3D MODEL-ASSISTED LEARNING FOR OBJECT DETECTION AND POSE ESTIMATION

by

Georgios Georgakis
A Dissertation
Submitted to the
Graduate Faculty
of
George Mason University
In Partial fulfillment of
The Requirements for the Degree
of
Doctor of Philosophy
Computer Science

Committee:

_____	Dr. Jana Košecká, Dissertation Director
_____	Dr. Zoran Duric, Committee Member
_____	Dr. Jessica Lin, Committee Member
_____	Dr. Daniel Lofaro, Committee Member
_____	Dr. Huzefa Rangwala, Department Chair
_____	Dr. Kenneth S. Ball, Dean, Volgenau School of Engineering

Date: _____ Spring Semester 2020
George Mason University
Fairfax, VA

3D Model-Assisted Learning for Object Detection and Pose Estimation

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy at George Mason University

By

Georgios Georgakis
Master of Science
George Mason University, 2015
Diploma of Engineering
Technical University of Crete, 2012

Director: Dr. Jana Košecká, Professor
Department of Computer Science

Spring Semester 2020
George Mason University
Fairfax, VA

Dedication

To Stavros, Elli, Marianna, and Kyriaki

Acknowledgments

Obtaining my PhD was not a lonely endeavor that extended exclusively during graduate school. It was rather the culmination of experiences and contributions of multiple people that have shaped me along the way since childhood.

First of all I would like to thank my family for their unconditional love and support and for teaching me the value of constantly improving myself. This I especially owe to my father Stavros who always encouraged me to learn new things and not approach any subject superficially. His principle is, if you are going to do something, do it right. My mother Elli taught me to appreciate the "unseen" contributions of others, by always being there for me without asking anything in return. Through my younger sister Marianna I learned how to care about somebody else other than myself from a very young age. I am also grateful to other members of my extended family such as Alecos, Froso, Stephanie, Maro, Kostakis, Yiannakis and so many more that have provided a healthy environment while growing up.

Of course I owe a lot to my advisor Jana Košecká. I am grateful to her for giving me the opportunity to work in her lab. If she hadn't hired me as a research assistant back in 2013 I would not be here today. Under her guidance I learned how to constantly question my ideas and to always go back to the underlying principles of a problem whenever I was stuck. I believe the most important lesson I learned from her is to be pragmatic. She taught me how to ground my thoughts and ideas into practical solutions along with the value of being concise, which stems from her ability to get directly to the heart of the matter in any research topic. In addition, her help in writing papers had been invaluable. I hope that in the future I'll be able to reach her writing skills.

During my time at GMU I had the pleasure of working with multiple great people. I am grateful to the other members of the lab - Gautam Singh, Alimoor Reza, Sugata Banerji, Shenghui Zhou, Arsalan Mousavian, Phi-Hung Le, Sanaz Rajabi, Hui Zheng, and Yimeng Li. Special thanks to Gautam and Alimoor who I viewed as my mentors and helped me a lot in my first steps of my PhD. I will always remember our discussions that often ventured outside of our research topics; from Indian and Greek mythology to football (don't worry Gautam, I am sure Manchester United will return to the top of EPL eventually). I thank Eadom Dessalene and Kevin Molloy with whom I had interesting interactions. I would also like to thank the professors that I worked as a teaching assistant with, Mark Snyder and Yutao Zhong. They are both excellent instructors and I learned a lot by teaching lab sections under their supervision.

I had the joy of spending three summers at various internship positions for which I am most thankful of my mentors Srikrishna Karanam and Ziyan Wu. Srikrishna has probably the most positive - let's get things done - energy I have encountered during my PhD, while Ziyan always had very insightful advise about research or otherwise. I have found two excellent research collaborators and friends in them. I would also like to thank the other people I came across that made my interships such fruitful and enjoyable experiences - Kuan-Chuang Peng, Arun Innanje, Varun Manjunatha, Yunye Gong, Kunpeng Li, Spondon

Kundu, Rajat Singh, Prithviraj Dhar, Xuejian Rong, Lezi Wang, Mohamed Elfeki, Jan Ernst, Changjiang Cai, Lidan Wang, Terrence Chen, and Meng Zeng.

I consider myself lucky to have had lovely roommates such as Gino Panza, Kaitlyn McIntyre-Panza, Donal Murray, Caroline Milsk, Odin the dog, and Dexter the cat. Gino was the first person that I met in the US and taught me a lot about the American way of life. He also introduced me to Odin, the most gentle slobber machine I have ever met.

I am grateful of having friends in Fairfax such as George Panteras, Kostas Kollias, Nikos and Ellie Kiourti, and Panagiotis and Stella Chatzigianni. Their company was a constant reminder that there was life outside the lab. Special thanks to Kostas with whom I always had interesting discussions especially about movies.

I thank the professors at University of Cyprus, Constantinos Pattichis and Yiorgos Chrysanthou with whom I made my first steps in computer vision. I will always remember my summer 2013 UCY project with fellow master's student Savvas Christodoulou.

I thank my undergraduate advisor Michail Lagoudakis for the endless hours of coaching me over my thesis at TUC. My interactions with him were what inspired me to pursue a PhD in the first place.

I am indebted to my friend and fellow TUC student Constantinos Chatzipetrou to whom I owe my first steps in coding.

I am blessed to still be in the company of my childhood friends Panikos, Chrysovalantis, Dimitris, and Pambos. They are responsible for many fond memories and have profoundly influenced my character in more ways than I can count. Our discussions over politics, sports, ethics, and many more have taught me how to approach a subject from multiple perspectives. Thank you for being so close even though I am so far away.

Finally and most importantly, I would like to thank my partner in life, Kyriaki. Thank you for being there during all the hard times. You are, truly, a gift.

Table of Contents

List of Tables

List of Figures

Abstract

3D MODEL-ASSISTED LEARNING FOR OBJECT DETECTION AND POSE ESTIMATION

Georgios Georgakis, PhD

George Mason University, 2020

Dissertation Director: Dr. Jana Košecká

Supervised learning paradigm for training Deep Convolutional Neural Networks (DCNN) rests on the availability of large amounts of manually annotated images, which are necessary for training deep models with millions of parameters. In this thesis, we present novel techniques for mitigating the required manual annotation, by generating large object instance datasets through compositing textured 3D models onto commonly encountered background scenes to synthesize training images. The generated training data augmented with real world annotations outperforms models trained only on real data. Non-textured 3D models are subsequently used for keypoint learning and matching, and 3D object pose estimation from RGB images. The proposed methods showcase promising results with regards to generalization on new and standard benchmark datasets. In the final part of the thesis, we investigate how these perception capabilities can be leveraged and encoded in a spatial map, in order to enable an agent to successfully navigate towards a target object.

Chapter 1: Introduction

Powerful deep learning methods now perform almost as good as humans on certain benchmarks [8], a fact that was unthinkable until recently. These methods demonstrated significant performance improvements on core computer vision problems such as object detection [9–11], and pose estimation on objects [12–14], and humans [15–17]. However, as these methods have considerable amounts of parameters, the improvements are contingent on the existence of large annotated training sets, sometimes containing millions of examples. The annotation is often performed by humans, an extremely time-consuming procedure.

This thesis explores alternative approaches for training deep convolutional neural networks with large number of parameters, which do not require manual annotations. The core ideas suggest creative ways of using 3D object models to tackle the problems of object instance detection, keypoint detection and descriptor learning, and object pose estimation. Furthermore, we investigate the usefulness of object detection in the context of target driven navigation.

In summary, the topics discussed in this thesis are the following:

1. **Object Instance Detection in Indoor Scenes**, which involves both the localization and recognition of object instances of interest in an image.

2. **Keypoint Detection and Descriptor Learning**, where the objective is to establish correspondences between two depth images.

3. **Object Pose Estimation**, where given an RGB image and object bounding box, the goal is to estimate the 3D pose of the object.

4. **Target driven Navigation**, where the objective is to learn a policy that successfully navigates to a given semantic target and simultaneously constructs a partial map of the environment with semantic information.

Figure 1.1: Annotations for object instance detection in the GMU-Kitchens [2] dataset.

In the following sections we provide more details about the problems considered, along with our contributions.

Object Instance Detection in Indoor Scenes. Given an image, the goal of object detection is to localize all bounding boxes of objects of interest along with identifying the class of each object from a predefined set of classes $C = \{c_1, c_2, ..., c_K\}$. Each bounding box is represented by the image space parameters x, y, w, h, which correspond to the left x-coordinate, the upper y-coordinate, the width, and the height of the box respectively. Examples of ground-truth bounding boxes for object instances are shown in Figure 1.1. We focus on the problem of object instance detection, and consider smaller objects that afford manipulation.

For object category detection, current state-of-the-art methods such as Faster R-CNN [9] and SSD [11] employ deep Convolutional Neural Network (CNN) approaches that have large amounts of parameters and require large annotated datasets for training. For everyday object instances found in indoor environments such datasets are not readily available. In order to address this problem, we have developed (Chapter 2) a novel automated approach for generating synthesized training sets that can be utilized to train CNN-based object detectors [9, 11]. Existing works that make use of synthetic data either render 3D CAD models on simple backgrounds using a randomized procedure [18, 19], or collect image data from

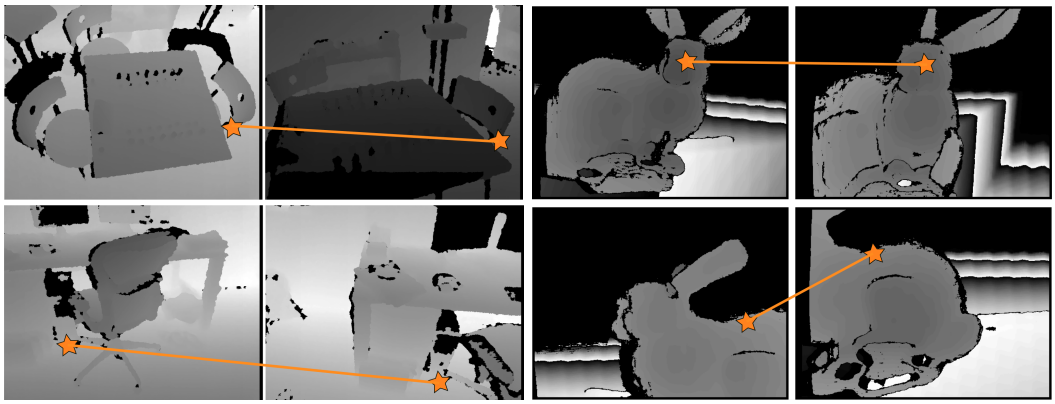


Figure 1.2: Examples of retrieved matches in noisy depth images by our method from the MSR-7 [1] dataset (left) and the Stanford 3D scanning repository [3] (right). For each pair, the query image is to the left, and the image to the right illustrates a retrieved match from a repository of keypoints.

video game engines [20]. In contrast, our approach superimposes real cropped object images in background scenes in an informative manner that respects semantic and geometric constraints in the scene. Specifically, the objects are placed in contextually meaningful positions in the images which improves the overall final performance of the detector. This is demonstrated by comparing different superimposition strategies that range from simple randomized procedures to our informed positioning on the GMU-Kitchens [2] and Washington RGB-D [21] datasets. Additionally, we investigate the augmentation of a small set of real annotated images with our synthesized data and show their superior performance to using only real annotations. This work was published in [22].

Keypoint Detection and Descriptor Learning. This is the task of detecting salient keypoints and their feature representations with the objective of establishing correspondences between pairs of images taken from different poses or between images and 3D models, and can be used for image retrieval, registration, and pose estimation. An example related to our work is shown in Figure 1.2. Traditional methods relied on hand-engineered features such as Harris3D [23], SIFT [24], and FPFH [25]. Deep learning based methods in recent

literature attempt to tackle this problem in the RGB domain and require keypoint annotations that are either collected by humans or by traditional methods [26, 27]. This strategy implicitly pre-defines which points in an image should be considered discriminative, which is ambiguous by nature. In contrast, our method presented in Chapter 3, avoids these problems by taking advantage of readily available CAD models and generates the appropriate ground-truth on-the-fly during training from pairs of rendered noisy depth images. The optimization of the keypoint detector and descriptors is performed jointly and ensures keypoint repeatability and feature representations that are robust to viewpoint variations. The approach is validated on multiple publicly available 3D datasets [1, 3], showing improved performance over the matching objective and was published in [28].

Object Pose Estimation. The third task in this thesis is object pose estimation. This entails the estimation of the 3D rotation and translation $(R, T) \in SE(3)$ of an object of interest. Rotation R is usually parametrized by the Euler angles azimuth, elevation, and in-plane rotation. An example of a ground-truth pose for a "bed" object is shown in Figure 1.3. Given 2D-3D correspondences between the image and a 3D textured instance model, this problem can be solved using the Perspective-n-Point algorithm. Traditionally, interest point detectors and hand-engineered descriptors such as SIFT [24] were used during the matching procedure. More recently, end-to-end CNN-based approaches have been in the forefront of object pose estimation demonstrating superior performance using as input a single cropped RGB image of an object [14, 29–31]. These approaches are very data-hungry and require accurate 3D pose annotations on real RGB images. Currently, the largest dataset that offers *precise* alignments between images and 3D models is the Pix3D [4] which contains the relatively small amount of 10069 images and 395 3D shapes of 9 object categories. In response to this problem, we present (Chapter 4) a novel approach for 3D object pose estimation which does not make use of either 3D texture models or expensive 3D pose annotations. Instead, it requires only textureless CAD models and aligned RGB-D frames of a subset of

Figure 1.3: Rendered image (right) using the provided ground-truth pose for the bed category (left). Images are from the Pix3D [4] dataset.

object instances in order to learn to infer the 3D pose. This is achieved through a series of constraints that enforce viewpoint and modality invariance for local features, and learn how to select keypoints consistently across the RGB and depth modalities. The selection of the keypoints is also learned without any keypoints annotations, through a relative pose estimation objective. During testing, keypoints are extracted from a query RGB image and matched to keypoints extracted from rendered depth images. The approach is demonstrated on the Pix3D [4] dataset for object pose estimation, as well as generalization to object instances not seen during training. The proposed method was published in [32].

Target Driven Navigation. Finally, target driven navigation seeks to learn a policy $\pi(a|o;c)$ that maps the current observation o of an agent to the best possible action a such that it can approach the semantic target c. Figure 1.4 illustrates an instance of the task. Traditional methods focused on constructing a 3D metric map of the environment [33] followed by path planning and control. This requires building the 3D metric map when a novel scene is encountered and it is unable to bring prior knowledge, such as common

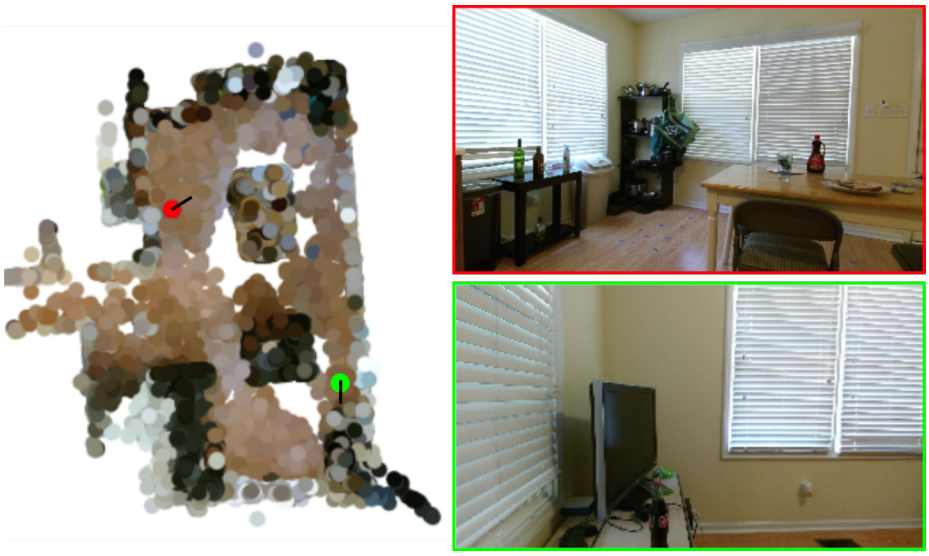


Figure 1.4: A bird-eye view of a scene from the AVD [5] dataset with initial (red) and target (green) poses of the agent and their corresponding views. The target object here is the tv. Best viewed in color.

spatial patterns, when planning in new environments. These priors can be crucial when a target object is not initially visible and the agent needs to make informed decisions about its whereabouts. Recently, several learning based approaches attempt to learn navigation strategies by mapping pixels directly to actions [34, 35]. While these models are able to bring some useful priors during navigation, they do not have a mechanism to encode the spatial patterns in the environment. In Chapter 5, we present a novel method that learns a navigation policy on top of a semantically informed map that does not assume perfect localization. The method consists of a modular architecture for simultaneous mapping and target driven navigation in novel indoor environments. Deep convolutional neural networks are used to extract semantic and appearance representation from RGB images, semantic segmentation and object detection masks, which is then stored in a 2.5D map. Given this representation, the mapping module learns to localize the agent and register consecutive observations in the map. Then, the navigation module learns a policy for reaching semantic targets using the current observations and the up-to-date map. We demonstrate that the use of semantic information improves localization accuracy and the ability of storing spatial

semantic map aids the target driven navigation policy. The two modules are evaluated separately and jointly on Active Vision Dataset [5] and Matterport3D environments [36], demonstrating improved performance on both localization and navigation tasks.

Chapter 2: Synthesizing Training Data for Object Detection in Indoor Scenes

The capability of detecting and searching for common household objects in indoor environments is the key component of the 'fetch-and-delivery' task commonly considered one of the main functionalities of service robots. Existing approaches for object detection are dominated by machine learning techniques focusing on learning suitable representations of object instances. This is especially the case when the objects of interest are to be localized in environments with large amounts of clutter, variations in lighting, and a range of poses. While the problem of detecting object instances in simpler table top settings has been tackled previously using local features, these methods are often not effective in the presence of large amounts of clutter or when the scale of the objects is small.

Current leading object detectors exploit convolutional neural networks (CNNs) and are either trained end-to-end [11] for sliding-window detection or follow the region proposal approach which is jointly fine-tuned for accurate detection and classification [37] [9]. In both approaches, the training and evaluation of object detectors requires labeling of a large number of training images with objects in various backgrounds and poses with the bounding boxes or even segmentations of objects from background.

Often in robotics, object detection is a prerequisite for tasks such as pose estimation, grasping, and manipulation. Notable efforts have been made to collect 3D models for object instances with and without textures, assuming that objects of interest are in proximity, typically on a table top. Existing approaches to these challenges often use either 3D CAD models [38] or texture mapped models of object instances obtained using traditional reconstruction pipelines [39, 40].

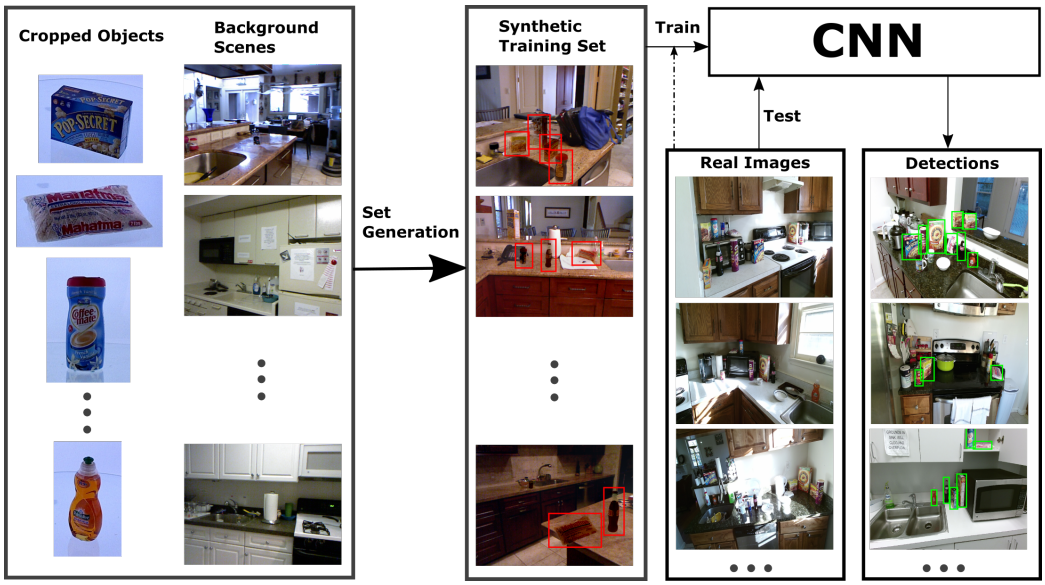


Figure 2.1: Given cropped object images and background scenes we propose an automated approach for generating synthetic training sets that can be used to train current state-of-the-art object detectors, which can then be applied to real test images. The generation procedure takes advantage of scene understanding methods in order to place the objects in meaningful positions in the images. We also explore using a combination of synthetic and real images for training and demonstrate higher detection accuracy compared to training with only real data. Best viewed in color.

In this work we explore the feasibility of using such existing datasets of standalone objects on uniform backgrounds for training object detectors [9, 11] that can be applied in real-world cluttered scenes. We create "synthetic" training images by superimposing the objects into images of real scenes. We investigate effects of different superimposition strategies ranging from purely image-based blending all the way to using depth and semantics to inform positioning of the objects. Toward this end we exploit the geometry and the semantic segmentation of a scene obtained using the state of the art method of [6] to restrict the locations and size of the superimposed object model. We demonstrate that, in the context of robotics applications in indoor environments, these positioning strategies improve the final performance of the detector. This is in contrast with previous approaches [18, 19] which used large synthetic datasets with mostly randomized placement. In summary, our contributions are the following:

1. We propose an automated approach to generate synthetic training data for the task of object detection, which takes into consideration the geometry and semantic information of the scene.

2. Based on our results and observations, we offer insights regarding the superimposition design choices, that could potentially affect the way training sets for object detection are generated in the future.

3. We provide an extensive evaluation of current state-of-the-art object detectors and demonstrate their behavior under different training regimes.

2.1 Related Work

We first briefly review related works in object detection to motivate our choice of detectors, then discuss previous attempts to use synthetic data as well as different datasets and evaluation methodologies.

Object Detection Traditional methods for object detection in cluttered scenes follow the sliding window based pipeline with hand designed flat feature representations (e.g. HOG) along with discriminative classifiers, such as linear or latent SVMs. Examples include DPMs [41] which exploit efficient methods for feature computation and classifier evaluation. These models have been used successfully in robotics for detection in the table top setting [42]. Other effectively used strategies for object detection used local features and correspondences between a model reference image and the scene. These approaches [43, 44] worked well with textured household objects, taking advantage of the discriminative nature of the local descriptors. In an attempt to reduce the search space of the sliding window techniques, alternative approaches concentrated on generating category-independent object proposals [45, 46] using bottom up segmenation techniques followed by classification using traditional features. The flat engineered features have been recently superseded by

approaches based on Convolutional Neural Networks (CNN), which learn features with increased amount of invariance by repeated layering of convolutional and pooling layers. While these methods have been intially introduced for image classification task [47], extensions to object detection include [48] [49]. The R-CNN approach [48] relied on finding object proposals and extracting features from each crop using a pre-trained network, making the proposal generating module independent from the classification module. Recent state of the art object detectors such as Faster R-CNN [9] and SSD [11] are trained jointly in a so called end-to-end fashion to both find object proposals and also classify them.

Synthetic Data There are several previous attempts to use synthetic data for training CNNs. The work of [18] used existing 3D CAD models, both with and without texture, to generate 2D images by varying the projections and orientations of the objects. The approach was evaluated on 20 categories in PASCAL VOC2007 dataset. That work used earlier CNN models [48] where the proposal generation module was independent from fine-tuning the CNN classifier, hence making the dependence on the context and background less prominent than in current models. In the work of [19] the authors used the rendered models and their 2D projections on varying backgrounds to train a deep CNN for pose estimation. In these representative works, objects typically appeared on simpler backgrounds and were combined with the object detection strategies that rely on the proposal generation stage. Our work differs in that we perform informed compositing on the background scenes, instead of placing object-centric synthetic images at random locations. This allows us to train the CNN object detectors to produce higher quality object proposals, rather than relying on unsupervised bottom-up techniques. In [20], a Grand Theft Auto video game engine was used to collect scenes with realistic appearance and their associated category pixel level labels for the problem of semantic segmentation. Authors showed that using these high realism renderings can significantly reduce the effort for annotation. They used a combination of synthetic data and real images to train models for semantic segmentation. Perhaps the closest work to ours is [50], which also generates a synthetic training set by

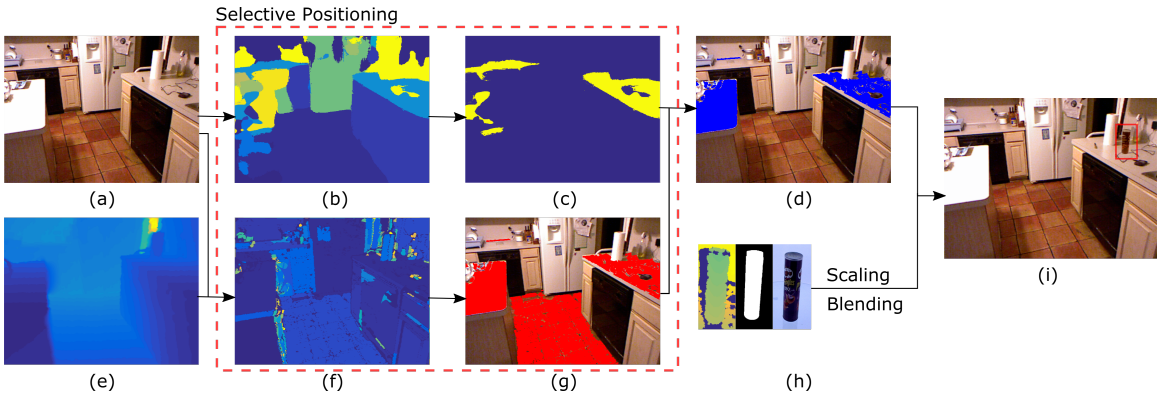


Figure 2.2: Overview of the procedure for blending an object in a background scene. We take advantage of estimated support surfaces (g) and predictions for counters and tables (c) in order to find regions for object placement (d). The semantic segmentation of the scene [6], and the plane extraction are shown in (b) and (f) respectively. (h) presents an example of an object's RGB, depth, and mask images, while (i) shows the final blending result. RGB and depth images of the background scene are in (a) and (e) respectively. Best viewed in color.

taking advantage of scene segmentation to create synthetic training examples, however the task is that of text localization instead of object detection.

2.2 Approach

2.2.1 Synthetic Set Generation

CNN-based object detectors require large amounts of annotated data for training, due to the large number of parameters that need to be learned. For object instance detection the training data should also cover the variations in the object's viewpoint and other nuisance parameters such as lighting, occlusion and clutter. Manually collecting and annotating scenes with the aforementioned properties is time-consuming and costly. Another factor in annotation is the sometimes low generalization capability of trained models across different environments and backgrounds. The work of [51] addressed this problem by building a map of an environment including objects of interest and using Amazon Mechanical Turk for annotation and subsequent training of object detectors in each particular environment.

The authors demonstrated this approach on commonly encountered categories (≈ 20) of household objects. This approach uses human labeling effort for each new each scene and object combination, potentially limiting scalability.

Our approach focuses on object instances and their superimposition into real scenes at different positions, scales, while reducing the difference in lighting conditions and exploiting proper context. To this end, we use cropped images from existing object recognition datasets such as BigBird [39] rather than using 3D CAD models [18, 19]. This allows us to have real colors and textures for our training instances as opposed to rendering them with randomly chosen or artificial samples. The BigBird dataset captures 120 azimuth angles from 5 different elevations for a total of 600 views per object. It contains a total of 125 object instances with a variety of textures and shapes. In our experiments we use the 11 object instances that can be found in the GMU-Kitchens dataset.

The process of generating a composite image with superimposed objects can be summarized in the following steps. First, we choose a background scene and estimate the positions of any support surfaces. This is further augmented by semantic segmentation of the scene, used to verify the support surfaces found by plane fitting. The objects of interest are placed on support surfaces, ensuring their location in areas with appropriate context and backgrounds. The next step is to randomly choose an object and its pose, followed by choosing a position in the image. The object scale is then determined by the depth value of the chosen position and finally the object is blended into the scene. An example of this process is shown in Figure 2.2. We next describe these steps in more detail.

Selective Positioning In natural images, small hand-held objects are usually found on supporting surfaces such as counters, tables, and desks. These planar surfaces are extracted using the method described in [52], which applies RANSAC to fit planes to regions after an initial over-segmentation of the image. Given the extracted planar surfaces's orientations, we select the planes with large extent, which are aligned with the gravity direction as candidate support surfaces. To ensure that the candidate support surfaces belong to a desired semantic category, a support surface is considered valid if it overlaps in the image

with semantic categories of counters, tables and desks obtained by semantic segmentation of the RGB-D image.

Semantic Segmentation To determine the semantic categories in the scene, we use the semantic segmentation CNN of [6], which is pre-trained on MS-COCO and PASCAL-VOC datasets, and fine-tuned on NYU Depth v2 dataset for 40 semantic categories. The model is jointly trained for semantic segmentation and depth estimation, which allows the scene geometry to be exploited for better discrimination between some of the categories. We do not rely solely on the semantic segmentation for object positioning, since it rarely covers the entire support surface, as can be seen in Figure 2.2(c). The combination of the support surface detection and semantic segmentation produces more accurate regions for placing the objects. The aforementioned regions that belong to valid support surfaces are then randomly chosen for object positioning. Finally, occlusion levels are regulated by allowing a maximum of 40% overlap between positioned object instances in the image.

Selective Scaling and Blending The size of the object is determined by using the depth of the selected position and scaling the width w and height h accordingly:

$$\hat{w} = \frac{w\bar{z}}{z} \qquad\qquad \hat{h} = \frac{h\bar{z}}{z}$$

where $\bar{z}$ is the median depth of the object's training images, z is the depth at the selected position in the background image, and $\hat{w}$, $\hat{h}$ are the scaled width and height respectively.

The last step in our process is to blend the object with the background image in order to mitigate the effects of changes in illumination and contrast. We use the implementation from Fast Seamless Cloning [53] with a minor modification. Instead of blending a rectangular patch of the object, we provide a masked object to the fast seamless cloning algorithm which produces a cleaner result. Figure 2.3 illustrates examples of scenes with multiple blended objects.

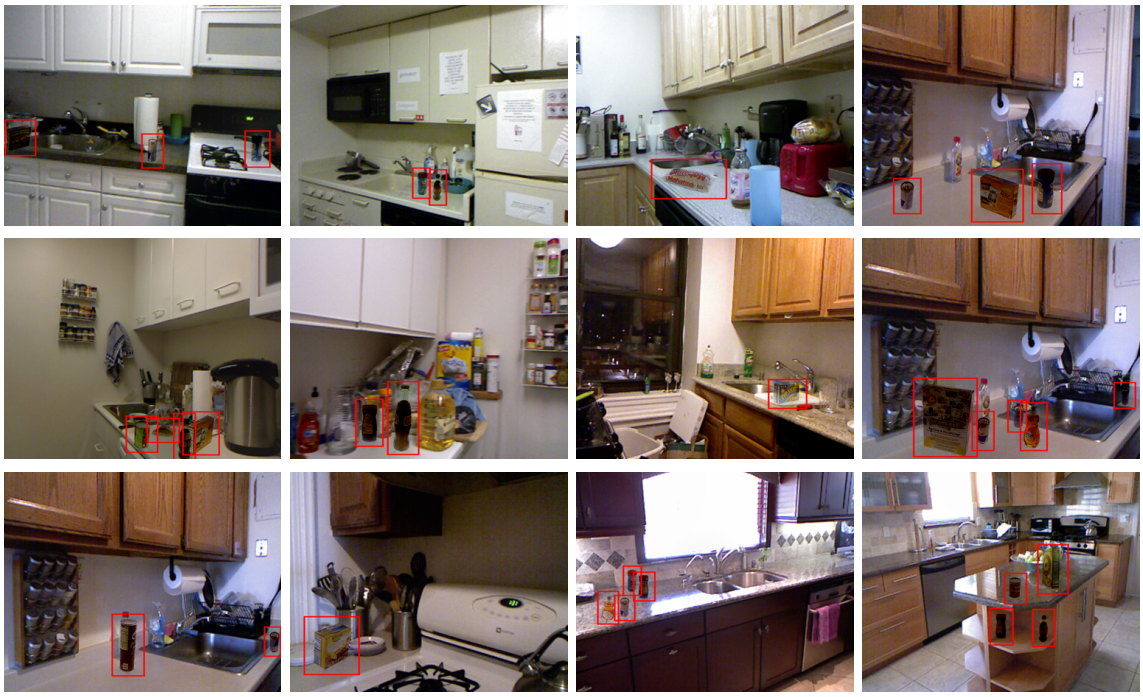


Figure 2.3: Examples of blending object instances from the BigBird dataset into scenes from the NYU Depth V2 dataset. The blended objects are marked with a red bounding box. Best viewed in color.

2.2.2 Object Detectors

For our experiments we employ two state-of-the-art object detectors, Faster R-CNN [9] and Single-Shot Multibox Detector (SSD) [11]. Both Faster R-CNN and SSD are trained end-to-end but their architectures are different. Faster R-CNN consists of two modules. The first module is the Region Proposal Network (RPN) which is a fully convolutional network that outputs object proposals and also an objectness score for each proposal reflecting the probability of having an object inside the region. The second detection network module resizes the feature maps, corresponding to each object proposal to a fixed size, classifies it to an object category and refines the location and the height and width of the bounding box associated with each proposal. The advantage of Faster R-CNN is the modularity of the model; one module that finds object proposals and the second module which classifies each of the proposals. The downside of Faster R-CNN is that it uses the same feature map

Figure 2.4: Comparison between masks from BigBird (top row), and masks after refinement with Graph-cut (bottom row).

to find objects of different sizes which causes problems for small objects. SSD tackles this problem by creating feature maps of different resolutions. Each cell of the coarser feature maps captures larger area of the image for detecting large objects whereas the finer feature maps are detecting smaller objects. These multiple feature maps allow higher accuracy for a given input resolution, providing SSD's speed advantage for similar accuracy. Both detectors have difficulties for objects with small size in pixels, making input resolution an important factor.

2.3 Experiments

In order to evaluate the object detectors trained on composited images, we have conducted three sets of experiments on two publicly available datasets, the GMU-Kitchen Scenes [2] and the Washington RGB-D Scenes v2 dataset [21]. In the first experiment, training images are generated by choosing different compositing strategies to determine the effect of positioning, scaling, and blending on the performance. The object detectors are trained on composited images and evaluated on real scenes. In the second set of experiments we examine the effect of varying proportion of synthetic/composited images and real training images. Finally we use synthetic data for both training and testing in order to show the

Table 2.1: Average precision results for the Faster R-CNN detector for all experiments on the GMU-Kitchens dataset. The Synthetic+Real to Real experiments were performed using the SP-BL-SS set plus the percentage of real data shown in the table.

Faster R-CNN	coca cola	coffee mate	honey bunches	hunts sauce	mahatma rice	nature valley 1	nature valley 2	palmolive orange	pop secret	pringles bbq	red bull	mAP
Real to Real												
1. **3-Fold**	78.9	92.0	91.9	81.8	74.7	93.4	85.9	76.6	90.7	86.4	54.6	82.5
Synthetic to Real												
2. **RP-SI-RS**	37.2	68.3	**72.5**	26.1	32.3	70.2	57.2	29.0	46.9	2.9	20.4	42.1
3. **RP-BL-RS**	**62.4**	69.3	58.2	32.4	4.3	51.7	47.1	**39.3**	32.2	**59.2**	**30.0**	44.2
4. **SP-SI-SS**	45.2	**71.7**	66.6	26.0	**45.5**	80.5	**78.4**	37.8	46.1	27.1	9.7	48.6
5. **SP-BL-SS**	55.5	67.9	71.2	**34.6**	30.6	**82.9**	66.2	33.1	**54.3**	54.8	17.7	**51.7**
Synthetic+Real to Real												
6. **1% real**	65.1	85.8	85.7	62.3	51.6	90.4	85.6	54.3	79.4	70.6	32.2	69.3
7. **10% real**	70.5	91.5	89.6	82.2	62.8	94.6	87.4	66.3	89.5	87.4	49.5	79.2
8. **50% real**	79.3	92.5	91.1	77.3	**86.2**	95.4	87.9	77.8	91.6	90.1	52.2	83.8
9. **100% real**	**82.6**	**92.9**	**91.4**	**85.5**	81.9	**95.5**	**88.6**	**78.5**	**93.6**	**90.2**	**54.1**	**85.0**
Synthetic to Synthetic												
10. **RP-SI-RS**	**99.6**	**100**	**99.7**	**99.6**	**99.6**	**99.8**	**99.7**	**98.9**	**99.7**	**99.4**	**98.7**	**99.5**
11. **SP-BL-SS**	79.2	84.4	94.8	79.3	94.6	92.6	89.5	79.9	93.1	89.1	65.8	85.7

reduction of over-fitting to superimposition artifacts during training when the proposed approach of data generation is employed.

2.3.1 Datasets and Backgrounds

For our experiments, we utilized the following datasets:

GMU Kitchen Scenes dataset [2] The GMU-Kitchens dataset includes 9 RGB-D videos of kitchen scenes with 11 object instances from the BigBird dataset. We also used all 71 raw kitchen videos from the NYU Depth Dataset V2 [54] with a total of around 7000 frames as background images. For each image we generate four synthetic images with different variations in objects that are added to the scene, pose, scale, and the location that the objects are put. The object identities and their poses are randomly sampled from the BigBird dataset, from 360 examples per object with 3 elevations and 120 azimuths.

Table 2.2: Average precision results for the SSD detector for all experiments on the GMU-Kitchens dataset. The Synthetic+Real to Real experiments were performed using the SP-BL-SS set plus the percentage of real data shown in the table.

SSD	coca cola	coffee mate	honey bunches	hunts sauce	mahatma rice	nature valley 1	nature valley 2	palmolive orange	pop secret	pringles bbq	red bull	mAP
Real to Real												
1. **3-Fold**	46.7	73.7	81.8	64.5	62.9	83.9	70.3	69.8	76.1	64.8	27.7	65.6
Synthetic to Real												
2. **RP-SI-RS**	9.1	15.9	49.4	9.8	**13.5**	61.6	42.2	15.5	36.9	1.0	0.8	23.2
3. **RP-BL-RS**	9.1	15.7	41.4	**10.7**	9.9	41.2	39.0	**28.7**	36.2	11.5	0.7	22.2
4. **SP-SI-SS**	10.5	17.5	47.2	0.1	9.1	44.9	36.5	24.0	9.1	5.5	**10.3**	19.5
5. **SP-BL-SS**	**18.3**	**22.1**	**58.9**	9.5	11.1	**75.7**	**65.5**	23.8	**59.4**	**14.6**	9.1	**33.5**
Synthetic+Real to Real												
6. **1% real**	39.4	71.8	80.4	50.2	45.2	82.6	74.9	57.8	78.1	54.2	28.5	60.3
7. **10% real**	59.4	83.8	83.7	66.2	60.7	87.3	**79.8**	72.6	83.4	77.6	33.0	71.6
8. **50% real**	**64.6**	84.2	**87.6**	70.4	67.1	**89.2**	79.7	**75.4**	80.1	79.3	**37.6**	**74.1**
9. **100% real**	59.0	**84.5**	85.1	**74.2**	**67.5**	87.4	78.9	71.3	**85.2**	**79.9**	**37.6**	73.7
Synthetic to Synthetic												
10. **RP-SI-RS**	**90.8**	**90.9**	**90.8**	**90.8**	**90.9**	**90.9**	**90.8**	**90.7**	**90.9**	**90.8**	**90.6**	**90.8**
11. **SP-BL-SS**	84.3	86.7	88.1	81.7	88.9	83.5	80.8	83.1	84.5	86.4	74.0	83.8

The images where the support surfaces were not detected are removed from the training set, making our effective set around 5000 background images. Cropped object images from BigBird dataset of the 11 instances contained in GMU-Kitchens were used for superimpositioning. We refine the provided object masks with GraphCut [55], in order to get cleaner outlines for the objects. This helps with the jagged and incomplete boundaries of certain objects (e.g. coke bottle), which are due to imperfect masks obtained from the depth channel of RGB-D data caused by reflective materials. Figure 2.4 illustrates a comparison between masks from BigBird and masks refined with GraphCut algorithm. For comparison with the rest of the experiments we also provide the performance of the object detectors (row 1 of Tables 2.1 and 2.2) trained and tested on the real data. The train-test split follows the division of the dataset into three different folds. In each fold six scenes are used for training and three are used for testing, as shown in [2].

Table 2.3: Average precision results for the Faster R-CNN detector for all experiments on the WRGB-D dataset. The Synthetic+Real to Real experiments were performed using the SP-BL-SS set plus the percentage of real data shown in the table.

Faster R-CNN	bowl	cap	cereal box	coffee mug	soda can	mAP
Real to Real						
1.**Scenes 1-7**	99.7	95.5	99.6	96.7	96.7	97.9
Synthetic to Real						
2. **RP-SI-RS**	65.2	39.4	69.2	57.0	29.4	52.0
3. **RP-BL-RS**	82.6	47.3	93.0	**74.9**	52.7	70.1
4. **SP-SI-SS**	**86.6**	62.4	93.6	68.6	55.5	73.4
5. **SP-BL-SS**	82.3	**70.9**	**96.8**	74.2	**66.3**	**78.1**
Synthetic+Real to Real						
6. **1% real**	98.3	92.7	98.6	96.6	94.9	96.2
7. **10% real**	**99.6**	96.0	99.6	96.9	97.1	97.8
8. **50% real**	99.5	96.5	**99.9**	**97.3**	97.8	**98.2**
9. **100% real**	99.4	**97.0**	99.3	97.2	**98.1**	**98.2**
Synthetic to Synthetic						
10. **RP-SI-RS**	**98.5**	**99.5**	**99.5**	**96.9**	92.6	**97.4**
11. **SP-BL-SS**	97.4	97.5	97.3	95.1	**93.5**	96.2

Washington RGB-D Scenes v2 dataset (WRGB-D) [21] The WRGB-D dataset includes 14 RGB-D videos of indoor table-top scenes containing instances of objects from five object categories: bowl, cap, cereal box, coffee mug, and soda can. The synthetic training data is generated using the provided background scenes (around 3000 images) and cropped object images for the present object categories in the WRGB-D v1 dataset [42]. For each background image we generate five synthetic images to get a total of around 4600 images. As mentioned earlier, images without a support surface are discarded. The images that belong to seven of these scenes are used for training and the rest is used for testing. Line 1 in Tables 2.3 and 2.4 shows the performance of the two object detectors with this split of the real training data.

Table 2.4: Average precision results for the SSD detector for all experiments on the WRGB-D dataset. The Synthetic+Real to Real experiments were performed using the SP-BL-SS set plus the percentage of real data shown in the table.

SSD	bowl	cap	cereal box	coffee mug	soda can	mAP
Real to Real						
1.**Scenes 1-7**	90.8	90.2	90.9	89.9	89.3	90.2
Synthetic to Real						
2. **RP-SI-RS**	77.2	78.5	**90.9**	74.1	70.1	78.2
3. **RP-BL-RS**	71.5	62.9	90.3	73.1	65.2	72.6
4. **SP-SI-SS**	**77.8**	**79.8**	90.8	73.4	**75.5**	**79.5**
5. **SP-BL-SS**	71.9	75.9	90.7	**74.3**	75.0	77.5
Synthetic+Real to Real						
6. **1% real**	87.7	88.3	90.8	88.1	89.5	88.9
7. **10% real**	90.8	89.5	90.8	90.4	**90.8**	90.5
8. **50% real**	**90.9**	**90.6**	**90.9**	90.3	90.6	90.7
9. **100% real**	**90.9**	90.5	**90.9**	**90.8**	**90.8**	**90.8**
Synthetic to Synthetic						
10. **RP-SI-RS**	90.5	**90.9**	**90.9**	**90.2**	**90.0**	**90.5**
11. **SP-BL-SS**	**90.7**	90.7	90.4	89.5	89.2	90.1

2.3.2 Synthetic to Real

In this experiment we use the synthetic training sets generated with different combinations of generation parameters for training, and test on real data. The generation parameters that we vary are: Random Positioning (**RP**) / Selective Positioning (**SP**), Simple Superimposition (**SI**) / Blending (**BL**), and Random Scale (**RS**) / Selective Scale (**SS**), where **SP**, **SS**, and **BL** are explained in Section 2.2.1. For **RP** we randomly sample the position for the object in the entire image, for **RS** the scale of the object is randomly sampled from the range of 0.2 to 1 with a step of 0.1, and for **SI** we do not use blending but instead we superimpose the masked object directly on the background.

The objective of this experiment is to investigate the effect of the generation parameters on the detection accuracy. For example, if a detector is trained on a set generated with

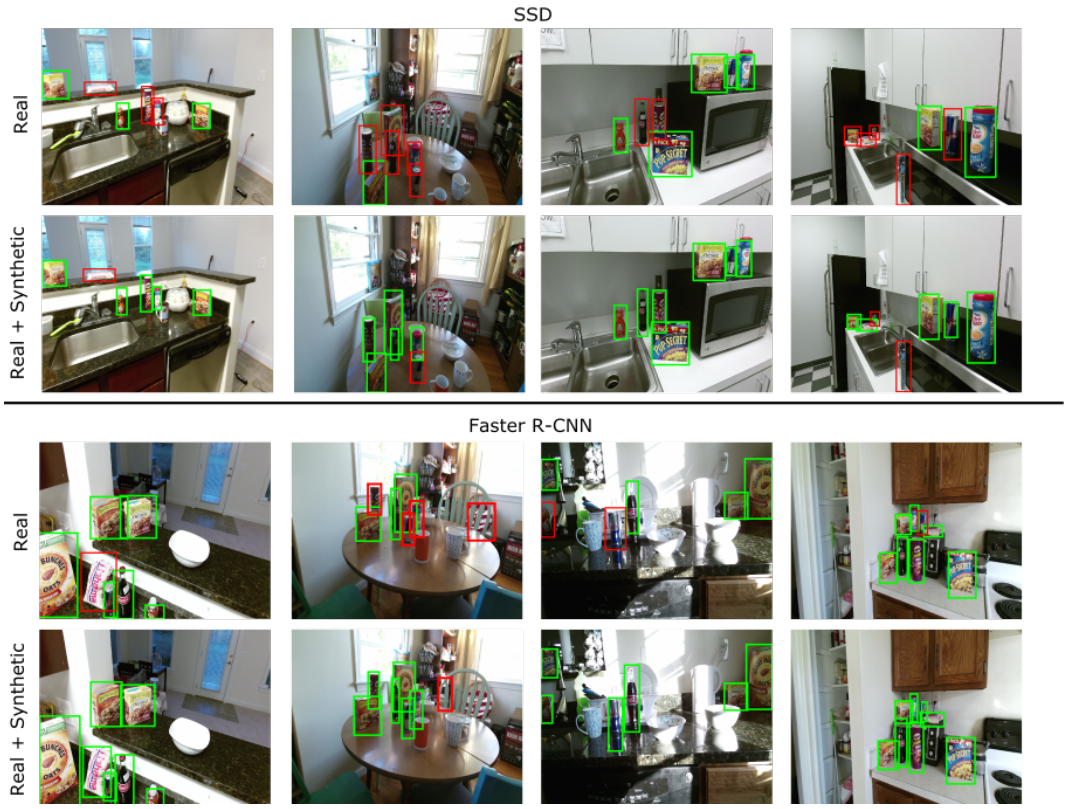


Figure 2.5: Detection examples for the SSD and Faster R-CNN object detectors on the GMU-Kitchens dataset. Rows 1 and 3 show results when only the real training data were used, while rows 2 and 4 present results after the detectors were trained with the synthetic set **SP-BL-SS** and 50% of the real training data. The green bounding boxes depict correct detections, while the red represent false classifications and missed detections. Training with a combination of synthetic and real data proves beneficial for the detection task, as the detectors are more robust to small objects and viewpoint variation. Best viewed in color.

selective positioning, with blending, and selective scale, how does it compare to another detector which is trained on a completely randomly generated set with blending? If the former demonstrates higher performance than the latter, then we can assume that selective positioning and scaling are important and superior to random positioning. For each trained detector, a combination of the generation parameters (e.g. **SP-BL-SS**) is chosen, and then the synthetic set is generated using our proposed approach along with its bounding box annotations for each object instance. The detector is trained only on the synthetic data and then tested on the real data.

The results are shown on lines 2-5 in Tables 2.1 (Faster R-CNN) and 2.2 (SSD) for the GMU-Kitchens dataset and in Tables 2.3 (Faster R-CNN) and 2.4 (SSD) for the WRGB-D dataset. Note that for the GMU-Kitchens dataset, all frames from 9 scenes videos were used for testing. We report detection accuracy on four combinations of generation parameters, **RP-SI-RS**, **RP-BL-RS**, **SP-SI-SS**, and **SP-BL-SS**. Other combinations such as **SP-BL-RS** and **RP-BL-SS** have also been tried, however we noticed that applying selective positioning without selective scaling and vice-versa, does not yield any significant improvements.

For both datasets, we first notice that using only synthetic data for training considerably lowers the detection accuracy compared to using real training data. Nevertheless, when training with synthetic data, the **SP-BL-SS** generation approach produced an improvement of 10.3% and 9.6% for SSD and Faster R-CNN respectively over the randomized generation approach, **RP-SI-RS**, on the GMU-Kitchens dataset. This suggests that selective positioning and scaling are important factors when generating the training set.

In the case of the WRGB-D dataset, different blending strategies work better for SSD and Faster R-CNN, **SP-SI-SS** and **SP-BL-SS** respectively. The right choice of blending strategy seems to improve Faster R-CNN somewhat more, while the overall performance of the two detectors is comparable. The positioning strategy, **SP** vs **RP**, affects the two detectors differently on this dataset. SSD achieves higher performance with the random positioning **RP-SI-RS**, while Faster R-CNN shows a large improvement of 26.1% when it is trained with **SP-BL-SS**. This can be explained by the fact that Faster R-CNN is trained on proposals from the Region Proposal Network (RPN), which under-performs when objects are placed randomly in the image (as in **RP-SI-RS**). On the other hand, SSD does not have any prior knowledge about the location of the objects so it learns to regress the bounding boxes from scratch. The bounding boxes in the beginning of the training are generated randomly until the SSD learns to localize the objects. This trend is not observed for the GMU-Kitchens dataset since it has more clutter in the scenes and higher variability of backgrounds, which makes the localization of the objects harder. To justify this argument,

Table 2.5: Recall (%) results for the RPN on the GMU-Kitchens and WRGB-D datasets on two different Intersection over Union (IoU) thresholds. In all cases, RPN generated 3000 proposals per image.

IoU	GMU-Kitchens	WRGB-D
0.5	76.6	98.0
0.7	28.6	60.8

we performed a side-experiment where we run the pre-trained RPN on both WRGB-D and GMU-Kitchens dataset and evaluated in terms or recall. Results can be seen in Table 2.5, where RPN performs much better on the WRGB-D dataset than on GMU-Kitchens.

2.3.3 Synthetic+Real to Real

We are interested to see how effective our synthetic training set is when combined with real training data. Towards this end the two detectors are trained using the synthetic set with selective positions and blending **SP-BL-SS** with certain percentage of the real training data: 1%, 10%, 50%, and 100%. For the real training data, besides the case of 100%, the images are chosen randomly.

Results are shown in lines 6-9 in Tables 2.1 (Faster R-CNN) and 2.2 (SSD) for the GMU-Kitchens dataset and in Tables 2.3 (Faster R-CNN) and 2.4 (SSD) for the WRGB-D dataset. *What is surprising in these results* is that when synthetic training data is combined with only 10% of the real training data, we achieve higher or comparable detection accuracy than when the training set is only comprised with real data (see line 1 in both tables). In the case of SSD in the GMU-Kitchens dataset, we observe an increase of 6%. Only exception is Faster R-CNN on the GMU-Kitchens dataset which achieves a 2.3% lower performance, however, when we use 50% of the real training data we get a better performance of 1.3%. In all cases, when the synthetic set is combined with 50% and 100% of the real data, it outperforms the training with the real training set.

The results suggest that our synthetic training data can effectively augment existing

Table 2.6: Comparison in performance of SSD / Faster R-CNN between training with only real to training with real+synthetic, with varying amounts of real data. The amount of synthetic data is constant.

GMU-Kitchens		
Percentage of Real	Only Real	Real+Synthetic
1%	57.4 / **70.8**	**60.3** / 69.3
10%	61.5 / **81.1**	**71.6** / 79.2
50%	66.4 / 82.4	**74.1** / **83.8**
100%	65.6 / 82.5	**73.7** / **85.0**
WRGB-D		
Percentage of Real	Only Real	Real+Synthetic
1%	**89.1** / 95.6	88.9 / **96.2**
10%	89.4 / 97.5	**90.5** / **97.8**
50%	90.2 / 97.6	**90.7** / **98.2**
100%	90.2 / 97.9	**90.8** / **98.2**

datasets even when the actual number of real training examples is small. This is particularly useful when only a small subset of the data is annotated. Specifically, in our settings, the 10% of real training data refers to around 400 images in the GMU-Kitchens dataset, and around 600 in the WRGB-D dataset. Figure 2.5 presents examples for which the detectors were unable to detect objects when they were trained with only real data, but succeeded when the training set was augmented with our synthetic data.

We further support our argument by comparing the performance of the detectors trained only on varying percentages of the real data to being trained by real+synthetic in Table 2.6. The synthetic set here is also generated using **SP-BL-SS**. Note that for most of the cases the accuracy increases when the detectors are trained with both real and synthetic data, and the largest gain is observed for SSD.

Finally, we present results for the GMU-Kitchens dataset when the percentage of synthetic data (**SP-BL-SS**) is varied, while the real training data remains constant, in Table 2.7. Again, SSD shows a large and continuing improvement as the amount of the synthetic data increases, while Faster R-CNN achieves top performance when half of the synthetic data are used for training.

Table 2.7: Comparison in performance of SSD / Faster R-CNN on the GMU-Kitchens dataset for increasing amounts of the synthetic data, while all real data are used.

Training Set	Accuracy
Real	65.6 / 82.5
Real+Synth(10%)	69.0 / 84.5
Real+Synth(50%)	72.7 / **85.7**
Real+Synth(100%)	**73.7** / 85.0

2.3.4 Synthetic to Synthetic

In this experiment, the object detectors are trained and tested on synthetic sets. The objective is to show the reduction of over-fitting on the training data when using our approach to generate the synthetic images, instead of creating them randomly. We used the synthetic sets of **RP-SI-RS** and **SP-BL-SS** and split them in half in order to create the train-test sets.

The results are presented on lines 10 and 11 in Tables 2.1 (Faster R-CNN) and 2.2 (SSD) for the GMU-Kitchens dataset and in Tables 2.3 (Faster R-CNN) and 2.4 (SSD) for the WRGB-D dataset. For GMU-Kitchens, we observe that **RP-SI-RS** achieves results of over 90%, and in the case of Faster R-CNN almost 100%, while at the same time it is the least performing synthetic set in the synthetic to real experiment (see line 2 in table 2.1) described in Section 2.3.2. This is because the detectors over-fit on the synthetic data and cannot generalize to an unseen set of real test data. While the detectors still seem to over-fit on **SP-BL-SS**, the gap between the accuracy on the synthetic testing and real testing data is much smaller, at the order of 17.3% for SSD, and 23.4% for Faster R-CNN (see line 5 in tables 2.1 and 2.2).

On the other hand, for the WRGB-D dataset both synthetic training sets achieve similar results on their synthetic test sets. This is not surprising as the complexity of the scenes is much lower in WRGB-D than in the GMU-Kitchens dataset. Please see section 2.3.2 for more details.

2.3.5 Additional Discussion

We have seen in the results of section 2.3.2, that when a detector is trained on synthetic data and then applied on real data, the performance is consistently lower that training on real data. While this can be attributed to artifacts introduced during the blending process, one other factor is the large difference of backgrounds between the NYU V2 dataset and the GMU-Kitchens. We investigated this through a simple object recognition experiment. We trained the VGG [56] network on the BigBird dataset on the cropped images with elevation angles from cameras 1, 3, and 5, tested on the images with elevation angles from cameras 2 and 4, and achieved recognition accuracy of 98.2%. For comparison, when the VGG is trained on all images from BigBird, and tested on cropped images from the GMU-Kitchens, which contain real background scenes, the accuracy drops down to 79.0%.

2.4 Conclusion

One of the advantages of our method is that it is scalable both with the number of objects of interest and with the set of the possible backgrounds, which makes our method suitable for robotics application. For example, the object detectors can be trained with significantly less annotated data using our proposed training data augmentation. We also showed that our method is more effective when the object placements are based on semantic and geometric context of the scene. This is due to the fact that CNNs implicitly consider the surrounding context of the objects and when superimposition is informed by semantic and geometric factors, the gain in accuracy is larger. Another related observation is that for SSD, accuracy increases more than for Faster R-CNN when training data is augmented by synthetic composite images.

While we showed it is possible to train an object detector with fewer annotated images using synthetically generated images, alternative domain adaptation approaches can be also explored towards the goal of reducing the amount of human annotation required.

In conclusion, we have presented an automated procedure for generating synthetic training data for deep CNN object detectors. The generation procedure takes into consideration geometry and semantic segmentation of the scene in order to make informed decisions regarding the positions and scales of the objects. We have employed two state-of-the-art object detectors and demonstrated an increase in their performance when they are trained with an augmented training set. In addition, we also investigated the effect of different generation parameters and provided some insight that could prove useful in future attempts to generate synthetic data for training object detectors.

Chapter 3: End-to-end Learning of Keypoint Detector and Descriptor for Pose Invariant 3D Matching

Keypoint representations have been a central component of matching, retrieval, pose estimation, and registration pipelines. With the advent of approaches based on deep neural networks, global representations became pervasive in solving these type of problems as they can be trained in a straightforward way in an end-to-end fashion. Their shortcomings are caused by occlusions, partial views or scenes that contain large amount of clutter. In case of local feature representations, deep learning has been also applied to the different stages of the matching pipeline, considering detection, description, or metric learning objectives. Most of the frameworks considered the above objectives separately, used image data, and required a large number of training examples. In order to mitigate these issues, we propose to use deep convolutional networks for learning keypoint representations and a keypoint detector for 3D matching jointly without the need for separate annotations. The costly annotation stage can be avoided due to the availability of large repositories of 3D models and the capability of obtaining depth images from different viewpoints.

For the problem of jointly learning keypoint detectors and descriptors, we define a Siamese network architecture that receives as input a pair of depth images and their pose annotations. Each branch of the architecture is a proposal generation network used to generate patches in the two depth images. The branches share weights and lead to a sampling layer which selects pairs of patches. The pairs are labeled as positive or negative depending on the proximity of their 3D re-projection calculated from the pose labels. In other words, the sampling layer is used to create ground truth data on-the-fly by taking advantage of the initial pose annotations. For training the network, we use the contrastive loss which attempts to minimize the distance in the feature space between positive pairs, and maximize the distance between negative pairs. Therefore, for patches that are very

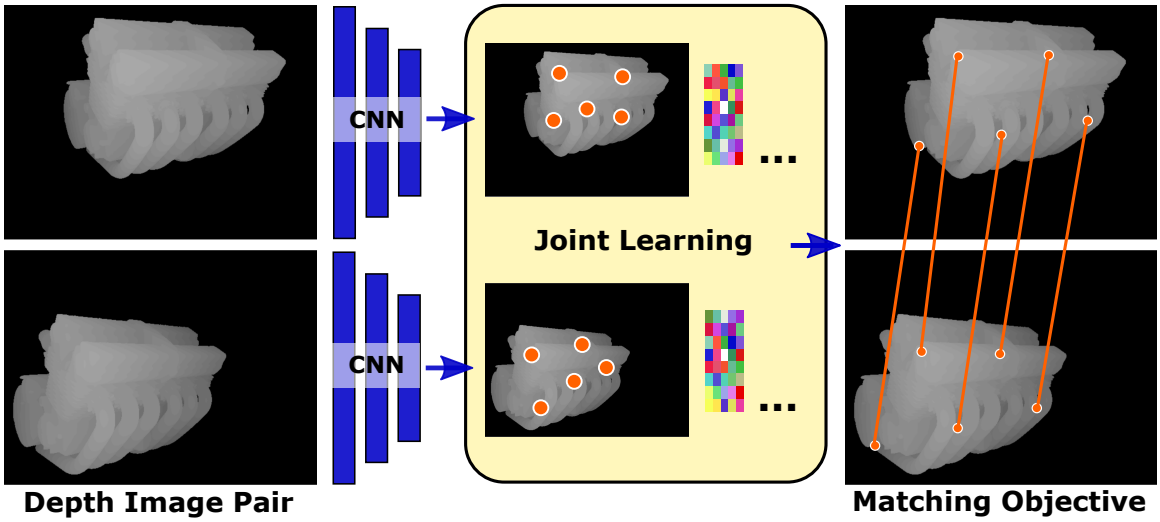


Figure 3.1: We propose a new method for jointly learning keypoint detection and patch-based representations in depth images towards the keypoint matching objective.

close in the 3D space, but sampled from different images, we are learning a representation that has minimal distance in the feature space. In order to learn where to select patches from, we define a *score loss* to gauge the performance of the target task. For example, for pose estimation the *score loss* should consider the number of positive matches between two images from different viewpoints. To summarize, the key contributions of this method include the following:

- We propose the first end-to-end framework for joint learning of keypoint detector and local feature representations for 3D matching,
- We propose a novel sampling layer that can generate labels for local patch correspondence on-the-fly, and
- We design a *score loss* encapsulating task specific objectives that can implicitly provide supervision for joint learning of keypoint detector and its feature representation.

We evaluate the matching accuracy of the proposed approach on multiple benchmark datasets and demonstrate improvements over state-of-the-art methods.

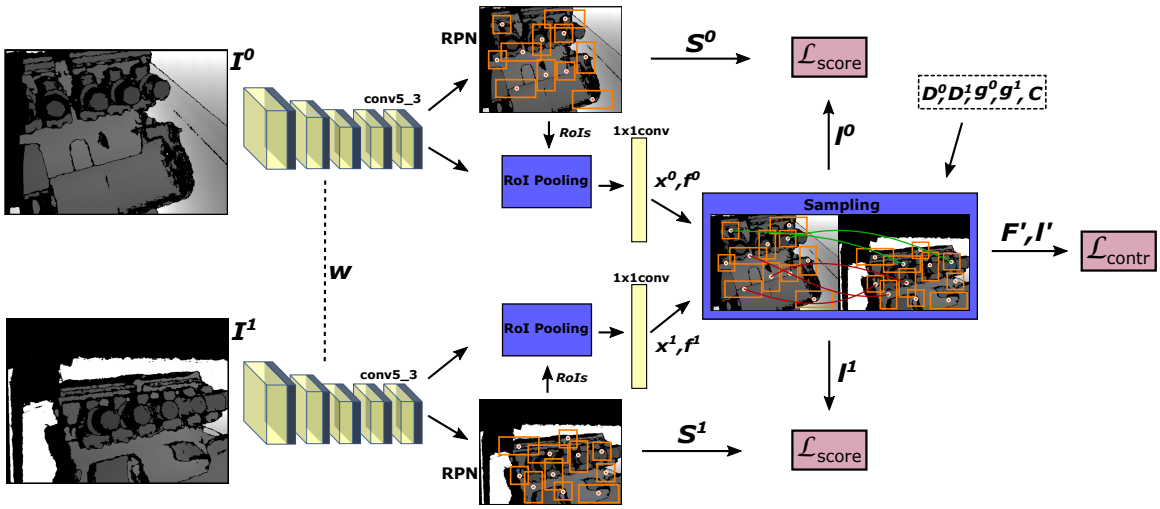


Figure 3.2: Overview of our Siamese architecture. Each branch is a modified Faster R-CNN which receives as an input a depth image and uses VGG-16 as the base representation network. Features from conv5_3 are fed into both the Region Proposal network (RPN) and the Region of Interest (RoI) pooling layer. Given a set of proposals from RPN, we pass their scores to the score loss, while their RoIs are fed to the RoI pooling layer and a fully connected layer to extract the feature vectors. The RoI centroids and the features from both branches are then passed to the sampling layer which organizes them into pairs used by the contrastive loss. Note that the weights between the two branches are shared. For more details on the notations please see section 3.2.2.

3.1 Related Work

There is a large body of work on keypoint detectors and descriptors for both images and 3D depth maps. For images, features such as SIFT [24], FAST [57], BRISK [58], and ORB [59] have been used effectively for various matching tasks. Detectors and descriptors specifically designed for 3D data, including feature histograms [60] and geometry histograms [61] are already included in the Point Cloud Library (PCL) along with many others [62]. These representations were hand-engineered with specific keypoint matching accuracy and/or efficiency goals. A comprehensive review of 3D descriptors can be found in [63].

Advances in convolutional networks led to works in learning descriptors and distance metrics for various matching tasks. The descriptor learning problem has been extensively tackled in case of images and was typically formulated as a supervised learning problem. Given positive and negative examples of pairs of descriptors, the goal is to learn representations where the positive examples are nearby and negative examples are far apart. The methods vary between those which use fixed descriptors and learn a discriminative metric to approaches which take raw patches and learn new representations, or both. 3D reconstructions are often used to obtain large amounts of training data. A comprehensive evaluation of existing approaches can be found in [64].

Most relevant to our task are the descriptor learning methods of Zagoruyko et al. [65], Han et al. [66], and Wohlhart et al. [67], where patch representations are learned discriminatively by means of Siamese or Triplet networks, considering pairs or triplets of descriptors. Similar approaches have been proposed for learning feature representations for matching 3D data [68–70]. Both in case of images and depth maps, the feature descriptors were typically computed at fixed sized patches or patches determined by sampling both spatial locations and scale.

The problem of learning the detector was addressed by Salti et al. [71], where a descriptor specific keypoint detector was proposed by casting the problem of selecting keypoint locations and spatial support as a binary classification task. Savinov et al. [72] formulates

the keypoint detection problem as the problem of learning how to rank points consistently over various image transformations. Other methods such as [27, 73, 74] rely on hand-crafted interest point detectors to collect training data, which is done separately from the training process, affecting the learning of the keypoint detector. In contrast to these approaches, we formulate the problem of selecting keypoints (locations and spatial support) and their feature representations in a single, unified framework, enabling joint optimization of the parameters for both.

3.2 Approach

We are interested in jointly learning a keypoint detector and a view-invariant descriptor using depth data. In contrast to other approaches ([73], [71]), our work does not use hand-crafted keypoint detectors or descriptors as initialization for the learning procedure. Since it is unclear in case of 3D data which keypoint locations should be labeled as "interesting", we do not rely on any hand-labeled datasets with keypoint annotations. Instead, we use a modified Faster R-CNN [9] as the head of our architecture to bootstrap the learning process. Specifically, given two depth images with some pose perturbation, we first generate two sets of proposals, one for each image. Then, we project the proposals in 3D using the known image poses in order to establish positive and negative pairs. Proposals with a small distance in 3D are considered correspondences and are therefore labeled as positives. The pairs are then passed to a contrastive loss in an attempt to minimize feature distance between positive pairs and maximize the distance between negative pairs. Additionally, we introduce a new score loss, which finetunes the parameters of the Region Proposal Network (RPN) of the Faster R-CNN [9] to generate high-scoring proposals in regions of the depth maps for which we can consistently find correspondences. To the best of our knowledge, this is the first work that attempts to jointly optimize the keypoint detection and representation learning process in a purely self-supervised fashion.

3.2.1 Architecture

We choose to use Faster R-CNN [9] as the basis for our architecture because of its modularity. Even though it is initially trained for the task of object detection, its components can provide us with patch-based representations and a trainable mechanism for selecting those patches. We use Faster R-CNN as part of a Siamese model with shared weights. Both branches are connected to a layer responsible for finding correspondences which we call the sampling layer. A contrastive loss is used to train the representation and each branch has a score loss for training the keypoint detection stage. An overview of the architecture with more details can be seen in Figure 3.2.

3.2.2 Training

In order to train our model, we require pairs of depth images $\{I^0, I^1\}$ each with its camera pose information $\{g^0, g^1\}$ and the intrinsic camera parameters C. These can be obtained by rendering a 3D model from multiple viewpoints or using RGB-D video sequences with registered frames ([75], [76]). To pass the depth images $\{I^0, I^1\}$ through our network, we first normalize their depth values in the RGB range and replicate the single channel into a 3-channel image. The rest of the inputs g^0, g^1, C, and the depth images with their values in meters D^0, D^1 are passed directly to the sampling layer.

For each depth image, the Region Proposal Network (RPN) generates a set of scores, and regions of interest (RoIs) for which we use their centroids as the keypoint locations. Each RoI also determines the spatial extent used for feature computation for the current keypoint and after RoI pooling layer, we obtain the representation for each keypoint. We keep the top t keypoints based on their scores and establish our set of keypoints, $K^m = \{(\mathbf{x}_0^m, s_0^m, f_0^m), ..., (\mathbf{x}_t^m, s_t^m, f_t^m)\}$, where $m = \{0, 1\}$ corresponds to the pair of depth images, $\mathbf{x}_t^m = (x_t, y_t)$ are 2D coordinates on the image plane, s_t^m is the score which signifies the saliency level of the keypoint, and f_t^m is the corresponding feature vector.

The sampling layer then receives the sets of keypoint centroids and their features from

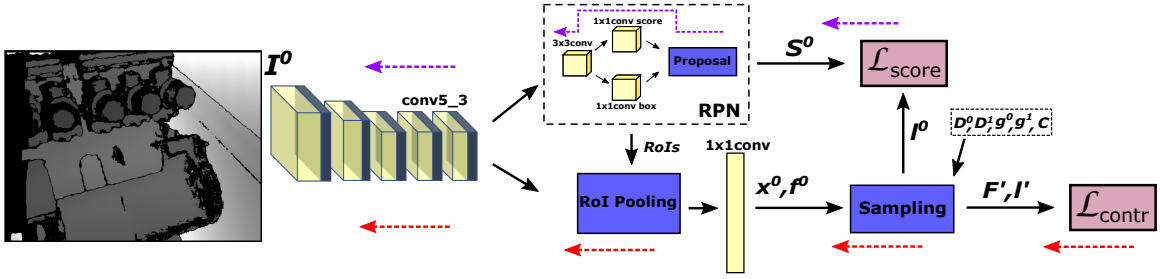


Figure 3.3: Gradient backpropagation during training of our network. The figure only shows one branch. The purple and red arrows show the path of the gradients from the score and the contrastive loss respectively. Notice that no gradients are passed in the 1x1 conv bbox layer, since we are not optimizing towards bounding box regression.

both images, $\{\mathbf{x}^0, \mathbf{x}^1, f^0, f^1\}$. To determine the correspondences between the keypoints of the two images, the centroids are first projected in 3D space. For each keypoint $\mathbf{x}_i^0$, we find the closest $\mathbf{x}_j^1$ in 3D space based on Euclidean distance and form the n^{th} pair of features $F'_n = (f_i^0, f_j^1)$. If the distance is less than a small threshold, we label it as positive ($l'_n = l_i^0 = l_j^1 = 1$), otherwise it is considered as a negative pair ($l'_n = l_i^0 = l_j^1 = 0$). This can possibly lead to one class vastly outnumbering the other. However, this can be advantageous for learning the keypoints, as the number of positive pairs indicates how many keypoints were generated consistently between the two input depth images. This is different from the correspondence layer used in [77], which performed dense sampling of correspondences, and had no notion of keypoints or their repeatability.

Joint Optimization: As mentioned earlier, we are interested in jointly learning a view-invariant representation along with a keypoint detector. Towards this end we introduce the following multi-task loss:

$$L(\{K^0\}, \{K^1\}) = \lambda_c L_c(F', l') + \lambda_s L_s^0(s^0, l^0) + \lambda_s L_s^1(s^1, l^1) \tag{3.1}$$

where, L_c is a slightly modified contrastive loss which operates on the pairs of the keypoints and optimizes over the representation, L_s^m, are the score loss components which use the keypoint scores in order to optimize the detector, l' is the set of labels of the set of feature pairs F', and λ_c and λ_s are the weight parameters. Note that since we formed the features into the set of pairs F', we use the notation n to signify the n^{th} feature pair (f_n^0, f_n^1). The contrastive loss is defined as:

$$L_c(F', l') = \frac{\sum_{n=1}^{N} l'_n ||f_n^0 - f_n^1||^2}{2N_{pos}} + \frac{\sum_{n=1}^{N} (1 - l'_n) max(0, v - ||f_n^0 - f_n^1||)^2}{2N_{neg}} \tag{3.2}$$

where v is the margin, and N_{pos}, N_{neg} are the number of positive and negative pairs respectively ($N = N_{pos} + N_{neg}$). Each class contribution to the loss was normalized based on its population to account for the imbalance between the positive and negative pairs. The score loss is defined as:

$$L_s^m(s^m, l^m) = \frac{1}{1 + N_{pos}} - \frac{\gamma \sum_{i=1}^{N} l_i^m \log s_i^m}{1 + N_{pos}} \tag{3.3}$$

where l_i^m is the label for the i^{th} keypoint from image I^m whose value depends whether the keypoint belongs to a positive or negative pair, and γ is a regularization parameter. Note that since the pairs are formed by picking a keypoint from each image and each keypoint can belong to only one pair, then $|l^0| = |l^1| = N$.

The objective of the score loss is to maximize the number of correspondences between two views. We specifically avoid looking for discriminative keypoints as that would entail defining the meaning of a discriminative keypoint. This is ambiguous by nature as discriminativeness can be subjective, depended also on the task at hand. Instead, we consider

"interesting" keypoints as those for which we can find correspondences between two viewpoints, and ideally we want RPN to rank them higher than others. Therefore, we optimize towards generating as many positive keypoints as we can, in addition to maximizing their scores. We consider only the positive pairs and penalize them if their generated score is low. The loss is normalized by the number of positives, however, γ can be utilized to regulate the trade-off between optimizing for the number of keypoints versus optimizing for the scores.

Furthermore, our framework allows regulating the trade-off between number of matches and localization accuracy during training, by adjusting the 3D distance threshold in the sampling layer. For example, with a small threshold, the model will learn to associate few keypoints with high accuracy as opposed to a large number with a more relaxed threshold. Since we are generating annotations on-the-fly, this enables us to train systems with varying trade-off between matching likelihood and accuracy to address application needs.

During backpropagation, we pass the gradient for each keypoint at the appropriate location in the gradient maps, by storing their locations during the forward pass and implementing the backwards functionality in the region proposal layer. For the score loss, the gradients are passed through the convolutional layers that are responsible for predicting the scores. In contrast to the traditional Faster R-CNN, we do not finetune the bounding box regressor as there are no ground-truth boxes available for our task. However, our training scheme implicitly affects the bounding box generation, as all preceding layers are trained. An illustration of how the gradients are backpropagated for both losses in one branch of our network can be seen in Figure 3.3.

3.3 Experiments

In order to validate our approach, we compare its matching capabilities to hand-crafted features, the keypoint learning method KPL [71], and the state-of-the-art 3DMatch [68] which learns 3D local geometric descriptors using a siamese deep learning architecture. For the hand-crafted features we form 4 baselines from the combinations of the 3D keypoint detectors Harris3D [23] and ISS [78] and the 3D descriptors FPFH [25] and SHOT [79] found

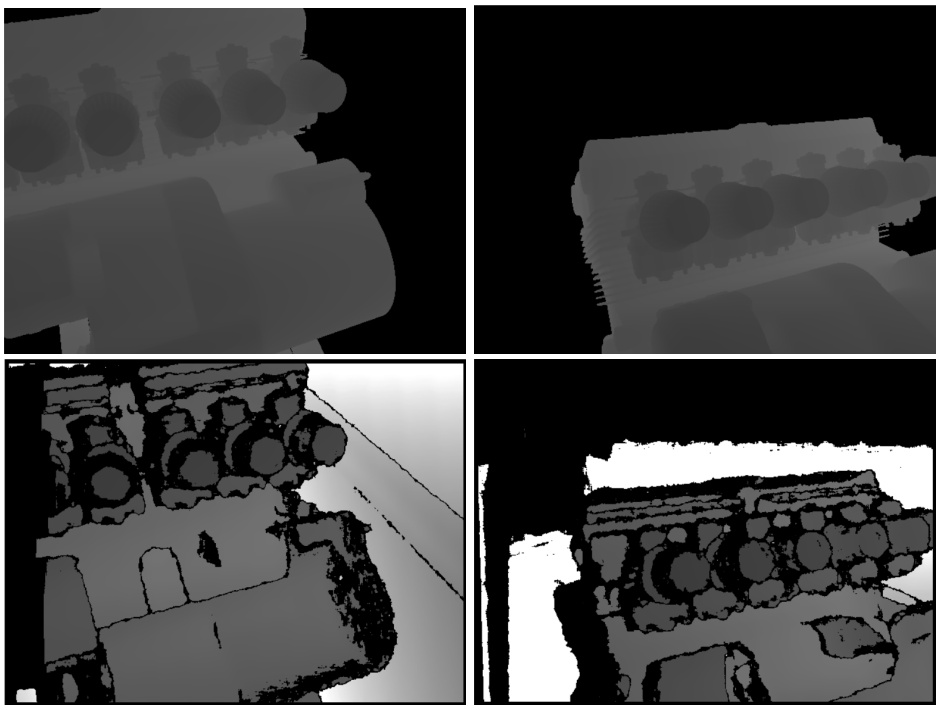


Figure 3.4: A training pair for the *Engine* model shown here both noise-free (top row) and noisy (bottom row) created using DepthSynth [7].

in the Point Cloud Library (PCL) [62]. KPL [71] is a descriptor-specific keypoint learning approach for which we use the provided trained model. We combine it with the SHOT descriptor as it was proposed by the authors of [71]. Similarly, since 3DMatch is a local 3D descriptor, we combine it with Harris3D keypoint detector and use the model trained for keypoint matching provided by the authors. In addition, we add one more baseline which is a variation of our method, where we train using only the contrastive loss. We refer to this baseline as *Ours-No-Score*.

Two main experiments are performed. First, we test on a set of 3D models, both with clean and noisy data, and second, we evaluate on two datasets captured by a real depth sensor. Our motivation for choosing these datasets is to compare the performance of our work with the baselines when dealing with noise from a depth sensor. Other works [80] usually apply Gaussian noise on the 3D models to simulate the noise, however, this does

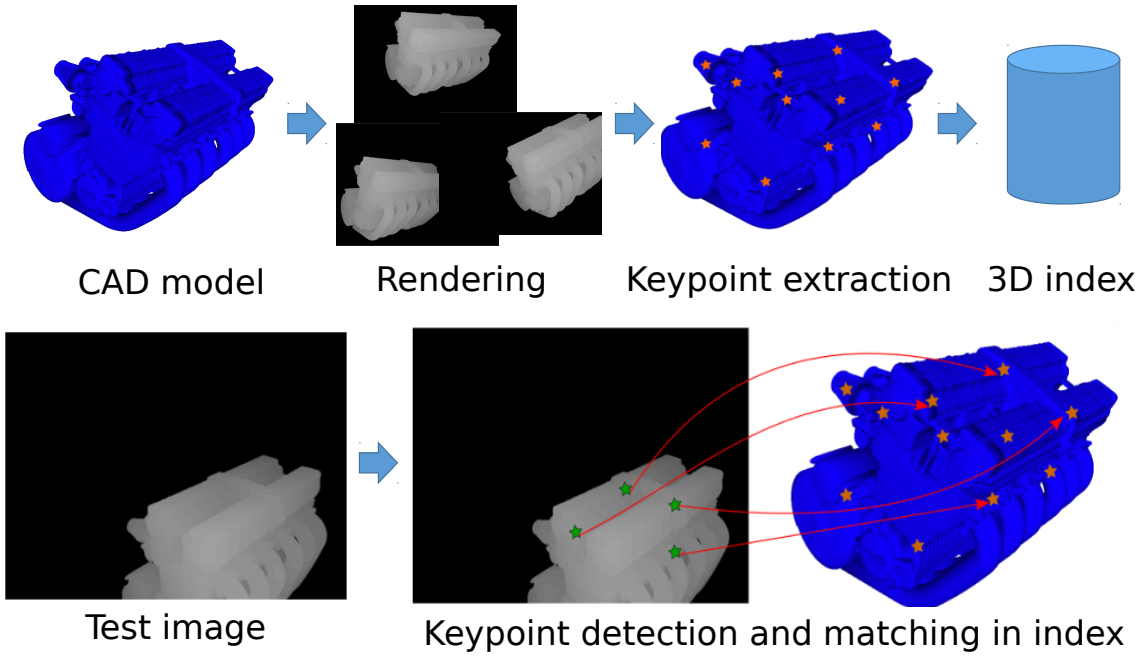


Figure 3.5: Overview of the evaluation pipeline used in our experiments. The top row describes the repository creation, while the bottom shows the test procedure.

not sufficiently represent realistic scenarios. Therefore, for the first experiment, we use DepthSynth [7], which synthetically generates realistic depth data from 3D CAD models by modeling vital factors such as sensor noise, material reflectance, and surface geometry that affect the scanning process. The synthetic noisy images produced by DepthSynth are thus much closer to the real depth images output from structured light depth sensors.

3.3.1 3D Models

For this experiment, we use a set of five 3D models, four taken from the Stanford 3D scanning repository [3] (*Armadillo*, *Bunny*, *Dragon*, *Buddha*) and the Honda CBX1000 engine CAD model, from now on referred to simply as *Engine*. Initially, for each 3D model we randomly generate a large number of noise-free views as rendered from the model. The views are grouped into pairs by first simulating a camera from a certain viewpoint, and then

Table 3.1: Keypoint matching accuracies (%) comparison on both noise-free and noisy views from the *Engine* 3D model.

Method	Noise-Free	Noisy
ISS [78]+SHOT [79]	47.9	0.5
KPL [71]+SHOT [79]	57.2	2.8
ISS [78]+FPFH [25]	61.1	2.9
Harris3D [23]+SHOT [79]	60.1	5.9
Harris3D [23]+FPFH [25]	**79.1**	12.8
Harris3D [23]+3DMatch [68]	66.2	20.7
Ours-Rnd	29.8	7.3
Ours-No-Score	40.7	11.1
Ours-Transfer	-	17.8
Ours	67.4	**23.8**

by adding some pose perturbation in order to generate a pair image with some overlap to the first. We use around 10000 image pairs and sample 50 keypoints per image for training each model. For each view, we add simulated depth sensor noise using DepthSynth [7]. The resulting depth images offer much more challenges as noise is present not only on the parts of the 3D model but on its background as well. An example of a pair of views, both noise-free and noisy, can be seen in Figure 3.4.

Testing protocol. First, separate training and testing sets of views are generated. After we train our model, a subset of the training set (500 views) is used to generate a repository of descriptors, each assigned to a 3D coordinate. Specifically, we pass each view through our model, collect the descriptors at the predicted keypoint locations, and then project those locations in world coordinates. Then, we apply our model on each view from the test set and match the collected descriptors to the repository. For each descriptor, its nearest neighbour is retrieved. When deciding whether this is a true match, we use a small 3D distance threshold (5 cm) on the distance between the 3D location of the descriptor and its retrieval, and increment the number of true matches accordingly. The reported number is the number of true matches towards the total number of matches. An overview of this procedure is shown in Figure 3.5. Note that we do not use any threshold on the descriptor

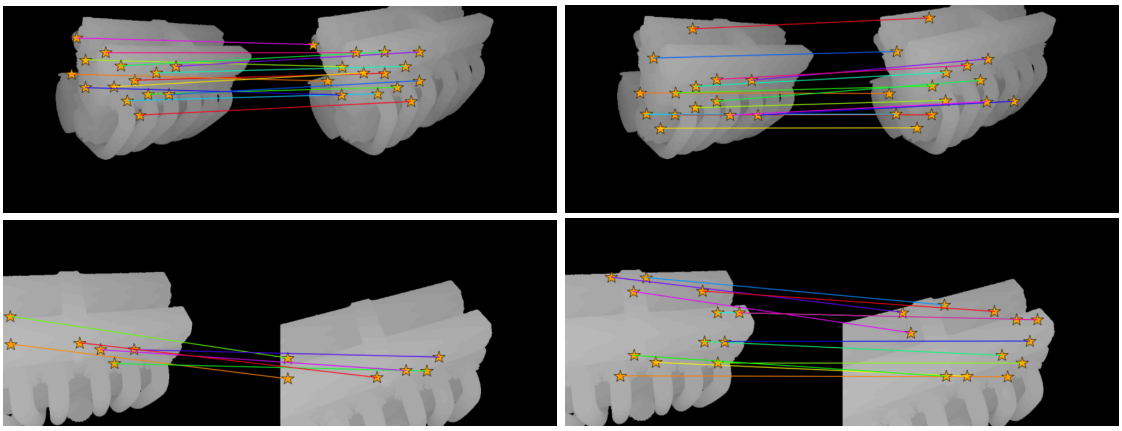


Figure 3.6: Qualitative demonstration of the contribution of the score loss on matching examples on the noise-free views from the *Engine* model. The examples where the model was trained without the score loss (left column) contain smaller number and less accurate matches in comparison to the examples with the model trained with the score loss (right column). Best viewed in color.

distance to obtain the set of matches. The same testing procedure is also used for the baselines. For fairness, we tried to keep roughly the same number of generated keypoints per method and per view.

Engine 3D model. We investigate the performance of the baselines and our approach on both noise-free and noisy views from the *Engine* model. For this particular experiment we add two more baselines, *Ours-Transfer* and *Ours-Rnd*. For *Ours-Transfer* we train a model on noise-free views, and then test it on the noisy data with the purpose of investigating how well our model can transfer between the noise-free and noisy domains. *Ours-Rnd* randomly selects keypoints instead of using those with the highest scores during the testing procedure. Table 3.1 presents the matching accuracies. For the noise-free case, *Ours* is outperformed only from the combination of *Harris3D+FPFH*, which outperforms the deep learning based method of *Harris3D+3DMatch* as well. This is not surprising as these approaches are specifically designed to operate in clean point clouds, however, that is not a realistic setting. Even so, *Ours* demonstrates higher matching accuracy than all the rest of the baselines.

Table 3.2: Keypoint matching accuracies (%) comparison on noisy data from the Stanford 3D models.

Method	Armadillo	Bunny	Dragon	Buddha	Average
ISS [78]+SHOT [79]	0.8	0.5	0.6	0.4	0.6
ISS [78]+FPFH [25]	2.0	1.7	2.4	1.4	1.9
Harris3D [23]+SHOT [79]	8.0	11.4	6.9	6.7	8.3
KPL [71]+SHOT [79]	18.0	12.8	15.4	9.1	13.8
Harris3D [23]+FPFH [25]	14.5	16.0	16.4	10.5	14.4
Harris3D [23]+3DMatch [68]	14.9	17.7	27.8	15.1	18.8
Ours-No-Score	10.0	18.3	25.2	12.5	16.5
Ours	**25.2**	**31.9**	**45.7**	**27.7**	**32.6**

For the noisy case, we first notice a significant drop in performance from all approaches compared to the noise-free evaluation. *Ours* is the best performing approach, with a difference of 3.1%, 6% and 11% towards *Harris3D+3DMatch*, *Ours-Transfer* and *Harris3D+FPFH* respectively. The relatively small gap between *Ours* and *Ours-Transfer* suggests that our model learns to generate good keypoints regardless of the domain it is applied. It is also important to note the large difference between the *Ours-Rnd* and *Ours-No-Score* baselines to *Ours*, which suggests the importance of the score loss during training. Additional qualitative examples are presented in Figure 3.6 to support our argument, where matching examples are compared between the two methods. After visually examining the examples, we notice that *Ours* produces higher quality matches, most likely due to the consistency of the generated keypoints learned by the score loss.

Simulated depth sensor noisy views. Here, we use the 3D models from the Stanford repository and evaluate on their noisy depth images. The testing protocol described earlier is followed, except that we change the 3D distance threshold to 10 cm to account for the errors in the projections of the points. Results shown in Table 3.2 follow the same trend as in the *Engine-noisy* evaluation. *Ours* is the top performing method, outperforming the next-best baselines *Harris3D+3DMatch* by 13.8% and *Harris3D+FPFH* by 18.2%. Both

Table 3.3: Keypoint matching accuracies (%) on the MSR-7 [1] dataset.

Method	Accuracy
ISS [78]+SHOT [79]	23.0
ISS [78]+FPFH [25]	24.3
Harris3D [23]+FPFH [25]	37.4
Harris3D [23]+SHOT [79]	37.9
Harris3D [23]+3DMatch [68]	38.2
Ours	**41.2**

combinations with *ISS* fail to retrieve almost any true matches, as *ISS* seems to be the keypoint detector most affected by the simulated sensor noise. This is a particularly challenging setting for approaches that do not have mechanisms to avoid background noise when generating the keypoints. A qualitative evaluation of the keypoints shown in Figure 3.7 reveals the tendency of the other methods to generate keypoints on noise, while *Ours* focuses on the object. This demonstrates that our method is much less susceptible to the depth sensor noise, and validates our claim for learning the keypoint generation process jointly with the representation.

Computational cost. Our end-to-end method requires only 0.14s per image to perform a forward pass of the network and generate keypoints and descriptors. For comparison, LIFT [73] takes 2.78s per image on the same machine. During training our method needs 0.4s per iteration. All times reported are on a Titan X GPU for the honda engine noisy data with 50 keypoints generated per image.

3.3.2 Real Depth Sensor

MSR-7. For this experiment, we use the publicly available MSR-7 scenes dataset [1], which offers RGB-D sequences captured with Kinect and reconstructions of indoor scenes. We followed the train-test sequence split provided, and trained a model for each scene on 10-frame-apart pairs of the depth images. The same testing protocol of keypoint matching as in the previous experiments is employed. We do not use the baseline *Ours-No-Score* as

it consistently underperformed in the previous experiments, nor the *KPL* because it was trained on a very different dataset.

Table 3.3 shows the average matching accuracy over all scenes. Again, our method seems to have the edge over the baselines, with a 3% improvement over the second-best *Harris3D+3DMatch*. This result suggests that our approach can be successfully applied on sequences captured by a real sensor, besides 3D models with simulated noise. In Figure 3.8 we present some retrieval examples from different scenes in the dataset. We make a similar observation as in the noise-free experiment, where true matches were retrieved from larger viewpoint variations than the ones provided during training. Note that the training pairs, 10-frames-apart, have small pose differences.

GMU-Kitchens In this experiment we qualitatively investigate the performance of our method on objects captured by Kinect-v2. We use the publicly available GMU-Kitchens [2] dataset which contains 9 RGB-D videos of kitchen scenes with 11 object instances from the BigBIRD [39] dataset. Unlike the previous experiment where we generate keypoints in the scenes, here we focus on matching keypoints generated inside the bounding boxes of objects. In particular, we use the train-test split of fold 1 as defined in [2], and for each object we create a repository of descriptors. Then, the descriptors collected from the bounding boxes in the test scenes are matched to the appropriate object repository (see Figure 3.9). Note that the model was trained by sampling keypoints from the depth maps in the scenes, similar to the MSR-7 experiment, and not specifically from the object bounding boxes.

3.4 Conclusions

We presented a unified, end-to-end, framework to simultaneously learn a keypoint detector and view-invariant representations of keypoints for 3D keypoint matching. To learn view-invariant representations, we presented a novel sampling layer that creates ground-truth data on-the-fly, generating pairs of keypoint proposals that we use to optimize a constrastive loss objective function. Furthermore, to learn to generate the right keypoint proposals from a keypoint matching perspective, we introduced a new score loss objective

that maximizes the number of positive matches between images from two viewpoints. We conducted keypoint matching experiments on multiple 3D benchmark datasets and demonstrated qualitative and quantitative improvements over the existing state-of-the-art.

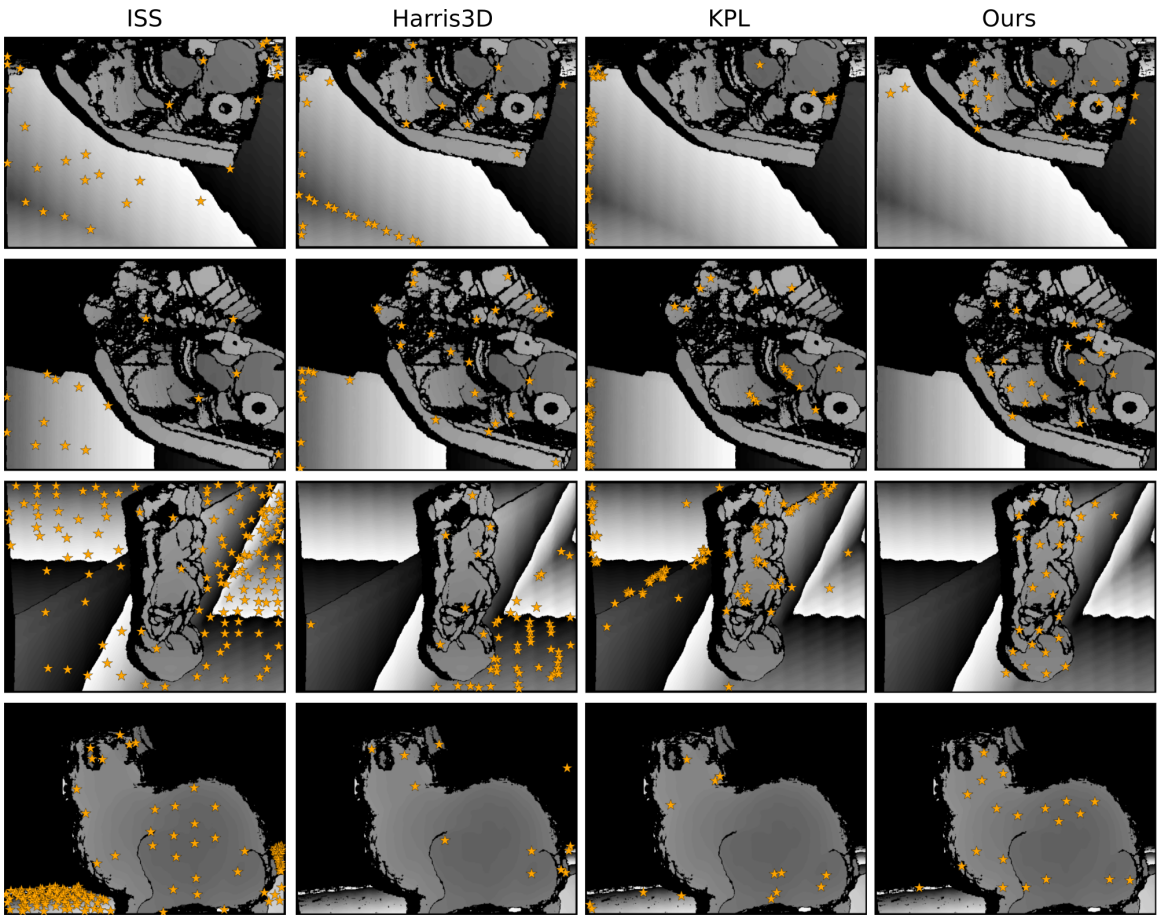


Figure 3.7: Qualitative evaluation of keypoint generation on the noisy views. Each column represents a different approach. From left to right we have ISS, Harris3D, KPL, and Ours. Notice that the first three methods frequently generate keypoints on background noise, in contrast to our method which generates keypoints mostly on the object. Best viewed in color.

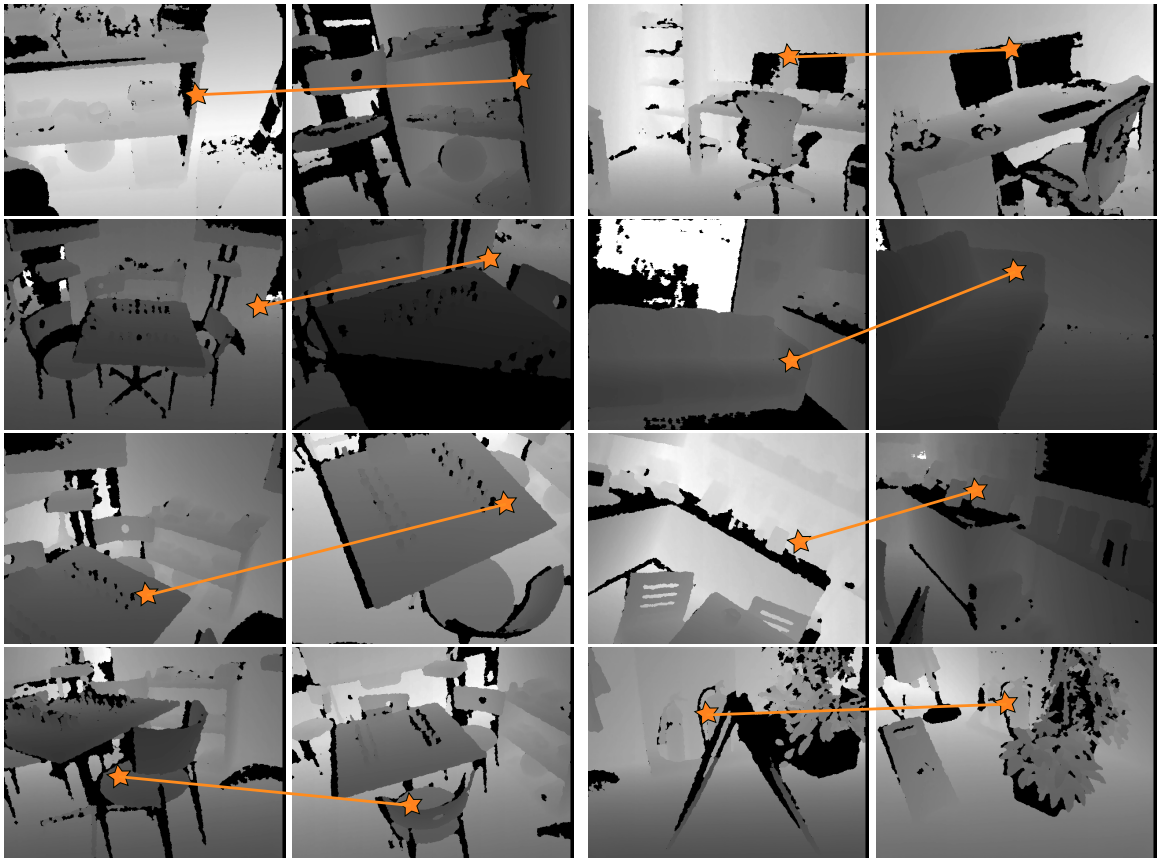


Figure 3.8: Keypoint matching examples on the MSR-7 scenes. Columns 1 and 3 show test images and columns 2 and 4 show their retrievals from the repository of descriptors.

Figure 3.9: Matching examples from GMU-Kitchens. First column shows queries and retrieved points are color-coded (zoomed-in for clarity). Note that we use the depth map for our experiments but we show the retrievals in RGB for the sake of clarity.

Chapter 4: Learning Local RGB-to-CAD Correspondences for Object Pose Estimation

Estimating the 3D pose of objects is an important capability for enabling robots' interaction with real environments and objects as well as augmented reality applications. While several approaches to this problem assume RGB-D data [81,82], most mobile and wearable cameras are not paired with a depth sensor, prompting recent research focus on the RGB domain. Furthermore, even though several methods have shown promising results on 3D object pose estimation with real RGB images, they either require accurate 3D annotations [14,29–31,83] or 3D object models with realistic textures [81, 84–86] in the training stage. Currently available datasets [87, 88] are not large enough to capture real world diversity, limiting the potential of these methods in generalizing to a variety of applications. In addition, capturing real RGB data and manual pose annotation is an arduous procedure.

The problem of object pose estimation is an inherently 3D problem; it is the shape of the object which gives away its pose regardless of its appearance. Instead of attempting to learn an intrinsic decomposition of images [89], we focus on finding the association of parts of objects depicted in RGB images with their counterparts in 3D depth images. Ideally, we would like to learn this association in order to establish correspondences between a query RGB image and a rendered depth image from a CAD model, without requiring any existing 3D annotations. This, however, requires us to address the problem of the large appearance gap between these two modalities.

In this chapter, we propose a new framework for estimating the 3D pose of objects in RGB images, using only 3D textureless CAD models of objects instances. The easily available CAD models can generate a large number of synthetically rendered depth images from multiple viewpoints. In order to address the aforementioned problems, we define

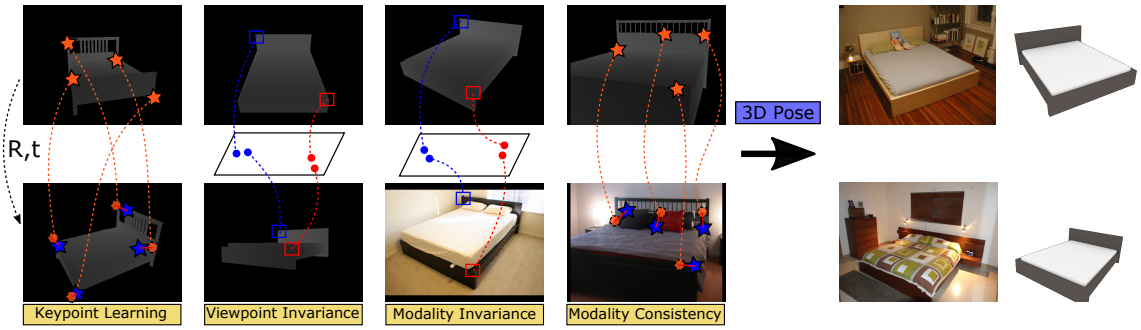


Figure 4.1: We present a new method that matches RGB images to depth renderings of CAD models for object pose estimation. It does not require either textured CAD models or 3D pose annotations for RGB images during training. This is achieved by enforcing viewpoint and modality invariance for local features, and learning consistent keypoint selection across modalities.

a *quadruplet* convolutional neural network to jointly learn keypoints and their associated descriptors for robust matching between different modalities and changes in viewpoint. The general idea is to learn the keypoint locations using a pair of rendered depth images from a CAD model from two different poses, followed by learning how to match keypoints across modalities using an aligned RGB-D image pair. Figure 5.1 outlines our training constraints. At test time, given a query RGB image, we extract keypoints and their representations and match them with a database of keypoints and their associated descriptors extracted from rendered depth images. These are used to establish 2D-3D correspondences, followed by a RANSAC and PnP algorithm for pose estimation.

To summarize, our key contributions include: **1)** A new framework for 3D object pose estimation using only textureless CAD models and aligned RGB-D frames in the training stage, without explicitly requiring 3D pose annotations for the RGB images. **2)** An end-to-end learning approach for keypoint selection optimized for the relative pose estimation objective, and transfer of keypoint predictions and their representations from rendered depth to RGB images. **3)** Demonstration of the generalization capability of our method to new (unseen during training) instances of the same object category.

4.1 Related Work

There is a large body of work on 3D object pose estimation. Here, we review existing methods based on the type and the amount of used training data and its modalities.

Using 3D textured instance models. Notable effort was devoted to the problem of pose estimation for object instances from images, where 3D textured instance models were available during the training stage [84, 85, 90]. Early isolated approaches led to the development of more recent benchmarks for this problem [91]. Traditional approaches of this type included template matching [90, 92], where the target pose is retrieved from the best matched model in a database, and local descriptor matching [84, 85], where hand-engineered descriptors such as SIFT [24] are used to establish 2D-3D correspondences with a 3D object model followed by the PnP algorithm for 6-DoF pose. Additionally, some works employed a patch-based dense voting scheme [81, 86, 93, 94], where a function is learned to map local representations to 3D coordinates or to pose space. However, these approaches assume that the 3D object models were created from real images and contain realistic textures. In contrast, our work uses only textureless CAD models of object instances.

2D-to-3D alignment with CAD models. Other work has sought to solve 3D object pose estimation as a 2D-to-3D alignment problem by utilizing object CAD models [82, 87, 95–98]. For example, Aubry *et al.* [95] learned part-based exemplar classifiers from textured CAD models and applied them on real images to establish 2D-3D correspondences. In a similar fashion, Lim *et al.* [87] trained a patch detector from edge maps for each interest point. The work of Massa *et al.* [96] learned how to match view-dependent exemplar features by adapting the representations extracted from real images to their CAD model counterparts. The closest work to ours in this area is Rad *et al.* [82], which attempts to bridge the domain gap between real and synthetic depth images, by learning to map color features to real depth features and subsequently to synthetic depth features. In their attempt to bridge the gap between the two modalities, these approaches were required to either learn a huge number of exemplar classifiers, or learn how to adapt features for each specific category and

viewpoint. We avoid this problem by simply adapting keypoint predictions and descriptors between the two modalities.

Pose estimation paired with object detection. With the recent success of deep convolutional neural networks (CNN) on object recognition and detection, many works extended 3D object instance pose estimation to object categories, from an input RGB image [12, 14, 29–31, 83, 99–101]. In Mahendran *et al.* [30] a 3D pose regressor was learned for each object category. In Mousavian *et al.* [14], a discrete-continuous formulation for the pose prediction was introduced, which first classified the orientation to a discrete set of bins and then regressed the exact angle within the bin. Poirson *et al.* [31] and Kehl *et al.* [83] both extended the SSD [11] object detector to predict azimuth and elevation or the 6-DoF pose respectively. In Kundu *et al.* [12], an analysis-by-synthesis approach was introduced, in which, given predicted pose and shape, the object was rendered and compared to 2D instance segmentation annotations. All of these approaches require 3D pose annotations for the RGB images during training, as opposed to our work, which only needs the CAD models of the objects.

Keypoint-based methods. Another popular direction in the pose estimation literature is learning how to estimate keypoints, which can be used to infer the pose. These methods are usually motivated by the presence of occlusions [102, 103] and require keypoint annotations. For example, Wu *et al.* [104] trained a model for 2D keypoint prediction on real images and estimated the 3D wireframes of objects using a model trained on synthetic shapes. The 3D wireframe is then projected to real images labeled with 2D keypoints to enforce consistency. In Li *et al.* [105], the authors manually annotated 3D keypoints on textured CAD models and generated a synthetic dataset which provides multiple layers of supervision during training, while Tekin *et al.* [106] learned to predict the 2D image locations of the projected vertices of an object's 3D bounding box before using the PnP algorithm for pose estimation. Furthermore, Tulsiani *et al.* [13] exploited the relationship between viewpoint and visible keypoints and refined an existing coarse pose estimation using keypoint predictions. Our work, rather than relying on existing keypoint annotations, optimizes the keypoint selection

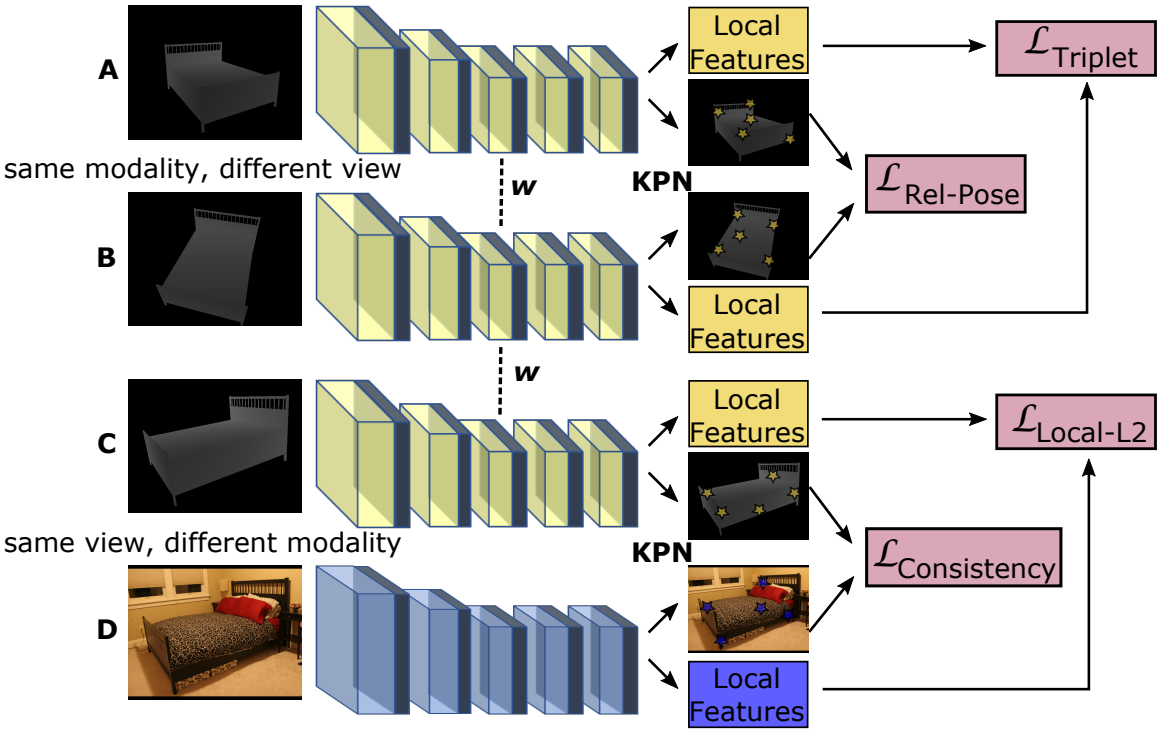


Figure 4.2: Outline of the proposed architecture depicting the four branches, their inputs, and the training objectives. The color coding of the CNNs signifies weight sharing.

based on a relative pose estimation objective. Related approaches also learn keypoints [26, 28, 107, 108], but either rely on hand-crafted detectors to collect training data [26], or do not extend to real RGB pose estimation [28, 107, 108].

Synthetic data generation. In an attempt to address the scarcity of annotated data, some approaches rely on the generation of large amounts of synthetic data for training [4, 109, 110]. A common technique is to render textured CAD models and superimpose them on real backgrounds. In order to ensure diversity in the training data, rendering parameters such as pose, shape deformations, and illumination are randomly chosen. However, training exclusively on synthetic data has shown to be detrimental to the learned representations as the underlying statistics of real RGB images are usually very different.

4.2 Approach

We are interested in estimating the 3D pose of objects in RGB images by matching keypoints to the object's CAD model. Our work does not make use of pose annotations, but instead relies on CAD model renderings of different poses that are easily obtained with an off-the-shelf renderer, such as Blender [111]. These rendered depth images are used to learn keypoints and their representations optimized for the task of pose estimation. The learned representations are then transferred to the RGB domain. In summary, our work can be divided into four objectives: keypoint learning, view-invariant descriptors, modality-invariant descriptors, and modality consistent keypoints.

Specifically, each training input is provided as a quadruplet of images, consisting of a pair of rendered depth images sampled from the object's view sphere and a pair of aligned depth and RGB images (see Figure 4.2). For each image, we predict a set of keypoints and their local representations, but the optimization objectives differ for the various branches. For the first two branches A and B, L_{rel_pose} loss enforces the pose consistency of the keypoints selection and the similarity of keypoint descriptors for their matching is enforced using a triplet loss $L_{triplet}$. The two bottom branches C and D are utilized to enforce consistent keypoint prediction between the depth and the RGB modalities $L_{consistency}$ and for matching their local representations across the modalities L_{local_l2}. The general idea of our approach is to learn informative keypoints and their associated local descriptors from abundant rendered depth images and transfer this knowledge to the RGB data.

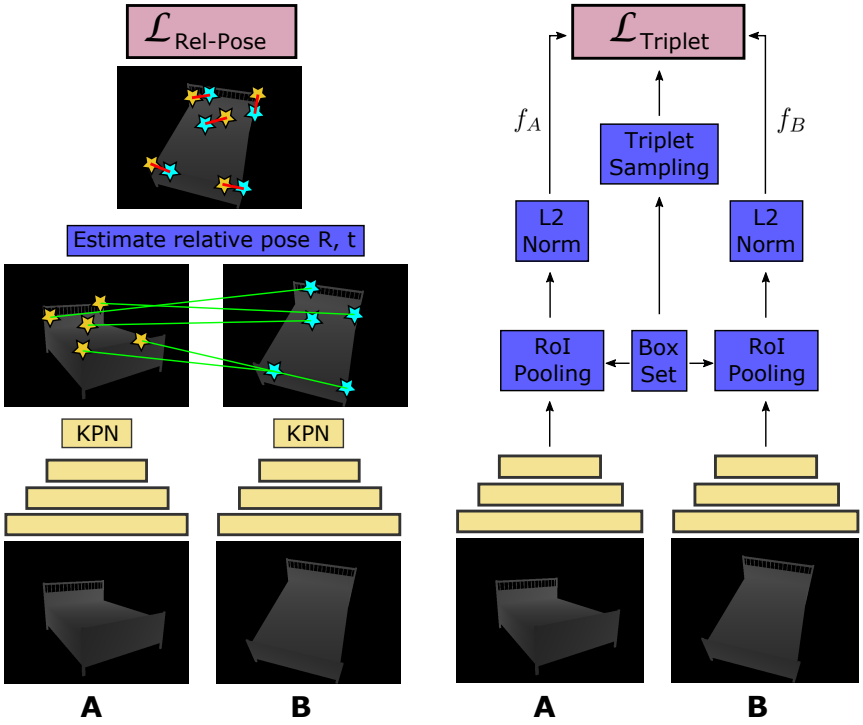
Architecture. Our proposed architecture is a Quadruplet convolutional neural network (CNN), where each branch has a backbone CNN (e.g., VGG) to learn feature representations and a keypoint proposal network (KPN) comprised of two convolutional layers. The output feature maps from the backbone's last convolutional layer are fed as input to the KPN. KPN produces a score map of dimensions $\frac{H}{s} \times \frac{W}{s} \times D$, where H and W are the input image's height and width respectively, s is the network stride, and $D = 2$ is a score whether the particular location is a keypoint or not. Softmax is then applied on D such that each

location on the KPN output map has a 2-D probability distribution. This output map can be seen as a keypoint confidence score for a grid-based set of keypoint locations over the 2D image. The density of the keypoint sampling depends on the network stride s, which in our case was 16 (i.e. a keypoint proposal every 16 pixels). In order to extract a descriptor (dim-2048) for each keypoint, the backbone's feature maps are passed to the region-of-interest (RoI) pooling layer along with a set of bounding boxes each centered at a keypoint location. The first pair of branches (A, B) of the network are trained with a triplet loss applied to local features, while a relative pose loss is applied to the keypoint predictions. Branch D is trained using a Euclidean loss on the local features and with a consistency loss that attempts to align its keypoint predictions and local representations to those of branch C. Note that branches A, B, and C share their weights, while branch D is a different network. Since branch D receives as input a different modality than the rest and we desire branches C and D to produce the same outputs, their weights during training must be independent. In the following sections, we describe the details of the loss functions and training.

4.2.1 Keypoint Learning by Relative Pose Estimation

The overall idea behind learning keypoint predictions is to select keypoints that can be used for relative pose estimation between the input depth images in branches A and B. Specifically, given the two sets of keypoints, we establish correspondences in 3D space, estimate the rotation R and translation t, and project the keypoints from depth image A to depth image B. Any misalignment (re-projection error) between the projected keypoints is used to penalize the initial keypoint selections. A pictorial representation of the relative pose objective is shown in Figure 4.3a.

The relative pose objective is formulated as a least squares problem, which finds the rotation R and translation t for which the error of the weighted correspondences is minimal. Formally, for two sets of corresponding points: $P = \{p_1, p_2, \ldots, p_n\}, Q = \{q_1, q_2, \ldots, q_n\}$



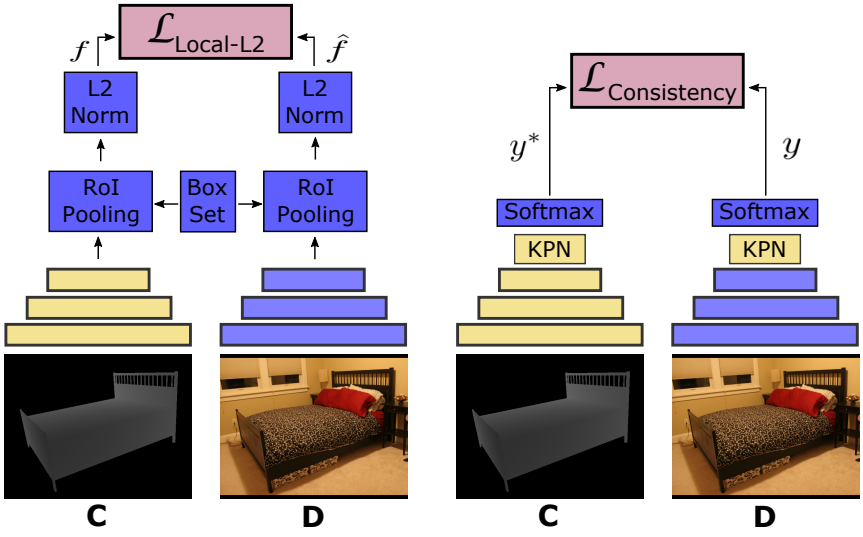
(a) Relative pose loss. (b) Triplet loss.

Figure 4.3: Relative pose and triplet losses.

we wish to estimate R and t such that:

$$(R,t) = \underset{R\in SO(3),t\in\mathbb{R}^3}{\arg\min} \sum_{i=1}^{n} w_i ||(Rp_i + t) - q_i||^2 \tag{4.1}$$

where $w_i = s_i^A + s_i^B$ is the weight of correspondence i and s_i^A and s_i^B are the predicted keypoint probabilities, as given by KPN followed by a Softmax layer, that belong to correspondence i from branches A and B respectively. Given a set of correspondences and their weights, an SVD-based closed-form solution for estimating R and t that depends on w can be found in [112]. The idea behind this formulation is that correspondences with high re-projection error should have low weights, therefore a low predicted keypoint score, while correspondences with low re-projection error should have high weights, therefore high



(a) Local Euclidean loss. (b) Keypoint consistency loss.

Figure 4.4: Local Euclidean and keypoint consistency losses.

predicted keypoint score. With this intuition, we formulate the relative pose loss as:

$$L_{rel\_pose} = \frac{1}{n}\sum_{i=1}^{n} w_i g(w_i) \tag{4.2}$$

where $g(w_i) = ||(Rp_i + t) - q_i||^2$. Since our objective is to optimize the loss function with respect to estimated keypoint scores, we penalize each keypoint score separately by estimating the gradients for each correspondence and backpropagating them accordingly.

4.2.2 Learning Keypoint Descriptors

In order to match keypoint descriptors across viewpoints, we apply a triplet loss on local features extracted from branches A and B. This involves using the known camera poses of the rendered pairs of depth images and sampling of training keypoint triplets (anchor-positive-negative). Specifically, for a randomly selected keypoint as an anchor from the first image, we find the closest keypoint in 3D from the paired image and use it as a positive,

and also select a further away point in 3D to serve as the negative. The triplet loss then optimizes the representation such that the feature distance between the anchor and the positive points is smaller than the feature distance between the anchor and the negative points plus a certain margin, and is defined as follows:

$$L_{triplet} = \frac{1}{N}\sum_{i}^{N} \max(0, ||f_i^a - f_i^p||^2 - ||f_i^a - f_i^n||^2 + m) \tag{4.3}$$

where f_i^a, f_i^p, and f_i^n are the local features for the anchor, positive, and negative correspondingly of the i^{th} triplet example and m is the margin. Traditionally, the margin hyperparameter is manually defined as a constant throughout the training procedure; however, we take advantage of the 3D information and define the margin to be equal to $D_n - D_p$, where D_n is the 3D distance between the anchor and negative, and D_p is the 3D distance between the anchor and positive. Ideally, D_p should be 0, but practically due to the sampling of the keypoints in the image space it is usually a small number close to 0. Essentially this ensures that the learned feature distances are proportional to the 3D distances between the examples and assumes that the features and 3D coordinates are normalized to unit vectors. Note that the triplet loss only affects the backbone CNN during training and not the KPN. A pictorial representation of the triplet objective is shown in Figure 4.3b.

4.2.3 Cross-modality Representation Learning

Finally, we can transfer the learned features and keypoint proposals from branches (A, B) to branch D, using branch C as a bridge, similar to knowledge distillation techniques [113]. To accomplish this, network parameters in branches A, B, and C are shared, and the outputs of branches C and D are compared and penalized according to any misalignment. The core idea is to enforce both the backbone and KPN in branches C and D to generate as similar outputs as possible. This objective can be accomplished by means of two key components that are described next.

Local Feature Alignment. In order to align local feature representations in branches C and D (see Figure 4.4a), we consider the predicted keypoints in branch C and compute each keypoint's feature representation, $f_i, i = 1, \ldots, k$. Keypoint features at corresponding spatial locations from branch D are represented as $\hat{f}_i, i = 1, \ldots, k,$. Formally, we optimize the following objective function:

$$L_{local\_l2} = \frac{1}{k} \sum_{i=1}^{k} \|\hat{f}_i - f_i\| \tag{4.4}$$

Since we want to align $\hat{f}_i$ with f_i, during backpropagation, we fix f_i as ground-truth and backpropagate gradients of L_{local_l2} only to the appropriate locations in branch D.

Keypoint Consistency. Enforcement of the keypoint consistency constraint requires the KPN from branch D to produce the same keypoint predictions as the KPN from branch C. It can be achieved using a cross-entropy loss, which is equivalent to a log loss with binary labels: $L = -\frac{1}{n} \sum_{i=1}^{n} y_i^* \log y_i$, where y_i^* is the ground-truth label and y_i is the prediction. This in our case becomes:

$$L_{consistency} = -\frac{1}{n} \sum_{i=1}^{n} y_i^C \log y_i^D \tag{4.5}$$

where y_i^C are the keypoint predictions from branch C, which serve as the ground-truth, and y_i^D are the keypoint predictions from branch D. This loss penalizes any misalignment between the keypoint predictions of the two branches and forces branch D to imitate the outputs of branch C. Figure 4.4b illustrates inputs to $L_{consistency}$.

Overall objective. Our overall training objective is the combination of the losses described above:

$$L_{all} = \lambda_1 L_{triplet} + \lambda_2 L_{rel\_pose} + \lambda_3 L_{local\_l2} + \lambda_4 L_{consistency} \tag{4.6}$$

where each λ is the weight for the corresponding loss.

4.3 Experiments

In order to validate our approach, we perform experiments on the Pascal3D+ [88] dataset and the newly introduced Pix3D [4] dataset.We conduct four key experiments. First, we compare to supervised state-of-the-art methods by training on Pix3D and testing on Pascal3D+ (sec. 4.3.1); second, we perform an ablation study on Pix3D and evaluate the performance of different parts of our approach (sec. 4.3.2); third, we test how our model generalizes to new object instances by training only on a subset of provided instances and testing on unseen ones (sec. 4.3.3); and finally, data from an external dataset, such as NYUv2 [54] is used to train and test on Pix3D (sec. 4.3.4). The motivation for the fourth experiment is to demonstrate that our framework can utilize RGB-D pairs from another realistic dataset, where the alignment between the RGB and the depth is provided by the sensor. We use the geodesic distance for evaluation: $\Delta(R_1, R_2) = \frac{||\log(R_1^T R_2)||_F}{\sqrt{2}}$, reporting percentage of predictions within $\frac{\pi}{6}$ of the ground-truth $Acc_{\frac{\pi}{6}}$ and $MedErr$. Additionally, we show the individual accuracy of the three Euler angles, where the distance is the smallest difference between two angles: $\Delta(\theta_1, \theta_2) = \min(2\pi - ||\theta_1 - \theta_2||, ||\theta_1 - \theta_2||)$. For the last metric we also use a threshold of $\frac{\pi}{6}$.

Implementation details. We use VGGNet as each branch's backbone and start from ImageNet pretrained weights, while KPN is trained from scratch. We set the learning rate to 0.001 and all λ weights to 1. In order to regularize the relative pose loss such that it

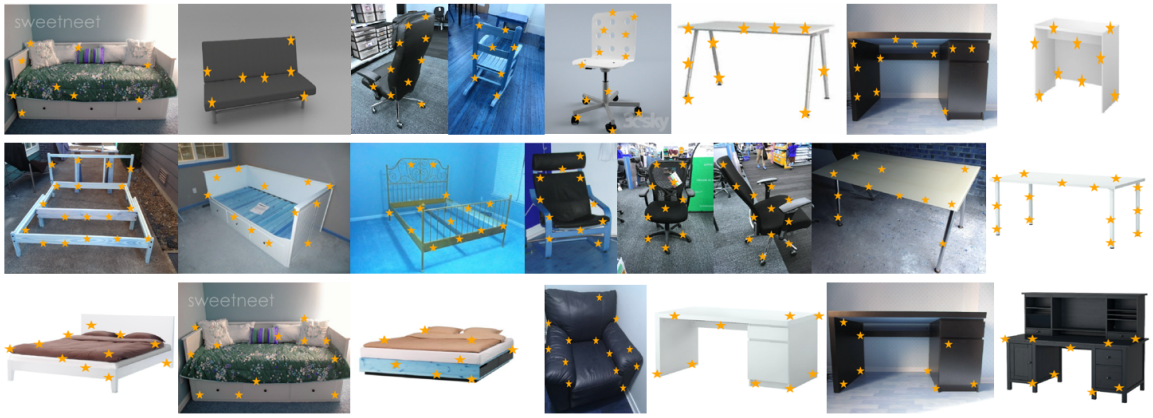


Figure 4.5: Keypoint prediction examples on test images from the Pix3D dataset. Top, middle, and bottom rows show results from experiments of sections 4.3.2, 4.3.3, and 4.3.4 respectively. Note that we applied non-maximum suppression (NMS) on the keypoint predictions in order to select the highest scoring keypoint from each region.

predicts keypoints inside objects, we add a mask term, realized as a multinomial logistic loss. The ground-truth is a binary mask of the object in the rendered depth. This loss is only applied on branches A and B with a smaller weight of 0.25. Finally, the bounding box dimensions for the RoI layer are set to 32×32.

Training data. All our experiments require a set of quadruplet inputs. For the first two inputs, we first sample from each object's viewsphere and render a view every 15 degrees in azimuth and elevation for three different distances. Then, we sample rendered pairs such that their pose difference is between $\frac{\pi}{12}$ and $\frac{\pi}{3}$. For the last two inputs, we require aligned depth and RGB image pair. In order to demonstrate our approach on the Pix3D dataset, we generate these alignments using the dataset's annotations, however, we do not use annotations during training in any other capacity. As we show in sec. 4.3.4, alternatively the aligned depth and RGB images can be sampled from an existing RGB-D dataset or through hand-alignment [98]. Note that for each quadruplet, the selection of the first pair of inputs is agnostic to the pose of the object in the last two inputs. We further note that, given sufficient viewsphere sampling, what is important is how the quadruplet training data is generated (particularly pairs for branches A & B). If the pairs have a small pose difference

(e.g., $\leq \frac{\pi}{12}$), the model does not adequately learn view-invariant representations. On the other hand, with larger pose differences (e.g., $\frac{\pi}{2}$), overlapping areas between the two views are small, so finding correspondences across views is harder. We found sampling pairs with a maximum pose difference of $\frac{\pi}{3}$ provides a good balance. A possible future extension can be to incorporate "interesting" viewpoints [114], which are typically task-dependent, into our pipeline for further improvements (e.g., reduced data requirements or training time).

Testing protocol. For every CAD model instance used in our experiments, we first create a repository of descriptors each assigned to a 3D coordinate. To do so, 20 rendered views are sampled from the viewing sphere of each object, similarly to how the training data are generated, and keypoints are extracted from each view. Note that for this procedure, we use the trained network that corresponds to branch A of our architecture. Then we pass a query RGB image through the network of branch D, generate keypoints and their descriptors and match them to the repository of the corresponding object instance. Finally, the established correspondences are passed to RANSAC and PnP algorithm to estimate the pose of the object. For every keypoint generation step we use the keypoints with the top 100 scores during database creation and top 200 scores for the testing RGB images. When testing on Pix3D, we have defined a test set which contains untruncated and unoccluded examples of all category instances, with 179, 1451, and 152 images in total for *bed*, *chair*, and *desk* category respectively. For Pascal3D+ we follow the provided test sets and make use of the ground-truth bounding boxes.

4.3.1 Comparison with supervised approaches

Given our approach does not use any pose annotations during training, it is challenging to evaluate it against existing state-of-the-art methods, which use pose annotations during training. In addition, our method cannot be trained on Pascal3D+ because it requires paired RGB and depth images, which cannot be generated from the dataset's annotations. Therefore, we designed the following experiment for a fair comparison: we train all methods on Pix3D and test on Pascal3D+. We compare to the state-of-the-art methods of

Table 4.1: Comparison with supervised approaches when trained on Pix3D and tested on Pascal3D+ on $Acc_{\frac{\pi}{6}}$ (%) and $MedErr$ (radians).

Category	Chair		Sofa	
Metric	$Acc_{\frac{\pi}{6}} \uparrow$	$MedErr \downarrow$	$Acc_{\frac{\pi}{6}} \uparrow$	$MedErr \downarrow$
Render for CNN [109]	4.3	2.1	11.6	1.2
Vps & Kps [13]	10.3	1.7	23.3	1.2
Deep3DBox [14]	10.8	1.9	25.6	**1.0**
Proposed	**13.4**	**1.6**	**30.2**	1.1

Deep3DBox [14], Render for CNN [109], and Viewpoints & Keypoints [13], all of which require pose annotations for RGB images. Other approaches, such as Pavlakos *et al.* [102], were considered for comparison but unfortunately they require semantic keypoint annotations during training which Pix3D does not provide. We conduct this evaluation on the common categories between Pix3D and Pascal3D+ (*chair* and *sofa*) and report results in Table 4.1.

As expected, all approaches generally underperform when applied on a new dataset. Our method demonstrates better generalization and achieves higher $Acc_{\frac{\pi}{6}}$ for both objects, even though it does not explicitly require 3D pose annotations during training. This is due to fundamental conceptual differences between these approaches and ours. These methods formulate viewpoint estimation as a classification problem where a large number of parameters in fully-connected layers are to be learned. This increases the demand for data and annotations and confines the methods mostly to data distributions that were trained on. On the other hand, we exploit CAD models to densely sample from the object's viewsphere, and explicitly bridge the gap between the synthetic data and real images, thereby reducing the demand for annotations. Furthermore, the learned local correspondences allow more flexibility in understanding the geometry of unseen objects, as we also show in sec. 4.3.3.

Table 4.2: Results for $Acc_{\frac{\pi}{6}}$ (%) and $MedErr$ (radians) for the sec 4.3.2 experiment.

Category	Bed		Chair		Desk	
Metric	$Acc_{\frac{\pi}{6}} \uparrow$	$MedErr \downarrow$	$Acc_{\frac{\pi}{6}} \uparrow$	$MedErr \downarrow$	$Acc_{\frac{\pi}{6}} \uparrow$	$MedErr \downarrow$
Baseline-A	7.3	1.7	3.3	2.0	2.6	2.2
Baseline-ZDDA	21.8	1.5	11.5	1.7	3.9	2.0
Proposed - joint	31.3	1.0	31.1	**0.9**	25.0	1.1
Proposed - alternate	**50.8**	**0.5**	**31.2**	1.0	**34.9**	**0.9**

Table 4.3: Results for azimuth, elevation, and in-plane rotation accuracy for the sec 4.3.2 experiment.

Category	Bed			Chair			Desk		
Metric	Az.	El.	Pl.	Az.	El.	Pl.	Az.	El.	Pl.
Baseline-A	51.4	39.1	35.2	30.2	43.2	20.0	28.9	30.9	20.4
Baseline-ZDDA	48.6	50.3	41.9	35.3	48.3	26.6	24.3	23.7	21.1
Proposed - joint	69.8	51.9	58.1	**55.3**	**62.7**	44.7	57.2	48.7	51.0
Proposed - alternate	**83.2**	**67.0**	**70.4**	54.7	60.1	**47.0**	**65.1**	**55.3**	**58.6**

Figure 4.6: Illustration of rendered estimated poses on test RGB images from the Pix3D dataset for the sec. 4.3.2 experiment.

4.3.2 Ablation study

To understand each objective's contribution, we have carefully designed a set of baselines, which we train and test on Pix3D, and compare them on the task of pose estimation for the *bed*, *chair*, and *desk* categories.

Table 4.4: Results for $Acc_{\frac{\pi}{6}}$ (%) and $MedErr$ (radians) for the sec. 4.3.3 experiment.

Category	Bed		Chair		Desk	
Metric	$Acc_{\frac{\pi}{6}} \uparrow$	$MedErr \downarrow$	$Acc_{\frac{\pi}{6}} \uparrow$	$MedErr \downarrow$	$Acc_{\frac{\pi}{6}} \uparrow$	$MedErr \downarrow$
Baseline-A	9.7	1.9	3.7	1.9	5.6	2.0
Baseline-ZDDA	4.9	2.3	7.6	1.9	13.6	1.7
Proposed - joint	29.2	0.9	15.1	1.4	13.6	1.3
Proposed - alternate	**45.1**	**0.6**	**21.2**	**1.2**	**18.4**	**1.2**

Table 4.5: Results for azimuth (%), elevation (%), and in-plane rotation (%) accuracy for the sec. 4.3.3 experiment.

Category	Bed			Chair			Desk		
Metric	Az.	El.	Pl.	Az.	El.	Pl.	Az.	El.	Pl.
Baseline-A	38.2	39.6	30.6	28.6	41.4	20.3	37.6	34.4	28.8
Baseline-ZDDA	29.9	39.6	22.2	30.1	44.6	21.5	36.8	43.2	30.4
Proposed - joint	66.7	50.0	62.5	43.7	50.4	31.3	59.2	44.0	41.6
Proposed - alternate	**75.7**	**61.1**	**74.3**	**52.0**	**57.4**	**38.0**	**62.4**	**44.0**	**53.6**

Baseline-A. In order to assess the importance of the cross-modality representation learning (sec. 4.2.3), we learn view-invariant depth representations and depth keypoints and simply use these keypoints and representations during testing. In practice, this corresponds to removing the local euclidean and keypoint consistency losses, and using only the triplet and relative pose losses during training. Consequently this baseline is utilizing only depth data during training, but is applied on RGB images during testing.

Baseline-ZDDA. Another baseline would be to only learn RGB-D modality invariant representations, i.e., similar features for RGB and depth images, which can then be used to match RGB images to depth renderings from CAD models. In practice, this would correspond to training our proposed approach with only the local feature alignment objective by sampling all possible keypoint locations. This is similar in spirit to and an improved version of ZDDA [115], a domain adaptation approach that maps RGB and depth modalities to the same point in the latent space.

Joint and alternate training. Finally we use all objectives in our approach and investigate two different training strategies. First we try training all objectives jointly in a single optimization session and report this baseline as *Proposed-joint*. Second, we define a three-step alternating training, where we initially optimize using only the triplet and relative pose losses (i.e. branches A, B, C), then we optimize only with the local euclidean and keypoint consistency losses (i.e. branch D), and in the last step all objectives are jointly optimized together. This baseline is reported as *Proposed-alternate*. Note that also experiments in sec. 4.3.1 and 4.3.4 follow this training paradigm.

Results. We first show, in Figure 4.5 (top row), qualitative keypoint prediction results on test images, where we see keypoint predictions that generally satisfy our intuition of good keypoints. We then adopt the testing protocol described above to report quantitative pose estimation results for test RGB images. Performance analysis is shown in Tables 4.2 and 4.3 for the three object categories. As can be noted from the results, our proposed model generally achieves higher accuracy when compared to the baseline approaches. In particular, the improvements over Baseline-A suggests that keypoint and representation modality adaptation enforced in our model is critical. Furthermore, the improvements over Baseline-ZDDA suggests that simply performing modality adaptation for the RGB and depth features is not sufficient, and learning keypoints and view-invariant representations, as is done in our method, is important to achieve good performance. Finally, we observe that alternating training outperforms the joint strategy, demonstrating the importance of learning good keypoints and representations first, before transferring to the RGB modality.

4.3.3 Model transferability

In this section, we demonstrate the transfer capability, where the goal is for a model, trained according to the proposed approach, to generalize well to category instances **not seen** during training. This is key to practical usability of the approach since we cannot possibly have relevant CAD models of all instances of interest during training. To this end, the baselines introduced in sec. 4.3.2 are re-used with the following experimental protocol:

during training, quadruplets are sampled from a subset of the available instances for each category, and test on RGB images corresponding to all other instances. For instance, for the *bed* category, we use 10 instances for training and 9 instances for testing. Similarly, for *chair* and *desk*, we use 111 and 12 instances respectively for training and the rest for testing. During testing, we use the same protocol as above. We present qualitative keypoint predictions in Figure 4.5 (middle row) and report quantitative performance in Tables 4.4 and 4.5. We see our model shows good transferability, providing (a) a similar level of detail in the predicted keypoints as before, (b) improved accuracy when compared to the baselines, and (c) absolute accuracies that are not too far from those in Tables 4.2 and 4.3.

4.3.4 Framework flexibility

While the results above use RGB-D pairs from Pix3D for model training, in principle, our approach can be used in conjunction with other datasets that provide aligned RGB-D pairs as well. Such capability will naturally make it easier to train models with our framework, leading to improved framework flexibility. To demonstrate this aspect, we train our model as before, but now for input to branches C and D, we use aligned RGB-D pairs from the NYUv2 [54] dataset. Since these pairs contain noisy depth images from a real depth sensor, we synthetically apply realistic noise on the clean rendered depth images, used for branches A and B, using DepthSynth [116]. This ensures branches A, B, and C still receive the same modality as input. Note that we do not test on NYUv2, but rather we use it to collect auxiliary training data and perform testing on Pix3D. Similarly to all other experiments, we do not use any pose annotations for the RGB images as part of training our model and we follow the previous testing protocol. Figure 4.5 (bottom row), shows some keypoint prediction results on test data from Pix3D. In Table 4.6, we report quantitative results. We can make several observations- while the numbers are lower than those with the proposed method, which is expected, they are higher than all the baselines reported in Tables 4.2 and 4.3. Please note that the baselines were trained with alignment from Pix3D, whereas our model here was trained with alignment from NYUv2. These results, along with those

Table 4.6: Results for sec. 4.3.4 experiment. All numbers are % except $MedErr$ (radians).

Metric	Az.	El.	Pl.	$Acc_{\frac{\pi}{6}} \uparrow$	$MedErr \downarrow$
Bed	65.9	54.1	44.0	24.0	1.0
Chair	44.3	51.0	31.0	15.2	1.6
Desk	50.0	45.4	31.6	7.2	1.9

in the previous section, show the potential of our approach in learning generalizable models for estimating object pose, while not explicitly requiring pose annotations during training.

4.4 Conclusions

We proposed a new framework for 3D object pose estimation in RGB images, which does not require either textured CAD models or 3D pose annotations for RGB images during training. We achieve this by means of a novel end-to-end learning pipeline that guides our model to discover keypoints in rendered depth images optimized for relative pose estimation as well as transfer the keypoints and representations to the RGB modality. Our experiments have demonstrated the effectiveness of the proposed method on unseen testing data compared to supervised approaches, suggesting that it is possible to learn generalizable models without depending on pose annotations.

Chapter 5: Simultaneous Mapping and Target Driven Navigation

Navigation is one of the fundamental capabilities of autonomous agents. In recent years there was a surge of novel approaches, which study the problem of goal or target driven navigation using end-to-end learning strategies [34, 117–119]. These approaches use data-driven techniques for learning navigation policies deployable in previously unseen environments without constructing an explicit spatial representation of the environment. These models exploit the power of recurrent neural networks for learning predictions from sequences of observations.

The approaches that address the navigation and planning by learning spatial representations of the environment often assume perfect localization both in the training and testing stage [120, 121]. The problem of localization and mapping is challenging on its own and existing approaches for learning spatial representations which are optimized for localization tasks [122] have been shown to outperform traditional simultaneous localization and mapping methods (SLAM) on the localization task [33].

The presented work investigates the problem of simultaneous mapping and target driven navigation in previously unseen environments. The problem of target driven navigation is a problem of an agent finding its way through a complex environment to a target (e.g. go to the couch). The goal of our work is to exploit mapping and localization module to guide navigation strategies and relax the assumption of perfect localization and at the same time endow the map representation with richer semantic information. Towards this end we propose to build and use spatial allocentric 2.5D map, which will facilitate both localization and semantic target navigation. Instead of using map representation derived directly from

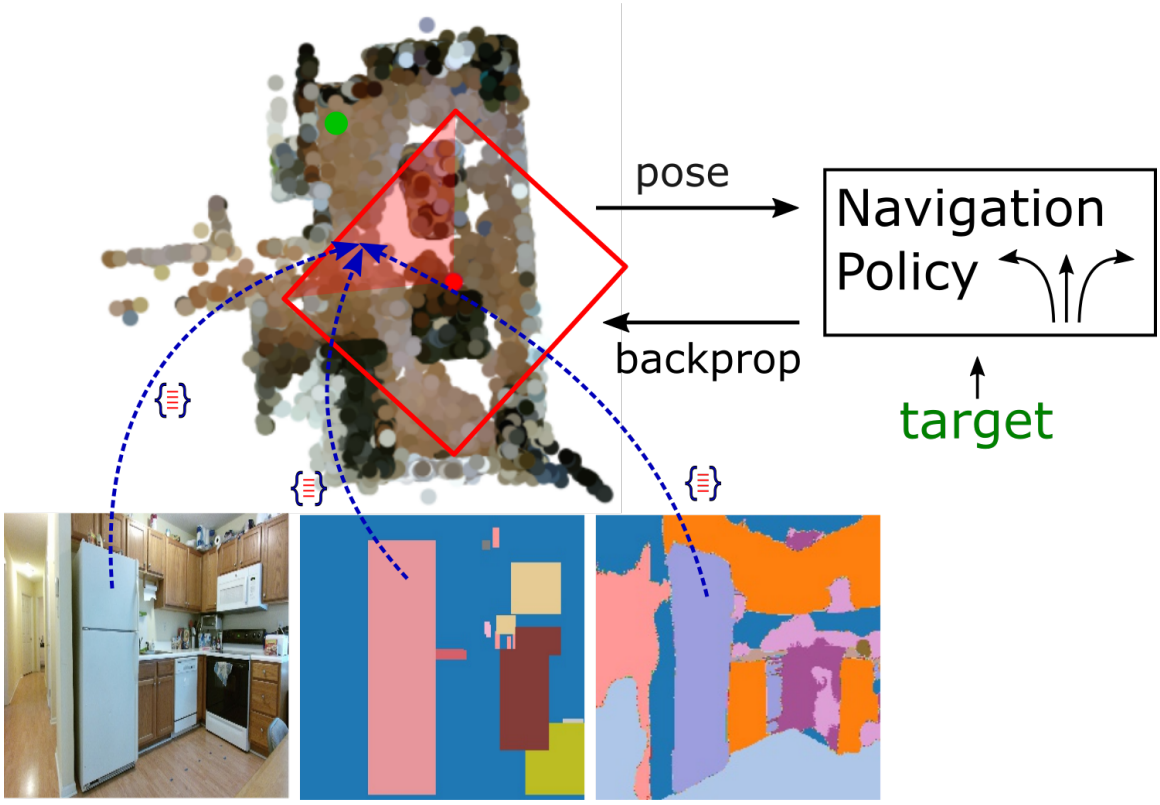


Figure 5.1: We present a new method for target driven navigation that leverages an 2.5D allocentric map with learned semantic representations suitable for both localization and semantic target driven navigation.

pixel values, we propose to learn suitable task related embeddings from outputs of an object detector and semantic segmentation.

The proposed method consists of two modules. First, a semantic map inspired by MapNet architecture [122] is responsible for continuous localization and registration of agent's observations in the map. Second, is a navigation module that uses the partial map, predicted pose of the agent along with current observations for learning a target reaching policy. In summary our contributions are as follows:

- We extend the MapNet [122] approach and endow 2.5D memory with semantically informed features and demonstrate its improvement on the localization task.

- We show the effectiveness of spatial mapping for target driven navigation and learn the navigation policies for the task.
- We evaluate both localization and target driven navigation tasks on real Active Vision Dataset [5] and Matterport3D [36] environment, demonstrating superior performance compared to previous methods.

5.1 Related Work

Traditional approaches for mapping and navigation problem focus on 3D metric and semantic mapping of the environment [33] followed by path planning and control. They require building a 2D or 3D map ahead of time before planning, and do not exploit general semantics and contextual cues in the planning and decision stage. In recent years there was a surge of novel approaches, which study the problem of goal or target driven navigation using end-to-end deep reinforcement learning and vary in the proposed architectures and reward structure to train the models [34, 117–119, 123] These methods use variations of Recurrent Neural networks (i.e. LSTM) with the memory implicitly represented by the hidden state of the model. Majority of the above mentioned methods do not have explicit notion of the map or spatial representation of the environment.

The methods which explicitly learn a spatial representation of the environment, proposed task-dependent differentiable spatial memories to represent the environment [120–122, 124–126] and typically focus on goal or target driven navigation, localization or exploration tasks. For example, Henriques *et al.* [122] proposed an architecture that dynamically updates an agent's allocentric representation for the task of localization. In Gupta *et al.* [120] a mapping module fused information from learned image embeddings across multiple views in an egocentric top-view map of the environment. The mapping module was trained for goal point and semantic target based navigation tasks and assumed accurate localization both in training and testing stage. Authors in [124] train a policy that takes as input a predicted egocentric map and outputs long-term goal for a planner, while Gordon *et al.* [125]

uses a GRU to perform egocentric updates to a local window within a spatial memory given the agent's current location and viewpoint. The spatial memory contains object confidences at each location of a 2D grid, but it does not encode the 3D spatial capacity of the objects in the environment and thus cannot take advantage of multi-view information to deal with occlusions. The work of Chen *et al.* [127] considers the exploration task and constructs the top-view occupancy map by unprojecting 3D points observed in the depth images. The egocentric map is then passed to an exploration policy. With the exception of [125], the learned maps do not consider semantic and contextual information which has been shown to be important in learning generalizable navigation policies [35, 118].

The effectiveness of semantic component, semantic segmentation and object detection has been in approaches which use recurrent neural networks [119,128]. For example, Fang *et al.* [119] uses a scene memory comprised of separately embedded observations at different time steps. While this scene memory encodes semantic information, the lack of structure neglects the spatial configurations of the objects and other semantic categories in the scene.

Our work is also related to large body of work on target driven navigation. The existing approaches differ in the level of supervision, model architectures and tasks. [34, 35, 117, 118, 123, 129–132]. For instance, Mousavian *et al.* [35] and Ye *et al.* [131] learn effective representations for navigation using the outputs of object detectors and semantic segmentations in order to enable better generalization to novel environments and consider navigation policies, with state modelled by LSTM. Sadeghi *et al.* [129] focuses on collision avoidance for the goal reaching task and relies on convolutional LSTM to keep track of the goal's position with respect to the agent. The works of Ye *et al.* [130] and Zhu *et al.* [34] use deep reinforcement learning to train room-type specific navigation policies using feed-forward architectures, where the goal is provided as an image cropped from the scene. Finally, Das *et al.* [118] uses embeddings of a feed-forward model pre-trained on various tasks (i.e. semantic segmentation, depth prediction) as input observation for training the policy. Even though these methods usually include semantic information as an input, they do not explicitely store in a spatial memory and use LSTM modules to retain the history

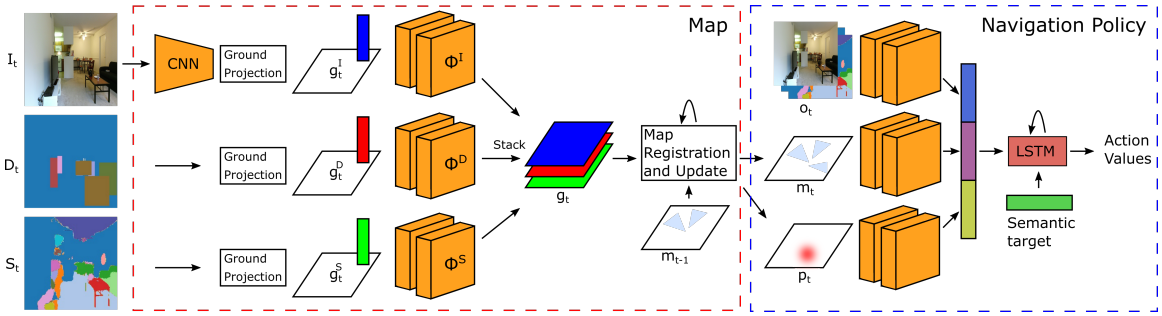


Figure 5.2: Overview of our simultaneous target driven navigation and mapping approach for a single timestep. We use as inputs the egocentric observations RGB image I_t, the detection masks D_t, and semantic segmentation S_t. Each input is first projected to a ground grid before extracting a feature embedding from each grid location. The grids are stacked and passed through a recurrent map registration and update module (see text for more details) which provides the updated map m_t and localization prediction p_t. These, along with the egocentric observations o_t are passed to a navigation module that extracts and concatenates their embeddings with the semantic target. Finally, the embeddings are passed to an LSTM that predicts the values for the next actions. Orange color signifies convolutional blocks, while other colors in the figure denote other feature representations.

of observations and actions.

5.2 Approach

Here we present the two modules of our method. First, we describe in detail the spatial map and how it is endowed with semantic information. Then, we outline the navigation policy and explain how the map is used in its training. An overview of the proposed architecture is in Figure 5.2.

5.2.1 Learned Semantic Map

We are interested in building an allocentric spatial map that encodes the agent's experiences during navigation episodes. Following MapNet [122] formulation, the map at time t is represented as a grid $m_t \in \mathbb{R}^{u \times v \times n}$ of spatial dimensions $u \times v$ with feature embedding of size n at each grid location. Besides an RGB image representation, we also extend the map to

include the semantic information from object detection masks and semantic segmentation. The inputs are initially projected to an egocentric ground grid $g_t \in \mathbb{R}^{u' \times v' \times n}$. For the projection we use the available depth image and the camera intrinsic parameters to obtain a 3D point cloud from the image points. Each 3D point is then mapped to one of the $u' \times v'$ grid coordinates: $x_g = \lfloor \frac{x}{x_b} \rfloor + \frac{u'-1}{2}$, $z_g = \lfloor \frac{z}{z_b} \rfloor + \frac{v'-1}{2}$, where x_g, z_g are the grid coordinates, x_b, z_b are the dimensions of each bin in the grid, and x, z are coordinates of the 3D point. The y coordinate corresponding to the height of the point is neglected in this version of our work. Since multiple 3D points project to the same grid cell, the projected inputs are pooled to form a single vector. Specifically for each input type we get a grid as follows:

RGB image. Given an input image I_t, we obtain a feature map $x_t \in \mathbb{R}^{h \times w \times n'}$ from any backbone CNN (e.g. VGG-16, ResNet50). In order to aggregate the features from different image regions we perform max-pooling over all features vectors projected to the same grid cell to yield the final grid $g_t^I \in \mathbb{R}^{u' \times v' \times n'}$.

Detection mask. We run Faster R-CNN [9] which is pre-trained on COCO [133] and convert the detections to $h \times w \times c_d$ binary bounding box masks. Each channel has detection masks of a particular class from COCO, where c_d is the number of available classes. We get the grid $g_t^D \in \mathbb{R}^{u' \times v' \times c_d}$ by averaging over the occurrences of each detected class in a bin.

Semantic segmentation. We use the model of [6] trained on NYUv2 dataset [54] that outputs a $h \times w$ semantic segmentation of an image. Each pixel takes a value between 0 and $c_s - 1$ where c_s is the number of classes in the NYUv2 dataset. The grid $g_t^S \in \mathbb{R}^{u' \times v' \times c_s}$ for this observation holds a probability distribution over the semantic labels in each bin. Different inputs create separate grids, which are then passed through a small CNNs, comprised of two convolutional layers providing per grid cell feature embedding for each input. This step is deliberately applied on the grids rather than the images directly, such that the learned embeddings can capture spatial dependencies present in the map grid.

We then stack the outputs of the small CNNs to form the egocentric 2D grid g_t at time t:

$$g_t = \left[\phi_I(g_t^I), \phi_D(g_t^D), \phi_S(g_t^S)\right] \tag{5.1}$$

where ϕ_I, ϕ_D, and ϕ_S denote small CNNs applied to embeddings of RGB image, detection masks, and semantic segmentation. The details about choices of individual parameters are described in the experimental sections.

Given this semantically informed representation, we follow the strategy of [122] for localization and registration stage. In order to register g_t in the current map m_{t-1} we densely match g_t with m_{t-1} over all possible locations $u \times v$ and over multiple orientations r. This operation is carried out through cross-correlation (equivalent to convolution in deep learning literature) and produces a tensor $p_t \in \mathbb{R}^{u \times v \times r}$ of scores which denotes the likelihood of the agent's position and orientation in the map at time t. In practice, multiple rotated copies of g_t are stacked together to obtain a $g'_t \in \mathbb{R}^{u' \times v' \times n \times r}$ tensor. After the cross-correlation, the output is passed through a softmax activation function to get p_t. Before inserting g_t in the map, we need to rotate it and translate it according to its localization prediction p_t. This is achieved through a deconvolution operation between g'_t and p_t that can be seen as a linear combination of g'_t weighted by p_t. The result is a tensor that contains the egocentric grid observations at time t, and is aligned to and has the same dimensions as m_{t-1}.

Finally, localized egocentric map g_t is used to update the current map m_{t-1} using a long short-term memory unit (LSTM). Each location's feature embedding is passed through LSTM and updated independently. We have also experimented with other update methods, such as averaging the features, but found the informed updating due to LSTM's trainable parameters to be superior. This can be also attributed to the fact that the LSTM learns how to combine the embeddings of different modalities that comprise g_t in order to be more effective during localization. The model is trained using localization loss.

Localization loss. We use cross-entropy loss to supervise the prediction of p_t:

$$L_{loc} = -\frac{1}{T}\sum_{t}^{T}\sum_{k}^{K}\hat{p}_{tk}\log p_{tk} \tag{5.2}$$

where $\hat{p}_t$ is a one-hot vector representing the ground-truth pose, T is the length of an episode, and $K = u \times v \times r$ is the number of classes corresponding to the discreet spatial locations and orientations in the map. We assume p_0 to be at the center of the map facing to the right, and all subsequent ground-truth poses are relative to p_0.

5.2.2 Navigation Policy

Our task involves navigation to a semantic target within unknown environment. Therefore, it can be formulated as a partially observable Markov decision process (POMDP) $(S, A, O, P(s'|s,a), R(s,a))$, where the state space S consists of the agent's pose, action space A consists of a discrete set of actions, and observation space O is comprised of the egocentric RGB images. The reward $R(s,a) = d(s,c) - d(s',c)$ is defined as the progress towards the semantic target c when at state s the action a is executed that leads to state s', where $d(.,.)$ is the number of steps required on the shortest path between a state and the semantic target. Finally, $P(s'|s,a)$ represents the transition probabilities.

We are interested in learning a policy that can leverage the rich semantic and structured information in the map. To this end, the input to our policy is the allocentric spatial map m_t which holds all past experiences of the agent during the episode. Since we do not assume perfect localization, the map is accompanied by the pose prediction p_t in order to help the policy to pay attention to the relevant parts of the map.

The learned policy $\pi(a|o_t, m_t, p_t; c)$ outputs a distribution over the action space given both egocentric observations $o_t \in O$ and the map (that can be thought as a spatial memory). Since our focus is to navigate in novel environments, the map is being build as the agent moves along and the policy uses as input the accumulated map up to time t. The semantic

target c is represented as a one-hot vector over the set of classes. Finally, a collision indicator represented as a single bit is concatenated to the rest of the inputs in order to encourage the policy to recover after a collision.

Training. Following the work of Mousavian et al [35], we train our policy model to predict the cost of each action a at a certain state s and a given target c using an L1 loss:

$$L_{nav} = \frac{1}{T|A|} \sum_{t}^{T} \sum_{a \in A} \left| y(o_t, m_t, p_t, a; c) - \hat{y}(s, a; c) \right| \tag{5.3}$$

where $\hat{y}(s, a; c) = -R(s, a)$ is the ground-truth cost and $y(o_t, m_t, p_t, a; c)$ is the predicted cost. Given the definition of $R(s, a)$, $\hat{y}(s, a; c)$ can only take one of three values; -1 if the action takes the agent one step closer to target, 1 if it takes the agent one step further from the target, or 0 if the distance remains unchanged. The last case is possible since there can be multiple target poses in an episode. If an action leads to a collision, then we assign $\hat{y}(s, a; c) = 1$ even though the agent has not moved, while if an action leads to a goal we assign $\hat{y}(s, a; c) = -2$.

The policy model is trained in a supervised fashion using an online variant of DAgger [134]. In particular, we first generate training episodes by sampling a random starting point and target in a scene and selecting the actions along the shortest path (expert policy). At this stage we also sample trajectories along randomly chosen paths in order to increase the coverage of the observation space O in the scenes. During training we sample the next minibatch either from the initially generated episodes (expert and random), or by unrolling the current policy to select new episodes. We start with a high probability of selecting from the initial episodes and gradually decrease this probability with exponential decay.

To accommodate this training paradigm, the environment is represented by a graph, where the nodes are discrete poses of the agent and edges represent possible actions. Each pose has corresponding RGB and depth images and the absence of edges between two nodes is treated as collision. In this setting, the shortest path between two nodes can be easily

Table 5.1: Localization results on the AVD and Matterport3D dataset using map models trained with different combinations of input modalities. The Average Position Error (APE) is reported in millimeters for episodes of length 5 and 20.

Dataset	AVD		Matterport3D
Map Model	APE-5	APE-20	APE-20
RGB	285	800	993
RGB-SSeg-Det	215	692	803
SSeg-Det	**179**	**647**	**655**

computed and used as supervision.

Model architecture. The policy $\pi(a|o_t, m_t, p_t; c)$ is modeled by a convolutional NN. The image observations are first passed through a separate network that computes a 128 dimensional embedding. The map m_t and pose estimation p_t are passed through a convolutional layer of $3 \times 3 \times 8$ followed by batch normalization and max-pooling of kernel size 2 and stride 2. The outputs are flattened and passed though a fully connected layer followed by dropout to get the embedding. For the egocentric observation o_t we stack any available images (i.e. RGB, detection masks, or semantic segmentation) and use a pretrained ResNet18 (without the last layer) to extract 512 dimensional features, prior to computing the embedding. The one-hot vector that denotes the semantic target is also encoded to a 128 dimensional embedding. Then, all embeddings are concatenated and used as input to an LSTM layer of 512 units. Finally, a fully connected layer predicts the cost of each action $y(o_t, m_t, p_t, a; c)$.

Controller. During inference, instead of directly choosing the action with the lowest cost, we sample from the predicted probability distribution over the actions by applying soft-max on the negated predicted costs. This still assigns the highest probability to the action with the predicted lowest cost, however it allows for some flexibility during decision making to avoid getting stuck in limited space situations.

Figure 5.3: Example inputs from the AVD (top row) and Matterport3D (bottom row) datasets. From left to right we show the RGB image, detection masks and semantic segmentations.

5.3 Experiments

We perform two main experiments, evaluation of the localization accuracy in the trained spatial map (sec. 5.3.1) and evaluation of the learned navigation policy on unknown environments (sec. 5.3.2). The proposed method is demonstrated on two publicly available datasets, Active vision dataset (AVD) [5] and Matterport3D [36]. We illustrate input examples from the two datasets in Figure 5.3.

AVD. This dataset contains around 20,000 real RGB-D images from typical indoor scenes densely captured by a robot on a 2D grid every $30cm$. Each location on the grid offers multiple views at 30° intervals. The data for each scene are organized as a graph where the edges are defined over a discrete set of actions. This provides with the ability to

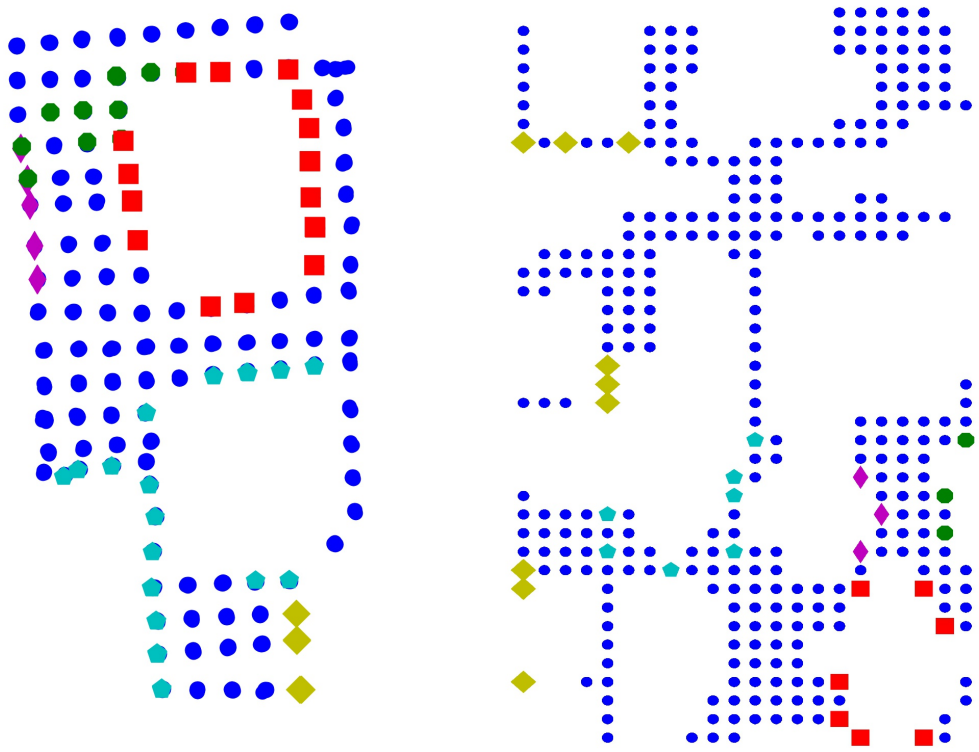


Figure 5.4: Visualizations of graphs along with target locations from an AVD scene (left) and a Matterport3D scene (right). The different shapes and colors denote different target objects.

simulate the movement of an agent in a scene but with the luxury of having real images as observations.

Matterport3D. This dataset contains visually realistic reconstructions of indoor scenes with varying appearance and layout. We endow the dataset with the same structure as AVD by densely sampling navigable positions at $30cm$ intervals on the occupancy map of each scene. At each navigable position, we render RGB images, semantic segmentations and depth images from 12 different orientations at 30° intervals through Habitat-Sim [135]. The images are connected through actions based on their spatial neighborhood and orientations. The built dataset contains more than $10,000$ images for each scene, which is considerably

larger than AVD. We use 17 scenes for training and 5 for testing.

5.3.1 Localization

To validate the effectiveness of using the semantic information for mapping, we present localization results for allocentric maps that were trained with different input combinations and episode lengths of 5 and 20 steps. For both datasets an agent is simulated through random walks in order to collect $33,000$ and $88,000$ training episodes for AVD and Matterport3D respectively. The trained models are applied on episodes not seen during training and are evaluated on Average Position Error (APE), which measures the average Euclidean distance between the ground-truth pose and the predicted pose.

Implementation details. RGB image is passed through a pretrained truncated ResNet50 using only the first 11 layers. The small CNNs for each input grid modality (ϕ_I, ϕ_D, ϕ_S) are realized with two convolutional layers of $3 \times 3 \times 64$ and $3 \times 3 \times l$, where $l = 32$ for ϕ_I, and $l = 16$ for both ϕ_D and ϕ_S. The number of units n for the LSTM corresponds to the summation of the embedding dimensions of the modalities used for the particular experiment. Regarding the hyper-parameters of the map, we define the map dimensions $u = v = 29$, the egocentric observation grid dimensions $u' = v' = 21$, the number of rotations $r = 12$, and the grid cell size $x_b = z_b = 300mm$.

Results and discussion. The results are illustrated in Table 5.1, where semantic segmentation is denoted as *SSeg* and detection masks as *Det*. There is no direct comparison to the results from [122] since the exact train and test sets are not provided, however our map model trained only with RGB can be considered analogous to the MapNet trained in [122]. We observe that the model with the lowest localization error in both 5 and 20 step cases is *SSeg-Det*, which is not using RGB information. This can be attributed to the fact that the RGB image representation needs to capture view-invariant properties, which is difficult to achieve, such that the egocentric ground grid can be accurately matched to the allocentric map. On the other hand, this is not necessary in the case of *SSeg-Det*, since

Table 5.2: Results of semantic target navigation in novel scenes in AVD and Matterport3D.

Dataset	AVD		Matterport3D	
Method	Succ. (%)	Path Len. Rat.	Succ. (%)	Path Len. Rat.
Random Walk	24.3	4.4	13.6	10.7
Non-learning	35.3	9.3	42.8	3.6
No-Map-AVD [35]	48.0	-	-	-
No-Map-SUNCG [35]	54.0	-	-	-
Ours	**64.6**	**1.9**	**69.5**	2.3

it operates on recognition outputs. This is also highlighted on the Matterport3D results, where the images are synthetically generated and the average position error difference is more in favour of *SSeg-Det*. This effectively demonstrates that a scene can be memorized with respect to only semantic information such that it is useful for re-localization. In fact, when the RGB images are added then the error slightly increases. It is also important to note that the authors of [122] demonstrated superior performance of their approach with respect to traditional ORB-SLAM [136] on AVD using only RGB images. We exhibit that additional semantic information further improves the localization ability of the agent.

5.3.2 Navigation to semantic target

Here we investigate the effectiveness of the proposed target driven navigation policy. Our objective in this experiment is twofold. First, we would like to demonstrate the effect of using a spatial map in comparison to LSTM policies that do not use a map, and second, investigate navigation policies that are learned with spatial maps of different modalities.

The training procedure is as follows. First, the spatial map model is trained with the localization objective. The training of the navigation policy uses the frozen mapping module to update the map and predict the agent's pose at each step. During this procedure, the mapping module is fine-tuned through the navigation objective. Note that at the beginning of each episode the map contains no information. Unless otherwise specified, we train the navigation policy using the *SSeg-Det* map model for AVD, while the *RGB-SSeg-Det* map

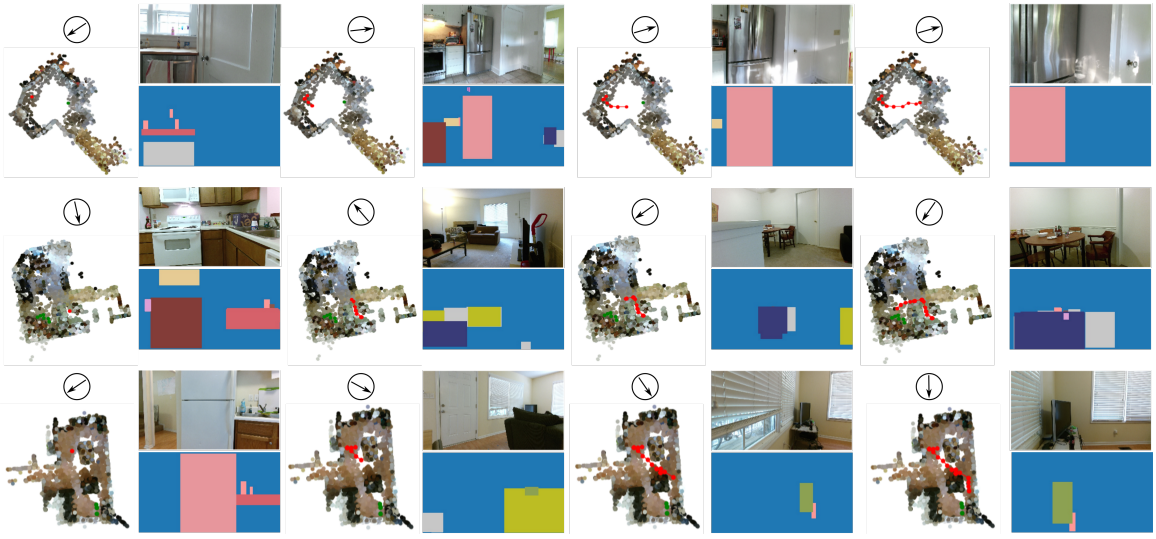


Figure 5.5: Qualitative navigation results on the AVD dataset. Each row corresponds to a different episode. From top to bottom, the target object is the fridge, dining table, and TV. For each step of an episode we present the map with the agent's trajectory up to that time, the agent's orientation, RGB image and detection masks. Notice that in all three episodes the agent moves quickly towards the target once it is detected and placed in the map.

model is utilized when learning the policy on Matterport3D.

For both AVD and Matterport3D five semantic targets are identified: {*dining_table, refrigerator, tv, couch, microwave*}. Figure 5.4 presents examples of scene graphs with marked target objects. In the case of AVD we compare our approach to [35], therefore we follow their train/test split of *different environments* (11 scenes for training and 3 for testing), and use the target locations of object categories they provide. As mentioned in sec. 5.2.2 the training data are generated using DAGGER with 55,000 and 88,000 initial training episodes for AVD and Matterport3D respectively. For evaluation, the percentage of successful episodes is reported along with the average path length ratio. The episode is successful when the agent is at least 5 steps away from a target pose. The path length ratio is the ratio between the predicted path and the shortest path length and is calculated only for successful episodes. The maximum number of steps for each episode is 100.

Comparison to other policies. To demonstrate our method's superiority to policies which do not use a spatial map we have defined three baselines:

Random walk. The agent chooses random actions until it reaches the target.

Non-learning baseline. Similarly to [35], the agent chooses random actions until the target object is detected, in which case the agent computes the shortest path to the target. Note that for this baseline, the agent has full knowledge of the environment and its graph once it detects the target.

Learned LSTM policy without a map. The method proposed in [35]. The policy is learned using only egocentric observations without any spatial memory. We compare to a policy trained on AVD that uses detection masks denoted as *No-Map-AVD* and policy trained on both AVD and the large synthetic dataset SUNCG [137] that uses detection masks and semantic segmentations, here referred to as *No-Map-SUNCG*. This is the best performing method reported in [35] as it leverages the vast amounts of data offered in SUNCG's 200 synthetic environments.

Results are shown in Table 5.2. On AVD dataset the presented approach outperforms the second best method by 10.6%, without any use of complementary synthetic data during training. The biggest advantage of our method compared to the other learned baselines is that our agent has access to semantic information stored in a structured memory that corresponds to the history of observations. This reduces the amount of information that LSTM retains and helps during the optimization of the policy. Furthermore, the lower average path length ratio than the *Non-learning* baseline, in both datasets, suggests that our navigation model learns contextual cues from the semantic map that reduce the time spent searching for the target. In addition, as demonstrated in Figure 5.5, the agent learns to quickly move towards the target upon detection. Note that the baseline uses an optimal path when the target is detected. Note that the average path length ratio result is not directly comparable to the other learned baselines since they require a *stop* action to end an episode, which we do not use.

Table 5.3: Results of our ablation study on AVD, illustrating the performance of navigation models trained with different map models, without fine-tuning the map (NF), or without using any egocentric observations (NE).

Model	Success Rate (%)
1. Nav-RGB	60.1
2. Nav-RGB-Det	61.1
3. Nav-RGB-SSeg-Det	63.2
4. Nav-SSeg-Det	**64.6**
5. Nav-RGB-NF	58.2
6. Nav-SSeg-Det-NF	60.4
7. Nav-RGB-NF-NE	53.9
8. Nav-SSeg-Det-NF-NE	56.9

Table 5.4: Results of our ablation study on Matterport3D, illustrating the performance of navigation models trained with different map models and without fine-tuning the map (NF).

Model	Success Rate (%)
1. Nav-SSeg-Det	62.9
2. Nav-RGB	66.7
3. Nav-RGB-SSeg-Det	**69.5**
4. Nav-SSeg-Det-NF	59.7
5. Nav-RGB-NF	64.2

Ablation Study In this section we attempt to get a better understanding of how certain components of our method contribute to the overall performance.

Variations of spatial map modalities. Here we pose the question of which are the most suitable map feature representations when learning to navigate. To this end, we trained multiple navigation models using spatial maps that were learned with different modality inputs. Results are presented in lines 1-4 of Table 5.3 for AVD, and lines 1-3 of Table 5.4 for Matterport3D. In the case of AVD, we notice that the *Nav-SSeg-Det* outperforms *Nav-RGB* by 4.5%. This validates our assumption that navigating to a semantic target can be

successful using a map with purely semantic features. Note also that the map model *SSeg-Det* demonstrated the lowest localization error (see Table 5.1). However, for Matterport3D we observe that *Nav-RGB* outperforms *Nav-SSeg-Det*, while the best model is the one using all modalities (*Nav-RGB-SSeg-Det*). This can be explained by the fact that the detections are very noisy due to the artifacts in Matterport3D's images. Hence, they are not as reliable when training the policy. Another reason could be that Matterport3D has larger scenes and object encounters are sparser, therefore providing less useful information in the map.

Joint training. To see the effect of fine-tuning the mapping module we re-trained selected navigation policies but kept the map network parameters frozen. The results are in lines 5, 6 of Table 5.3 for AVD and lines 4, 5 of Table 5.4 for Matterport3D, where the models trained without fine-tuning are denoted as NF. There is a consistent reduction of performance when fine-tuning is not performed in both datasets. This shows that the map embeddings learned during the initial training of the map module are not immediately applicable for navigation and further adjustment is required.

Effect of egocentric observation. We rely mainly on the allocentric map and pose prediction to decide future actions, which requires a depth sensor during the ground projection step. However, in cases where the agent is very close to an object and therefore outside of the effective range of the depth sensor, the projection is unreliable. We argue that using complementary egocentric observations as input to the navigation policy can help mitigate this problem. Results of models trained with and without egocentric observations are reported in lines 7, 8 of Table 5.3 for AVD. We observe that there is a rough 4% degradation in the performance compared to lines 5, 6 in the same table, which validates our assumption.

5.4 Conclusions

We have presented a new method for simultaneous mapping and target driven navigation in novel environments. The mapping component leverages the outputs of object detection and semantic segmentation to construct a spatial representation of a scene which contains some semantic information. This representation is then used to optimize a navigation policy

that takes advantage of the agent's experiences during an episode encoded in the allocentric spatial map. The experiments on AVD and Matterport3D environments demonstrate that our approach outperforms only RGB baselines for the task of localization, and non-mapping baselines for the target-driven navigation.

Chapter 6: Conclusions and Future Work

We have proposed several techniques to mitigate the need for manual image annotation in multiple tasks. Our contributions include the development of new architectures and objective functions for object instance detection, object pose estimation, and target driven navigation.

In Chapter 2 we have presented a novel automated approach for generating synthesized datasets that can be used to train state-of-the-art deep CNN object detectors. The approach leverages geometric and semantic cues in order to informatively place and scale cropped object masks in meaningful locations in a scene. Furthermore, a blending technique ensures that the composited object masks do not carry any artifacts. Our method is attractive because it is scalable with both the number of objects of interest and with a variety of indoor scenes.

Chapters 3 and 4 present methods for 3D object pose estimation through keypoint learning and matching. In Chapter 3 a method is proposed that jointly optimizes the keypoint locations and representation learning for 3D keypoint matching in depth images. The main contribution of this work is that the optimization is performed without attempting to define apriori "discriminative" keypoints. The approach is evaluated on depth data from a structured light sensor, such as Kinect, where robust matching is demonstrated under large viewpoint variations. The method in Chapter 4 extends the work of Chapter 3 and proposes a new method for pose estimation in the RGB domain, where a new keypoint learning objective is introduced that takes into consideration the downstream task of pose estimation. The key contribution is that the method does not require explicit 3D pose annotations for the RGB images, rather it takes advantage of abundant data, rendered from CAD models to learn the keypoint detector and descriptor representation. The method is demonstrated on object instances not seen during training.

Finally, the last part of this thesis (Chapter 5) introduces a new method for target driven navigation that leverages a semantically informed spatial map during policy optimization without assuming perfect localization. Two key ingredients of our method are the use of an allocentric map that learns how to encode the past experiences of the agent, and the exploitation of semantic information, such as object detection masks, while learning the map representation.

6.1 Future Work

A possible extension of this work would be to scale the current methods on larger sets of objects. For example, our object instance synthesization method was demonstrated on 11 object instances and only in kitchen scenes. This can be scaled up by collecting more scans of objects and RGB-D images of background scenes, without any annotation required. Similarly, the object pose estimation method presented in Chapter 4 can benefit from further collection of RGB-D images of objects. As explained in Section 4.3.4, where we investigate the flexibility of our framework, the approach can be trained with any available auxiliary data. Unfortunately, suitable large-scale datasets do not exist, so the experiment was carried out on limited amounts of data from the NYUv2 dataset and showed low performance. We believe that with the proper amounts of data, the model can perform at least on par with the rest of the results shown in the experiments.

Including more auxiliary objectives during optimization would be another direction for improving the current methods. For example, the keypoints and descriptors learned in Chapter 3 could be informed by the 3D structure encoded by learning novel view synthesis in depth images. Currently, the proposed approach does not have a specific structure-aware mechanism. In addition, enforcing cyclic viewpoint consistency on multiple frames can provide stronger constraints in generating repeatable keypoints and learning view-invariant representations, than the current training image pairs.

In learning based target driven navigation there are still many unanswered questions, especially regarding the representation of the environment. The work presented in Chapter

5 uses a 2D grid and shows promising results for relatively shorter episodes (on average 20 steps), but it is not clear how well it would scale to longer trajectories. Perhaps a future direction for this work would be to investigate graph-based environment representations which are more concise when summarizing the information in a scene.

While this thesis has contributed several methods in relieving the dependence on manual annotations, in the long term, we aspire to develop techniques that are purely self-supervised. An example involves embodied agents that can actively gather their own data by moving and interacting with their environment. Self-supervision could be a first step in the overarching problem of artificial intelligence of building systems that can understand their environment and continuously learn from past behaviors. While we still have a long way to go, the recent progress in the field is allowing our imagination to be informed by our expertise and perhaps be put to the test by the right tools. I am greatly excited by our potential under these conditions.

Bibliography

Bibliography

[1] J. Shotton, B. Glocker, C. Zach, S. Izadi, A. Criminisi, and A. Fitzgibbon, "Scene coordinate regression forests for camera relocalization in RGB-D images," in *CVPR*, 2013, pp. 2930–2937.

[2] G. Georgakis, M. Reza, A. Mousavian, P. Le, and J. Kosecka, "Multiview RGB-D dataset for object instance detection," in *IEEE International Conference on 3DVision (3DV)*, 2016.

[3] "http://graphics.stanford.edu/data/3dscanrep/," in *Stanford 3D scanning repository*.

[4] X. Sun, J. Wu, X. Zhang, Z. Zhang, C. Zhang, T. Xue, J. B. Tenenbaum, and W. T. Freeman, "Pix3d: Dataset and methods for single-image 3d shape modeling," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2974–2983.

[5] P. Ammirato, P. Poirson, E. Park, J. Košecka, and A. C. Berg, "A dataset for developing and benchmarking active vision," in *ICRA*, 2017.

[6] A. Mousavian, H. Pirsiavash, and J. Kosecka, "Joint semantic segmentation and depth estimation with deep convolutional networks," in *IEEE International Conference on 3DVision (3DV)*, 2016.

[7] B. Planche, Z. Wu, K. Ma, S. Sun, S. Kluckner, T. Chen, A. Hutter, S. Zakharov, H. Kosch, and J. Ernst, "Depthsynth: Real-time realistic synthetic data generation from CAD models for 2.5D recognition," *3DV*, 2017.

[8] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[9] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Conference on Neural Information Processing Systems (NIPS)*, 2015.

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

[11] W. Liu, D. Anguelov, C. S. D. Erhan, S. Reed, C. Fu, and A. C. Berg, "SSD: Single shot multibox detector," in *European Conference on Computer Vision (ECCV)*, 2016.

[12] A. Kundu, Y. Li, and J. M. Rehg, "3d-rcnn: Instance-level 3d object reconstruction via render-and-compare," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 3559–3568.

[13] S. Tulsiani and J. Malik, "Viewpoints and keypoints," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1510–1519.

[14] A. Mousavian, D. Anguelov, J. Flynn, and J. Košecká, "3d bounding box estimation using deep learning and geometry," in *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*. IEEE, 2017, pp. 5632–5640.

[15] N. Kolotouros, G. Pavlakos, and K. Daniilidis, "Convolutional mesh regression for single-image human shape reconstruction," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4501–4510.

[16] A. Kanazawa, M. J. Black, D. W. Jacobs, and J. Malik, "End-to-end recovery of human shape and pose," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 7122–7131.

[17] G. Pavlakos, V. Choutas, N. Ghorbani, T. Bolkart, A. A. Osman, D. Tzionas, and M. J. Black, "Expressive body capture: 3d hands, face, and body from a single image," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 10 975–10 985.

[18] X. Peng, B. Sun, K. Ali, and K. Saenko, "Learning deep object detectors from 3D models," in *IEEE International Conference on Computer Vision (ICCV)*, 2015.

[19] H. Su, C. Qi, Y. Li, and L. Guibas, "Render for cnn: Viewpoint estimation in images using cnns trained with rendered 3D model views," in *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[20] S. Richter, V. Vineet, S. Roth, and V. Koltun, "Playing for data: Ground truth from computer games," in *European Conference on Computer Vision (ECCV)*, 2016.

[21] K. Lai, L. Bo, and D. Fox, "Unsupervised feature learning for 3D scene labeling," in *IEEE International Conference on on Robotics and Automation (ICRA)*, 2014.

[22] G. Georgakis, A. Mousavian, A. C. Berg, and J. Kosecka, "Synthesizing training data for object detection in indoor scenes," *arXiv preprint arXiv:1702.07836*, 2017.

[23] I. Sipiran and B. Bustos, "Harris 3D: a robust extension of the harris operator for interest point detection on 3d meshes," *The Visual Computer*, vol. 27, no. 11, pp. 963–976, 2011.

[24] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *IJCV*, vol. 60, no. 2, pp. 91–110, 2004.

[25] R. B. Rusu, N. Blodow, and M. Beetz, "Fast point feature histograms (FPFH) for 3D registration," in *ICRA*. IEEE, 2009, pp. 3212–3217.

[26] K. M. Yi, E. Trulls, V. Lepetit, and P. Fua, "Lift: Learned invariant feature transform," in *European Conference on Computer Vision*. Springer, 2016, pp. 467–483.

[27] H. Altwaijry, A. Veit, S. J. Belongie, and C. Tech, "Learning to detect and match keypoints with deep architectures." in *BMVC*, 2016.

[28] G. Georgakis, S. Karanam, Z. Wu, J. Ernst, and J. Košecká, "End-to-end learning of keypoint detector and descriptor for pose invariant 3d matching," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 1965–1973.

[29] 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.

[30] S. Mahendran, H. Ali, and R. Vidal, "3d pose regression using convolutional neural networks," in *IEEE International Conference on Computer Vision*, vol. 1, 2017, p. 4.

[31] P. Poirson, P. Ammirato, C.-Y. Fu, W. Liu, J. Kosecka, and A. C. Berg, "Fast single shot detection and pose estimation," in *3D Vision (3DV), 2016 Fourth International Conference on*. IEEE, 2016, pp. 676–684.

[32] G. Georgakis, S. Karanam, Z. Wu, and J. Kosecka, "Learning local rgb-to-cad correspondences for object pose estimation," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 8967–8976.

[33] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. D. Reid, and J. J. Leonard, "Simultaneous localization and mapping: Present, future, and the robust-perception age," *IEEE Transactions on Robotics*, 2016.

[34] Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei, and A. Farhadi, "Target-driven visual navigation in indoor scenes using deep reinforcement learning," in *2017 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2017, pp. 3357–3364.

[35] A. Mousavian, A. Toshev, M. Fišer, J. Košecká, A. Wahid, and J. Davidson, "Visual representations for semantic target driven navigation," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 8846–8852.

[36] A. Chang, A. Dai, T. Funkhouser, M. Halber, M. Niessner, M. Savva, S. Song, A. Zeng, and Y. Zhang, "Matterport3d: Learning from RGB-D data in indoor environments," *3DV*, 2017.

[37] R. Girshick, "Fast R-CNN," in *IEEE International Conference on Computer Vision (ICCV)*, 2015.

[38] J. Mahler, F. T. Pokorny, B. Hou, M. Roderick, M. Laskey, M. Aubry, K. Kohlhoff, T. Kroeger, J. Kuffner, and K. Goldberg, "Dex-net 1.0: A cloud-based network of 3D objects for robust grasp planning using a multi-armed bandit model with correlated rewards," in *International Conference on Robotics and Automation (ICRA)*, 2016.

[39] A. Singh, J. Sha, K. Narayan, T. Achim, and P. Abbeel, "A large-scale 3D database of object instances." in *IEEE International Conference on Robotics and Automation (ICRA)*, 2014.

[40] Y. Sun, M. Bianchi, J. Bohg, and A. Dollar, "http://rhgm.org/activities/workshopicra16/," in *Workshop on Grasping and Manipulation Datasets (ICRA)*, 2016.

[41] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part based models," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 32, no. 9, pp. 1627–1645, 2010.

[42] K. Lai, L. Bo, X. Ren, and D. Fox, "A large-scale hierarchical multi-view RGB-D object dataset," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2011.

[43] A. Collet, M. Martinez, and S. Srinivasa, "The MOPED framework: Object recognition and pose estimation for manipulation," in *International Journal of Robotics Research (IJRR)*, 2011.

[44] J. Tang, S. Miller, A. Singh, and P. Abbeel, "A textured object recognition pipeline for color and depth image data," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2012.

[45] J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, and A. W. M. Smeulders, "Selective search for object recognition," *International Journal of Computer Vision (IJCV)*, 2013. [Online]. Available: http://www.huppelen.nl/publications/selectiveSearchDraft.pdf

[46] M. M. Cheng, Z. Zhang, W. Y. Lin, and P. Torr, "BING: Binarized normed gradients for objectness estimation at 300fps," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[47] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with Deep Convolutional Neural Networks," in *Conference on Neural Information Processing Systems (NIPS)*, 2012.

[48] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *IEEE Conference Computer Vision and Pattern Recognition (CVPR)*, 2014.

[49] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, "Overfeat: Integrated recognition, localization and detection using convolutional networks," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013.

[50] A. Gupta, A. Vedaldi, and A. Zisserman, "Synthetic data for text localisation in natural images," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[51] S. Song, L. Zhang, and J. Xiao, "Robot in a room: Toward perfect object recognition in closed environments," in *arXiv:1507.02703 [cs.CV] 9 Jul 2015*, July 2015.

[52] C. Taylor and A. Cowley, "Parsing indoor scenes using RGB-D imagery," in *Robotics: Science and Systems (RSS)*, July 2012.

[53] M. Tanaka, R. Kamio, and M. Okutomi, "Seamless image cloning by a closed form solution of a modified poisson problem," in *SIGGRAPH Asia 2012 Posters*, ser. SA '12, 2012.

[54] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, "Indoor segmentation and support inference from rgbd images," in *European Conference on Computer Vision*. Springer, 2012, pp. 746–760.

[55] Y. Boykov, O. Veksler, and R. Zabih, "Fast approximate energy minimization via graph cuts," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 23, no. 11, pp. 1222–1239, Nov. 2001. [Online]. Available: http://dx.doi.org/10.1109/34.969114

[56] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014.

[57] E. Rosten, R. Porter, and T. Drummond, "Faster and better: A machine learning approach to corner detection," *IEEE T-PAMI*, vol. 32, no. 1, pp. 105–119, 2010.

[58] S. Leutenegger, M. Chli, and R. Y. Siegwart, "BRISK: Binary robust invariant scalable keypoints," in *ICCV*. IEEE, 2011, pp. 2548–2555.

[59] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," in *ICCV*. IEEE, 2011, pp. 2564–2571.

[60] R. B. Rusu, N. Blodow, Z. C. Marton, and M. Beetz, "Aligning point cloud views using persistent feature histograms," in *IROS*. IEEE, 2008, pp. 3384–3391.

[61] A. Frome, D. Huber, R. Kolluri, T. Bülow, and J. Malik, "Recognizing objects in range data using regional point descriptors," *ECCV*, pp. 224–237, 2004.

[62] R. B. Rusu and S. Cousins, "3D is here: Point cloud library (PCL)," in *ICRA*. IEEE, 2011, pp. 1–4.

[63] Y. Guo, M. Bennamoun, F. Sohel, M. Lu, J. Wan, and N. M. Kwok, "A comprehensive performance evaluation of 3d local feature descriptors," *IJCV*, vol. 116, no. 1, pp. 66–89, 2016.

[64] J. L. Schönberger, H. Hardmeier, T. Sattler, and M. Pollefeys, "Comparative evaluation of hand-crafted and learned local features," in *CVPR*, 2017.

[65] S. Zagoruyko and N. Komodakis, "Learning to compare image patches via convolutional neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 4353–4361.

[66] X. Han, T. Leung, Y. Jia, R. Sukthankar, and A. C. Berg, "Matchnet: Unifying feature and metric learning for patch-based matching," in *CVPR*, 2015, pp. 3279–3286.

[67] P. Wohlhart and V. Lepetit, "Learning descriptors for object recognition and 3D pose estimation," in *CVPR*, 2015, pp. 3109–3118.

[68] A. Zeng, S. Song, M. Nießner, M. Fisher, J. Xiao, and T. Funkhouser, "3dmatch: Learning local geometric descriptors from RGB-D reconstructions," in *CVPR*, 2017, pp. 1802–1811.

[69] S. Song and J. Xiao, "Deep sliding shapes for amodal 3D object detection in RGB-D images," in *CVPR*, 2016, pp. 808–816.

[70] D. Maturana and S. Scherer, "3d convolutional neural networks for landing zone detection from lidar," in *ICRA*. IEEE, 2015, pp. 3471–3478.

[71] S. Salti, F. Tombari, R. Spezialetti, and L. Di Stefano, "Learning a descriptor-specific 3D keypoint detector," in *ICCV*, 2015, pp. 2318–2326.

[72] N. Savinov, A. Seki, L. Ladicky, T. Sattler, and M. Pollefeys, "Quad-networks: unsupervised learning to rank for interest point detection," *CVPR*, 2017.

[73] K. M. Yi, E. Trulls, V. Lepetit, and P. Fua, "LIFT: Learned Invariant Feature Transform," in *ECCV*, 2016.

[74] K. Lenc and A. Vedaldi, "Learning covariant feature detectors," in *ECCV Workshops*. Springer, 2016, pp. 100–117.

[75] J. L. Schonberger and J.-M. Frahm, "Structure-from-motion revisited," in *CVPR*, 2016, pp. 4104–4113.

[76] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison *et al.*, "Kinectfusion: Real-time 3D reconstruction and interaction using a moving depth camera," in *UIST*. ACM, 2011, pp. 559–568.

[77] T. Schmidt, R. Newcombe, and D. Fox, "Self-supervised visual descriptor learning for dense correspondence," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 420–427, 2017.

[78] Y. Zhong, "Intrinsic shape signatures: A shape descriptor for 3D object recognition," in *ICCV Workshops*. IEEE, 2009, pp. 689–696.

[79] S. Salti, F. Tombari, and L. Di Stefano, "Shot: Unique signatures of histograms for surface and texture description," *CVIU*, vol. 125, pp. 251–264, 2014.

[80] ——, "A performance evaluation of 3D keypoint detectors," in *3DIMPVT*. IEEE, 2011, pp. 236–243.

[81] W. Kehl, F. Milletari, F. Tombari, S. Ilic, and N. Navab, "Deep learning of local rgb-d patches for 3d object detection and 6d pose estimation," in *European Conference on Computer Vision*. Springer, 2016, pp. 205–220.

[82] M. Rad, M. Oberweger, and V. Lepetit, "Domain transfer for 3d pose estimation from color images without manual annotations," *arXiv preprint arXiv:1810.03707*, 2018.

[83] W. Kehl, F. Manhardt, F. Tombari, S. Ilic, and N. Navab, "Ssd-6d: Making rgb-based 3d detection and 6d pose estimation great again," in *Proceedings of the International Conference on Computer Vision (ICCV 2017), Venice, Italy*, 2017, pp. 22–29.

[84] A. Collet, M. Martinez, and S. S. Srinivasa, "The moped framework: Object recognition and pose estimation for manipulation," *The International Journal of Robotics Research*, vol. 30, no. 10, pp. 1284–1306, 2011.

[85] J. Tang, S. Miller, A. Singh, and P. Abbeel, "A textured object recognition pipeline for color and depth image data," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE, 2012, pp. 3467–3474.

[86] A. Doumanoglou, R. Kouskouridas, S. Malassiotis, and T.-K. Kim, "Recovering 6d object pose and predicting next-best-view in the crowd," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 3583–3592.

[87] J. J. Lim, H. Pirsiavash, and A. Torralba, "Parsing ikea objects: Fine pose estimation," in *Proceedings of the IEEE International Conference on Computer Vision*, 2013, pp. 2992–2999.

[88] Y. Xiang, R. Mottaghi, and S. Savarese, "Beyond pascal: A benchmark for 3d object detection in the wild," in *Applications of Computer Vision (WACV), 2014 IEEE Winter Conference on*. IEEE, 2014, pp. 75–82.

[89] M. Janner, J. Wu, T. D. Kulkarni, I. Yildirim, and J. Tenenbaum, "Self-supervised intrinsic image decomposition," in *Advances in Neural Information Processing Systems*, 2017, pp. 5936–5946.

[90] S. Hinterstoisser, S. Holzer, C. Cagniart, S. Ilic, K. Konolige, N. Navab, and V. Lepetit, "Multimodal templates for real-time detection of texture-less objects in heavily cluttered scenes," in *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE, 2011, pp. 858–865.

[91] T. Hodan, F. Michel, E. Brachmann, W. Kehl, A. G. Buch, D. Kraft, B. Drost, J. Vidal, S. Ihrke, X. Zabulis, S. Caner, M. Fabian, T. Federico, K. Tae-Kyun, M. Jiri, and R. Carsten, "Bop: Benchmark for 6d object pose estimation," in *ECCV*, 2018.

[92] M. Zhu, K. G. Derpanis, Y. Yang, S. Brahmbhatt, M. Zhang, C. Phillips, M. Lecce, and K. Daniilidis, "Single image 3d object detection and pose estimation for grasping," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE, 2014, pp. 3936–3943.

[93] E. Brachmann, A. Krull, F. Michel, S. Gumhold, J. Shotton, and C. Rother, "Learning 6d object pose estimation using 3d object coordinates," in *European conference on computer vision*. Springer, 2014, pp. 536–551.

[94] A. Tejani, D. Tang, R. Kouskouridas, and T.-K. Kim, "Latent-class hough forests for 3d object detection and pose estimation," in *European Conference on Computer Vision*. Springer, 2014, pp. 462–477.

[95] M. Aubry, D. Maturana, A. A. Efros, B. C. Russell, and J. Sivic, "Seeing 3d chairs: exemplar part-based 2d-3d alignment using a large dataset of cad models," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 3762–3769.

[96] F. Massa, B. C. Russell, and M. Aubry, "Deep exemplar 2d-3d detection by adapting from real to rendered views," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 6024–6033.

[97] H. Izadinia, Q. Shan, and S. M. Seitz, "Im2cad," in *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*. IEEE, 2017, pp. 2422–2431.

[98] A. Bansal, B. Russell, and A. Gupta, "Marr revisited: 2d-3d alignment via surface normal prediction," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 5965–5974.

[99] R. Mottaghi, Y. Xiang, and S. Savarese, "A coarse-to-fine model for 3d pose estimation and sub-category recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 418–426.

[100] Z. Wang, W. Li, Y. Kao, D. Zou, Q. Wang, M. Ahn, and S. Hong, "Hcr-net: A hybrid of classification and regression network for object pose estimation." in *IJCAI*, 2018, pp. 1014–1020.

[101] Y. Kao, W. Li, Z. Wang, D. Zou, R. He, Q. Wang, M. Ahn, S. Hong *et al.*, "An appearance-and-structure fusion network for object viewpoint estimation." in *IJCAI*, 2018, pp. 4929–4935.

[102] G. Pavlakos, X. Zhou, A. Chan, K. G. Derpanis, and K. Daniilidis, "6-dof object pose from semantic keypoints," in *Robotics and Automation (ICRA), 2017 IEEE International Conference on*. IEEE, 2017, pp. 2011–2018.

[103] M. Hueting, P. Reddy, V. Kim, N. Carr, E. Yumer, and N. Mitra, "Seethrough: finding chairs in heavily occluded indoor scene images," *CoRR abs/1710.10473*, 2017.

[104] J. Wu, T. Xue, J. J. Lim, Y. Tian, J. B. Tenenbaum, A. Torralba, and W. T. Freeman, "3d interpreter networks for viewer-centered wireframe modeling," *International Journal of Computer Vision*, pp. 1–18, 2018.

[105] C. Li, M. Z. Zia, Q.-H. Tran, X. Yu, G. D. Hager, and M. Chandraker, "Deep supervision with shape concepts for occlusion-aware 3d object parsing," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[106] 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*, 2018, pp. 292–301.

[107] S. Suwajanakorn, N. Snavely, J. Tompson, and M. Norouzi, "Discovery of latent 3d keypoints via end-to-end geometric reasoning," *arXiv preprint arXiv:1807.03146*, 2018.

[108] X. Zhou, A. Karpur, C. Gan, L. Luo, and Q. Huang, "Unsupervised domain adaptation for 3d keypoint estimation via view consistency," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 137–153.

[109] H. Su, C. R. Qi, Y. Li, and L. J. Guibas, "Render for cnn: Viewpoint estimation in images using cnns trained with rendered 3d model views," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 2686–2694.

[110] S. Gupta, P. Arbeláez, R. Girshick, and J. Malik, "Inferring 3d object pose in rgb-d images," *arXiv preprint arXiv:1502.04652*, 2015.

[111] Blender, "https://www.blender.org/."

[112] O. Sorkine-Hornung and M. Rabinovich, "Least-squares rigid motion using svd," *no*, vol. 3, pp. 1–5, 2017.

[113] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[114] S. Zhao and W. T. Ooi, "Modeling 3d synthetic view dissimilarity," *The Visual Computer*, vol. 32, no. 4, pp. 429–443, 2016.

[115] K.-C. Peng, Z. Wu, and J. Ernst, "Zero-shot deep domain adaptation," in *European Conference on Computer Vision*. Springer, 2018, pp. 793–810.

[116] B. Planche, Z. Wu, K. Ma, S. Sun, S. Kluckner, O. Lehmann, T. Chen, A. Hutter, S. Zakharov, H. Kosch *et al.*, "Depthsynth: Real-time realistic synthetic data generation from cad models for 2.5 d recognition," in *3D Vision (3DV), 2017 International Conference on*. IEEE, 2017, pp. 1–10.

[117] P. Mirowski, R. Pascanu, F. Viola, H. Soyer, A. J. Ballard, A. Banino, M. Denil, R. Goroshin, L. Sifre, K. Kavukcuoglu *et al.*, "Learning to navigate in complex environments," *arXiv preprint arXiv:1611.03673*, 2016.

[118] A. Das, S. Datta, G. Gkioxari, S. Lee, D. Parikh, and D. Batra, "Embodied question answering," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2018, pp. 2054–2063.

[119] K. Fang, A. Toshev, L. Fei-Fei, and S. Savarese, "Scene memory transformer for embodied agents in long-horizon tasks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 538–547.

[120] S. Gupta, J. Davidson, S. Levine, R. Sukthankar, and J. Malik, "Cognitive mapping and planning for visual navigation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 2616–2625.

[121] E. Parisotto and R. Salakhutdinov, "Neural map: Structured memory for deep reinforcement learning," *arXiv preprint arXiv:1702.08360*, 2017.

[122] J. F. Henriques and A. Vedaldi, "Mapnet: An allocentric spatial memory for mapping environments," in *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8476–8484.

[123] Y. Wu, Y. Wu, G. Gkioxari, and Y. Tian, "Building generalizable agents with a realistic and rich 3D environment," in *NIPS Deep Reinforcement Learning Symposium*, 2017.

[124] D. Chaplot, S. Gupta, A. Gupta, and R. Salakhutdinov, "Modular visual navigation using active neural mapping," 2019.

[125] D. Gordon, A. Kembhavi, M. Rastegari, J. Redmon, D. Fox, and A. Farhadi, "Iqa: Visual question answering in interactive environments," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4089–4098.

[126] J. Zhang, L. Tai, J. Boedecker, W. Burgard, and M. Liu, "Neural slam: Learning to explore with external memory," *arXiv preprint arXiv:1706.09520*, 2017.

[127] T. Chen, S. Gupta, and A. Gupta, "Learning exploration policies for navigation," *arXiv preprint arXiv:1903.01959*, 2019.

[128] A. Khan, C. Zhang, N. Atanasov, K. Karydis, V. Kumar, and D. D. Lee, "Memory augmented control networks," *arXiv preprint arXiv:1709.05706*, 2017.

[129] F. Sadeghi, "Divis: Domain invariant visual servoing for collision-free goal reaching," *arXiv preprint arXiv:1902.05947*, 2019.

[130] X. Ye, Z. Lin, H. Li, S. Zheng, and Y. Yang, "Active object perceiver: Recognition-guided policy learning for object searching on mobile robots," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 6857–6863.

[131] X. Ye, Z. Lin, J.-Y. Lee, J. Zhang, S. Zheng, and Y. Yang, "Gaple: Generalizable approaching policy learning for robotic object searching in indoor environment," *IEEE Robotics and Automation Letters*, 2019.

[132] E. Wijmans, S. Datta, O. Maksymets, A. Das, G. Gkioxari, S. Lee, I. Essa, D. Parikh, and D. Batra, "Embodied question answering in photorealistic environments with point cloud perception," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[133] 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 *European conference on computer vision*. Springer, 2014, pp. 740–755.

[134] D. B. Stéphane Ross, Geoffrey Gordon, "A reduction of imitation learning and structured prediction to no-regret online learning," *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, 2011.

[135] M. Savva, A. Kadian, O. Maksymets, Y. Zhao, E. Wijmans, B. Jain, J. Straub, J. Liu, V. Koltun, J. Malik *et al.*, "Habitat: A platform for embodied ai research," *arXiv preprint arXiv:1904.01201*, 2019.

[136] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "Orb-slam: a versatile and accurate monocular slam system," *IEEE transactions on robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[137] S. Song, F. Yu, A. Zeng, A. X. Chang, M. Savva, and T. Funkhouser, "Semantic scene completion from a single depth image," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1746–1754.

Biography

Georgios Georgakis received his Diploma of Engineering from the Technical University of Crete, Greece in 2012, and his Master of Science in Computer Science from George Mason University in 2015. He has spent time as research intern at Siemens Corporate Technology and United Imaging Intelligence. His research interests lie at the intersection of Computer Vision and Robotics and has worked on the topics of object detection and pose estimation, keypoint and descriptor learning, visual-based navigation, and human pose estimation.