DISSERTATION
submitted to
the Combined Faculty for the Natural
Sciences and Mathematics
of
Heidelberg University, Germany
for the degree of
Doctor of Natural Sciences

Put forward by
M.Sc. Omid Hosseini Jafari
Born in Tehran, Iran
Oral examination:

Exploring Subtasks of Scene Understanding: Challenges and Cross-Modal Analysis

Advisor: Prof. Dr. Carsten Rother

Acknowledgements

I would like to thank my supervisor, Prof. Dr. Carsten Rother, for his support and supervision. He made an interesting atmosphere in the lab and gave us the freedom to explore the things we want. Without his support, this thesis would not have been possible.

During my Ph.D., I was privileged to work with Prof. Andreas Geiger for one year. I would like to thank him for his support and supervision.

I would like to thank all the members of our lab in Heidelberg and Dresden for their constructive discussions and feedback. Particularly, I want to thank my magnificent colleges Hassan, Siva, Weihao, and Sid for making my Ph.D. life more fun.

I would also like to thank my parents, my brother, and my sister for their support and love. Most importantly, I would like to thank my wonderful wife, Sarah, who has stood by me all the time. Without her unlimited support and love, this thesis would not have been possible. And last but not least, I would like to thank my son for making our life magical.

Abstract

Scene understanding is one of the most important problems in computer vision. It consists of many subtasks such as image classification for describing an image with one word, object detection for finding and localizing objects of interest in the image and assigning a category to each of them, semantic segmentation for assigning a category to each pixel of an image, instance segmentation for finding and localizing objects of interest and marking all the pixels belonging to each object, depth estimation for estimating the distance of each pixel in the image from the camera, etc. Each of these tasks has its advantages and limitations. These tasks have a common goal to achieve that is to understand and describe a scene captured in an image or a set of images. One common question is if there is any synergy between these tasks. Therefore, alongside single task approaches, there is a line of research on how to learn multiple tasks jointly.

In this thesis, we explore different subtasks of scene understanding and propose mainly deep learning-based approaches to improve these tasks. First, we propose a modular Convolutional Neural Network (CNN) architecture for jointly training semantic segmentation and depth estimation tasks. We provide a setup suitable to analyze the cross-modality influence between these tasks for different architecture designs. Then, we utilize object detection and instance segmentation as auxiliary tasks for focusing on target objects in complex tasks of scene flow estimation and object 6d pose estimation.

Furthermore, we propose a novel deep approach for object co-segmentation which is the task of segmenting common objects in a set of images. Finally, we introduce a novel pooling layer that preserves the spatial information while capturing a large receptive field. This pooling layer is designed for improving the dense prediction tasks such as semantic segmentation and depth estimation.

Zusammenfassung

Das Szenenverstndnis ist eines der wichtigsten Probleme in der Bildverarbeitung. Es besteht aus vielen Unteraufgaben wie Bildklassifizierung zum Beschreiben eines Bildes mit einem Wort, Objekterkennung zum Finden und Lokalisieren von interessierenden Objekten im Bild und Zuweisen einer Kategorie zu jedem von ihnen, semantische Segmentierung zum Zuweisen einer Kategorie zu jedem Pixel eines Bildes , Instanzsegmentierung zum Finden und Lokalisieren von interessierenden Objekten und Markieren aller zu jedem Objekt gehrenden Pixel, Tiefenschtzung zum Schtzen des Abstands jedes Pixels im Bild von der Kamera usw. Jede dieser Aufgaben hat ihre Vor- und Nachteile. Diese Aufgaben haben das gemeinsame Ziel, eine in einem Bild oder einer Reihe von Bildern aufgenommene Szene zu verstehen und zu beschreiben. Eine hufig gestellte Frage ist, ob zwischen diesen Aufgaben Synergien bestehen. Daher gibt es neben Einzelaufgabenanstzen eine Reihe von Forschungsarbeiten zum gemeinsamen Lernen mehrerer Aufgaben.

In dieser Arbeit untersuchen wir verschiedene Teilaufgaben des Szenenverstndnisses und schlagen hauptschlich auf tiefem Lernen basierende Anstze vor, um diese Aufgaben zu verbessern. Zunchst schlagen wir eine modulare CNN-Architektur (Convolutional Neural Network) vor, mit der Tiefenschtzungs- und semantische Segmentierungsaufgaben gemeinsam trainiert werden knnen. Wir bieten ein Setup, das geeignet ist, den modalittsbergreifenden Einfluss zwischen diesen Aufgaben fr verschiedene Architekturdesigns zu analysieren. Anschlieend verwenden wir die Objekterkennung und Instanzsegmentierung als Hilfsaufgaben, um Zielobjekte in komplexen Aufgaben der Szenenflussschtzung und der Objekt-6d-Posenschtzung zu fokussieren.

Darber hinaus schlagen wir einen neuartigen tiefen Ansatz fr die Objekt-Co-Segmentierung vor, bei dem gemeinsame Objekte in einer Reihe von Bildern segmentiert werden. Schlielich fhren wir eine neuartige Pooling-Schicht ein, die die rumlichen Informationen bewahrt und gleichzeitig ein groes Empfangsfeld erfasst. Diese Pooling-Schicht dient zur Verbesserung der Aufgaben der dichten Vorhersage wie der semantischen Segmentierung und der Tiefenschtzung.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

In this chapter, we give a summary of scene understanding subtasks which are addressed in the thesis. Then we present the challenges in scene understanding that we cover in this thesis. Finally, we present the main contributions, a list of publication and the thesis outline.

1.1 Scene Understanding

Human visual perception is the most important factor to understand the scene around us. Therefore, computer vision is essential for designing a system to understand different aspects of a scene. Among different tasks in computer vision, scene understanding is one of the most interesting ones with many applications such as robotics and autonomous driving. Scene understanding itself consists of many subtasks. Each of these subtasks captures and describes a different aspect of a scene and the outcome of each subtask is necessary to achieve an overall understanding of a scene.

As an example, a set of questions can be raised to describe and summarize the scene in Fig. 1.1, including What kind of objects are existing in the scene? Where are they located in the 2D image? What are their 3D locations with respect to the camera? What is the precise boundary around each object in the image? Depending on the application, a lot of questions are desired to be answered and subtasks of scene understanding can be utilized. In the following, we present a summary of semantic and geometric tasks that are related to this thesis.

1.1.1 Semantic Tasks

In the following, we look at semantic tasks with different levels of detail. These tasks focus on semantic properties of the scene.

Figure 1.1: **Scene understanding.** Looking at a scene, there exists many different aspects to describe and understand it. Depending on the purpose, even we as humans will look at different sets of clues and features.
Photo source: "Abbey Road" by Iain Macmillan [1969].

Image classification: The goal of image classification task is to assign a category to an image. More formally given an image, we want to assign a class to the image from a list of predefined categories. This is a high level task in scene understanding. The question we are trying to answer in this task is:

"In one word, what are we looking at?",

for example, consider the class *"person"* for Fig. 1.2(d). However, there are some ambiguities in image classification. If the image is cluttered, it would sometimes even be hard for a human to assign a specific class to the image, as we can see in Fig. 1.2(a).

Object detection and localization: Given an image with multiple instances of predefined categories, the goal is to localize all the objects of interests in the image and classify each of them. As it is shown in Fig. 1.3, the objects are localized using bounding-boxes around them with different color for each category. This task answers the following questions:

"Which objects are existing in the scene?"
"Where are these objects located in the 2D image?"
"How many objects from each category are existing in the scene?"

For example in Fig. 1.3, seven people and nine cars are detected. Object detection provides more information about the scene compared to image classification task, i.e. it provides labeled overlapping bounding-boxes in the image instead of a single word.

(a) {crosswalk, street, people, cars} (b) {street, cars} (c) car (d) person

Figure 1.2: **Image classification.** The ideal input for image classification task contains a strong salient object in the center of the image (c and d). However in real world cases, there exists multiple objects in different locations inside the scene (a). In such cases, each region in the image can be classified as a different category which is not necessarily the same as the category for the whole image (b, c and d).

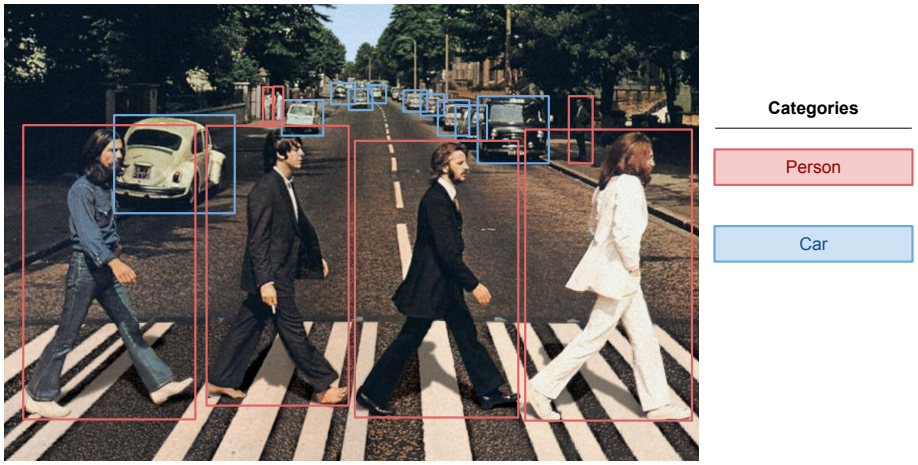


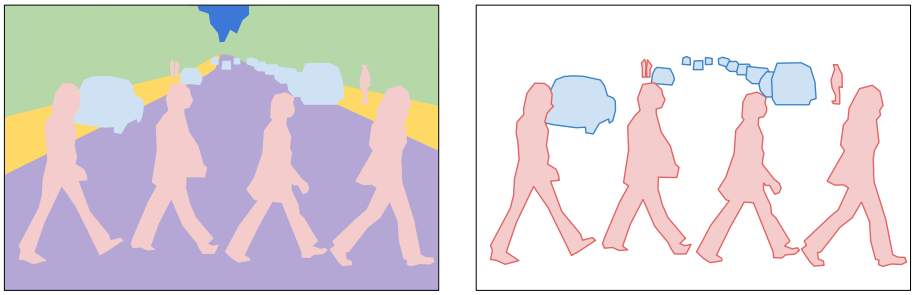
Figure 1.3: **Object detection.**

Semantic Segmentation: Given an image, the goal is to assign a category to each pixel of the image (see Fig. 1.4(a)). Similar to image classification and object detection the categories are predefined. Semantic segmentation task answers the following questions:

"Which categories are existing in the image?"
"What ratio of the image is filled by different categories?"

Similar to object detection, semantic segmentation also provides more information compared to the image classification task. They both discover existing

categories in the image. However in contrast to object detection, semantic segmentation task does not tell us how many instances of each category are in the image. On the other hand, semantic segmentation assigns a category to each region and pixel in the image, while in object detection task each region or pixel may be assigned to zero, one or more categories. In Fig. 1.3, the leftmost person (George Harrison) and the leftmost car have overlapping bounding boxes. Therefore, the intersection region has two classes (person and car). Due to mentioned limitations, the object detection task is incapable of finding a precise boundary around the objects while the semantic segmentation task is incapable of separating the instances. This leads us to a more interesting task called instance segmentation.



(a) semantic segmentation (b) instance segmentation

Figure 1.4: **Semantic segmentation and instance segmentation.**

Instance segmentation : Given an image with multiple instances of predefined categories similar to the object detection task, the goal is to localize all the objects of interests in the image and classify them. The difference is that the instance segmentation task localizes each object by masking the pixels belonging to the object (see Fig. 1.4(b)). This task answers the following questions:

"Which objects are existing in the scene?"
"Where are these objects located in the 2D image?"
"How many objects from each category are existing in the scene?"
"What is the precise boundary of each object in the image?"

Compared to semantic segmentation and object detection, instance segmentation is the best of both worlds. The instance segmentation task is particularly interesting for improving more complex and expensive tasks in scene understanding due to its object masks. In this thesis, we explore instance-aware approaches to solve and improve complex tasks such as scene flow estimation and object 6D pose estimation.

1.1.2 Geometric Tasks

In this section, we look at geometric tasks. These tasks focus on physical properties of the scene.

Depth estimation : The aim of this task is to find the distance of each pixel in the image from the camera (see Fig. 1.5). Similar to semantic segmentation, the depth estimation is a pixel level task and it provides no information regarding individual objects existing in the scene.

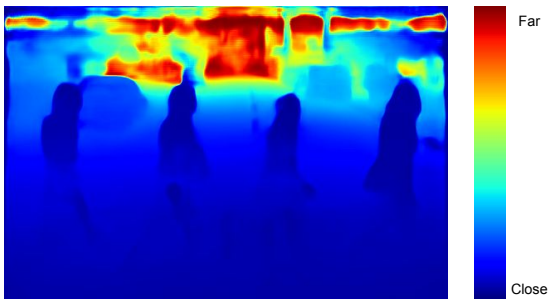


Figure 1.5: **Depth estimation.**

Object 3D pose estimation: Given an image including multiple objects of interest, the goal is to find the 3D location of the object with respect to the camera (see Fig. 1.6). This task can be defined in a different ways depending on how to localize the objects in 3D (e.g. with 3D bounding boxes or 3D mesh of specific objects), or what extra inputs are available (e.g. depthmap or groundplane). This task answers the following questions:

"Which objects are present in the scene?"
"Where are these objects located in 3D camera coordinate system?"
"What is the distance between each pair of objects?"

1.2 Scene Understanding Challenges

Joint learning of multiple tasks: One possible argument can be that all of the mentioned tasks contribute to one central goal of scene understanding. In the last decade a lot of research effort has focused on solving individual tasks as well as possible. While it is certainly important to gauge the limits of individual tasks, various researchers have recently raised the question of whether the next big step forward can be achieved by focusing on improving single tasks or by considering different tasks in a joint fashion. This question is particularly

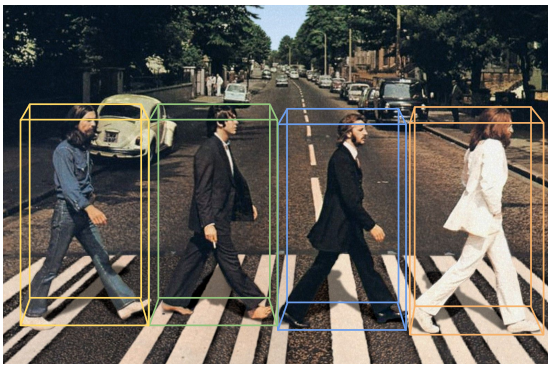


Figure 1.6: **Object 3D pose estimation.** An example of estimating 3D pose of pedestrians in the scene. Each pose is presented by a 3D bounding box around the target pedestrian.

emphasized in robotics setups where the coordination of multiple tasks and consolidation of various predictions is constitutive.

Recently, deep learning methods have shown significant improvement on solving scene understanding tasks. However, designing a network architecture to jointly learn and solve multiple tasks is challenging. One factor is the number of parameters or the capacity of the network. A low capacity network performs poorly while a very high capacity network may decouple the learning of tasks. The other factor is how to fuse the features learned from different tasks. Improper fusion may result in negative effect of one task on the others.

Feature matching: Feature matching is a challenging problem in complex tasks with multiple inputs, such as finding the matching points:

- in temporal inputs for optical flow estimation
- in stereo images for disparity estimation
- between point-cloud and 3D-CAD model for object 3D pose estimation.

The main challenges in feature matching are existence of the large search space for matching the pairs from all over the inputs, and mismatches due to repeating structures in the scene.

Object co-segmentation: Finding and segmenting common objects in a group of images is an interesting and challenging task in scene understanding. Object co-segmentation aims to segment all common objects in a set of given images (see Fig. 1.7). This task is specially useful in cases that the target object is not predefined and the goal is to understand and learn the appearance and structure of the object category from unlabeled data.

Dense prediction tasks: Recently, deep learning has improved many tasks of computer vision significantly. Convolutional Neural Networks (CNNs) are

Figure 1.7: **Co-segmentation.** The goal is to segment all common objects in input images. In this example koalas are the common objects. These results are generated by our approach in chapter 5.

the most successful architectures for computer vision tasks. The main wining factor of these networks is their large receptive field thanks to the combination of convolution layers and pooling layers. These networks are originally designed for image classification task. They map the input image into high level features and normally the spatial dimension of these features are downscaled with a factor of 32. These coarse features can perfectly summarize the information in the image and are suitable for image level classification. After the success of these networks in image classification, they are adapted for dense prediction tasks such as semantic segmentation, depth estimation, object detection. However, due to their spatial information loss, the performance of these networks on dense prediction tasks is not as well as on image classification.

1.3 Contribution

The main contribution of this thesis is to improve and analyze different aspects of scene understanding in three main directions: multiple tasks cooperation, common-object segmentation, and dense prediction tasks.

Multiple tasks cooperation: As mentioned in the previous section, there are many tasks for understanding a scene and each of these tasks has its advantages and limitations. In this thesis, we explore the benefits of combining multiple tasks, as summarized below.

- We addresses the task of designing a modular neural network architecture that jointly solves different tasks. As an example we use the tasks of depth estimation and semantic segmentation given a single RGB im-

age. The main focus of this work is to analyze and achieve the so-called synergy effect between the different tasks. While most previous works solely focus on measuring improvements in accuracy, we are the first to quantify the synergy effect. We show that there is a relationship between accuracy and the synergy effect, although not a simple linear one. Hence a larger synergy effect does not necessarily mean an improved accuracy. This relationship can be utilized to understand different design choices in network architectures. Towards this end we propose a shallow Convolutional Neural Network (CNN) architecture that fuses the state of the state-of-the-art results for depth estimation and semantic labeling. By doing so we achieve improved results for both tasks using the NYU-Depth v2 benchmark.

- Existing methods for scene flow estimation often fail in the presence of large displacement or local ambiguities, e.g. texture-less or reflective surfaces. However, these challenges are omnipresent in dynamic road scenes, which is the focus of this work. Our main contribution is to overcome these 3D motion estimation problems by exploiting recognition. In particular, we investigate the importance of recognition granularity, from coarse 2D bounding box estimates over 2D instance segmentations to fine-grained 3D object part predictions. We compute these cues using CNNs trained on a newly annotated dataset of stereo images, and integrate them into a slanted-plane CRF model for robust 3D scene flow estimation - an approach we term Instance Scene Flow. We analyse the importance of each recognition cue in an extensive ablation study and observe that the instance segmentation cue is by far strongest, in our setting. We demonstrate the effectiveness of our method on the challenging KITTI 2015 scene flow benchmark where we achieve state-of-the-art performance at time of submission.
- We address the task of 6D pose estimation of known rigid objects from single input images in scenarios where the objects are partly occluded. Recent RGB-D-based methods are robust to moderate degrees of occlusion. For RGB inputs, no previous method works well for partly occluded objects. Our main contribution is to present the first deep learning-based system that estimates accurate poses for partly occluded objects from RGB-D and RGB input. We achieve this with a new instance-aware pipeline that decomposes 6D object pose estimation into a sequence of simpler steps, where each step removes specific aspects of the problem. The first step localizes all known objects in the image using an instance segmentation network, and hence eliminates surrounding clutter and occluders. The second step densely maps pixels to 3D object surface positions, so called object coordinates, using an encoder-decoder network, and hence eliminates object appearance. The third, and final, step predicts the 6D pose using geometric optimization. We demonstrate that we significantly outperform the state-of-the-art for pose estimation of partly occluded objects for both RGB and RGB-D input.

Common object segmentation: We present a deep object co-segmentation (DOCS) approach for segmenting common objects of the same class within a pair of images. This means that the method learns to ignore common, or uncommon, background *stuff* and focuses on common *objects*. If multiple object classes are presented in the image pair, they are jointly extracted as foreground. To address this task, we propose a CNN-based Siamese encoder-decoder architecture. The encoder extracts high-level semantic features of the foreground objects, a mutual correlation layer detects the common objects, and finally, the decoder generates the output foreground masks for each image. To train our model, we compile a large object co-segmentation dataset consisting of image pairs from the PASCAL dataset with common objects masks. We evaluate our approach on commonly used datasets for co-segmentation tasks and observe that our approach consistently outperforms competing methods, for both seen and unseen object classes.

Dense prediction tasks: There are a variety of approaches to obtain a vast receptive field with convolutional neural networks (CNNs), such as pooling or striding convolutions. Most of these approaches were initially designed for image classification and later adapted to dense prediction tasks, such as semantic segmentation. However, the major drawback of this adaptation is the loss of spatial information. Even the popular dilated convolution approach, which in theory is able to operate with full spatial resolution, needs to subsample features for large image sizes in order to make the training and inference tractable. In this work, we introduce Split-Merge pooling to fully preserve the spatial information without any subsampling. By applying Split-Merge pooling to deep networks, we achieve, at the same time, a very large receptive field. We evaluate our approach for dense semantic segmentation of large image sizes taken from the Cityscapes and GTA-5 datasets. We demonstrate that by replacing max-pooling and striding convolutions with our split-merge pooling, we are able to improve the accuracy of different variations of ResNet significantly.

1.4 Publications

The thesis is based on the following publications:

1. **Analyzing Modular CNN Architectures for Joint Depth Prediction and Semantic Segmentation**[1]
   Omid Hosseini Jafari, Oliver Groth, Alexander Kirillov, Michael Ying Yang, Carsten Rother
   ICRA 2017 (Oral Presentation)
2. **Bounding Boxes, Segmentations and Object Coordinates: How Important is Recognition for 3D Scene Flow Estimation in Autonomous Driving Scenarios?**[2]
   Aseem Behl*, Omid Hosseini Jafari*, Siva Karthik Mustikovela*, Hassan Abu Alhaija, Carsten Rother, Andreas Geiger
   ICCV 2017 (Poster Presentation)

(* equal contribution)
Declaration: The original idea of this work proposed jointly. We developed and improved the original idea together. I particularly focused on dataset analysis/preparation/generation and object coordinate estimation network design/implementation. The experimental setup is conducted in a joint discussion and we wrote the paper together.

3. **iPose: Instance-Aware 6D Pose Estimation of Partly Occluded Objects**[3]
   Omid Hosseini Jafari*, Siva Karthik Mustikovela*, Karl Pertsch, Eric Brachmann, Carsten Rother
   ACCV 2018 (Poster Presentation)
   (*equal contribution)
   Declaration: I and Siva Mustikovela came up with the original idea in a joint discussion. Siva focused mostly on ICP refinement of the predicted poses for RGB-D setup and I focused on the rest of the pipeline, i.e. dataset preparation and generation, instance segmentation, object coordinate estimation and pose refinement of RGB setup. The experimental setup is conducted in a joint effort and we wrote the paper together.
4. **Deep Object Co-Segmentation**[4]
   Weihao Li*, Omid Hosseini Jafari*, Carsten Rother
   ACCV 2018 (Poster Presentation)
   (*equal contribution)
   Declaration: Weiaho Li proposed the original idea. The idea improved and developed in a joint discussions of me and Weihao. I mainly focused on design and implementation side of the approach. The experimental setup is conducted in a joint effort and we wrote the paper together.
5. **Split Merge Pooling**[5]
   Omid Hosseini Jafari, Carsten Rother
   arxiv 2020

I also contributed to the following publications during my PhD but these works are not discussed in the thesis:

1. **Real-time RGB-D based template matching pedestrian detection** [6]
   Omid Hosseini Jafari, Michael Ying Yang
   ICRA 2016 (Oral Presentation)
2. **Semantic-Aware Image Smoothing**[7]
   Weihao Li, Omid Hosseini Jafari, Carsten Rother
   VMV 2017
3. **Localizing Common Objects Using Common Component Activation Map**[8]
   Weihao Li, Omid Hosseini Jafari, Carsten Rother
   CVPR 2019 Explainable AI Workshop

1.5 Thesis Outline

In chapter 2, we introduce a modular network for improving two tasks of semantic segmentation and depth estimation. In chapter 3, we introduce an instance-aware scene flow approach. In chapter 4, we propose an instance-aware object 6D pose estimation method for partially occluded objects. In chapter 5, we introduce a novel deep learning approach to address the problem of object co-segmentation. In chapter 6, we introduce a novel pooling layer that preserves the spatial information and improves the performance of dense prediction tasks. Finally, in chapter 7, we discuss the possible future work directions.

Chapter 2

Analyzing Modular CNN Architectures for Joint Depth Prediction and Semantic Segmentation

2.1 Introduction

Machine perception is an important and recurrent theme in the robotics and computer vision community. Computer vision has contributed a broad range of tasks to the field of perception, such as estimating physical properties from an image, e.g. depth, motion, or reflectance, as well as estimating semantic properties, e.g. labeling each pixel with a semantic class. One may argue that all of these tasks contribute to one central goal, which can be broadly described as "holistic scene understanding". In the last decade a lot of research effort has focused on solving individual tasks as good as possible. While it is certainly important to gauge the limits of individual tasks, various researchers have recently raised the question of whether the next big step forward can be achieved by focusing on improving single tasks or by considering different tasks in a joint fashion, e.g. [9].[1] This question is particularly emphasized in robotics setups where the coordination of multiple tasks and consolidation of various predictions is constitutive. In this work we focus on the question of *"How to analyze and exploit the cross-modality influence between depth and semantic predictions in order to solve tasks jointly."*. While the idea of a "beneficial influence between different tasks" is not new, it has in our opinion not received enough attention. This is in contrast to other fields, such as neuroscience, psychology and machine learning. In principle, there are two different ways

[1] See for example the recent workshop on "Recognition meets Reconstruction", where one aim is to solve two tasks jointly.

to consider multiple tasks in a joint framework. One option is to have one big, joint model. For example this can be a neural network which has a single RGB image I as input and outputs a semantic segmentation S and a depth labeling D; or a graphical model which represents $p(S, D|I)$. While this is a popular choice and many impressive results have been achieved, e.g. [10] (for depth, semantic segmentation and more) or [11] (for depth, surface reflectance and lighting), it has its drawbacks. Firstly, the models become rapidly complex and are hence rarely used in follow-up works. Secondly, it is very difficult to analyze whether there is indeed a beneficial influence between different tasks. For instance, as we will see later, a joint model may have no interdependency between modalities and tasks and can in fact be considered as two separate models. The second possible approach for solving multiple tasks jointly is to follow a modular design. In this work, we pursue this option. We propose a simple modular design where individual tasks are first inferred separately and then fed into our *joint refinement network* (see Fig. 2.1). The aim of this network is to leverage a beneficial cross-modality influence between the soft (probabilistic) input modalities in order to jointly refine both task outputs. We show experimentally that there is indeed a relation between the cross-modality influence and an improvement in accuracy for each individual task. However, the relation is not linear, i.e. a larger cross-modality influence does not necessarily mean higher accuracy.

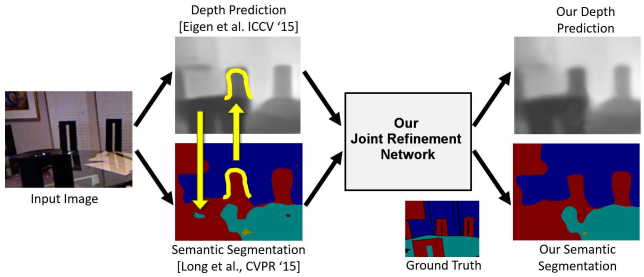


Figure 2.1: **Example processing flow of our joint refinement network**. A single RGB image is first processed separately by two state-of-the-art neural networks for depth estimation and semantic segmentation. The two resulting predictions contain information which can mutually improve each other: (1) yellow arrow from depth to semantic segmentation means that a smooth depth map does not support an isolated region (cyan means furniture); (2) yellow arrow from semantic segmentation to depth map means that the exact shape of the chair can improve the depth outline of the chair. (3) In most areas the two modalities positively enforce each other (e.g. the vertical wall (dark blue) supports a smooth depth map. The *cross-modality influences* between these two modalities are exploited by our joint refinement network, which fuses the features from the input prediction maps and jointly processes both modalities for an overall prediction improvement. *(Best viewed in color.)*

While such a modular design is not as rich as a complex joint model, it brings many advantages: (i) New modalities can be easily integrated. For instance a module that estimates the reflectance properties can be integrated. (ii) We can quantify the cross-modality influence between different modalities, as discussed in detail later. (iii) It is easier to train all the tasks, in contrast to a full joint model. For example, in practice we often have many training images for individual modalities but fewer training images for all modalities jointly. A joint model would have to be trained in a semi-supervised fashion in order to cope with such heterogeneous data, while in a modular architecture each module is trained with the applicable training data. (iv) Since in our case each module, i.e. for the individual task and the joint refinement, is realized in the form of Convolution Neural Networks (CNNs), it is possible to conduct end-to-end training.

The advantages of modular architectures are not new and indeed David Marr describes it properly in his book ([12] page 102): "This principle [of modular design] is important because if a process is not designed in this way, a small change in one place has consequences in many other places. As a result, the process as a whole is extremely difficult to debug or to improve, whether by a human designer or in the course of natural evolution."

To summarize, our **main contributions** are threefold:

- For both tasks, semantic segmentation and depth estimation, we improve on the state-of-the-art results for the NYU-Depth v2 benchmark [13]. We achieve this by proposing a new joint refinement network which takes as input the results of the current state-of-the-art networks for the individual tasks.
- For modular architecture designs we propose an experimental setup to measure the cross-modality influence quantitatively. Such experiments are well-known in neuroscience, but have not yet been used in computer vision or robotics, to the best of our knowledge.
- We analyze different network designs with respect to their cross-modality influence and show that there is indeed a relationship between the cross-modality influences and tasks performances. Although not linear, this relationship can be used to understand different design choices in network architectures.

2.2 Related Work

A large body of work in computer vision has focused on the two separate problems of semantic segmentation and depth estimation. In the review below, we focus on techniques that specifically address multi-modal architectures or perform semantic segmentation and depth estimation from a single monocular image.

Single tasks. Conditional Random Fields (CRFs) are popular models that have been used in both depth estimation task, e.g. [14, 15, 16, 17, 18, 19],

and semantic segmentation task, e.g. [20, 21]. Such approaches predominantly use hand-crafted features. Recently, convolutional neural networks (CNNs) are driving advances in computer vision, such as for image classification [22], object detection [23, 24], recognition [25, 26], semantic segmentation [27, 28], pose estimation [29] and depth estimation [30]. The success of CNNs is attributed to their ability to learn rich feature representations as opposed to hand-designed features. Eigen et al. [30] trained multi-scale CNNs for depth map prediction from a single image. Liu et al. [31] propose deep convolutional neural fields for depth estimation, where a CRF is used to explicitly model the relations of neighboring superpixels, and the potentials are learned in a unified CNN framework. Eigen and Fergus [32] extend their previous method [30] to predict depth, surface normals and semantic labels sequentially with a common multi-scale CNN. A number of recent approaches, including recurrent CNNs (R-CNNs) [33] and fully convolutional networks (FCN) [28] have shown a significant boost in accuracy by adapting state-of-the-art CNN-based image classifiers to the semantic segmentation problem. Pinheiro and Collobert [33] present a feed-forward approach for scene labeling based on an R-CNN. The system is trained in an end-to-end manner over raw pixels and models complex spatial dependencies with low computational cost. FCNs [28] address the coarse-graining effect of the CNN by upsampling the feature maps in deconvolution layers and combining fine-grained and coarse-grained features during prediction.

Joint models. Joint models of multiple tasks have been exploited in the computer vision literature to a certain extent, e.g. joint image segmentation and stereo reconstruction [34, 35, 36], joint object detection and semantic segmentation [37], joint instance segmentation and depth ordering [38], as well as joint intrinsic image, objects, and attributes estimation [10]. However, joint semantic segmentation and depth estimation from a single image has been rarely addressed, with a few exceptions [39, 40]. These works explicitly reason about class segmentation as well as depth estimation from a single image. Ladicky *et al.* [39] jointly trained a canonical classifier considering both the loss from semantic and depth labels of the objects. However, they use local regions with hand-crafted features for prediction, which is only able to generate very coarse depth and semantic maps. Wang et al. [40] formulate the joint inference problem in a two-layer Hierarchical Conditional Random Field (HCRF). The unary potentials in the bottom layer are pixel-wise depth values and semantic labels, which are predicted by a CNN trained globally using the full image, while the unary potentials in the upper layer are region-wise depth and semantic maps which come from another CNN-based regressor trained on local regions. The mutual interactions between depth and semantic information are captured through the joint training of the CNNs and are further enforced in the joint inference of HCRF. They consider an alternating optimization strategy by minimizing one, fixing the other. In contrast, our model performs full joint inference.

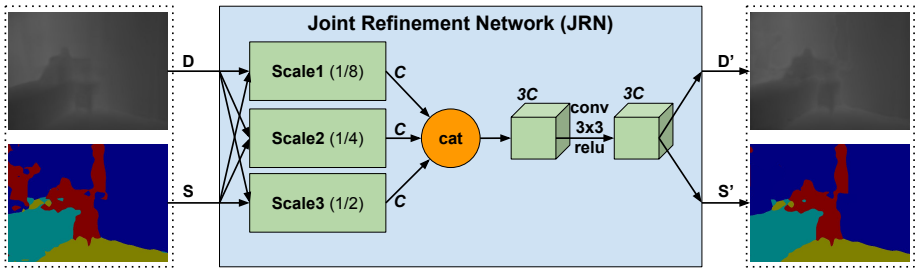


Figure 2.2: **Overall Network.** JRN receives as input the predictions of two independent single modalities: Depth and semantic labeling. Inspired by [32] the inputs are considered at different scales (1/8, 1/4 and 1/2 of the total image resolution) in order to capture different levels of details. C is the number of output feature channels from each scale branch (see 2.3.2). After processing the three scale branches, the computed features are concatenated, convolved and then mapped to the two respective output maps.

Multi-modal learning and representation. Many different communities have addressed the problem of multi-modal learning and representation, such as machine learning [41, 42, 43], human-computer interaction [44, 45], and neuroscience [46, 47]. In [41], the authors present a series of tasks for multi-modal learning and show how to train deep networks that learn features to address these tasks. In particular, they demonstrate cross modality feature learning, where better features for downstream classification tasks are learned from a video if both audio and video signals are present during the feature learning stage. While [41] deals with an unsupervised feature learning, our approach uses supervised learning. Furthermore, unlike [41] we perform an analysis on the effect of different network architectures on the cross-modality influence. Similarly, in the neuroscience community, the authors of [46] investigated the influence of the *face-benefit* in speech and speaker recognition. Apparently, people who have heard the voice and seen the face of a speaker during training time are more likely to recognize both the speaker and the spoken words from recorded audio only during test time. Additionally, [47] revisited the *face-benefit* experiment and showed a joint audio-visual processing by the brain for the classification task, indicating a joint feature representation is key to superior performance. Canonical correlation analysis (CCA) [48] is the de-facto approach for learning a common representation of two different modalities (so-called views) in the machine learning literature. Deep CCA, a deep learning version of CCA, is introduced in [49]. It aims at learning a complex non-linear transformation of two views such that the resulting representation is highly correlated. It can be considered as a non-linear extension of the linear CCA.

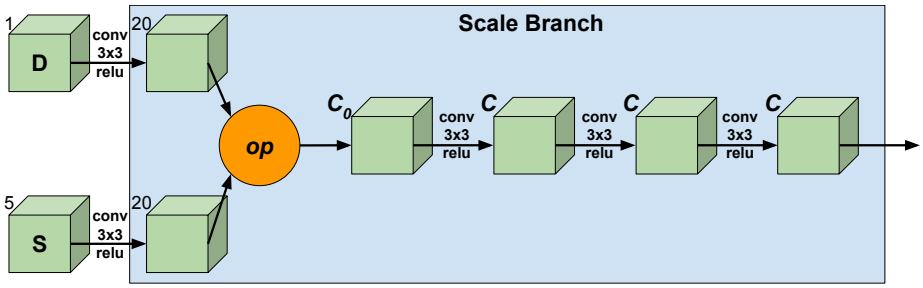


Figure 2.3: **A Scale Branch.** On each scale there are first 20-dimensional feature vectors extracted by performing 3x3 convolutions on each input modality. Immediately thereafter these modality features are fused by operation ***op***. We also consider the number of channels $\boldsymbol{C_0}$ after the fusion operation as a network design variable, which affects the cross-modality influence. The subsequent channel number $\boldsymbol{C}$ is 60 by default (`Cat60`, `Sum60`). Based on ***op***, $\boldsymbol{C_0}$ and $\boldsymbol{C}$ we design five different network architectures and analyze their properties in Sec. 2.4.2.

2.3 Joint Refinement Network

In this section, we present the details of the CNN architecture which we used to predict jointly the depth map and the semantic labeling. We also discuss different architectural design decisions and their relation to the cross-modality influence between two modalities.

2.3.1 Network Architecture

Our network decomposes into two parts: (i) independent *single-modality* models that output predictions for each modality separately and (ii) our joint refinement network (JRN) that takes as input these prediction maps and outputs refined predictions of all modalities. Our model does not have any constraints on the choice of *single-modality* models. In order to capture dependencies on different scales [32], we employ a multi-scale architecture for JRN, as illustrated in Fig. 2.2. It has three *scale-branches* `Scale1`, `Scale2` and `Scale3` that work with different scales of the input and have the same architecture, described in Fig. 2.3. On each scale, 20-dimensional features are extracted by performing 3×3 convolutions on each input modality. After each convolutional layer, a ReLU non-linearity is used. In the next section, we consider different architecture designs for the combination.

2.3.2 JRN Variants

In order to analyze and understand the cross-modality influence, we design different variations of JRN. Then we measure the cross-modality influence for

designed JRN architecture using the approach which we propose in section 2.3.4.

Concatenation. In the *concatenation* architecture we have chosen concatenation operations for ***op*** in all scale branches consistently (cf. Fig. 2.3). Therefore, the number of channels $\boldsymbol{C_0}$ after concatenation has doubled to 40. The simple concatenation of the single-modality features aims at achieving cross modality features because the single-modality features are subsequently convolved together. We call this variant `Cat60`. Since concatenation is the weakest form of our possible feature interactions, we subsequently refer to this model as featuring a *loose computation*.

Summation. The *summation* architecture features an element-wise sum as fusion operation ***op*** on each scale. Since the channels are summed up during fusion, the number of subsequent channels $\boldsymbol{C_0}$ is equal to 20. We call this variant `Sum60`. In contrast to the `Cat60` architecture this network forces a joint feature representation. We refer to this architecture as having a *coupled computation* later on.

Channel Squeezing. Besides summation of input features we also consider the number of channels used after fusion as an influencing factor on the cross-modality influence and the overall network performance. We hypothesize that fewer channels would generally enforce network to take into account both modalities more during training of the convolutional layers. Directly after the concatenation (***op***) of the input features, which yields a 40-channel feature, the number of channels $\boldsymbol{C}$ can be 10, 5 or 1. This gives us three more network variations `Cat10`, `Cat5` and `Cat1`. We still classify the `Cat10` network as *loose computation* whereas the `Cat5` and `Cat1` networks feature *coupled computation*.

2.3.3 JRN Training

We define D' and S' as the depth and semantic label prediction maps and D^* and S^* as the respective ground truth maps. D' and D^* are maps assigning a depth value in the range of $[0.0, 10.0]$ meters to every pixel. S' and S^* are k-channel maps, assigning a probability distribution over k semantic classes to each pixel. We restrict the loss computation to n valid pixels where we have both a depth value and a semantic label as ground truth. The loss function for JRN is a simple summation of single-task losses:

$$L_{joint}(D', S', D^*, S^*) = \underbrace{\frac{1}{n}\sum_i \frac{(D'_i - D^*_i)^2}{D^*_i}}_{L_{depth}} - \underbrace{\frac{1}{n}\sum_i S^*_i \log(S'_i)}_{L_{semantic}} \tag{2.1}$$

The depth loss L_{depth} is a relative quadratic distance between prediction and ground truth map. For the semantic loss $L_{semantic}$ we use the cross-entropy loss, where $S'_i = \exp[z_i]/\sum_s \exp[z_{i,s}]$ is the class prediction at pixel i given the semantic output slice z of the last convolutional layer of JRN. During training we

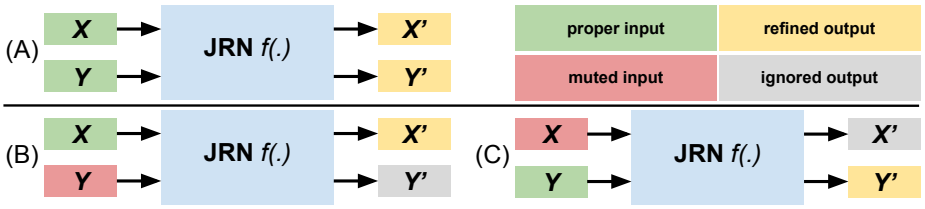


Figure 2.4: **Cross-Modality Influence Test.** In order to quantify the cross-modality influence numbers we need to consider three different inference setups: (A) Predicting both outputs X' and Y' from two proper input maps X and Y: $(X', Y') = f(X, Y)$; (B) Muting the input channel Y and computing X' and Y': $(X', Y') = f(X)$; and (C) Muting the input channel X and computing X' and Y': $(X', Y') = f(Y)$.

keep the single-task networks fixed and use their predictions as inputs to JRN. In the future we plan to also perform end-to-end training of all networks, single task networks and JRN. The internal weights of JRN are initialized randomly. We train JRN jointly on both tasks with the standard NYU-Depth v2 [13] train-test split.

2.3.4 Quantifying the Cross-Modality Influence

Inspired by the "face-benefit" experiment in [46] we propose an evaluation proxy to measure the influence of a modality on the final model performance and quantify it in a so-called *influence number*. During training time the JRN is trained to jointly predict two outputs X' and Y' from two input modalities X and Y. During inference time we consider three different measurement setups as explained in Fig. 2.4. The *performance* of the JRN predicts a particular modality X' which is measured by a function A_X (e.g. "mean IOU" for X being semantic labeling or "rms(linear)" for X being a depth map).
The cross-modality influences between the modalities are *directional* and dependent on a particular JRN architecture (represented by its transformation function $f(.)$) as well as two performance functions A_X and A_Y. The cross-modality influence of input modality Y on the performance of the prediction of X' measured under A_X is defined as:

$$\omega_{Y \to X'} = A_X(f(X, Y)) - A_X(f(X)). \tag{2.2}$$

Consequently the complementary influence from X to Y' is defined as:

$$\omega_{X \to Y'} = A_Y(f(X, Y)) - A_Y(f(Y)). \tag{2.3}$$

Note that those influences are *not necessarily symmetric*. We compute the influence values in the setup of a joint semantic segmentation (X) and depth

estimation (Y) in Sec. 2.4.3 and analyze the relationship between the cross-modality influences and the model performance.

2.4 Experiments

In this section, we introduce the dataset, and then describe details of our implementation. After that we present a quantitative and qualitative comparison. We conclude with a cross-modality influence analysis.

2.4.1 Experimental setup

Dataset. We evaluate our proposed method on the NYU-Depth V2 dataset [13]. This dataset consists of 1449 RGBD images of indoor scenes, among which 795 are used for training and 654 for test (we use the standard train-test split provided with the dataset). Following [40], we also map the semantic labels into five categories conveying strong geometric properties, i.e. Ground, Vertical, Ceiling, Furniture and Object, as it is shown in Fig. 2.5.

Implementation details. We use two state-of-the-art single modality CNN models for providing the input depth-map and semantic segmentation prediction map. For depth input, we use the inference code with pretrained model of Eigen *et al.* [32] which is publicly available[2]. For semantic segmentation prediction maps, we use the FCN model of Long *et al.* [28].

We implement our network in Caffe framework [50]. We train joint refinement networks (`Cat60`, `Cat10`, `Cat5`, `Cat1` and `Sum60`) with 795 training images from NYU-Depth V2 dataset [13] using SGD solver with batches of size one. The learning rate is 0.001 for all the convolutional layers and the momentum is 0.9. The global scale of the learning rate is tuned to a factor of 5. Depending on the different architectures and the number of channels in scale branches, training JRN took 5 to 6 hours using an NVidia GTX Titan X. We pass the absolute depth maps and the semantic prediction maps to JRN.

Evaluation metrics. To evaluate the semantic segmentation, we take Intersection over Union (IOU) and pixel accuracy percentage as metrics. For the depth estimation task we use several measures, which are also commonly used in prior works [17, 30]. Given the predicted depth value of a pixel d_i and the ground truth depth d_i^*, the evaluation metrics are:

- Abs relative error (rel): $\frac{1}{N}\sum_i \frac{|d_i^* - d_i|}{d_i^*}$;
- Squared relative error (rel(sqr)): $\frac{1}{N}\sum_i \frac{|d_i^* - d_i|^2}{d_i^*}$;
- Average $\log_{10}$ error (log10): $\frac{1}{N}\sum_i |\log_{10} d_i^* - \log_{10} d_i|$;

[2] `http://www.cs.nyu.edu/~deigen/dnl/`

	Error (Depth) (lower is better)					Accuracy (Depth) (higher is better)			Accuracy (Seg.) (higher is better)	
	rel	rel(sqr)	log10	rms(linear)	rms(log)	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$	**Mean IOU**	**Pix.Acc.**
Input [32]&[28]	0.158	0.125	0.070	0.687	0.221	0.751	0.946	0.987	53.284	72.268
Cat60	0.158	0.124	0.0686	0.678	0.218	0.760	0.947	0.987	**54.206**	72.957
Sum60	**0.157**	**0.123**	**0.068**	0.673	**0.216**	**0.762**	**0.948**	**0.988**	54.184	**72.967**
Cat10	0.158	0.125	0.069	0.681	0.219	0.756	0.946	0.987	54.080	72.953
Cat5	0.160	0.125	0.068	0.670	0.218	**0.762**	0.946	0.986	54.120	72.952
Cat1	0.161	0.126	0.069	**0.669**	0.219	0.759	0.946	0.987	53.989	72.864

Table 2.1: **Comparison of different JRN architectures.** We compare our different JRN networks with each other and also with our input single modality networks for depth [32] and semantic segmentation [28]. Best results are shown in bold.

	Error (lower is better)					Accuracy (higher is better)		
	rel	rel(sqr)	log10	rms(linear)	rms(log)	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Eigen et al. [30]	0.215	0.212	-	0.907	0.285	0.611	0.887	0.971
Joint HCRF [40]	0.220	0.210	0.094	0.745	0.262	0.605	0.890	0.970
Eigen & Fergus [32]	0.158	0.125	0.070	0.687	0.221	0.751	0.946	0.987
Liu et al. [31]	0.213	-	0.087	0.759	-	0.650	0.906	0.976
Ours (Sum60)	**0.157**	**0.123**	**0.068**	**0.673**	**0.216**	**0.762**	**0.948**	**0.988**

Table 2.2: **Depth comparison.** Baseline comparisons of depth estimation on the NYU-Depth v2 dataset. Our method outperforms state-of-the-art methods.

- Root mean squared error (rms(linear)): $\sqrt{\frac{1}{N}\sum_i (d_i^* - d_i)^2}$;
- Root mean squared error (rms(log)): $\sqrt{\frac{1}{N}\sum_i |\log d_i^* - \log d_i|^2}$;
- Accuracy with threshold thr: percentage (%) of $d_i \; s.t. \max(\frac{d_i^*}{d_i}, \frac{d_i}{d_i^*}) = \delta < thr$, where $thr \in \{1.25, 1.25^2, 1.25^3\}$;

2.4.2 Comparison of Results

We first compare our five different JRN architectures described in Sec. 2.3.2 with each other (see Table 2.1). We observe that none of the networks consistently outperforms all others in all metrics. However, the `Sum60` network is nearly the best for all metrics. Hence we chose it for comparison with other models from related work.

For *depth estimation*, we compare our results with four most recent methods, i.e. Eigen et al. [30], Joint HCRF [40], Eigen & Fergus [32], and Liu et al. [31]. Table 2.2 shows the quantitative results from all the algorithms. Our JRN network consistently outperforms all the state-of-the art algorithms in all metrics. The main difference to the models of Eigen *et al.* [30], Eigen & Fergus [32] and Liu et al. [31] is that they only deal with the depth estimation task, and hence cannot exploit cross-modality influence. However, the Joint HCRF [40] also jointly predicts a depth map and semantic labeling, yet our model outper-

	Ground	Vertical	Ceiling	Furniture	Object	Mean IOU	Pix.Acc.
Semantic HCRF [40]	61.84	66.344	15.977	26.291	43.121	42.715	69.351
Joint HCRF [40]	63.791	66.154	20.033	25.399	45.624	44.2	70.287
FCN16s NYU-5 [28]	66.578	67.354	46.351	35.71	50.429	53.284	72.268
Ours (Sum60)	**67.87**	**68.707**	**48.166**	**35.82**	**50.770**	**54.267**	**73.035**

Table 2.3: **Semantic segmentation comparison.** Class-wise results reporting mean IOU and pixel-wise accuracy for semantic segmentation on NYU-Depth v2 with five classes. Best results are shown in bold.

forms theirs by a large margin both in depth estimation (8.7% rel(sqr) decrease) and in semantic segmentation (10% mean IOU increase, see Table 2.3). We evaluate the published predicted depth maps from Eigen & Fergus [32] and Eigen et al. [30] with our evaluation script, and for Eigen et al. [30] we obtain the same reported numbers. However, we do not obtain the same numbers for [32] as reported. Our goal in this work is to improve the performance of the input predictions. Therefore, this comparison is fair since we use the same evaluation script for the input and the output of our network.

For *semantic segmentation*, we compare two recent methods: Our baseline FCN [28] and Joint HCRF [40]. Results are shown in Table 2.3. We outperform the other methods for all five classes. Compared with the baseline FCN [28] our method is 1% better in mean IOU.

Qualitative results of both tasks are shown in Fig. 2.5. Even though our method does not use superpixels or any explicit CRF model, it tends to produce large homogeneously labeled regions.

2.4.3 Performance Cross-Modality Influence Analysis

We compute the cross-modality influence for all five JRN networks by looking at the relation between the cross-modality influence numbers and the performance in the respective modalities (see Fig. 2.7). We observe that there is no linear relationship between cross-modality influence and performance but they rather lie within an area which is, according to our observations, upper-bounded by a concave curve. This means that a larger influence between modalities does not guarantee better performance in the respective metric. Indeed a large negative effect can hamper performance (see Fig. 2.6).

Based on our findings about cross-modality influence, we hypothesize that the relationship between cross-modality influence and performance can be generalized into a plot which is sketched in Fig. 2.8. The cross-modality influence arises from certain model design decisions, as well as from modality combinations for a particular end-task. For example, we have seen in our experiments that moderately transferring shapes and class-wise depth priors from the semantic map into the depth map can help improving depth estimation (see Fig. 2.5 bottom). However, the cross-modality influence $\omega_{S \to D'}$ can also be too strong (large positive influence number) which causes a decrease in performance. For example, shapes from semantic segmentation can cause halos and artifacts in the depth map (see Fig. 2.6). We conclude that inspecting performance

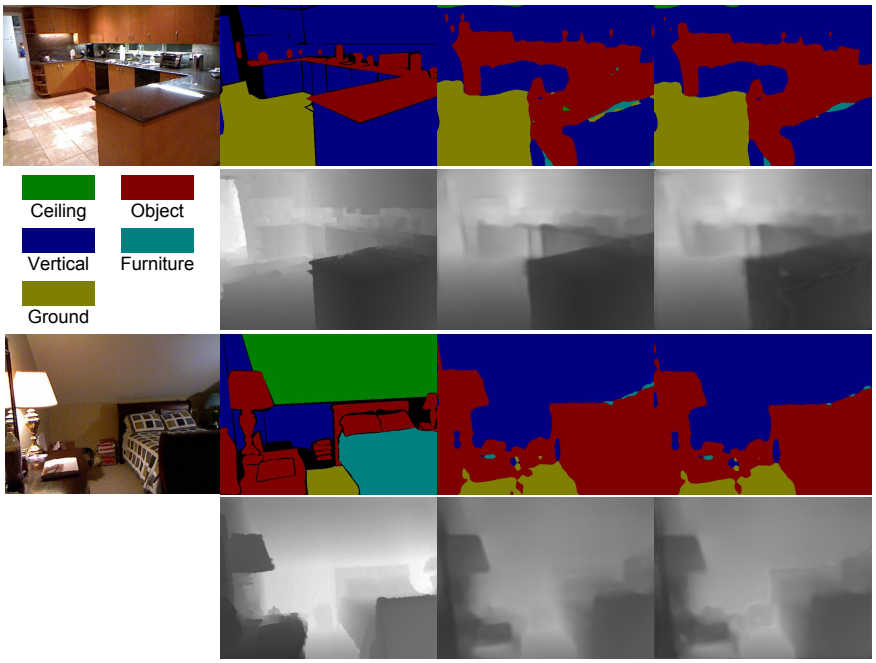


Figure 2.5: Two qualitative results of our `Sum60` network compared with the input (i.e state-of-the-art). Each top row (from left to right): Image, ground truth semantic labeling, result of [28] and our result. Each bottom row (from left to right): Ground truth depth map, result of [32] and our result. The first example depicts an improvement in semantic labeling, where ground label (yellow) has been removed (next to object label (red)). In the second example, the depth edges of the upper bed frame are better recovered in our result (best viewed zoomed-in). Please note that our results are smooth and follow edges in the input image, despite having no explicit CRF model.

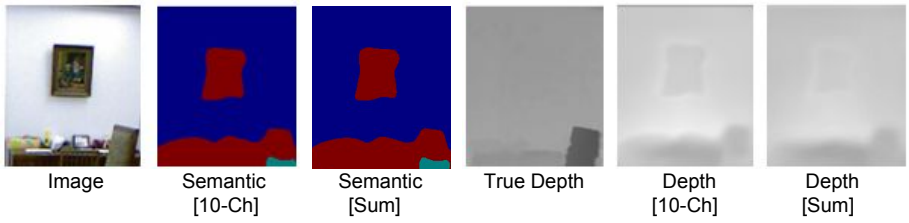
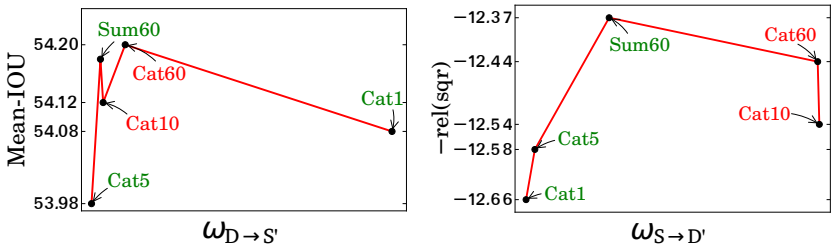


Figure 2.6: **Negative influence.** From left to right: Input image, our semantic labeling of the `Cat10` network, semantic labeling of the `Sum60` network, ground truth depth, result of our `Cat10` network, result of our `Sum60` network. Despite the `Cat10` network having a higher cross-modality influence number $\omega_{S\to D'}$ than the `Sum60` network (cf. Fig. 2.7 top right), the respective depth accuracy (-rel(sqr)) of the `Cat10` network is lower. This is visible in the image where the picture frame has received a wrong depth in the `Cat10` network result, compared to the `Sum60` network result. (Image crops shown for visualization purpose.)



	$\omega_{D\to S'}$	$\omega_{S\to D'}$
Cat60	0.94	4.9
Cat10	0.39	4.93
Cat5	0.10	-0.24
Sum60	0.32	1.11
Cat1	7.59	-0.40

Figure 2.7: **Performance vs. cross-modality influence plots for all models.** We use the mean IOU measure for semantic labels (A_S), and $-$rel(sqr)$*$100 for depth (A_D). The right table shows all influence numbers $\omega_{D\to S'}$ and $\omega_{S\to D'}$ for all models. The top figures are the respective performance-influence plot. Both plots exhibit a peak where the optimal trade-off between cross-modality influence and evaluated performance is achieved. We see that the `Sum60` and `Cat60` models are at the peaks of the respective plots. We colored models in red which feature *loose computation* and in green which feature *coupled computation* (see Sec. 2.3.2). This supports the idea that the cross-modality influence number can facilitate the systematic exploration of network architectures.

vs. cross-modality influence plots is a useful way to find appropriate modular architectures. Furthermore, these plots may help identifying complementary modalities to further leverage the cross-modality influence for task performance improvements.

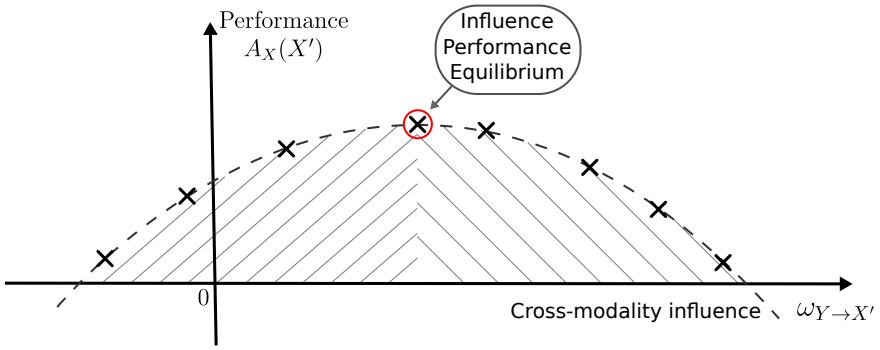


Figure 2.8: **The performance vs. cross-modality influence curve.** For each modality pair (X, Y) and in each modality influence direction $Y \to X'$ and $X \to Y'$ the relationship between the magnitude of the cross-modality influence and the prediction performance needs to be balanced into an equilibrium. This cross-modality influence analysis will be helpful when designing models which should operate on multiple modalities and carry out joint predictions.

2.5 Discussion

Inspired by work in neuroscience we have introduced a systematic way to measure the cross-modality influence present in our JRN networks. By doing so, we were able to identify a network which achieves a measurable influence between modalities, has an overall good performance compared to other JRN networks, and is consistently better than the state-of-the-art input modalities.

Chapter 3

Instance-aware Scene Flow Estimation

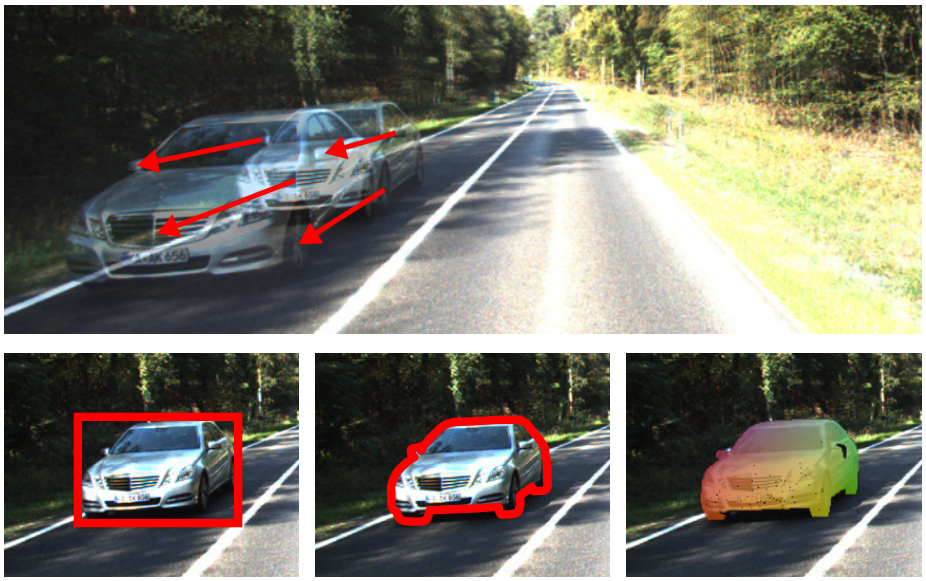
3.1 Introduction

3D motion estimation is a core problem in computer vision with numerous applications. Consider, for instance, an autonomous car navigating through a city. Besides recognizing traffic participants and identifying their 3D locations, it needs to precisely predict their 3D position in the future.

In this paper, we focus on 3D scene flow estimation for autonomous driving scenarios. More specifically, given two consecutive stereo images we want to predict the 3D motion of every pixel. With the advent of challenging real-world benchmarks, such as the KITTI 2012 [51] and KITTI 2015 [52], great progress has been made in this area. In particular, segmentation-based formulations which gain their strength by reasoning over piece-wise planar patches [53] or segmenting the scene into its rigidly moving components [52] dominate the leaderboards.

However, existing methods rely heavily on local features for computing the data term and for initialization. As a consequence, even state-of-the-art methods fail in the presence of very large displacements or local ambiguities from, e.g., textureless or reflective surfaces. Consider Fig. 3.1 (top) which shows an image from the KITTI 2015 scene flow dataset [52]. Due to the large displacement between frames, the front wheel in the first frame appears similar to the back wheel in the second frame, resulting in wrong predictions. Furthermore, the small amount of texture and the reflective car surface complicate the matching task.

In this paper, we propose to exploit recognition to facilitate this problem. In particular, we investigate the benefits of semantic grouping and fine-grained geometric recognition for this task as illustrated in Fig. 3.1. We will formulate the output of the recognition task as a new constraint, besides standard constraints such as local appearance and 3D motion-smoothness within an image,



2D Bounding Box 2D Instance Segmentation 3D Object Coordinates

Figure 3.1: **Motivation.** Top: Two consecutive frames (overlaid) from the KITTI 2015 scene flow dataset. Large displacements and specular surfaces are challenging for current scene flow estimation algorithms. Bottom: Recognition can provide powerful geometric cues to help with this problem. In this work, we investigate: 2D bounding boxes, 2D instance segmentations and 3D object coordinates.

in a CRF-based framework.

For semantic grouping, we consider two scenarios: i) a bounding box around the visible part of each semantic instance, and ii) a pixel-wise segmentation of the visible part of each semantic instance. While the latter output provides a more detailed mask of the object, and hence may be more beneficial for the scene flow task, it is also a harder task to solve, and hence may contain segmentation errors. While these cues do not provide additional geometric evidence, they constrain the space of possible rigid body motions: pixels which are grouped together are likely to move as a single rigid object in the case of vehicles. We integrate both cues into the scene flow task by enforcing consistency between neighboring views (either in time or space). In short, a pixel within an instance region (bounding box or segment) in one frame should be mapped to an instance in the other frame. Furthermore, all pixels within an instance should move as one rigid entity.

To obtain the bounding box or segmentation masks, we utilize an existing state-of-the-art approach - the multi-task network cascade (MNC) from Dai et al. [54]. In contrast to [54], we achieve improved outputs, by providing a dense depth map as additional input to the network. To train the CNN, we annotated

400 stereo images from the KITTI 2015 benchmark of Menze et al. [52] using in-house annotators.

For fine-grained geometric recognition we train a new convolutional neural network (CNN) on 2D instance segmentations of MNC which predicts the continuous 3D object parts, also known as 3D object coordinates [55, 56, 57, 58, 59], for each pixel inside the instance mask. The 3D object coordinates specify the relative 3D location of an object's surface point with respect to the object's local coordinate frame as shown in Fig. 3.1 (bottom-right) with illustrative colors. Thus, they provide a detailed geometric cue for matching pixels in neighboring frames, even in the presence of large displacements.

The fine-grained geometric recognition and the semantic grouping have, certainly, a different trade-off between modeling-power versus prediction accuracy. If the recognition task was to be solved perfectly well, the continuous object part would be the strongest geometric cue, followed by the segmentation mask and the bounding box. On the other hand, as we will see experimentally, the continuous object parts are most challenging to predict precisely, followed by the segmentation mask and the bounding box. Given this trade-off, the key question addressed in this work is:

Which level of recognition is most beneficial for the scene flow task, when combining different cues, i.e. local appearance, 3D motion-smoothness and recognition-based geometric constraint, in a CRF-based framework?

Our CRF framework is based on [52] and termed Instance Scene Flow. We validate the benefits of recognition cues for scene flow on the challenging KITTI 2015 benchmark [52]. Firstly, we conduct a detailed ablation study for the three levels of recognition granularity, i.e. 2D bounding box, 2D segmentation mask, and continuous 3D object parts. From this study, we conclude that the instance segmentation cue is by far strongest, in our setting. Secondly, we show that our Instance Scene Flow method significantly boosts the scene flow accuracy, in particular in challenging foreground regions. Alongside, we obtain the lowest overall test errors amongst all methods at time of submission. Our code and datasets will be made available upon publication.

In short, our **contributions** are:

- A new 3D scene flow method, leveraging recognition-based geometric cues to achieve state-of-the-art performance on the challenging KITTI 2015 scene flow benchmark, at the time of submission.
- A detailed ablation study of the importance of recognition granularity, from coarse 2D bounding boxes over 2D instance segmentations to fine-grained 3D object part predictions, within our scene flow framework.
- High-quality, instance-level annotations of 400 stereo images from the KITTI 2015 benchmark [52].

3.2 Related Work

In the following, we review the most related works on *image-based* scene flow estimation, semantic priors and object coordinates. For an overview of RGB-D

methods (e.g., using the Kinect), we refer to [60, 61, 62, 63, 64, 61].

Scene Flow: Following the work of Vedula et al. [65, 66] several approaches formulate 3D scene flow estimation as a variational optimization problem [67, 68, 69, 70, 71, 72]. Unfortunately, coarse-to-fine variational optimization suffers when displacements are large. Thus, slanted-plane models [73, 74] have recently been introduced [75, 76, 52, 77, 78] which gain their robustness by decomposing the scene into a collection of rigidly moving planes and exploiting discrete-continuous optimization techniques.

While these methods have demonstrated impressive performance on the challenging KITTI benchmark [51, 52], they fail to establish correspondences in textureless, reflective or fast-moving regions due to violations and ambiguities of the data term and weak prior assumptions. In this paper, we propose to approach this problem using fine-grained instance recognition and 3D geometry information, resulting in significant accuracy gains.

Semantic Priors: Several works have considered semantic information for stereo or optical flow estimation. Gney et al. [79] presented a model for stereo estimation where 3D object hypotheses from a CAD model database are jointly estimated with the disparity map. Hur et al. [80] proposed a model for joint optical flow and semantic segmentation. In particular, they classify the scene into static and dynamic regions and introduce a constraint which measures how well the homography of a superpixel meets the epipolar constraint. Sevilla-Lara et al. [81] used semantic segmentation to identify object instances and combine per-object layered flow predictions [82] with DiscreteFlow [83]. Bai et al. [84] proposed a model which first identifies car instances and then predicts each rigidly moving component in the scene with a separate epipolar flow constraint.

While existing methods leverage recognition to aid either reconstruction or motion estimation, in this paper we consider recognition for the 3D scene flow problem, i.e., the combination of the two. In contrast to flow-only methods [84] that need to search for correspondences along epipolar lines, our method exploits the semantic cues as well as the geometry which allows us to estimate the rigid motion between frames more robustly and leads to significantly improved results compared to all baselines. Furthermore, we are (to the best of our knowledge) the first to investigate the impact of recognition granularity on the task, ranging from coarse 2D boxes to fine grained 3D object coordinates predictions.

3D Object Coordinates: Continuous 3D object parts, also known as 3D object coordinates, have so far mainly been leveraged in the context of 3D pose estimation [59, 55, 56, 58], camera re-localization [57] and model-based tracking [85]. The typical approach is to train a random forest for predicting instance-specific object probabilities and 3D object coordinates with respect to a fixed local coordinate system. These predictions are used to fit a 3D model of the known object, resulting in the 3D object pose.

In this work, we explore the possibility of using object coordinates as a continuous labeling for the surface points of the object.

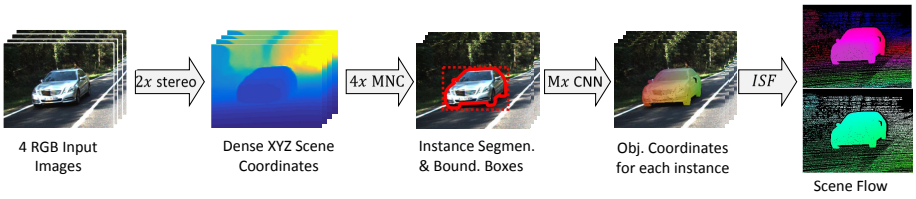


Figure 3.2: **Work flow for our approach**. Note that each intermediate step uses as input all of the previous results. Given the four RGB input images (t/t+1, left/right) we compute 3D points (XYZ) for each pixel. For each of the four RGB,XYZ image-blocks we obtain instance segmentations, alongside bounding boxes. The M instances are processed individually to obtain object coordinates for each instance, using our object coordinates CNN. Finally, all this information is integrated into our Instance Scene Flow method (ISF) to produce the final output.

3.3 Method

This section describes our approach to 3D scene flow estimation. The overall work flow of our approach is illustrated in Fig. 3.2. Given the 4 RGB images we extract 3D points (XYZ) for each pixel in camera coordinate system using a stereo method (see Section 5.4.3 for details). Based on the RGB and XYZ values, we train a multi network cascade (MNC) [54] to predict 2D bounding boxes and 2D instance segmentations. We train a CNN to predict object coordinates for car instances. Finally, we integrate the bounding box, instance and object coordinates cues into a slanted-plane formulation and analyze the importance of each cue for the scene flow estimation task. The remainder of this section is structured as follows.

As our goal is to analyze the impact of different levels of recognition granularity, we first describe the inputs to our method in Section 3.3.1 and Section 3.3.2. In particular, we are interested in improving scene flow estimation of vehicles which are challenging due to their large motion and non-lambertian appearance. In Section 3.3.3, we finally show how these predictions can be integrated into a CRF model for 3D scene flow estimation.

3.3.1 2D Bounding Boxes and Instances

As discussed before, recognizing and segmenting objects is imperative for our approach. For this task, we use the state-of-the-art Multi-task Network Cascades (MNC) proposed by Dai et al. [54] to obtain bounding boxes and segmentation masks of all cars present in the scene. Unlike the standard MNC framework which operates on RGB images, we provide the network with an RGB-XYZ image, where XYZ denotes the 3D scene coordinates, i.e., the 3D location of every pixel in the scene in camera coordinates. The 3D location for each pixel

is computed from disparity maps, see Section 5.4.3 for details.

We pre-train the network using Pascal VOC and fine-tune it using 3200 coarse annotations of KITTI [51] provided by [86]. We obtained 200 images with 1902 pixel-accurate instance annotations from the KITTI 2015 Stereo benchmark [52] using in-house annotators. We further refined the model with these fine annotations. The final accuracy (IoU-50%) on the validation set of [86] is 83% as compared to 78% when using only RGB images.

3.3.2 3D Object Coordinates

3D object coordinates specify the 3D location of the surface of an object with respect to a local coordinate frame, see Fig. 3.1 for an illustration. They can be viewed as a fine grained unique geometric labeling of the object's surface which is independent of the viewpoint and can be used to establish correspondences between frames, which we expect to be more robust to appearance changes than correspondences based on sparse feature matching.

While random forests have been used for estimating object coordinates when the target instance is known [55, 56, 57, 85], we found that a CNN-based approach leads to significantly better object coordinates predictions as also evidenced by our experiments in Section 5.4.3. We use a modified version of the encoder-decoder style CNN proposed in [87] for estimating the object coordinates at each pixel. As above, the input to the CNN is an RGB-XYZ image as well as the instance prediction from the MNC. The output of the CNN is a 3 layer regression which stores the X, Y and Z coordinates for each point of the input. We refer to the architecture of our CNN in supplementary material. The encoder part comprises a set of 5 convolutional layers with a stride of 2 in each layer, followed by a set of 3 fully connected layers. The decoder part has 5 deconvolutional layers followed by a 3D regression layer which predicts the 3D object coordinates.

3.3.3 Scene Flow Model

We now describe our scene flow model which is based on [52] but adds two new components to it, in particular an instance sensitive appearance term and a term which encourages object coordinates to align across frames. For making the paper self-contained, we specify the full model.

Notation: We first describe the notation of the inputs to the scene flow model which are pre-computed as described in the previous section and fixed during inference. Let $\mathcal{V} = \{0, 1, 2, 3\}$ denote the set of views as illustrated in Fig. 3.3 and let $\mathbf{I}_v \in \mathbb{R}^{w \times h \times 3}$ denote the input image corresponding to each view. For each view $v \in \mathcal{V}$, we compute the following information: first, our MNC predicts instance label maps $\mathbf{M}_v \in \{0, \ldots, |M_v|\}^{w \times h}$ which determine the predicted semantic instance label for each pixel in each view. In our experiments, these maps are either coarse 2D bounding box segmentations or more accurate 2D instance segmentations which are both computed via MNC. Background

pixels are assigned the label $\mathbf{M}_v(\mathbf{p}) = 0$ and foreground pixels are assigned positive labels. Note that instance labels do not correspond across frames as the correspondence is unknown a-priori and needs to be inferred jointly with the scene flow. Furthermore, we denote the 3D object coordinates predicted by the network with $\mathbf{C}_v \in [-1, 1]^{w \times h \times 3}$.

We now describe the parameters of our model which are optimized during inference. Let $\mathcal{S}$ denote the set of superpixels in the reference view and $\mathcal{O}$ denote the set of objects in the scene. Each superpixel $i \in \mathcal{S}$ is associated with a region $\mathcal{R}_i$ in the reference image and a variable $\mathbf{s}_i = (\mathbf{n}_i, k_i)^\mathsf{T}$, where $\mathbf{n}_i \in \mathbb{R}^3$ describes a plane in 3D via $\mathbf{n}_i^\mathsf{T}\mathbf{x} = 1$. Further, let $k_i \in \{0, \dots, |\mathcal{O}|\}$ index the object which the superpixel is associated with. Here $|\mathcal{O}|$ denotes an upper bound on the number of objects we expect to see, and $k = 0$ refers to the background object with $k > 0$ to other traffic participants. Each object $j \in \mathcal{O}$ is associated with a variable $\mathbf{o}_j \in SE(3)$ which describes its rigid body motion in 3D. Each superpixel associated with object j inherits its rigid motion parameters $\mathbf{o}_j \in SE(3)$. In combination with the plane parameters $\mathbf{n}_i$, this fully determines the 3D scene flow at each pixel inside the superpixel.

Energy Model: Given the left and right input images of two consecutive frames (Fig. 3.3), our goal is to infer the 3D geometry of each superpixel $\mathbf{n}_i$ in the reference view, the association to objects k_i and the rigid body motion of each object $\mathbf{o}_j$. We formulate the scene flow estimation task as an energy minimization problem comprising data, smoothness and instance terms:

$$\hat{\mathbf{s}}, \hat{\mathbf{o}} = \underset{\mathbf{s},\mathbf{o}}{\operatorname{argmin}} \underbrace{\varphi(\mathbf{s}, \mathbf{o})}_{\text{data}} + \underbrace{\psi(\mathbf{s})}_{\text{smooth.}} + \underbrace{\chi(\mathbf{s}, \mathbf{o})}_{\text{instance}} \tag{3.1}$$

We summarize all variables involved in the optimization with $\mathbf{s} = \{\mathbf{s}_i | i \in \mathcal{S}\}$ and $\mathbf{o} = \{\mathbf{o}_i | i \in \mathcal{O}\}$. For clarity of exposition we omit all weight parameters of the model.

Data Term: Our data term encodes the assumption that corresponding points across all images should be similar in appearance:

$$\varphi(\mathbf{s}, \mathbf{o}) = \sum_{i \in \mathcal{S}} \sum_{\mathbf{p} \in \mathcal{R}_i} \sum_{v \in \mathcal{V}} \varphi_v^{\mathrm{D}}(\mathbf{p}, \mathbf{q}) \tag{3.2}$$

$$\mathbf{q} = \underbrace{\mathbf{K}\left(\mathbf{R}_v(\mathbf{o}_{k_i}) - \mathbf{t}_v(\mathbf{o}_{k_i})\,\mathbf{n}_i^\mathsf{T}\right)\mathbf{K}^{-1}}_{\text{homography (view 0 } \rightarrow \text{ view } v)}\,\mathbf{p} \tag{3.3}$$

Here, $\mathcal{V} = \{1, 2, 3\}$ denotes the set of target views and $\mathbf{q}$ is the location of pixel $\mathbf{p}$ in reference view 0 mapped into the target view $v \in \mathcal{V}$ according to the calibration matrix $\mathbf{K}$, the rigid body motion $\mathbf{R}_v|\mathbf{t}_v$ of the corresponding object $\mathbf{o}_{k_i}$ and the plane parameters of the associated superpixel $\mathbf{n}_i$. See Fig. 3.3 for an illustration.

The data cost $\varphi_v^{\mathrm{D}}(\mathbf{p}, \mathbf{q})$ compares the appearance at pixel $\mathbf{p}$ in reference image 0 with the appearance at pixel $\mathbf{q}$ in the target view $v \in \mathcal{V}$. In our experiments, we use Census descriptors [88] which are robust to simple photometric

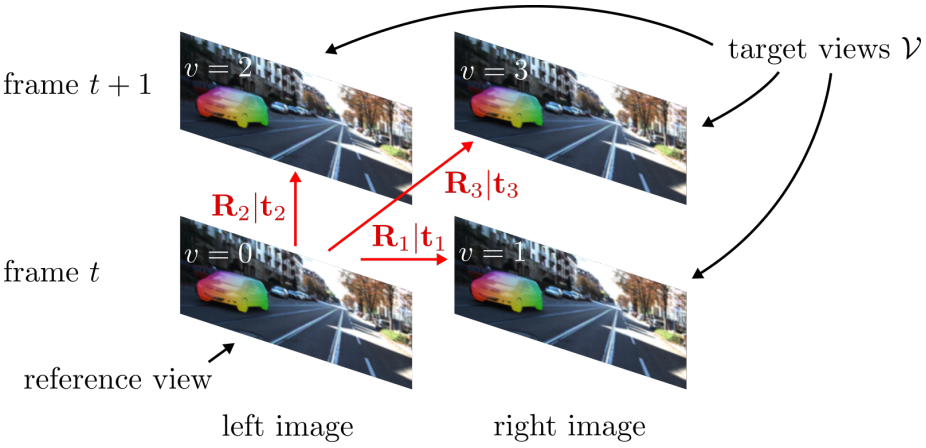


Figure 3.3: **Geometric Relationship** between reference and target views. Pixels in the reference view are mapped to a pixels in a target view via their depth and rigid body motion.

variations [52, 53, 74]. To guide the optimization process and overcome local minima we additionally add a robust ℓ_1 loss to φ_v^{D}. This loss measures the difference with respect to sparse DiscreteFlow correspondences [83] for the flow terms ($v = 2, 3$) and depth estimates from SPS-stereo [74] for the stereo term ($v = 1$).

Smoothness Term: The smoothness term encourages coherence of adjacent superpixels in terms of depth, orientation and motion. It decomposes as

$$\psi(\mathbf{s}) = \gamma_{ij}^{\mathrm{S}} \sum_{i \sim j} \psi_{ij}^{\mathrm{G}}(\mathbf{n}_i, \mathbf{n}_j) + \psi_{ij}^{\mathrm{M}}(\mathbf{s}_i, \mathbf{s}_j) \tag{3.4}$$

with the following geometry (G) and motion (M) terms:

$$\begin{aligned} \psi_{ij}^{\mathrm{G}}(\mathbf{n}_i, \mathbf{n}_j) &= \sum_{\mathbf{p} \in \mathcal{B}_{ij}} \rho\left(d(\mathbf{n}_i, \mathbf{p}) - d(\mathbf{n}_j, \mathbf{p})\right) \\ &+ \rho\left(1 - |\mathbf{n}_i^{\mathsf{T}} \mathbf{n}_j| / (\|\mathbf{n}_i\| \|\mathbf{n}_j\|)\right), \\ \psi_{ij}^{\mathrm{M}}(\mathbf{s}_i, \mathbf{s}_j) &= \gamma_{ij}^{\mathrm{M}}(\mathbf{n}_i, \mathbf{n}_j)\, [k_i \neq k_j]. \end{aligned}$$

Here, $d(\mathbf{n}, \mathbf{p})$ denotes the disparity of plane $\mathbf{n}$ at pixel $\mathbf{p}$ in the reference image, $\mathcal{B}_{ij}$ is the set of shared boundary pixels between superpixel i and superpixel j, and ρ is the robust ℓ_1 penalty. The instance-sensitive weight γ_{ij}^{S} is defined as

$$\gamma_{ij}^{\mathrm{S}} = 1 - \beta_{\mathrm{S}} \cdot [(i, j) \in \mathcal{M}] \tag{3.5}$$

where $\mathcal{M}$ denotes the set of adjacent superpixel pairs where exactly one of the superpixels lies on an object instance. $\beta \in [0, 1]$ is a hyper-parameter which

weighs down the costs of discontinuities for adjacent superpixels in $\mathcal{M}$. Note that this weighting is only possible in the presence of instance predictions.

The geometry-sensitive motion weight is defined as

$$\begin{aligned}\gamma_{ij}^{\mathrm{M}}(\mathbf{n}_i, \mathbf{n}_j) &= \exp\left(-\frac{\lambda}{|\mathcal{B}_{ij}|}\sum_{\mathbf{p}\in\mathcal{B}_{ij}}(d(\mathbf{n}_i,\mathbf{p}) - d(\mathbf{n}_j,\mathbf{p}))^2\right) \\ &\quad \times |\mathbf{n}_i^{\mathsf{T}}\mathbf{n}_j|/(\|\mathbf{n}_i\|\|\mathbf{n}_j\|)\end{aligned}$$

encouraging motion boundaries to align with 3D folds and discontinuities rather than within smooth surfaces.

Instance Term: The instance term $\chi(\mathbf{s}, \mathbf{o})$ measures the compatibility of appearance and part-labeling induced by the 3D object coordinates when warping the detected instances into the next frame. It takes the following form

$$\chi(\mathbf{s}, \mathbf{o}) = \sum_{i\in\mathcal{S}}\sum_{\mathbf{p}\in\mathcal{R}_i}\sum_{v\in\mathcal{V}}\chi_v^{\mathrm{I}}(\mathbf{p}, \mathbf{q}) \tag{3.6}$$

with

$$\begin{aligned}\chi_v^{\mathrm{I}}(\mathbf{p}, \mathbf{q}) = \quad & [\mathbf{M}_0(\mathbf{p}) = 0 \vee \mathbf{M}_v(\mathbf{q}) = 0] \cdot \lambda \; + \\ & [\mathbf{M}_0(\mathbf{p}) > 0 \wedge \mathbf{M}_v(\mathbf{q}) > 0] \cdot \left(\chi_v^{\mathrm{A}}(\mathbf{p}, \mathbf{q}) + \chi_v^{\mathrm{L}}(\mathbf{p}, \mathbf{q})\right)\end{aligned} \tag{3.7}$$

Here, $\mathbf{q}$ is calculated as in Eq. 3.3 and the appearance (A) potential and the part labeling (L) potential are defined as

$$\chi_v^{\mathrm{A}}(\mathbf{p}, \mathbf{q}) = \|\mathbf{I}_0(\mathbf{p}) - \mathbf{I}_v(\mathbf{q})\|_1 \tag{3.8}$$

$$\chi_v^{\mathrm{L}}(\mathbf{p}, \mathbf{q}) = \|\mathbf{C}_0(\mathbf{p}) - \mathbf{C}_v(\mathbf{q})\|_1 \tag{3.9}$$

and measures the difference in appearance $\mathbf{I}$ and 3D object coordinates $\mathbf{C}$ between image location $\mathbf{p}$ in the reference view and $\mathbf{q}$ in the target view, respectively. While the data term in Eq. 3.2 also evaluates appearance, we found that the Census descriptors work well mostly for textured background regions. In contrast, it returns noisy and unreliable results in the presence of textureless, specular surfaces such as on cars. However, as evidenced by our experiments, including an additional ℓ_1 constraint on appearance for instances leads to significantly better estimates in those cases. This observation is in accordance with recent works on direct visual odometry and SLAM [89, 90, 91] which use similar measures to reliably estimate the camera pose in weakly textured environments. In contrast to them, here we exploit this constraint to estimate the relative pose of each individual weakly textured object in the scene.

The intuition behind this term is as follows: when warping the recognized instances from the reference frame into the target frame according to the estimated geometry and motion, the appearance as well as the part labeling induced by the object coordinates should agree. The term $[\mathbf{M}_0(\mathbf{p}) > 0 \wedge \mathbf{M}_v(\mathbf{q}) > 0]$ ensures that these constraints are only evaluated if both the reference and the target pixel belong to a detected instance (i.e., when $\mathbf{M} > 0$). However, as

Algorithm 1 Optimization

Input: $\mathbf{I}_v$, $\mathbf{M}_v$, $\mathbf{C}_v$ for $v \in \mathcal{V}$
Initialize $\mathbf{s}$ and $\mathbf{o}$ as described in "Initialization"
for all iterations $n = 1, \ldots, N$ **do**
    for all $i \in \mathcal{S}$ **do**
        Draw samples for $\mathbf{s}_i$ (Gaussian)
    for all $j \in \mathcal{O}$ **do**
        Draw samples for $\mathbf{o}_j$ (MCMC)
    Run TRW-S [92] on discretized problem
Output: $\hat{\mathbf{s}}$, $\hat{\mathbf{o}}$

both χ_v^{A} and χ_v^{L} are positive terms the model prefers to associate instances with background regions which incurs no additional cost compared to associating instances with instances. We therefore incorporate an additional term which yields an appropriate bias λ and favors the association of instances.

Optimization: For optimizing Eq. 3.1, we leverage max-product particle belief propagation with TRW-S [92] as proposed in [52]. At each iteration this optimization algorithm discretizes the continous variables by drawing samples around the current maximum-a-posteriori (MAP) solution and runs TRW-S until convergence before resampling. However, we found that this approach gets easily trapped in local minima, in particular due to the highly non-convex energy function associated with the appearance of the foreground objects.

We thus modify their sampling strategy as shown in Algorithm 1, which leads to better results. While the original algorithm [52] discretizes the continuous variables (in particular the rigid body motions $\mathbf{o}$) by sampling from a Gaussian centered at the current MAP estimate, we create samples by running a Markov chain based on the appearance term in Eq. 3.6 for each object individually. More specifically, our sampling energy warps all pixels inside an instance based on the predicted depth and the rigid body motion of the sample, and measures the photoconsistency between reference and target views using the ℓ_1 norm. This "informed" way of sampling ensures that TRW-S has access to high-quality samples compared to noisy estimates drawn around the current MAP solution. For sampling the geometry variables (i.e., normals), we follow [52].

The hyper parameters in our model are estimated using block coordinate descent on a train/validation split.

Initialization: As we are presented with a complex NP hard optimization problem in Eq. 3.1, initialization of parameters is an important step. We describe the details of our initialization procedure in the following.

We initialize the geometry parameters using dense disparity maps and the object hypotheses/motion variables using sparse flow predictions and the predicted bounding boxes/instances. More specifically, we robustly fit all superpixel parameters to the disparity estimates and aggregate all sparse flow estimates for each instance to robustly fit a rigid body motion to it via RANSAC. We

refer to Section 5.4.3 for details on the particular choice of input algorithms.

For the instance-based algorithms, we aggregate the sparse flow estimates directly based on the instance masks and robustly fit a rigid body motion to it via RANSAC. Given the high quality of the instance predictions, this leads to very robust initializations. For baselines which do not leverage recognition we follow [52] and initialize objects based on clustering sparse 3D scene flow estimates which disagree with the background motion. Based on this clustering, we initialize the associations and poses using robust fitting using RANSAC. We refer the reader to [52] for further details.

While we have also experimented with color histograms and pose prediction to support the assignment of instances across frames, we found that aggregating sparse flow vectors [83] and dense geometry [93] allows for correctly associating almost all objects. We thus don't use such an additional term in the model, which, however, could be easily integrated to solve more challenging association problems as present in the KITTI 2015 scene flow benchmark.

Runtime: Our MATLAB implementation with C++ wrappers requires on average 40 seconds for each of the 10 iterations of the optimization described in Algorithm 1. This leads to a runtime of 7 minutes for processing one scene (4 images) on a single i7 core running at 3.0 Ghz. In addition, the inputs to our methods namely, dense disparity maps, sparse flow predictions and predicted bounding boxes/instances require on average 3 minutes for processing one scene, leading to a total runtime of 10 minutes.

3.4 Experimental Evaluation

	D1			D2			Fl			SF		
	bg	*fg*	*bg+fg*	*bg*	*fg*	*bg+fg*	*bg*	*fg*	*bg+fg*	*bg*	*fg*	*bg+fg*
OSF	4.00	8.86	4.74	5.16	17.11	6.99	6.38	20.56	8.55	7.38	24.12	9.94
ISF-BBox	3.94	8.81	4.69	5.10	10.77	5.97	6.46	12.90	7.44	7.42	17.11	8.90
ISF-SegMask	4.06	7.97	4.66	5.26	9.20	5.86	6.72	10.78	7.34	7.74	14.60	8.79
ISF-SegMask-ObjCoord	4.08	7.98	4.68	5.27	9.20	5.87	6.72	10.84	7.35	7.75	14.66	8.80
ISF-SegMask-CNNDisp	3.55	3.94	3.61	4.86	4.72	4.84	6.36	7.31	6.50	7.23	8.72	7.46

Table 3.1: **Quantitative results from ablation study on KITTI 2015 Validation Set.** We report the disparity (D1,D2), flow (Fl) and scene flow (SF) error averaged over our validation set for *OSF*[52] (no recognition input), *ISF-BBox* (bounding box input), *ISF-SegMask* (segmentation input) and *ISF-SegMask-ObjCoord* (segmentation + object coordinates input). Additionally, we also report results of *ISF-SegMask-CNNDisp* (ISF-SegMask with higher quality CNN based disparity input).

3.4.1 Effect of recognition granularity

In this section, we study the impact of different levels of recognition granularity for estimating the 3D scene flow of dynamic (i.e., foreground) objects. In addi-

tion to the recognition cues, we use sparse optical flow from sparse DiscreteFlow correspondences [83] and dense disparity maps from SPS-stereo [74] for both rectified frames. We obtain the superpixel boundaries using StereoSLIC[73]. Table 3.1 provides a quantitative comparison of the performance of *OSF* [52] (no recognition input), *ISF-BBox* (2D bounding boxes as recognition input), *ISF-SegMask* (2D instance segmentations as recognition input) and *ISF-SegMask-ObjCoord* (2D instance segmentations in conjunction with 3D object coordinates as recognition input) on our validation set (which is a subset of the KITTI 2015[52] scene flow training set). We report the error with respect to four different outputs: disparities in the first and second frame (D1,D2), optical flow (FL) and scene flow (SF).

Our results indicate that recognition (both 2D bounding box and 2D instance segmentation) provides large improvements for optical flow estimation (and in turn scene flow estimation) on foreground parts of the scene. Note that we do not tackle the recognition of static background objects in this paper which are estimated relatively well without such priors. Furthermore, we note that instance segmentations as input improve performance in particular at the boundary of objects. We attribute this effect to the more fine grained nature of the 2D segmentation input compared to using rough 2D bounding boxes input. We remark that for evaluation, the KITTI 2015 benchmark considers only a subset of the cars in the scene as foreground, specifically dynamic cars within a threshold distance to the camera. In contrast, as defined in section 3.3.3, our scene flow estimation model considers all car instances detected by our CNN as foreground. Figure 3.4 provides a qualitative comparison illustrating the differences between the scene flow errors of methods employing different levels of recognition granularity. We encourage the reader to have a look at the first section of our supplementary material for additional examples.

Moreover, we observe that 3D object coordinates do not increase performance beyond 2D instance segmentations (and hence yield the same estimates, i.e., the weight of the object coordinate terms are zero after optimization). We attribute this to the quality of the state-of-the-art object coordinate predictions. Specifically, the accuracy of 3D object coordinate predictions from state-of-the-art CNN-based methods is below what is required to further improve 3D scene flow estimation due to the high level of accuracy requested by current benchmarks (i.e., 3 pixels error in KITTI). We refer the reader to the second section of our supplementary material for a detailed analysis of why 3D object coordinate predictions do not provide any further gains.

Experiments with CNN based disparity as input: Furthermore, we evaluated our method with the optimal level of recognition granularity selected based on the ablation study (*ISF-SegMask*) on higher quality disparity inputs computed from DispNetC [93] and MC-CNN-acrt [94]. In particular, combining DispNetC results for instance pixels and MC-CNN-acrt results for the rest performed best on our validation set. We report the results of our top performing method *ISF-SegMask-CNNDisp* on the validation set in Table 3.1.

	D1			D2			Fl			SF		
	bg	*fg*	*bg+fg*	*bg*	*fg*	*bg+fg*	*bg*	*fg*	*bg+fg*	*bg*	*fg*	*bg+fg*
PRSM* [75]	3.02	10.52	**4.27**	5.13	15.11	6.79	**5.33**	13.40	6.68	6.61	20.79	8.97
OSF+TC* [95]	4.11	9.64	5.03	5.18	15.12	6.84	5.76	13.31	7.02	7.08	20.03	9.23
OSF [52]	4.54	12.03	5.79	5.45	19.41	7.77	5.62	18.92	7.83	7.01	26.34	10.23
ISF-SegMask-CNNDisp	**4.12**	**6.17**	4.46	**4.88**	**11.34**	**5.95**	5.40	**10.29**	**6.22**	**6.58**	**15.63**	**8.08**

Table 3.2: **Quantitative Results on the KITTI 2015 Scene Flow Evaluation Server.** This table shows the disparity (D1/D2), flow (Fl) and scene flow (SF) errors averaged over all 200 test images for our method in comparison to other leading methods (OSF[52], OSF-TC[95] and PRSM[75]) on the KITTI 15 benchmark. Methods with * use more than two temporal frames.

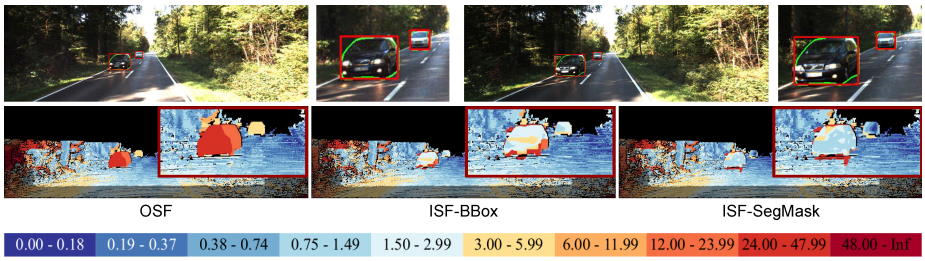


Figure 3.4: **Qualitative Results from Ablation Study on KITTI 2015 Validation Set.** The top row shows predicted masks and bounding boxes which form the input to our method overlaid onto the left camera image at time t and $t+1$. Each figure in the bottom row (from left to right) shows scene flow error of *OSF*[52] (no recognition input), *ISF-BBox* (bounding box input) and *ISF-SegMask* (segmentation input) using the color scheme depicted in the legend. The red box shows zoom-ins.

3.4.2 Results on the KITTI Benchmark

In this section, we present results of our top performing method *ISF-SegMask-CNNDisp* evaluated on the KITTI 2015 scene flow evaluation server. Table 3.2 compares the performance of our method with other leading methods on the benchmark: PRSM [53], OSF [52] and OSF-TC [95]. We obtain state-of-the-art performance at the time of submission, even including anonymous submissions. Notably, our method also outperforms methods which use more than two temporal frames (OSF-TC [95] and PRSM[53]). Figure 3.5 shows scene flow errors for examples where other leading methods fail to effectively estimate scene flow in foreground regions. Scene flow estimation is challenging for this region as the texture-less car undergoes large displacement and is occluded in the second frame. Our method employing recognition cues performs comparatively better than other methods. Without recognition cues, only very few matches could be established in those regions and most of them would be wrong. We encourage the reader to have a look at our supplementary material for additional qualitative comparisons of our method to other state-of-the-art methods.

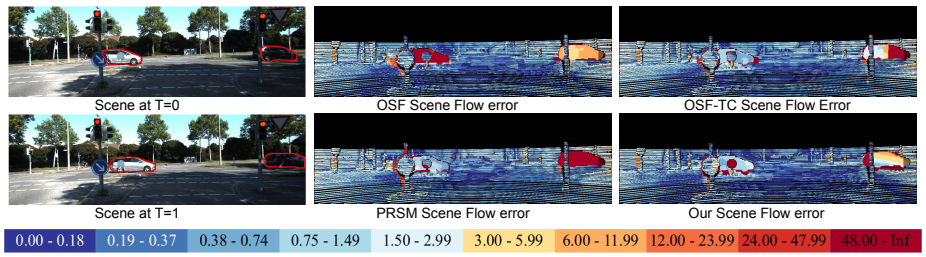


Figure 3.5: **Qualitative Comparison on KITTI-15 Test Set.** The first column shows the input images, followed by scene flow error maps of OSF[52], OSF-TC[95], PRSM[75] and our method using the color scheme depicted in the legend.

3.4.3 3D object coordinates prediction

We model our encoder-decoder network for 3D object coordinate prediction in Caffe [96]. The network is trained by minimizing a robust and smooth Huber loss [97] using the Adam solver with a momentum of 0.9 and learning rate of 1e-5. Furthermore, in order to compare the performance of our CNN model with existing state-of-the-art approaches for predicting the object coordinates, we train a random forest[55] with 3 trees and maximum depth of 64. We use the RGB image and the depth map as input feature and sample 3 million random points during training. We find the quality of 3D object coordinate predictions significantly better using our CNN architecture with an average Euclidean error of 0.6 meters in comparison to an error of 2.89 meters using the random forest method. We attribute the lower error of the CNN based predictions to the network's ability to generalize well to cars with high intra class variation. We refer the reader to the last section of the supplementary material for a detailed description of the CNN architecture we employed and a comparison of the quality of our 3D object coordinate predictions to other state-of-the-art methods.

3.5 Discussion

In this work, we studied the impact of different levels of recognition granularity on the problem of estimating scene flow for dynamic foreground objects. Our results indicate that recognition cues such as 2D bounding box and 2D instance segmentation provide large improvements on foreground parts of the scene in the presence of challenges such as large displacement or local ambiguities. Furthermore, we showed that our method achieves state-of-the-art performance on the challenging KITTI scene flow benchmark.

Chapter 4

iPose: Instance-Aware 6D Pose Estimation of Partly Occluded Objects

4.1 Introduction

Localization of object instances from single input images has been a long-standing goal in computer vision. The task evolved from simple 2D detection to full 6D pose estimation, i.e. estimating the 3D position and 3D orientation of the object relative to the observing camera. Early approaches relied on objects having sufficient texture to match feature points [98]. Later, with the advent of consumer depth cameras, research focused on texture-less objects [99] in increasingly cluttered environments. Today, heavy occlusion of objects is the main performance benchmark for one-shot pose estimation methods.

Recent RGB-D-based methods [100, 101] are robust to moderate degrees of object occlusion. However, depth cameras fail under certain conditions, e.g. with intense sunlight, and RGB cameras are prevalent on many types of devices. Hence, RGB-based methods still have high practical relevance. In this work, we present a system for 6D pose estimation of rigid object instances with heavy occlusion, from single input images. Our method outperforms the state-of-the-art for both RGB and RGB-D input modalities.

During the last decade, computer vision has seen a large shift towards learning-based methods. In particular, deep learning has massively improved accuracy and robustness for many tasks. While 6D object pose estimation has also benefited from deep learning to some extent, with recent methods being able to estimate accurate poses in real time from single RGB images [102, 103, 104], the same does not hold when objects are partly occluded. In this case, aforementioned methods, despite being trained with partly occluded objects, either break down [103, 104] or have to simplify the task by estimating poses from

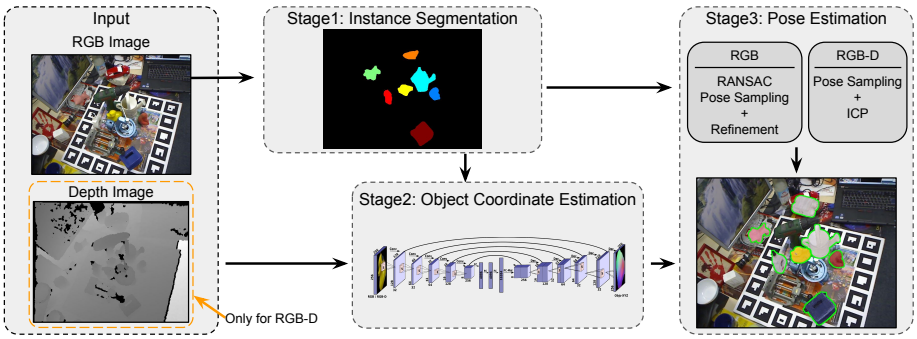


Figure 4.1: **Illustration of our modular, 3-stage pipeline** for both RGB and RGB-D input images.

tight crops around the ground truth object position [102]. To the best of our knowledge, we are the first to show that deep learning can improve results considerably for objects that are moderately to heavily occluded, particularly for the difficult case of RGB input.

At the core, our method decomposes the 6D pose estimation problem into a sequence of three subtasks, or modules (see Fig. 4.1). We first detect the object in 2D, then we locally regress correspondences to the 3D object surface, and, finally, we estimate the 6D pose of the object. With each subtask, we can remove specific aspects of the problem, such as object background and object appearance. In the first module, 2D detection is implemented by an instance segmentation network which estimates a tight mask for each object. Thus, we can separate the object from surrounding clutter and occluders, making the following steps invariant to the object environment, and allowing us to process each detected instance individually. In the second module, we present an encoder-decoder architecture for densely regressing so-called *object coordinates* [105], i.e. 3D points in the local coordinate frame of the object which define 2D-3D correspondences between the image and the object. The third module is a purely geometric pose optimization which is not learned from data because all aspects of object appearance have been removed in the previous steps. Since we estimate 6D poses successively from 2D instance segmentation, we call our approach *iPose*, short for "instance-aware pose estimation".

Our decomposition strategy is conceptually simple, but we show that it is considerably superior to other deep learning-based methods that try to reason about different aspects of these steps jointly. In particular, several recent works propose to extend state-of-the-art object detection networks to output 6D object poses directly. Kehl et al. [103] extend the SSD object detector [106] to recognize discretized view-points of specific objects, i.e. re-formulating pose regression as a classification problem. Similarly, Tekin et al. [104] extend the YOLO object detector [107] by letting image grid cells predict object presence,

and simultaneously the 6D pose. Both approaches are highly sensitive to object occlusion, as we will show in the experimental evaluation. Directly predicting the 6D pose from observed object appearance is challenging, due to limited training data and innumerable occlusion possibilities.

We see three reasons for the success of our approach. Firstly, we exploit the massive progress in object detection and instance segmentation achieved by methods like MNC [54] and Mask R-CNN [108]. This is similar in spirit to the work of [103, 104], but instead of extending the instance segmentation to predict 6D poses directly, we use it as a decoupled component within our step-by-step strategy. Secondly, the rich structural output of our dense object coordinate regression step allows for a geometric hypothesize-and-verify approach that can yield a good pose estimate even if parts of the prediction are incorrect, e.g. due to occlusion. Such a robust geometry-based step is missing in previous deep learning-based approaches [102, 103, 104]. Thirdly, we propose a new data augmentation scheme specifically designed for the task of 6D object pose estimation. Data augmentation is a common aspect of learning-based pose estimation methods, since training data is usually scarce. Previous works have placed objects at random 2D locations over arbitrary background images [109, 102, 103], which yields constellations where objects occlude each other in physically impossible ways. In contrast, our data augmentation scheme infers a common ground plane from ground truth poses and places additional objects in a physically plausible fashion. Hence, our data augmentation results in more realistic occlusion patterns which we found crucial for obtaining good results.
We summarize our main **contributions**:

- We propose *iPose*, a new deep learning architecture for 6D object pose estimation which is remarkably robust with respect to object occlusion, using a new three-step task decomposition approach.
- We are the first to surpass the state-of-the-art for partly occluded objects with a deep learning-based approach for both RGB-D and RGB inputs.
- We present a new data augmentation scheme for object pose estimation which generates physically plausible occlusion patterns, crucial for obtaining good results.

4.2 Related Work

Here, we give an overview of previous methods for 6D object pose estimation. Early pose estimation methods were based on matching sparse features [98] or templates [110]. Templates work well for texture-less objects where sparse feature detectors fail to identify salient points. Hinterstoisser et al. proposed the LINEMOD templates [99], which combine gradient and normal cues for robust object detection given RGB-D inputs. Annotating the template database with viewpoint information facilitates accurate 6D pose estimation [111, 112, 113, 114, 115]. An RGB version of LINEMOD [116] is less suited for pose estimation [109]. In general, template-based methods suffer from sensitivity to occlusion [105].

With a depth channel available, good results have been achieved by voting-based schemes [117, 118, 119, 120, 121, 101]. In particular, Drost et al. [121] cast votes by matching point-pair features which combine normal and distance information. Recently, the method was considerably improved in [101] by a suitable sampling scheme, resulting in a purely geometric method that achieves state-of-the-art results for partly occluded objects given RGB-D inputs. Our deep learning-based pipeline achieves higher accuracy, and can also be applied to RGB images.

Recently, CNN-based methods have become increasingly popular for object pose estimation from RGB images. Rad et al. [102] presented the BB8 pipeline which resembles our decomposition philosophy to some extent. But their processing steps are more tightly coupled. For example, their initial detection stage does not segment the object, and can thus not remove object background. Also, they regress the 6D pose by estimating the 2D location of a sparse set of control points. We show that dense 3D object coordinate regression provides a richer output which is essential for robust geometric pose optimization. Rad et al. [102] evaluate BB8 on occluded objects but restrict pose prediction to image crops around the ground truth object position[1]. Our approach yields superior results for partly occluded objects *without* using prior knowledge about object position.

Direct regression of a 6D pose vector by a neural network, e.g. proposed by Kendall et al. for camera localization [122], exhibits low accuracy [123]. The works discussed in the introduction, i.e. Kehl et al. [103] and Tekin et al. [104], also regress object pose directly but make use of alternative pose parametrizations, namely discrete view point classification [103], or sparse control point regression [104] similar to BB8 [102]. We do *not* predict the 6D pose directly, but follow a step-by-step strategy to robustly obtain the 6D pose despite strong occlusions.

Object coordinates have been used previously for object pose estimation from RGB-D [105, 124, 100] or RGB inputs [109]. In these works, random forest matches image patches to 3D points in the local coordinate frame of the object, and the pose is recovered by robust, geometric optimization. Because few correct correspondences suffice for a pose estimate, these methods are inherently robust to object occlusion. In contrast to our work, they combine object coordinate prediction and object segmentation in a single module, using random forests. These two tasks are disentangled in our approach, with the clear advantage that each individual object mask is known for object coordinate regression. In this context, we are also the first to successfully train a neural network for object coordinate regression of known objects. Overall, we report superior pose accuracy for partly occluded objects using RGB and RGB-D inputs. Note that recently Behl et al. [125] have trained a network for object coordinate regression of vehicles (i.e. object class). However, our network, training procedure, and data augmentation scheme differ from [125].

[1]Their experimental setup relies on ground truth crops and is not explicitly described in [102]. We verified this information via private email exchange with the authors.

To cope well with limited training data, we propose a new data augmentation scheme which generates physically plausible occlusion patterns. While plausible data augmentation is becoming common in object class detection works, see e.g. [126, 127, 128], our scheme is tailored specifically towards object instance pose estimation where previous works resorted to pasting 2D object crops on arbitrary RGB backgrounds [109, 102, 103]. We found physically plausible data augmentation to be crucial for obtaining good results for partly occluded objects.

To summarize, only few previous works have addressed the challenging task of pose estimation of partly occluded objects from single RGB or RGB-D inputs. We present the first viable deep learning approach for this scenario, improving state-of-the-art accuracy considerably for both input types.

4.3 Method

In this section, we describe our three-stage, instance-aware approach for 6D object pose estimation. The overall workflow of our method is illustrated in Fig. 4.1. Firstly, we obtain all object instances in a given image using an instance segmentation network (Sec. 4.3.1). Secondly, we estimate dense 3D object coordinates for each instance using an encoder-decoder network (Sec. 4.3.2). Thirdly, we use the pixel-wise correspondences between predicted object coordinates and the input image to sample 6D pose hypotheses, and further refine them using an iterative geometric optimization (Sec. 4.3.3). In Sec. 4.3.4, we describe our object-centric data augmentation procedure which we use to generate additional training data with realistic occlusions for the encoder-decoder network of step 2.

We denote the RGB input to our pipeline as I and RGB-D input as *I-D*. $\mathcal{K} = \{1, ..., K\}$ is a set of all known object classes, a subset of which could be present in the image. The goal of our method is to take an image $I/$*I-D* containing n objects $\mathcal{O} = \{O_1, ..., O_n\}$, each of which has a class from $\mathcal{K}$, and to estimate their 6D poses. Below, we describe each step of our pipeline in detail.

4.3.1 Stage 1: Instance Segmentation

The first step of our approach, instance segmentation, recognizes the identity of each object, and produces a fine grained mask. Thus we can separate the RGB(-D) information pertaining only to a specific object from surrounding clutter and occluders. To achieve this, we utilize instance segmentation frameworks such as [54, 108]. Given an input I, the output of this network is a set of n instance masks $\mathcal{M} = \{M_1, ..., M_n\}$ and an object class $k \in \mathcal{K}$ for each mask.

4.3.2 Stage 2: Object Coordinate Regression

An object coordinate denotes the 3D position of an object surface point in the object's local coordinate frame. Thus given a pixel location p and its predicted

object coordinate C, a (p, C) pair defines a correspondence between an image I and object O. Multiple such correspondences, at least three for RGB-D data and four for RGB data, are required to recover the 6D object pose (see Sec. 4.3.3). In order to regress pixelwise object coordinates C for each detected object, we use a CNN with an encoder-decoder style architecture with skip connections. The encoder consists of 5 convolutional layers with a stride of 2 in each layer, followed by a set of 3 fully connected layers. The decoder has 5 deconvolutional layers followed by the 3 layer output corresponding to 3-dimensional object coordinates. Skip connections exist between symmetrically opposite conv-deconv layers. As input for this network, we crop a detected object using its estimated mask M, resize and pad the crop to a fixed size, and pass it through the object coordinate network. The output of this network has 3 channels containing the pixelwise X, Y and Z values of object coordinates C for mask M. We train separate networks for RGB and RGB-D inputs.

4.3.3 Stage 3: Pose Estimation

In this section, we describe the geometric pose optimization step of our approach for RGB-D and RGB inputs, respectively. This step is not learned from data, but recovers the 6D object pose from the instance mask M of stage 1 and the object coordinates C of stage 2.

RGB-D Setup. Our pose estimation is inspired by the original object coordinate framework of [105]. Compared to [105], we use a simplified scoring function to rank pose hypotheses, and an Iterative Closest Point (ICP) refinement.

In detail, we use the depth channel and the mask M_O to calculate a 3D point cloud P_O associated with object O with respect to the coordinate frame of the camera. Also, stage 2 yields the pixelwise predicted object coordinates C_O. We seek the 6D pose H_O^* which relates object coordinates C_O with the point cloud P_O. For ease of notation, we drop the subscript O, assuming that we are describing the process for that particular object instance. We randomly sample three pixels j_1, j_2, j_3 from mask M, from which we establish three 3D-3D correspondences (P^{j_1}, C^{j_1}), (P^{j_2}, C^{j_2}), (P^{j_3}, C^{j_3}). We use the Kabsch algorithm [129] to compute the pose hypothesis H_i from these correspondences. Using H_i, we transform C^{j_1}, C^{j_2}, C^{j_3} from the object coordinate frame to the camera coordinate frame. Let these transformed points be T^j. We compute the Euclidean distance, $\|P^j, T^j\|$, and if the distances of all three points are less than 10% of the object diameter, we add H_i to our hypothesis pool. We repeat this process until we have collected 210 hypotheses. For each hypothesis H, we obtain a point cloud $P^*(H)$ in the camera coordinate system via rendering the object CAD model. This lets us score each hypothesis using

$$S_{\text{RGB-D}}(H) = \frac{\sum_{j \in M} \left[||P^j - P^{*j}(H)|| < d/10 \right]}{|M|}, \tag{4.1}$$

where $[\cdot]$ returns 1 if the enclosed condition is true, and the sum is over pixels

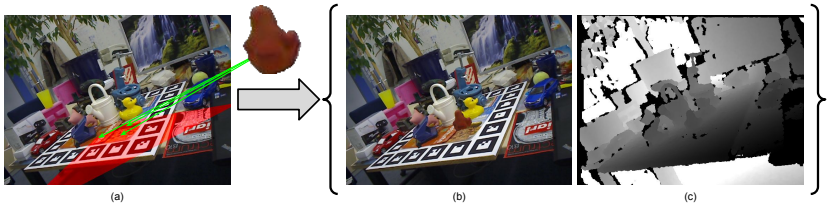


Figure 4.2: **Object centric data augmentation pipeline.** (a) If the cropped object (Ape) is inserted within the red area, it can cause a physically plausible occlusion for the center object (Can). (b) shows the resulting augmented RGB image, and (c) shows the resulting augmented depth image.

inside the mask M and normalized. The score $S_{\text{RGB-D}}(H)$ computes the average number the pixels inside the mask for which the rendered camera coordinates $P^{*j}(H)$ and the observed camera coordinates P^j agree, up to a tolerance of 10% of the object diameter d. From the initial pool of 210 hypotheses we select the top 20 according to the score $S_{\text{RGB-D}}(H)$. Finally, for each selected hypothesis, we perform ICP refinement with P as the target, the CAD model vertices as the source, and H_i as initialization. We choose the pose with the lowest ICP fitting error H_{ICP} for further refinement.

Rendering-Based Refinement. Under the assumption that the estimate H_{ICP} is already quite accurate, and using the instance mask M, we perform the following additional refinement: using H_{ICP}, we render the CAD model to obtain a point cloud P_r of the visible object surface. This is in contrast to the previous ICP refinement where all CAD model vertices were used. We fit P_r inside the mask M to the observed point cloud P via ICP, to obtain a refining transformation H_{ref}. This additional step pushes P_r towards the observed point cloud P, providing a further refinement to H_{ICP}. The final pose is thus obtained by $H^*_{\text{RGB-D}} = H_{\text{ICP}} * H_{\text{ref}}$.

Our instance-based approach is a clear advantage in both refinement steps, since we can use the estimated mask to precisely carve out the observed point cloud for ICP.

RGB Setup. Given RGB data, we follow Brachmann et al. [109] and estimate the pose of the objects through hypotheses sampling [105] and pre-emptive RANSAC [130]. At this stage, the predicted object mask M and the predicted object coordinates C inside the mask are available. For each pixel j at the 2D position p_j inside M, the object coordinate network estimates a 3D point C^j in the local object coordinate system. Thus, we can sample 2D-3D correspondences between 2D points of the image and 3D object coordinate points from the area inside the object mask. Our goal is to search for a pose hypothesis H^* which maximizes the following score:

$$S_{\text{RGB}}(H) = \sum_{j \in M} \left[\|p_j - AHC^j\|_2 < \tau_{\text{in}} \right], \tag{4.2}$$

where A is the camera projection matrix, τ_{in} is a threshold, and $[\cdot]$ is 1 if the

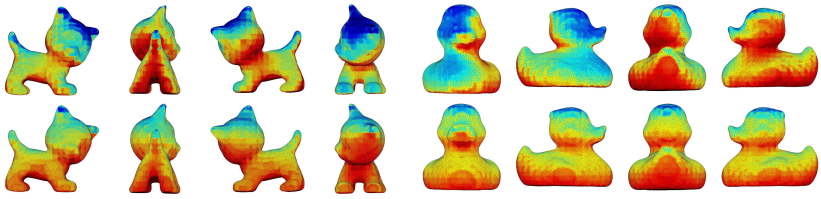


Figure 4.3: **Impact of our data augmentation.** Top row illustrates the on-object occlusion distribution of the base training set before augmentation and the bottom row shows the same for augmented data using our object centric data augmentation. Red indicates that the part is often occluded and blue indicates rare occlusion in the dataset.

statement inside the bracket is true, otherwise 0. The score $S_{\text{RGB}}(H)$ counts the number of pixel-residuals of re-projected object coordinate estimates which are below τ_{in}. We use pre-emptive RANSAC to maximize this objective function. We start by drawing four correspondences from the predicted mask M. Then, we solve the perspective-n-point problem (PnP) [131, 132] to obtain a pose hypothesis. If the re-projection error of the initial four correspondences is below threshold τ_{in} we keep the hypothesis. We repeat this process until 256 pose hypotheses have been collected. We score each hypothesis with $S_{\text{RGB}}(H)$, but only using a sub-sampling of N pixels inside the mask for faster computation. We sort the hypotheses by score and discard the lower half. We refine the remaining hypotheses by re-solving PnP using their inlier pixels according to $S_{\text{RGB}}(H)$. We repeat scoring with an increased pixel count N, discarding and refining hypotheses until only one hypothesis H^*_{RGB} remains as the final estimated pose.

4.3.4 Data Augmentation

Data augmentation is crucial for creating the amount of data necessary to train a CNN. Additionally, data augmentation can help to reduce dataset bias, and introduce novel examples for the network to train on. One possibility for data augmentation is to paste objects on a random background, where mutually overlapping objects occlude each other. This is done e.g. in [109, 102, 103] and we found this strategy sufficient for training our instance segmentation network in step 1. However, the resulting images and occlusion patterns are highly implausible, especially for RGB-D data where objects float in the scene, and occlude each other in physically impossible ways. Training the object coordinate network in step 2 with such implausible data made it difficult for the network to converge and also introduced bias towards impossible object occlusion configurations. In the following, we present an object-centric data augmentation strategy which generates plausible object occlusion patterns, and analyze its impact on the dataset. We assume that for each target object k in the set of all known objects $\mathcal{K}$, a sequence of images is available where the object is not

occluded. For each image, we compute the ground plane on which the target object stands, as well as the distance between its base point and the camera. Then, as shown in Fig. 4.2(a)(red), a surface of interest is defined on the ground plane in front of the target object, representing a cone with an opening angle of 90°. Next, we search for images of other objects in $\mathcal{K}$, where the ground plane normal is close to that of the target object, and which are located in the defined surface of interest, based on their distance from the camera. Finally, by overlaying one or more of these chosen objects in front of the target object, we generate multiple augmented RGB and depth images (cf. Fig. 4.2(b,c)). Using this approach, the resulting occlusion looks physically correct for both the RGB and depth images.

To analyze the impact of our data augmentation scheme, we visualize the distribution of partial occlusion on the object surface in the following way: we first discretize the 3D bounding box surrounding each object into $20 \times 20 \times 20$ voxels. Using the ground truth 6D pose and the 3D CAD model, we can render the full mask of the object. Each pixel that lies inside the rendered mask but not inside the ground truth mask is occluded. We can look-up the ground truth object coordinate of each occluded pixel, and furthermore the associated bounding box voxel. We use the voxels as histogram bins and visualize the occlusion frequency as colors on the surface of the 3D CAD model.

The impact of our object-centric data augmentation for two objects of the LINEMOD dataset [111] is illustrated in Fig. 4.3. Firstly, by looking at the visualization (top row), we notice that the un-augmented data contains biased occlusion samples (irregular distribution of blue and red patches) which could induce overfitting on certain object parts, leading to reduced performance of the object coordinate network of step 2. In the second row, we see that the augmented data has a more regular distribution of occlusion. This visualization reveals the bias in the base training set, and demonstrates the efficacy of our object-centric data augmentation procedure in creating unbiased training data.

4.4 Experiments

In this section, we present various experiments quantifying the performance of our approach. In Sec. 4.4.1, we introduce the dataset which we use for evaluating our system. In Sec. 4.4.2, we compare the performance of our approach to existing RGB and RGB-D-based pose estimation approaches. In Sec. 4.4.2, we analyze the contribution of various modules of our approach to the final pose estimation performance. Finally, in Sec. 4.4.3 and 4.4.4, we discuss the performance of our instance segmentation and object coordinate estimation networks. Please see the supplemental materials for a complete list of parameter settings of our pipeline.

4.4.1 Datasets and Implementation

We evaluate our approach on *occludedLINEMOD*, a dataset published by Brachmann et al. [105]. It was created from the LINEMOD dataset [111] by annotating ground truth 6D poses for various objects in a sequence of 1214 RGB-D images. The objects are located on a table and embedded in dense clutter. Ground truth poses are provided for eight of these objects which, depending on the camera view, heavily occlude each other, making this dataset very challenging. We test both our RGB and RGB-D-based methods on this dataset.

To train our system, we use a separate sequence from the LINEMOD dataset which was annotated by Michel et al. [100]. For ease of reference we call this the LINEMOD-M dataset. LINEMOD-M comes with ground truth annotations of seven objects with mutual occlusion. One object of the test sequence, namely the Driller, is not present in this training sequence, so we do not report results for it. The training sequence is extremely limited in the amount of data it provides. Some objects are only seen from few viewpoints and with little occlusion, or occlusion affects only certain object parts.

Training Instance Segmentation. To train our instance segmentation network with a wide range of object viewpoints and diverse occlusion examples, we create synthetic images in the following way. We use RGB backgrounds from the NYUD dataset [133], and randomly overlay them with objects picked from the original LINEMOD dataset [111]. While this data is physically implausible, we found it sufficient for training the instance segmentation component of our pipeline. We combine these synthetic images with LINEMOD-M to obtain 9000 images with ground truth instance masks. We use Mask R-CNN [108] as our instance segmentation method. For training, we use a learning rate of $1e$-3, momentum of 0.9 and weight decay of $1e$-4. We initialize Mask R-CNN with weights trained on ImageNet, and finetune on our training set.

Training Object Coordinate Regression. For training the object coordinate estimation network, we found it important to utilize physically plausible data augmentation for best results. Therefore, we use the LINEMOD-M dataset along with the data obtained using our object-centric data augmentation pipeline described in Sec. 4.3.4. Note that the test sequence and our training data are strictly separated, i.e. we did not use parts of the test sequence for data augmentation. We trained our object coordinate network by minimizing a robust Huber loss function [134] using ADAM [135]. We train a separate network for each object. We rescale inputs and ground truth outputs for the network to 256x256px patches.

4.4.2 Pose Estimation Accuracy

RGB Setup.

We estimate object poses from RGB images ignoring the depth channel. We evaluate the performance using the *2D Projection* metric introduced by Brachmann et al. [109]. This metric measures the average re-projection error of 3D

	Acceptance Threshold: 5 px			Acceptance Threshold: 10 px				
	BB8[102] (GT crops)	Brachmann [109]	**Ours**	BB8[102] (*GT crops*)	Brachmann [109]	SSD-6D [103]	SSS-6D [104]	**Ours**
Ape	*28.5%*	**31.8%**	24.2%	*81.0%*	51.8%	0.5%	0%	**56.1%**
Can	*1.2%*	4.5%	**30.2%**	*27.8%*	19.1%	0.6%	0%	**72.4%**
Cat	*9.6%*	1.1%	**12.3%**	*61.8%*	7.1%	0.1%	0%	**39.7%**
Duck	*6.8%*	1.6%	**12.1%**	*41.3%*	6.4%	0%	5%	**50.1%**
Glue	*4.7%*	0.5%	**25.9%**	*37.7%*	6.4%	0%	0%	**55.1%**
HoleP.	*2.4%*	6.7%	**20.6%**	*45.4%*	2.6%	0.3%	1%	**61.2%**
Avg	*8.9%*	7.7%	**20.8%**	*49.2%*	17.1%	0.3%	0.01%	**56.0%**

Table 4.1: **Results using RGB only.** Comparison of our pose estimation accuracy for RGB inputs with competing methods. *Italic* numbers were generated using ground truth crops, thus they are not directly comparable.

model vertices transformed by the ground truth pose and the estimated pose. A pose is accepted if the average re-projection error is less than a threshold.

In Table 4.1, we compare the performance of our pipeline to existing RGB-based methods using two different thresholds for the 2D projection metric. We see that our approach outperforms the previous works for most of the objects significantly. Our RGB only pipeline surpasses the state-of-the-art for a 5 pixel threshold by 13% and for a 10 pixel threshold by 39% on average. Note that the results of BB8 [102] were obtained from image crops around the ground truth object position. Similar to [102] and [104], we do not report results for *EggBox* since we could not get reasonable results for this extremely occluded object using RGB only. Note that SSD-6D [103] and SSS-6D [104] completely fail for partly occluded objects. We obtained the results of SSS-6D directly from [104], and of SSD-6D [103] using their publicly available source code and their pre-trained model. However, they did not release their pose refinement method, thus we report their performance without refinement. In the supplement, we show the accuracy of SSD-6D using different 2D re-projection thresholds. Most of the detections of SSD-6D are far off (see also their detection performance in Fig. 4.5, right), therefore we do not expect refinement to improve their results much. We show qualitative pose estimation results for the RGB setting in Fig 4.4.

RGB-D Setup.

Similar to the RGB setup, we measure accuracy as the percentage of correctly estimated poses. Following Hinterstoisser et al. [111], we accept a pose if the average 3D distance between object model vertices transformed using ground truth pose and predicted pose lies below 10% of the object diameter.

In Fig. 4.6, left, we compare the performance of our approach to Michel et al. [100] and Hinterstoisser et al. [101]. We significantly outperform the state-of-the-art on average by 6%, and show massive improvements for some objects. Fig. 4.7 shows qualitative results from our RGB-D pipeline and an illustration of the performance of our method on an object under increasing occlusion. Fig. 4.6, right represents the percentage of correct poses as a function of occluded object

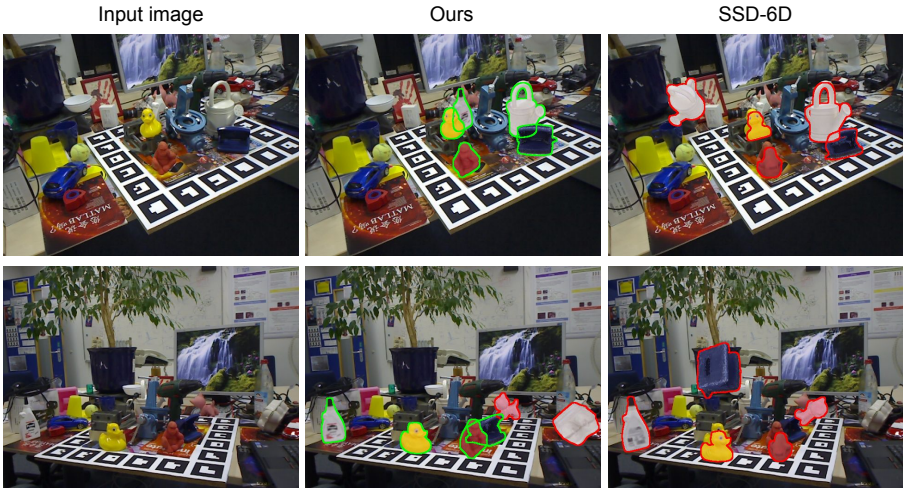


Figure 4.4: **Qualitative results from the RGB setup.** From left to right: input image, our results, results of SSD-6D [103].

surface. We see that for cases of mild occlusion, our method surpasses accuracy of 90% for all objects. For cases of heavy occlusion (above 60%) our method can still recover accurate poses.

Ablation Study.

We investigate the contribution of each step of our method towards the final pose estimation accuracy for the RGB-D setup. As discussed before, our method consists of three steps, namely instance mask estimation, object coordinate regression and pose estimation. We compare to the method of Brachmann et al. [105] which has similar steps, namely soft segmentation (not instance-aware), object coordinate regression, and a final RANSAC-based pose estimation. The first two steps in [105] are implemented using a random forest, compared to two separate CNNs in our system. Fig 4.5, left shows the accuracy for various re-combinations of these modules. The first row is the standard baseline approach of [105] which achieves an average accuracy of 52.9%. In the second row, we replace the soft segmentation estimated by [105] with an instance segmentation method, Multi-task Network Cascades (MNC) [54]. The instance masks effectively constrain the 2D search space which leads to better sampling of correspondences between depth points and object coordinate predictions. Next, we replace the object coordinate predictions of the random forest with our CNN-based predictions. Although we still perform the same pose optimization, this achieves a 4.6% performance boost, showing that our encoder-decoder network architecture predicts object coordinates more precisely. Next, we use

Mask	Obj. Coord.	Pose Estimation	Accuracy
RF[16]	RF[16]	Brachmann [16]	52.9%
Ours (MNC)	RF[16]	Brachmann [16]	56.4%
Ours (MNC)	Ours (CNN)	Brachmann [16]	61.0%
Ours (MNC)	Ours (CNN)	Ours	75.7%
Ours (Mask R-CNN)	Ours (CNN)	Ours	**80.7%**

Method	MAP
Hinterstoisser [3]	0.21
Brachmann [17]	0.51
SSD-6D [14]	0.38
SSS-6D [15]	0.48
Ours	**0.84**

Figure 4.5: **Left.** Pose estimation accuracies on the RGB-D dataset using various combinations of mask estimation, object coordinates estimation and pose estimation approaches. **Right.** Comparison of 2D detection performance.

Object	Michel et al. [4]	Hinterstoisser et al. [5]	Ours
Ape	80.7%	81.4%	**83.0%**
Can	88.5%	**94.7%**	89.6%
Cat	**57.8%**	55.2%	57.5%
Duck	74.4%	**79.7%**	76.6%
Eggbox	47.6%	65.5%	**82.1%**
Glue	73.8%	52.1%	**78.8%**
Holep.	96.3%	95.5%	**97.0%**
Avg.	74.2%	74.9%	**80.7%**

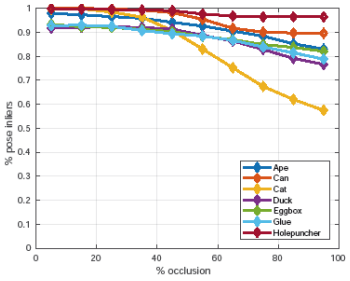


Figure 4.6: **Left.** Comparison of our pose estimation accuracy (RGB-D) with competing methods. **Right.** The percentage of correctly estimated poses as a function of the level of occlusion.

the instance masks as above and object coordinates from our network with our geometric ICP-based refinement which further boosts the accuracy to 75.7%. Finally, in the last row, we use our full pipeline with masks from Mask R-CNN followed by our other modules to achieve state-of-the-art performance of 80.7%. The table clearly indicates that the accuracy of our pipeline as a whole improves when any of the modules improve. On the other hand, we trained our Obj.Coord. network without the proposed data augmentation, and observe a decline in average pose accuracy from 80.7% to 73.2% (-7.4%).

4.4.3 Instance Segmentation

The performance of instance segmentation is crucial for our overall accuracy. Fig. 4.5, right shows the mean average precision of our method for a 2D bounding box IoU > 0.5 compared to other methods. Since our RGB only instance segmentation network is used for both, the RGB and RGB-D setting, the MAP is equal for both settings. We significantly outperform all other pose estimation methods, showing that our decoupled instance segmentation step can reliably detect objects, making the task for the following modules considerably easier.

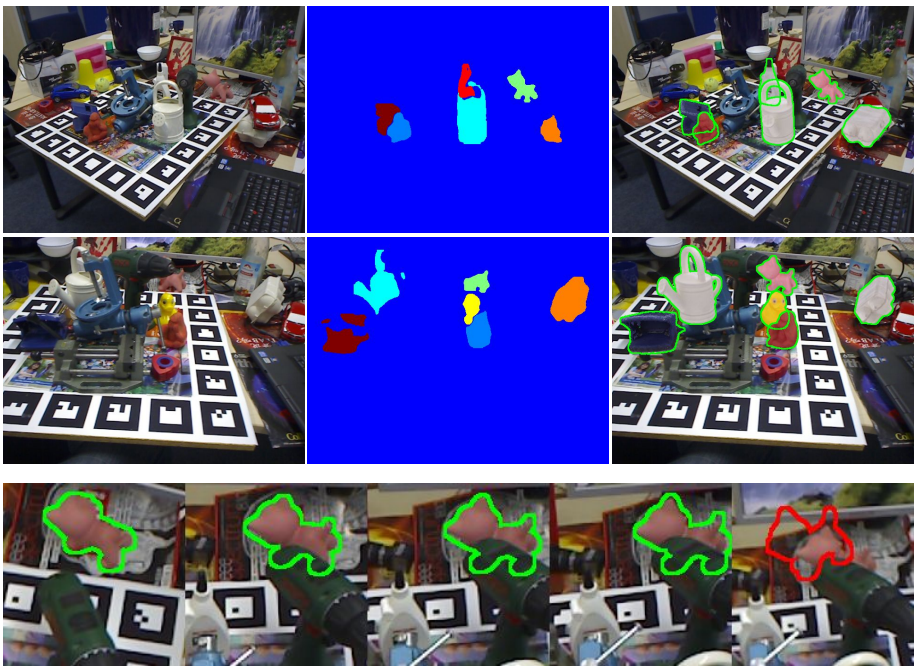


Figure 4.7: **Qualitative results from the RGB-D setup.** Our approach reliably estimates poses for objects which are heavily occluded. (First two rows) The middle column shows estimated object masks of our instance segmentation step. (Last row) We show a sequence of estimated poses for the Cat object under increasing occlusion. We reliably estimate the correct pose until ca. 50% of the object is occluded.

4.4.4 Object Coordinate Estimation

We trained our object coordinate network with and without our data augmentation procedure (Sec. 4.3.4). We measure the average inlier rate, i.e. object coordinate estimates that are predicted within 2cm of ground truth object coordinates. When the network is trained only using the LINEMOD-M dataset, the average inlier rate is 44% as compared to 52% when we use the data created using our object centric data augmentation procedure. A clear 8% increase in the inlier rate shows the importance of our proposed data augmentation.

4.5 Discussion

We have presented *iPose*, a deep learning-based approach capable of estimating accurate poses of partly occluded objects. Our approach surpasses the state-of-the-art for both RGB and RGB-D inputs. We attribute the success of our method to our decomposition philosophy, and therefore the ability to leverage state-of-the-art instance segmentation networks. We are also the first to successfully train an encoder-decoder network for dense object coordinate regression, that facilitates our robust geometric pose optimization.

Chapter 5

Deep Object Co-Segmentation

5.1 Introduction

Object co-segmentation is the task of segmenting the common objects from a set of images. It is applied in various computer vision applications and beyond, such as browsing in photo collections [136], 3D reconstruction [137], semantic segmentation [138], object-based image retrieval [139], video object tracking and segmentation [136], and interactive image segmentation [136].

There are different challenges for object co-segmentation with varying level of difficulty: (1) Rother et al. [136] first proposed the term of *co-segmentation* as the task of segmenting the *common parts* of an image pair simultaneously. They showed that segmenting two images jointly achieves better accuracy in contrast to segmenting them independently. They assume that the common parts have similar appearance. However, the background in both images are significantly different, see Fig. 5.1(a). (2) Another challenge is to segment the same object instance or similar objects of the same class with low intra-class variation, even with similar background [140, 139], see Fig. 5.1(b). (3) A more challenging task is to segment common objects from the same class with large variability in terms of scale, appearance, pose, viewpoint and background [141], see Fig. 5.1(c).

All of the mentioned challenges assume that the image set contains only one common object and the common object should be salient within each image. In this work, we address a more general problem of co-segmentation without this assumption, i.e. multiple object classes can be presented within the images, see Fig. 5.1(d). As it is shown, the co-segmentation result for one specific image including multiple objects can be different when we pair it with different images. Additionally, we are interested in co-segmenting objects, i.e. *things* rather than *stuff*. The idea of object co-segmentation was introduced by Vicente et al. [139] to emphasize the resulting segmentation to be a *thing* such as a 'cat' or

a 'monitor', which excludes common, or uncommon, *stuff* classes like 'sky' or 'sea'.

Figure 5.1: **Different co-segmentation challenges:** (a) segmenting common parts, in terms of small appearance deviation, with varying background [136], (b) segmenting common objects from the same class with low intra-class variation but similar background [140, 142], (c) segmenting common objects from the same class with large variability in terms of scale, appearance, pose, viewpoint and background [141]. (d) segmenting common objects in images including more than one object from multiple classes. Second row shows our predicted co-segmentation of these challenging images.

Segmenting objects in an image is one of the fundamental tasks in computer vision. While image segmentation has received great attention during the recent rise of deep learning [143, 144, 145, 146, 147], the related task of object co-segmentation remains largely unexplored by newly developed deep learning techniques. Most of the recently proposed object co-segmentation methods still rely on models without feature learning. This includes methods utilizing superpixels, or proposal segments [139, 148] to extract a set of object candidates, or methods which use a complex CRF model [149, 147] with hand-crafted features [147] to find the segments with the highest similarity.

In this work, we propose a simple yet powerful method for segmenting objects of a common semantic class from a pair of images using a convolutional encoder-decoder neural network. Our method uses a pair of Siamese encoder networks to extract semantic features for each image. The mutual correlation layer at the network's bottleneck computes localized correlations between the semantic features of the two images to highlight the heat-maps of common objects. Finally, the Siamese decoder networks combine the semantic features from each image with the correlation features to produce detailed segmentation masks through a series of deconvolutional layers. Our approach is trainable in

an end-to-end manner and does not require any, potentially long runtime, CRF optimization procedure at evaluation time. We perform an extensive evaluation of our deep object co-segmentation and show that our model can achieve state-of-the-art performance on multiple common co-segmentation datasets. In summary, our main contributions are as follows:

- We propose a simple yet effective convolutional neural network (CNN) architecture for object co-segmentation that can be trained end-to-end. To the best of our knowledge, this is the first pure CNN framework for object co-segmentation, which does not depend on any hand-crafted features.
- We achieve state-of-the-art results on multiple object co-segmentation datasets, and introduce a challenging object co-segmentation dataset by adapting Pascal dataset for training and testing object co-segmentation models.

5.2 Related Work

We start by discussing object co-segmentation by roughly categorizing them into three branches: co-segmentation without explicit learning, co-segmentation with learning, and interactive co-segmentation. After that, we briefly discuss various image segmentation tasks and corresponding approaches based on CNNs.

Co-Segmentation without Explicit Learning. Rother et al. [136] proposed the problem of image co-segmentation for image pairs. They minimize an energy function that combines an MRF smoothness prior term with a histogram matching term. This forces the histogram statistic of common foreground regions to be similar. In a follow-up work, Mukherjee et al. [150] replace the l_1 norm in the cost function by an l_2 norm. In [151], Hochbaum and Singh used a reward model, in contrast to the penalty strategy of [136]. In [142], Vicente et al. studied various models and showed that a simple model based on Boykov-Jolly [152] works the best. Joulin et al. [153] formulated the co-segmentation problem in terms of a discriminative clustering task. Rubio et al. [154] proposed to match regions, which results from an over-segmentation algorithm, to establish correspondences between the common objects. Rubinstein et al. [141] combined a visual saliency and dense correspondences, using SIFT flow, to capture the sparsity and visual variability of the common object in a group of images. Fu et al. [155] formulated object co-segmentation for RGB-D input images as a fully-connected graph structure, together with mutex constraints. In contrast to these works, our method is a pure learning based approach.

Co-Segmentation with Learning. In [139], Vicente et al. generated a pool of object-like proposal-segmentations using constrained parametric min-cut [156]. Then they trained a random forest classifier to score the similarity of a pair of segmentation proposals. Yuan et al. [157] introduced a deep dense conditional random field framework for object co-segmentation by inferring co-occurrence maps. These co-occurrence maps measure the objectness scores, as well as, similarity evidence for object proposals, which are generated using selective search [158]. Similar to the constrained parametric min-cut, selective search also uses

hand-crafted SIFT and HOG features to generate object proposals. Therefore, the model of [157] cannot be trained end-to-end. In addition, [157] assume that there is a single common object in a given image set, which limits application in real-world scenarios. Recently, Quan et al. [147] proposed a manifold ranking algorithm for object co-segmentation by combining low-level appearance features and high-level semantic features. However, their semantic features are pre-trained on the ImageNet dataset. In contrast, our method is based on a pure CNN architecture, which is free of any hand-crafted features and object proposals and does not depend on any assumption about the existence of common objects.

Interactive Co-Segmentation. Batra et al. [140] firstly presented an algorithm for interactive co-segmentation of a foreground object from a group of related images. They use users' scribbles to indicate the foreground. Collins et al. [159] used a random walker model to add consistency constraints between foreground regions within the interactive co-segmentation framework. However, their co-segmentation results are sensitive to the size and positions of users' scribbles. Dong et al. [160] proposed an interactive co-segmentation method which uses global and local energy optimization, whereby the energy function is based on scribbles, inter-image consistency, and a standard local smoothness prior. In contrast, our work is not a user-interactive co-segmentation approach.

Convolutional Neural Networks for Image Segmentation. In the last few years, CNNs have achieved great success for the tasks of image segmentation, such as semantic segmentation [143, 161, 162, 163, 146, 164], interactive segmentation [146, 165], and salient object segmentation [166, 167, 168].

Semantic segmentation aims at assigning semantic labels to each pixel in an image. Fully convolutional networks (FCN) [143] became one of the first popular architectures for semantic segmentation. Nor et al. [161] proposed a deep deconvolutional network to learn the upsampling of low-resolution features. Both U-Net [144] and SegNet [169] proposed an encoder-decoder architecture, in which the decoder network consists of a hierarchy of decoders, each corresponding to an encoder. Yu *et al.* [162] and Chen *et al.* [170] proposed dilated convolutions to aggregate multi-scale contextual information, by considering larger receptive fields. Salient object segmentation aims at detecting and segmenting the salient objects in a given image. Recently, deep learning architectures have become popular for salient object segmentation [166, 167, 168]. Li and Yu [166] addressed salient object segmentation using a deep network which consists of a pixel-level multi-scale FCN and a segment scale spatial pooling stream. Wang et al. [167] proposed recurrent FCN to incorporate saliency prior knowledge for improved inference, utilizing a pre-training strategy based on semantic segmentation data. Jain et al. [168] proposed to train a FCN to produce pixel-level masks of all object-like regions given a single input image.

Although CNNs play a central role in image segmentation tasks, there has been no prior work with a pure CNN architecture for object co-segmentation. To the best of our knowledge, our deep CNN architecture is the first of its kind for object co-segmentation.

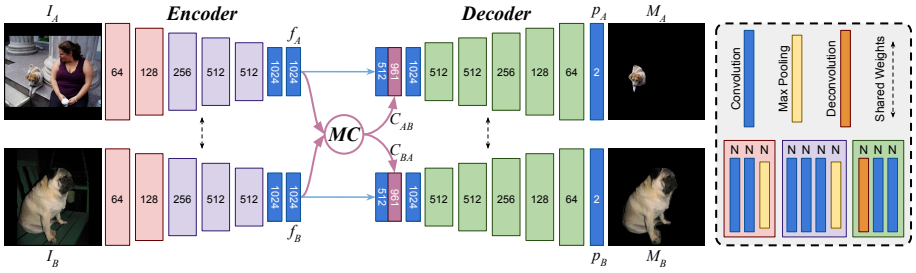


Figure 5.2: **Deep Object Co-Segmentation Network**. Our network includes three parts: (i) passing input images I_A and I_B through a Siamese encoder to extract feature maps f_A and f_B, (ii) using a mutual correlation network to perform feature matching to obtain correspondence maps C_{AB} and C_{BA}, (iii) passing concatenation of squeezed feature maps and correspondence maps through a Siamese decoder to get the common objects masks M_A and M_B.

5.3 Method

In this section, we introduce a new CNN architecture for segmenting the common objects from two input images. The architecture is end-to-end trainable for the object co-segmentation task. Fig. 5.2 illustrates the overall structure of our architecture. Our network consists of three main parts: (1) Given two input images I_A and I_B, we use a Siamese encoder to extract high-level semantic feature maps f_A and f_B. (2) Then, we propose a mutual correlation layer to obtain correspondence maps C_{AB} and C_{BA} by matching feature maps f_A and f_B at pixel-level. (3) Finally, given the concatenation of the feature maps f_A and f_B and correspondence maps C_{AB} and C_{BA}, a Siamese decoder is used to obtain and refine the common object masks M_A and M_B.

In the following, we first describe each of the three parts of our architecture in detail. Then in Sec 5.3.4, the loss function is introduced. Finally, in Sec 5.3.5, we explain how to extend our approach to handle co-segmentation of a group of images, i.e. going beyond two images.

5.3.1 Siamese Encoder

The first part of our architecture is a Siamese encoder which consists of two identical feature extraction CNNs with shared parameters. We pass the input image pair I_A and I_B through the Siamese encoder network pair to extract feature maps f_A and f_B. More specifically, our encoder is based on the VGG16 network [171]. We keep the first 13 convolutional layers and replace *fc6* and *fc7* with two 3×3 convolutional layers *conv6-1* and *conv6-2* to produce feature maps which contain more spatial information. In total, our encoder network has 15 convolutional layers and 5 pooling layers to create a set of high-level semantic features f_A and f_B. The input to the Siamese encoder is two 512×512 images

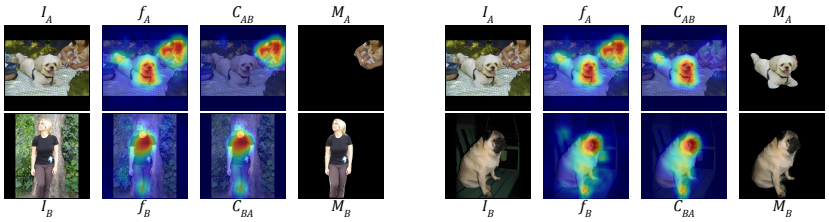


Figure 5.3: **The visualization of the heat-maps.** Given a pair of input images I_A and I_B, after passing them through the Siamese encoder, we extract feature maps f_A and f_B. We use the mutual correlation layer to perform feature matching to obtain correspondence maps C_{AB} and C_{BA}. Then, using our Siamese decoder we predict the common objects masks M_A and M_B. As shown before correlation layer, the heat-maps are covering all the objects inside the images. After applying the correlation layer, the heat-maps on uncommon objects are filtered out. Therefore, we utilize the output of the correlation layer to guide the network for segmenting the common objects.

and the output of the encoder is two 1024-channel feature maps with a spatial size of 16×16.

5.3.2 Mutual Correlation

The second part of our architecture is a mutual correlation layer. The outputs of encoders f_A and f_B represent the high-level semantic content of the input images. When the two images contain objects that belong to a common class, they should contain similar features at the locations of the shared objects. Therefore, we propose a mutual correlation layer to compute the correlation between each pair of locations on the feature maps. The idea of utilizing the correlation layer is inspired by Flownet [172], in which the correlation layer is used to match feature points between frames for optical flow estimation. Our motivation of using the correlation layer is to filter the heat-maps (high-level features), which are generated separately for each input image, to highlight the heat-maps on the common objects (see Fig. 5.3). In detail, the mutual correlation layer performs a pixel-wise comparison between two feature maps f_A and f_B. Given a point (i, j) and a point (m, n) inside a patch around (i, j), the correlation between feature vectors $f_A(i, j)$ and $f_B(m, n)$ is defined as

$$C_{AB}(i, j, k) = \langle\ f_A(i, j), f_B(m, n)\rangle \tag{5.1}$$

where $k = (n - j)D + (m - i)$ and $D \times D$ is patch size. Since the common objects can locate at any place on the two input images, we set the patch size to $D = 2 * max(w - 1, h - 1) + 1$, where w and h are the width and height of the feature maps f_A and f_B. The output of the correlation layer is a feature map C_{AB} of size $w \times\ h \times D^2$. We use the same method to compute the correlation map C_{BA} between f_B and f_A.

5.3.3 Siamese Decoder

The Siamese decoder is the third part of our architecture, which predicts two foreground masks of the common objects. We squeeze the feature maps f_A and f_B and concatenate them with their correspondence maps C_{AB} and C_{BA} as the input to the Siamese decoder (Fig. 5.2). The same as the Siamese encoder, the decoder is also arranged in a Siamese structure with shared parameters. There are five blocks in our decoder, whereby each block has one deconvolutional layer and two convolutional layers. All the convolutional and deconvolutional layers in our Siamese decoder are followed by a ReLU activation function. By applying a Softmax function, the decoder produces two probability maps p_A and p_B. Each probability map has two channels, background and foreground, with the same size as the input images.

5.3.4 Loss Function

We define our object co-segmentation as a binary image labeling problem and use the standard cross entropy loss function to train our network. The full loss score $\mathcal{L}_{AB}$ is then estimated by $\mathcal{L}_{AB} = \mathcal{L}_A + \mathcal{L}_B$, where the $\mathcal{L}_A$ and the $\mathcal{L}_B$ are cross-entropy loss functions for the image A and the image B, respectively.

5.3.5 Group Co-Segmentation

Although our architecture is trained for image pairs, our method can handle a group of images. Given a set of N images $\mathcal{I} = \{I_1, ..., I_N\}$, we pair each image with $K \leq N-1$ other images from $\mathcal{I}$. Then, we use our DOCS network to predict the probability maps for the pairs, $\mathcal{P} = \left\{p_n^k : 1 \leq n \leq N, 1 \leq k \leq K\right\}$, where p_n^k is the predicted probability map for the kth pair of image I_n. Finally, we compute the final mask M_n for image I_n as

$$M_n(x, y) = \text{median}\{p_n^k(x, y)\} > \sigma. \tag{5.2}$$

where σ is the acceptance threshold. In this work, we set $\sigma = 0.5$. We use the median to make our approach more robust to groups with outliers.

5.4 Experiments

5.4.1 Datasets

Training a CNN requires a lot of data. However, existing co-segmentation datasets are either too small or have a limited number of object classes. The MSRC dataset [173] was first introduced for supervised semantic segmentation, then a subset was used for object co-segmentation [139]. This subset of MSRC only has 7 groups of images and each group has 10 images. The iCoseg dataset, introduced in [140], consists of several groups of images and is widely used to evaluate co-segmentation methods. However, each group contains images of the same object instance or very similar objects from the same class. The Internet

dataset [141] contains thousands of images obtained from the Internet using image retrieval techniques. However, it only has three object classes: *car, horse* and *airplane*, where images of each class are mixed with other noise objects. In [174], Faktor and Irani use PASCAL dataset for object co-segmentation. They separate the images into 20 groups according to the object classes and assume that each group only has one object. However, this assumption is not common for natural images.

Inspired by [174], we create an object co-segmentation dataset by adapting the PASCAL dataset labeled by [175]. The original dataset consists of 20 foreground object classes and one background class. It contains $8,498$ training and $2,857$ validation pixel-level labeled images. From the training images, we sampled $161,229$ pairs of images, which have common objects, as a new co-segmentation training set. We used PASCAL validation images to sample $42,831$ validation pairs and $40,303$ test pairs. Since our goal is to segment the common objects from the pair of images, we discard the object class labels and instead we label the common objects as foreground. Fig. 5.1(d) shows some examples of image pairs of our object co-segmentation dataset. In contrast to [174], our dataset consists of image pairs of one or more arbitrary common classes.

5.4.2 Implementation Details and Runtime

We use the Caffe framework [50] to design and train our network. We use our co-segmentation dataset for training. We did not use any images from the MSRC, Internet or iCoseg datasets to fine tune our model. The *conv1-conv5* layers of our Siamese encoder (VGG-16 net [171]) are initialized with weights trained on the Imagenet dataset [176]. We train our network on one GPU for 100K iterations using Adam solver [177]. We use small mini-batches of 10 image pairs, a momentum of 0.9, a learning rate of $1e-5$, and a weight decay of 0.0005.

Our method can handle a large set of images in linear time complexity $\mathcal{O}(N)$. As mentioned in Sec. 5.3.5 in order to co-segment an image, we pair it with K ($K \leq N-1$) other images. In our experiments, we used all possible pairs to make the evaluations comparable to other approaches. Although in this case our time complexity is quadratic $\mathcal{O}(N^2)$, our method is significantly faster than others.

Number of images	Others time	Our time
2	8 minutes [153]	0.1 seconds
30	4 to 9 hours [153]	43.5 seconds
30	22.5 minutes [178]	43.5 seconds
418 (14 categories, $\sim$ 30 images per category)	29.2 hours [174]	10.15 minutes
418 (14 categories, $\sim$ 30 images per category)	8.5 hours [179]	10.15 minutes

To show the influence of number of pairs K, we validate our method on the Internet dataset w.r.t. K (Table 5.1). Each image is paired with K random images from the set. As shown, we achieve state-of-the-art performance even with $K = 10$. Therefore, the complexity of our approach is $\mathcal{O}(KN) = \mathcal{O}(N)$ which is linear with respect to the group size.

Table 5.1: Influence of number of pairs K.

Internet	K=10		K=20		K=99(all)	
(N=100)	P	J	P	J	P	J
Car	93.93	82.89	93.91	82.85	93.90	82.81
Horse	92.31	69.12	92.35	69.17	92.45	69.44
Airplane	94.10	65.37	94.12	65.45	94.11	65.43
Average	93.45	72.46	93.46	72.49	93.49	72.56

5.4.3 Results

We report the performance of our approach on MSRC [173, 142], Internet [141], and iCoseg [140] datasets, as well as our own co-segmentation dataset.

Metrics.

For evaluating the co-segmentation performance, there are two common metrics. The first one is *Precision*, which is the percentage of correctly segmented pixels of both foreground and background masks. The second one is *Jaccard*, which is the intersection over union of the co-segmentation result and the ground truth foreground segmentation.

PASCAL Co-Segmentation.

As we mentioned in Sec 5.4.1, our co-segmentation dataset consists of 40, 303 test image pairs. We evaluate the performance of our method on our co-segmentation test data. We also tried to obtain the common objects of same classes using a deep semantic segmentation model, here FCN8s [143]. First, we train FCN8s with the PASCAL dataset. Then, we obtain the common objects from two images by predicting the semantic labels using FCN8s and keeping the segments with common classes as foreground. Our co-segmentation method (**94.2%** for *Precision* and **64.5%** for *Jaccard*) outperforms FCN8s (**93.2%** for *Precision* and **55.2%** for *Jaccard*), which uses the same VGG encoder, and trained with the same training images. The improvement is probably due to the fact that our DOCS architecture is specifically designed for the object co-segmentation task, which FCN8s is designed for the semantic labeling problem. Another potential reason is that generating image pairs is a form of data augmentation. We would like to exploit these ideas in the future work. Fig. 5.4 shows the qualitative results of our approach on the PASCAL co-segmentation dataset. We can see that our method successfully extracts different foreground objects for the left image when paired with a different image to the right.

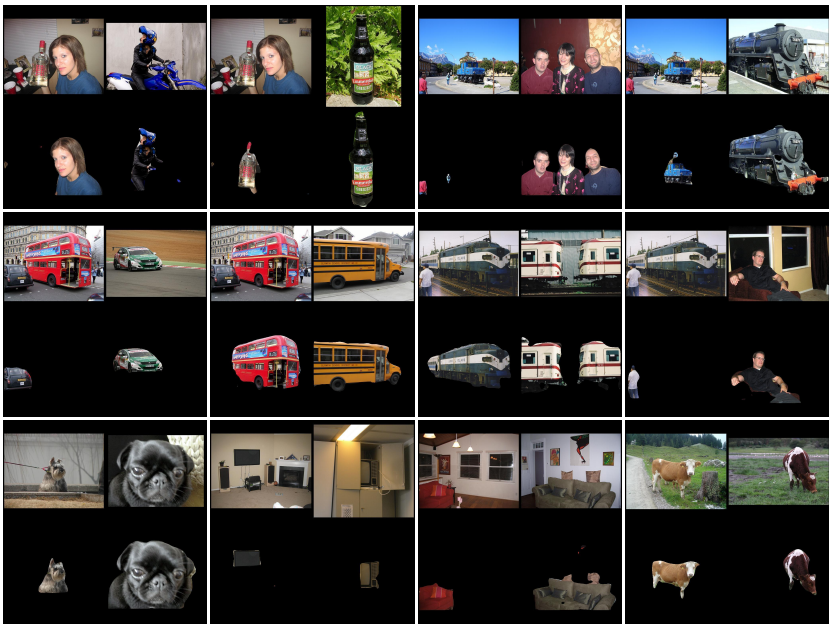


Figure 5.4: **Our qualitative results on PASCAL Co-segmentation dataset.** (odd rows) the input images, (even rows) the corresponding object co-segmentation results.

MSRC.

The MSRC subset has been used to evaluate the object co-segmentation performance by many previous methods [142, 141, 174, 178]. For the fair comparison, we use the same subset as [142]. We use our group co-segmentation method to extract the foreground masks for each group. In Table. 5.2, we show the quantitative results of our method as well as four state-of-the-art methods [139, 141, 174, 178]. Our *Precision* and *Jaccard* show a significant improvement compared to previous methods. It is important to note that [139] and [178] are supervised methods, i.e. both use images of the MSRC dataset to train their models. We obtain the new state-of-the-art results on this dataset even without training or fine-tuning on any images from the MSRC dataset. Visual examples of object co-segmentation results on the subset of the MSRC dataset can be found in Fig. 5.5.

Internet.

In our experiment, for the fair comparison, we followed [141, 180, 147, 157] to use the subset of the Internet dataset to evaluate our method. In this subset, there are 100 images in each category. We compare our method with five previous

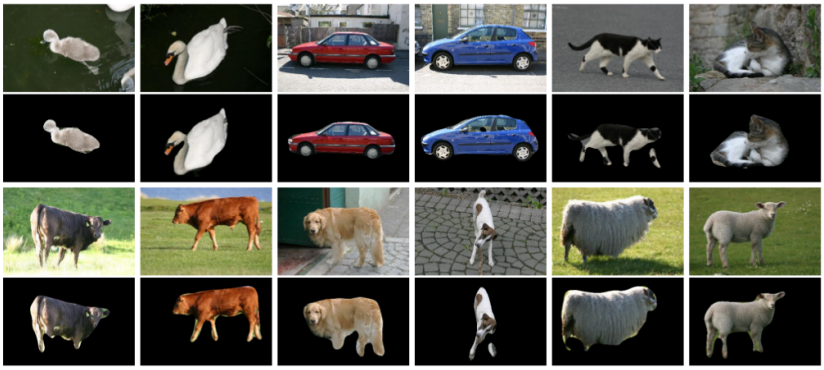


Figure 5.5: **Our qualitative results on the MSRC dataset (seen classes).** (odd rows) the input images, (even rows) the corresponding object co-segmentation results.

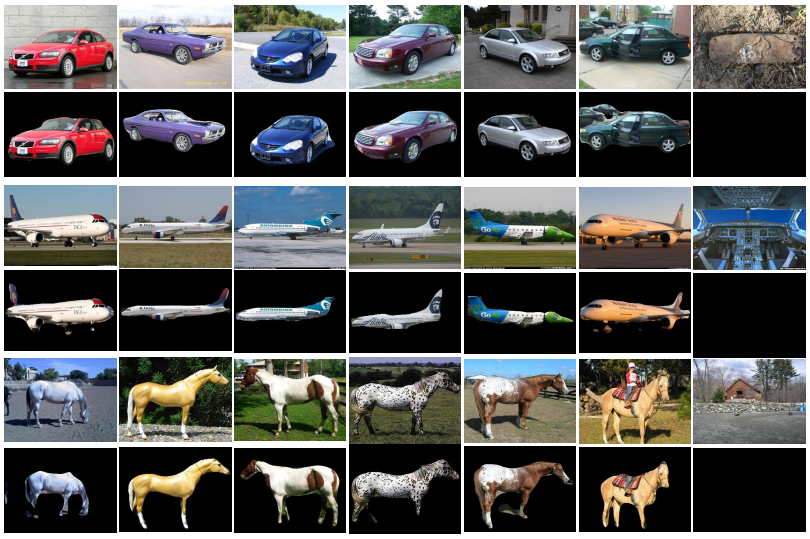


Figure 5.6: **Our qualitative results on the Internet dataset (seen classes)**. (odd rows) the input images, (even rows) the corresponding object co-segmentation results.

Table 5.2: **Quantitative results on the MSRC dataset (seen classes).** Quantitative comparison results of our DOCS approach with four state-of-the-art co-segmentation methods on the co-segmentation subset of the MSRC dataset.

MSRC	[139]	[141]	[178]	[174]	Ours
Precision	90.2	92.2	92.2	92.0	**95.4**
Jaccard	70.6	74.7	-	77.0	**82.9**

Table 5.3: **Quantitative results on the Internet dataset (seen classes).** Quantitative comparison of our DOCS approach with several state-of-the-art co-segmentation methods on the co-segmentation subset of the Internet dataset. 'P' is the *Precision*, and 'J' is the *Jaccard*.

Internet		[153]	[141]	[180]	[147]	[157]	Ours
Car	P	58.7	85.3	87.6	88.5	90.4	**93.9**
	J	37.1	64.4	64.9	66.8	72.0	**82.8**
Horse	P	63.8	82.8	86.2	89.3	90.2	**92.4**
	J	30.1	51.6	33.4	58.1	65.0	**69.4**
Airplane	P	49.2	88.0	90.3	92.6	91.0	**94.1**
	J	15.3	55.8	40.3	56.3	**66.0**	65.4
Average	P	57.2	85.4	88.0	89.6	91.1	**93.5**
	J	27.5	57.3	46.2	60.4	67.7	**72.6**

approaches [153, 180, 141, 147, 157]. Table 5.3 shows the quantitative results of each object category with respect to *Precision* and *Jaccard*. We outperform most of the previous methods [153, 180, 141, 147, 157] in terms of *Precision* and *Jaccard*. Note that [157] is a supervised co-segmentation method, [180] trained a discriminative Latent-SVM detector and [147] used a CNN trained on the ImageNet to extract semantic features. Fig. 5.6 shows some quantitative results of our method. It can be seen that even for the 'noise' images in each group, our method can successfully recognize them. We show the 'noise' images in the last column.

iCoseg

To show that our method can generalize on *unseen classes*, i.e. classes which are not part of the training data, we need to evaluate our method on *unseen classes*. Batra et al. [140] introduced the iCoseg dataset for the *interactive* co-segmentation task. In contrast to the MSRC and Internet datasets, there are multiple object classes in the iCoseg dataset which do not appear in PASCAL VOC dataset. Therefore, it is possible to use the iCoseg dataset to evaluate

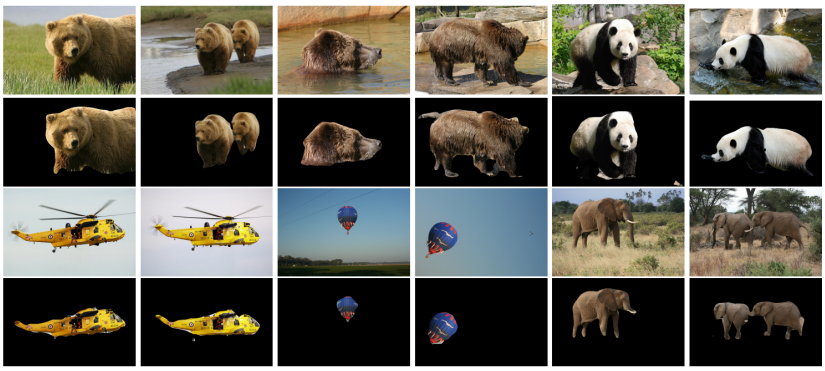


Figure 5.7: **Our qualitative results on iCoseg dataset (unseen classes).** Some results of our object co-segmentation method, with input image pairs in the odd rows and the corresponding object co-segmentation results in the even rows. For this dataset, the object classes were not known during training of our method (i.e. *unseen*).

Table 5.4: **Quantitative results on the iCoseg dataset (unseen classes).** Quantitative comparison of our DOCS approach with four state-of-the-art co-segmentation methods on some object classes of the iCoseg dataset, in terms of Jaccard. For this dataset, these object classes were not known during training of our method (i.e. *unseen*).

iCoseg	[141]	[181]	[174]	[179]	Ours
bear2	65.3	70.1	72.0	67.5	**88.7**
brownbear	73.6	66.2	**92.0**	72.5	91.5
cheetah	69.7	75.4	67.0	**78.0**	71.5
elephant	68.8	73.5	67.0	79.9	**85.1**
helicopter	80.3	76.6	**82.0**	80.0	73.1
hotballoon	65.7	76.3	88.0	80.2	**91.1**
panda1	75.9	80.6	70.0	72.2	**87.5**
panda2	62.5	71.8	55.0	61.4	**84.7**
average	70.2	73.8	78.2	74.0	**84.2**

the generalization of our method on *unseen object classes*. We choose eight groups of images from the iCoseg dataset as our unseen object classes, which are *bear2, brown_bear, cheetah, elephant, helicopter, hotballoon, panda1* and *panda2*. There are two reasons for this choice: firstly, these object classes are not included in the PASCAL VOC dataset. Secondly, in order to focus on *objects*, in contrast to *stuff*, we ignore groups like *pyramid, stonehenge* and *taj-mahal*. We compare our method with four state-of-the-art approaches [181, 141, 174, 179] on unseen

objects of the iCoseg dataset. Table 5.4 shows the comparison results of each unseen object groups in terms of *Jaccard*. The results show that for 5 out of 8 object groups our method performs best, and it is also superior on average. Note that the results of [181, 141, 174, 179] are taken from Table X in [179]. Fig. 5.7 shows some qualitative results of our method. It can be seen that our object co-segmentation method can detect and segment the common objects of these unseen classes accurately.

Furthermore to show the effect of number of PASCAL classes on the performance of our approach on unseen classes, we train our network on partial randomly picked PASCAL classes, i.e. $\{5, 10, 15\}$, and evaluate it on the iCoseg unseen classes. As it is shown in Table 5.5, our approach can generalize to unseen classes even when it is trained with only 10 classes from PASCAL.

Table 5.5: Analyzing the effect of number of training classes on unseen classes.

iCoseg	P(5)	P(10)	P(15)	P(20)
average	75.5	83.9	83.7	84.2

5.4.4 Ablation Study

To show the impact of the mutual correlation layer in our network architecture, we design a baseline network *DOCS-Concat* without using mutual correlation layers. In detail, we removed the correlation layer and we concatenate f_A and f_B (instead of C_{AB}) for image I_A and concatenate f_B and f_A (instead of C_{BA}) for image I_B. In Table 5.6, we compare the performance of different network designs on multiple datasets. As shown, the mutual correlation layer in *DOCS-Corr* improved the performance significantly.

5.5 Discussion

In this work, we presented a new and efficient CNN-based method for solving the problem of object class co-segmentation, which consists of jointly detecting and segmenting objects belonging to a common semantic class from a pair of images. Based on a simple encoder-decoder architecture, combined with the mutual correlation layer for matching semantic features, we achieve state-of-the-art

Table 5.6: Impact of mutual correlation layer.

	DOCS-Concat		DOCS-Corr	
	Precision	Jaccard	Precision	Jaccard
Pascal VOC	92.6	49.9	**94.2**	**64.5**
MSRC	92.6	72.0	**95.4**	**82.9**
Internet	91.8	62.7	**93.5**	**72.6**
iCoseg(unseen)	93.6	78.9	**95.1**	**84.2**

performance on various datasets, and demonstrate good generalization performance on segmenting objects of new semantic classes, unseen during training. To train our model, we compile a large object co-segmentation dataset consisting of image pairs from PASCAL dataset with shared objects masks.

Chapter 6

Split-Merge Pooling

6.1 Introduction

Convolutional neural networks (CNNs) are the method of choice for image classification [182, 183, 171, 184, 176]. CNNs capture multi-scale contextual information in an image via subsampling the intermediate features through the network layers [182, 185]. This approach is successful due to the expansion of the receptive fields, which are large enough to capture the context of the image for classification.

In this work, we consider dense prediction tasks, which are popular in computer vision [186]. One desideratum of dense prediction tasks is to have pixel-accurate predictions, for example in semantic segmentation [184, 28, 161, 144, 187] or depth estimation [30, 32]. Even small inaccuracies, such as missing a small object lying on the street, may lead to an accident of an autonomous vehicle.

Most state-of-the-art approaches for dense prediction adapt existing CNNs which were originally designed for image classification. However, these adapted CNNs lose a vast amount of spatial information due to subsampling. As a result this reduces the prediction performance, mostly because of missing small objects or predicting coarse and inaccurate object boundaries. In order to mitigate this problem, other methods are proposed, such as progressive upsampling [30, 32, 28], deconvolution (or transpose convolutions) [161], skip connections [144, 187, 169], utilizing multiple scales of features [1, 188], and attention mechanism [189]. Another line of work preserves the resolution by replacing subsampling layers with dilated convolution layers [190, 162, 191] and widely used in other approaches [188, 189, 192, 193]. In theory, it is possible to design a dilated convolution network without using any subsampling operation to obtain an output with the same size as the input. Unfortunately, for large input sizes the training and inference of the resulting CNN is extremely slow or even sometimes intractable due to limited amount of memory on a GPU, hence subsampling is still needed in practice.

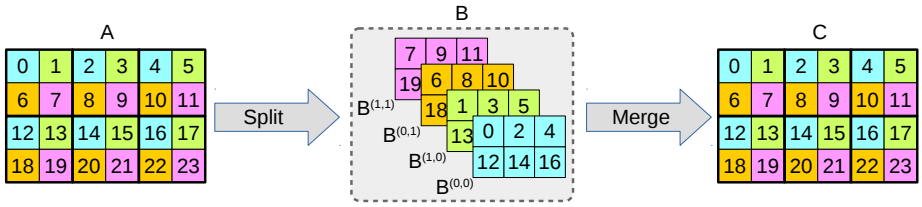


Figure 6.1: **Split and Merge Pooling.** The illustration of split and merge pooling layers with a window size of 2×2. The advantage of splitting the input into batches is to make it possible to process each part of the input (i.e. each split batch) independently. In this example, after splitting A into batches in B, we can send each batch $B^{(i,j)}$ to a different GPU and continue the forward pass from this point onward on multiple GPUs, or process one batch at a time on a single GPU. This enables us to execute dense prediction tasks with very deep networks and for large images, while always preserving the spatial information.

In this work, we propose novel pooling layers called Split-Merge Pooling (SMP). The split pooling layer splits (re-arranges) a feature tensor along its spatial dimension and treats the resulting splits as individual batches (see Fig. 6.1). The split pooling reduces the spatial size with a fixed scaling factor, related to the size of the non-overlapping pooling window. The merge layer acts as the reverse of split pooling operation, i.e., it receives the split batches and re-arranges the elements of the batches to their original locations. In our experiments, we replace all subsampling layers of standard ResNet networks by our split pooling. Finally, we merge the resulting split batches using merge pooling layers to output full resolution prediction. In this way, we do not lose any spatial information since it is preserved in the split batches. The batches are processed independently after the split, which has the major advantage that after the split each batch can be processed on a different GPU. This enables us to perform dense prediction tasks with very large networks on large images, while always preserving the spatial information. Furthermore, to increase training speed, we introduced the Shrink and Expand pooling layers as a batch-subsampling of the split and merge pooling.
In summary, our contributions are as follows:

- We propose a novel pooling method called Split-Merge Pooling (SMP) which enables the unique mapping of each input element to one output element, without losing any spatial information.
- We propose a batch-subsampling variant of SMP, termed Shrink-Expand Pooling, to make the training efficient and tractable for very deep networks.
- To show the effectiveness of SMP on dense prediction tasks, we choose semantic segmentation and apply SMP to ResNet networks with varying depths. We chose ResNet as our baseline since it is used as the backbone in state-of-the-art approaches. For semantic segmentation of large images

from Cityscapes [194] and GTA-5 [195] datasets, our SMP networks outperform the corresponding ResNet networks by a significant margin, with up to 6.8% in IoU score.
- We even observe that a SMP version of a shallow ResNet (ResNet18) outperforms the original ResNet101 by 2.8% in IoU score, although ResNet101 is 3.8 times deeper than ResNet18.

6.2 Related Work

Utilizing CNNs for image classification became very popular with the introduction of the Imagenet challenge [176] and after some popular network architectures such as Alexnet [183], VGG [171] and ResNet [184] emerged. Pooling layers in CNNs were originally introduced for image classification tasks on MNIST dataset [182]. The aim of pooling layers is to summarize the information over a spatial neighborhood.

Dense prediction tasks. Most of the dense prediction models are based on adapted versions of image classification networks. Eigen et al. [30, 32] adapt the Alexnet [183] for single image depth estimation. Since the output of such networks is very coarse, they upsample the output progressively and combine it with local features from the input image. Long et al. [28] introduce the first fully covolutional network (FCN) for semantic image segmentation by adapting VGG network [171] for this task. They map intermediate features with higher resolution to label space and combine them with the coarse prediction of the network in order to recover the missing spatial information. Noh et al. [161] introduce more parameters in a decoder consisting of transpose convolutions (Deconvolution layers) to upsample the coarse output of the encoder. However, it is still difficult to recover the missing information from the coarse output with this approach. Furthermore, other approaches [144, 187, 169] use skip connections between encoder and decoder to obtain more detailed information from lower level features.

Yu et al. introduced the concept of dilated convolutions [162] and later developed the dilated residual network (DRN) [191], which uses dilated convolutions to preserve the spatial information. However, they still use three subsampeling layers to reduce the spatial resolution to make training feasible, i.e. to fit the model into memory. Therefore, DRNs lose spatial information and need upsampling to obtain the full size output prediction. In contrast, our approach does not lose any spatial information. The batch-based design of our split pooling gives us the flexibility of distributed processing of batches on multiple GPUs or sequential processing of batches on one GPU during inference time (see Sec 6.3.1). Furthermore, for training, we learn our network weights using only one subsampled split batch to make the training faster and tractable (see Sec 6.3.2).

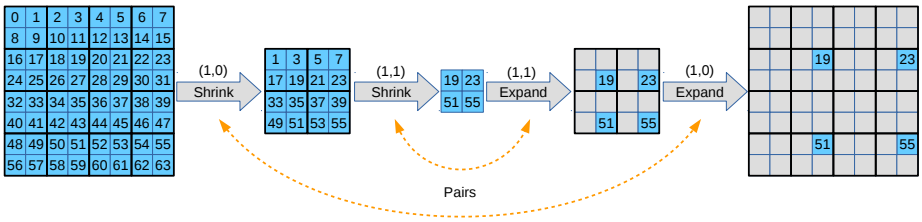


Figure 6.2: **Shrink and Expand Pooling example.** The illustration of two pairs of shrink and expand pooling layers with window size of 2×2 and sampling location of $(1, 0)$ and $(1, 1)$. During training the expand pooling layers back-propagate the error of sparse elements.

6.3 Method

The goal of this work is to design a set of pooling layers that preserve spatial information, in contrast to subsampling operations such as max-pooling and striding convolutions. Additionally, the new pooling layers have to be applicable to very deep networks. Losing spatial information caused by subsampling is a prevalent problem for dense prediction tasks, such as object detection, semantic segmentation, instance segmentation, or depth estimation. Small objects in the input image can get lost completely with subsampling. Moreover, recovering precise object boundaries in an image becomes challenging due to missing spatial information. These problems, caused by losing spatial information, obviously reduced prediction accuracy. However, simply storing all the information with the original spatial resolution is not a practical option since it causes storage issues, in particular for training on large image datasets.

Our idea is to preserve all the spatial information by splitting the feature tensor into multiple downsampled batches (see Fig. 6.1) instead of preserving it inside the original spatial resolution. In this way, we have both advantages of large receptive fields due to standard downsampling, as well as keeping all the available spatial information for obtaining precise dense predictions.

6.3.1 Split-Merge Pooling

The split pooling layer splits an input feature tensor along spatial dimensions and outputs each split as a batch. The number of output batches depends on the window size of split pooling, e.g. a split pooling with a window size of 2×2 splits the input into 4 batches (see Fig. 6.1). The purpose of the split pooling is to downsample the input while preserving the whole information. Hence, the pooling window covers each element of the input once, without any gap or overlap.

The merge pooling layer acts as the inverse of split pooling, i.e., it takes the split batches and merges them into one batch. For dense prediction tasks, we apply the same number of merge pooling layers on the final output of network as

the number of split pooling layers in a network. The result will be a one-to-one mapping between the input and output pixels of the network.

Formally, given an input $A_{W \times H}$, a split pooling with window size of $w \times h$ splits A into $w * h$ batches

$$\mathcal{B} = \{ B^{(k,l)} \mid 0 \leq k < w \wedge 0 \leq l < h \}.$$

The elements of A are assigned to batch $B^{(k,l)}$ as follows

$$b_{i,j}^{(k,l)} = a_{i*w+k,j*h+l} \tag{6.1}$$

for all $0 \leq i < W/w$ and $0 \leq j < H/h$. The spatial size of each batch will be $W/w \times H/h$. If the input to the split pooling consist of multiple batches, the split pooling splits each input batch separately and returns all resulting splits as a set of batches.

The merge pooling performs the inverse of Eq. 6.1, i.e., if $\mathcal{B}$ is the input and A is the output, the elements of $\mathcal{B}$ are assigned to A as follows

$$a_{i*w+k,j*h+l} = b_{i,j}^{(k,l)} \tag{6.2}$$

for all $0 \leq k < w$, $0 \leq l < h$, $0 \leq i < W/w$ and $0 \leq j < H/h$.

Although preserving the spatial information is beneficial, it increases the number of elements to store during the forward pass. The advantage of splitting the inputs into batches is that we can distribute the computation of batches on multiple GPUs or processing them sequentially on one GPU. This is ideal for inference. However, in contrast to inference, training phase additionally requires to compute and store the gradients. This makes the training of very deep networks intractable for large batches of inputs with high resolution images. We handle this issue by introducing Shrink-Expand pooling layers.

6.3.2 Shrink-Expand Pooling

The batch-based design of the split pooling layer makes the forward process of each part (split batch) of the input independent of the other parts (split batches). Furthermore, the one-to-one mapping between the elements of input and output gives a clear path between these elements in both forward and backward direction. Giving these two properties, it is possible to train a network using a subset of split batches produced by each split pooling layer. If we reduce the size of the batch subset to one, the space complexity will be the same as the space complexity of the max pooling and striding convolutions. Also, the training time complexity stays the same.

Giving this reasoning, we introduce the shrink and expand pooling layers, which are the batch-subsampled versions of the split and merge pooling layers. The shrink pooling samples one element at a fixed location within the pooling window, and the corresponding expand pooling uses the same fixed location to perform the reverse of the shrink pooling (see Fig. 6.2). Hence, the output of expand pooling is sparse. In other words, the shrink pooling is the same as split pooling except it samples only one of the split batches and returns it.

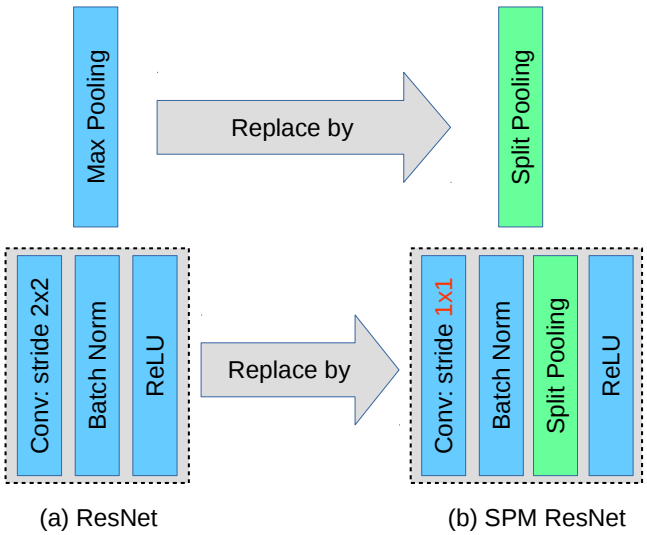


Figure 6.3: **Applying split pooling to ResNet.** For applying split pooling to ResNet, we replace the max pooling layer with split pooling (top). Furthermore, we add a split pooling layer after batch normalization layer of convolutional blocks with the stride of 2×2 and set the stride to 1×1 (bottom).

The sampling location (i, j) is set randomly in each forward pass during training to avoid overfitting to part of the training data. For each pair of shrink and expand pooling in the network, we sample only two numbers (i, j). Fig. 6.2 shows an example of using a sequence of shrink and expand pooling layers. During training, the expand layers only backpropagate the error of sparse valid elements.

6.4 Experiments

We evaluate the effectiveness of our approach for the semantic segmentation task. To examine the impact of our pooling layer, we modify the ResNet and then compare the performance of the modified version and the original one.

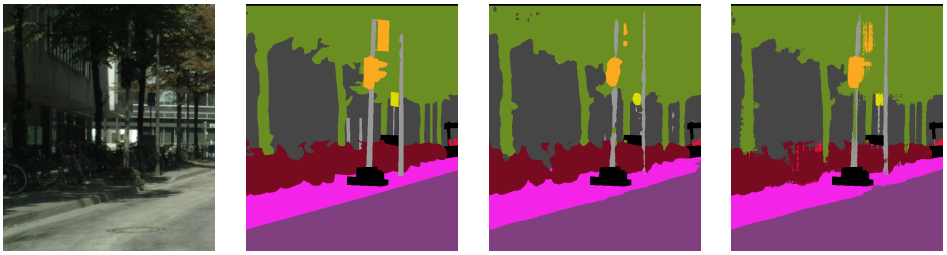
6.4.1 Experimental Setup

Baseline FCN32s. Our experimental models are based on FCN32s [28] with a variant of ResNet [184] as backbone. We adapt the ResNet for semantic segmentation task by removing the average pooling and fully connected layer and replacing them by a 1×1 convolutional layer which maps the output channels of last layer (layer4) to the number of semantic classes. We refer to this model as FCN32s and use it as baseline. FCN32s predictions are 32 times smaller than

the input image to the network; thus, the coarse predictions are upsampled to full resolution using bilinear interpolation.

SMP-{18, 34, 101}. As illustrated in Fig. 6.3, in order to apply SMP to ResNet backbone of FCN32s, we simply replace the max pooling layer with a split pooling layer and add a split pooling layer after the batch normalization layer of convolutional blocks (Conv-BN-ReLU) with a stride of 2×2 and set their strides to 1×1. In our experiments, the window size of split pooling layers is 2×2. We call the resulting model, according to the type of backbone ResNet, SMP-18, SMP-34 and SMP-101. The output of SMP-X consist of 1024 batches, since we always have 5 SMP layers each giving 4 batches. We merge the output split batches by performing merge pooling five times. The final result has the same resolution as the input. During training phase, we replace split and merge pooling layer by shrink and expand layers.

FCN8s. Furthermore, we compare our models to the FCN8s model with original ResNet backbone. The main difference between FCN32s and FCN8s is that FCN8s has two extra 1×1 convolution layers to map the features channel of layer2 and layer3 outputs of ResNet to the number of semantic classes. Then, the output of the network and these new convolutions are resized to the same size and are added together. The final output of FCN8s is 8 times smaller than the input image and should be upsampled to full resolution using bilinear interpolation.



(a) Image (b) Ground-truth (c) FCN8s (d) Ours

Figure 6.4: **An example of a slightly inaccurate annotation in Cityscapes dataset.** Although the trunk of the tree in the image (a) is visible through bicycles, it is labeled as bicycle in ground truth (b). Our method can obtain detailed boundaries (d) while FCN8s with max pooling cannot (c). Hence, to fully validate the full potential of our method we need a pixel-accurate ground-truth, such as GTA-5.

6.4.2 Implementation Details

Data Augmentation. For all the models we used random crop of size 512×512 and horizontal flip data augmentation. We used a fixed seed for data aug-

mentation for all the experiments.

Training. We initialize the ResNet backbones with pretrained models from Imagenet. We optimize the parameters using Adam solver [135] with learning rate of $1e-5$ and weight decay of $5e-4$. We use the batch size of 10 for all the experiments.

Table 6.1: **Cityscapes quantitative results.** Please note that the architecture of SMP-X networks is the same as the FCN32s. The FCN8s architecture has two extra 1×1 convolutions.

	Backbone	Pooling	IoU	Size
FCN32s	ResNet34	MP	64.5	21.29M
FCN8s	ResNet34	MP	67.8	21.30M
SMP-34 (ours)	ResNet34	SMP	**68.8**	21.29M
FCN32s	ResNet101	MP	65.5	42.54M
FCN8s	ResNet101	MP	69.1	42.56M
SMP-101 (ours)	ResNet101	SMP	**69.2**	42.54M

Table 6.2: **Quantitative results for GTA-5.** Please note that the architecture of SMP-X networks is the same as the FCN32s, while the FCN8s has two extra 1×1 convolutions.

	Backbone	Pooling	IoU	Size
FCN32s	ResNet18	MP	71.1	11.186M
FCN8s	ResNet18	MP	74.2	11.193M
SMP-18 (ours)	ResNet18	SMP	**77.5**	11.186M
FCN32s	ResNet34	MP	73.1	21.29M
FCN8s	ResNet34	MP	76.7	21.30M
SMP-34 (ours)	ResNet34	SMP	**80.2**	21.29M
FCN32s	ResNet101	MP	74.6	42.54M
FCN8s	ResNet101	MP	76.7	42.56M
SMP-101 (ours)	ResNet101	SMP	**80.3**	42.54M

6.4.3 Cityscapes

The Cityscapes dataset [194] consists of images with the size of 2048×1024. The images are annotated with 19 semantic classes. We evaluate the performance of our SMP-34 and SMP-101 on Cityascapes validation set and compare it with FCN32s and FCN8s with ResNet34 and ResNet101 backbone networks. The full size images are used for evaluation, i.e. without downsampling or cropping.

Table 6.1 summarizes the comparison of different methods with respect to their setups, performances and number of parameters (size). Each SMP model

Table 6.3: **Performance on Cityscapes for small and thin objects.**

	pole	traffic light	traffic sign	person	rider	motorcycle	bicycle	IoU
FCN32s-Res34	35.4	51.5	63.0	70.4	50.3	51.8	68.4	55.8
FCN8s-Res34	53.5	57.6	69.9	77.1	53.6	51.0	72.2	62.1
SMP-34(ours)	**60.5**	**63.7**	**74.5**	**78.8**	**54.1**	**52.9**	**73.9**	**65.5**
FCN32s-Res101	39.2	58.1	66.9	71.9	51.4	53.9	70.6	58.9
FCN8s-Res101	56.1	63.1	72.2	78.2	55.4	52.8	74.7	64.6
SMP-101(ours)	**63.3**	**68.7**	**75.8**	**79.9**	**56.1**	**52.9**	**76.6**	**67.6**

Table 6.4: **Performance on GTA-5 for small and thin objects**

	pole	traffic light	traffic sign	person	rider	motorcycle	bicycle	IoU
FCN32s-Res18	44.6	45.3	60.7	69.3	64.5	59.8	38.5	54.7
FCN8s-Res18	54.0	52.7	63.8	74.4	**70.4**	66.5	40.3	60.3
SMP-18(ours)	**72.7**	**69.2**	**73.6**	**76.8**	67.8	**69.3**	**46.5**	**68.0**
FCN32s-Res34	46.0	47.7	63.0	70.0	67.9	63.9	49.5	58.3
FCN8s-Res34	55.6	57.2	68.3	76.3	**74.0**	65.8	51.3	64.1
SMP-34(ours)	**74.3**	**71.4**	**73.7**	**81.0**	70.5	**73.7**	**58.5**	**71.9**
FCN32s-Res101	47.3	50.3	68.7	70.3	63.9	66.2	58.1	60.7
FCN8s-Res101	57.1	59.0	70.8	75.5	67.1	**70.6**	62.4	66.1
SMP-101(ours)	**76.8**	**74.9**	**78.9**	**79.0**	**69.2**	69.6	**67.8**	**73.7**

outperforms the corresponding FCN32s and FCN8s with the same ResNet backbone significantly. SMP-34 even outperforms FCN32s-ResNet101 by 3.3%, although the latter is almost 2 times larger. The performance of our approach on objects from small and thin classes is reported in Table 6.6. As we can see in the table, SMP models outperforms their corresponding original models on small and thin objects. Both SMP models outperform their corresponding FCN32s models for *pole* by 24%, for *traffic light* by more than 10%, for *traffic sign* by more than 9%, and for *person* by 8%. As it is shown in Fig. 6.5, the improvement of these classes can be noticed visually as well. The performance for the remaining objects is mostly better, or sometimes slightly worse.

Qualitative results are shown in Fig. 6.5. For the sake of improved visibility we have cropped the results. The full size output images can be found in supplementary material.

6.4.4 GTA-5

Cityscapes is one of the most accurately annotated semantic segmentation datasets, however it is still not pixel-accurate (see Fig. 6.4). Obtaining pixel-accurate annotations from real data is extremely challenging and expensive. Therefore, for analysing the full potential of our method, we evaluate our method on GTA-5 [195], which is a synthetic dataset with the same semantic classes as Cityscapes. Since GTA-5 is a synthetic, the annotations are pixel-accurate and ideal for our purpose. The GTA-5 dataset [195] consist of 24,999 realistic synthetic images with pixel-accurate semantic annotations. We randomly select 500 images as validation set, which we did not use for training.

Table 6.1 summarizes the comparison of different methods with respect to their setups, performances and number of parameters (size). As we can see, due to the pixel-accurate annotations of GTA-5 dataset, the improvement of our proposed models, over their baselines, is more significant compared to Cityscapes. Each SMP model outperforms the corresponding FCN32s and FCN8s with the same ResNet backbone significantly. Particularly, our SMP-18 even outperforms its FCN32s-ResNet101 and FCN8s-ResNet101 counterparts, although it has 4 times fewer parameters. The performance of our approach on objects from small and thin classes is reported in Table 6.7. As we can see, similarly to Cityscapes, SMP models outperforms their corresponding original models on small and thin objects. Compared to FCN32s models, our corresponding SMP models improve the categories *pole* by more than 28%, *traffic light* by more than 23%, *traffic sign* by more than 10%, and *person* by more than 7%. The improvement over these classes is also visually significant (see Fig. 6.6).

6.4.5 Run-time Analysis

For analyzing the time complexity of the Split-Merge pooling, we designed small networks to just focus on the proposed pooling layers instead of analyzing the time complexity of them on a particular task with specific network architecture. As it is shown in Fig. 6.7, we consider three networks with (a) max pooling, (b) dilated convolution, and (c) Split pooling. We choose to compare our proposed pooling setup (c) with dilated convolutions (b) due to the success of the dilated convolution in dense prediction tasks. Almost all state-of-the-art approaches in dense prediction tasks (such as semantic segmentation, depth estimation, optical flow estimation) are using dilated convolutions in their architectures to achieve a detailed output.

In Table 6.5, we show the Giga floating-point operation (GFLOP) of each component of each setup for an input tensor of size $1 \times 3 \times 256 \times 256$. As we can see, the dilated convolution setup and split pooling setup have the same GFLOPs which means dilated convolution layers can be replaced with our proposed pooling layers in an arbitrary architecture without changing the complexity of the network. However, our split pooling layer has two advantages:

1. faster training time using shrink-expand layers
2. faster inference time by parallelizing the forward-computation of split

Figure 6.5: **Cityscapes qualitative results.** In each block, the top row is related to models with ResNet34 backbone and the bottom row to ResNet101. The last column of each block shows the input image (top) and the ground-truth (bottom). To enhance the visual comparison of the results, we have cropped the output labelling. Further results, also in full resolution, can be found in supplementary material.

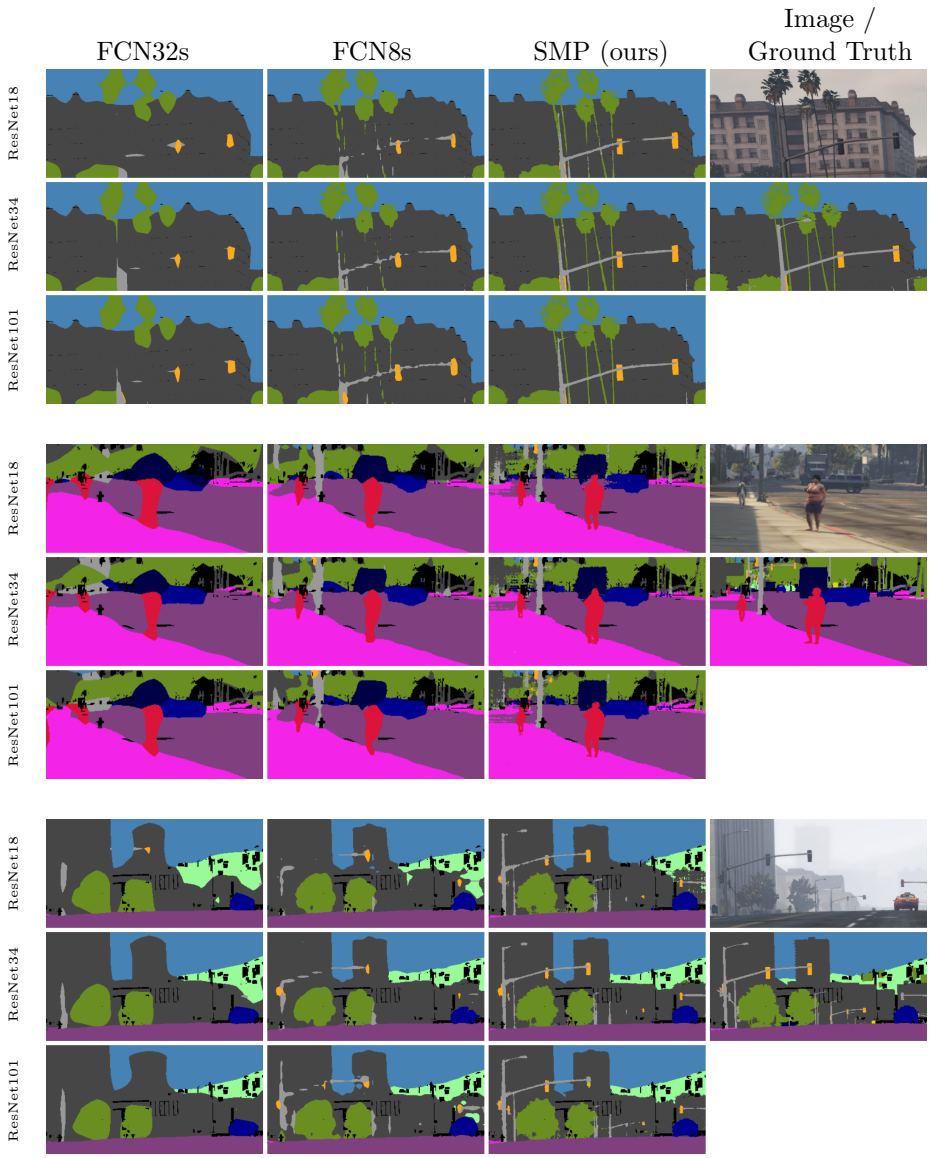


Figure 6.6: **GTA-5 qualitative results.** In each block, the top row is related to ResNet18 backbone, middle row to ResNet34, and bottom row to ResNet101. The last column of each block shows the input image (top) and the ground-truth (bottom). To enhance the visual comparison of the results, we have cropped the output labelling. Further results, also in full resolution, can be found in supplementary material.

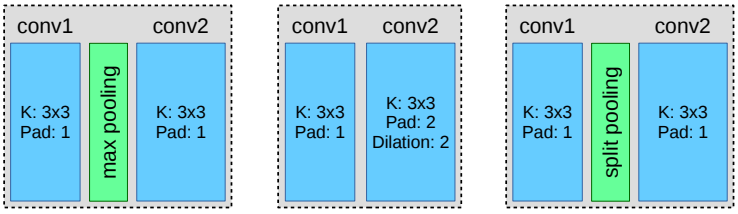


Figure 6.7: Architectures used for runtime analysis. *conv2* in (c) is identical to *conv2* in (a) while *conv2* in (c) is dilated convolution with padding 2.

layer output batches (in this example setup computation of *conv2*, see Table 6.5)

Table 6.5: GFLOPs of the models calculated on the input size of $1\times3\times256\times256$. Note that the number of batches are increased after split pooling.

	batches	conv1	pooling	batches	conv2	total
(a) Max Pooling	1	0.23	0	1	2.42	2.65
(b) Dilated Conv.	1	0.23	-	1	9.68	9.92
(c) Split Pooling	1	0.23	0	4	4x2.42 (9.68)	9.92

6.4.6 Detailed Quantitative Results

Table 6.6 and Table 6.7 show the detailed quantitative results of our proposed models on Cityscapes and GTA-5 datasets respectively.

6.5 Conclusion

We proposed a novel pooling method SMP with the goal of preserving the spatial information throughout the entire network. SMP can be used instead of any subsampling operations in a network architecture. We show that by replacing subsampling operations with SMP in ResNet, we achieved two important properties for any dense prediction task at the same time: i) the network has a large receptive field, ii) the network provides a unique mapping from input pixels to output pixels. Furthermore, the computation of a network with SMP can be distributed to multiple GPUs due to batch-based design of SMP. We show experimentally that the resulting network outperforms the original one significantly.

Table 6.6: **Cityscapes - detailed.** SMP models outperforms their corresponding original models on small and thin objects. Both SMP models outperform their corresponding FCN32s models for *pole* by 24%, for *traffic light* by more than 10%, for *traffic sign* by more than 10%, and for *person* by 8%.

	road	sidewalk	building	wall	fence	pole	traffic light	traffic sign	vegetation	terrain	sky	person	rider	car	truck	bus	train	motorcycle	bicycle	IoU
FCN32s-Res34	97.2	78.5	88.5	40.1	48.5	35.4	51.5	63.0	88.9	56.5	89.3	70.4	50.3	91.2	48.3	64.9	43.5	51.8	68.4	64.5
FCN8s-Res34	97.3	80.0	89.6	37.2	49.5	53.5	57.6	69.9	90.9	57.7	92.4	77.1	53.6	92.5	**49.9**	69.4	**46.6**	51.0	72.2	67.8
SMP-34(ours)	**97.3**	**80.6**	**90.5**	**42.2**	**50.6**	**60.5**	**63.7**	**74.5**	**91.4**	**58.7**	**92.8**	**78.8**	**54.1**	**92.9**	48.5	**70.6**	32.3	**52.9**	**73.9**	**68.8**
FCN32s-Res101	97.3	79.6	88.9	38.3	51.3	39.2	58.1	66.9	89.4	55.4	91.3	71.9	51.4	91.8	43.0	65.1	41.3	53.9	70.6	65.5
FCN8s-Res101	97.5	81.2	90.3	**41.0**	**49.9**	56.1	63.1	72.2	91.4	59.8	92.8	78.2	55.4	92.9	**49.9**	**68.6**	**44.7**	52.8	74.7	69.1
SMP-101(ours)	**97.5**	**82.7**	**90.6**	39.6	48.4	**63.3**	**68.7**	**75.8**	**91.9**	**61.0**	**93.4**	**79.9**	**56.1**	**93.2**	41.5	61.2	40.0	**52.9**	**76.6**	**69.2**

Table 6.7: **GTA-5 - detailed.** SMP models outperforms their corresponding original models on small and thin objects for GTA-5 as well. Compare to FCN32s models, our corresponding SMP models improve the categories *pole* by more than 28%, *traffic light* by more than 23%, *traffic sign* by more than 10%, and *person* by more than 7%.

	road	sidewalk	building	wall	fence	pole	traffic light	traffic sign	vegetation	terrain	sky	person	rider	car	truck	bus	train	motorcycle	bicycle	IoU
FCN32s-Res18	95.4	81.8	87.3	62.9	54.8	44.6	45.3	60.7	79.9	70.0	93.3	69.3	64.5	88.3	**83.9**	87.8	82.6	59.8	38.5	71.1
FCN8s-Res18	96.1	84.6	88.6	66.6	**56.4**	54.0	52.7	63.8	83.3	72.5	94.8	74.4	**70.4**	89.5	80.7	**90.4**	85.1	66.5	40.3	74.2
SMP-18(ours)	**96.5**	**85.6**	**89.6**	**67.2**	55.7	**72.7**	**69.2**	**73.6**	**88.4**	**75.0**	**97.9**	**76.8**	67.8	**90.6**	82.3	80.8	**86.1**	**69.3**	**46.5**	**77.5**
FCN32s-Res34	96.5	85.2	88.0	64.2	55.4	46.0	47.7	63.0	80.9	72.4	93.5	70.0	67.9	89.2	86.2	85.1	84.2	63.9	49.5	73.1
FCN8s-Res34	97.0	87.6	89.5	68.9	**59.4**	55.6	57.2	68.3	83.9	74.4	94.9	76.3	**74.0**	90.5	86.8	**90.4**	84.9	65.8	51.3	76.7
SMP-34(ours)	**97.3**	**88.1**	**91.4**	**69.6**	58.3	**74.3**	**71.4**	**73.7**	**89.1**	**77.1**	**98.2**	**81.0**	70.5	**92.6**	**88.3**	83.7	**87.6**	**73.7**	**58.5**	**80.2**
FCN32s-Res101	96.6	86.0	89.2	70.4	59.3	47.3	50.3	68.7	81.6	72.6	93.7	70.3	63.9	89.6	88.2	**89.1**	77.1	66.2	58.1	74.6
FCN8s-Res101	96.6	86.6	89.2	61.7	60.6	57.1	59.0	70.8	84.5	73.9	95.1	75.5	67.1	90.0	**88.6**	86.0	81.9	**70.6**	62.4	76.7
SMP-101(ours)	**97.3**	**88.7**	**91.7**	**71.0**	**62.2**	**76.8**	**74.9**	**78.9**	**89.6**	**79.0**	**97.8**	**79.0**	**69.2**	**90.6**	86.0	71.9	**84.3**	69.6	**67.8**	**80.3**

Chapter 7

Conclusion

In this thesis, we addressed several subtasks of scene understanding as follows:

- We designed a modular CNN for training and refining tasks of semantic segmentation and depth estimation jointly. We introduced a setup for analyzing the cross modality effect between these two tasks, and using this analysis we found a desirable architecture design that improved both tasks and achieved state of the art performance at the time of publication.
- We utilized the object detection and instance segmentation tasks for improving the scene flow estimation and object 6D pose estimation tasks by focusing on target objects.
- We introduced the first novel deep learning approach for object co-segmentation and achieved state of the art performance.
- We introduced a new pooling layer Split-Merge pooling for preserving the spatial information while increasing the receptive field. This pooling layer can work with normal convolution layers, and its batch-based design makes it possible to parallelize the computation of batches on multiple GPUs.

One of the main focuses of this thesis was to study the interaction between different subtasks of scene understanding. Jointly training multiple tasks with the same final goal of understanding the scene is crucial and is actively being studied recently [186, 196]. As we discussed in this thesis, it can be done either by restricting the search-space of one task using the outcome of another one or by sharing part of the learning parameters between multiple tasks to achieve a consistent and efficient system.

7.1 Future work

In this section, we discuss some possible directions for further improving the following works from this thesis:

Multi task learning: In chapter 2, we designed a modular network for improving two tasks of semantic segmentation and depth estimation. We analyze different numbers of channels and different operations to fuse the features. One possible direction for further analysis is to perform the fusion in different depths of the networks. Another interesting future work is to consider more tasks; this is especially interesting in applications like autonomous driving.

Instance-aware scene flow estimation: In chapter 3, we introduced a pipeline for estimating scene flow. For performing object detection and instance segmentation, we used MNC, which was the best instance segmentation approach at the time. One direction to further improve our proposed method is to use MaskRCNN, which is a better approach for instance segmentation.

iPose: In chapter 4, we proposed an instance-aware object 6D pose estimation method for partially occluded objects. In this approach, we generated synthetic training images from real samples. One way to improve the performance of our approach is to generate a dataset by rendering the objects in a synthetic scene. Afterwards, for removing the domain gap between real images during test and training images, we can apply domain adaptation techniques. This could help significantly to improve the quality of object coordinate predictions.

Another direction for improving the object coordinate estimation is to redesign its encoder-decoder network by applying the Split-Merge pooling layers (i.e. introduced in this thesis in chapter 6). In this case there is no need to have a decoder since the output of the encoder has the same resolution as the input. This can improve the object coordinate predictions by keeping all the details on the surface of objects.

Object co-segmentation: In chapter 5, we introduced a novel deep learning approach to address the problem of object co-segmentation. This task segments all instances of common objects as foreground. Therefore, it is not possible to separate different instances of common classes. One future work direction for this work can be to extend it to an object instance co-segmentation task. As another direction, it would be interesting to train the model in an unsupervised manner. To this end, one may use an unsupervised clustering approach instead of a cross-entropy loss. In this case, there is no need to use any binary mask annotation for training categories.

Split-Merge pooling: In chapter 6, we presented the Split-Merge pooling layer that preserves the spatial information and improves the performance of dense prediction tasks. We applied it to basic Resnet networks for semantic segmentation task. One future work could be to apply this pooling layer on more advanced network architectures and see the impact of this pooling layer with respect to accuracy and time complexity. Another suggestion for future work is to apply it to other dense tasks such as depth estimation or instance segmentation.

Finally, the Split-Merge pooling layer is completely invertible, i.e. merge layer is the invert operation for the split pooling layer. Therefore, this pooling layer can be used in invertible networks and it is another interesting direction for future work.

Bibliography

[1] O. Hosseini Jafari, O. Groth, A. Kirillov, M. Ying Yang, and C. Rother, "Analyzing modular cnn architectures for joint depth prediction and semantic segmentation," in *ICRA*, 2017.

[2] A. Behl, O. Hosseini Jafari, S. K. Mustikovela, H. Abu Alhaija, C. Rother, and A. Geiger, "Bounding boxes, segmentations and object coordinates: How important is recognition for 3d scene flow estimation in autonomous driving scenarios?" in *ICCV*, 2017.

[3] O. Hosseini Jafari, S. K. Mustikovela, K. Pertsch, E. Brachmann, and C. Rother, "ipose: instance-aware 6d pose estimation of partly occluded objects," in *ACCV*, 2018.

[4] W. Li, O. Hosseini Jafari, and C. Rother, "Deep object co-segmentation," in *ACCV*, 2018.

[5] O. H. Jafari and C. Rother, "Split-merge pooling," 2020.

[6] O. Hosseini Jafari and M. Ying Yang, "Real-time rgb-d based template matching pedestrian detection," in *ICRA*, 2016.

[7] W. Li, O. Hosseini Jafari, and C. Rother, "Semantic-aware image smoothing," in *VMV*, 2017.

[8] W. Li, O. Hosseini Jafari, and C. Rother, "Localizing common objects using common component activation map," in *CVPR Workshops*, 2019.

[9] C. Li, A. Kowdle, A. Saxena, and T. Chen, "Toward holistic scene understanding: Feedback enabled cascaded classification models," *PAMI*, vol. 34, no. 7, pp. 1394–1408, 2012.

[10] V. Vineet, C. Rother, and P. Torr, "Higher order priors for joint intrinsic image, objects, and attributes estimation," in *NIPS*, 2013, pp. 557–565.

[11] J. T. Barron and J. Malik, "Shape, illumination, and reflectance from shading," *PAMI*, 2015.

[12] D. Marr, *Vision*. Freeman, 1982.

[13] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, "Indoor segmentation and support inference from rgbd images," in *ECCV*, 2012.

[14] A. Saxena, A. Ng, and S. Chung, "Learning Depth from Single Monocular Images," in *NIPS*, 2005.

[15] A. Saxena, M. Sun, and A. Y. Ng, "Make3d: Learning 3d scene structure from a single still image," *PAMI*, vol. 31, no. 5, pp. 824–840, 2009.

[16] B. Liu, S. Gould, and D. Koller, "Single image depth estimation from predicted semantic labels," in *CVPR*, 2010, pp. 1253–1260.

[17] M. Liu, M. Salzmann, and X. He, "Discrete-continuous depth estimation from a single image," in *CVPR*, 2014, pp. 716–723.

[18] C. Hane, L. Ladicky, and M. Pollefeys, "Direction matters: Depth estimation with a surface normal classifier," in *CVPR*, 2015.

[19] W. Zhuo, M. Salzmann, X. He, and M. Liu, "Indoor scene structure analysis for single image depth estimation," in *CVPR*, 2015.

[20] J. Shotton, J. Winn, C. Rother, and A. Criminisi, "Textonboost for image understanding: Multi-class object recognition and segmentation by jointly modeling texture, layout, and context," *IJCV*, vol. 81, no. 1, pp. 2–23, 2009.

[21] L. Ladicky, C. Russell, P. Kohli, and P. H. S. Torr, "Associative hierarchical random fields," *PAMI*, vol. 36, no. 6, pp. 1056–1077, 2014.

[22] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *NIPS*, 2012, pp. 1097–1105.

[23] N. Zhang, J. Donahue, R. B. Girshick, and T. Darrell, "Part-based r-cnns for fine-grained category detection," in *ECCV*, 2014, pp. 834–849.

[24] S. Gupta, R. Girshick, P. Arbelaez, and J. Malik, "Learning rich features from RGB-D images for object detection and segmentation," in *ECCV*, 2014, pp. 345–360.

[25] P. Agrawal, R. B. Girshick, and J. Malik, "Analyzing the performance of multilayer neural networks for object recognition," in *ECCV*, 2014, pp. 329–344.

[26] M. Oquab, L. Bottou, I. Laptev, and J. Sivic, "Learning and transferring mid-level image representations using convolutional neural networks," in *CVPR*, 2014, pp. 1717–1724.

[27] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *CVPR*, 2014.

[28] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *CVPR*, 2015.

[29] A. Toshev and C. Szegedy, "Deeppose: Human pose estimation via deep neural networks," in *CVPR*, 2014, pp. 1653–1660.

[30] D. Eigen, C. Puhrsch, and R. Fergus, "Depth map prediction from a single image using a multi-scale deep network," in *NIPS*, 2014, pp. 2366–2374.

[31] F. Liu, C. Shen, G. Lin, and I. Reid, "Learning depth from single monocular images using deep convolutional neural fields," *PAMI*, 2016.

[32] D. Eigen and R. Fergus, "Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture," in *ICCV*, 2015.

[33] P. Pinheiro and R. Collobert, "Recurrent convolutional neural networks for scene labeling," in *ICML*, 2014.

[34] M. Bleyer, C. Rother, P. Kohli, D. Scharstein, and S. Sinha, "Object stereo - joint stereo matching and object segmentation," in *CVPR*, 2011, pp. 3081–3088.

[35] J.-Y. Guillemaut and A. Hilton, "Joint multi-layer segmentation and reconstruction for free-viewpoint video applications," *IJCV*, vol. 93, no. 1, pp. 73–100, 2011.

[36] L. Ladicky, P. Sturgess, C. Russell, S. Sengupta, Y. Bastanlar, W. Clocksin, and P. Torr, "Joint optimization for object class segmentation and dense stereo reconstruction," *IJCV*, vol. 100, no. 2, pp. 122–133, 2012.

[37] J. Yao, S. Fidler, and R. Urtasun, "Describing the scene as a whole: Joint object detection, scene classification and semantic segmentation," in *CVPR*, 2012, pp. 702–709.

[38] Z. Zhang, A. G. Schwing, S. Fidler, and R. Urtasun, "Monocular object instance segmentation and depth ordering with cnns," in *ICCV*, 2015.

[39] L. Ladicky, J. Shi, and M. Pollefeys, "Pulling things out of perspective," in *CVPR*, 2014.

[40] P. Wang, X. Shen, Z. Lin, S. Cohen, B. Price, and A. L. Yuille, "Towards unified depth and semantic prediction from a single image," in *CVPR*, 2015.

[41] J. Ngiam, A. Khosla, M. Kim, J. Nam, H. Lee, and A. Y. Ng, "Multimodal deep learning," in *ICML*, 2011.

[42] N. Srivastava and R. R. Salakhutdinov, "Multimodal learning with deep boltzmann machines," in *NIPS*, 2012, pp. 2222–2230.

[43] S. Chandar, M. M. Khapra, H. Larochelle, and B. Ravindran, "Correlational neural networks," *Neural Computation*, 2015.

[44] Z. Obrenovic and D. Starcevic, "Modeling multimodal human-computer interaction," *Computer*, vol. 37, no. 9, pp. 65–72, 2004.

[45] A. Jaimes and N. Sebe, "Multimodal human-computer interaction: A survey," *CVIU*, vol. 108, no. 1-2, pp. 116–134, 2007.

[46] K. von Kriegstein, D. Ozgr, M. Grter, A.-L. Giraud, C. A. Kell, T. Grter, A. Kleinschmidt, and S. J. Kiebel, "Simulation of talking faces in the human brain improves auditory speech recognition," *PNAS*, vol. 105, no. 18, pp. 6747–6752, 2008.

[47] S. Schall, S. J. Kiebel, B. Maess, and K. von Kriegstein, "Early auditory sensory processing of voices is facilitated by visual mechanisms," *NeuroImage*, vol. 77, pp. 237 – 245, 2013.

[48] H. Hotelling, "Relations between two sets of variates," *Biometrika*, vol. 28, pp. 321–377, 1936.

[49] G. Andrew, R. Arora, J. Bilmes, and K. Livescu, "Deep canonical correlation analysis," in *ICML*, 2013, pp. 1247–1255.

[50] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," *arXiv preprint arXiv:1408.5093*, 2014.

[51] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," 2012.

[52] M. Menze and A. Geiger, "Object scene flow for autonomous vehicles," 2015.

[53] C. Vogel, K. Schindler, and S. Roth, "3d scene flow estimation with a piecewise rigid scene model," vol. 115, no. 1, pp. 1–28, 2015.

[54] J. Dai, K. He, and J. Sun, "Instance-aware semantic segmentation via multi-task network cascades," 2016.

[55] E. Brachmann, A. Krull, F. Michel, S. Gumhold, J. Shotton, and C. Rother, "Learning 6d object pose estimation using 3d object coordinates," 2014.

[56] E. Brachmann, F. Michel, A. Krull, M. Y. Yang, S. Gumhold, and C. Rother, "Uncertainty-driven 6d pose estimation of objects and scenes from a single rgb image," 2016.

[57] J. P. C. Valentin, M. Nießner, J. Shotton, A. W. Fitzgibbon, S. Izadi, and P. H. S. Torr, "Exploiting uncertainty in regression forests for accurate camera relocalization," 2015.

[58] F. Michel, A. Krull, E. Brachmann, M. Y. Yang, S. Gumhold, and C. Rother, "Pose estimation of kinematic chain instances via object coordinate regression," 2015.

[59] J. Taylor, J. Shotton, T. Sharp, and A. W. Fitzgibbon, "The vitruvian manifold: Inferring dense correspondences for one-shot human pose estimation," 2012.

[60] U. Franke, C. Rabe, H. Badino, and S. Gehrig, "6D-Vision: fusion of stereo and motion for robust environment perception," 2005.

[61] A. Wedel, C. Rabe, T. Vaudrey, T. Brox, U. Franke, and D. Cremers, "Efficient dense scene flow from sparse or dense stereo data," 2008.

[62] E. Herbst, X. Ren, and D. Fox, "RGB-D flow: Dense 3D motion estimation using color and depth." 2013.

[63] M. Hornacek, A. Fitzgibbon, and C. Rother, "SphereFlow: 6 DoF scene flow from RGB-D pairs," 2014.

[64] J. Quiroga, T. Brox, F. Devernay, and J. L. Crowley, "Dense semi-rigid scene flow estimation from RGB-D images," 2014.

[65] S. Vedula, S. Baker, P. Rander, R. Collins, and T. Kanade, "Three-dimensional scene flow," 1999.

[66] S. Vedula, P. Rander, R. Collins, and T. Kanade, "Three-dimensional scene flow," vol. 27, no. 3, pp. 475–480, 2005.

[67] T. Basha, Y. Moses, and N. Kiryati, "Multi-view scene flow estimation: A view centered variational approach," vol. 101, no. 1, pp. 6–21, 2013.

[68] F. Huguet and F. Devernay, "A variational method for scene flow estimation from stereo sequences," 2007.

[69] J.-P. Pons, R. Keriven, and O. Faugeras, "Multi-view stereo reconstruction and scene flow estimation with a global image-based matching score," vol. 72, no. 2, pp. 179–193, 2007.

[70] L. Valgaerts, A. Bruhn, H. Zimmer, J. Weickert, C. Stoll, and C. Theobalt, "Joint estimation of motion, structure and geometry from stereo sequences," 2010.

[71] A. Wedel, T. Brox, T. Vaudrey, C. Rabe, U. Franke, and D. Cremers, "Stereoscopic scene flow computation for 3D motion understanding," vol. 95, no. 1, pp. 29–51, 2011.

[72] C. Vogel, K. Schindler, and S. Roth, "3D scene flow estimation with a rigid motion prior," 2011.

[73] K. Yamaguchi, D. McAllester, and R. Urtasun, "Robust monocular epipolar flow estimation," 2013.

[74] K. Yamaguchi, D. McAllester, and R. Urtasun, "Efficient joint segmentation, occlusion labeling, stereo and flow estimation," 2014.

[75] C. Vogel, K. Schindler, and S. Roth, "Piecewise rigid scene flow," 2013.

[76] C. Vogel, S. Roth, and K. Schindler, "View-consistent 3D scene flow estimation over multiple frames," 2014.

[77] M. Menze, C. Heipke, and A. Geiger, "Joint 3d estimation of vehicles and scene flow," 2015.

[78] Z. Lv, C. Beall, P. Alcantarilla, F. Li, Z. Kira, and F. Dellaert, "A continuous optimization approach for efficient and accurate scene flow," 2016.

[79] F. Gney and A. Geiger, "Displets: Resolving stereo ambiguities using object knowledge," 2015.

[80] J. Hur and S. Roth, "Joint optical flow and temporally consistent semantic segmentation," 2016.

[81] L. Sevilla-Lara, D. Sun, V. Jampani, and M. J. Black, "Optical flow with semantic segmentation and localized layers," 2016.

[82] D. Sun, J. Wulff, E. Sudderth, H. Pfister, and M. Black, "A fully-connected layered model of foreground and background flow," 2013.

[83] M. Menze, C. Heipke, and A. Geiger, "Discrete optimization for optical flow," 2015.

[84] M. Bai, W. Luo, K. Kundu, and R. Urtasun, "Exploiting semantic information and deep matching for optical flow," 2016.

[85] A. Krull, F. Michel, E. Brachmann, S. Gumhold, S. Ihrke, and C. Rother, "6-dof model based tracking via object coordinate regression," 2014.

[86] L.-C. Chen, S. Fidler, A. L. Yuille, and R. Urtasun, "Beat the mturkers: Automatic image labeling from weak 3d supervision," 2014.

[87] M. Tatarchenko, A. Dosovitskiy, and T. Brox, "Multi-view 3d models from single images with a convolutional network," 2016.

[88] R. Zabih and J. Woodfill, "Non-parametric local transforms for computing visual correspondence," 1994.

[89] J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry," vol. 1607.02565, 2016.

[90] R. A. Newcombe, S. Lovegrove, and A. J. Davison, "DTAM: dense tracking and mapping in real-time," 2011.

[91] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: large-scale direct monocular SLAM," 2014.

[92] V. Kolmogorov, "Convergent tree-reweighted message passing for energy minimization," vol. 28, no. 10, pp. 1568–1583, 2006.

[93] N. Mayer, E. Ilg, P. Haeusser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox, "A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation," in *CVPR*, 2016.

[94] J. Žbontar and Y. LeCun, "Stereo matching by training a convolutional neural network to compare image patches," vol. 17, no. 65, pp. 1–32, 2016.

[95] M. Neoral and J. ochman, "Object scene flow with temporal consistency," 2017.

[96] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. B. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," 2014.

[97] R. B. Girshick, "Fast R-CNN," 2015.

[98] D. G. Lowe, "Local feature view clustering for 3D object recognition," *CVPR*, 2001.

[99] 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 *ICCV*, 2011.

[100] F. Michel, A. Kirillov, E. Brachmann, A. Krull, S. Gumhold, B. Savchynskyy, and C. Rother, "Global hypothesis generation for 6D object pose estimation," *CVPR*, 2017.

[101] S. Hinterstoisser, V. Lepetit, N. Rajkumar, and K. Konolige, "Going further with point pair features," in *ECCV*, 2016.

[102] M. Rad and V. Lepetit, "BB8: A scalable, accurate, robust to partial occlusion method for predicting the 3D poses of challenging objects without using depth," *ICCV*, 2017.

[103] 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 *ICCV*, 2017.

[104] B. Tekin, S. N. Sinha, and P. Fua, "Real Time Seamless Single Shot 6D Object Pose Prediction," in *CVPR*, 2018.

[105] E. Brachmann, A. Krull, F. Michel, S. Gumhold, J. Shotton, and C. Rother, "Learning 6D object pose estimation using 3D object coordinates," in *ECCV*, 2014.

[106] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. E. Reed, C. Fu, and A. C. Berg, "SSD: single shot multibox detector," in *ECCV*, 2016.

[107] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *CVPR*, 2016.

[108] K. He, G. Gkioxari, P. Dollr, and R. Girshick, "Mask r-cnn," in *ICCV*, 2017.

[109] E. Brachmann, F. Michel, A. Krull, M. Y. Yang, S. Gumhold, and C. Rother, "Uncertainty-driven 6D pose estimation of objects and scenes from a single RGB image," in *CVPR*, 2016.

[110] D. Huttenlocher, G. Klanderman, and W. Rucklidge, "Comparing images using the Hausdorff distance," *IEEE Trans. on PAMI*, 1993.

[111] S. Hinterstoisser, V. Lepetit, S. Ilic, S. Holzer, G. Bradski, K. Konolige, and N. Navab, "Model based training, detection and pose estimation of texture-less 3D objects in heavily cluttered scenes," in *ACCV*, 2012.

[112] R. Rios-Cabrera and T. Tuytelaars, "Discriminatively trained templates for 3D object detection: A real time scalable approach," in *ICCV*, 2013.

[113] T. Hodaň, X. Zabulis, M. Lourakis, Š. Obdržálek, and J. Matas, "Detection and fine 3D pose estimation of texture-less objects in RGB-D images," in *IROS*, 2015.

[114] W. Kehl, F. Tombari, N. Navab, S. Ilic, and V. Lepetit, "Hashmod: A hashing method for scalable 3D object detection," in *BMVC*, 2016.

[115] Y. Konishi, Y. Hanzawa, M. Kawade, and M. Hashimoto, "Fast 6D pose estimation from a monocular image using hierarchical pose trees," in *ECCV*, 2016.

[116] S. Hinterstoisser, C. Cagniart, S. Ilic, P. Sturm, N. Navab, P. Fua, and V. Lepetit, "Gradient response maps for real-time detection of texture-less objects," *IEEE Trans. on PAMI*, 2012.

[117] A. Tejani, D. Tang, R. Kouskouridas, and T.-K. Kim, "Latent-class Hough forests for 3D object detection and pose estimation," in *ECCV*, 2014.

[118] C. Zach, A. Penate-Sanchez, and M.-T. Pham, "A dynamic programming approach for fast and robust object pose recognition from range images," in *CVPR*, 2015.

[119] A. Doumanoglou, R. Kouskouridas, S. Malassiotis, and T. Kim, "6D object detection and next-best-view prediction in the crowd," in *CVPR*, 2016.

[120] 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 *ECCV*, 2016.

[121] B. Drost, M. Ulrich, N. Navab, and S. Ilic, "Model globally, match locally: Efficient and robust 3D object recognition," in *CVPR*, 2010.

[122] A. Kendall, M. Grimes, and R. Cipolla, "PoseNet: A convolutional network for real-time 6-DoF camera relocalization," in *ICCV*, 2015.

[123] E. Brachmann, A. Krull, S. Nowozin, J. Shotton, F. Michel, S. Gumhold, and C. Rother, "DSAC-Differentiable RANSAC for camera localization," *CVPR*, 2017.

[124] A. Krull, E. Brachmann, F. Michel, M. Y. Yang, S. Gumhold, and C. Rother, "Learning analysis-by-synthesis for 6D pose estimation in RGB-D images," *ICCV*, 2015.

[125] A. Behl, O. Hosseini Jafari, S. K. Mustikovela, H. A. Alhaija, C. Rother, and A. Geiger, "Bounding boxes, segmentations and object coordinates: How important is recognition for 3D scene flow estimation in autonomous driving scenarios?" in *ICCV*, 2017.

[126] A. Gaidon, Q. Wang, Y. Cabon, and E. Vig, "Virtual worlds as proxy for multi-object tracking analysis," in *CVPR*, 2016.

[127] H. A. Alhaija, S. K. Mustikovela, L. Mescheder, A. Geiger, and C. Rother, "Augmented reality meets deep learning for car instance segmentation in urban scenes," in *BMVC*, 2017.

[128] C. Li, M. Z. Zia, Q. Tran, X. Yu, G. D. Hager, and M. Chandraker, "Deep supervision with shape concepts for occlusion-aware 3D object parsing," in *CVPR*, 2017.

[129] W. Kabsch, "A solution for the best rotation to relate two sets of vectors," *Acta Crystallographica*, 1976.

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

[131] X.-S. Gao, X.-R. Hou, J. Tang, and H.-F. Cheng, "Complete solution classification for the perspective-three-point problem," *IEEE Trans. on PAMI*, 2003.

[132] V. Lepetit, F. Moreno-Noguer, and P. Fua, "EPNP: An accurate O(n) solution to the PNP problem," *IJCV*, 2009.

[133] P. K. Nathan Silberman, Derek Hoiem and R. Fergus, "Indoor segmentation and support inference from rgbd images," in *ECCV*, 2012.

[134] R. Girshick, "Fast R-CNN," in *ICCV*, 2015.

[135] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *ICLR*, 2015.

[136] C. Rother, T. Minka, A. Blake, and V. Kolmogorov, "Cosegmentation of image pairs by histogram matching-incorporating a global constraint into mrfs," in *CVPR*, 2006.

[137] A. Kowdle, D. Batra, W.-C. Chen, and T. Chen, "imodel: Interactive co-segmentation for object of interest 3d modeling," in *ECCV workshop*, 2010.

[138] T. Shen, G. Lin, L. Liu, C. Shen, and I. Reid, "Weakly supervised semantic segmentation based on co-segmentation," in *BMVC*, 2017.

[139] S. Vicente, C. Rother, and V. Kolmogorov, "Object cosegmentation," in *CVPR*, 2011.

[140] D. Batra, A. Kowdle, D. Parikh, J. Luo, and T. Chen, "icoseg: Interactive co-segmentation with intelligent scribble guidance," in *CVPR*, 2010.

[141] M. Rubinstein, A. Joulin, J. Kopf, and C. Liu, "Unsupervised joint object discovery and segmentation in internet images," in *CVPR*, 2013.

[142] S. Vicente, V. Kolmogorov, and C. Rother, "Cosegmentation revisited: Models and optimization," in *ECCV*, 2010.

[143] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *CVPR*, 2015.

[144] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *MICCAI*, 2015.

[145] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. H. S. Torr, "Conditional random fields as recurrent neural networks," in *ICCV*, 2015.

[146] N. Xu, B. Price, S. Cohen, J. Yang, and T. S. Huang, "Deep interactive object selection," in *CVPR*, 2016.

[147] R. Quan, J. Han, D. Zhang, and F. Nie, "Object co-segmentation via graph optimized-flexible manifold ranking," in *CVPR*, 2016.

[148] T. Taniai, S. N. Sinha, and Y. Sato, "Joint recovery of dense correspondence and cosegmentation in two images," in *CVPR*, 2016.

[149] C. Lee, W.-D. Jang, J.-Y. Sim, and C.-S. Kim, "Multiple random walkers and their application to image cosegmentation," in *CVPR*, 2015.

[150] L. Mukherjee, V. Singh, and C. R. Dyer, "Half-integrality based algorithms for cosegmentation of images," in *CVPR*, 2009.

[151] D. S. Hochbaum and V. Singh, "An efficient algorithm for co-segmentation," in *ICCV*, 2009.

[152] Y. Y. Boykov and M.-P. Jolly, "Interactive graph cuts for optimal boundary & region segmentation of objects in nd images," in *ICCV*, 2001.

[153] A. Joulin, F. Bach, and J. Ponce, "Discriminative clustering for image co-segmentation," in *CVPR*, 2010.

[154] J. C. Rubio, J. Serrat, A. López, and N. Paragios, "Unsupervised co-segmentation through region matching," in *CVPR*, 2012.

[155] H. Fu, D. Xu, S. Lin, and J. Liu, "Object-based rgbd image co-segmentation with mutex constraint," in *CVPR*, 2015.

[156] J. Carreira and C. Sminchisescu, "Constrained parametric min-cuts for automatic object segmentation," in *CVPR*, 2010.

[157] Z. Yuan, T. Lu, and Y. Wu, "Deep-dense conditional random fields for object co-segmentation," in *IJCAI*, 2017.

[158] J. R. Uijlings, K. E. Van De Sande, T. Gevers, and A. W. Smeulders, "Selective search for object recognition," *IJCV*, 2013.

[159] M. D. Collins, J. Xu, L. Grady, and V. Singh, "Random walks based multi-image segmentation: Quasiconvexity results and gpu-based solutions," in *CVPR*, 2012.

[160] X. Dong, J. Shen, L. Shao, and M.-H. Yang, "Interactive cosegmentation using global and local energy optimization," *IEEE Transactions on Image Processing*, 2015.

[161] H. Noh, S. Hong, and B. Han, "Learning deconvolution network for semantic segmentation," in *ICCV*, 2015.

[162] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," in *ICLR*, 2016.

[163] G. Lin, A. Milan, C. Shen, and I. Reid, "Refinenet: Multi-path refinement networks for high-resolution semantic segmentation," in *CVPR*, 2017.

[164] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *CVPR*, 2017.

[165] N. Xu, B. Price, S. Cohen, J. Yang, and T. Huang, "Deep grabcut for object selection," in *BMVC*, 2017.

[166] G. Li and Y. Yu, "Deep contrast learning for salient object detection," in *CVPR*, 2016.

[167] L. Wang, L. Wang, H. Lu, P. Zhang, and X. Ruan, "Saliency detection with recurrent fully convolutional networks," in *ECCV*, 2016.

[168] S. D. Jain, B. Xiong, and K. Grauman, "Pixel objectness," *arXiv:1701.05349*, 2017.

[169] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for scene segmentation," *TPAMI*, 2017.

[170] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Semantic image segmentation with deep convolutional nets and fully connected crfs," in *ICLR*, 2015.

[171] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *ICLR*, 2015.

[172] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. van der Smagt, D. Cremers, and T. Brox, "Flownet: Learning optical flow with convolutional networks," in *ICCV*, 2015.

[173] J. Shotton, J. Winn, C. Rother, and A. Criminisi, "Textonboost: Joint appearance, shape and context modeling for multi-class object recognition and segmentation," in *ECCV*, 2006.

[174] A. Faktor and M. Irani, "Co-segmentation by composition," in *ICCV*, 2013.

[175] B. Hariharan, P. Arbelaez, L. Bourdev, S. Maji, and J. Malik, "Semantic contours from inverse detectors," in *ICCV*, 2011.

[176] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *CVPR*, 2009.

[177] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *ICLR*, 2015.

[178] F. Wang, Q. Huang, and L. J. Guibas, "Image co-segmentation via consistent functional maps," in *ICCV*, 2013.

[179] K. R. Jerripothula, J. Cai, and J. Yuan, "Image co-segmentation via saliency co-fusion," *IEEE Transactions on Multimedia*, 2016.

[180] X. Chen, A. Shrivastava, and A. Gupta, "Enriching visual knowledge bases via object discovery and segmentation," in *CVPR*, 2014.

[181] K. R. Jerripothula, J. Cai, F. Meng, and J. Yuan, "Automatic image co-segmentation using geometric mean saliency," in *ICIP*, 2014.

[182] M. Ranzato, F. J. Huang, Y. Boureau, and Y. LeCun, "Unsupervised learning of invariant feature hierarchies with applications to object recognition," in *CVPR*, 2007.

[183] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, ser. NIPS'12, 2012.

[184] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016.

[185] D. Scherer, A. C. Müller, and S. Behnke, "Evaluation of pooling operations in convolutional architectures for object recognition," in *ICANN*, 2010, pp. 92–101.

[186] A. R. Zamir, A. Sax, W. B. Shen, L. J. Guibas, J. Malik, and S. Savarese, "Taskonomy: Disentangling task transfer learning," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2018.

[187] B. Hariharan, P. Arbelaez, R. Girshick, and J. Malik, "Hypercolumns for object segmentation and fine-grained localization," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[188] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *CVPR*, 2017.

[189] H. Zhao, Y. Zhang, S. Liu, J. Shi, C. C. Loy, D. Lin, and J. Jia, "PSANet: Point-wise spatial attention network for scene parsing," in *ECCV*, 2018.

[190] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Semantic image segmentation with deep convolutional nets and fully connected crfs," in *ICLR*, 2015.

[191] F. Yu, V. Koltun, and T. Funkhouser, "Dilated residual networks," in *Computer Vision and Pattern Recognition (CVPR)*, 2017.

[192] K. Sun, B. Xiao, D. Liu, and J. Wang, "Deep high-resolution representation learning for human pose estimation," in *CVPR*, 2019.

[193] J. Wang, K. Sun, T. Cheng, B. Jiang, C. Deng, Y. Zhao, D. Liu, Y. Mu, M. Tan, X. Wang, W. Liu, and B. Xiao, "Deep high-resolution representation learning for visual recognition," *CoRR*, vol. abs/1908.07919, 2019.

[194] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[195] S. R. Richter, V. Vineet, S. Roth, and V. Koltun, "Playing for data: Ground truth from computer games," in *European Conference on Computer Vision (ECCV)*, ser. LNCS, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds., vol. 9906. Springer International Publishing, 2016, pp. 102–118.

[196] A. Zamir, A. Sax, T. Yeo, O. Kar, N. Cheerla, R. Suri, Z. Cao, J. Malik, and L. Guibas, "Robust learning through cross-task consistency," *arXiv*, 2020.