATED WORK

2.1 *Datasets and Labeling Tools*

Several object pose datasets have emerged which use ground truth meshes. The most commonly used meshes are of the YCB object set [17]. The YCB-video dataset from [141] provides labeled 6D poses for objects in RGB images. The authors demonstrated that the dataset was capable of training their PoseCNN 6D object pose estimator. An initial estimate of the object poses from PoseCNN were used to generate the YCB-M dataset [40]. This dataset was collected with seven depth cameras and they used fiducial markers and depth refinement to obtain the common frame of reference between cameras. While a robot arm was used to facilitate data collection, the authors did not use the kinematics of the robot nor did they use hand-eye calibration in the labeling process. Moreover, both of these datasets are limited to the YCB object set. [129] uses object models, simulation and rendering to obtain a dataset of ground truth object poses.

While other labeling tools exist to create datasets of a priori unknown objects, these often have other limitations. [116] provides a semi-automated tool for creating 2D and 3D bounding box labels for multi-object scenes in RGB-D video. Their algorithm uses a GrabCut-based approach [97] to interpolate annotations over timesteps. However, the user still needs to adjust the propagated bounding boxes in each of the following frames. LabelFusion [77] can handle cluttered scenes, however, object meshes are required and must either be given or created manually using a scanning routine (e.g. with a handheld scanner or turntable). Our proposed method avoids this requirement altogether.

Finally, several methods train keypoint detectors using only a small set of labeled data. Simon et al. [111] bootstrap a keypoint detection dataset for hand pose estimation using multiple views of the scene. The authors ensure that each iteration introduces new information via multiview geometry. However, because of this, performance is tied to the number of cameras in the setup. Multiview geometry is used by [147] for human and animal pose estimation by deriving a differentiable semi-supervised loss function which is equivalent to minimizing epipolar divergence. They show that they can train a keypoint detection network using a large set of unlabeled images and comparatively few labeled images.

2.2 *Object Keypoints and 6D Pose Estimation*

Methods exist which predict keypoints, in 2D or 3D, to calculate the 6D object pose. Some estimate 2D points on an RGB image and solve for the pose using a PnP algorithm [50, 85, 86, 126, 152]. Others predict keypoints directly in 3D space [48, 123]. 6-PACK [135] presents a way to track single objects in real-time using keypoints which emerge in an unsupervised way. As the keypoints are learned end-to-end, additional components such as an attention mechanism are required in their keypoint tracking pipeline. S3K [133] is a self-supervised approach to learn semantic 3D keypoints. Similar to our approach, the authors use multiple camera views to propagate labels across images. However, in their case, they require a four-camera setup while our method is designed to work with a single camera. NOCS [136] uses a representation shared within an object category. The authors learn a model to regress to this representation from RGB and depth maps. PVN3D [48]

learns a model which produces semantic segmentation maps as well as per pixel keypoint and center votes from RGB-D frames. Similarly to PVN3D, we use a center prediction map to track multiple objects. However, PVN3D uses ground truth semantic instance segmentation maps to distinguish objects from each other, which are hard and expensive to label. We avoid this by associating keypoints directly with their corresponding object's center. This also circumvents the need for the expensive clustering step to aggregate pixel-wise predictions.

KPAM [37, 76] presented a way to track category-level object keypoints. However, their system can only track single objects due to the integral pose regression step it relies on [120]. Furthermore, for the same reason, it can't deal with many keypoints of the same type. KeyPose [71] is an object keypoint detection method and dataset, which also uses stereo views of a scene. This method is only applicable in single object scenes; detected keypoints are not associated to objects, which makes tracking multiple objects infeasible. Further, KeyPose only works with objects that have unique keypoints. Modeling objects such as the valve in our experiments is not possible, as it has several ambiguous keypoints. This limitation is due to the spatial softmax operation that is used in the output heatmaps. The dataset collection method proposed by KeyPose relies on fiduciary tags that are placed in the scan environment. We propose and demonstrate the feasibility of a method that does not require modifying the environment and that relies solely on a calibrated robot with a camera.

3 METHOD

Here we describe the components of our framework: the hardware setup, the procedure to collect video sequences with camera poses, our algorithm to compute ground truth labels, a stereo multiple object keypoint detection pipeline and a variation that uses only a monocular RGB camera.

3.1 *Hardware Setup*

Our method requires a calibrated image sensor along with a way to control it into different known viewpoints. For this, we use a StereoLabs ZED Mini stereo RGB camera, mounted on the wrist of a Franka Emika Panda robot

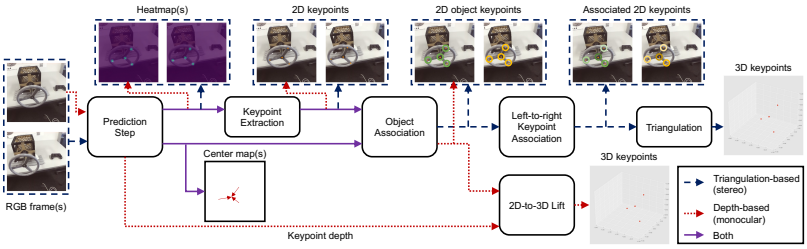


Figure 2: The components for both of the proposed keypoint tracking pipelines.

arm. The robot arm has accurate encoders at each joint which give readings on the position of each joint. Using a model of the robot and the position readings, we can accurately compute the position of the wrist, relative to the base frame of our robot.

We compute the intrinsic parameters and left-to-right-camera transformation of the stereo camera using Kalibr [84]. The wrist-to-camera frame transformation we calibrate using the ethz-asl/hand_eye_calibration package [36].

3.2 *Data Collection*

For training a neural network to detect keypoints, we need a dataset of image frames and a set of keypoint locations for each image. To obtain these, we collect 30s long sequences while our robot scans the target objects, observing the objects from multiple viewpoints. We save the pose of both camera frames, relative to the base frame of the robot and the RGB frames from the left and right camera sensors. In the next section, we describe how we obtain the keypoints in image coordinates for each image.

3.3 *StereoLabel: Labeling and Generating a Dataset*

For each object category of interest, we define a set of keypoints that is most convenient for manipulating objects from the category. For example, in the case of coffee cups, we can define the keypoints to be the bottom center, center top and the outermost point on the handle of the cup (see Fig. 1). This

allows us, if desired, to solve for the orientation of the cup. Or, we can grasp the cup by approaching the cup from the top center and grasping the side wall of the cup with a parallel jaw gripper. If there are several ambiguous keypoints, as is the case for the valve in our experiments (see Fig. 3(a)), then all occurrences of the keypoints are labeled and considered to be of the same type.

We developed a tool, StereoLabel, to label 3D keypoints from a sequence of images taken from different viewpoints. Fig. 1, shows the tool in use. The user is shown image frames from two viewpoints. The viewpoints are picked such that the z-axes of the image frames are as close to perpendicular as possible. The image frame is defined to be z-axis forward, y down and x to the right of the image. The user labels 2D keypoints on both frames by clicking on the keypoint location in the image. In case a specific keypoint is occluded or otherwise hard to pinpoint, the user can swap out either frame with a new one.

Once corresponding keypoints are labeled, we triangulate their 3D positions in the base frame of our robot using the homogeneous direct linear transformation method [4]. We backproject the triangulated points to both frames using each frame's projection matrix, so that the user can validate that the point was appropriately placed. The user can further validate correct placement by cycling through images in the sequence by pressing a button and checking the backprojected points. In addition to the labeled keypoints, we augment the set with one additional 3D keypoint; this is the average of all the other 3D keypoints which we call the center keypoint.

Once we have all the image sequences labeled and triangulated, we can generate a dataset for training a computer vision model. Fig. 2 shows the proposed triangulation-based (stereo) and depth-based (monocular) keypoint tracking pipelines. For each frame in a sequence, we create a set of ground truth heatmaps, one heatmap for each type of keypoint. Should there be several keypoints of the same type, we pack them onto the same heatmap. We compute the 2D image coordinate for each 3D keypoint by backprojection. We place a Gaussian distribution over the 2D keypoint location computed using an RBF kernel (output of the prediction step in Fig. 2). Finally, we normalize each heatmap to have values in the range $[0, 1]$.

As there might be multiple objects in a frame, we compute 2D vector fields with vectors pointing from non-center 2D keypoints to the center keypoint. We compute one vector for output pixel having a non-zero heatmap value.

With the center maps, we can associate keypoints to objects and detect multiple objects in a frame.

For the monocular version of our pipeline, we additionally compute a keypoint depth map containing the z-value of each 3D keypoint for each pixel within a fixed radius from each keypoint.

3.4 *Learning the Keypoint Network*

We use a convolutional neural network (CNN) to predict the heatmaps, along with center maps. We use CornerNet-Lite [64] as a backbone network. CornerNet-Lite is a stacked hourglass-style CNN architecture. The input is first downsampled through a series of convolutional layers and then upsampled through transposed convolutional layers in an hourglass module. Two hourglass modules are composed together.

We predict target maps with two prediction modules which take as input the output of each respective hourglass module. The prediction modules consist of three convolutional layers with batch normalization, 1×1 kernels with stride 1 and relu activation functions, except for the last layer. We use sigmoid activation functions at the heatmap heads and no activation function for the center map and relu for the depth map.

The input to our network has size 511×511 pixels and the output map resolution is 64×64. We initialize the backbone network weights by pretraining on COCO [68].

We use three types of losses to train our network: a heatmap loss, a center loss and a depth loss. For the heatmap loss we use binary cross entropy:

$$\mathbf{L}_h = -\sum_{c=1}^{C}\sum_{i=1}^{H}\sum_{j=1}^{W} y_{cij}\log p_{cij} + (1-y_{cij})\log(1-p_{cij}). \tag{1}$$

p_{cij} is the predicted heatmap value for a keypoint of type c at output index i,j. y_{cij} is the ground truth heatmap value for keypoint map c at index i,j. C, H and W denote the amount of keypoint types, and the height and width of the output maps.

For the center loss, we simply use a smooth L1 loss:

$$\mathbf{L}_c = \sum_{c=1}^{C} \sum_{i=1}^{H} \sum_{j=1}^{W} \text{smooth\_L1}(\hat{\mathbf{c}}_{cij} - \mathbf{c}_{cij}) \check{y}_{cij}, \tag{2}$$

where $\hat{\mathbf{c}}_{cij}$ is the center vector prediction for keypoint type c at index i, j. $\mathbf{c}_{cij}$ is the corresponding ground truth center vector. $\check{y}_{cij}$ is a binary value denoting whether the heatmap value for keypoint type c at index i, j is nonzero. The smooth L1 loss is squared below a value of 1 and linear otherwise and is applied elementwise.

To enable using a monocular camera, we additionally need an estimate of how far along the z-axis each keypoint is. To do this, we predict a pixelwise depth estimate for each keypoint type. We learn this using an L1 loss function:

$$\mathbf{L}_d = \sum_{c=1}^{C} \sum_{i=1}^{H} \sum_{j=1}^{W} \left\| z_{cij} - \hat{z}_{cij} \right\|_1 \check{y}_{cij}, \tag{3}$$

where z_{cij} is the ground truth depth value for keypoint c at location i, j, while $\hat{z}_{cij}$ is the corresponding estimate.

All losses are applied at both stages of the hourglass network and are combined by weighting parameters $\lambda_.$:

$$L = \lambda_h(L_{h1} + L_{h2}) + \lambda_c(L_{c1} + L_{c2}) + \lambda_d(L_{d1} + L_{d2}). \tag{4}$$

L_{h1} denotes the heatmap loss at the first hourglass, L_{h2} for the second hourglass, L_{c1} the center loss for the first hourglass and so forth. When training the triangulation-based pipeline, we set the depth loss weight λ_d to 0 to disregard it entirely. We train our network using the dataset generated in Section 3.3.

3.5 *Keypoint Extraction*

At runtime, we extract keypoint locations from the heatmaps by first applying a version of non-maxima supression, where we zero the non-maximum values in 5×5 regions surrounding each location. We then zero out all values below a threshold of 0.25. From each of the remaining heatmap values, we compute keypoint locations by weighing image indices by the predicted heatmap

density in a 5×5 region on the unprocessed heatmap predictions centered at the maxima location.

For each non-center keypoint, we compute the object center estimate by summing the center vector with the corresponding image index. We associate each keypoint with the center keypoint closest to the keypoint's predicted center position in pixel coordinates.

3.6 *Keypoint Association and Triangulation*

After predicting and extracting keypoints in left and right image frames, we need to associate each keypoint in the left frame, to its counterpart in the right frame. To do this, we select the keypoint in the right image where $\mathbf{x}'\mathbf{F}\mathbf{x}$ is below a cutoff value of 32.0. $\mathbf{F}$ is the fundamental matrix derived from the camera calibration, $\mathbf{x}$ is the homogeneous pixel coordinates of the keypoint in the left image and $\mathbf{x}'$ is the homogeneous pixel coordinates of the keypoint in the right image.

If several keypoints match, which happens when two keypoints are on the epipolar line, we shift the point by a fixed amount and pick the closest match. The fixed shift is equivalent to the difference in pixel coordinates between a point, projected onto both the left and right image frames, that is 60cm in front of the center of the left camera frame.

Finally, we triangulate the 3D location of the keypoints using the same direct linear transformation method used when creating the dataset.

3.7 *2D-to-3D*

In the monocular version of our pipeline, we use the depth prediction, combined with the camera matrix $\mathbf{K}$ to compute the 3D point $\mathbf{X}$ corresponding to the 2D detection $\mathbf{x}$:

$$\mathbf{X} = \mathbf{K}^{-1}\mathbf{x}\hat{z}, \tag{5}$$

where $\hat{z}$ is the depth estimate for keypoint $\mathbf{x}$.

4 EXPERIMENTS

We are interested in the following questions:

- Can we use our method to quickly build up object keypoint tracking datasets?
- Can we train our keypoint tracking method on an amount of data that can be easily collected by one user?
- Is the object tracking performance good enough to enable robotic manipulation?

4.1 *Valve: Single Object Tracking*

In this experiment, we track a valve with three spokes. We define four keypoints: one at the center hub of the valve, and three at the front center points where the spokes meet the rim of the valve, shown in Fig. 3(a). The three keypoints at the rim are indistinguishable from each other, and are thus considered to be of the same type and packed onto the same heatmap.

We collect 50 sequences of 30s using our data collection method, which we label using StereoLabel. The sequences differ in object arrangement, clutter, occlusion, background and lighting conditions. We split the resulting dataset into 45 sequences for training and 5 sequences for testing.

4.2 *Label Accuracy*

Our semi-automatic labeling approach has a few sources of error: synchronization between camera frames and joint encoder readings, intrinsics calibration error and hand-eye calibration error. These all result in some error in the triangulation and reprojection steps of our pipeline. Without the ground truth 3D keypoint locations, we instead manually label 2D keypoints frame-by-frame on 100 randomly sampled frames in our valve dataset and compare the human labels to ones produced by our system. This allows us to quantify how much the 2D keypoint labels drift as we observe the target object from different viewpoints. As the user does not always place the keypoints perfectly, even the human labels will have some error. We therefore

(a) (b)

Figure 3: (a) Valve setup showing keypoints for the value. (b) An image from the cup tracking scene.

establish a baseline by doing two manual frame-by-frame labeling passes to measure the variance.

4.3 *Comparison with KeyPose*

We compare our method against KeyPose [71]. KeyPose doesn't support multiple objects nor is it possible to detect keypoints on objects with multiple keypoints of the same type. We therefore can't run KeyPose on our datasets, but instead opt to evaluate our method on the KeyPose mugs dataset.

4.4 *Cups: Multi-object Tracking*

In this experiment, we seek to track up to four cups simultaneously in a scene. We collect a dataset with 100 sequences observing the cups from various viewpoints, varying the number of cups between 1 and 4 in the scene, changing the clutter, lighting conditions and background of the scene across sequences. For each scene, we randomly select between 1 and 4 cups from a set of 25 different cups. We split this dataset into 87 sequences for training and the rest for testing. We split the sequences such that 2 cups only ever occur in the test set.

5 RESULTS

5.1 *Valve: Single Object Tracking*

We timed how long it takes to label a sequence of images. Labeling a pair of valve images took us just under 15 seconds. With 50 sequences, this makes for a total of ~12 minutes and 30 seconds to label all 19'507 frames in our dataset.

Table 1 shows the keypoint tracking performance on a held out test set. **Mean** refers to the mean error of the 3D keypoints in centimeters. **xy** is the mean error, disregarding the depth axis in the left camera frame of reference. < 3 **cm** is the percentage of measurements that were within 3 centimeters of the labeled ground truth location. **25th** and **75th** respectively denote the 25th and 75th percentiles of the combined keypoint errors. GT refers to the tracking performance using ground truth heatmaps and center maps as input with the stereo pipeline, Stereo is the stereo pipeline with a learned model, Mono is using only the left view of our stereo camera and our monocular pipeline.

Both the stereo- and depth-based pipelines perform reasonably well. For both pipelines, errors are within the range of the width of a parallel jaw gripper, and much smaller in scale than the size of the object.

Valve Manipulation

We deployed our keypoint tracking system on a mobile manipulation system. The goal of the experiment was to use the system to rotate the valve in Fig. 3(a) using the manipulator. In this case, we know the type of the valve and have a CAD model of it. We first detect the valve using keypoint tracking, and when we have detected all four keypoints (center and three spokes), we further refine the pose using ICP to match the depth readings from our camera to the object model. After refining the object pose, we command the arm to track a trajectory that rotates the valve. See the supplementary video for a successful completion of the task.

5.2 *Label Accuracy*

Comparing our generated keypoints to a manually labeled dataset, we found that the mean label difference is 6.3 pixels on average with a standard deviation of 3.4 on images with a size of 1280x720. Comparing two manually and separately labeled instances of the same datasets yield a mean difference of 2.9 pixels with a standard deviation of 1.7 pixels. While the manually labeled examples have slightly less variance, they are of the same order of magnitude for both methods.

5.3 *Comparison with KeyPose*

Table 2 shows results on the KeyPose mugs dataset evaluating on the unseen `mugs_0` instance. Keypose performs slightly better. We attribute this to its more restricted problem formulation and the learned stereo image fusion employed in its network architecture.

5.4 *Cups: Multi-object Tracking*

It took us an average of 19 seconds to label a scene with 2 cups. Which makes for a total of roughly 32 minutes to label the 66'419 frames in our dataset.

Table 1 shows the accuracy when tracking multiple cups on a held out test set. Similarly to the valve tracking experiment, the error is larger in the depth direction. Performance of both the stereo and depth pipelines is acceptable, i.e. errors are within the width of a parallel jaw gripper. Performance is slightly worse than on the valve tracking experiment. However, we note that this is a harder task with several different objects and significantly more keypoint occlusion.

Failure modes for both pipelines include misdetecting a keypoint or associating a keypoint with the wrong object. Failure modes of the stereo pipeline also include misassociating keypoints from left-to-right and a bad triangulation due to slightly misdetected 2D keypoints. Additionally, both approaches fail when two keypoints of the same type align, either from the same or different objects, and occlude each other. In such cases, the keypoints will get detected as one and the center prediction might point toward either object in the case of multiple objects.

Table 3 shows how long each step of our stereo pipeline takes on average for our implementation. The measurements were made on a computer with an Nvidia RTX 2080 GPU and AMD EPYC 7742 CPU.

6 DISCUSSION AND CONCLUSIONS

In this paper, we presented a method to quickly collect and label object keypoint tracking datasets and a system that learns to recover the labels at runtime on unseen examples. We showed that we fully rely on calibration to avoid having to place markers in the environment. In experiments, we showed that we can generate accurate object keypoint labels much quicker than using a 2D labeling approach, while also annotating the z-dimension and without having to create segmentation maps. We showed that our presented system is able to detect keypoints on multiple objects simultaneously at real-time rates, using the produced datasets. We showed that it can be successfully used as part of a system to solve real world manipulation tasks.

While we presented two different keypoint tracking algorithms, the actual keypoint and object detection algorithm can be replaced with any other pipeline, as long as the labels can be derived from our data collection method. Additional information about the objects could also be used to further improve the estimated keypoints. The Perspective-n-Point algorithm could be used for known objects or a category-level object model could be fitted to the keypoint detections to further improve them, similar to what is used in [144]. Additional computation could be traded for accuracy by predicting keypoint heatmaps at a higher resolution. The 64x64 pixel output resolution we used is quite limiting.

One weakness of our data collection method, is that it relies on measurement timestamps from different sensors. On our platform, these are not hardware synchronized. Some cameras can be tightly synchronized, while triggering boards such as the VersaVIS board exist [132] which are able to synchronize several cameras and IMUs. Extending these to cover other types of sensors, such as joint encoders, would improve the usability and accuracy of our proposed method.

Finally, when collecting our data, we manually guide the robot into different viewpoints. This could be automated. Furthermore, the robot could semi-autonomously improve upon an initial learned model using an active

learning type approach. Different models and viewpoints could be used to bootstrap a dataset, similar to what is done in [111].

7 ACKNOWLEDGEMENTS

This project has received funding from the European Union's Horizon 2020 research and innovation programme under project PILOTING No H2020-ICT-2019-2 871542 and under grant agreement No 101017008 (HARMONY).

Table 1: Results from the valve and cup tracking experiments.

Valve					
Method	**Mean** (cm)	**xy** (cm)	< 3 **cm**	**25th** (cm)	**75th** (cm)
GT	0.39	0.18	99.3%	0.17	0.30
Stereo	3.63	1.43	59.5%	1.29	4.26
Mono	2.99	1.06	65.0%	1.61	3.55
Cups					
Method	**Mean** (cm)	**xy** (cm)	< 3 **cm**	**25th** (cm)	**75th** (cm)
GT	1.14	0.39	97.7%	0.09	0.22
Stereo	6.71	2.24	68.5%	1.29	3.53
Mono	3.1	1.56	62.2%	1.43	4.02

Table 2: Our monocular pipeline on the KeyPose mugs dataset.

Method	**MAE (cm)**	< 2 **cm**
KeyPose	1.6	78.6%
Ours monocular	2.0	66.4%
Ours Stereo	1.9	69.7%

Table 3: Time spent on each step of the stereo pipeline.

Stage	**Mean time (ms)**
Prediction	32.9
Keypoint extraction	6.8
Object association	0.62
Left-to-right association	0.1
Triangulation	0.2

Part B

SEGMENTATION AND SYNTHESIZING DATA

PAPER II

NERFING IT: OFFLINE OBJECT SEGMENTATION THROUGH IMPLICIT MODELING

Kenneth Blomqvist, Jen Jen Chung, Lionel Ott, Roland Siegwart

ABSTRACT

Most recently proposed methods for robotic perception are based on deep learning, which require very large datasets to perform well. The accuracy of a learned model is mainly dependent on the data distribution it was trained on. Thus for deploying such models, it is crucial to use training data belonging to the robot's environment. However, collecting and labeling data is a significant bottleneck, necessitating efficient data collection and labeling pipelines. This paper presents a method to compute high-quality object segmentation maps for RGB-D video sequences using minimal human labeling effort. We leverage the density learned by a Neural Radiance Field (NeRF) to infer the geometry of the scene, which we use to compute dense segmentation maps using a single 3D bounding box provided by a user. We study the accuracy of the computed segmentation maps and present a way to generate additional synthetic training examples observing the scene from novel viewpoints using the learned radiance fields.

Our results show that our method is able to compute accurate segmentation maps, outperforming baseline and state-of-the-art methods. We also show that using the synthetic training examples improves performance on a downstream object detection task.

Published in:
IEEE International Conference on Robotics and Automation 2023, 2023

1 INTRODUCTION

Robotic manipulation tasks require robots to detect and compute the pose of objects. The majority of perception methods used in robotics today are based on supervised learning [48, 69, 76, 141] and require large amounts of labeled training examples to fit parameters [121]. Many approaches have been devised to learn in an unsupervised or self-supervised fashion to circumvent the need for human annotated data. However, these tend to not work as well as supervised alternatives and can usually benefit from annotated data [56].

For semantic segmentation, images are usually labeled by drawing polygons on individual image examples, which can be extremely time-consuming. The cost of producing labeled datasets quickly exceeds levels that can be sustained by most robotics applications. LabelFusion [77] presented a system to rapidly annotate RGB-D data by building dense reconstructions of the environment. It can be used to compute 6D pose labels for objects with a known model. While this approach works very well when object meshes are given, in most cases, such models are not available. Additionally, intra-category variation or the fact that objects deform, can make relying on object models impractical. Having a tool that can very quickly provide ground-truth labels in the case of unknown object models, would allow us to deploy more powerful supervised learning algorithms where it previously has not been possible.

To overcome the need for ground truth mesh models of objects, we introduce a pipeline to generate semantic segmentation maps, 6D poses and 3D bounding boxes of objects for handheld RGB-D video sequences. Our method leverages neural radiance fields (NeRFs) [79] to recover the 3D structure of the scene and uses the learned NeRF model to compute object segmentation maps and bounding boxes of objects in each RGB-D frame.

NeRFs learn an implicit representation of a scene using only RGB images and known camera poses. As active depth sensors are increasingly common on robotic platforms, we propose to use depth measurements as an additional supervision signal to the NeRF model. We show that depth supervision reduces the amount of shape artifacts present in the recovered 3D shape which is crucial for generating accurate segmentation maps. Using NeRFs to represent the scene has the additional benefit that we can synthesize new viewpoints of the scene. We leverage this to generate additional synthetic training data examples for the task. We study the quality of the synthesized

training examples and show that these do in fact improve the performance of a learned model on a downstream object segmentation task.

To summarize, our contributions are as follows:

- A pipeline that computes scene reconstructions and high quality semantic segmentation maps for RGB-D image sequences using a depth-supervised formulation of NeRF.
- A method to generate additional training examples by synthesizing novel viewpoints and computing the target labels.

We evaluate our proposed pipeline on a large variety of different objects. We compare the quality of the produced labels against frame-by-frame annotated semantic segmentation maps and two different baseline methods. The first one using a ground truth mesh model of objects and the second using TSDF integration that does not require object models. Additionally, we compare against a state-of-the-art object annotation pipeline, Rapid Pose Labels [112]. We evaluate the effectiveness of our data augmentation scheme on a downstream object detection task.

2 RELATED WORK

2D ANNOTATION AND ACTIVE LEARNING Annotated images are needed for learning tasks such as object detection, keypoint detection or semantic segmentation. Human labeling can be expedited by tools that allow directly drawing on the image [99] or by reducing the problem of creating semantic masks to a keypoint annotation task [75]. Active learning [104] has also been incorporated to speed up the creation of ground truth datasets by applying pixelwise or viewpoint entropy [107, 109] to achieve equivalent performance using only a fraction of the training data. Such methods could very well be used in conjunction with our approach to make use of the different viewpoints we recover in the preprocessing step to further reduce the labeling burden.

3D DATA ANNOTATION While several 3D scene datasets exist [2, 9, 26, 27, 128] the goal of this work is to enable users to easily generate their own annotated datasets specific to their task. Existing tools can triangulate 2D annotations into 3D from multiple known viewpoints [11, 71]. Other

methods directly label objects in 3D using known object models [77] or through explicit scene differencing as objects are incrementally introduced [118]. RGB-D annotation can also be sped up by leveraging structure in point cloud data, either by propagating labels over successive frames [5] or via GrabCut [97] based approaches [116]. Each of these methods present restrictions either in terms of how the data is collected (manual modification of the scene, reliance on accurate depth) or they do not provide the full spectrum of 3D annotations (no segmentation masks, depth map or scene mesh).

Rapid Pose Labels [112] is the current state-of-the-art system for labeling object poses, segmentation masks, and bounding boxes for raw RGB-D video where CAD models of the objects are not available. Nevertheless, it requires multiple scans of the same object and cannot be applied to articulated or deformable objects. It assumes that depth measurements are available for all labeled 2D keypoints, hindering its application on reflective or transparent objects. Additionally, the method requires post-processing the generated object point clouds, which requires extra work per object type. Our method is able to address each of these limitations and we compare against Rapid Pose Labels in our experiments.

NEURAL RADIANCE FIELDS NeRF [79] introduced neural radiance fields as a way to encode the appearance of a scene into a neural network and realistically synthesize novel views of the scene. Since its inception, many variations have been introduced to improve object and scene recontruction using RGB only [148] or in combination with depth supervision [7, 29]. NeRFs have also been extended to infer semantic information [161, 162] and can also use this channel to improve geometry reconstruction [35]. Similar to [162] we make use of depth maps to supervise the NeRF model and produce high quality semantic segmentation maps with little human effort. However, our method does all learning and heavy processing in an offline step and does not require large amounts of computation while using the system making it suitable to run online onboard a robot.

3 METHOD

Our goal is to obtain high quality semantic segmentation maps, 3D bounding boxes and 6D poses of objects for each frame in a handheld RGB-D image sequence. Specifically, in the case where the objects vary from sequence to sequence and we do not have access to CAD models of the objects.

We propose a pipeline which consists of the following steps:

1. Compute camera poses for each frame.
2. Compute a 3D point cloud of the scene.
3. Learn a depth-supervised NeRF model of the scene.
4. Annotate objects with bounding boxes using a 3D graphical user interface.
5. Compute dense semantic segmentation maps using the learned NeRF model and the 3D bounding box labels.

3.1 *Obtaining Camera Poses*

As a prerequisite step, we have to compute camera poses for each RGB-D frame to propagate bounding box labels between frames and to train the NeRF model. In our experiments, we do this using hloc [100, 101], which we run on all the frames in our video sequence. We then scale the resulting trajectory by finding the scale factor that minimizes discrepancy between measured depth and points triangulated by hloc while filtering outliers in a RANSAC loop. However, we note that any other method can be used to compute metrically scaled camera poses. Given the camera poses and RGB-D frames, we can reconstruct a point cloud of the scene.

3.2 *Learning a NeRF Model of the Scene*

We use a NeRF [79], basing our implementation on JaxNeRF [28], to infer the geometry and appearance of a scene. The idea behind the NeRF algorithm, is to trace rays from known camera positions into the scene, and learn a radiance field that maps 3D points and viewing direction to color and density

values. The radiance field is modeled as a multi-layer perceptron (MLP). Image pixel values are obtained using differentiable volumetric rendering.

Neural radiance fields have many advantages. They learn a continuous representation of the scene that does not require setting any parameters, such as resolution, per-scene or task. The level of detail captured is mainly constrained by the capacity of the model learning the radiance field and the captured images. The only parameters that need to be set are the near and far bounds used for sampling. As we are dealing with RGB-D sequences, these can be set automatically to match the minimum and maximum depth readings in the clips. The MLP we use for predicting the density has 8 layers with 256 neurons and the view direction conditioned RGB network has a single layer with 256 neurons.

Following [79], we define a photometric loss:

$$L_{photo} = \left\| \hat{\mathbf{C}}(\mathbf{r}) - \mathbf{C}(\mathbf{r}) \right\|_2^2, \tag{1}$$

where $C(\mathbf{r})$ is the ground truth and $\hat{\mathbf{C}}(\mathbf{r})$ the predicted color.

In our initial tests, we observed that due to the shape-radiance ambiguity [156], reconstructing the scene using only the RGB images and known camera poses does not yield very good results for many types of scenes. Planar surfaces would get reconstructed as uneven or density would get assigned to free space. We therefore add a depth loss to help the model disambiguate and learn from the captured depth maps where available:

$$L_{depth} = \delta(\mathbf{r}) \left\| \hat{D}(\mathbf{r}) - D(\mathbf{r}) \right\|_1, \tag{2}$$

where $\delta(\mathbf{r})$ is 0 where we don't have a depth measurement and 1 otherwise, $D(\mathbf{r})$ is the ground truth and $\hat{D}(\mathbf{r})$ the predicted depth. We optimize the model using gradient descent, by minimizing the combined loss:

$$L(\mathbf{r}) = L_{photo}(\mathbf{r}) + \lambda_d L_{depth}(\mathbf{r}), \tag{3}$$

where λ_d is a weighting parameter to balance the photometric and depth loss. In our experiments, we study the impact of this parameter.

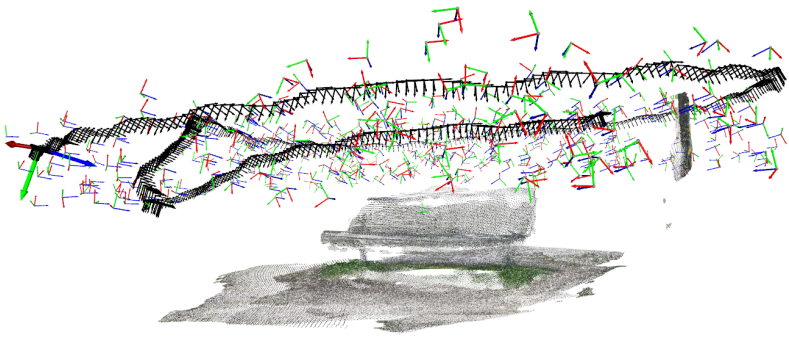


Figure 1: Example of sampled viewpoints. The original trajectory is shown in black, the red (x-axis), green (y-axis) and blue (z-axis) axes are the sampled poses.

3.3 *Computing Object Labels*

We use a graphical user interface to place 3D bounding boxes around objects of interest in the reconstructed point cloud of the scene. An example of this process is shown in the supplementary video. To compute segmentation masks, we render completed dense depth frames for each frame in our scan using the learned NeRF. We then convert each depth frame to a point cloud using the camera intrinsic parameters. Using the camera pose and object bounding boxes, we classify each point in the point cloud according to the object class and throw away points not belonging to any object. By re-projecting the object points back to the image frame, we obtain a dense segmentation mask of each object. We compute 2D bounding boxes for detection by computing the tightest bounding box containing the segmented object. Object poses can be computed by transforming the 3D bounding box pose into the camera coordinate frame.

3.4 *Generating Synthetic Training Examples*

NeRFs are able to synthesize high fidelity images from novel viewpoints, provided that the surfaces in the scene have been observed from a similar

viewpoint and the free space between the novel viewpoint and the scene surface has been observed. We design an algorithm to automatically sample suitable poses and compute color, depth and segmentation mask triplets for these novel viewpoints. In our experiments, we investigate whether such synthetically generated training examples actually improve performance of a learned model on a semantic object segmentation task.

To automatically sample viewpoints from a scene, we first compute a bounding box of camera poses in an object's coordinate frame and then sample camera positions uniformly inside this bounding box. We filter out positions which are too close to a structure as measured by distance to the closest point in the point cloud. Next we compute a viewpoint orientation that points the camera at an object in the scene. We then shift the orientation by a random rotation uniformly sampled from $[-\pi/4, \pi/4]$ rad for the x and y axis and $[-\pi, \pi]$ rad for the z-axis, where the z-axis points forward, x to the right and y downwards in the image. Sampled frames are visualized in Figure 1.

4 EXPERIMENTS

We capture a number of indoor and outdoor scenes with a variety of objects to evaluate our method. Data was collected using Apple iPhone 12 Pro smartphones, which are equipped with a time-of-flight depth sensor. We train on images that have a resolution of 960×720 pixels and depth frames have a lower resolution of 256×192 pixels, which we upsample to match the color images.

4.1 *Label Accuracy*

To validate the quality of the labels produced by our proposed method, we evaluate the semantic segmentation maps of our approach with two baseline methods and compare them to manually annotated semantic segmentation maps that are obtained by drawing polygon shapes over the objects on individual images. The first baseline method involves segmenting the scene directly from the captured depth maps using the user provided bounding box. To produce the segmentation map, we first convert the depth map to a point cloud using camera intrinsic parameters. We then classify each point as

being inside a bounding box or not and reproject them to the image frame to obtain the segmentation mask. The second baseline uses the mesh obtained through TSDF integration [25, 83] with a voxel size of 0.5cm and running marching cubes. We cut out the object from the mesh using the provided bounding box and render the cut out objects as a segmentation map.

Table 1 shows the mean intersection-over-union (mIoU) agreement across different scenes for the different methods against the manually annotated semantic segmentation masks; qualitative results can be seen in Figure 2. The NeRF-based method performs consistently better. As the depth maps captured by RGB-D sensors are noisy and have many pixels with no measurement, they produce significantly worse segmentation masks, especially for transparent and reflective objects. In some cases, the TSDF-based pipeline performs similarly to NeRF, especially for simple shapes where depth maps are of good quality, namely the fire hydrant and the park bench. Similarly, the TSDF baseline also struggles on transparent (wine glass, teapot) and reflective objects (cars, wine bottles). The NeRF approach does much better, though the masks are not quite as good in the case of the transparent objects. Figure 3 shows some of the failure cases of the NeRF-based approach, where object segmentation masks are not correctly inferred.

We compare our method to the state-of-the-art automated 3D object annotation method, Rapid Pose Labels [112]. Similarly, Rapid Pose Labels is able to produce dense segmentation maps and poses for objects. We compare the methods on an oolong ice-tea bottle dataset containing 5 different scenes[1]. Using Rapid Pose Labels we were able to achieve a mIoU agreement of **0.801** against hand-labeled examples. Our method in turn achieves an mIoU score of **0.902** across the same scenes, a considerable improvement.

4.2 *Depth Supervision*

As depth sensors can have a large amount of noise and missing values, we study the effect of using depth maps to supervise the NeRF model. We fit the NeRF using different weights λ_d on the depth loss and qualitatively analyze how well the resulting depth maps approximate the geometry of the scene. In Figure 4 we visualize produced segmentation maps obtained with different

[1] Available here at the time of writing: https://github.com/rohanpsingh/RapidPoseLabels

Figure 2: Segmentation masks obtained using our method on different scenes. Segmentations are in yellow. Scenes are in the same order from top-left to bottom-right as in Table 1.

levels of depth supervision. Table 2 shows the quantitative accuracy in terms of mIoU agreement across all the manually labeled examples.

4.3 *Novel View Synthesis*

As the learned NeRF is able to synthesise new viewpoints of the scene, we study the quality of the synthesized examples. To be of use in a downstream object segmentation, detection or pose estimation task, the produced color and depth would have to be qualitatively good and plausible. The segmentation masks would also have to stick to the object boundaries so as not to induce bias into the learned model.

We collect a dataset containing 19 scans of fire hydrants, which we split into 10 training scans and 9 test scans. We then annotate the scans using our method and create an additional synthetic dataset. We quantitatively verify the quality of the synthesized examples by labeling them by hand and comparing them against the produced masks.

Figure 3: In some cases, geometry might not get recovered properly, as shown with the hole in the laptop. The hole in the handle of the teapot is not correctly inferred as free space. In the other two cases, some density is assigned outside of the object.

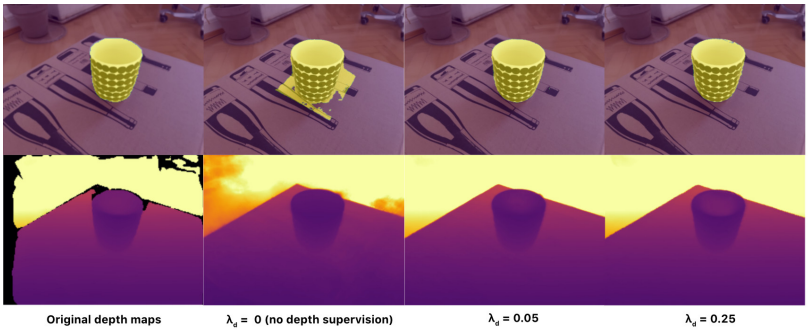


Figure 4: The produced depth and segmentation masks on the cups scene with different levels of depth supervision given during NeRF training. The leftmost image shows the original depth maps. We see that the NeRF model trained without depth supervision produces artifacts in the geometry which end up affecting the segmentation masks.

Column 5 of Table 1 shows the accuracy of the computed segmentation masks for images synthesized from novel viewpoints by the NeRF model. Figure 5 shows examples of synthesized color images and segmentation masks. The generated images are generally of good quality, but some suffer from visual artifacts, mostly on unseen parts of the scene or when observing a surface from an out-of-sample viewing direction or far away surfaces that have not been properly observed or are beyond the sampling range.

To study downstream task performance using the synthetic examples, we train two semantic segmentation models: one using only the original scans and another also using the synthetic examples. We then compare the

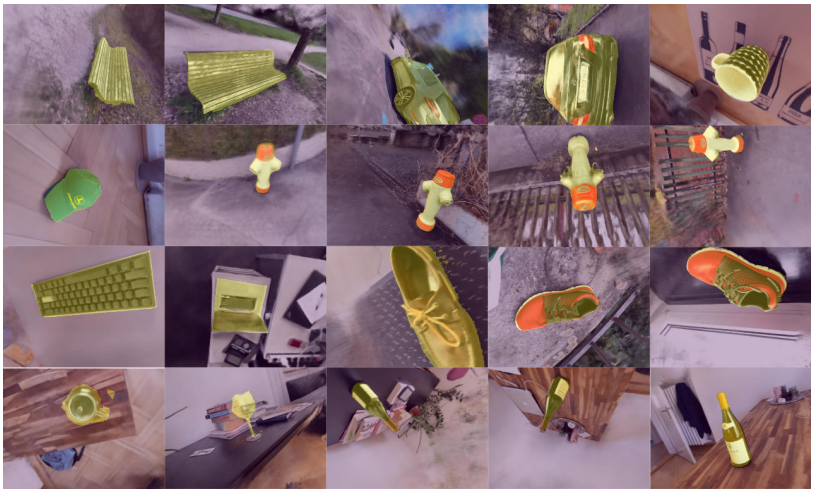


Figure 5: Synthetic training examples and their associated segmentation masks, generated using the proposed approach.

performance against manually labeled unseen examples to see if using the synthetic examples improve performance on this downstream task.

Figure 6 compares the accuracy of two different object detection models, Mask-RCNN [47] and Yolo-v3 [93], that were trained on the fire hydrant dataset with different ratios of synthetic to real data. For the Mask-RCNN model which does both segmentation and detection, performance on both detection and segmentation is shown. Yolo only does object detection, so we only report detection performance. Detection accuracy for both models increases up until 60% synthetic data, before decreasing. Object segmentation performance peaks a little bit earlier, likely due to the less realistic sharpness and texture of the synthetic examples, which are much more important for segmentation than bounding box detection.

5 DISCUSSION AND CONCLUSIONS

We introduced a pipeline that uses NeRFs to create dense pixelwise labels for semantic segmentation of objects. We showed that the geometry recovered

Scene	Method mIoU Real Data			Synthetic
	Depth	TSDF	NeRF	Mask mIoU
bench1	0.4054	**0.8650**	0.8581	0.9140
bench2	0.3664	0.4712	**0.9280**	0.9130
car1	0.2139	0.6960	**0.9323**	0.9307
car2	0.2761	0.7207	**0.9471**	0.9481
cup	0.9220	0.7153	**0.9498**	0.9385
hat	0.9392	0.9209	**0.9524**	0.8956
hydrant_1	0.7468	**0.8967**	0.8861	0.8710
hydrant_2	0.8502	0.9354	**0.9551**	0.9093
hydrant_3	0.8207	0.7373	**0.9033**	0.9066
hydrant_4	0.8078	0.7835	**0.9246**	0.8827
keyboard	0.9241	0.9352	**0.9397**	0.9087
laptop	0.8160	0.8686	**0.8743**	0.8783
shoe_1	0.9312	0.8666	**0.9373**	0.9452
shoe_2	0.9299	0.9184	**0.9755**	0.9568
shoe_3	0.9614	0.9491	**0.9719**	0.9047
teapot	0.5799	0.5257	**0.8458**	0.8664
wine_glass	0.0381	0.0000	**0.7565**	0.7272
wine_bottle_red_1	0.6877	0.7960	**0.8739**	0.8456
wine_bottle_red_2	0.6916	0.7046	**0.8879**	0.8862
wine_bottle_white	0.6670	0.8456	**0.9006**	0.8894

Table 1: Columns 2-4: mean intersection-over-union (mIOU) accuracy of segmentation masks for real examples, computed using the different methods. Column 5: mIOU accuracy of masks for synthetic examples.

by a NeRF can be used to generate high quality segmentation masks which outperformed baseline methods. We showed that the learned NeRF can be used to generate additional data in the form of unseen viewpoints of the scanned scenes which can be used to train an object detector, further reducing the data collection burden. A promising direction for future work would be to investigate methods that could further diversify the generated examples, for example by relighting the scenes.

scene $\backslash \lambda_d$	0.0	0.001	0.01	0.05	0.1	0.25
cup	0.691	0.755	0.920	0.9498	0.946	0.948

Table 2: Segmentation mIoU on the cup scene for different depth supervision weights λ_d.

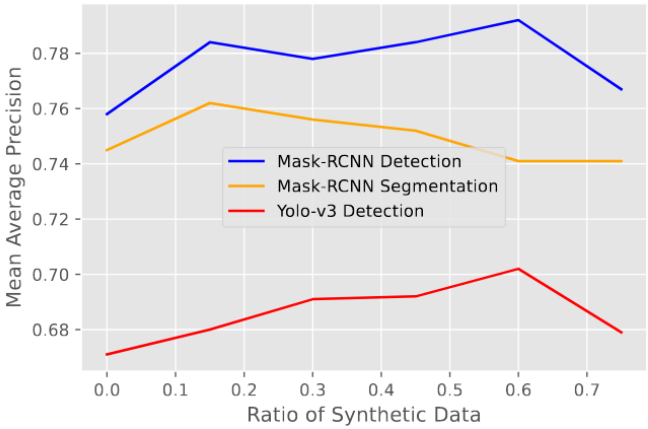


Figure 6: Average precision on the held out test set for models trained with different ratios of synthetic to real data.

Further research might focus on extending the method to scenes with moving objects or automating the placement of the bounding boxes, for example through active learning.

Our proposed method relies on accurate camera poses. Should the camera pose recovery step fail or produce bad results, the learned NeRF model will be severely impacted which leads to low-quality geometry reconstruction and thus poor segmentation masks. Estimating camera poses is an entire field of study in itself. However, this could be solved by mounting the camera on a manipulator and calibrating the system to obtain camera poses using proprioceptive sensor data, as done in [11].

As shown in the experiments, our method performs significantly better than the baseline methods on most objects, including transparent and reflective ones, but there are limits. As previously discussed, inferring the geometry of clear and fully transparent objects is still a challenge. Visual features in the images could be used to help the model cleanly segment and infer the density.

Our test scenes currently contain a single foreground object, often with other objects or clutter in the background. Since we use bounding boxes as

a source of supervision, in more cluttered scenes, other objects might enter an object's bounding box, producing noisy labels. Further work could go towards filtering label noise in such scenarios. Another source of error in the segmentation masks comes from the scene geometry not being perfectly inferred and label edges not matching the object's boundary. This could be addressed by further refining the scene geometry and labels by allowing a user to refine the labels online, providing fixes to the generated 2D segmentation maps and using the additional supervision to improve on both the representation and the generated semantics.

Part C

INTERACTIVE AUTOLABELING

PAPER III

BAKING IN THE FEATURE: VOLUMETRIC SEGMENTATION BY RENDERING FEATURE MAPS

Kenneth Blomqvist, Lionel Ott, Jen Jen Chung, Roland Siegwart

ABSTRACT

Methods have recently been proposed that densely segment 3D volumes into classes using only color images and expert supervision in the form of sparse semantically annotated pixels. While impressive, these methods still require a relatively large amount of supervision and segmenting an object can take several minutes in practice. Such systems typically only optimize the representation on the scene they are fitting, without leveraging prior information from previously seen images.

In this paper, we propose to use features extracted with models pre-trained on large existing datasets to improve segmentation performance on novel scenes. We bake this feature representation into a Neural Radiance Field (NeRF) by volumetrically rendering feature maps and supervising on features extracted from each input image. We show that by baking this representation into the NeRF, we make the subsequent classification task much easier. Our experiments show that our method achieves higher segmentation accuracy with fewer semantic annotations than existing methods over a wide range of scenes.

Published in:
IEEE International Conference on Intelligent Robots and Systems, 2023

1 INTRODUCTION

Neural Radiance Fields (NeRFs) [79] and other neural implicit representations have recently emerged as a popular representation for 3D scenes due to their many favourable properties. They can accurately infer the geometry of a scene by making use of strong geometric structure and rich supervision from captured calibrated images. They have few hyperparameters to tune and are able to handle a wide range of scales, geometries and materials.

As NeRFs use an MLP to map spatial coordinates to color values, they can easily be modified to predict other observable spatial properties. This led to NeRFs quickly being applied to semantic volumetric segmentation through SemanticNeRF [161], which is able to propagate semantic pixel labels from one frame of a scene to another and across image pixels through generalization. This was subsequently demonstrated within an interactive semantic segmentation system, iLabel [162], motivating the use of such a system to generate ground-truth data for downstream embedded, real-time computer vision algorithms.

While such systems achieve increasingly high accuracy as the amount of annotated pixels grows, they still require a lot of human supervision. SemanticNeRF used one labeled *randomly selected* pixel per class, per image. With over 900 training images in a scene, this is a lot of annotated pixels from a diverse set of viewpoints. If used to annotate data, providing such supervision can take an infeasibly long amount of time. An inherent limitation of current semantic NeRF approaches is that they blindly map 3D scene coordinates to radiance, density, and semantic class by optimizing a randomly initialized neural network per scene. Structure priors or previously learned information is not leveraged.

On the other hand, deep learning has fueled a whole body of work on representation learning, the goal often being to learn a feature representation that can be transferred to another task. Using pre-learned features in the context of NeRFs is not straightforward, as a NeRF maps scene-specific point coordinates to output values, which makes inducing desirable bias or structure into the model challenging. An example of this is *object* bias. For example, if a user of a semantic annotation system clicks on a cookie in a kitchen scene, the user might ideally want the system to, by default, automatically segment the entire cookie from other cookies and the background, without having to select each pixel.

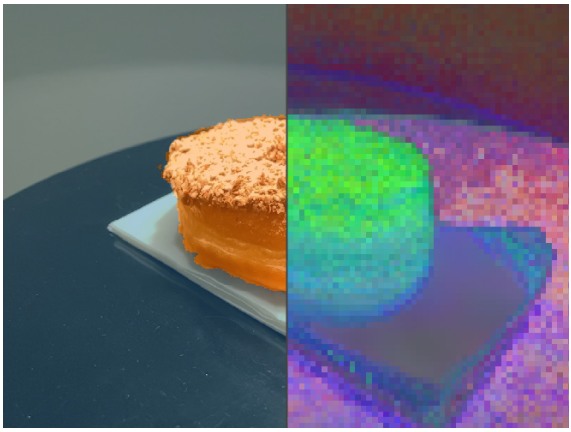


Figure 1: The left side shows an example image from our dataset, overlayed with a segmentation mask produced by our system. The right side shows DINO features, which we reconstruct in our algorithm, mapped to RGB values using PCA.

The goal of our work is to infer a dense 3D semantic segmentation of the scene, and infer dense 2D semantic segmentation maps for each input image, while spending as little expert annotation time as possible. To this end, we develop a user interface which allows a user to quickly segment RGB-D video sequences, by drawing sparse annotations on the images. While the user is using the system, the program infers a segmentation given the current user inputs and shows it to the user. The user can refine the segmentation, until they are happy with the result.

To infer the best possible segmentation using a sparse set of labels, we propose a novel algorithm that models the scene using an implicit NeRF representation and leverages semantic image features obtained using a neural network feature extractor. We supervise our implicit scene model on these features, effectively *baking* a feature representation into the learned MLP. We volumetrically render feature maps from the hidden layer activations of the semantic branch of the MLP, and minimize discrepancy between the rendered feature maps and extracted image feature vectors during training. This forces the MLP to encode additional structure, inducing desirable object, shape, and appearance bias into the learned representation, making the

subsequent semantic classification much easier from sparse supervision. Our hypothesis is that such features encapsulate relevant spatial, object and semantic properties which are hard to learn by purely regressing color and radiance from position. Figure 1 illustrates how extracted features can encode information that can help distinguish between objects in a scene.

To summarize, our contributions are:

- A semantic NeRF algorithm that uses extracted image features to improve segmentation performance
- A hybrid feature encoding that is better suited for the semantic segmentation task
- A volumetric segmentation system, including a graphical user interface, that can be used to quickly generate dense segmentation masks from sparse pixel annotations

In experiments we validate our pipeline on a diverse set of real-world scenes. We perform a larger scale evaluation on scenes from the Replica [114] dataset, which contain many more objects and semantic classes. On these datasets, we compare the performance against baseline methods, as well as using different feature extractors, namely Fully Convolutional Neural Networks [73] and DINO [33]. Our results show that our DINO supervised semantic NeRF formulation outperforms previously proposed semantic NeRFs on both accuracy and learning speed across all scenes, and can do with much less human supervision.

We make visual results, data and code available through the accompanying web page[1].

2 RELATED WORK

2.1 *Automated Labeling*

Our goal is to infer a 3D segmentation of a scene and 2D semantic segmentation maps as quickly as possible. Several works have tackled a similar problem of inferring scene properties in an automated manner.

[1] keke.dev/baking-in-the-feature

Object-based methods use knowledge of object geometry or category to speed up the workflow of annotating scenes. LabelFusion [77] introduced a tool to align an object model with a 3D scene with ICP refinement. While differentiable rendering was used to register the shape and pose of objects using shape priors in [153] and [10]. EasyLabel [118] introduced a semi-automatic method for obtaining instance segmentations of 3D scenes by incrementally building up the scene. RapidPoseLabels [112] presented a way to compute object pose and segmentation masks from sparse 2D keypoints, combining several pointclouds of an object.

Other approaches have explored speeding up RGB-D data annotation by leveraging structure in the data. SALT [116] introduced a GrabCut [97] based approach to speed up labeling of RGB-D data. DeepExtremeCut [75] computes dense object segmentation masks from extreme points of an object in an image.

2.2 *Neural Radiance Fields*

Mildenhall et al. [79] introduced NeRFs for novel view synthesis using volumetric rendering and have been extended in various ways. DS-NeRF [29] used depth measurements to speed up training, which we also levarage. SemanticNeRF [161] extended NeRF to also infer a semantic field from sparse pixel annotations and was extended with iLabel [162], a system for densely annotating scenes. We extend [161] to leverage representations which capture structure present in the input images to learn a better segmentation from fewer labels. While NeRF methods operate on single scenes, [65] introduced a representation that learns across multiple scenes. They disregard semantic labels, but their approach could be extended to segmentation. NeRF-Supervision [148] learns view-invariant dense object descriptors, generating image correspondences from a radiance field.

Other semantic fields include PanopticNeRF [35] which learns from noisy 2D predictions and 3D bounding box annotations. Panoptic Neural Field [63] learns several 4D neural fields that can reconstruct and track moving objects. NeSF [134] similarly uses known calibrated images and NeRF to learn a semantic field of a scene. Instead of adding a semantic output to the field, they learn a 3D UNet mapping density to semantics. They learn the semantic field across many scenes, not targeting single scene label propagation.

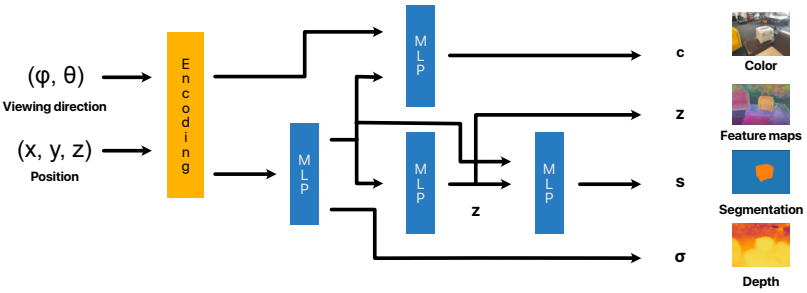


Figure 2: Architectural diagram of our model. Scene coordinates and viewing direction are mapped to RGB, density, depth, feature maps and segmentation maps.

Concurrent work, D3F [60] and N3F, [131] use neural rendering via [130] and feature maps to supervise a NeRF model. N3F tackle one-shot object recognition, also showing some results for object segmentation using a thresholding algorithm, as opposed to propagating sparse labels. In contrast to [60, 130], our method runs at interactive rates, thanks to hybrid feature encoding and lower dimensional, autoencoded, feature rendering.

3 METHOD

We design a NeRF-style algorithm, that maps each point in the scene to color, density, depth, semantic class, and semantic feature vector. Figure 2 shows a high-level diagram of our model, which consists of five main components: a feature encoder, a geometry MLP, a color MLP, a feature MLP, and a semantic classifier MLP. In this section, we first describe how we encode scene positions into feature vectors that are fed to the neural network. We then describe how to use volumetric rendering to compute 2D image maps from the 3D representation and describe how we learn the parameters of the scene representation from provided data. Finally, we describe our graphical user interface, which allows a user to interact with the system.

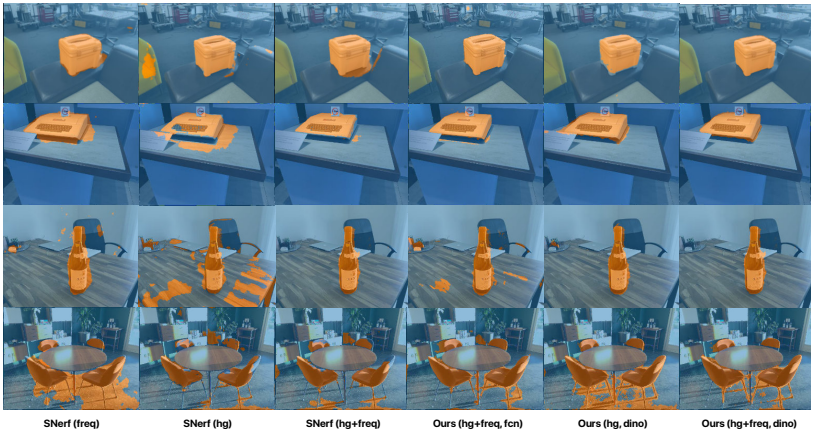


Figure 3: Segmentation masks produced on the box scene using the different methods with the same set of annotations. The frequency encoding (freq) allows for spatial context while the hash grid encoding (hg) allows for a high level of detail locally. The feature supervision again helps in reasoning about object boundaries.

3.1 *Positional Encoding*

NeRF [79] uses a frequency encoding based on sine and cosine functions:

$$\gamma(y) = [\sin 2^0 \pi y, \cos 2^0 \pi y, \ldots, \sin 2^{L-1} \pi y, \cos 2^{L-1} \pi y], \tag{1}$$

where y corresponds to any of the 3D scene coordinates, or a viewing direction, normalized to the range $[-1, 1]$. L defines the number of frequencies used. We use $L = 10$ for 3D coordinates and $L = 4$ for the viewing directions.

An alternative hash grid encoding was introduced in [81], which greatly speeds up the learning of NeRFs. They define a hierarchical voxel grid over the scene coordinates, with parameters at cell vertices for every level. To encode a point, parameters are looked up at each level of the grid, trilinearly interpolated and concatenated to produce an encoding. Parameters are learned jointly with the MLPs.

While the hash grid encoding produces great visual results for view synthesis and greatly speeds up training and rendering, it is not optimal for our application, as simply using the hash grid encoding causes the model

to overfit to the sparse user annotations. The resulting segmentation maps fail to infer the spatial relation between annotated points, and large areas not belonging to objects are assigned the wrong semantic class. See Figure 3 for an illustration.

To solve this problem, we propose a hybrid approach combining both the hash grid encoding with low frequency positional encoding with $L = 2$, which we concatenate together. The low frequency encodings allow the model to reason about coarse spatial location in the volume, while the grid parameters allow encoding finer details in the scene. As we are targeting an interactive system, using only frequency encoding would not be an option, as it requires many more iterations to learn high frequency details.

3.2 *Neural Radiance Fields*

To integrate the scalar and vector outputs of our spatial field, we define a function R to volumetrically render vector or scalar values from a function h along a given ray $\mathbf{r}$:

$$R(\mathbf{r}, h) = \sum_{i=1}^{N} T_i (1 - \exp(-\sigma_i \delta_i)) h(\mathbf{x}_i), \tag{2}$$

$$T_i = \exp(-\sum_{j=1}^{i-1} \sigma_j \delta_j), \tag{3}$$

where T_i is the transmittance function measuring the amount light transmitted through the ray $\mathbf{r}$ to sample i, δ_j is the distance between samples and σ_i is the predicted density for encoded point samples $\mathbf{x}_i$ along the ray $\mathbf{r}$. We use this rendering function to render pixel color values, depth, semantic outputs and image features for a given ray $\mathbf{r}$:

$$\hat{\mathbf{c}}(\mathbf{r}) = R(\mathbf{r}, \mathbf{c}), \tag{4}$$

$$\hat{d}(\mathbf{r}) = R(\mathbf{r}, d), \tag{5}$$

$$\hat{\mathbf{s}}(\mathbf{r}) = R(\mathbf{r}, \mathbf{s}), \tag{6}$$

$$\hat{\mathbf{f}}(\mathbf{r}) = R(\mathbf{r}, \mathbf{f}), \tag{7}$$

using the following functions: $\mathbf{c}$ for color, d for depth, $\mathbf{s}$ for the semantic vector, and $\mathbf{f}$ for the intermediate feature vector of our model for encoded point samples along a ray $\mathbf{r}$.

We define the same photometric loss L_{rgb} as in NeRF [79] and a depth loss L_d similar to the one used by [29]:

$$L_{rgb}(\mathbf{r}) = \|\hat{\mathbf{c}}(\mathbf{r}) - \bar{\mathbf{c}}(\mathbf{r})\|_2^2, \tag{8}$$

$$L_d(\mathbf{r}) = \begin{cases} \left\|\hat{d}(\mathbf{r}) - \bar{d}(\mathbf{r})\right\|_1, & \text{if } \bar{d} \text{ is defined for } \mathbf{r} \\ 0, & \text{otherwise} \end{cases} \tag{9}$$

where $\bar{\mathbf{c}}(\mathbf{r})$ is the ground truth and $\hat{\mathbf{c}}(\mathbf{r})$ the predicted color for ray $\mathbf{r}$, $\bar{d}(\mathbf{r})$ is the ground truth depth (if available) and $\hat{d}(\mathbf{r})$ the integrated depth predictions along ray $\mathbf{r}$.

To learn the semantic class, we define a cross entropy loss:

$$L_s(\mathbf{r}) = \begin{cases} -\log \frac{\exp(\hat{s}(\mathbf{r})_{\bar{s}(\mathbf{r})})}{\sum_{c=1}^{C_n} \exp(\hat{s}(\mathbf{r}))_c)}, & \text{if } \bar{s} \text{ is defined for } \mathbf{r} \\ 0, & \text{otherwise} \end{cases} \tag{10}$$

where $\bar{s}(\mathbf{r})$ is the ground truth class for ray $\mathbf{r}$, $\hat{\mathbf{s}}(\mathbf{r})$ is the integrated semantic MLP outputs and $\hat{s}(\mathbf{r})_c$ the output corresponding to class c.

3.3 *Learning Feature Maps*

We assume that we have a feature extractor, which maps images $\mathbf{I}^{H_i \times W_i \times 3}$ to feature maps $\bar{\mathbf{F}}^{H_f \times W_f \times M}$ containing feature vectors $\bar{\mathbf{f}}$. $\mathbf{I}$ is an input image with height H_i and width W_i, $\bar{\mathbf{F}}$ has height H_f and width W_f and M is the dimensionality of the feature vectors. The purpose of the feature extractor is to encode semantic and spatial information about a particular view, providing contextual information that cannot be inferred from a single scene, but can be learned by observing other data. Such functions include Vision Transformers [33] or Fully Convolutional Neural Networks [73] that come pre-trained a priori on large datasets.

To distill information in the feature maps into our 3D scene representation and to inform a downstream classifier, we propose to volumetrically render feature outputs $\mathbf{f}$ along a ray using (2) to produce rendered feature maps $\hat{\mathbf{f}}$ and supervise on corresponding image features $\bar{\mathbf{f}}$.

The dimensionality of the image features $\bar{\mathbf{f}}$ will vary depending on the chosen pre-trained feature extractor and may be too large to render with reasonable batch sizes on regular GPUs. Therefore, to reduce memory use and the amount of floating point operations when using our system, we use a simple MLP autoencoder to reduce the dimensionality of the image features $\bar{\mathbf{f}}$. This step is completely optional and could be skipped. The autoencoder has an encoder enc and decoder dec. The encoder maps feature vectors with M dimensions to D dimensions and the decoder maps them back to M dimensions. We set D to 64 throughout our experiments. We fit the autoencoder by minimizing a standard reconstruction loss:

$$L_{ae}(\bar{F}) = \left\| dec(enc(\bar{F}(\mathbf{p}))) - \bar{F}(\mathbf{p}) \right\|_2 + \lambda_{ae} \left\| enc(\bar{F}(\mathbf{p}) \right\|_1,$$

where $\bar{F}(\mathbf{p})$ is the feature corresponding to sampled pixel $\mathbf{p}$, and λ_{ae} is a weight for the sparsity term. We use an L_2 loss to minimize information loss. The sparsity term is to encourage a sparse feature representation that can easily be classified into different classes.

As image features only depend on raw input images and a given pre-trained feature extractor, we pre-compute the corresponding autoencoder offline before the user interacts with the system and keep it fixed while the scene representation is learned.

When fitting a scene in our volumetric segmentation pipeline, to bake the feature representation $\hat{\mathbf{f}}$ into the scene representation, we define an additional feature loss:

$$L_f(\mathbf{r}) = \left\| enc(\bar{\mathbf{f}}(\mathbf{r})) - \hat{\mathbf{f}}(\mathbf{r}) \right\|_1, \tag{11}$$

where $\mathbf{r}$ is an image ray, $\bar{\mathbf{f}}(\mathbf{r})$ is the image feature corresponding to ray $\mathbf{r}$, $\hat{\mathbf{f}}(\mathbf{r})$ is the rendered feature for ray $\mathbf{r}$. Should H_i and H_f differ, during training, we use nearest neighbor interpolation to lookup the corresponding image feature.

3.4 *Optimization and Sampling*

We optimize the combined loss using stochastic gradient descent using Adam over rays $\mathbf{r}$ sampled from the images:

$$L(\mathbf{r}) = L_{rgb}(\mathbf{r}) + \lambda_d L_d(\mathbf{r}) + \lambda_s L_s(\mathbf{r}) + \lambda_f L_f(\mathbf{r}), \tag{12}$$

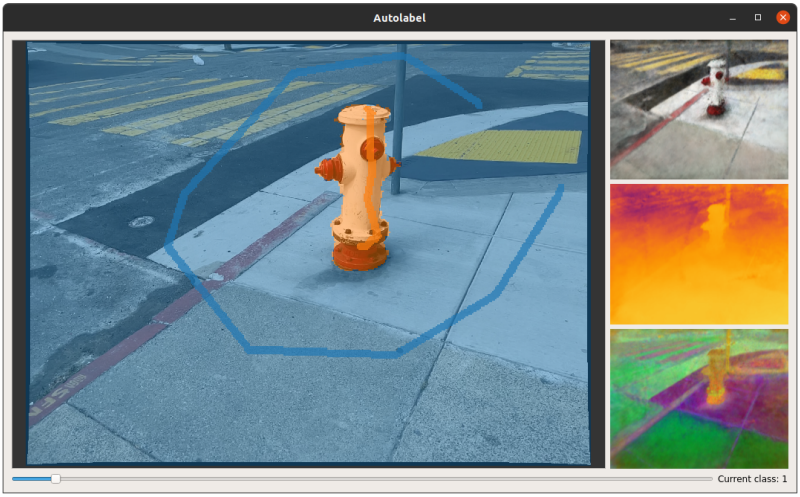


Figure 4: The graphical user interface. The user provides sparse annotations by drawing on images with the appropriate class. The system infers a volumetric segmentation given the current sparse annotations from all views and renders a dense segmentation map which can be corrected with further annotations.

to jointly fit the positional encoding and MLP parameters.

As most pixels do not have a semantic class associated with them, we use a sampling scheme to balance the task of predicting a semantic class with the other objectives. When sampling examples for optimization, we select half the samples uniformly from all pixels. For the remaining half, to not bias the resulting function towards any class, we first select a class uniformly from the available classes. A pixel is then sampled from all pixels which are annotated with the sampled class.

3.5 *Graphical User Interface*

Figure 4 shows the user interface of our system. The user can move through frames in a captured RGB-D video sequence and draw annotations, shown as opaque lines, while the system fits a model to the scene and annotations.

Scene	SNeRF (freq)	SNeRF (hg)	SNerf (hg+freq)	Ours (hg+freq, fcn)	Ours (hg, dino)	Ours (hg+freq, dino)
apple 2	0.744	0.565	0.853	0.903	0.878	**0.942**
bench	0.803	0.817	0.816	0.818	0.873	**0.912**
box	0.901	0.686	0.868	0.951	0.922	**0.962**
chairs	0.531	0.635	0.705	0.686	0.696	**0.761**
cup	0.837	0.514	0.586	0.565	0.854	**0.934**
doughnut	0.653	0.576	0.502	0.673	0.879	**0.910**
fire hydrant 1	0.838	0.561	0.792	0.805	0.751	**0.887**
fire hydrant 2	0.757	0.216	0.853	0.598	0.848	**0.890**
hat	0.911	0.937	**0.960**	0.950	0.919	0.959
keyboard	0.895	0.699	0.900	0.926	0.943	**0.947**
shoe	0.791	0.814	0.833	0.894	0.970	**0.979**
valve	0.588	0.250	0.691	0.721	0.692	**0.730**
wine bottle red	0.523	0.362	0.521	0.794	0.690	**0.856**
wine bottle white	0.569	0.233	0.323	0.573	0.830	**0.884**
Average	0.739	0.560	0.732	0.778	0.703	**0.897**

Table 1: Intersection-over-Union agreement of produced segmentation maps on our captured scenes against manually annotated frames.

The translucent mask shows the current segmentation, which the user can correct. The right side shows rendered color, depth and features mapped to RGB using PCA.

4 EXPERIMENTAL RESULTS

In our experiments we investigate whether our method improves segmentation accuracy when given the same amount of supervision and whether we improve labeling efficiency as part of an interactive labeling system.

4.1 *Baselines*

We compare our algorithm with SemanticNeRF[161]. We also compare the effect of using different positional encodings with both our algorithm and the baseline. For the positional encoding, **freq** refers to the frequency positional encoding in which case we use $L = 10$ frequencies for the position, **hg** refers to the hash grid encoding and **hg+freq** denotes hybrid encoding.

As our algorithm can make use of any features, we experiment with features from a Fully Convolutional Network [73] trained on COCO [68] on the semantic segmentation task, denoted **fcn** and DINO ViT-S/8 vision transformer features [18] trained on ImageNet, denoted **dino**.

To make the comparison fair, all baselines and our method use the same sampling and training pipelines. They simply differ in the scene model and loss functions used.

4.2 *Label Propagation Quality*

To investigate the first question, we scan a number of scenes using an RGB-D camera and run them through [103] to obtain camera poses. We annotate the scenes using the GUI by drawing squiggles on pixels belonging to each desired semantic class on 2 to 10 individual images, depending on the scene. What the annotations look like from a single view can be seen in Figure 4. From these annotations, the algorithms are tasked to segment the scene and infer what the user considers as belonging to each class. All algorithms receive the exact same set of annotations. We run each algorithm on each scene and produce semantic segmentation maps for all images. We compare the produced segmentation maps against ground truth masks, obtained by labeling a reference subset of the images with a polygon mask for each object. We then compute the Intersection-over-Union agreement between the inferred and reference masks.

It should be noted that results on this experiment are not indicative of the performance at the limit on dense segmentation maps, rather they measure the ability of the algorithms to generalize and figure out where object boundaries lie from a specific set of reasonable, sparse annotations.

Table 1 shows IoU agreement between manually labeled individual frames and the segmentations produced by the different methods. The best accuracy is achieved using hybrid encoding, supervised by DINO features on virtually all scenes.

As illustrated in Figure 3 and quantified in Table 1, using only hash grid encoding causes the model to overfit to the sparse annotations. Large areas outside of objects are assigned to the object class. This is especially apparent on the white wine scene (Figure 3, image row 3, column 2). Using hash grid encoding with DINO supervision removes some errors, as the model can reason about visual and contextual properties in the scene. However, object boundaries are not captured as sharply when not using feature supervision. The DINO features perform better than FCN features, indicating that the choice of feature extractor is an important consideration.

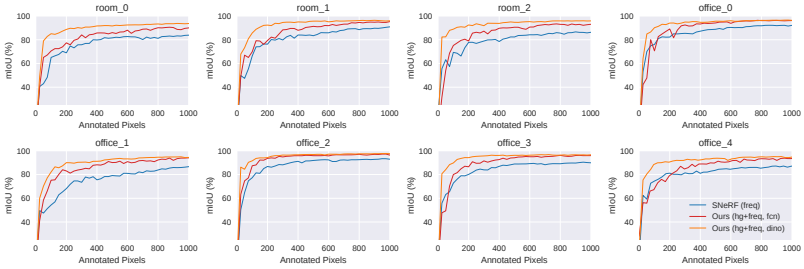


Figure 5: Performance on different scenes of the Replica dataset. We outperform baseline methods on all scenes for both learning speed and final accuracy.

4.3 *Human-in-the-loop Simulation*

As our system is designed to quickly create annotated data for downstream learning algorithms with little human input, we design a simulated human-in-the-loop experiment to test the algorithms in a fair, scalable, and reproducible way. As a dataset, we use the openly available Replica dataset [114] and more specifically, the rendered sequences published by [161], which include dense ground truth semantic annotations. This dataset contains room-scale scenes with multiple objects of multiple classes, presenting a challenging and realistic test environment.

We initialize each algorithm by training only on RGB, depth and features for 15k iterations. We then run an iterative process predicting semantic labels using the current model parameters and select five pixels for which semantic classes are falsely inferred and add their ground truth labels to the training set. We then run 250 optimization steps with all labels accumulated thus far, as would be possible between user actions, and repeat the process.

We record the intersection-over-union agreement with the ground truth segmentation masks at each iteration step to observe how quickly the label propagation algorithm is able to produce high quality semantic labels. The ideal algorithm would converge to perfect labels after one click per object class by the user.

Figure 5 shows results on different scenes of the Replica dataset for the human-in-the-loop simulation. Our method performs better at the limit than the baseline methods on all of the scenes. The results show that the biggest

benefit of feature supervision is that the model learns from much fewer labels, yet still achieves higher accuracy as more pixels are annotated.

The baseline SNeRF (freq) method takes on average over 5x longer to reach 80% IoU accuracy compared to our method with DINO features and hybrid positional encoding. This means a user would spend less than a fifth of the time using the system to achieve the same accuracy.

5 CONCLUSIONS

We presented an algorithm for volumetric segmentation, which we showcased in a human-in-the-loop data annotation and label propagation application. Our algorithm significantly speeds up learning and improves performance over baseline methods across our experiments. We performed an ablation study showing how different parts of our algorithm contribute to the overall performance. We demonstrated how using only hash grid encoding can cause a segmentation model to overfit to sparse user annotations and showed how this problem can be overcome by combining frequency encoding with hash grid encoding.

As shown by our experiments, the choice of feature is important. Learning a better feature representation that is viewpoint invariant, yet allows for efficient segmentation of objects presents a promising research direction. Furthermore, our system assumes a static scene, we intend to extend the work in the future to deal with dynamic scenes.

Our system produces high quality segmentation maps, but in many robotic applications computing other properties, such as object pose [77], shape [37] or keypoints [11] might be required. A framework which could with little input produce high quality segmentations in combination with object information, would be a breakthrough to fuel the learning-based perception algorithms proposed in recent years.

Part D

WEAKLY SUPERVISED SEMANTIC LEARNING FROM THE INTERNET

PAPER IV

NEURAL IMPLICIT VISION-LANGUAGE FEATURE FIELDS

Kenneth Blomqvist, Francesco Milano, Jen Jen Chung, Lionel Ott, Roland Siegwart

ABSTRACT

Recently, groundbreaking results have been presented on open-vocabulary semantic image segmentation. Such methods segment each pixel in an image into arbitrary categories provided at run-time in the form of text prompts, as opposed to a fixed set of classes defined at training time. In this work, we present a zero-shot *volumetric* open-vocabulary semantic scene segmentation method. Our method builds on the insight that we can fuse image features from a vision-language model into a neural implicit representation. We show that the resulting feature field can be segmented into different classes by assigning points to natural language text prompts. The implicit volumetric representation enables us to segment the scene both in 3D and 2D by rendering feature maps from any given viewpoint of the scene. We show that our method works on noisy real-world data and can run in real-time on live sensor data dynamically adjusting to text prompts. We also present quantitative comparisons on the ScanNet dataset.

Published in:
IEEE International Conference on Intelligent Robots and Systems,

1 INTRODUCTION

A key component of building intelligent robots capable of operating in unstructured and cluttered human environments is the representation used to model the robot's surroundings. Often times representations have to trade-off properties which depend on the usage scenario. These properties include the quality of the reconstruction, the ability to integrate sensor data continuously, and the computational complexity to query the representation. The importance of these aspects differs based on what components of a robotic system needs to use the representation, dictating the requirements for available capabilities, sensor data throughput, or query latency. For instance, an obstacle avoidance system needs to query for occupancy at high frequency, while a high-level planning system needs access to semantic knowledge, and finally a grasp planning system requires fine-grained segmentation information.

While in the past occupancy was the main information of interest, robotics has moved towards richer representations using semantics in recent years. A challenge is that most semantic approaches use a fixed, closed set, of pre-determined semantic labels. However, real environments contain more than a few dozen classes, and thus methods capable of handling arbitrary semantic classes, i.e. open set, are desirable. Additionally, objects in an environment do not necessarily belong to distinct, mutually exclusive classes. Certain objects might belong to several classes. A bookshelf is also a piece of furniture, for example. For high-level planning purposes, being able to reason about relations between their semantics might also be useful.

An environment representation that has wide applicability has several desirable properties, including: (1) can be built incrementally as the robot explores the environment, (2) enables real-time integration of new measurements, (3) has a compact memory footprint, (4) represents geometry at a high-level of detail, (5) is differentiable, (6) supports open set semantic queries, and (7) allows fast querying by downstream modules. Previously introduced 3D semantic scene representations are either built from global scene information [87], use closed set semantics [41, 78, 96, 162], operate on a fixed level of detail [41, 96], or are not differentiable [41, 96]. In this paper, we take a step towards a representation which has the above-mentioned properties.

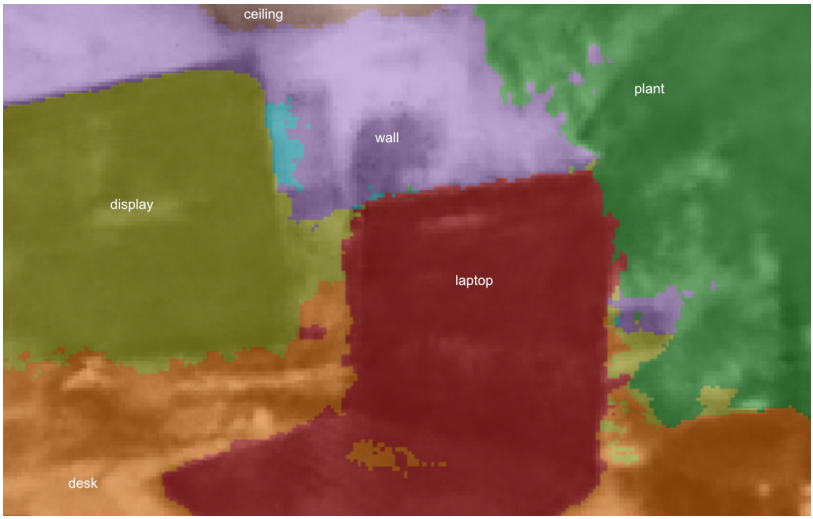


Figure 1: Our method enables real-time segmentation of scenes into arbitrary text classes provided at run-time.

Vision-language models (VLM) have shown remarkable performance on open vocabulary object detection [43, 154]. Recently, these results have been extended to dense semantic segmentation [38, 66, 155, 165]. Some of these methods [38, 66] associate each pixel with a semantically meaningful vector, which is embedded in the same high-dimensional vector space as natural language prompts through a text encoder. This allows direct computation of the similarity between text prompts and image features at run-time.

As vision-language models can be trained on massive web-scale datasets that can be collected automatically without human supervision, they often show better generalization capabilities than models trained on smaller closed-set manually curated datasets. Additionally, VLMs can capture the long tail of scenarios and classes that are so rare that they are unlikely to be included in curated datasets. These properties offer great promise for applications in robotics, where we might want our robots to be able to perform new tasks in never-before-seen environments.

In this paper, we present a method for grounding dense vision-language features into a 3D implicit neural representation that can be built up incrementally, in real-time, as new observations come in. We jointly model radiance, vision-language model features, and density in the scene using an implicit neural representation. Our representation can be incrementally built up given posed images of the scene and a pre-trained language model. We can directly compute the similarity between natural language text prompts and either 3D points or 2D image coordinates for any given viewpoint of the scene through volumetric rendering. This enables semantically segmenting a scene zero-shot into text categories provided at run-time, without having to fine-tune the system on any domain specific semantics.

In experiments, we showcase results in real-world experiments where we build up our scene representation in real-time on a real system, and demonstrate the ability to segment the scene into different classes provided as natural language prompts at run-time. We additionally present quantitative segmentation results on the large and diverse ScanNet dataset. To the best of our knowledge, our method is the first real-time capable 3D vision-language neural implicit representation. Our implementation will be made available through the Autolabel project [1].

2 RELATED WORK

Open Vocabulary Semantic Segmentation and Vision-Language Models

CLIP [90] introduced a visual-language model capable of mapping images into the same vector space as natural language queries by correlating images to their text descriptions mined from the open web. Open vocabulary segmentation methods typically learn dense features which are compared to text queries given at run-time [38, 66]. Others take a multi-task learning approach, fusing a task prompt with the architecture [155, 165]. Other methods such as Clippy [92] explored learning pixel-aligned visual-language models from large scale web datasets without requiring segmentation labels, potentially enabling large-scale open set training, if the results can be extended to full semantic segmentation.

[1] https://github.com/ethz-asl/autolabel

Language Models in Robotics

Large language models have been explored as an approach to high-level planning [3, 20, 51, 91, 113] and scene understanding [21, 45]. Vision-language models embedding image features into the same space as text have been applied to open vocabulary object detection [20, 113], natural language maps [15, 20, 51, 105, 124], and for language-informed navigation [74, 106, 139].

Recent methods have explored fusing global CLIP features [105], image caption embeddings [31], or dense pixel-aligned [87] visual-language model features into a point cloud representation for scene understanding. Concurrent work ConceptFusion [55] explores building multi-modal semantic maps by fusing features from vision-language models as well as audio into a reconstructed 3D point cloud. Similar to these, we also fuse VLM features into a 3D representation. Unlike [55, 87, 105], we use a continuous neural representation of geometry and semantics which we learn jointly through volumetric rendering. [31, 87] fuse image features from a pre-built point cloud using a multi-view fusion method and learn a 3D convolutional network to map scene points to dense features. Our representation can be built incrementally as measurements are collected and does not require global scene geometry upfront.

Semantic Scene Representations

Voxel-based map representations have been proposed to store semantic information about a scene [41, 82, 96, 102, 115]. These methods assign a semantic class to each individual voxel in the scene. Voxel-based dense semantic representations typically operate on static scenes, but some have explored modeling dynamic objects [42, 142].

Scene graphs [6, 52, 140] have also been proposed as a candidate for a semantic scene representation that can be built-up online. Such methods decompose the scene into a graph where edges model relations between parts of the scene. The geometry of the parts are typically represented as a signed distance functions stored in a voxel grid [51].

Neural implicit representations infer scene semantics [13, 35, 63, 72, 78, 110, 161, 162] jointly with geometry using a multi-layer perceptron or similar parametric model. These have been extended to dynamic scenes [61]. Neu-

ral feature fields [13, 60, 78, 131] are neural implicit representations which map continuous 3D coordinates to vector-valued features. Such representations have shown remarkable ability at scene segmentation and editing. [60] also presented some initial results on combining feature fields with vision-language features, motivating their use for language driven semantic segmentation and scene composition.

3 METHOD

Our method consists of two components: i) a NeRF-like feature field mapping points in a volume to color, density, and feature vector and ii) a vision-language model which both extracts features from image frames and can embed text prompts into the same vector space.

3.1 *Volumetric Scene Representation*

We want to associate 3D points in the volume of our scene to density, color, and a feature vector. From this, we can render corresponding maps of color, depth, and feature vectors through a NeRF-like [79] volumetric rendering function, visualized in Figure 2. We model these maps using a positional encoding function and three multilayer perceptrons (MLP). The first MLP, indicated as (1), outputs density and a geometric code. The second MLP, labeled (2), outputs color from the geometric code and an encoded viewing direction. The third MLP, denoted by (3), takes the geometric code and outputs the feature vector.

To encode the x, y, and z position in the volume, we use the hybrid positional encoding introduced by [13]. We concatenate the vector valued hashgrid encoding introduced in [81] with the low-frequency values of traditional NeRF [79] frequency encoding with $L = 2$. The low-frequency components allows us to model the coarse spatial location in the scene, whereas the parameters in the hashgrid grid allow us to quickly learn high-frequency details.

The resulting encoding is fed into an MLP, (1) in Figure 2, which outputs a 15-dimensional geometric code vector and scalar density σ. The geometric code is fed into two different MLPs. The first one outputs a feature vector $\mathbf{f}$. The other one takes as additional input the encoded viewing direction and

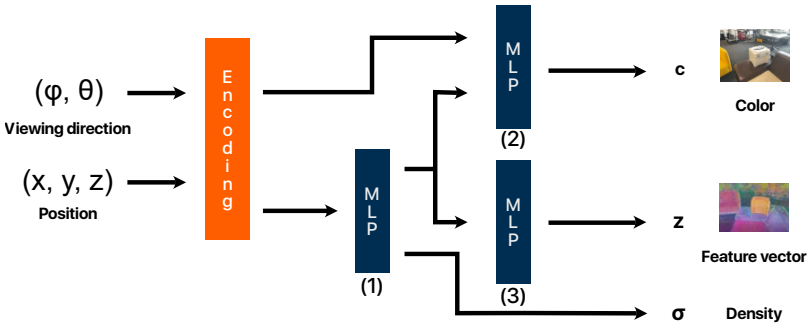


Figure 2: A diagram of the model used for our feature field.

outputs a color vector $\mathbf{c}$. To encode the viewing direction, we use the same spherical harmonic encoding as [79].

We use these outputs to volumetrically render color images and feature outputs using the rendering function:

$$R(\mathbf{r}, h) = \sum_{i=1}^{N} T_i (1 - \exp(-\sigma_i \delta_i)) h(\mathbf{x}_i), \tag{1}$$

$$T_i = \exp\left(- \sum_{j=1}^{i-1} \sigma_j \delta_j \right), \tag{2}$$

where h is a function outputting a vector or scalar quantity for points $\mathbf{x}_i$ within the volume, T_i is the transmittance function, δ_j is the distance between samples and σ_i is the predicted density for encoded point samples $\mathbf{x}_i$ along a ray $\mathbf{r}$. We use R to produce rendered quantities:

$$\begin{aligned} \hat{\mathbf{c}}(\mathbf{r}) &= R(\mathbf{r}, \mathbf{c}), \\ \hat{d}(\mathbf{r}) &= R(\mathbf{r}, z), \\ \hat{\mathbf{f}}(\mathbf{r}) &= R(\mathbf{r}, \mathbf{f}), \end{aligned} \tag{3}$$

using z for the depth component of samples, $\mathbf{c}$ for the color MLP output and $\mathbf{f}$ for the feature vector output of our MLP.

These quantities are learned by optimizing photometric, depth, and feature rendering error terms:

$$\mathcal{L}_{rgb}(\mathbf{r}) = \|\hat{\mathbf{c}}(\mathbf{r}) - \bar{\mathbf{c}}(\mathbf{r})\|_2^2, \tag{4}$$

$$\mathcal{L}_{d}(\mathbf{r}) = \begin{cases} \left\|\hat{d}(\mathbf{r}) - \bar{d}(\mathbf{r})\right\|_1, & \text{if } \bar{d} \text{ is defined for } \mathbf{r} \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

$$\mathcal{L}_{f}(\mathbf{r}) = \left\|\hat{\mathbf{f}}(\mathbf{r}) - \bar{\mathbf{f}}(\mathbf{r})\right\|_2^2 / D \tag{6}$$

where $\bar{\mathbf{c}}(\mathbf{r})$ is the ground truth and $\hat{\mathbf{c}}(\mathbf{r})$ the predicted color for ray $\mathbf{r}$, $\bar{d}(\mathbf{r})$ is the ground truth depth (if available), $\hat{d}(\mathbf{r})$ the predicted depth predictions along ray $\mathbf{r}$, $\hat{\mathbf{f}}$ rendered feature outputs, $\bar{\mathbf{f}}$ extracted image features for ray $\mathbf{r}$, and D the dimensionality of the image features.

The parameters in the hashgrid encoding volume and in the MLPs are jointly learned by optimizing the objective:

$$\mathcal{L}(\mathbf{r}) = \mathcal{L}_{rgb}(\mathbf{r}) + \lambda_d \mathcal{L}_d(\mathbf{r}) + \lambda_f \mathcal{L}_f(\mathbf{r}) \tag{7}$$

using stochastic gradient descent on a set of rays sampled uniformly from input images $\mathbf{I}$ along with corresponding feature vectors $\bar{\mathbf{f}}$. The parameters λ_d and λ_f are weighting parameters to weight the different components of the loss function. To learn the representation online, while our robot is exploring the environment, keyframes with image features can be added to the image set as they are captured.

3.2 *Vision-language Features and Zero-shot Segmentation*

Our framework presented above is capable of making use of arbitrary feature maps. Thus, we can use features from any feature extractor that produces dense pixel-aligned feature maps from images. To enable open set semantic queries in both 2D and 3D at run-time, we choose to use learned features for which the similarity with text prompts can be computed through a simple dot product. LSeg [66] and OpenSeg [38] are both suitable candidates for this purpose. In our experiments, we use LSeg features, as pretrained models are readily available. The model comes both with an image feature extractor $\bar{\mathbf{F}}$ and text encoder $\mathbf{E}$.

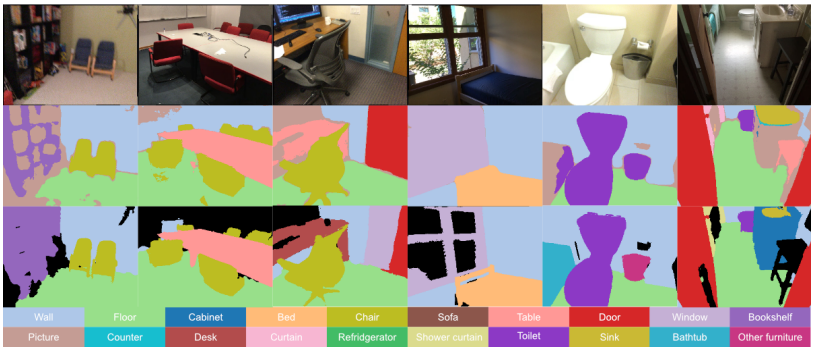


Figure 3: Randomly sampled 2D segmentation examples from the ScanNet validation set. Top row shows the original RGB images, second row shows our segmentation and the bottom row shows the ground truth segmentation from the ScanNet dataset. Black pixels in the ground truth segmentation correspond to classes not included in the 20 ScanNet evaluation classes.

Given a pose in the world frame of the volume, we can render color, depth, and feature maps using volumetric rendering, using equations 1 and 3. We compute the semantic class by assigning the feature $\hat{\mathbf{f}}$ to the most similar class given a set of user defined natural language class descriptions $t_i \in \mathcal{T}$ into which we want to segment our scene:

$$\hat{s}(\mathbf{r}) = \text{argmax}_i \mathbf{E}(t_i) \cdot \hat{\mathbf{f}}(\mathbf{r}). \tag{8}$$

For 3D queries at point $\mathbf{x}$, we can simply evaluate the feature MLP at $\mathbf{x}$, i.e.:

$$s(\mathbf{x}) = \text{argmax}_i \mathbf{E}(t_i) \cdot \mathbf{f}(\mathbf{x}). \tag{9}$$

4 EXPERIMENT RESULTS

In the following experiments we provide quantitative results on the ScanNet dataset. We compare our method to the OpenScene [87] work in terms of mean intersection over union (mIoU) and mean accuracy (mAcc). Then, to highlight the utility of our approach in robotics application we integrate our approach with a SLAM framework. Finally, we report run-time information

to demonstrate the feasibility of running our algorithm on a real robotic system.

In all our experiments, we use LSeg features [66] trained on the ADE20k dataset [163]. For the loss function, we use $\lambda_d = 0.1$ and $\lambda_f = 0.5$ throughout all experiments. Having tried a range of different values, we found that they perform similarly and settled on these values in the middle of the range. In case less noisy and more accurate depth measurements are available, a higher λ_d value might yield better results.

4.1 *ScanNet*

On the ScanNet dataset we perform evaluation both in 3D, by segmenting the provided ground truth point cloud, as well as in 2D by comparing our rendered segmentation maps to the ones provided in the dataset. We use the 20 classes from the ScanNet benchmark. Points or pixels that do not belong to these classes are ignored.

We first fit our representation using the given RGB, depth frames and camera poses using 20 000 optimization iterations. For 3D point cloud segmentation, we look up the feature vector for each point in the point cloud and assign it to the nearest text class using the ScanNet class label names as the text prompts. For 2D segmentation, we segment feature maps from each viewpoint in each scan and compare against the reference segmentation map.

	ScanNet mIoU	ScanNet mAcc
OpenScene - LSeg (3D)	54.2	66.6
OpenScene - OpenSeg (3D)	47.5	70.7
Ours - LSeg (3D)	47.4	55.8
Ours - LSeg (2D)	62.5	80.2

Table 1: Mean intersection-over-union agreement with the ScanNet validation set.

Table 1 shows mean intersection-over-union (mIoU) results on the ScanNet validation set, averaging over scenes and classes. LSeg[66]/OpenSeg [38] denotes the 2D image features used. 3D denotes segmentation agreement on the given ground truth point cloud whereas 2D shows agreement against the semantic segmentation maps.

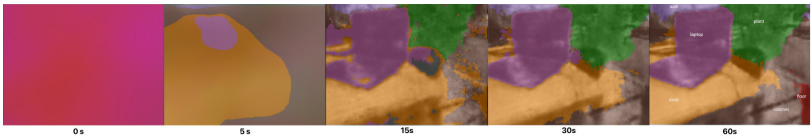


Figure 4: Snapshots from real-time zero shot volumetric segmentations from a fixed viewpoint at given intervals. Our representation is able to learn in real-time and is already useful after a dozen seconds. Each image shows RGB rendering output for the viewpoint, overlayed with the semantic segmentation given the 6 class prompts shown.

OpenScene [87] performs better overall, but it should be noted that it makes use of the ground truth scene point cloud, whereas we only use the color and depth frames and implicitly reconstruct the geometry. We only use the scene point cloud for evaluation. We additionally show 2D segmentation results compared with the ground truth segmentation frames in the dataset. As OpenScene only segments the point cloud, only 3D segmentation accuracy is shown.

Figure 3 shows qualitative 2D segmentation masks. Our method mostly performs well, but often struggles to distinguish between semantically similar classes such as “desk" and “table" or “curtain" and “shower curtain" in the ScanNet evaluation, as we do not make use of any tuning to align the semantics of the dataset with the semantics of the vision-language vector space. The ScanNet label quality is also not perfect and our method often gets details correct which are missed by the ScanNet ground-truth labels, such as legs of tables and chairs or other thin structures.

4.2 *Real-time SLAM Experiment*

To test our scene representation in a real-world robotics scenario, we integrate our system with a SLAM pipeline [2] using a Luxonis OAK-D Pro stereo camera. While the system is running, we integrate color, depth, and features extracted using LSeg from keyframes at 5 Hz with poses obtained from the SLAM system. In experiments, we use either the left (grayscale) camera image or

[2]Specifically the SpectacularAI SDK available here: https://github.com/SpectacularAI/sdk

RGB camera. Depth is computed using stereo matching and aligned to the keyframe camera's frame.

To test our system we give it classes in the form of text prompts while it is running and inspect the quality of the segmentation. Using the odometry poses provided by the SLAM system, we render color, depth maps and segmentation maps from the current camera viewpoint in real-time, segmenting the camera image into the given classes.

Figure 5 shows snapshots of a real-time experiment performed with a handheld camera in a regular office environment. The prompts used to produce the segmentation map are shown, but note that these can be changed at run-time to re-segment the scene. Figure 4 shows how quickly our representation is able to fit to a new scene when learned from scratch and integrating frames in real-time. After a dozen seconds, our method is able to produce good segmentation maps and scene reconstructions.

4.3 *Query Performance*

We time the latency and throughput of queries performed with our implementation on an Nvidia RTX 3070 GPU. 3D semantic and density point queries can be performed at over 7 million lookups per second with a latency of less than 10 milliseconds. 2D ray queries can be rendered and segmented at roughly 30 000 pixels per second using 256 samples per ray, but this can be adjusted to to suit the desired fidelity.

5 DISCUSSION AND CONCLUSIONS

While the results obtained are an encouraging first step towards open-set 3D semantic segmentation there are still many open questions to improve such approaches, some of which we discuss in the following.

Currently, the largest factor limiting segmentation performance is the quality of the vision-language features. While LSeg uses natural language features from CLIP trained on a very large dataset, the visual encoder is trained on the small closed-set ADE20K dataset. If we were able to compute dense pixel-aligned visual-language features from open-set web scraped data without requiring any human annotations, we believe that results could eventually surpass supervised learning methods. [92] presented some promising initial

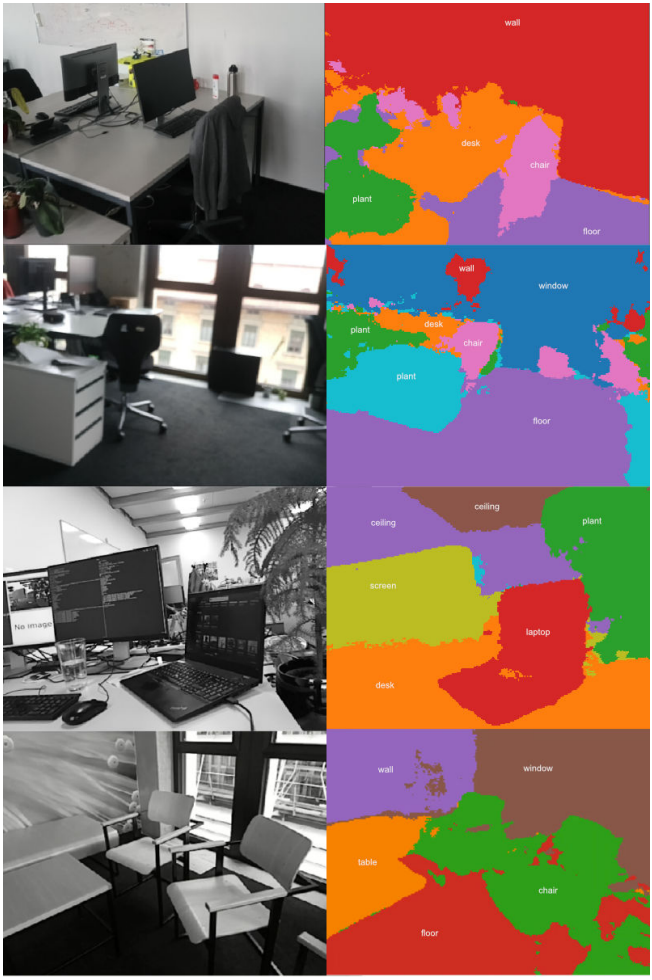


Figure 5: RGB renderings and semantic segmentation maps of our representation from our real-time experiment in an office environment given the prompts shown below the images.

results on learning pixel aligned features without using segmentation masks or other expert annotations.

In real-time experiments, our system relied on poses coming from a SLAM system. If many bad poses are computed by the SLAM system, the 3D representation could become corrupted by bad updates. Possible solutions include treating the sparse SLAM poses as initial guesses and optimizing the poses jointly with scene geometry, as in [117, 164], or bad poses could be filtered out by analyzing the photometric or geometric error across frames.

In robotics, downstream modules, such as motion planners and high-level planning systems, might benefit from a more explicit and principled representation of geometry than what we presented in this paper. For example, signed distance function based approaches [137] might provide better surface and occupancy reconstruction and have other favorable properties, such as the ability to compute the normal of a surface by differentiating through the distance function. For the time being, our method is limited to static scenes. Dealing with moving objects within scenes remains an open problem, but promising recent research [61] suggests that extending neural implicit representations to dynamic scenes might be feasible.

To conclude, we proposed a volumetric neural representation which is able to jointly learn geometry, radiance, and semantic feature information of a scene. We have shown that by using dense pixel-aligned vision-language features, our resulting learned representation can be used to volumetrically segment scenes into, at run-time, user defined categories. We have also shown how the representation can be used to produce dense 2D segmentation maps for queried viewpoints. Experiments on the ScanNet dataset showed competitive performance and our real-world experiments demonstrate that the method could be run onboard a robotic system.

BIBLIOGRAPHY

[1] S. Agarwal, N. Snavely, S. M. Seitz, and R. Szeliski. Bundle adjustment in the large. In *Computer Vision–ECCV 2010: 11th European Conference on Computer Vision, Heraklion, Crete, Greece, September 5-11, 2010, Proceedings, Part II 11*, pages 29–42. Springer, 2010.

[2] A. Ahmadyan, L. Zhang, A. Ablavatski, J. Wei, and M. Grundmann. Objectron: A Large Scale Dataset of Object-Centric Videos in the Wild with Pose Annotations. In *the IEEE/CVF CVPR*, 2021.

[3] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, et al. Do As I Can, Not As I Say: Grounding Language in Robotic Affordances. *arXiv preprint arXiv:2204.01691*, 2022.

[4] A. M. Andrew. Multiple view geometry in computer vision. *Kybernetes*, 2001.

[5] H. A. Arief, M. Arief, G. Zhang, Z. Liu, M. Bhat, U. G. Indahl, H. Tveite, and D. Zhao. SAnE: Smart Annotation and Evaluation Tools for Point Cloud Data. *IEEE Access*, 8:131848–131858, 2020.

[6] I. Armeni, Z.-Y. He, J. Gwak, A. R. Zamir, M. Fischer, J. Malik, and S. Savarese. 3D Scene Graph: A Structure for Unified Semantics, 3D Space, and Camera. In *IEEE/CVF ICCV*, 2019.

[7] D. Azinović, R. Martin-Brualla, D. B. Goldman, M. Nießner, and J. Thies. Neural rgb-d surface reconstruction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2022.

[8] L. Bainbridge. Ironies of automation. In *Analysis, design and evaluation of man–machine systems*, pages 129–135. Elsevier, 1983.

[9] G. Baruch, Z. Chen, A. Dehghan, Y. Feigin, P. Fu, T. Gebauer, D. Kurz, T. Dimry, B. Joffe, A. Schwartz, et al. ARKitScenes: A diverse real-world dataset for 3D indoor scene understanding using mobile rgb-d data. In

Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1), 2021.

[10] D. Beker, H. Kato, M. A. Morariu, T. Ando, T. Matsuoka, W. Kehl, and A. Gaidon. Monocular differentiable rendering for self-supervised 3d object detection. In *ECCV*, pages 514–529. Springer, 2020.

[11] K. Blomqvist, J. J. Chung, L. Ott, and R. Siegwart. Semi-automatic 3D object keypoint annotation and detection for the masses. In *International Conference on Pattern Recognition*, 2022.

[12] K. Blomqvist, F. Milano, J. J. Chung, L. Ott, and R. Siegwart. Neural implicit vision-language feature fields. In *IEEE IROS*, 2023.

[13] K. Blomqvist, L. Ott, J. J. Chung, and R. Siegwart. Baking in the Feature: Accelerating Volumetric Segmentation by Rendering Feature Maps. *IEEE IROS*, 2023.

[14] Bloomberg. Gm's cruise expands robotaxi service to phoenix and austin even with safety probe, 2023. URL: https://www.bloomberg.com/news/articles/2022-12-20/gm-s-cruise-expands-robotaxi-service-amid-federal-safety-probe Accessed: 2023-07-26.

[15] V. Blukis, R. Knepper, and Y. Artzi. Few-shot Object Grounding and Mapping for Natural Language Robot Instruction Following. In *Conference on Robot Learning*, 2021.

[16] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, X. Chen, K. Choromanski, T. Ding, D. Driess, A. Dubey, C. Finn, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023.

[17] B. Calli, A. Singh, A. Walsman, S. Srinivasa, P. Abbeel, and A. M. Dollar. The ycb object and model set: Towards common benchmarks for manipulation research. In *2015 international conference on advanced robotics (ICAR)*, pages 510–517. IEEE, 2015.

[18] M. Caron, H. Touvron, I. Misra, H. Jégou, J. Mairal, P. Bojanowski, and A. Joulin. Emerging properties in self-supervised vision transformers.

In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2021.

[19] 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*, pages 11165–11174, 2023.

[20] B. Chen, F. Xia, B. Ichter, K. Rao, K. Gopalakrishnan, M. Ryoo, A. Stone, and D. Kappler. Open-vocabulary Queryable Scene Representations for Real World Planning. *arxiv preprint arXiv:2209.09874*, 2022.

[21] W. Chen, S. Hu, R. Talak, and L. Carlone. Leveraging Large Language Models for Robot 3D Scene Understanding. *arXiv preprint arXiv:2209.05629*, 2022.

[22] Y. Chen, Q. Yuan, Z. Li, Y. L. W. W. C. Xie, X. Wen, and Q. Yu. Upst-nerf: Universal photorealistic style transfer of neural radiance fields for 3d scene. *arXiv preprint arXiv:2208.07059*, 2022.

[23] A. Cramariuc, L. Bernreiter, F. Tschopp, M. Fehr, V. Reijgwart, J. Nieto, R. Siegwart, and C. Cadena. maplab 2.0–a modular and multi-modal mapping framework. *IEEE Robotics and Automation Letters*, 8(2):520–527, 2022.

[24] A. Cui, A. Jahanian, A. Lapedriza, A. Torralba, S. Mahdizadehaghdam, R. Kumar, and D. Bau. Local relighting of real scenes. *arXiv preprint arXiv:2207.02774*, 2022.

[25] B. Curless and M. Levoy. A Volumetric Method for building Complex Models from Range Images. In *Computer Graphics and Interactive Techniques*, 1996.

[26] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner. ScanNet: Richly-annotated 3D Reconstructions of Indoor Scenes. In *IEEE/CVF CVPR*, 2017.

[27] M. Danielczuk, M. Matl, S. Gupta, A. Li, A. Lee, J. Mahler, and K. Goldberg. Segmenting unknown 3D objects from real depth images using mask r-cnn trained on synthetic point clouds. In *IEEE ICRA*, 2019.

[28] B. Deng, J. T. Barron, and P. P. Srinivasan. JaxNeRF: an efficient JAX implementation of NeRF, 2020. URL `https://github.com/google-research/google-research/tree/master/jaxnerf`.

[29] K. Deng, A. Liu, J.-Y. Zhu, and D. Ramanan. Depth-supervised nerf: Fewer views and faster training for free. In *IEEE/CVF CVPR*, pages 12882–12891, 2022.

[30] M. Devecka. Did the greeks believe in their robots? *The Cambridge Classical Journal*, 59:52–69, 2013.

[31] R. Ding, J. Yang, C. Xue, W. Zhang, S. Bai, and X. Qi. Language-driven Open-Vocabulary 3D Scene Understanding. *arXiv preprint arXiv:2211.16312*, 2022.

[32] Y. Ding, L. Liu, C. Tian, J. Yang, and H. Ding. Don't stop learning: Towards continual learning for the clip model. *arXiv preprint arXiv:2207.09248*, 2022.

[33] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[34] B. Dynamics. Leaps, Bounds and Backflips, 2022. https://bostondynamics.com/blog/leaps-bounds-and-backflips/ accessed 2023-07-26.

[35] X. Fu, S. Zhang, T. Chen, Y. Lu, L. Zhu, X. Zhou, A. Geiger, and Y. Liao. Panoptic NeRF: 3D-to-2D label transfer for panoptic urban scene segmentation. *arXiv preprint arXiv:2203.15224*, 2022.

[36] F. Furrer, M. Fehr, T. Novkovic, H. Sommer, I. Gilitschenski, and R. Siegwart. Evaluation of combined time-offset estimation and hand-eye calibration on robotic datasets. In *Field and Service Robotics: Results of the 11th International Conference*, pages 145–159. Springer, 2018.

[37] W. Gao and R. Tedrake. kpam-sc: Generalizable manipulation planning using keypoint affordance and shape completion. In *2021 IEEE ICRA*, pages 6527–6533. IEEE, 2021.

[38] G. Ghiasi, X. Gu, Y. Cui, and T.-Y. Lin. Scaling Open-Vocabulary Image Segmentation with Image-Level Labels. In *ECCV*, 2022.

[39] K. Grauman, A. Westbury, E. Byrne, Z. Chavis, A. Furnari, R. Girdhar, J. Hamburger, H. Jiang, M. Liu, X. Liu, et al. Ego4d: Around the world in 3,000 hours of egocentric video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18995–19012, 2022.

[40] T. Grenzdörffer, M. Günther, and J. Hertzberg. Ycb-m: A multi-camera rgb-d dataset for object recognition and 6dof pose estimation. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3650–3656. IEEE, 2020.

[41] M. Grinvald, F. Furrer, T. Novkovic, J. J. Chung, C. Cadena, R. Siegwart, and J. Nieto. Volumetric Instance-Aware Semantic Mapping and 3D Object Discovery. *IEEE Robotics and Automation Letters*, 2019.

[42] M. Grinvald, F. Tombari, R. Siegwart, and J. Nieto. TSDF++: A Multi-Object Formulation for Dynamic Object Tracking and Reconstruction. In *IEEE ICRA*, 2021.

[43] X. Gu, T.-Y. Lin, W. Kuo, and Y. Cui. Open-vocabulary Object Detection via Vision and Language Knowledge Distillation. *arXiv preprint arXiv:2104.13921*, 2021.

[44] S. Gunasekar, Y. Zhang, J. Aneja, C. C. T. Mendes, A. Del Giorno, S. Gopi, M. Javaheripi, P. Kauffmann, G. de Rosa, O. Saarikivi, et al. Textbooks are all you need. *arXiv preprint arXiv:2306.11644*, 2023.

[45] H. Ha and S. Song. Semantic Abstraction: Open-World 3D Scene Understanding from 2D Vision-Language Models. In *Conference on Robot Learning*, 2022.

[46] A. Haque, M. Tancik, A. A. Efros, A. Holynski, and A. Kanazawa. Instruct-nerf2nerf: Editing 3d scenes with instructions. *arXiv preprint arXiv:2303.12789*, 2023.

[47] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[48] Y. He, W. Sun, H. Huang, J. Liu, H. Fan, and J. Sun. Pvn3d: A deep point-wise 3d keypoints voting network for 6dof pose estimation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11632–11641, 2020.

[49] J. Hoffmann, S. Borgeaud, A. Mensch, E. Buchatskaya, T. Cai, E. Rutherford, D. d. L. Casas, L. A. Hendricks, J. Welbl, A. Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

[50] Y. Hu, J. Hugonot, P. Fua, and M. Salzmann. Segmentation-driven 6D object pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3385–3394, 2019.

[51] C. Huang, O. Mees, A. Zeng, and W. Burgard. Visual Language Maps for Robot Navigation. *arXiv preprint arXiv:2210.05714*, 2022.

[52] N. Hughes, Y. Chang, and L. Carlone. Hydra: A Real-time Spatial Perception System for 3D Scene Graph Construction and Optimization. In *Robotics: Science and Systems*, 2022.

[53] A. Jaiswal, A. R. Babu, M. Z. Zadeh, D. Banerjee, and F. Makedon. A survey on contrastive self-supervised learning. *Technologies*, 9(1):2, 2020.

[54] C. Jambon, B. Kerbl, G. Kopanas, S. Diolatzis, G. Drettakis, and T. Leimkühler. Nerfshop: Interactive editing of neural radiance fields. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 6 (1), 2023.

[55] K. M. Jatavallabhula, A. Kuwajerwala, Q. Gu, M. Omama, T. Chen, S. Li, G. Iyer, S. Saryazdi, N. Keetha, A. Tewari, et al. ConceptFusion: Open-set Multimodal 3D Mapping. *arXiv preprint arXiv:2302.07241*, 2023.

[56] L. Jing and Y. Tian. Self-supervised visual feature learning with deep neural networks: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 43(11):4037–4058, 2020.

[57] B. Katz, J. Di Carlo, and S. Kim. Mini cheetah: A platform for pushing the limits of dynamic quadruped control. In *2019 international conference on robotics and automation (ICRA)*, pages 6295–6301. IEEE, 2019.

[58] J. Kerr, C. M. Kim, K. Goldberg, A. Kanazawa, and M. Tancik. Lerf: Language embedded radiance fields. *arXiv preprint arXiv:2303.09553*, 2023.

[59] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, et al. Segment anything. *arXiv preprint arXiv:2304.02643*, 2023.

[60] S. Kobayashi, E. Matsumoto, and V. Sitzmann. Decomposing NeRF for Editing via Feature Field Distillation. In *NeurIPS*, 2022.

[61] X. Kong, S. Liu, M. Taher, and A. Davison. vMAP: Vectorised Object Mapping for Neural Field SLAM. *arxiv preprint arXiv:2302.01838*, 2023.

[62] Z. Kuang, F. Luan, S. Bi, Z. Shu, G. Wetzstein, and K. Sunkavalli. Palettenerf: Palette-based appearance editing of neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20691–20700, 2023.

[63] A. Kundu, K. Genova, X. Yin, A. Fathi, C. Pantofaru, L. J. Guibas, A. Tagliasacchi, F. Dellaert, and T. Funkhouser. Panoptic Neural Fields: A Semantic Object-Aware Neural Scene Representation. In *IEEE/CVF CVPR*, 2022.

[64] H. Law, Y. Teng, O. Russakovsky, and J. Deng. CornerNet-Lite: Efficient keypoint based object detection. *arXiv preprint arXiv:1904.08900*, 2019.

[65] V. Lazova, V. Guzov, K. Olszewski, S. Tulyakov, and G. Pons-Moll. Control-nerf: Editable feature volumes for scene rendering and manipulation. *arXiv preprint arXiv:2204.10850*, 2022.

[66] B. Li, K. Q. Weinberger, S. Belongie, V. Koltun, and R. Ranftl. Language-driven Semantic Segmentation. *arXiv preprint arXiv:2201.03546*, 2022.

[67] Z. Li, T. Müller, A. Evans, R. H. Taylor, M. Unberath, M.-Y. Liu, and C.-H. Lin. Neuralangelo: High-fidelity neural surface reconstruction.

In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8456–8465, 2023.

[68] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *ECCV*, pages 740–755. Springer, 2014.

[69] Y. Lin, J. Tremblay, S. Tyree, P. A. Vela, and S. Birchfield. Single-stage keypoint-based category-level object pose estimation from an rgb image. In *IEEE ICRA*, 2022.

[70] Y. Lin, M. Chen, W. Wang, B. Wu, K. Li, B. Lin, H. Liu, and X. He. Clip is also an efficient segmenter: A text-driven approach for weakly supervised semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15305–15314, 2023.

[71] X. Liu, R. Jonschkowski, A. Angelova, and K. Konolige. Keypose: Multi-view 3D labeling and keypoint estimation for transparent objects. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11602–11610, 2020.

[72] Z. Liu, F. Milano, J. Frey, M. Hutter, R. Siegwart, H. Blum, and C. Cadena. Unsupervised Continual Semantic Adaptation through Neural Rendering. *arXiv preprint arXiv:2211.13969*, 2022.

[73] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[74] A. Majumdar, G. Aggarwal, B. Devnani, J. Hoffman, and D. Batra. ZSON: Zero-Shot Object-Goal Navigation using Multimodal Goal Embeddings. *arXiv preprint arXiv:2206.12403*, 2022.

[75] K.-K. Maninis, S. Caelles, J. Pont-Tuset, and L. Van Gool. Deep Extreme Cut: From Extreme Points to Object Segmentation. In *IEEE/CVF CVPR*, pages 616–625, 2018.

[76] L. Manuelli, W. Gao, P. Florence, and R. Tedrake. kpam: Keypoint affordances for category-level robotic manipulation. In *The International Symposium of Robotics Research*, pages 132–157. Springer, 2019.

[77] P. Marion, P. R. Florence, L. Manuelli, and R. Tedrake. Label Fusion: A Pipeline for Generating Ground Truth Labels for Real RGBD Data of Cluttered Scenes. In *IEEE ICRA*, 2018.

[78] K. Mazur, E. Sucar, and A. J. Davison. Feature-Realistic Neural Fusion for Real-Time, Open Set Scene Understanding. *IEEE ICRA*, 2023.

[79] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. In *ECCV*, 2020.

[80] A. Moreau, N. Piasco, D. Tsishkou, B. Stanciulescu, and A. de La Fortelle. Lens: Localization enhanced by nerf synthesis. In *Conference on Robot Learning*, pages 1347–1356. PMLR, 2022.

[81] T. Müller, A. Evans, C. Schied, and A. Keller. Instant Neural Graphics Primitives with a Multiresolution Hash Encoding. *arXiv preprint arXiv:2201.05989*, 2022.

[82] G. Narita, T. Seno, T. Ishikawa, and Y. Kaji. PanopticFusion: Online Volumetric Semantic Mapping at the Level of Stuff and Things. In *IROS*, 2019.

[83] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon. KinectFusion: Real-Time Dense Surface Mapping and Tracking. In *IEEE International Symposium on Mixed and Augmented Reality*, 2011.

[84] L. Oth, P. Furgale, L. Kneip, and R. Siegwart. Rolling shutter camera calibration. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1360–1367, 2013.

[85] K. Park, T. Patten, and M. Vincze. Pix2Pose: Pixel-wise coordinate regression of objects for 6D pose estimation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 7668–7677, 2019.

[86] S. Peng, Y. Liu, Q. Huang, X. Zhou, and H. Bao. PVNet: Pixel-wise voting network for 6DoF pose estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4561–4570, 2019.

[87] S. Peng, K. Genova, C. Jiang, A. Tagliasacchi, M. Pollefeys, T. Funkhouser, et al. OpenScene: 3D Scene Understanding with Open Vocabularies. *arXiv preprint arXiv:2211.15654*, 2022.

[88] J. Qin, J. Wu, P. Yan, M. Li, R. Yuxi, X. Xiao, Y. Wang, R. Wang, S. Wen, X. Pan, et al. Freeseg: Unified, universal and open-vocabulary image segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19446–19455, 2023.

[89] F. Radenovic, A. Dubey, A. Kadian, T. Mihaylov, S. Vandenhende, Y. Patel, Y. Wen, V. Ramanathan, and D. Mahajan. Filtering, distillation, and hard negatives for vision-language pre-training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6967–6977, 2023.

[90] 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*, 2021.

[91] S. S. Raman, V. Cohen, E. Rosen, I. Idrees, D. Paulius, and S. Tellex. Planning with Large Language Models via Corrective Re-prompting. *arXiv preprint arXiv:2211.09935*, 2022.

[92] 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.

[93] J. Redmon and A. Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[94] K. Rematas, A. Liu, P. P. Srinivasan, J. T. Barron, A. Tagliasacchi, T. Funkhouser, and V. Ferrari. Urban radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12932–12942, 2022.

[95] P. Ren, C. Li, H. Xu, Y. Zhu, G. Wang, J. Liu, X. Chang, and X. Liang. Viewco: Discovering text-supervised segmentation masks via multi-view semantic consistency. *arXiv preprint arXiv:2302.10307*, 2023.

[96] A. Rosinol, M. Abate, Y. Chang, and L. Carlone. Kimera: an Open-Source Library for Real-Time Metric-Semantic Localization and Mapping. In *IEEE ICRA*, 2020.

[97] C. Rother, V. Kolmogorov, and A. Blake. "GrabCut" Interactive Foreground Extraction using Iterated Graph Cuts. *ACM Transactions on Graphics*, 2004.

[98] V. Rudnev, M. Elgharib, W. Smith, L. Liu, V. Golyanik, and C. Theobalt. Nerf for outdoor scene relighting. In *European Conference on Computer Vision*, pages 615–631. Springer, 2022.

[99] B. C. Russell, A. Torralba, K. P. Murphy, and W. T. Freeman. LabelMe: a database and web-based tool for image annotation. *International Journal of Computer Vision*, 2008.

[100] P.-E. Sarlin, C. Cadena, R. Siegwart, and M. Dymczyk. From coarse to fine: Robust hierarchical localization at large scale. In *CVPR*, 2019.

[101] P.-E. Sarlin, D. DeTone, T. Malisiewicz, and A. Rabinovich. SuperGlue: Learning feature matching with graph neural networks. In *CVPR*, 2020.

[102] L. Schmid, J. Delmerico, J. L. Schönberger, J. Nieto, M. Pollefeys, R. Siegwart, and C. Cadena. Panoptic Multi-TSDFs: a Flexible Representation for Online Multi-resolution Volumetric Mapping and Long-term Dynamic Scene Consistency. In *IEEE ICRA*, 2022.

[103] J. L. Schonberger and J.-M. Frahm. Structure-From-Motion Revisited. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4104–4113, 2016.

[104] B. Settles. From Theories to Queries: Active Learning in Practice. In *Active Learning and Experimental Design workshop In conjunction with AISTATS 2010*, pages 1–18. JMLR Workshop and Conference Proceedings, 2011.

[105] N. M. M. Shafiullah, C. Paxton, L. Pinto, S. Chintala, and A. Szlam. CLIP-Fields: Weakly Supervised Semantic Fields for Robotic Memory. *arXiv preprint arXiv:2210.05663*, 2022.

[106] D. Shah, B. Osinski, B. Ichter, and S. Levine. LM-Nav: Robotic Navigation with Large Pre-Trained Models of Language, Vision, and Action. *arXiv preprint arXiv:2207.04429*, 2022.

[107] G. Shin, W. Xie, and S. Albanie. All you need are a few pixels: semantic segmentation with pixelpick. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1687–1697, 2021.

[108] M. Shridhar, L. Manuelli, and D. Fox. Perceiver-actor: A multi-task transformer for robotic manipulation. In *Conference on Robot Learning*, pages 785–799. PMLR, 2023.

[109] Y. Siddiqui, J. Valentin, and M. Nießner. ViewAL: Active Learning with Viewpoint Entropy for Semantic Segmentation. In *IEEE/CVF CVPR*, pages 9433–9443, 2020.

[110] Y. Siddiqui, L. Porzi, S. R. Buló, N. Müller, M. Nießner, A. Dai, and P. Kontschieder. Panoptic Lifting for 3D Scene Understanding with Neural Fields. *arXiv preprint arXiv:2212.09802*, 2022.

[111] T. Simon, H. Joo, I. Matthews, and Y. Sheikh. Hand keypoint detection in single images using multiview bootstrapping. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 1145–1153, 2017.

[112] R. P. Singh, M. Benallegue, Y. Yoshiyasu, and F. Kanehiro. Rapid pose label generation through sparse representation of unknown objects. In *2021 IEEE ICRA*, pages 10287–10293. IEEE, 2021.

[113] C. H. Song, J. Wu, C. Washington, B. M. Sadler, W.-L. Chao, and Y. Su. LLM-Planner: Few-Shot Grounded Planning for Embodied Agents with Large Language Models. *arXiv preprint arXiv:2212.04088*, 2022.

[114] J. Straub, T. Whelan, L. Ma, Y. Chen, E. Wijmans, S. Green, J. J. Engel, R. Mur-Artal, C. Ren, S. Verma, et al. The replica dataset: A digital replica of indoor spaces. *arXiv preprint arXiv:1906.05797*, 2019.

[115] M. Strecke and J. Stuckler. EM-Fusion: Dynamic Object-Level SLAM With Probabilistic Data Association. In *IEEE/CVF ICCV*, 2019.

[116] D. Stumpf, S. Krauß, G. Reis, O. Wasenmüller, and D. Stricker. SALT: A Semi-automatic Labeling Tool for RGB-D Video Sequences. *arXiv preprint arXiv:2102.10820*, 2021.

[117] E. Sucar, S. Liu, J. Ortiz, and A. J. Davison. iMAP: Implicit Mapping and Positioning in Real-Time. In *IEEE/CVF ICCV*, 2021.

[118] M. Suchi, T. Patten, D. Fischinger, and M. Vincze. EasyLabel: A Semi-Automatic Pixel-wise Object Annotation Tool for Creating Robotic RGB-D Datasets. In *ICRA*, 2019.

[119] C. Sun, A. Shrivastava, S. Singh, and A. Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *Proceedings of the IEEE international conference on computer vision*, pages 843–852, 2017.

[120] X. Sun, B. Xiao, F. Wei, S. Liang, and Y. Wei. Integral human pose regression. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 529–545, 2018.

[121] N. Sünderhauf, O. Brock, W. Scheirer, R. Hadsell, D. Fox, J. Leitner, B. Upcroft, P. Abbeel, W. Burgard, M. Milford, et al. The Limits and Potentials of Deep Learning for Robotics. *The International Journal of Robotics Research*, 2018.

[122] R. Sutton. The bitter lesson, 2019. URL: http://www.incompleteideas.net/IncIdeas/BitterLesson.html Accessed: 2023-08-16.

[123] S. Suwajanakorn, N. Snavely, J. J. Tompson, and M. Norouzi. Discovery of latent 3D keypoints via end-to-end geometric reasoning. In *Advances in Neural Information Processing Systems*, pages 2059–2070, 2018.

[124] S. Tan, M. Ge, D. Guo, H. Liu, and F. Sun. Self-supervised 3D Semantic Representation Learning for Vision-and-Language Navigation. *arXiv preprint arXiv:2201.10788*, 2022.

[125] M. Tancik, V. Casser, X. Yan, S. Pradhan, B. Mildenhall, P. P. Srinivasan, J. T. Barron, and H. Kretzschmar. Block-nerf: Scalable large scene neural view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8248–8258, 2022.

[126] B. Tekin, S. N. Sinha, and P. Fua. Real-time seamless single shot 6d object pose prediction. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 292–301, 2018.

[127] F. Tosi, A. Tonioni, D. De Gregorio, and M. Poggi. Nerf-supervised deep stereo. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[128] J. Tremblay, T. To, and S. Birchfield. Falling things: A synthetic dataset for 3D object detection and pose estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 2038–2041, 2018.

[129] J. Tremblay, T. To, B. Sundaralingam, Y. Xiang, D. Fox, and S. Birchfield. Deep object pose estimation for semantic robotic grasping of household objects. *arXiv preprint arXiv:1809.10790*, 2018.

[130] V. Tschernezki, D. Larlus, and A. Vedaldi. Neuraldiff: Segmenting 3d objects that move in egocentric videos. In *2021 International Conference on 3D Vision (3DV)*, pages 910–919. IEEE, 2021.

[131] V. Tschernezki, I. L. D. Larlus, and A. Vedaldi. Neural Feature Fusion Fields: 3D Distillation of Self-Supervised 2D Image Representations. In *Conference on 3D Vision*, 2022.

[132] F. Tschopp, M. Riner, M. Fehr, L. Bernreiter, F. Furrer, T. Novkovic, A. Pfrunder, C. Cadena, R. Siegwart, and J. Nieto. VersaVIS—an open versatile multi-camera visual-inertial sensor suite. *Sensors*, 20(5):1439, 2020.

[133] M. Vecerik, J.-B. Regli, O. Sushkov, D. Barker, R. Pevceviciute, T. Rothörl, R. Hadsell, L. Agapito, and J. Scholz. S3k: Self-supervised semantic keypoints for robotic manipulation via multi-view consistency. In *Conference on Robot Learning*, pages 449–460. PMLR, 2021.

[134] S. Vora, N. Radwan, K. Greff, H. Meyer, K. Genova, M. S. M. Sajjadi, E. Pot, A. Tagliasacchi, and D. Duckworth. NeSF: Neural semantic fields for generalizable semantic segmentation of 3d scenes. *Transactions on Machine Learning Research*, 2022. URL `https://openreview.net/forum?id=ggPhsYCsm9`.

[135] C. Wang, R. Martín-Martín, D. Xu, J. Lv, C. Lu, L. Fei-Fei, S. Savarese, and Y. Zhu. 6-PACK: Category-level 6D pose tracker with anchor-based keypoints. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10059–10066. IEEE, 2020.

[136] H. Wang, S. Sridhar, J. Huang, J. Valentin, S. Song, and L. J. Guibas. Normalized object coordinate space for category-level 6D object pose and size estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2642–2651, 2019.

[137] P. Wang, L. Liu, Y. Liu, C. Theobalt, T. Komura, and W. Wang. NeuS: Learning Neural Implicit Surfaces by Volume Rendering for Multi-view Reconstruction. *arXiv preprint arXiv:2106.10689*, 2021.

[138] Y. Wang, Q. Han, M. Habermann, K. Daniilidis, C. Theobalt, and L. Liu. Neus2: Fast learning of neural implicit surfaces for multi-view reconstruction. *arXiv preprint arXiv:2212.05231*, 2022.

[139] Z. Wang, M. Li, M. Wu, M.-F. Moens, and T. Tuytelaars. Find a Way Forward: a Language-Guided Semantic Map Navigator. *arXiv preprint arXiv:2203.03183*, 2022.

[140] S.-C. Wu, J. Wald, K. Tateno, N. Navab, and F. Tombari. SceneGraphFusion: Incremental 3D Scene Graph Prediction from RGB-D Sequences. In *IEEE/CVF CVPR*, 2021.

[141] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. *arXiv preprint arXiv:1711.00199*, 2017.

[142] B. Xu, W. Li, D. Tzoumanikas, M. Bloesch, A. Davison, and S. Leutenegger. MID-Fusion: Octree-based Object-Level Multi-Instance Dynamic SLAM. In *IEEE ICRA*, 2019.

[143] H. Xu, S. Xie, P.-Y. Huang, L. Yu, R. Howes, G. Ghosh, L. Zettlemoyer, and C. Feichtenhofer. Cit: Curation in training for effective vision-language data. *arXiv preprint arXiv:2301.02241*, 2023.

[144] H. Yang, C. Doran, and J.-J. Slotine. Dynamical pose estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5926–5935, 2021.

[145] L. Yao, R. Huang, L. Hou, G. Lu, M. Niu, H. Xu, X. Liang, Z. Li, X. Jiang, and C. Xu. Filip: fine-grained interactive language-image pre-training. *arXiv preprint arXiv:2111.07783*, 2021.

[146] R. Yao, G. Lin, S. Xia, J. Zhao, and Y. Zhou. Video object segmentation and tracking: A survey. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 11(4):1–47, 2020.

[147] Y. Yao, Y. Jafarian, and H. S. Park. Monet: Multiview semi-supervised keypoint detection via epipolar divergence. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 753–762, 2019.

[148] L. Yen-Chen, P. Florence, J. T. Barron, T.-Y. Lin, A. Rodriguez, and P. Isola. NeRF-Supervision: Learning Dense Object Descriptors from Neural Radiance Fields. *arXiv preprint arXiv:2203.01913*, 2022.

[149] M. Yi, Q. Cui, H. Wu, C. Yang, O. Yoshie, and H. Lu. A simple framework for text-supervised semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7071–7080, 2023.

[150] Y. Yu, A. Meka, M. Elgharib, H.-P. Seidel, C. Theobalt, and W. A. Smith. Self-supervised outdoor scene relighting. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXII 16*, pages 84–101. Springer, 2020.

[151] Y.-J. Yuan, Y.-T. Sun, Y.-K. Lai, Y. Ma, R. Jia, and L. Gao. Nerf-editing: geometry editing of neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18353–18364, 2022.

[152] S. Zakharov, I. Shugurov, and S. Ilic. DPOD: 6D pose object detector and refiner. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1941–1950, 2019.

[153] S. Zakharov, W. Kehl, A. Bhargava, and A. Gaidon. Autolabeling 3d objects with differentiable rendering of sdf shape priors. In *IEEE/CVF CVPR*, pages 12224–12233, 2020.

[154] A. Zareian, K. D. Rosa, D. H. Hu, and S.-F. Chang. Open-Vocabulary Object Detection Using Captions. In *IEEE/CVF CVPR*, 2021.

[155] H. Zhang, P. Zhang, X. Hu, Y. Chen, L. Li, X. Dai, L. Wang, L. Yuan, J. Hwang, and J. Gao. GLIPv2: Unifying Localization and Vision-Language Understanding. In *NeurIPS*, 2022.

[156] K. Zhang, G. Riegler, N. Snavely, and V. Koltun. Nerf++: Analyzing and improving neural radiance fields. *arXiv preprint arXiv:2010.07492*, 2020.

[157] X. Zhang and A. Boularias. Optical flow boosts unsupervised localization and segmentation. 2023.

[158] X. Zhang, N. Tseng, A. Syed, R. Bhasin, and N. Jaipuria. Simbar: Single image-based scene relighting for effective data augmentation for automated driving vision tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3718–3728, 2022.

[159] Y. Zhang, Z. Wang, J. H. Liew, J. Huang, M. Zhu, J. Feng, and W. Zuo. Associating spatially-consistent grouping with text-supervised semantic segmentation. *arXiv preprint arXiv:2304.01114*, 2023.

[160] X. Zhao, W. Ding, Y. An, Y. Du, T. Yu, M. Li, M. Tang, and J. Wang. Fast segment anything. *arXiv preprint arXiv:2306.12156*, 2023.

[161] S. Zhi, T. Laidlow, S. Leutenegger, and A. J. Davison. In-Place Scene Labelling and Understanding with Implicit Scene Representation. In *IEEE/CVF International Conference on Computer Vision*, 2021.

[162] S. Zhi, E. Sucar, A. Mouton, I. Haughton, T. Laidlow, and A. J. Davison. iLabel: Revealing Objects in Neural Fields. *IEEE Robotics and Automation Letters*, 2022.

[163] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba. Scene Parsing through ADE20K Dataset. In *IEEE CVPR*, 2017.

[164] Z. Zhu, S. Peng, V. Larsson, W. Xu, H. Bao, Z. Cui, M. R. Oswald, and M. Pollefeys. NICE-SLAM: Neural Implicit Scalable Encoding for SLAM. In *IEEE/CVF CVPR*, 2022.

[165] X. Zou, Z.-Y. Dou, J. Yang, Z. Gan, L. Li, C. Li, X. Dai, H. Behl, J. Wang, L. Yuan, et al. Generalized Decoding for Pixel, Image, and Language. *arXiv preprint arXiv:2212.11270*, 2022.