TU Informatics

Geometrische Analyse und Posenschätzung mittels maschinellem Lernen

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Data Science

eingereicht von

Philipp Ausserlechner, BSc
Matrikelnummer 01118433

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Markus Vincze
Mitwirkung: Senior Scientist, Dipl.-Ing. Dr.techn. Csaba Beleznai

Wien, 27. August 2021

Philipp Ausserlechner	Markus Vincze

TU Informatics

Machine Learning guided geometric analysis and pose estimation

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Data Science

by

Philipp Ausserlechner, BSc
Registration Number 01118433

to the Faculty of Informatics

at the TU Wien

Advisor: Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Markus Vincze
Assistance: Senior Scientist, Dipl.-Ing. Dr.techn. Csaba Beleznai

Vienna, 27th August, 2021

Philipp Ausserlechner

Markus Vincze

Erklärung zur Verfassung der Arbeit

Philipp Ausserlechner, BSc

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 27. August 2021

Philipp Ausserlechner

Danksagung

Diese Arbeit entstand in Zusammenarbeit zwischen dem ACIN Institut der Technischen Universität Wien (TU Wien) und dem Institut für Assistive und Autonomous Systems des Austrian Institute of Technology (AIT).
Mein besonderer Dank gilt Dr. Csaba Beleznai für seine Betreuung und Wegweisung während des gesamten Projekts. Ich möchte auch meinen Dank an Ao. Univ. Prof. Dipl.-Ing. Dr.techn. Markus Vincze für sein Feedback und seine Ratschläge ausdrücken.
Schließlich möchte ich meiner Familie dafür danken, dass sie mich auf meinem Bildungsweg von der Grundschule bis zum Universitätsabschluss unterstützt haben.

Acknowledgements

This thesis was a collaboration between the ACIN Institute from the Technical University of Vienna (TU Wien) and the Institute for Assistive and Autonomous Systems from the Austrian Institute of Technology (AIT).
My special thanks go to Dr. Csaba Beleznai for his supervision and guidance throughout the project. I also want to express my gratitude towards Ao. Univ. Prof. Dipl.-Ing. Dr.techn. Markus Vincze for his feedback and counseling.
Lastly, I want to thank my family for supporting me on my education journey, from primary school to university diploma.

Kurzfassung

Jüngste Trends in Computer Vision ermöglichen zunehmend eine verbesserte räumliche Wahrnehmung einer gegebenen Umgebung und ihrer Objekte. Das Erlernen erscheinungsbasierter Darstellungen aus RGB-Bildern ist Gegenstand intensiver Forschung. Das Thema der Interpretation von Tiefendaten in Bezug auf gelernte Modelle umfasst jedoch noch viele offene Forschungsfragen. In dieser Studie führen wir ein Encoder-Decoder basiertes Lernverfahren ein, um Objektposen aus Tiefenbildern und den entsprechenden Oberflächennormalen zu schätzen. Unsere Zielobjekte sind mehrere Instanzen von quaderförmigen Primitiven unbekannter Größe. In unseren Experimenten verwenden wir monokulare Tiefenschätzung und lernbasierte Stereo-Matching-Methoden, um Tiefenbilder zu erzeugen. Die Neuartigkeit unseres Ansatzes besteht in dem vorgeschlagenen geometriebasierten Erkennungsschema, das ausschließlich anhand von synthetischen Bildern trainiert werden kann und in der Lage ist, parametrische (orientierte) Objektteile zu schätzen. Die Ergebnisse zeigen eine genaue räumliche Lokalisierung von generischen Quadermodellen in realen Szenarien. Wir zeigen auch, dass das vorgeschlagene generische Schema leicht für andere Geometrietypen umkonfiguriert werden kann. Wir präsentieren Ergebnisse für Paletten bekannter Größe, bei denen die Kanten zwischen einer Reihe von ko-planaren Punkten unser geschätztes Strukturmodell darstellen.

Abstract

Recent trends in computer vision increasingly allow for an enhanced spatial perception of a given environment and its objects. Learning appearance-based representations from RGB images is the subject of intense research. However, the topic of interpreting depth data in terms of learned models still encompasses many open research questions. In this study, we introduce an encoder-decoder-type learning scheme to estimate object poses from depth images and its corresponding surface normals. Our targeted objects are multiple instances of cuboid primitives of unknown size. In our experiments, we employ monocular depth estimation and learning-based stereo-matching methods to generate depth images. The novelty of our approach is given by the proposed geometry-aware detection scheme, which can be trained solely from synthetic images and can estimate parametric (oriented) object parts. Results demonstrate accurate spatial localization of generic cuboid models in real scenarios. We also demonstrate that the proposed generic scheme can be easily re-configured for other geometry types. We show results for pallets of known size, where edges between a set of co-planar points represent our estimated structural model.

Contents

CHAPTER 1

Introduction and Motivation

Object detection and pose estimation are well-known tasks in Computer Vision, substantially contributing to model a perceived environment in terms of a set of pre-defined entities. Robotics and autonomous driving are example task domains, where such representational traits are needed. In recent years, appearance-based scene interpretation has demonstrated significant progress, where appearance variations within targeted classes are often learned in an exhaustive manner. However, learning schemes focusing on the scene geometry and spatial relations therein still hold many open research perspectives. Most importantly, instead of learning all possible object appearances, learning geometric representations discards photometric and appearance variations and it captures a simpler innate object property. This reduced representational space implies less data needed for learning. Furthermore, a rich set of shapes can be defined as parametric structures, implying opportunities to formulate learning as a direct regression task.

1.1 Problem Statement

Our goal is to algorithmically construct the pose of cuboid-shaped objects of unknown size and appearance from single-camera RGB and stereo-camera input (Fig. 1.1). Additionally, we compare the influence of the geometric representations on the pose estimation results. The class of cuboid-like objects ranges from boxes to cargo containers. Since the number of target size and appearance configurations is practically unlimited, this scenario can be called a semi-open world. Therefore the detector should not be influenced by the object's appearance, size, or texture and instead focus on their uniting property, the geometry. Additionally, we train our model solely on synthetic data to overcome the necessity of labor-intensive manual annotation, which is the main cost factor and a limitation of modern learning-based object detection and pose estimation methods. The geometry-driven detection scheme is a step towards a more general-purpose robotic vision since it enables an intelligent robotic agent to operate in an environment without specific knowledge about every single object within this scene. Our approach resembles the interaction of a human with its environment. People rely on a geometric interpretation of a scene to navigate within.

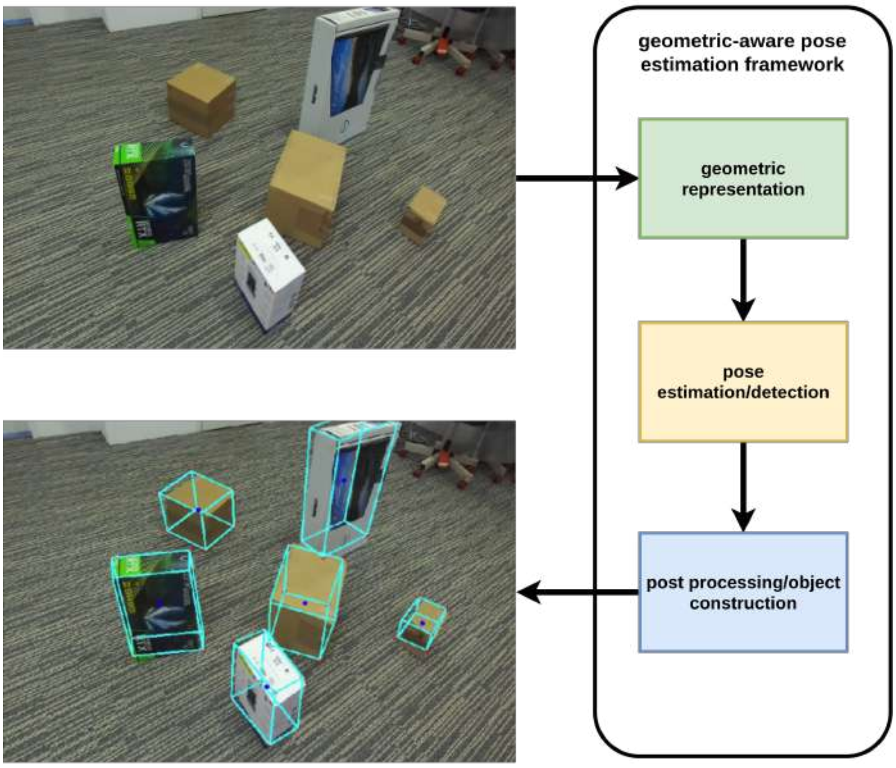

geometric-aware pose estimation framework
geometric representation
pose estimation/detection
post processing/object construction

Figure 1.1: Illustration of the overall geometric-aware pose estimation method, with a real world inference example of the final framework

1.2 Challenges

We have to design a data synthesis framework to create a vast training data set with corresponding ground truth. The data set has to have a close synth-to-real gap to allow our trained pose estimation model to generalize to real-world scenarios. The training data set has to reflect reality to enable the detector to deal with occlusions, different views, and object textures of real-world images. Moreover, we need to construct a geometric representation of the images to overcome object appearance configurations and reduce the gap between synthetic and real-world domains.
Further, we present a methodology for cuboid shapes, but the method is not limited to this geometry class. We have to design a robust pose estimation model, which allows us to capture the relevant cuboid key points to create a wire-frame representation of the objects. This includes initial detection and additional algorithmic refinement steps.

Lastly, we have to test the detection pipeline on annotated real-world samples and choose an appropriate metric to evaluate those results.
The main challenge of this thesis is to deal with the endless variety of different cuboid shapes, appearances, and textures in such a way that the pose of those can be estimated, without any additional prior knowledge about the objects, besides their cuboid shape. State-of-the-art methods for pose estimation usually have a closed world assumption where they only detect objects which are known beforehand. This is not the case in our scenario since we want to detect every cuboid-shaped object, which can have all kinds of different sizes and appearances. In other words, it should not matter if the cuboid is a shipping container or a matchbox.

1.3 Contributions

The presented method in this thesis combines data synthesis and geometric representation of the data. To estimate the pixel-wise disparity, we use a pre-trained monocular depth estimation network (MiDaS [RLH+19]) and a deep learning-based stereo-matching model (AANET [XZ20]). From the disparity maps, we can calculate the surface normal representation of the data. Due to this procedure, we exclude the appearance information of the original input images. This geometrical representation is less complex and therefore reduces training time.
For the pose estimation, we use a single-stage encoder-decoder backbone with multiple heads, closely related to CenterNet framework [ZWK19]. The output heads allow us to detect the object corners and regress from their orientations and corresponding edge lengths. With those oriented corners we can construct the cuboid poses in a bottom-up manner via a from us designed so-called cuboid construction algorithm.
The two key contributions of this thesis are:

- demonstrating that geometry-based detection and pose estimation from purely synthetic data is feasible in real-world scenarios
- an extensible representational concept towards other geometries

The main contribution from a scientific point of view is the geometry-based learning strategy. This approach stands in stark contrast to the widely used appearance-based learning strategies. Additionally, our bottom-up strategy is a step towards a more general-purpose pose estimation without prior knowledge of the objects themselves.
From an applied point of view, the main contribution depends on our ability to overcome the necessity of large amounts of hand-annotated real-world training data. Our approach allows us to train pose estimation models for real-world robotic tasks without labor and cost-intensive manual data-set collection. This method can be used in a wide range of automation scenarios and in particular in the case of an autonomous forklift, as indicated by our palette detection results.

1.4 Results Preview

The results show the feasibility of our semi-open-world cuboid detection and pose estimation method on real-world data. Since the monocular depth estimation is not suitable for 3D pose recovery, we rely on estimating the 2D planar projection of the 3D pose. Thus, we can quantitatively compare the stereoscopic and monocular data representations. To evaluate the pose estimation results, we choose the Intersection over Union (IoU) metric. The test data set includes different viewpoints, object sizes as well as occlusions and clutter. Since there was no appropriate cuboid-pose-estimation benchmark set, which fits our purpose, we decided to collect our internal test dataset and annotated it.

Additionally, we show qualitative pose estimation results for a practically highly relevant object in the domain of warehouse logistics, namely the palette. Those results indicate the generality of our geometric-aware pose estimation method. We can learn all kinds of geometrically distinct objects. In all experiments, we solely trained on synthetic data, which we generated with our Blender pipeline. This procedure and the ability to generalize to real-world scenarios make our approach beneficial for practical applications.

1.5 Thesis Outline

The following two chapters present the state-of-the-art section of this thesis. The first chapter discusses learning-based pose estimation methods, which are usually appearance-based.

The second chapter focuses on depth measuring and estimation techniques. Here we explain state-of-the-art methods for monocular depth estimation and learning-based stereo-matching methods.

The fourth chapter outlines the details of the geometric aware pose estimation method. The section includes the data generation process, the detection methodology, and the algorithmic cuboid refinement procedure for capturing the 8-corner structure of a generic cuboid.

The fifth chapter is the result section, where we discuss the experiment setup, the evaluation, and the summary of the cuboid and palette experiments.

The last chapter gives an overall conclusion of the thesis and outlines potential future experiments.

CHAPTER 2

Pose estimation using End-to-end Learning

In this section, we discuss state-of-the-art learning methods for performing 2D and 3D pose estimation. So far, data from depth-sensing (such as stereo, ToF, LiDAR) methods have been the standard to estimate the spatial parameters. However, the recent emergence of modern End-to-end learning, as so-called "universal function approximators" increasingly allows the estimation of pose parameters directly from images. Therein, important representational aspects are how to describe and model ambiguities involving the sought object-pose parameters.

2.1 Definition of Main Concepts and Paradigms

In this section, we discuss the architectural prerequisites to perform state-of-the-art pose estimation. Thus, we examine convolutional neural networks (CNN), residual layers, the incorporation of feature representations from all scales, and multi-task learning.

2.1.1 CNNs for Object Detection and Pose Estimation

Since the rise of convolutional neural networks (CNNs) in the domain of computer vision, due to its breakthrough results in the ImageNet classification challenge [KSH12] in 2012, this architecture dominates most applications in this area of research. The ImageNet dataset [DDS$^+$09] consists of around 15 million images with 22000 different categories, which makes it a hard object classification task. Compared to fully connected neural networks the main advantage of convolution depends on their much lower amount of learn-able parameters/weights, which makes them significantly easier to train. We achieve the parameter reduction by learning a single convolution matrix for every channel. Those channels are feature maps, where every channel encodes a characteristic of the input image. An example of such a map would be encoding the edges, corners, colors, and so on. After a convolution layer typically follows a max-pooling layer, which extracts the maximum value of a predefined size pixel square. The standard building block of a CNN is visualized in figure 2.1.

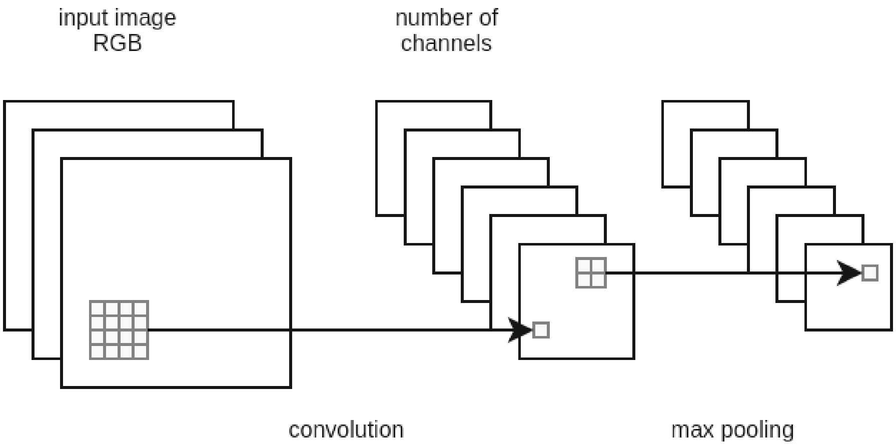


Figure 2.1: Illustration of a convolution and max pooling layer

An important configuration of the CNN building block is the size of the sliding window and the offset from one window to another, also known as the stride. Figure 2.2 illustrates the sliding window for a max-pooling layer.

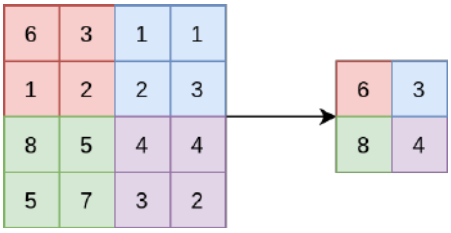


Figure 2.2: Max pooling with a 2x2 window size and a stride of 2.

An additional difficulty when designing a neural network is choosing an appropriate amount of hidden layers, often also referred as depth. Networks with a high amount of layers tend to be very hard to train. Often it is not even possible to train an identity function on such deep structures. To overcome this issue, He and Zhang [HZRS16] proposed a deep residual learning strategy, which allowed them to train deep networks also on relatively simple tasks with moderate amounts of training data. Their architecture called ResNet won the ImageNet [DDS$^+$09] challenge and is widely used as the backbone for many tasks in computer vision. The main idea of the ResNet architecture is the introduction of so-called skip or residual layers (Fig. 2.3).

These skip layers serve as shortcut connections for one or more layers. The idea is to add the identity x from the first layer to the last layer of the residual block. With this scheme

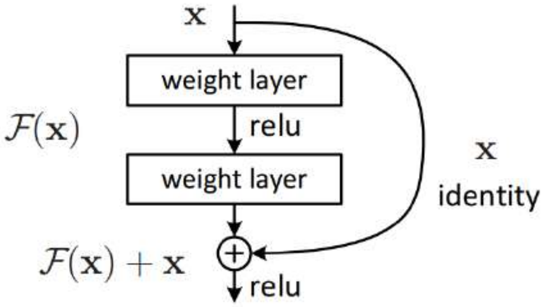

x
weight layer
relu
weight layer
$\mathcal{F}(\mathbf{x})$
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Figure 2.3: Illustration of a residual block [HZRS16]

the authors [HZRS16] were able to tackle the so-called degradation problem, which means that with increasing network depth the accuracy saturates and then degrades. This is not the case for architectures that use residual blocks. However, we still face the problem that stacked convolutional layers, especially if decreasing by size, tend to allow only considerations of global features and lack more region-specific details. This common issue is the theme of the next section.

2.1.2 Incorporating Feature Representations from Different Scales

In this section we describe, the representational strategies evolved from multi-scale detection approaches towards encoder-decoder type networks simultaneously estimating multiple parameters or tasks. Most modern computer vision detectors use a CNN backbone due to their ability to generate and learn feature representations from training data. Noteworthy examples for such detectors are R-CNN [GDDM14], Fast R-CNN [Gir15] and Yolo [RDGF16]. The problem with a simple convolutional encoder backbone is its lack of capability to incorporate information of different scales. For example, if we use a pyramid-like structure consisting of convolutional layers, we construct a representation of the overall image features at the final encoder layer. However, such a representation loses a lot of the regional-specific feature information, which is part of earlier convolutional stages. This issue leads to approaches wherein a first step generates regional proposals, which are classified via a convolutional neural network, like in the case of R-CNN [GDDM14]. Those two-stage detectors come with the drawback that they lack end-to-end trainability. Further, they tend to be computational very expensive due to the exhaustive region proposal process. A much more elegant way to tackle this problem is aggregating information of different convolutional stages to a more generalized representation of the original input. One way to achieve this is the usage of the so-called Deep Layer Aggregation [YWSD18], which is an architectural solution to build a one-stage detector for feature incorporation on different scales. Yu and Wang propose two strategies to achieve this behavior. The first method is the iterative deep layer aggregation (IDA). The second is the hierarchical deep layer aggregation (HDA) (Fig. 2.4 & 2.5).

The IDA method resembles the concept of the skip layer but expands the scheme to a

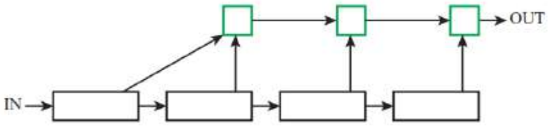


Figure 2.4: Illustration of the iterative deep layer aggregation [YWSD18]

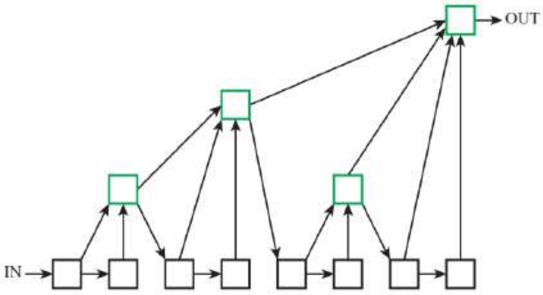


Figure 2.5: Illustration of the hierarchical deep layer aggregation [YWSD18]

more systematic methodology. The idea to aggregate feature maps of early convolutional stages into the final network output was already explored in earlier works ([KKS+18] & [LSD15]). But those methods cannot incorporate the feature maps on different stages, which results in a less generalized representation. Those last stage feature compounding architectures are called shallow aggregation networks. On the other hand, the iterative method allows compounding the convolutional representations on every stage. However, the IDA has its drawbacks, the architecture cannot fuse more than one block of the network, since it still works sequentially. To overcome this Yu and Wang [YWSD18] also designed a tree-like hierarchical aggregation strategy (HDA). The HDA merges blocks of feature channels on different stages and feeds them into the backbone as input for the next tree block. With this approach, the feature representation propagates from all previous blocks and not only from the preceding block of the network. Those aggregation strategies allowed Yu and Wang [YWSD18] to produce a new state-of-the-art benchmark in many computer vision tasks. Throw out the thesis we will use a DLA backbone for our experiments due to their performance and end-to-end train-ability.

Another promising method to incorporate feature representations from different scales throw out the network is so-called encoder-decoder architecture. Newell, Yang, and Deng [NYD16] introduced such an encoder-decoder network for pose estimation. Their hourglass architecture (Fig. 2.6) achieves a similar behavior as the HDA but without the necessity to explicitly design an aggregation tree structure, where you have to define on which stages information gets compounded. The hourglass was designed for human pose estimation and is closely related to other encoder-decoder architectures, which show promising results in numerous computer vision tasks like image segmentation [BKC17], capturing material reflectance [RRF+16] and classification [ZMGL15]. The concept of the hourglass is consecutive down and up-sampling, where intermediate feature maps from the down-sampling get combined with their up-sampling counterpart in a residual skip connection manner.

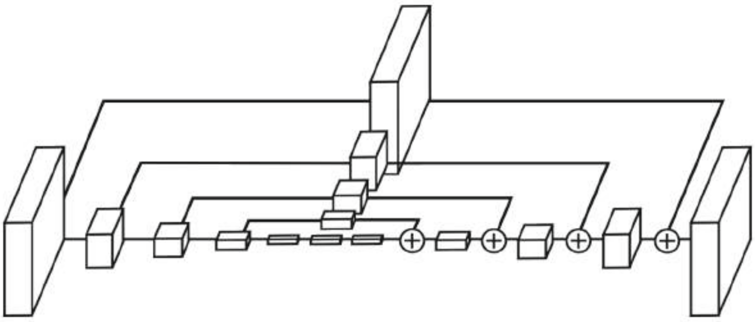


Figure 2.6: Illustration of an hourglass module, which is a typical encoder-decoder structure [NYD16]

The bottleneck of the encoder-decoder architecture allows capturing global information of the whole input image. The shallow layers allow consideration of more local features. The network design enables the effective processing of multiple resolutions and combines those features at later stages in the network. A single hourglass module produces a heatmap-type output (2D multi-modal Gaussian distributions), including the key points of interest. To push this methodology even further, Newell, Yang, and Deng [NYD16] stacked multiple hourglass modules on each other (Fig. 2.7) to enable feature reconsideration for even more robust detection results and reevaluation of initial estimates.

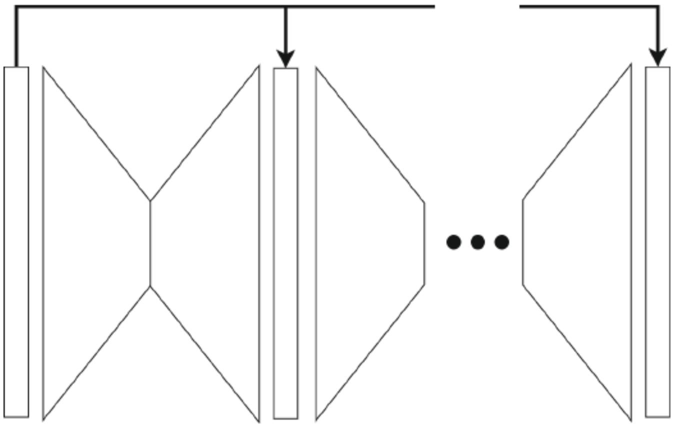


Figure 2.7: Visualization of stacked hourglass modules

In conclusion, the Deep Layer Aggregation [YWSD18] as well as the hourglass architecture [NYD16] can incorporate features of different scales and allow using all information at the final stage of the network. These properties qualify such architectures as an appropriate backbone for state-of-the-art image processing networks. Such a representation can be applied to many different computer vision tasks, but we have only considered single task networks.

One common backbone is not limited to a single task, it can fulfill many purposes at the same time, which leads us to the next topic, namely multi-task-learning.

2.1.3 Multi-task Learning

The idea of multi-task learning is to build multiple heads on a common backbone. All output heads are trained simultaneously via a composite loss function, which is a weighted linear combination of losses. Where each loss represents one output head of the detector. An example is the CornerNet [LD18], which predicts heatmaps (heads) with the left-upper and right-lower bounding box anchor for objects of interest. In this case, the hourglass network outputs an additional 1-dimensional embedding map to group the bounding box anchors with each other (Fig. 2.8).

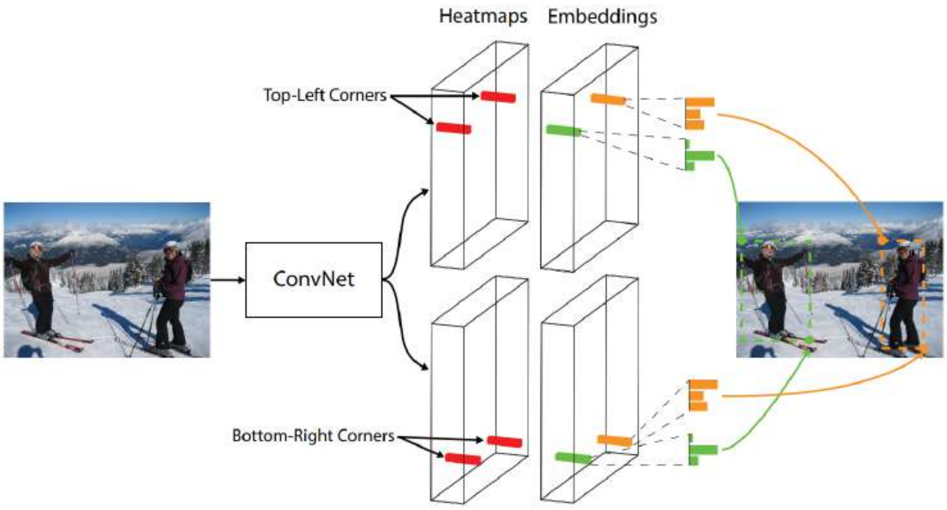


Figure 2.8: Visualization of the multiple heads of the CornerNet [LD18]

An even more elegant solution for object detection is the CenterNet [ZWK19], which directly estimates the center of an object and regresses from their additional key points for bounding box creation. The main output of the CenterNet detector is a heatmap with the object centers. Additional properties like bounding box size, orientation, or object-associated key points are the output of secondary heads. The CenterNet concept is very flexible and can be easily modified to other object detection and pose estimation tasks and provides robust detection results in many computer vision applications. Throughout this thesis, we will extend the CenterNet detection methodology towards the incorporation of geometric knowledge to evolve detection to overcome appearance-based information dependency.

2.2 State-of-the-art in Pose Estimation

Detecting the pose in a 2D projection is a common task in computer vision and gives the possibility to combine the ambiguous appearance of 2D projection with learned priors from data to perform 6DoF estimation tasks. However, to enable robots and autonomous agents to simultaneously map and locate (SLAM) in an unknown environment, a 3D representation is required. The 3D object pose often also referred as 6D pose or 6 DoF (degrees of freedom) consists of 3 Cartesian coordinates $\mathbf{x} = (x, y, z)$ and the 3 Eulerian angles $\theta = (\theta_r, \theta_p, \theta_y)$ roll, pitch and yaw. The object translation in a scene is described by the $\mathbf{x}$ and the corresponding rotation of the object by θ. Those measures are relative to the camera, where the scene is projected onto an image plane.

Peng and Liu [PLH$^+$19] propose a Pixel-wise Voting Network (PVNet) to achieve a 6D object pose recovery (Fig. 2.9). The PVNet is a two-stage process, where during the first stage the 2D keypoints are detected. In a second stage, the 6D pose parameters are computed via Perspective-n-Point (PnP) algorithm [WH06]. To detect the key points, Peng and Liu use a voting-based keypoint localization on RGB images. They create a vector field where each pixel represents the direction to the closest key point. Additionally, they add a segmentation head that classifies each pixel as a background or target object. This technique forces the model to focus on local information, which shows robust results concerning occlusions. The spatial probability for a keypoint location is calculated via a RANSAC-like voting scheme. The downside of this approach is its closed world assumption, which means that only beforehand known objects can be detected and reconstructed in a 6D pose.

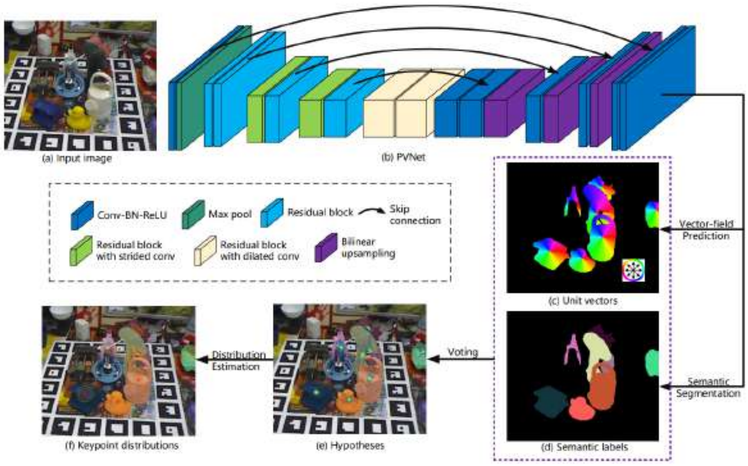


Figure 2.9: Visualization of the PVNet 6D pose estimation scheme [PLH$^+$19]

Another quite similar method to achieve pose estimation for known objects is PoseCNN [XSNF17]. Xiang also uses a pixel-wise vector-field representation, but instead of detecting the object keypoints, they directly detect the object centers. Since they also depend on a closed world assumption, they can calculate the translation matrix T from the center and the object size. An additional output head for 3D rotation regression estimates the rotational matrix R with a Quaternion representation. The PoseCNN approach is a mixture between a top-down and a bottom-up methodology. They use a pixel-wise labeling step, which is bottom-up. Afterward, they regress the pose from those labeled pixels, which once more only works for known objects and therefore is a top-down approach.

Lepetit and Pitteri [PIL19] use a different strategy to recover the 6D object pose from RGB-D images, which is much more related to the methodology we discuss in this thesis. The method relies on object geometries and does not require re-training for every object which they want to detect. Thus, this provides significant practical advantages, due to the avoidance of resource-intensive re-training procedures.

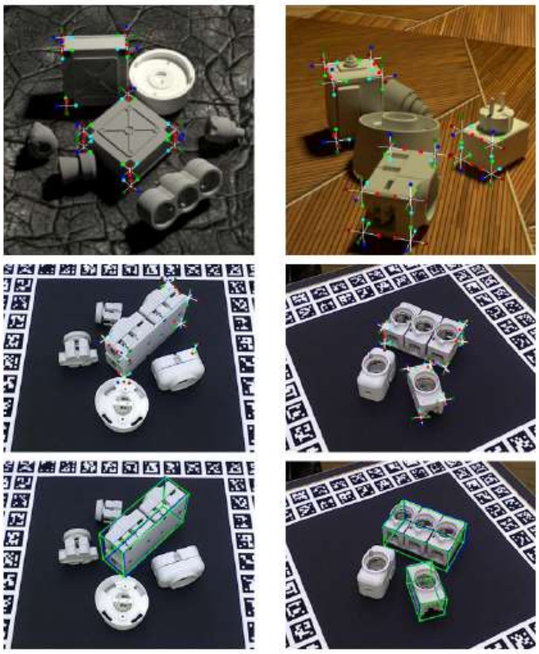


Figure 2.10: Visualization the 2D corner detection and corresponding 3D bounding box [PIL19]

The world of industrial objects is very rectangular and consists of a lot of geometric primitives. Lepetit and his group make use of this fact in their detection method. They use a Faster R-CNN [RHGS15] backbone to detect regions of interest and estimate 2D corners within those regions. Those 2D object corners get matched with a 3D CAD model in a RANSAC-like fashion. The 2D corners are the basis for estimating the 3D position of the object, then those corners are projected back on the plane for verification (Fig. 2.10). Just by switching to another CAD object model, the same detector can estimate the 3D pose of another object type. Lepetit and Pitteri achieve state-of-the-art results on the T-LESS dataset [HHO+17], which includes a white range of symmetric objects, which are typically quite challenging for pose estimation, due to symmetry-dependent ambiguities.

2.3 Summary

Most modern pose estimation methods are dependent on closed world assumptions and often use a vector-field representation to regress the necessary key points for object pose recovery. The detectors commonly use RGB appearance information as input and therefore neglect the geometric properties of the objects of interest. Recently, the trend is moving away from closed-world assumptions, and the availability of accurate and cheap depth sensors is emerging. This thesis proposes a methodology to leverage these trends since we focus on incorporating geometric information for pose estimation purposes. Depending on a first stage depth estimation procedure to get ride-off the vast amount of possible object appearances. Additionally, we detect geometric primitives, namely cuboids, of unknown size in a bottom-up manner. The bigger picture of this methodology is to construct the environment and included objects as cuboid composites.

CHAPTER 3

Depth Estimation as the Basis of Geometric Representation

In this thesis, we target pose estimation via geometric cues. Therefore depth estimation is crucial, as it encodes spatial relationships in a scene. Estimating the depth of a scene is a ubiquitous task for humans to navigate and operate in any environment. We achieve these tasks with visual input, depending on two separate images, one image per eye. The image pairs parallaxes properties are exploited to estimate distances. This seemingly effortless ability of humans is difficult for machines and crucial for various tasks in the reign of robotics.
In contrast to humans, artificial agents use a greater variety of different sensing modalities to measure the depth of the environment. The most widely used are:

1. active depth sensing, such as LIDAR
2. depth estimation from a stereo image pair (passive sensing)
3. monocular depth estimation

LIDAR (Light detection and ranging) sensors determine distances, via sending a laser and measuring the time for the reflected light to return to the receiver. This method is widely used in the domain of autonomous driving [YTO+14], due to its capability to measure the distance on long ranges accurately. Kashani [KOPW15] gives a comprehensive overview of different LIDAR systems. However, a disadvantage of LIDAR systems is that they are costly and provide only a spatially sparse depth representation. Therefore, we will focus on stereo vision and monocular depth estimation methods. Those methods are cost-effective and capable of estimating accurate depth on short distances, which meets our expectations in this thesis.

3.1 Monocular Depth Sensing and Learning-based Methods in Particular

To some extent, monocular depth estimation resembles the intuitive geometric awareness of a human due to its dependence on knowledge of the world. Object properties, shadows, and geometric distortion are taken into consideration to estimate the depth of scenes. It is very similar to closing one eye and estimate which objects are further away than others. In most scenarios, it is still possible to give a reasonable estimate, but only because of the observer's general knowledge of the world.

A possibility to incorporate geometric understanding into the depth estimation process is to create hand-crafted features combined with statistical modeling. An example for such a methodology would be [HEH05a] [HEH05b], where simple assumptions like the existence of a ground floor, rectangular areas that stand on this ground, and a background sky is assumed. These methods are studied extensively and lead to useful results in some scenarios. However, they tend to do not generalize well since it is hard to construct a representation that is applicable in a broad range of scenarios.

An alternative to this concept is the nowadays dominating method to algorithmically extracting the feature representation from the data itself. Since the rise of CNNs (convolutional neural networks) [KSH12], these approaches dominate in the reign of image classification and steadily widen their rule to related topics like object detection, image segmentation, pose estimation (as discussed before) as well as depth estimation. In this thesis, we focus on learning-based approaches to estimate the depth of images. Zhao [ZSZ+20] gives a concise overview of different deep learning-based depth estimation techniques.

However, this mapping between a 3D world and its 2D projection also comes with challenges like choosing an adequate network architecture, acquiring a sufficient training dataset, and composing an appropriate loss function to converge to a robust model that generalizes well. Ranftl and Lasinger [RLH+19] take a novel approach to deal with these problems. They use a so-called Zero-shot Cross-dataset transfer method, which means training on a completely different dataset than testing. They argue that the biggest challenge in deep learning-based depth estimation is the generalization of the model, which is crucially dependent on the universality of the training data. There are already existing depth datasets like MegaDepth [LS18], Kitti [GLU12] and TUM [SEE+12] available, but all have their own intrinsic biases, are small, or only contain images from a particular scenario (for example only indoor images [KJMS18]). Ranftl and Lasinger [RLH+19] combine 6 of those benchmark datasets and further incorporate data from 3D movies to a vast mixed depth dataset.

The difficulty of mixing different data sources relies on the scale and shift inconsistency between the corresponding ground truths. Sometimes the ground truth is accumulated by stereo cameras (with known calibration or unknown), via laser scanners (which measure the absolute depth), or by a structured light sensor, like in the case of a Kinect [Zha12] camera. To overcome this cross dataset shift and scale ambiguity, they developed a novel scale- and shift-invariant loss (Eq.3.2). The first step of their method is normalizing the disparity information for the ground truth d and the predicted disparity d^* for each pixel.

$$\begin{gathered} t(d) = median(d), \quad s(d) = \frac{1}{M}\sum_{i=1}^{M} |d - t(d)| \\ \hat{d} = \frac{d - t(d)}{s(d)}, \quad \hat{d^*} = \frac{d^* - t(d^*)}{s(d^*)} \end{gathered} \tag{3.1}$$

Then they formulate a composite loss 3.2 which ensures to minimize the disparity difference between prediction and ground truth, this behaviour is ensured by the $\mathcal{L}_{ssi}$ loss (Eq.3.2). Additionally, they introduce a regularization loss term $\mathcal{L}_{reg}$, with the aim of minimizing the gradient of $R_i = \hat{d}_i - \hat{d}_i^*$ for different scale levels k. That ensures a continuous depth estimation map, exhibiting smooth spatial variations.

$$\begin{gathered} \mathcal{L}_l = \frac{1}{N_l}\sum_{n=1}^{N_l} \mathcal{L}_{ssi}(\hat{d}^n, (\hat{d}^*)^n) + \alpha \mathcal{L}_{reg}(\hat{d}^n, (\hat{d}^*)^n) \\ \mathcal{L}_{reg}(\hat{d}, \hat{d}^*) = \frac{1}{M}\sum_{k=1}^{K}\sum_{i=1}^{M}(|\nabla_x R_i^k| + |\nabla_y R_i^k|), \\ \mathcal{L}_{ssi}(\hat{d}, \hat{d}^*) = \frac{1}{2M}\sum_{i=1}^{M}(\hat{d}_i - \hat{d}_i^*)^2 \end{gathered} \tag{3.2}$$

Lastly, they separate the training procedure from testing, which means that their reported testing results are from datasets that were not part of any training step. The monocular depth estimation methodology of Ranftl and Lasinger [RLH+19] produces state-of-the-art on many benchmarks, such as Kitti [GLU12], which was not part of any training procedure. Therefore, we employ their MiDaS model as the first step in our geometric aware detection pipeline, for estimating the depth of a scene.

Figure 3.1 gives an impression of the quality of the depth estimation results from the MiDaS model for quantitative analysis of their model performance [RLH+19]. For all depth estimation steps later in this thesis, we use the pre-trained MiDaS model from Ranftl and Lasinger [RLH+19].

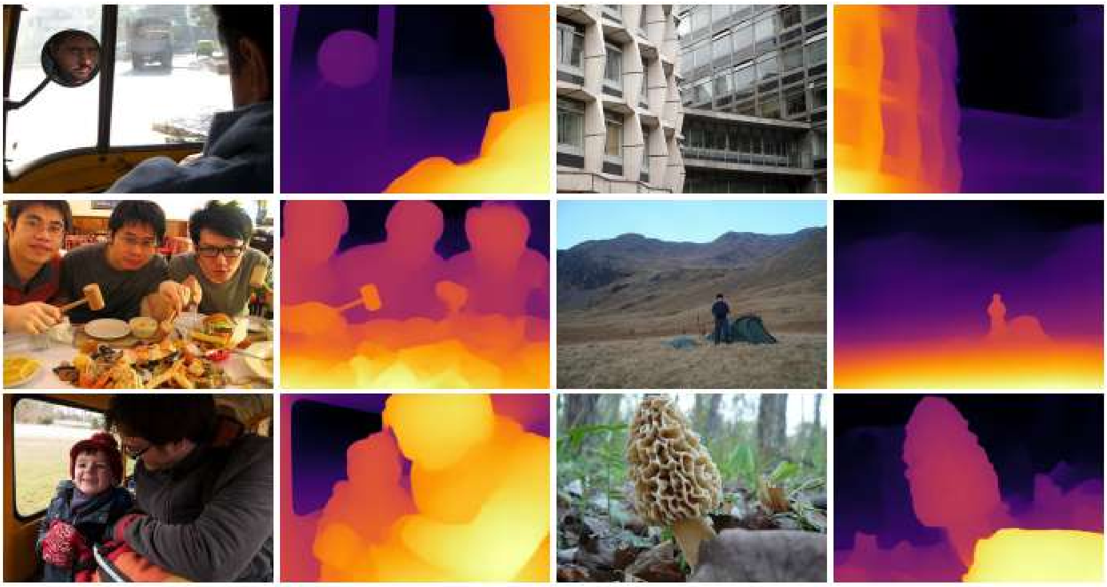


Figure 3.1: Qualitative results from MiDaS monodepth estimation [RLH+19]

3.2 Stereo Vision

Classical stereo vision is a well-established method for depth estimation, where you calculate the depth of objects from synchronous image pairs [BF82].
To achieve this, at least one known distance is required. In the case of stereo vision, this is typically the baseline between the two cameras. Additionally, the focal length for a camera setup is known, which allows us to compute the distance by simple trigonometry. Image 3.2 illustrates this setup, which is more commonly known as parallaxes.

To calculate the distance Z to the point of interest P, we first have to obtain the interocular distance b, which is known in the case of a stereo camera. Additionally, we have to calculate the shift d between the two images from the synchronous cameras 3.3.

$$\begin{aligned} d &= |x_l - x_r|, \\ b &= |C_1 - C_2| \end{aligned} \tag{3.3}$$

After deriving b and d, we can use the property of similar triangles to obtain the distance to point P by transforming the equation 3.4.

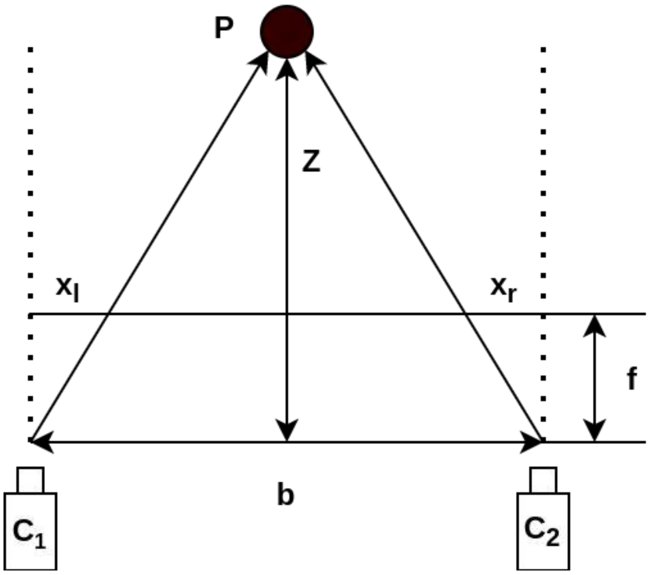


Figure 3.2: Illustration of the parallaxes for depth estimation in stereo vision

$$
\begin{aligned}
\frac{b-d}{Z-f} &= \frac{b}{Z}, \\
\implies Z &= \frac{bf}{d}
\end{aligned}
\tag{3.4}
$$

This methodology allows us to estimate the depth of objects but runs into difficulties when scenes include reflective surfaces, thin structures, or texture-less components. However, a disadvantage of the stereo vision approach is that the quality of depth estimates deteriorates with increasing target distance.
Modern binocular stereo algorithms successfully enrich the trigonometric approach with representational knowledge due to deep learning enhanced matching quality. These enhanced stereo-matching methods are the subject of the next section.

3.3 Learning-based Stereo-matching Methods

Learning-based methods for estimating pixel-wise disparity use deep neural networks to learn representations from data. Thus, machine learning enhanced stereo vision methods achieve better results, in situations where thin structures and textureless areas appear. The first end-to-end trainable architecture for disparity estimation from stereo images was DispNetC [MIH+16a]. They use a correlation layer to measure the similarity between left and right images.

Another common approach is 3D convolution, where the left and right feature representations are concatenated and afterward aggregated via a convolution operation. GC-Net [KMD+17] is an early example of the 3D convolution methodology. A more recent candidate of the 3D convolution approach is PSMNET [CC18], where they additionally use convolution for cost aggregation, which leads to state-of-the-art performance in disparity estimation. However, the 3D convolution methods come with the drawback of being computationally expensive, which leads to high deployment costs and being slow.
Therefore, Xu and Zhang [XZ20] propose a learning-based stereo-matching method without 3D convolution. Their so-called Adaptive Aggregation Network (AANet) uses a sparse point-based representation for intra-scale cost aggregation (Fig. 3.3).

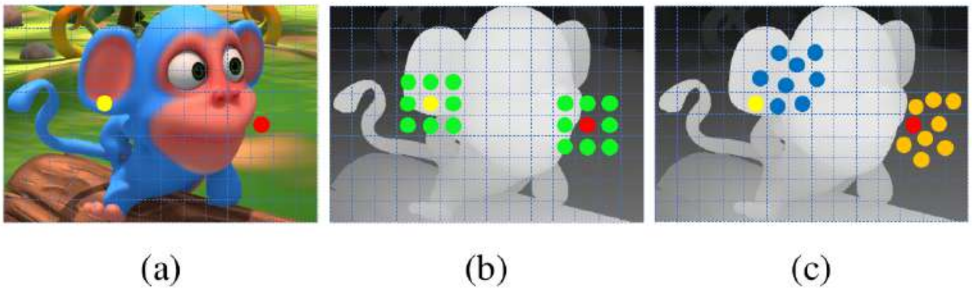


Figure 3.3: Illustration of sampled locations from the left image of a stereo pair (a). The middle image (b) shows classical convolution for aggregation. The right image (c) shows adaptive sampling locations via deformable convolution like in the case of the AANET [XZ20].

The main idea of the AANET [XZ20] depends on the combination of two complementary modules, an adaptive intra-scale cost aggregation (ISA) and an adaptive cross-scale cost aggregation (CSA).
The ISA module leverages a sparse point-based representation for flexible cost aggregation, closely related to deformable convolution to tackle the common issue of edge-fattening. On the other hand, the CSA module deals with low-texture regions, where aggregating information from different scales can be beneficial for estimating disparities.
These two modules combined result in a final aggregation module (AAModule), which is stacked on each other. The overall architecture of the AANET (Fig. 3.4) consists of two feature pyramids with shared weights to extract the features from the rectified stereo image pair. Afterward, those representations are correlated on corresponding scales to construct the multi-scale cost volumes. Further, the cost volumes are fed into the consecutive AAModules to calculate the final disparity prediction.

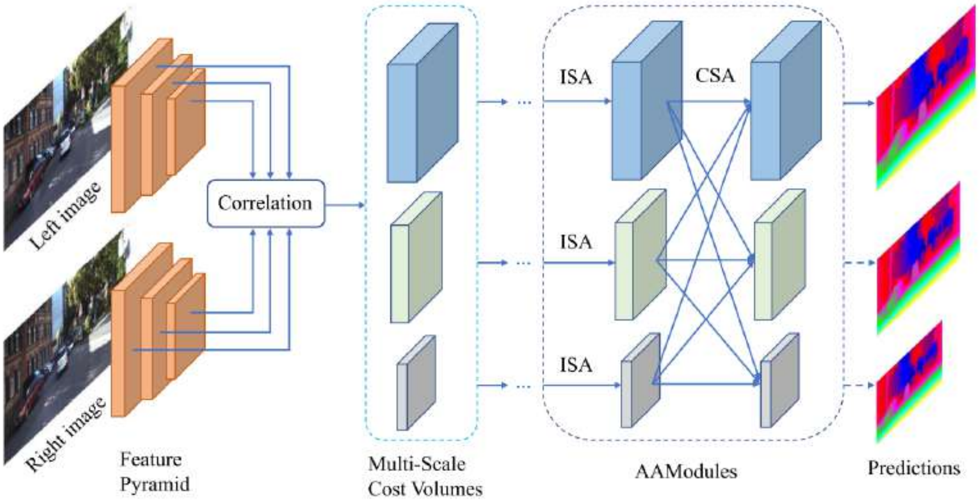


Figure 3.4: Illustration of the learning-based stereo-matching procedure of the AANET [XZ20]. Consisting of convolution-based shared feature extraction and subsequent correlation to construct the Multi-Scale Cost Volumes. Afterward, the cost volumes are used to estimate the pixel-wise disparity via the consecutive AAModules.

The ANNet is trained in an end-to-end manner and achieved state-of-art results on KITTI [GLU12] and, Scene Flow dataset [MIH+16b] while having a significantly lower runtime. Additionally, the AANET predicts a dense disparity map. Therefore, we employ a pretrained AANET as an alternative to the MiDaS-based monocular depth estimation to compare the influence of the different geometric representations.

CHAPTER 4

Proposed Methodology

In this section, we discuss the proposed method for geometric-aware pose estimation in the case of general geometric primitives, namely cuboids and their real-world pendants. Examples of such are containers, boxes, cupboards, and much more.

In the first section, we walk through the data synthesis process. This includes constructing the geometric representation via monocular depth estimation as well as learning-based stereo-matching methods. From the resulting dense depth maps, we calculate the surface normal representations. Afterward, the advantages of this procedure compared to standard appearance-based learning will be outlied.

In the next section, we compare the two different pose estimation strategies. The first strategy is an object-centered top-down approach, which means we directly detect the cuboid centers and regress the necessary attributes for pose construction. The second strategy is a bottom-up or part-based scheme, where we detect corners and construct the cuboid pose from those. In the case of the part-based approach, this includes an algorithmic cuboid construction post-processing step. Additionally, we will show the feasibility of our method for a palette. This entity is more interesting from an application-related point of view. Overall the cuboids and palettes are industry-relevant on their own, since detecting and estimating the pose of those is crucial in the scenario of an autonomous forklift, see Figure 4.1. The forklift is only one example, there are many more scenarios where cuboid shapes play a role in applications.

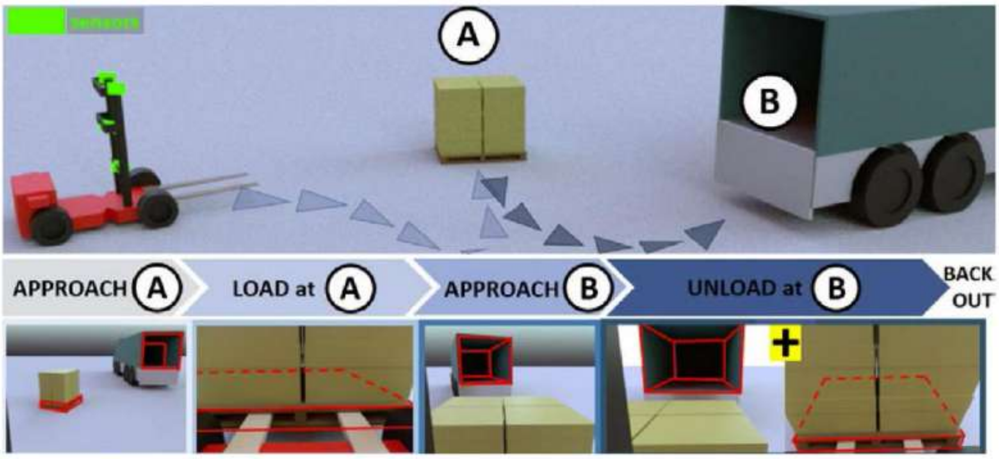


Figure 4.1: Visualization of an application scenario for the geometry based pose estimation

4.1 Data Generation and Employed Representations

In the domain of supervised learning, the task of collecting data and annotating the corresponding ground truth is crucial for success. The role of the data in such machine learning and particular deep-learning-driven experiments is at least equally important to the choice of the algorithm or the network architecture.

A key complexity of this matter is collecting and annotating a vast data-set, which is very labor-intensive and subsequent also cost-intensive. This approach is often only feasible for big institutions and especially for big internet companies.

Synthetic data is a more and more used alternative to real-world data since the corresponding ground truth is generated in an automated manner. To built a synthesis framework for generating the necessary training data, we use the open-source 3D animation software blender (`https://www.blender.org/`), which comes with an interface for python scripting. Blender allows us to write python scripts to procedurally generate training data, where we randomly alternate the conditions of the renderings.

Following properties are varied to generate a diverse set of appearance, view geometry, and scene photometric conditions:

- the number of objects
- the size and shape of the objects (in case of the cuboid)
- the object textures
- the positioning of the objects
- the camera angle
- the distance of the camera to the objects
- the positioning of the light source

The randomization of those factors is necessary to mimic the diversity of real-world situations. Figure 4.2 shows a few synthetic training samples from the blender pipeline.

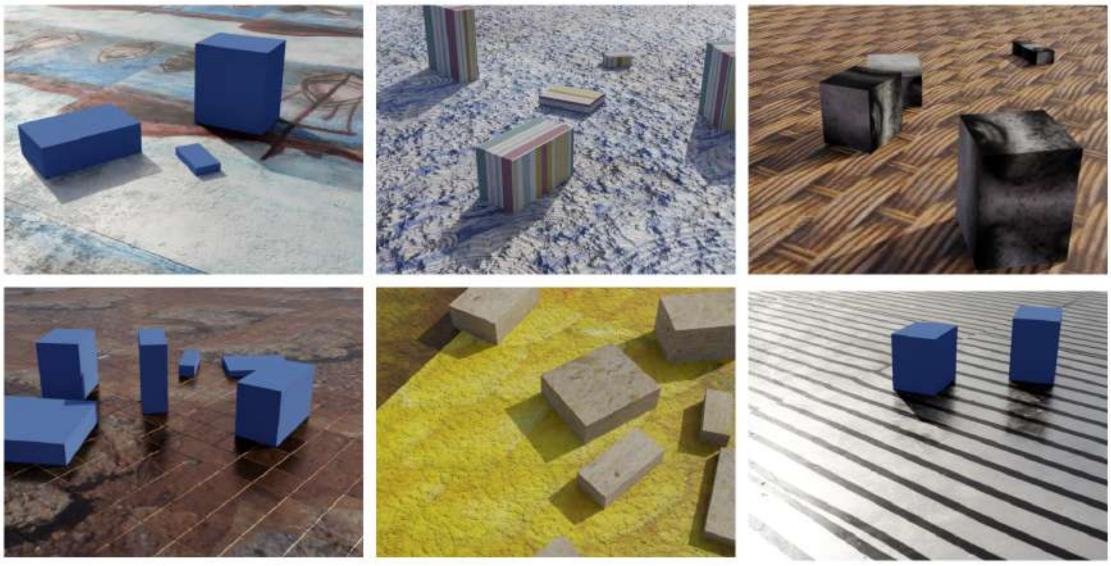


Figure 4.2: A few examples of the procedural generated training data for cuboid detection and pose estimation

However also the usage of synthetic data comes with its drawbacks. The synthetic data can reduce the generalization ability of a model on real-world domains. The issue depends on the vast amount of different appearance configurations a real-world object could have (Fig. 4.3).

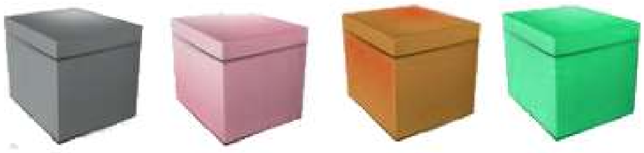


Figure 4.3: The same object can exhibit a large space of appearance variations, but can also be described by one distinct geometry

If we want to train a model on the raw renderings, we have to consider every possible appearance configuration during data synthesis. In simple terms, if the training set only includes green and blue cuboids, the detector could not detect red cuboids during inference mode. Since to resolve this issue, we have to know all possible appearance configurations beforehand, which would lead us to a closed-world assumption. In our general-purpose cuboid detection scenario, considering all appearance modalities is not feasible. The number of possible cuboid colors and textures in the real world is practically spoken unlimited. We deal with this issue by focusing on the geometric object properties instead of the appearances. We achieve this via the representation of the data. Thus, we use pre-trained depth estimation networks to estimate the disparity of each pixel in the rendered images. We apply MiDaS [RLH+19] for monocular depth estimation and AANET [XZ20] for stereoscopic depth estimation. Afterward, we calculate with the Sobel operator the surface normal representation from the predicted relative depth maps. We map the surface normals to RGB channels to simplify learning on established appearance-based concepts. In this way, we get rid of the appearance information, which forces the model to focus on geometric properties for decision-making during training.

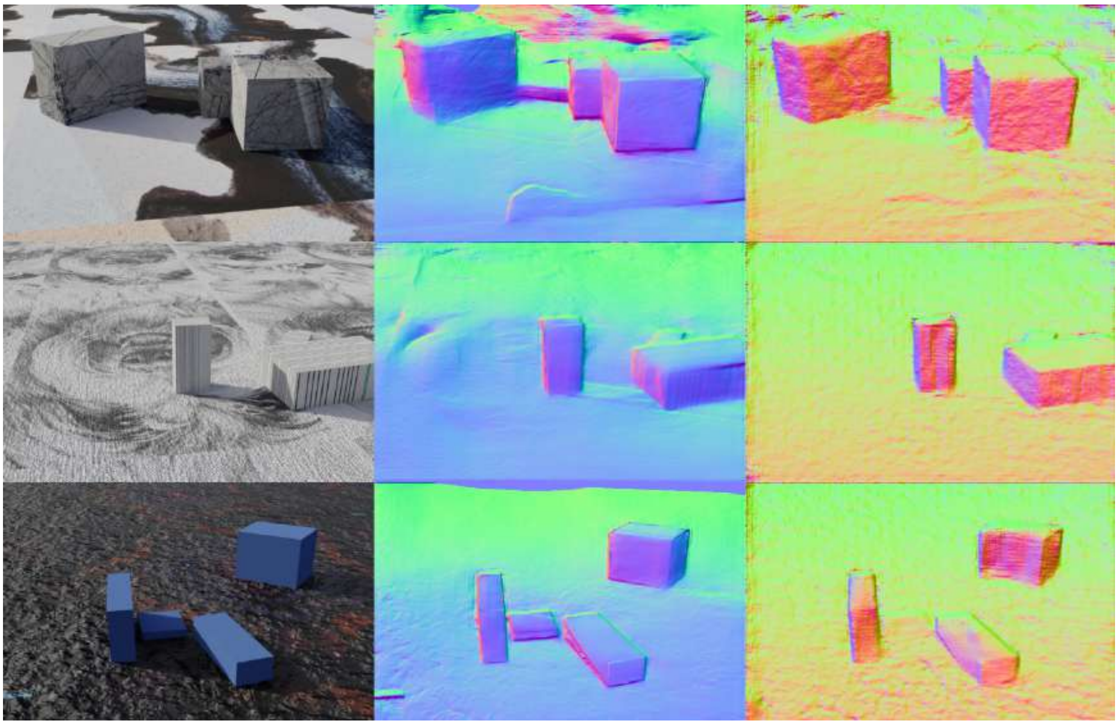


Figure 4.4: The left image shows the left part of the rectified stereo image pair from the synthetically generated training samples. The image in the middle shows the via MiDaS [RLH+19] estimated monocular surface normals. The right image shows the via AANET [XZ20] estimated stereoscopic surface normals.

From Figure 4.4 we can obtain that the estimated surface normals contain less complexity compared to RGB images. The less complex input data simplifies the detection task and leads to faster convergence of the loss. Therefore, we reduce the number of necessary training samples. Another advantage of the geometric transformation of the renderings is the containment of imperfections and noise due to depth estimation errors. The roughness of the depth estimation helps to narrow the gap between the synthetic and real-world images. Figure 4.5 illustrates the geometric-representation induced convergence between artificial and real-world domains.

We can clearly distinguish the real from the synthetic scene on the right (Fig. 4.5). However, after the geometric transformation, this is not the case anymore. It is barely possible to decide which picture is a real-world sample and which is not. The visual

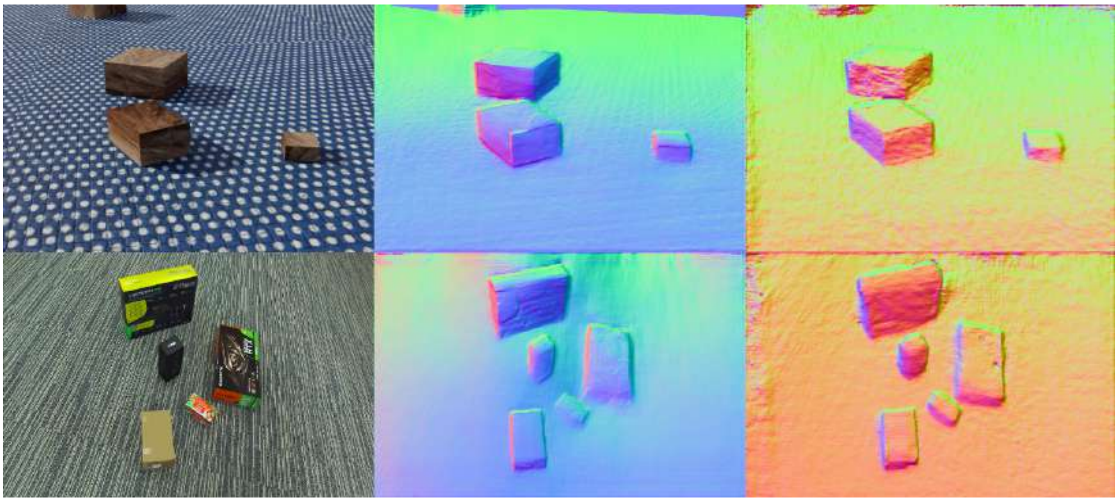


Figure 4.5: The upper RGB image is a synthetic sample from the blender pipeline, with the corresponding estimated surface normal representations from MiDaS [RLH+19] in the middle and AANET [XZ20] at the right. The lower image is a real-world image, with the corresponding surface normal estimations.

convergence of the images is highly promising for generalization. A detector that is built on such synthetic geometric data should be able to work also on real-world images. Due to this data synthesis pipeline, we can generate the corresponding ground truth in an automated manner and store it in YAML files. The ground truth information consists of:

- cuboid center position
- cuboid corner positions
- an integer to store the matching between corners and center

The data generation Blender pipeline allowed us to create a vast training set with around 70.000 annotated samples. It would not have been feasible for us to collect and annotate such a big data set from real-world samples.

4.2 Detection and Pose Estimation

In this section, we discuss two complementary approaches to construct cuboid poses from synthetic surface normal images with corresponding ground truth data. Figure 4.6 gives an overview of the experiments.

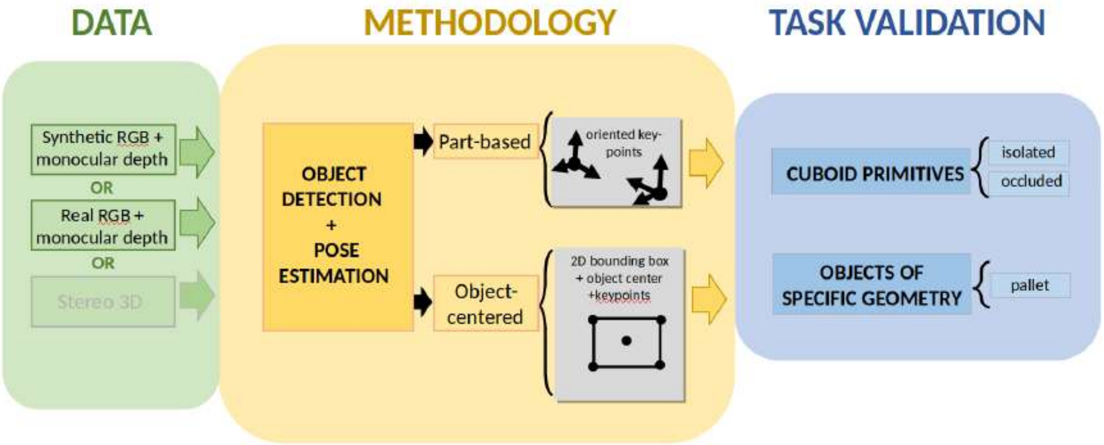

DATA
METHODOLOGY
TASK VALIDATION
Synthetic RGB + monocular depth
OR
Real RGB + monocular depth
OR
Stereo 3D
OBJECT DETECTION + POSE ESTIMATION
Part-based
oriented key-points
Object-centered
2D bounding box + object center +keypoints
CUBOID PRIMITIVES
isolated
occluded
OBJECTS OF SPECIFIC GEOMETRY
pallet

Figure 4.6: Overview of the proposed methodologies for object pose estimation

Both methods are closely related to the CenterNet [ZWK19] detection scheme. We use a single encoder-decoder backbone, with multiple output heads. The main head outputs a heatmap with Gaussian probabilities for the object centers. However, the output could also be the center of an object part, leading to a part-based bottom-up proposal generation scheme. Additional heads allow us to regress key-point attributes corresponding to the centers. To obtain the associated key-point attribute we take the value from the secondary head at the same position as the maximum of the corresponding main head. This is visualized in Figure 4.7.

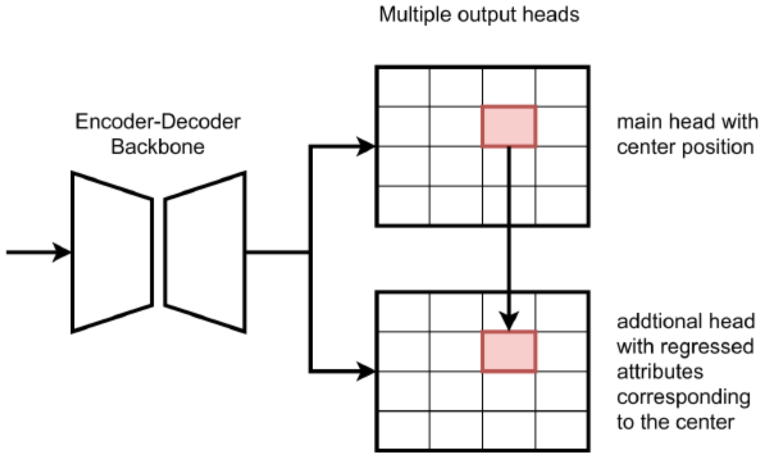

Multiple output heads
Encoder-Decoder Backbone
main head with center position
addtional head with regressed attributes corresponding to the center

Figure 4.7: Illustration of the multiple head based detection

The model is trained end-to-end and does not need any intermediary steps. The training loss is composite $\mathcal{L}$, which is calculated like:

$$\mathcal{L} = \sum_{h}^{H} w_h \mathcal{L}_h \tag{4.1}$$

It is a weighted sum of separate loss terms for each output head $\mathcal{L}_h$. This linear combination allows emphasizing the attributes on which the detector focuses during training. For example, if you increase the weight for an attribute type, the corresponding loss term will be more influential on the overall loss. The two complementary detection schemes are:

- top-down object-centered method
- bottom-up part-based method

In the case of the top-down object-centered detection scheme, we define the cuboid or palette centers as the output of the main head. From there we regress the corner locations which is the output of a second head.
The second method is a bottom-up part-based detection scheme, where we represent the individual cuboid corner location as centers and encode neighboring corner location vectors as regressed attributes. So instead of cuboids, we detect corners from which we construct the cuboid poses. Both methodologies are discussed in detail in the following sections.

4.2.1 Object-centered Detection Scheme

In the case of the object-centered approach, we focus on detecting the cuboid centers and regress from their corresponding cuboid corners as key-points (Fig. 4.8).

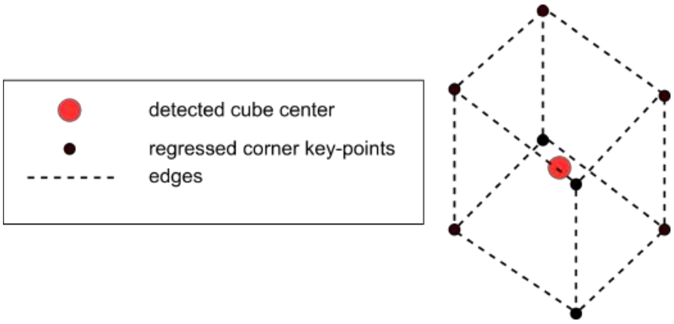


Figure 4.8: Illustration of object centered detection method

This method is a top-down detection process since we know beforehand the type of objects we are looking for, namely cuboids and palettes. This procedure resembles the human pose estimation method of Law [ZWK19]. They detect the centers of humans and regress from their joints to construct the pose of the human skeleton. Figure 4.9 illustrates the output heads of the CenterNet human pose estimation.

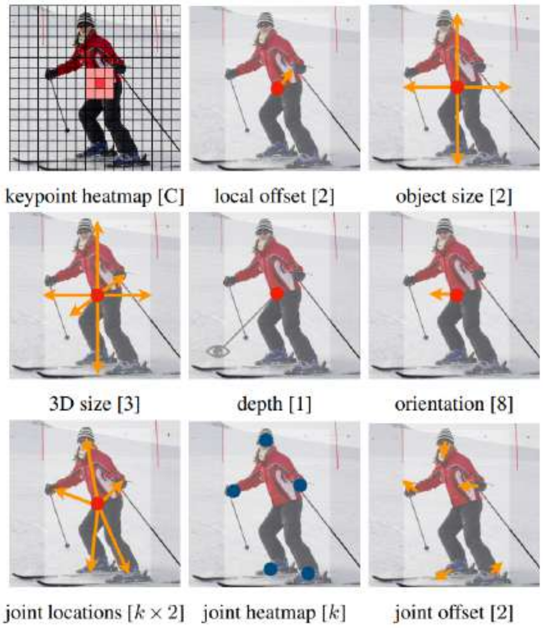


Figure 4.9: Output heatmaps for appearance-based human pose estimation from [ZWK19]

In the case of Law [ZWK19] the main head shows the Gaussian probability distributions for the human center and the secondary heads show the additional joints and offsets to

construct the pose.
For the cuboids, the output heads reduce to two, the main head for the cuboid center probability density and the second head for the corner coordinates. In conclusion, the weighted composite loss formulates like the following:

$$\mathcal{L} = \alpha\mathcal{L}_{cent} + \beta\mathcal{L}_{corn} \tag{4.2}$$

Where $\mathcal{L}_{cent}$ is a focal loss term for the cuboid centers and $\mathcal{L}_{corn}$ is an euclidean L2 norm loss. For the object-centered experiment we chose to weight $\alpha = \beta = 1.0$

4.2.2 Part-based Detection Scheme

The idea of the part-based detection methodology is to divide the cuboids into their basic geometric units and detect those. We chose to train the detector on the cuboid corners since those are easily identifiable in the image surface normal representation and are an even more general geometric primitive than the cuboid itself. The corner representation consists of the actual location which is encoded in the main heatmap. In the additional heatmaps, we regress the corresponding edge vectors. Each edge vector is defined as $(\sin\theta, \cos\theta, |e|)$, where $|e|$ represents the length of the edge in the 2D image. θ denotes the edge orientation with respect to the horizontal direction. The edge vector indices are sorted according to increasing orientation angles. The sorting step is necessary to unambiguously compare and score possible matching edge structures during training. The corner representation with corresponding edges is visualized in figure 4.10.

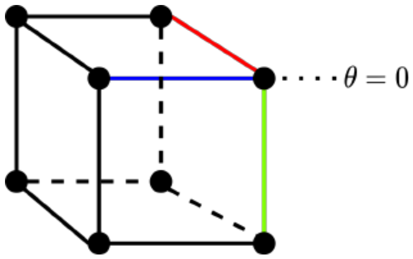


Figure 4.10: Illustration of the cuboid corner representation. Green-, red-, blue-colored edges represent edge orientations with increasing θ values

We train the model with a composite loss consisting of three terms. The main part is a focal loss term $\mathcal{L}_k$, which ensures that the corner positioning is right. Additionally, we use an angular loss $\mathcal{L}_a$, which is simply a regularized L1 norm of the $\sin\theta$ and $\cos\theta$ terms. We also introduce a length-loss term $\mathcal{L}_l$, in form of a regularized L1 norm of the edge length difference. The overall loss term is the following:

$$\mathcal{L} = \alpha\mathcal{L}_k + \beta\mathcal{L}_a + \gamma\mathcal{L}_l \tag{4.3}$$

Those composite loss terms are balanced by weighting factors α, β, and γ to ensure an optimal learning objective. Our experiments lead to $\alpha = 1.0$, $\beta = 0.8$ and $\gamma = 0.6$ to achieve the best detection results. We use this parameter setting throughout the part-based experiments in this thesis.

In a post-processing step, we construct the cuboids from the detected corners and edges. To achieve this we designed a center-voting based cuboid construction algorithm (Alg. 4.1). The algorithm starts with the output of the detector, which consists of the corner locations *det_corn*, the angle representation *ang_kps*, the edge lengths *len_kps*, and the detection confidence scores *scores_kps*. With those measures we can construct the hypothetical corners *est_corn*, this happens in line 1 of the algorithm. Afterward, we calculate the euclidean distance between the detected and the estimated corners *dist_m* and keep track of the edge distance correspondence *edge_m*. Then we initialize lists for the final constructed cubes, the center coordinates, and the scores (line 3 of the pseudo-code). The main loop runs until the Euclidean distance between the position of the closest corner estimate and corner detection *dist* reaches the hyper-parameter *dist_max*. Based on image structural co-linearity, we hypothesize a possible linked corner pair (line 7 of the pseudo-code) and set the distance between the chosen corners to infinity in the adjacency matrix *dist_m*. Two associated corners are enough to generate six image points, which are used in a camera re-sectioning step to estimate object centers in the 3D space. Those coordinates are projected back into the 2D plane and allow us to obtain eight cuboid corners and the center (line 8 of the pseudo-code). The corner pair cuboid hypothesizing is visualized in figure 4.11.

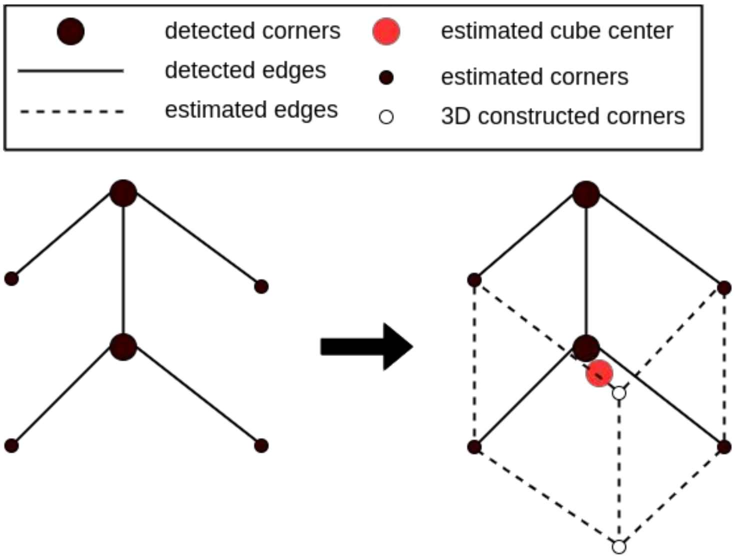


Figure 4.11: Illustration of the center-voting-based cuboid construction algorithm

Afterward, we check if the cuboid was already proposed (line 8), we achieve this by calculating the euclidean distance between the new center position with the proposed center coordinates. We choose the closest if the distance is lower than the threshold

center_thresh. In that case, we sum $\frac{1}{dist}$ to the score for this particular cuboid.
If the center does not coincide with an already counted one, we append the cube to the cubes list. Additionally, we append the center and initialize a corresponding detection score (lines 14, 15, and 16). The detection score is inverse to the distance between detected and estimated corner, which resembles the confidence of the detection in the score.

Lastly, we check for all proposed cuboids if their score reaches the final threshold *thresh*.

Algorithm 4.1: cuboid center voting algorithm

```
Input: $det\_corn$, $ang\_kps$, $len\_kps$, $scores\_kps$ from the detector and the
       additional hyper-parameters $det\_thresh$, $dist\_max$, $center\_thresh$,
       $thresh$
Output: $cubes$ which is a list of ordered coordinates for the eight cuboid corners
        and the corresponding center coordinates $center$
$est\_corn \leftarrow$
  $calc\_est\_corners(det\_corn, ang\_kps, len\_kps, scores\_kps, det\_thresh)$;
$dist\_m, edge\_m \leftarrow calc\_dist\_det\_est(det\_corn, est\_corn)$;
$centers, cubes, scores \leftarrow \emptyset$;
$dist \leftarrow 0$;
while $dist < dist\_max$ do
    $dist \leftarrow min(dist\_m)$;
    $cube\_est, inpindx, dist\_m \leftarrow$
      $create\_cube\_est(det\_corn, est\_corn, dist\_m, edge\_m)$;
    $cube\_2D, center\_2D \leftarrow construct\_and\_project(cube\_est, inpindx)$;
    $is\_counted, idx \leftarrow$
      $check\_if\_cuboid\_counted(centers, center\_2D, center\_thresh)$;
    if $is\_counted$ then
        $scores[idx] += \frac{1}{dist}$;
    end
    else
        $cubes.append(cube\_2D)$;
        $centers.append(center\_2D)$;
        $scores.append(\frac{1}{dist})$;
    end
end
for $cube, score$ $in$ $zip(cubes, scores)$ do
    if $score > thresh$ then
        keep $cube$;
    end
    else
        reject $cube$;
    end
end
Result: cubes,centers
```

The threshold parameter allows configuring the sensitivity of the detector without re-training the model. So a low threshold results in a high recall, which means most cuboids will be detected, but there will also happen wrong detections (False Positives). On the other hand, a high threshold will result in high precision, which means that detected objects will be most likely cuboids (low False Positive rate), but the detector will also tend to miss some True Positives.
Overall the part-based detection combined with the center-voting cuboid construction algorithm allows us to construct the 2D projected pose of the highly symmetric cuboids. Figure 4.12 gives some qualitative impressions of the predicted poses on real-world images.

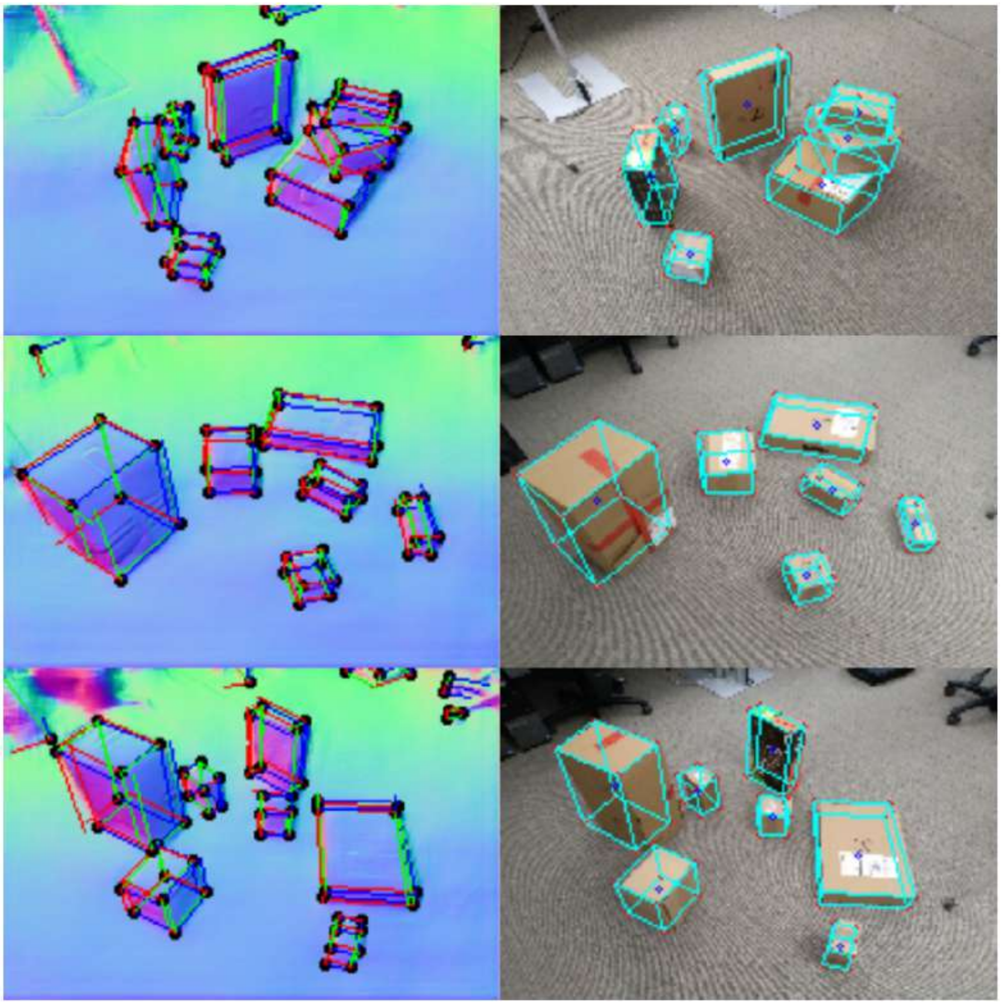


Figure 4.12: The left images show the output of the detector, with the detected corners and edges drawn on the surface normal representation. The right images show the back-projected 3D pose of the center-voting post processing algorithm on real world images

4.3 Summary

In summary, the object-centered top-down and the part-based bottom-up detection schemes complement each other. The object-centered approach starts with detecting the object and regresses the associated key points from the center. To some extent, this is the most obvious way to tackle the detection problem. On the other hand, the bottom-up part-based approach localizes the object corners with their corresponding edge orientations. This is an even more general detection approach, which can be applied to all kinds of rectangular and boxy objects. However, the part-based approach requires a post-processing step to associate the corners for constructing the object poses. We achieve this via the center-voting cuboid construction algorithm, but this is only applicable for cuboids. Therefore the post-processing limits the generality of the part-based detection scheme and requires a redesign of the construction algorithm for other rectangular objects.

CHAPTER 5

Results and Discussion

In this section, we discuss the results of the object-centered and the part-based experiments. We focus on evaluating the geometric aware cuboid detection since the cuboids are geometric primitives with many real-world pendants and all kinds of possible appearance configurations. Further, the cuboids have a clear and distinct geometry, which makes them easy to recognize in the surface normal images. Additionally, many objects from the man-made world are cuboids, or cuboid composites, which increases the practical relevance of detecting those. In conclusion, cuboids are the optimal target for testing the detectors' ability to overcome appearance dependency and focusing on the geometric object properties. Lastly, we will also show qualitative results for a palette, to illustrate the feasibility and the generality of the geometric aware pose estimation.

5.1 Overview of Experiments

We start with the object-centered experiment results and conclude with the part-based experiment results.
To evaluate the estimated cuboid poses we choose the Intersection over Union (IoU) metric. This is a common metric for evaluating 2D detectors, where we calculate the ratio between overlap and the union of the convex hulls (Fig. 5.1).

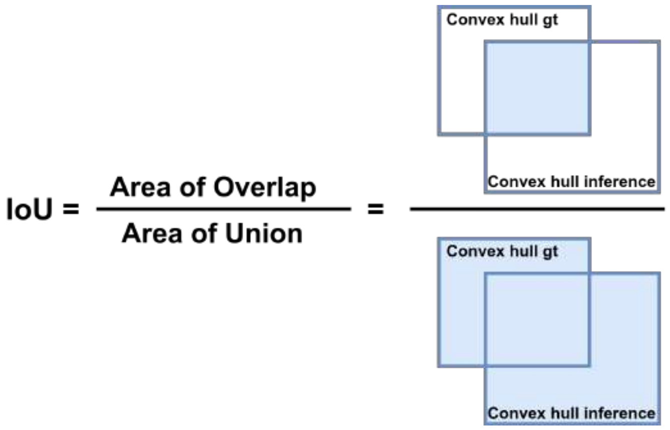


Figure 5.1: Visualization of the Intersection of Union score

The IoU is not a 3D object detection metric, which would be desirable, especially in robotic scenarios. However, we only have relative depth information due to the monocular depth estimation, so we can not calculate the distance to the objects. Additionally, we do not have any prior knowledge about the size of the cuboids, since we tackle a semi-open world scenario. We want to detect all cuboid-like objects, which makes the 3D detection significantly more difficult than detecting a static set of prior known specific objects. In the case of the stereoscopy-based depth estimations, we have access to absolute distance measurements. However, to achieve comparability between the detection results from both geometric representations, we rely on the 2D projection of the 3D poses. All models are trained solely on synthetic data from our blender synthesis pipeline. The training data-set consists of around 70.000 annotated cuboid samples. On the other hand, we test our pose estimation pipelines on real-world images. Since we are not aware of a cuboid detection benchmark data set with IoU ground truth metric, we decided to create our own test sets. One set consists of 50 single camera RGB images of real-world boxes in occluded and cluttered scenarios. We use this set to test and fine-tune the monocular depth estimation pipeline. Additionally, we took 28 images with a ZED-2 stereo camera (`https://www.stereolabs.com/zed-2/`) to compare the pose estimation results from both, the stereoscopy-based and the monocular-based geometric representations. In both cases, we took the images our-self, and also the annotation of the ground truth was done by us (Fig. 5.2).

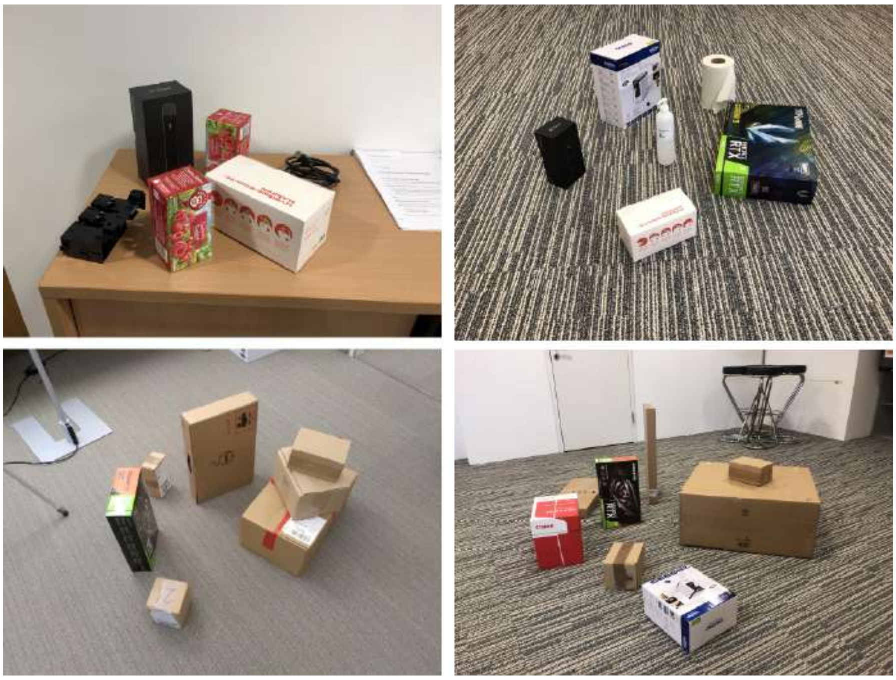


Figure 5.2: A few samples of the hand annotated real world test data set

5.2 Object-centered Experiment

Figure 5.3 visualizes the overall pipeline for the object-centered experiment. It consists of a pre-processing step, where we estimate the disparity via the pre-trained monocular depth estimation model [RLH$^+$19]. Afterward, we calculate the surface normal representation, which serves as the input for the pose estimation model. The detector outputs the object center heatmap in the main head and the corresponding corners in the secondary head. During the post-processing, we construct the wire-frame model from the object center and the corresponding key points.

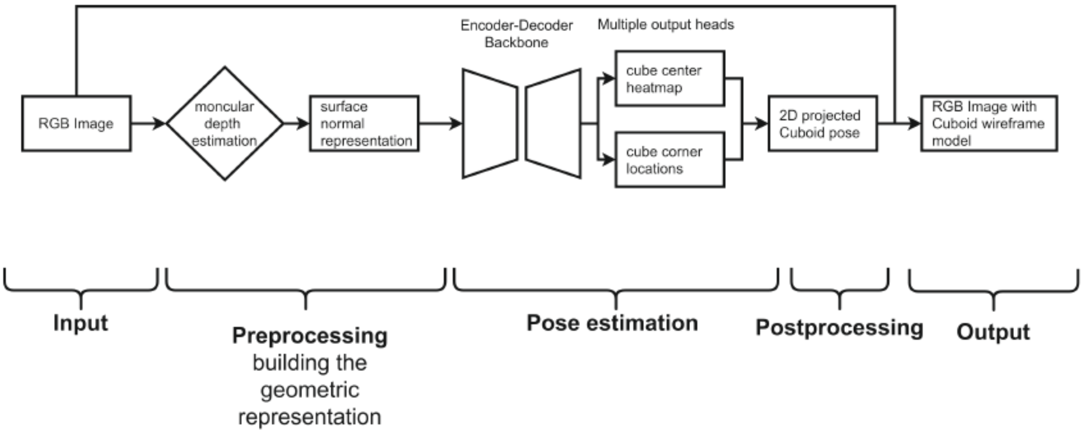


Figure 5.3: Visualization of the object centered detection pipeline

The simplicity of this method comes with significant drawbacks since the key-point regression process demands distinguishable and therefore uniquely identifiable key-point attributes. However, for the cuboid-shaped objects, this is not the case, their corners are not uniquely identifiable. We tried to come up with a feasible enumeration strategy for the corners. For example, we ranked the corners by the Euclidean distance from the left upper image border. But the highly symmetric cuboids are not suitable for this kind of approach. Figure 5.4 shows some inference result of the trained object-centered detector. On the right side of the images are the output heatmaps of the multiple detector heads. The upper heatmap with the red dots indicates the Gaussian probabilities for the object center positions, which appear promising. The right lower heatmap shows the probabilities for the positions of the distinct corners. Every color indicates one kind of corner.

Since the enumeration of the corners is not coherent enough for the detector, the training gets stuck in a local minimum where the corner positions are grouped in the middle or at one of the sites for lengthy shaped boxes.
For the less symmetric palettes, the detection scheme works, since the key points are distinguishable. Thus, we do not encounter wrong penalizations during training. The palette results (Fig. 5.5) qualitatively underline this claims.

Overall the object-centered results are surprising since we thought that the pose estimation should be easier for cuboids compared to more geometrically complicated structures, like the pallet. The opposite is the case. However, the detection of the palette indicates the potential feasibility for this geometric-aware center-focused pose estimation scheme for all kinds of application relevant objects. Still, in the case of cuboids we have to consider a different strategy for estimating the pose, this is part of the next experiment subsection.

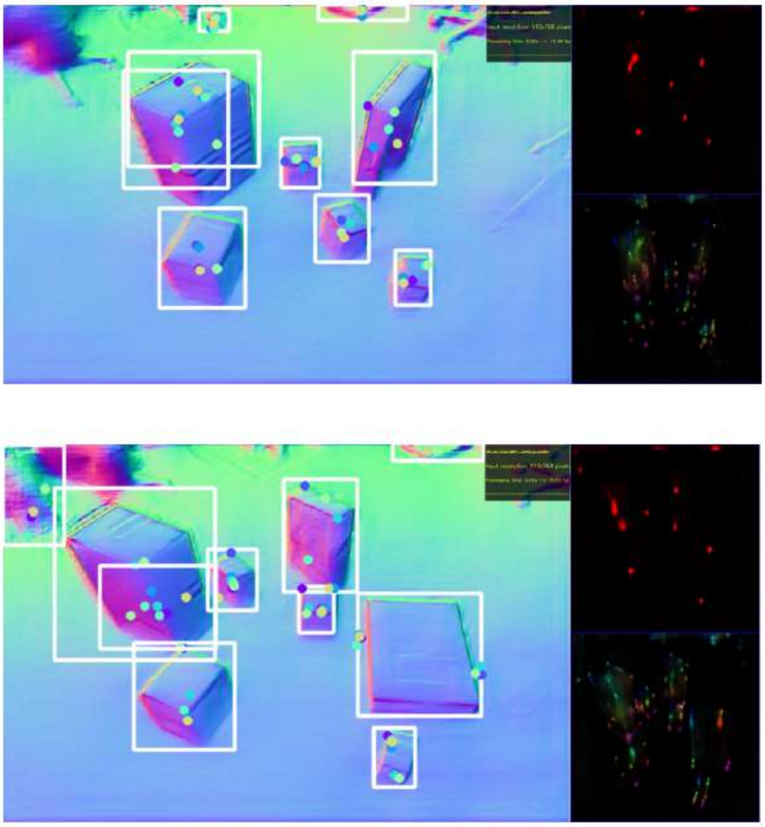


Figure 5.4: The left images show the surface normal representations of real-world input images, with estimated bounding boxes and corresponding key points. The left upper heatmaps visualize the cuboid center estimates. The left lower heatmaps show the cuboid key point estimates.

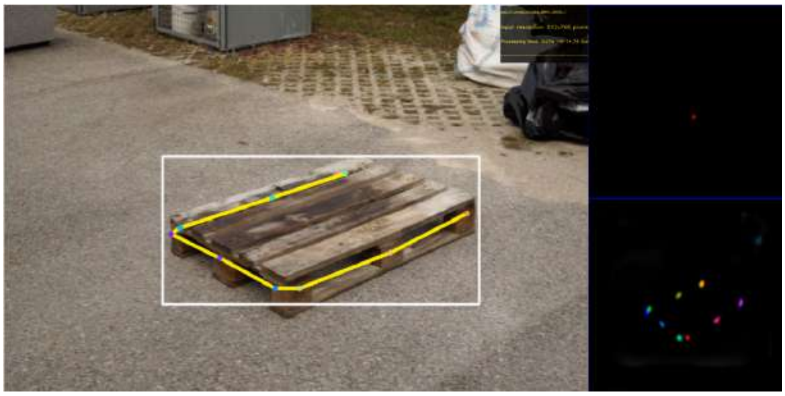


Figure 5.5: The left image shows a real-world input image, with the estimated bounding boxes and corresponding palette key points. The left upper heatmap visualizes the palette center estimate. The left lower heatmap shows the palette key point estimates.

5.3 Part-based Experiment

In this section, we discuss the part-based experiments for geometric-aware cuboid detection. Figure 5.6 illustrates the part-based detection pipeline. We compare monocular and stereoscopic depth estimation methods to construct the surface normals. The resulting geometric representations of the data serve as input for the encoder-decoder-based pose estimation network.

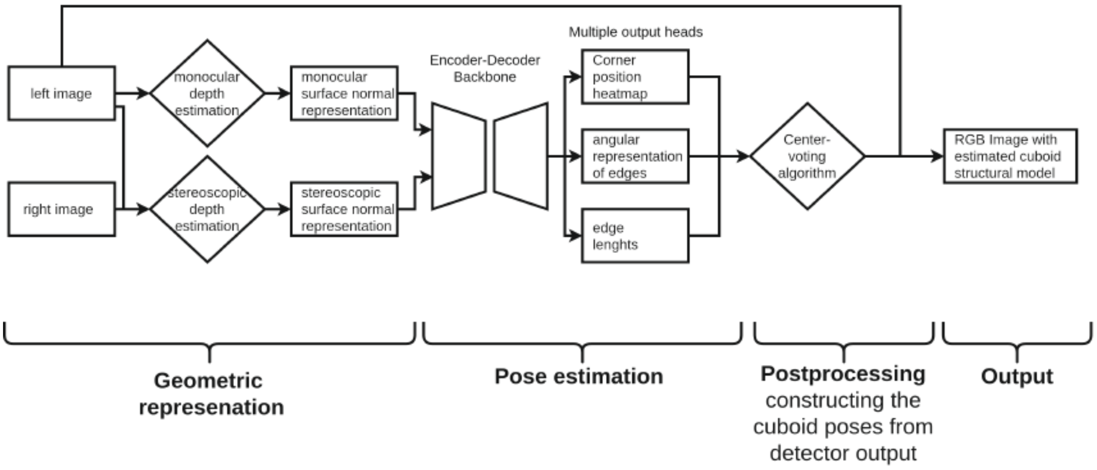
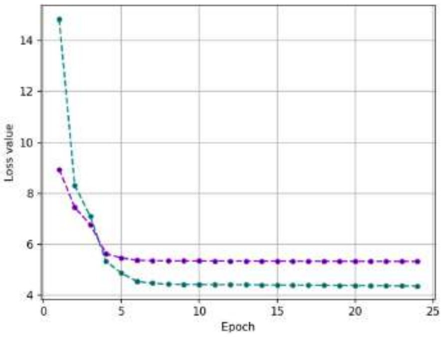

Multiple output heads
left image
right image
monocular depth estimation
stereoscopic depth estimation
monocular surface normal representation
stereoscopic surface normal representation
Encoder-Decoder Backbone
Corner position heatmap
angular representation of edges
edge lenghts
Center-voting algorithm
RGB Image with estimated cuboid structural model
Geometric represenation
Pose estimation
Postprocessing
constructing the cuboid poses from detector output
Output

Figure 5.6: Overview of the part-based cuboid detection pipeline. The geometric representation is created via monocular or stereoscopic depth estimation.

The detector has three output heads, the main head shows the corner position probability-density heatmap. The secondary two heads give us the edge representation ($\sin\theta$, $\cos\theta$, $|e|$). Those outputs serve as inputs for the center-voting cuboid construction algorithm (Alg. 4.1). The depth estimation steps simplify the training procedure due to the reduced complexity of the detector input. Figure 5.9 support this claim since the loss converges after only six epochs for both experiments.

We train the monocular depth-based detector and the stereoscopic depth-based detector on the same composite loss weighting terms. The corner positioning gets a weighting of $\alpha = 1.0$, the angular representation of the corresponding edges $\beta = 0.8$, and the edge length associated loss term $\gamma = 0.6$.
After the detection, we have to choose adequate hyperparameters for the cuboid construction post-processing. Therefore, we perform a grid search on a separate real-world test set consisting of 50 hand-annotated images. We measure the influence of each cuboid-construction parameter on the overall IoU score (Fig. 5.14).

The results indicate that 0.4 is optimum for the detection_thresh. This parameter influences the sensitivity of the detector. If the detection_thresh is high, we only accept corners where the model assigned a high probability. Further, we can obtain that 0.2 is optimum for the center-voting threshold. The center-voting parameter controls how many



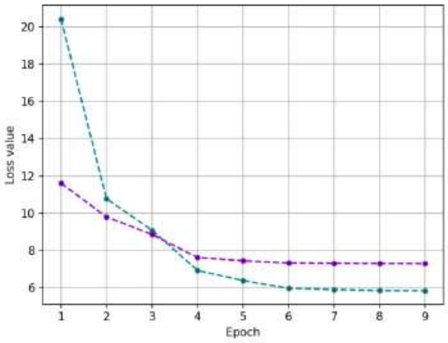


Figure 5.7: train and validation loss of the monocular-based experiment

Figure 5.8: train and validation loss of the stereoscopy-based experiment

Figure 5.9: The turquoise lines are the training losses and the purple lines are the validation losses, in the monocular and the stereoscopy-based experiments the loss converges after around 6 training epochs.

corner pairs have to vote for a center. Six cuboid points from a corner pair take part in a camera resectioning process to estimate a center. Camera resectioning is a geometric camera calibration process, where we estimate the camera matrix via 2D projection and 3D model correspondence. The optimal value for the other two hyperparameters is not that obvious. Therefore we choose the overall best parameter settings from the grid search (tab. 5.1).

thresh_detection	thresh_dist	voting_thresh	same_center_thresh	IoU
0.4	10.0	0.2	0.20	0.695
0.4	15.0	0.2	0.25	0.692
0.4	10.0	0.2	0.25	0.690
0.4	10.0	0.2	0.30	0.689
0.4	15.0	0.2	0.20	0.689

Table 5.1: Results of the grid search for tuning the threshold hyperparameters of the center-voting cuboid construction algorithm

Figure 5.15 shows the results of the best parameter setting for the monocular depth-based bottom-up cuboid pose estimation pipeline. From figure 5.15 we can obtain that the detector sometimes hallucinates corners that are not there, like in the case of the paper roll at the bottom image. But the cuboid construction algorithm, with the center-voting threshold, prevents the system from constructing a wrong cuboid wire-frame. The first image also shows that the detection framework is in principle able to deal with partial occlusions, but this does not always work like for the white box at the back.

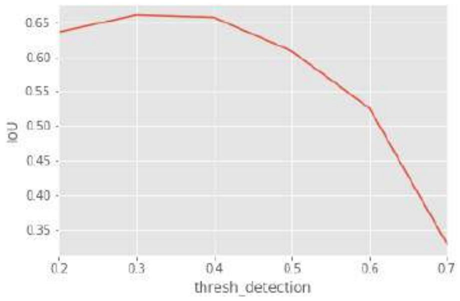


Figure 5.10: Detection threshold

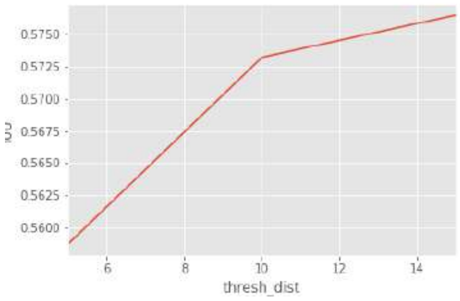


Figure 5.11: Cuboid distance threshold

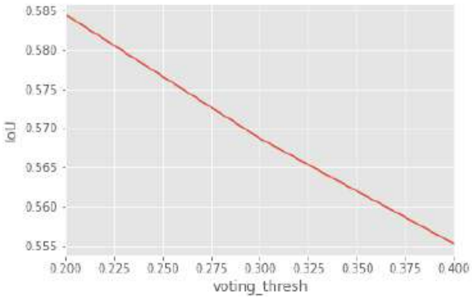


Figure 5.12: Center voting threshold

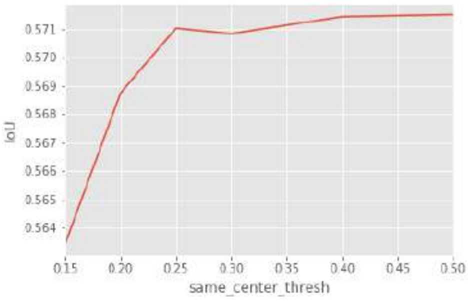


Figure 5.13: Same center threshold

Figure 5.14: Fine tuning results of the center-voting cuboid construction algorithm hyperparameters

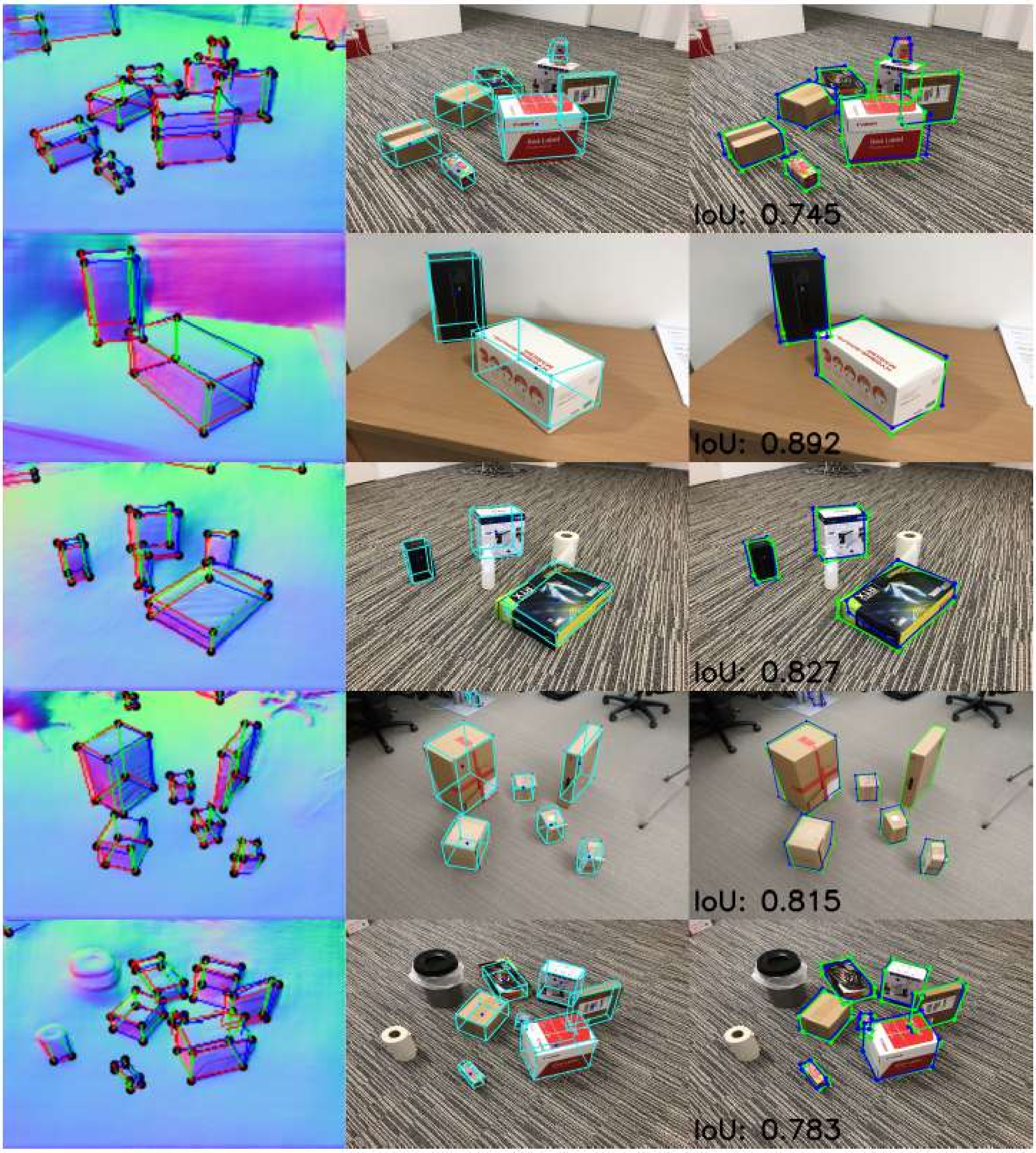
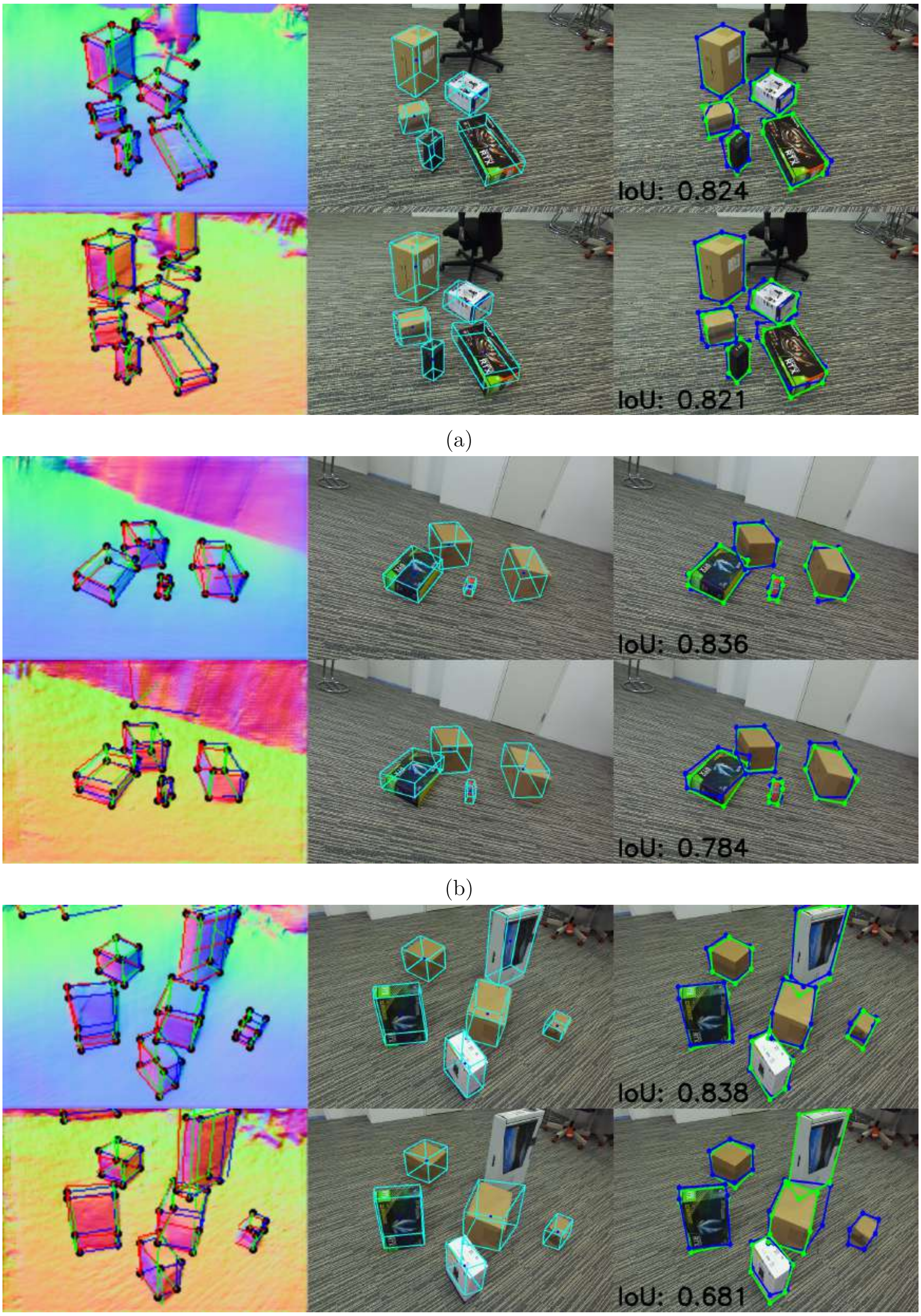


Figure 5.15: A few samples from the final tests for the part-based detection scheme, with detector output left, constructed cuboids in the middle and the convex hulls with corresponding IoU on the right

For the final evaluation, we use the via grid search obtained parameter settings. The test set consists of 28 real-world stereo image pairs, which we took with a ZED-2 stereo camera (`https://www.stereolabs.com/zed-2/`). We use the left image as input for the monocular cuboid pose estimation pipeline and the image pair for the stereoscopic cuboid pose estimation pipeline. Figures 5.16, 5.17 & 5.18 show the outputs of both experiments. The upper images show the monocular, and the lower images the stereoscopic results. Both pipelines construct similar cuboid poses, independent of the underlying geometric representation. Figure 5.17(a) is an example of the importance of the center-voting post-processing, since also corners of the chair and background are detected, but still do not lead to wrong cuboid proposals. Figure 5.17(c) shows that the part-based detection scheme also detects corners of the room. Further experiments could allow the construction of a geometric representation of the room itself. The overall IoU scores of both experiments (tab.5.2) show that the stereoscopy-based cuboid pose estimation pipeline is more accurate than the monocular based. On the other hand, the monocular counterpart has a significant run time advantage and does not require a stereo camera as an input device.

depth estimation method	IoU	runtime $[\frac{s}{image}]$
monocular depth estimation	0.69	0.73
stereoscopic depth estimation	0.74	0.97

Table 5.2: Results of the part-based detection pipeline for monocular and stereoscopic surface normal representation

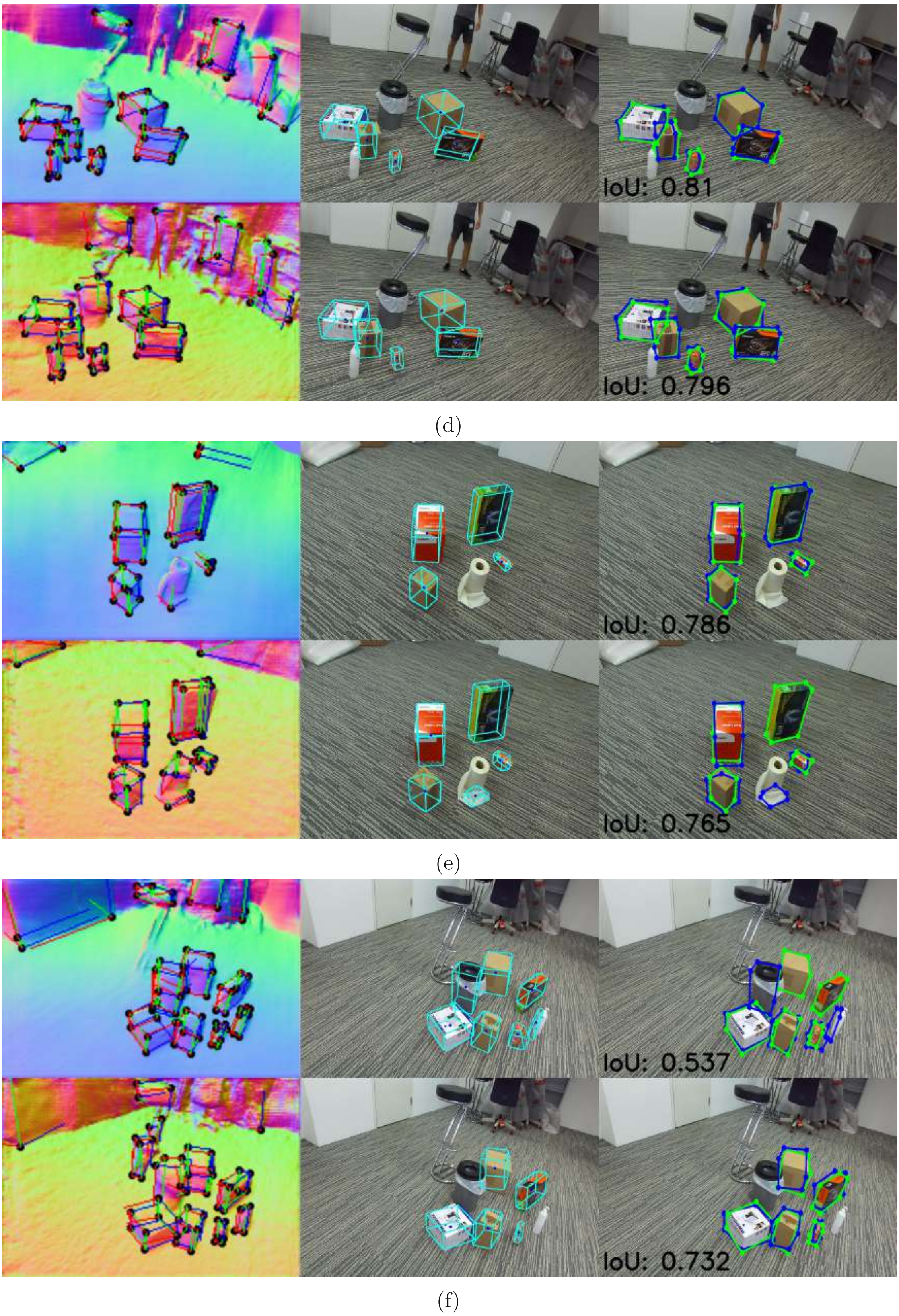


(a)

(b)

(c)

Figure 5.16: Real-world inference results of the cuboid detection framework. The upper left images show the detector output on monocular estimated surface normals. The lower left images show the detector output on stereoscopic estimated surface normals. The middle images show the corresponding outputs of the cuboid-construction post-processing step. The right images show the convex hulls of the ground truth and the prediction, with subsequent IoU score.

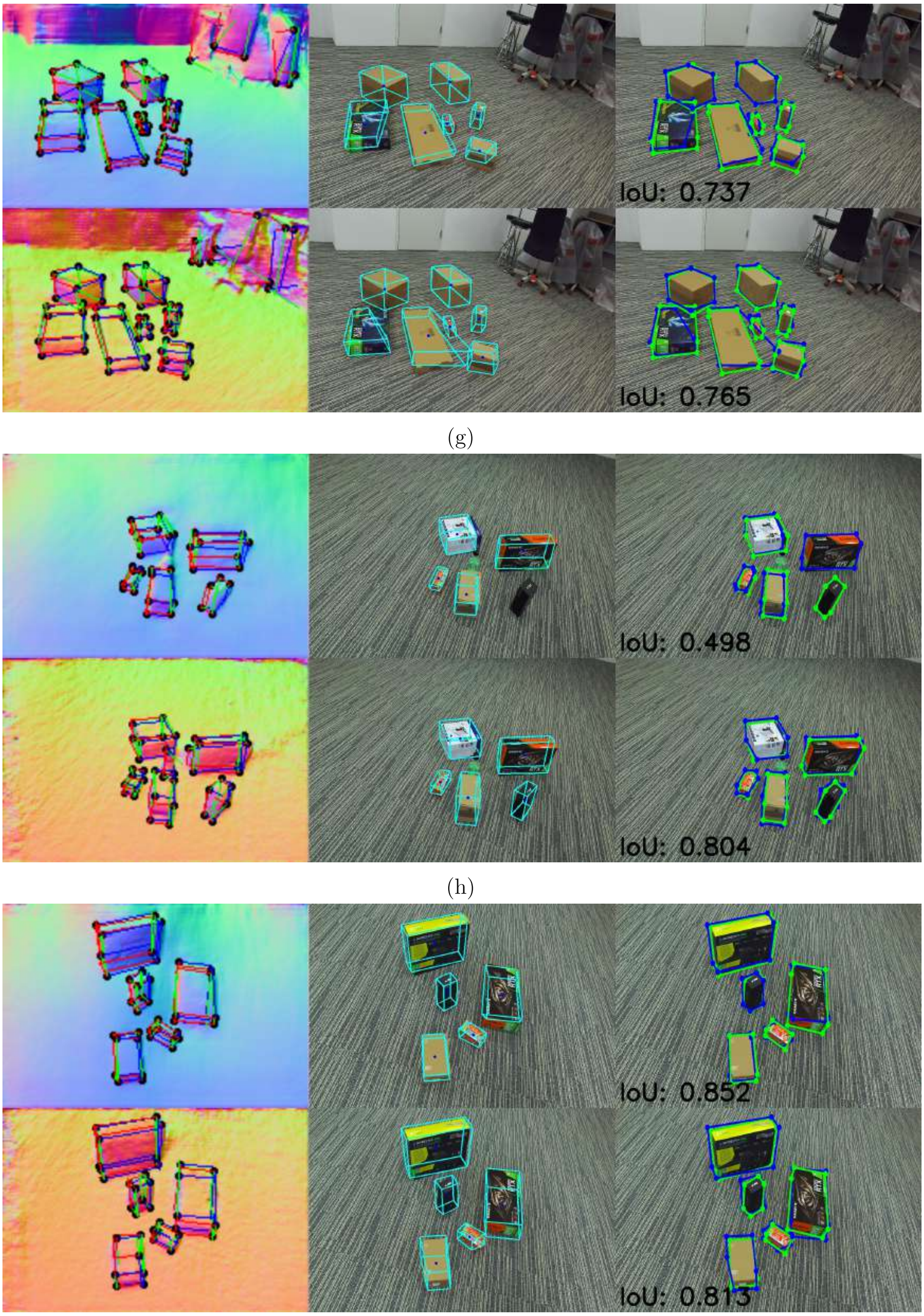


(d)

(e)

(f)

Figure 5.17: Real-world inference results of the cuboid detection framework. The upper left images show the detector output on monocular estimated surface normals. The lower left images show the detector output on stereoscopic estimated surface normals. The middle images show the corresponding outputs of the cuboid-construction post-processing step. The right images show the convex hulls of the ground truth and the prediction, with subsequent IoU score.



(g)

(h)

(i)

Figure 5.18: Real-world inference results of the cuboid detection framework. The upper left images show the detector output on monocular estimated surface normals. The lower left images show the detector output on stereoscopic estimated surface normals. The middle images show the corresponding outputs of the cuboid-construction post-processing step. The right images show the convex hulls of the ground truth and the prediction, with subsequent IoU score.

5.4 Summary and Discussion

In summary, the object-centered detection pipeline does not work for the highly symmetric cuboids. The regressed corners have to be distinguishable, which is not the case. On the other hand, for palettes, the object-centered detection scheme works fine. Thus, the top-down detection approach has a lot of potential applications in robotics and autonomous systems. The part-based detection pipeline allows us to construct the 2D embedding of the cuboid poses. We achieve this by focusing on the corners that are even more basic geometric entities than the cuboids. Further, the monocular and the stereoscopic surface normal representations serve as adequate geometric representations for the part-based pose estimation approach. The monocular depth estimation comes with the benefits of a better run-time and less expensive camera requirements. On the other hand, the stereo-matching-based depth estimations lead to slightly more accurate results and offer absolute depth measurements, which is necessary for full 3D pose recovery. The most important insight of the experiments is that we can construct the object poses with a detector that has never seen a real cuboid during training. Further, the geometric representation allows the model to overcome appearance dependency and focuses on the geometric properties. Another benefit of the geometric representation is the less complex input for the training, which leads to fast convergence of the loss.

CHAPTER 6

Conclusion and Future Work

In this section, we give a conclusion for the geometric-aware detection schemes. Further, we discuss the advantages and drawbacks of this methodology compared to the widely-used appearance-based detection strategies. We compare the part-based and object-centered methods and argue the benefits of each technique in different scenarios.
Lastly, we give an outlook for future experiments and potential new branches of research in the domain of geometric-aware detection and pose estimation.

6.1 Conclusions

The experiments show that the learning-based geometric-aware detection works and can serve as an alternative to appearance-based detection strategies. The geometric representation helps to narrow the gap between synthetic and real-world domains. The renderings of the synthesis pipeline look artificial and are distinguishable from the real-world test images. However, after the surface normal transformation, there is barely a difference between synthetic and real-world images. Additionally, we do not have to consider all possible object color and texture configurations, during data synthesis. Thus, geometric-aware detection schemes have a clear advantage over appearance-based detection if training on synthetic data is required. On the other hand, we can apply geometric-aware detection strategies only if the objects are solely identifiable by their geometry, which is a limitation and a disadvantage of our method.
From the perspective of 3D detection, monocular depth estimation is not a suitable way to construct the geometric representation, due to the lack of precise distance measurements. In the case of the stereoscopic depth estimations, we have access to absolute distance measurements, which makes it the desirable representation for 3D pose estimation.
Another way to deal with the absence of absolute depth measurements is restricting the class of detectable objects to a set where you know the size of each. We do not follow this research-path, since it stands in contrast to the semi-open cuboid world, which we

assumed for our experiments.
The palette experiment shows qualitatively promising results and indicates the feasibility of many real-world tasks, where geometric-aware detection can be applied.
The part-based detection experiments show promising results from a qualitative and quantitative perspective. It allowed us to construct the 2D projected poses of cuboids from all sizes in scenes with occlusion and clutter. This method of interpreting the geometric data could be beneficial for geometric scene understanding, which is a step towards general robotic perception.

6.2 Future Work

Future experiments should investigate full 3D pose recovery based on the stereoscopic geometric representation and the subsequential provided absolute depth measurements. Therefore, we could widen the detection framework by 3D pose estimation and bird's eye view estimation. An additional potential experiment path would be to develop further the part-based detection towards a geometric scene interpretation. Such a geometry-based scene representation could be beneficial for all kinds of robotic tasks and be a step towards a more human-like perception.
Additionally, it would be beneficial to train the model to detect some entities from a pose estimation benchmark data set, like T-Less [HHO$^+$17]. Thus, we could quantitatively compare the geometric-aware detection scheme with appearance-based methods. Further, we could enrich the detection framework with other geometric primitives, like cylinders or cones.
In conclusion, the area of geometric-aware pose estimation opens many research directions, with lots of potential applications in robotics and autonomous systems.

List of Figures

List of Tables

List of Algorithms

Bibliography

[BF82] Stephen T Barnard and Martin A Fischler. Computational stereo. *ACM Computing Surveys (CSUR)*, 14(4):553–572, 1982.

[BKC17] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 39(12):2481–2495, 2017.

[CC18] Jia-Ren Chang and Yong-Sheng Chen. Pyramid stereo matching network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5410–5418, 2018.

[DDS$^+$09] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[GDDM14] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[Gir15] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[GLU12] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3354–3361. IEEE, 2012.

[HEH05a] Derek Hoiem, Alexei A Efros, and Martial Hebert. Automatic photo pop-up. In *ACM SIGGRAPH 2005 Papers*, pages 577–584. 2005.

[HEH05b] Derek Hoiem, Alexei A Efros, and Martial Hebert. Geometric context from a single image. In *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, volume 1, pages 654–661. IEEE, 2005.

[HHO+17] Tomáš Hodan, Pavel Haluza, Štepán Obdržálek, Jiri Matas, Manolis Lourakis, and Xenophon Zabulis. T-less: An rgb-d dataset for 6d pose estimation of texture-less objects. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 880–888. IEEE, 2017.

[HZRS16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[KJMS18] Youngjung Kim, Hyungjoo Jung, Dongbo Min, and Kwanghoon Sohn. Deep monocular depth estimation via integration of global and local predictions. *IEEE transactions on Image Processing*, 27(8):4131–4144, 2018.

[KKS+18] Seung-Wook Kim, Hyong-Keun Kook, Jee-Young Sun, Mun-Cheon Kang, and Sung-Jea Ko. Parallel feature pyramid network for object detection. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 234–250, 2018.

[KMD+17] Alex Kendall, Hayk Martirosyan, Saumitro Dasgupta, Peter Henry, Ryan Kennedy, Abraham Bachrach, and Adam Bry. End-to-end learning of geometry and context for deep stereo regression. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 66–75, 2017.

[KOPW15] Alireza G Kashani, Michael J Olsen, Christopher E Parrish, and Nicholas Wilson. A review of lidar radiometric processing: From ad hoc intensity correction to rigorous radiometric calibration. *Sensors*, 15(11):28099–28128, 2015.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.

[LD18] Hei Law and Jia Deng. Cornernet: Detecting objects as paired keypoints. In *Proceedings of the European conference on computer vision (ECCV)*, pages 734–750, 2018.

[LS18] Zhengqi Li and Noah Snavely. Megadepth: Learning single-view depth prediction from internet photos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2041–2050, 2018.

[LSD15] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[MIH+16a] Nikolaus Mayer, Eddy Ilg, Philip Hausser, Philipp Fischer, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *Proceedings* *of the IEEE conference on computer vision and pattern recognition*, pages 4040–4048, 2016.

[MIH+16b] Nikolaus Mayer, Eddy Ilg, Philip Hausser, Philipp Fischer, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4040–4048, 2016.

[NYD16] Alejandro Newell, Kaiyu Yang, and Jia Deng. Stacked hourglass networks for human pose estimation. In *European conference on computer vision*, pages 483–499. Springer, 2016.

[PIL19] Giorgia Pitteri, Slobodan Ilic, and Vincent Lepetit. Cornet: generic 3d corners for 6d pose estimation of new objects without retraining. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, pages 0–0, 2019.

[PLH+19] Sida Peng, Yuan Liu, Qixing Huang, Xiaowei Zhou, and Hujun Bao. Pvnet: Pixel-wise voting network for 6dof pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4561–4570, 2019.

[RDGF16] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[RHGS15] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *arXiv preprint arXiv:1506.01497*, 2015.

[RLH+19] René Ranftl, Katrin Lasinger, David Hafner, Konrad Schindler, and Vladlen Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *arXiv preprint arXiv:1907.01341*, 2019.

[RRF+16] Konstantinos Rematas, Tobias Ritschel, Mario Fritz, Efstratios Gavves, and Tinne Tuytelaars. Deep reflectance maps. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4508–4516, 2016.

[SEE+12] Jürgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. A benchmark for the evaluation of rgb-d slam systems. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 573–580. IEEE, 2012.

[WH06] Yihong Wu and Zhanyi Hu. Pnp problem revisited. *Journal of Mathematical Imaging and Vision*, 24(1):131–141, 2006.

[XSNF17] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. *arXiv preprint arXiv:1711.00199*, 2017.

[XZ20] Haofei Xu and Juyong Zhang. Aanet: Adaptive aggregation network for efficient stereo matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1959–1968, 2020.

[YTO$^+$14] Keisuke Yoneda, Hossein Tehrani, Takashi Ogawa, Naohisa Hukuyama, and Seiichi Mita. Lidar scan feature for localization with highly precise 3-d map. In *2014 IEEE Intelligent Vehicles Symposium Proceedings*, pages 1345–1350. IEEE, 2014.

[YWSD18] Fisher Yu, Dequan Wang, Evan Shelhamer, and Trevor Darrell. Deep layer aggregation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2403–2412, 2018.

[Zha12] Zhengyou Zhang. Microsoft kinect sensor and its effect. *IEEE multimedia*, 19(2):4–10, 2012.

[ZMGL15] Junbo Zhao, Michael Mathieu, Ross Goroshin, and Yann Lecun. Stacked what-where auto-encoders. *arXiv preprint arXiv:1506.02351*, 2015.

[ZSZ$^+$20] ChaoQiang Zhao, QiYu Sun, ChongZhen Zhang, Yang Tang, and Feng Qian. Monocular depth estimation based on deep learning: An overview. *Science China Technological Sciences*, pages 1–16, 2020.

[ZWK19] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. Objects as points. *arXiv preprint arXiv:1904.07850*, 2019.