Dipl.-Ing. Alexander Grabner BSc.

Single Image 3D Vision for Objects in the Wild

DOCTORAL THESIS

to achieve the university degree of
Doktor der technischen Wissenschaften

submitted to

Graz University of Technology

Thesis Supervisor
Prof. Dr. Vincent Lepetit
Graz University of Technology, Austria

Thesis Referee
Prof. Dr. Federico Tombari
Technical University of Munich, Germany

Graz, Austria, 2020

To my parents.

Any sufficiently advanced technology
is indistinguishable from magic.

Arthur C. Clarke

Abstract

Understanding the 3D structure of a scene from a single image is one of the holy grails of computer vision. The ability to thoroughly reason about a scene in 3D is important for a wide range of applications including augmented reality, robotics, and computer-aided design. However, 3D vision from a single image is still an unsolved problem that poses many challenges in practice. For example, there is an infinite number of possibilities to spatially arrange objects in a scene.

In this thesis, we present state-of-the-art methods for estimating different 3D properties which contribute to a holistic 3D scene understanding. Our methods combine deep learning and geometry to perform object detection, 3D pose estimation, 3D model retrieval, and focal length estimation given only a single RGB image. We especially focus on in-the-wild scenarios where no prior knowledge about the underlying scene is available and raw image intensities are the only source of information.

We first propose a method that performs 3D pose estimation and 3D model retrieval for objects of arbitrary categories. For this purpose, we introduce a novel approach which recovers the 3D pose of an object from predicted virtual control points and 3D dimensions. We then show that the predicted 3D pose of an object provides an effective prior for 3D model retrieval by comparing RGB images of objects only to 3D model renderings under the predicted 3D pose. Utilizing this prior significantly increases the retrieval performance while reducing the computational complexity.

Next, we integrate our proposed 3D pose estimation approach into an object detection pipeline to deal with multiple objects in a single image. Additionally, we augment our pipeline to estimate the image focal length. In this way, we exploit the geometric prior given by the focal length for 3D pose estimation. Our method uses a joint optimization which enforces a geometric consensus between the predicted 3D poses and the focal length. This results in significantly improved 3D pose estimates.

To improve the 3D model retrieval performance, we further augment our pipeline to compute pose invariant 3D shape descriptors from objects in RGB images. By first mapping RGB images and 3D models to a common pose aware low-level representation and then computing pose invariant 3D shape descriptors, major parts of our pipeline benefit from training on a virtually infinite amount of synthetic data. In this way, we increase the retrieval performance while significantly reducing the size of the retrieval database.

Finally, we present a novel 3D pose refinement strategy based on differentiable rendering. By reprojecting our retrieved 3D models under our predicted 3D poses, we establish a feedback loop which we use to improve the 3D poses. Our method compares RGB images and 3D model renderings in a feature space which is optimized for 3D pose refinement and additionally learns to approximate the non-differentiable rasterization backward pass in differentiable rendering. We use this refinement method in combination with our 3D pose estimation and 3D model retrieval approach to predict fine-grained 3D poses for objects in the wild without providing initial 3D poses or ground truth 3D models given only a single RGB image.

Keywords. 3D Object Detection, 3D Pose Estimation, 3D Model Retrieval, Focal Length Estimation, Differentiable Rendering, Refinement, Deep Learning, Geometry

Kurzfassung

Einer der heiligen Grale des Maschinellen Sehens ist es die 3D-Struktur einer Szene auf Basis eines einzelnen Bildes zu verstehen. Die Fähigkeit eine Szene vollständig dreidimensional zu interpretieren ist wichtig für eine Vielzahl von Anwendungen, einschließlich Erweiterte Realität, Robotik und computergestütztem Konstruieren. Das 3D-Bildverstehen auf Basis eines einzelnen Bildes ist jedoch immer noch ein ungelöstes Problem, das in der Praxis viele Herausforderungen mit sich bringt. Zum Beispiel gibt es unendlich viele Möglichkeiten Objekte in einer Szene räumlich anzuordnen.

In dieser Arbeit präsentieren wir modernste Methoden zur Schätzung verschiedener 3D-Eigenschaften, die zu einem ganzheitlichen 3D-Verständnis einer Szene beitragen. Unsere Methoden kombinieren Tiefes Lernen und geometrisches Wissen, um Objekterkennung, 3D-Lagenschätzung, 3D-Modellsuche und Brennweitenschätzung auf Basis eines einzigen RGB-Bildes durchzuführen. Hierbei konzentrieren wir uns insbesondere auf Szenarien außerhalb von kontrollierten Laborbedingungen, in denen keine Vorkenntnisse über die zugrunde liegende Szene verfügbar sind und Bildintensitätswerte die einzige Informationsquelle sind.

Zuerst präsentieren wir eine Methode, die eine 3D-Lagenschätzung und eine 3D-Modellsuche für Objekte beliebiger Kategorien durchführt. Zu diesem Zweck stellen wir einen neuartigen Ansatz vor, der die 3D-Lage eines Objekts aus vorhergesagten virtuellen Kontrollpunkten und 3D-Dimensionen berechnet. Anschließend zeigen wir, dass die vorhergesagte 3D-Lage eines Objektes eine effektive Vorinformation für die Suche von 3D-Modellen bietet, indem RGB-Bilder von Objekten nur mit erzeugten 3D-Modellbildern unter der vorhergesagten 3D-Lage verglichen werden. Die Verwendung dieses Vorwissens erhöht die Genauigkeit erheblich und verringert gleichzeitig den Rechenaufwand.

Als Nächstes integrieren wir unseren vorgeschlagenen 3D-Lagenschätzungsansatz in ein Objekterkennungssystem, um mehrere Objekte in einem einzigen Bild zu behandeln. Zusätzlich erweitern wir unser System mit einer Bildbrennweitenschätzung. Auf diese Weise nutzen wir die geometrische Vorinformation, die die Brennweite für die 3D-Lagenschätzung bietet. Unsere Methode verwendet eine Optimierung, die einen geometrischen Konsens zwischen den vorhergesagten 3D-Posen und der Brennweite erzielt. Dies führt zu signifikant verbesserten 3D-Lagenschätzungen.

Um die Genauigkeit der 3D-Modellsuche zu verbessern, erweitern wir unser System zusätzlich mit der Berechnung von lageninvarianten 3D-Formdeskriptoren für Objekte in RGB-Bildern. Durch die Projektion von RGB-Bildern und 3D-Modellen in eine gemeinsame lagenbewusste Darstellung auf niedriger Ebene und die anschließende Berechnung lageninvarianter 3D-Formdeskriptoren profitieren große Teile unseres Systems von dem Lernen mit einer nahezu unbegrenzten Menge synthetischer Daten. Auf diese Weise erhöhen wir die Genauigkeit und reduzieren gleichzeitig die Größe der Suchdatenbank erheblich.

Abschließend präsentieren wir eine neuartige Strategie zur Verbesserung von geschätzten 3D-Lagen, die auf differenzierbarer Bildsynthese basiert. Indem wir unsere vorhergesagten 3D-Modelle unter unseren vorhergesagten 3D-Lagen projizieren, schließen wir eine Rückkopplungsschleife, mit der wir die vorhergesagten 3D-Lagen verbessern. Unsere Methode vergleicht RGB-Bilder und 3D-Modellbilder in einem Merkmalsraum, der für die Verfeinerung von 3D-Lagen optimiert ist, und lernt zusätzlich, den nicht differenzierbaren Rasterisierungsschritt in der differenzierbaren Bildsynthese zu approximieren. Wir verwenden diese Verfeinerungsmethode in Kombination mit unserer 3D-Lagenschätzung und unserer 3D-Modellsuche, um genaue 3D-Lagen für Objekte in freier Wildbahn auf Basis eines einzigen RGB-Bildes vorherzusagen, ohne initiale 3D-Lagen oder 3D-Modelle bereitzustellen.

Schlagwörter. 3D-Objekterkennung, 3D-Lagenschätzung, 3D-Modellsuche, Brennweitenschätzung, Differenzierbare Bildsynthese, Verfeinerung, Tiefes Lernen, Geometrie

Affidavit

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used.

The text document uploaded to TUGRAZonline is identical to the present doctoral thesis.

Date

Signature

Acknowledgments

I want to express my sincere gratitude to all people who supported me during the course of my Ph.D. studies.

First of all, I would like to thank Professor Vincent Lepetit, who instantly agreed to supervise me and offered me a position at the institute. He guided me, motivated me, and helped me with his knowledge and experience. He gave me the freedom to pursue my own ideas and always supported my decisions.

I am also grateful to Professor Federico Tombari for taking his time and acting as a reviewer for this thesis.

Next, I want to thank Peter M. Roth, who always had an open door for a meeting and an open ear for a discussion. He built me up when I was down, took pressure off me in stressful times, and provided fair, constructive, and decisive feedback which tremendously helped me to progress.

In fact, all past and current members of the ICG, who shared a part of the journey with me, would deserve a special mention. Because of them, the institute is not only a great place to work at but also a place where I felt at home. I got the chance to meet so many wonderful personalities that I had intense discussions, fun chats, exciting conference visits, interesting lectures, tasty lunches, and plenty of laughs with. To me, these people are more than colleagues, they are friends.

Finally, I want to thank my parents Elfriede Grabner and Walter Grabner. They unconditionally supported me not only during this thesis but during my entire life. Without them, I would be far from where I am today.

For me, the past three years were a journey that shaped me in both an academic as well as a personal way. It was a time that I will always look back upon with a smile on my face.

This work was supported by the Christian Doppler Laboratory for Semantic 3D Computer Vision, funded in part by Qualcomm Inc.

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

Contents

Vision allows humans to perceive the world around them without physical interaction [253]. We use our eyes to see and our brain to understand the seen [182]. This ability to not only see but *understand* our environment enables us to perform complex tasks like safely driving cars through crowded streets, accurately reading other people's emotions from their facial expressions, or correctly decrypting a university student's shaky handwriting on his first exam. Consequently, vision is our dominant sense and our most important source of information [46, 66, 229].

Inspired by the incredible capabilities of human vision, computer vision aims at providing machines with visual intelligence [265]. Teaching machines how to visually reason gives rise to a multitude of applications across different domains. Computer vision makes it possible to automate tasks like driving[1], enhance our own vision by seeing outside the spectrum of visible light[2], explore new use cases like virtual reality[3], and much more. It

[1] `https://www.tesla.com/autopilot`

[2] `https://www.flir.com/discover`

[3] `https://www.oculus.com/experiences`

is a technology of major industrial as well as social importance as it has the potential to make our lives not only easier but also safer.

1.1 Problem Statement

Recently, there have been significant advances and breakthroughs in the field of computer vision [83, 96, 146]. In the last decade, the advent of deep learning [82], the availability of large scale datasets [165, 233], and the access to exponentially growing computational resources [248] led to unprecedented progression in many disciplines. Today, computer vision models even surpass human-level performance for specific tasks like fine-grained classification [97] or industrial inspection [17].

However, if we move beyond constrained tasks and environments computer vision models fall short of human-level performance [84, 194]. For example, machines cannot match humans at interpreting an arbitrary image, as shown in Fig. 1.1. Given a single RGB image, humans can quickly reason about the 3D structure of the depicted scene. We subconsciously infer the presence of different objects. We build an internal representation of their 3D shapes, 3D poses, and 3D dimensions. We understand their spatial arrangement in the scene and derive high-level semantics, e.g., that the flower vase is placed on top of the table or that the geometric topography of the objects resembles a living room. We can even provide information about the camera that was used to capture the image. Connecting all these individual pieces of information gives us a detailed 3D understanding of the scene. Despite the large number of technological achievements in the past, computer vision models still do not achieve a similar level of general 3D scene understanding [116, 123, 278] due to the large number of challenges this task poses in practice (see Section 1.3).

In this thesis, we thus aim at advancing the state of the art in single image 3D vision [292]. In particular, we present methods for estimating different 3D properties which contribute to a holistic 3D scene understanding. Our methods perform object detection, 3D pose estimation, 3D model retrieval, and estimation of camera intrinsics given only a single RGB image. Combining all of our proposed methods allows us to infer a semantic 3D reconstruction of a scene, as shown in Fig. 1.1. This semantic reconstruction represents a dynamic configuration of objects, where a category label, a 3D model, and a 3D pose is assigned to each detected object. This results in a compact but detailed high-level representation of the scene with rich semantics and object hierarchies. Such a semantic 3D reconstruction paves the way for a host of applications (see Section 1.2).

All of our proposed methods use sophisticated combinations of deep learning and geometry to leverage prior information via learning while exploiting geometric constraints on the desired solutions (see Section 1.4). We especially focus on in-the-wild scenarios where no prior knowledge about the underlying scene is available and raw image intensities are the only source of information. In addition, all methods are designed to be fast and scalable during inference so that they can be used in real-time applications while being able to handle objects of arbitrary categories as well as multiple objects in a single image.

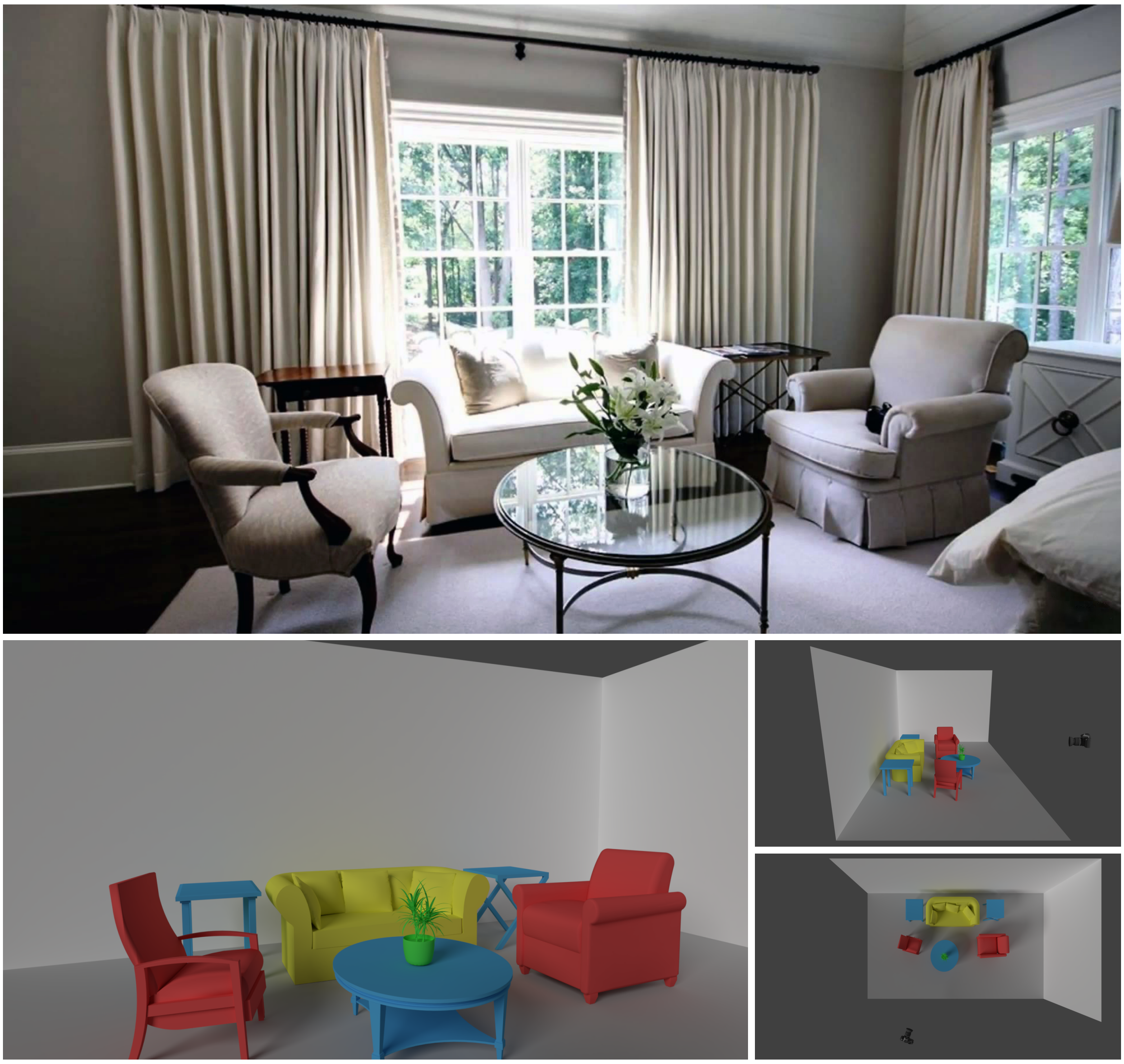


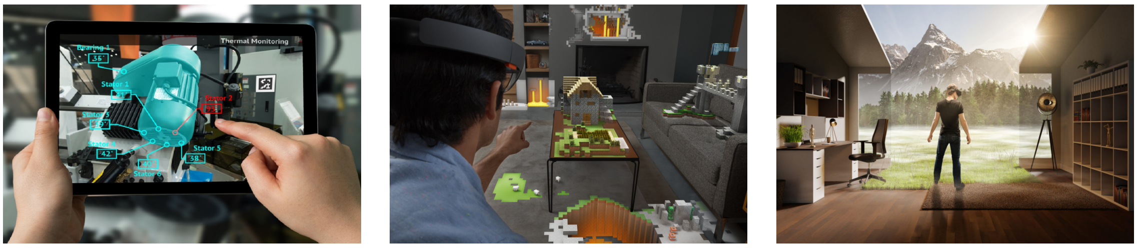
Figure 1.1: An image of an arbitrary scene captured in the wild (*big image*)[4]. A semantic 3D reconstruction of the scene inferred by the methods proposed in this thesis showing detected objects, predicted 3D poses, retrieved 3D models, and estimated camera intrinsics (*medium image*). Our semantic 3D reconstruction of the scene observed from different viewpoints (*small images*).

1.2 Applications

The ability to thoroughly reason about a scene in 3D is important for a wide range of applications. While some applications benefit from 3D vision, other applications only become possible thanks to this technology. Below, we describe a number of example use cases in practice.

[4] `https://www.decoratorist.com/monochromatic-living-rooms-white`

Mixed Reality: 3D vision is a prerequisite for any form of mixed reality technology like augmented reality or virtual reality. In augmented reality, virtual content is superimposed onto the real world (see Figs. 1.2a and 1.2b). In virtual reality, the user is immersed in a fully artificial world (see Fig. 1.2c). However, both technologies require their respective devices to be registered in the real world in 3D to create the illusion. In addition, 3D scene understanding is needed to enable advanced applications.



(a) (b) (c)

Figure 1.2: Examples of different mixed reality applications. (a) Augmented reality: Existing objects are augmented with virtual information[5]. (b) Augmented reality: Virtual objects are added to the real world[6]. (c) Virtual reality: The user is immersed in a fully artificial world[7].

In augmented reality, a device must reason about objects in 3D to overlay virtual information onto the real world in a geometrically and perspectively correct manner. In this way, existing objects can be augmented with virtual information, as shown in Fig. 1.2a. This is important for applications like instructed maintenance, guided repairs, point-of-view navigation, virtual wardrobes, street shopping, virtually assisted medical surgeries, and interactive learning experiences.

In addition, augmented reality makes it possible to add new virtual objects to the real world, as shown in Fig. 1.2b. These virtual objects can interact with real objects in geometrically and physically plausible ways. This enables applications like mixed reality gaming, CAD sculpting, 3D drawing, head-up displays, virtual input devices, at-home customer buying experiences, interactive teaching, cinematic movie prototyping, and holographic video conferencing.

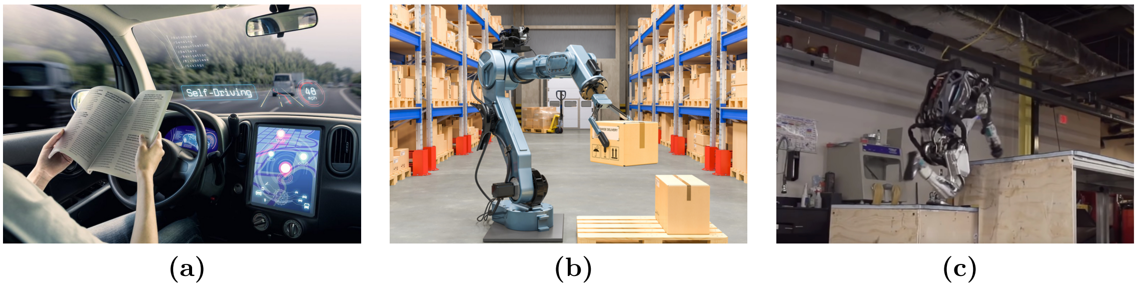
In virtual reality, an important part of the experience is the ability to move freely in the artificial world, as shown in Fig. 1.2c. In this case, a device needs to understand the user's 3D environment and keep track of the user's motion to avoid accidental collisions with objects in the real world like chairs or tables. This feature is important for applications like gaming, telepresence, and virtual education.

[5] https://www.strategy4.org/my-blog/augmented-reality-exponential-technologies-in-the-smart-factory

[6] https://www.microsoft.com/hololens

[7] https://www.virsabi.com/vr-revolutionizing-architecture-and-design

Robotics: Robots are physical agents in our 3D world. By understanding their environment in 3D, they are able to interact with it in diverse ways. Hence, robots provided with visual intelligence have many use cases.



(a) **(b)** **(c)**

Figure 1.3: Examples of different robotics applications. (a) 3D vision allows robots to autonomously navigate through their environment[8]. (b) 3D scene understanding lets robots grasp and manipulate nearby objects[9]. (c) Visual intelligence enables robots to perform complex tasks like running an obstacle parkour[10].

For example, 3D vision allows robots to autonomously navigate through their environment, as shown in Fig. 1.3a. In order to navigate safely, robots must recognize their static surrounding and dynamic objects in it. In this way, they can track other object's trajectories and avoid crashes. In contrast to human-operated machines, autonomously navigating robots do not get distracted or fatigued. Additionally, they can have superhuman abilities like 360° vision or night sight, which further increase safety. Autonomous navigation is not only important for self-driving cars (see Fig. 1.3a) but also for trucks, buses, trains, trams, planes, drones, ships, submarines, agricultural machines, domestic robots, spacecraft, and other vehicles.

3D scene understanding does not only allow robots to navigate their environment but also to interact with it. By sensing their surrounding, robots can grasp and manipulate nearby objects, as shown in Fig. 1.3b. To reliably grab objects, robots must reason about the 3D shape and 3D pose of objects. This enables robots to automate tasks like warehouse logistics, conveyor assembly, delivery or pickup services, mechanic repairs, agricultural harvesting, hospital assistance, and elderly care.

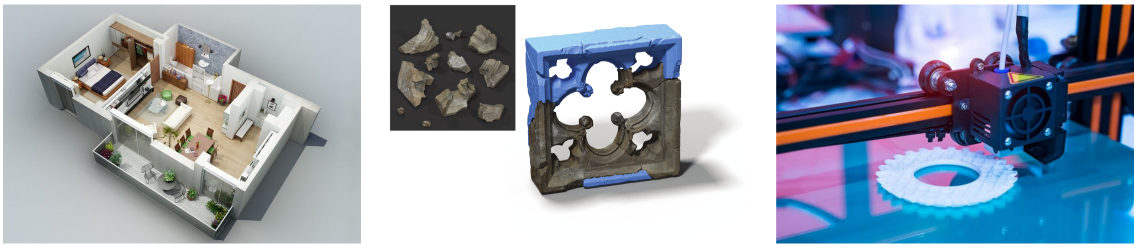
Visual intelligence makes it possible for robots to perform complex tasks that were previously exclusive to humans like running an obstacle parkour, as shown in Fig. 1.3c. In the future, visual reasoning will even allow robots to execute tasks beyond the human spectrum of capabilities.

[8] `https://www.bbc.com/news/business-45048264`

[9] `https://www.ehstoday.com/safety-technology/article/21920298/how-warehouse-robotics-reduce-worker-injuries`

[10] `https://www.bostondynamics.com/atlas`

Computer-Aided Design: Today, 3D content is becoming increasingly popular. Digital 3D models provide rich experiences and have many use cases in different domains as they resemble three-dimensional entities in the real world. 3D vision can assist users in the creation of 3D content and help them to interact more efficiently with it.



(a) (b) (c)

Figure 1.4: Examples of different computer-aided design applications. (a) A semantic reconstruction of real-world apartment[11]. (b) A 3D object is automatically restored from shattered fragments[12]. (c) A 3D printer manufactures a retrieved spare part[13].

Many applications require parts of the real world to be reconstructed virtually. Such reconstructions are typically created by 3D artists in a lengthy and tedious process. 3D scene understanding makes it possible to automate the creation of 3D content from real-world environments. In this way, both large environments (Fig. 1.4a) as well as individual object fragments (see Fig. 1.4b) can be digitalized. This is not only important for real estate and archeology applications but also for mapping entire cities without dynamic objects like cars, capturing forensic crime scenes, documenting car accidents, and accelerating the creation of virtual worlds based on real-world scenes for games, movies, and virtual reality experiences.

Another application in this domain is visual search in 3D databases. In this context, 3D models of spare parts for damaged machines can be found given only an image of a faulty part. In addition, a 3D printer can be used to manufacture the retrieved 3D model on demand, as shown in Fig. 1.4c. This technology significantly reduces the downtime of systems by accelerating the repair process in both industrial as well as consumer settings. It is also possible to retrieve 3D models for complex devices consisting of multiple parts like prosthetics.

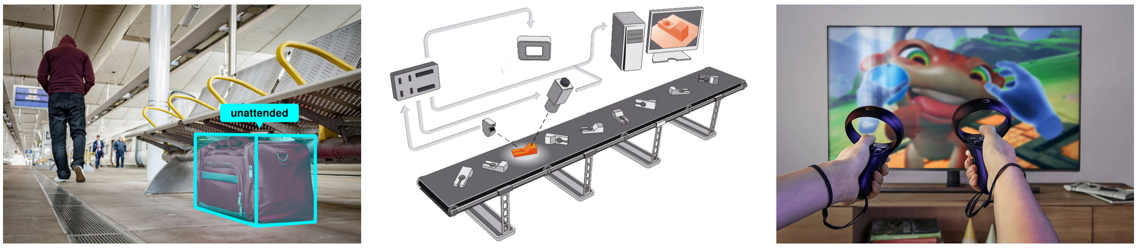
Visual Intelligence: The ability to visually reason about a scene in 3D enables smart applications across different domains. For example, in the area of surveillance, 3D scene understanding can be used to detect unattended luggage at airports (see Fig. 1.5a), recognize forgotten objects in public transport, or find missing keys at home. It can also

[11] `https://www.goodshomedesign.com/awesome-3d-plans-apartments`

[12] `https://www.tugraz.at/institutes/cgv/schreck/3d-object-retrieval`

[13] `https://www.bigstockphoto.com/search/?contributor=kenny001`

prevent security systems from raising false alarms by differentiating intruders from pets or randomly falling objects.



(a) (b) (c)

Figure 1.5: Examples of different visual intelligence applications. (a) Surveillance: Detection of unattended luggage[14]. (b) Industry: Quality control of manufactured goods[15]. (c) Human assistance: Input devices like controllers are tracked in 3D[16].

In the area of industrial applications, 3D reasoning can automate the quality control of manufactured goods (see Fig. 1.5b), the inspection of machines, and the stocktaking in stores. This is especially important for industrial areas which are difficult to access by humans due to limited space, extreme temperatures, or toxic environmental conditions.

In the area of human assistance, 3D vision can be used to track 3D input devices like controllers, as shown in Fig. 1.5c. Visual reasoning can also be used to provide acoustic scene descriptions for blind people or detect emergency cases for elderly people living alone. These are only a handful of examples from the realm of possible applications.

1.3 Challenges

Predicting 3D properties from single 2D images poses many challenges in practice. Below, we describe the most important challenges in single image 3D vision.

2D-3D Information Loss: Images are 2D projections of the 3D world [65]. This dimensionality reduction from 3D to 2D space causes a significant information loss [253]. In addition, digital images are lossy discretizations of continuous signals as they are generated by sampling in the spatial domain and quantization in the intensity domain [275]. As a consequence, the task of recovering 3D information from a 2D observation is ill-posed from an information theory point of view [231]. Hence, it is not surprising that the majority of breakthroughs in computer vision have been achieved in 2D tasks like classification [99], detection [225], and segmentation [170] while 3D tasks remain an open challenge.

However, by exploiting a massive pool of prior information paired with geometric constraints humans can narrow down the space of possible solutions and find 3D config-

[14] https://www.bis.cz

[15] https://www.mvtec.com/de/services-support/technologien/3d-vision

[16] https://www.occulus.com/quest

urations which explain the 2D observations well. Thus, this task should also be possible for artificial 3D vision systems.

Appearance Variations: Another main challenge in single image 3D vision is the huge variability in appearance. The appearance of an object in an image is subject to a large number of factors. First, the 3D shape of an object effects its appearance. Even though objects within a certain category share common characteristics, they can have vastly different 3D shapes [33], as shown in Fig. 1.6a. Second, the appearance of an object can change significantly with its 3D pose, as shown in Fig. 1.6b. Not only the 3D rotation but also the 3D translation effects the appearance of an object [86, 151]. Third, even objects with the same 3D shape and 3D pose can have diverse appearances due to different color, texture, and material, as shown in Fig. 1.6c. Especially transparent, translucent, and reflective materials cause confusion in computer vision systems.

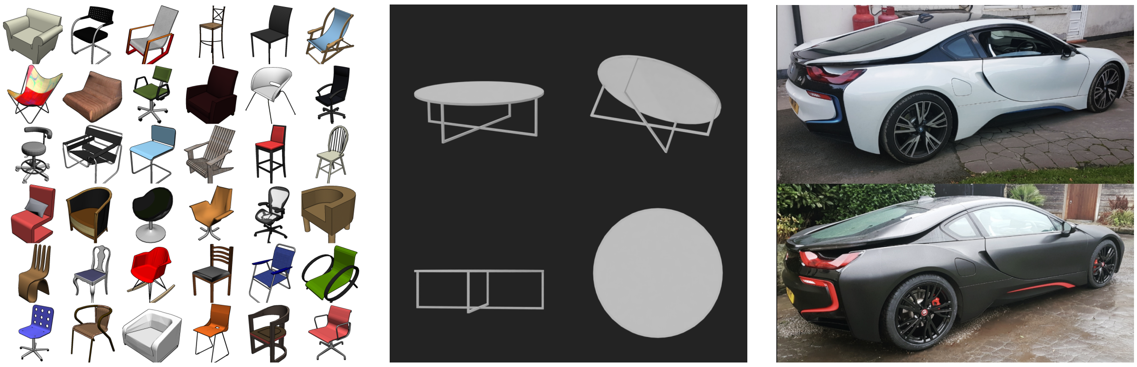

(a) (b) (c)

Figure 1.6: The appearance of an object is subject to a large number of factors: (a) 3D shape effects the appearance of objects. Even within a certain category, e.g., *chair*, objects can have vastly different 3D shapes[17]. (b) The appearance of an object can change significantly with its 3D pose[17]. (c) Different materials cause diverse appearances even for objects with the same 3D shape and 3D pose[18].

Moreover, many man-made objects have symmetries that result in ambiguous appearances [33, 108]. For example, the *table* in Fig. 1.6b can be rotated by any multiple of $\pi/2$ around the axis perpendicular to its surface and through its center in 3D without effecting its appearance in 2D. This makes it hard to perform 3D pose estimation for objects with symmetries. Even worse, there might be uninformative views of an object that make both 3D pose estimation and 3D model retrieval ambiguous (see Fig. 1.6b, *bottom right*).

In addition to properties of the object itself, the environment of an object can effect its appearance. For example, other objects might partially occlude the object of interest [119, 198], as demonstrated in Fig. 1.7a. Objects can also be truncated so that they are only

[17] https://www.shapenet.org
[18] https://www.ardengarage.co.uk/vehicle-wrapping

partially visible. Surrounding geometry effects the global illumination and, thus, the appearance of an object [52]. Moreover, cluttered background makes it difficult to detect objects in the first place [165], as illustrated in Fig. 1.7b. Finally, also environmental conditions like lighting, shadows, fog, rain, snow, and heat haze effect the appearance of objects and can even result in ambiguities, as shown in Fig. 1.7c. Single image 3D vision systems need to be robust to all these appearance variations.

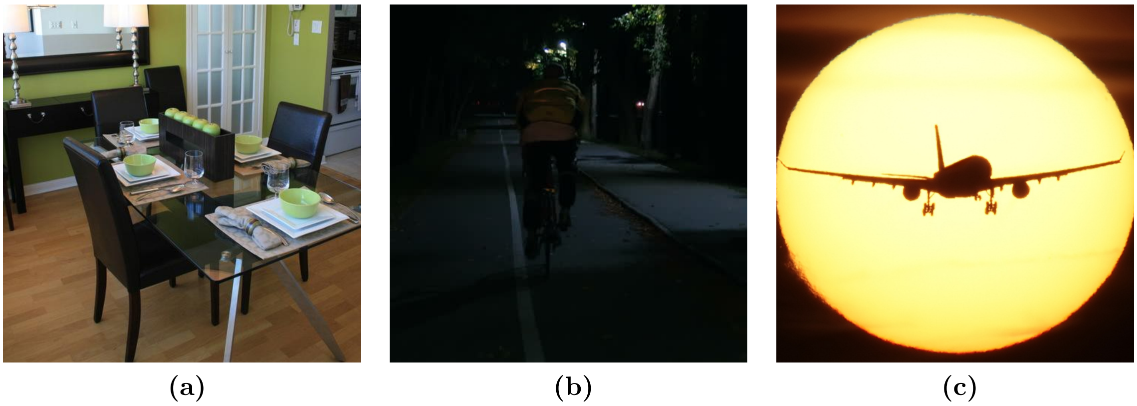


Figure 1.7: The environment of an object can effect its appearance. (a) Other objects can occlude the object of interest[19]. (b) Objects with similar appearance as their background are difficult to detect[20]. (c) Environmental conditions like lighting effect the appearance. Due to challenging illumination conditions it is unclear if the plane is flying towards or away from the camera[21].

Uncalibrated Cameras: Knowledge about the camera that was used to capture an image provides a huge prior for the prediction of many 3D properties [95]. However, in the wild, images are taken with different cameras but the camera intrinsics are in many cases unknown [302, 303]. Information about the camera might also not be available via image meta tags [128]. Thus, 3D vision in the case of uncalibrated cameras is highly ambiguous as two cameras with different intrinsics can produce similar-looking images from different viewpoints [86], for example.

In addition, different cameras use different sensors, lens systems, exposures, image processing pipelines, and compression algorithms which result in miscellaneous image artifacts [112].

Lack of Data: Today, state-of-the-art computer vision approaches are data-driven but data in the form of 2D images with 3D annotations is scarce [258]. One reason for this

[19] `https://www.houzz.com/photos/lissee-interiors-contemporary-dining-room-toronto-phvw-vp~33996`

[20] `https://www.shutterstock.com/video/clip-27658489-cyclist-riding-bicycle-night-city-close-up`

[21] `https://www.skystef.be/BRU-pics-sun-moon`

dilemma is that labeling 3D properties for 2D images is hard and time-consuming [288, 303]. As a consequence of this cumbersome annotation process, not only the amount of available labeled data is limited but also the quality of the annotations is insufficient [302]. In many cases, images are only partially labeled, inaccurately labeled, or even mislabeled such that there is a high degree of label noise in the data.

One strategy to compensate for this lack of annotated real-world images is to use synthetic data [257]. However, there is a significant domain gap between real and rendered images which limits the benefit of synthetic data in practice [185, 218, 249].

1.4 Strategies

As discussed in the previous sections, single image 3D vision has a lot of applications but poses many challenges in practice. Consequently, the field raises considerable research interest and many different approaches for predicting individual 3D properties from a single RGB image have been proposed. However, from a high-level perspective, common strategies, concepts, and evolutions surface across different tasks in 3D vision. Thus, the existing literature can be roughly categorized into two eras.

For more than 30 years, sophisticated hand-crafted pipelines have been engineered to address different tasks in 3D vision. These pipelines consist of multiple individual components that are independent of each other. While some components are of course highly specialized for a specific task, other components have a broader field of application and repeatedly appear across different tasks. For example, to perform 3D pose estimation, traditional pipelines (1) detect keypoints [94, 230, 247], (2) compute feature descriptors [15, 31, 175], (3) match descriptors against a template [202, 251], (4) find robust correspondences [63, 274], and (5) recover 3D poses of from correspondences using geometric optimizations [68, 155, 216]. Similar multi-stage pipelines with almost the same components have also been used for object detection [45, 61], estimation of camera intrinsics [276, 317], and retrieval [60, 272].

Traditional pipelines make explicit use of projective geometry mathematics [95]. In the context of 3D vision, geometric rules provide important task priors and make it possible to recover 3D information from 2D observations. However, the performance of traditional pipelines is limited because the individual components of a pipeline are developed independently of each other. As a consequence, only the last component of a pipeline specifically addresses the actual task while the other components are typically not optimized for the given task.

To overcome this limitation, there has been a paradigm shift from traditional hand-crafted pipelines to deep learning systems in recent years. In contrast to hand-crafted pipelines, deep learning systems consists of a single component that learns to perform a specific task from data instead of being explicitly programmed [82]. For this purpose, deep learning systems train models that map arbitrarily shaped inputs to arbitrarily shaped outputs. In most cases, these models are learned from large collections of input/output data pairs in a supervised offline training phase [146]. A learned model could directly map an RGB image of an object to a 3D pose [279], for example. Today, there is even a trend towards multi-task models that use a single component with branched substructures to perform multiple different tasks given a single input [75, 140].

Deep learning systems have two main advantages. First, trained models are optimized end-to-end. This means all computations between input and output are jointly optimized for the given task. Second, the development of systems is accelerated since simply collecting data supersedes the need for specific domain expertise to solve a task. However, deep learning systems often make poor predictions in practice, especially in 3D vision [238]. In many cases, they do not learn the desired mapping from input to output robustly but overfit to random structures in the data, which results in poor generalization [9, 264]. The trained models are black boxes that cannot be interpreted, do not provide guarantees, and cannot express true confidences for their predictions. Poor generalization is particularly prominent for single image 3D vision tasks due to the limited amount of training data in the form of 2D images with 3D annotations [258]. Moreover, because deep learning systems learn models entirely from data, they cannot exploit prior knowledge about the task, e.g., in the form of geometric constraints.

In this work, we, thus, propose approaches that combine the benefits of traditional pipelines and deep learning systems to increase the performance of single image 3D vision systems. In particular, we interweave learned models and geometric algorithms to leverage the power of learning complex tasks from data while exploiting geometric constraints on the desired solutions. For example, we replace multiple components of traditional pipelines with deep learning systems, predict geometrically interpretable intermediate results in trained models, approximate non-differentiable geometric operations with learned models, perform geometry aware data augmentation, and much more. Our experiments show that our proposed methods indeed increases the performance and achieve state-of-the-art results across different single image 3D vision tasks.

1.5 Contributions

In this thesis, we advance the state of the art in single image 3D vision. We present novel methods for estimating different 3D properties which contribute to a holistic 3D scene understanding. Our methods perform object detection, 3D pose estimation, 3D dimension estimation, 3D model retrieval, and estimation of camera intrinsics given only a single RGB image. All of our proposed methods use sophisticated combinations of deep learning and geometry to leverage prior information via learning while exploiting geometric constraints on the desired solutions.

Our methods gradually contribute to a joint system that infers a semantic 3D reconstruction of a scene given only a single RGB image. This system is able to detect multiple objects of arbitrary categories at different scales, estimate a 3D pose for each object, retrieve a 3D model for each object, predict the focal length of the camera, and jointly refine all predictions using differentiable rendering. In this way, we achieve a compact but detailed high-level 3D scene understanding with rich semantics and object hierarchies that paves the way for a host of applications. In particular, we make the following contributions:

- In Chapter 5, we introduce a novel 3D pose estimation and 3D model retrieval approach for objects of arbitrary categories. This method recovers the 3D pose of an object from predicted virtual 2D-3D correspondences. We then show that the predicted 3D pose of an object provides an effective prior for 3D model retrieval by comparing RGB images of objects only to 3D model renderings under the predicted 3D pose. Utilizing this prior significantly increases the retrieval performance while reducing the computational complexity.

- In Chapter 6, we propose improvements to the 3D pose estimation part of our approach introduced in Chapter 5. In particular, we integrate our 3D pose estimation method into an object detection pipeline to deal with multiple objects in a single image. Additionally, we augment our system to estimate the camera focal length. We then use a joint optimization which enforces a geometric consensus between predicted 3D poses and the focal length. In this way, we exploit the geometric prior given by the focal length for 3D pose estimation and significantly improve our 3D pose estimates.

- In Chapter 7, we propose improvements to the 3D model retrieval part of our approach introduced in Chapter 5. In particular, we first map RGB images and 3D models to a common pose aware low-level representation and then compute pose invariant 3D shape descriptors for retrieval. In this way, major parts of our system benefit from training on a virtually infinite amount of synthetic data. This results in increased retrieval performance while significantly reducing the size of the retrieval database.

- Finally, we present a novel 3D pose refinement strategy based on differentiable rendering in Chapter 8. By reprojecting our retrieved 3D models under our predicted 3D poses computed as described in Chapters 5 to 7, we establish a feedback loop which we use to improve the 3D poses. Our method compares RGB images and 3D model renderings in a feature space which is optimized for 3D pose refinement and additionally learns to approximate the non-differentiable rasterization backward pass in differentiable rendering. In this way, we precisely align 3D models to objects in RGB images and compute 3D poses which are in many cases visually indistinguishable from ground truth 3D poses.

1.6 Publications

This thesis covers the following peer-reviewed publications which have been accepted at top-tier international conferences in the field of computer vision. The publications are listed in chronological order of acceptance along with their respective abstracts.

3D Pose Estimation and 3D Model Retrieval for Objects in the Wild

Alexander Grabner, Peter M. Roth, and Vincent Lepetit
In: *Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR)*
June 2018, Salt Lake City, USA
(Accepted for poster presentation)

Abstract: We propose a scalable, efficient, and accurate approach to retrieve 3D models for objects in the wild. Our contribution is twofold. We first present a 3D pose estimation approach for object categories which significantly outperforms the state-of-the-art on Pascal3D+. Second, we use the estimated pose as a prior to retrieve 3D models which accurately represent the geometry of objects in RGB images. For this purpose, we render depth images from 3D models under our predicted pose and match learned image descriptors of RGB images against those of rendered depth images using a CNN-based multi-view metric learning approach. In this way, we are the first to report quantitative results for 3D model retrieval on Pascal3D+, where our method chooses the same models as human annotators for 50% of the validation images on average. In addition, we show that our method, which was trained purely on Pascal3D+, retrieves rich and accurate 3D models from ShapeNet given RGB images of objects in the wild.

GP^{2}C: Geometric Projection Parameter Consensus for Joint 3D Pose and Focal Length Estimation in the Wild

Alexander Grabner, Peter M. Roth, and Vincent Lepetit
In: *Proceedings of International Conference on Computer Vision (ICCV)*
October 2019, Seoul, South Korea
(Accepted for poster presentation)

Abstract: We present a joint 3D pose and focal length estimation approach for object categories in the wild. In contrast to previous methods that predict 3D poses independently of the focal length or assume a constant focal length, we explicitly estimate and integrate the focal length into the 3D pose estimation. For this purpose, we combine deep learning techniques and geometric algorithms in a two-stage approach: First, we estimate an initial focal length and establish 2D-3D correspondences from a single RGB image using a deep network. Second, we recover 3D poses and refine the focal length by minimizing the reprojection error of the predicted correspondences. In this way, we exploit the geometric prior given by the focal length for 3D pose estimation. This results in two advantages: First, we achieve significantly improved 3D translation and 3D pose accuracy compared to existing methods. Second, our approach finds a geometric consensus between the individual projection parameters, which is required for precise 2D-3D alignment. We evaluate our proposed approach on three challenging real-world datasets (Pix3D, Comp, and Stanford) with different object categories and significantly outperform the state-of-the-art by up to 20% absolute in multiple different metrics.

Location Field Descriptors: Single Image 3D Model Retrieval in the Wild

Alexander Grabner, Peter M. Roth, and Vincent Lepetit
In: *Proceedings of International Conference on 3D Vision (3DV)*
September 2019, Quebec City, Canada
(Accepted for oral presentation)

Abstract: We present Location Field Descriptors, a novel approach for single image 3D model retrieval in the wild. In contrast to previous methods that directly map 3D models and RGB images to an embedding space, we establish a common low-level representation in the form of location fields from which we compute pose invariant 3D shape descriptors. Location fields encode correspondences between 2D pixels and 3D surface coordinates and, thus, explicitly capture 3D shape and 3D pose information without appearance variations which are irrelevant for the task. This early fusion of 3D models and RGB images results in three main advantages: First, the bottleneck location field prediction acts as a regularizer during training. Second, major parts of the system benefit from training on a virtually infinite amount of synthetic data. Finally, the predicted location fields are visually interpretable and unblackbox the system. We evaluate our proposed approach on three challenging real-world datasets (Pix3D, Comp, and Stanford) with different object categories and significantly outperform the state-of-the-art by up to 20% absolute in multiple 3D retrieval metrics.

Geometric Correspondence Fields: Learned Differentiable Rendering for 3D Pose Refinement in the Wild

Alexander Grabner, Yaming Wang, Peizhao Zhang, Tong Xiao, Peihong Guo, Peter Vajda, Peter M. Roth, and Vincent Lepetit
In: *Proceedings of European Conference on Computer Vision (ECCV)*
August 2020, Glasgow, UK
(Currently under review)

Abstract: We present a novel 3D pose refinement approach based on differentiable rendering for objects of arbitrary categories in the wild. In contrast to previous methods, we make two main contributions: First, instead of comparing real-world images and synthetic renderings in the RGB or mask space, we compare them in a feature space optimized for 3D pose refinement. Second, we introduce a differentiable renderer that learns to approximate the rasterization backward pass instead of relying on a hand-crafted algorithm. For this purpose, we predict deep cross-domain correspondences between RGB images and 3D model renderings in the form of geometric correspondence fields. These correspondence fields serve as pixel-level gradients which are analytically propagated backward through the rendering pipeline to perform gradient descent directly on the 3D pose. In this way, we precisely align 3D models to objects in RGB images which results in significantly improved 3D pose estimates. We evaluate our approach on the challenging Pix3D dataset and achieve up to 55% relative improvement compared to state-of-the-art refinement methods in multiple metrics.

1.7 Outline

The remainder of this thesis is structured as follows:

- In Chapter 2, we explain fundamentals in 3D geometry and deep learning that are required to understand the contributions of this work.
- In Chapter 3, we present an extensive literature review on related work for different 3D vision tasks.
- In Chapter 4, we discuss available datasets consisting of 2D images with 3D annotations that are indispensable for learning to solve tasks from data.
- In Chapter 5, we present our novel 3D pose estimation and 3D model retrieval approach for objects of arbitrary categories. This method exploits the prior given by the 3D pose for retrieval.
- In Chapter 6, we describe ways to improve our 3D pose estimation. In particular, we augment our method to deal with multiple objects in a single image and exploit focal length priors to improve accuracy.
- In Chapter 7, we propose improvements to our 3D model retrieval. For this purpose, we predict pose invariant 3D shape descriptors from both RGB images and 3D model renderings which improves accuracy and reduces database sizes.
- In Chapter 8, we present our differentiable rendering based refinement approach that closed a feedback loop via reprojection to improve our 3D pose predictions.
- Finally, we draw conclusions and discuss future directions in Chapter 9.

CHAPTER 2

Preliminaries

Contents

In this chapter, we explain fundamentals in 3D geometry (see Section 2.1) and deep learning (see Section 2.2) that are required to understand the contributions of this thesis.

2.1 3D Geometry

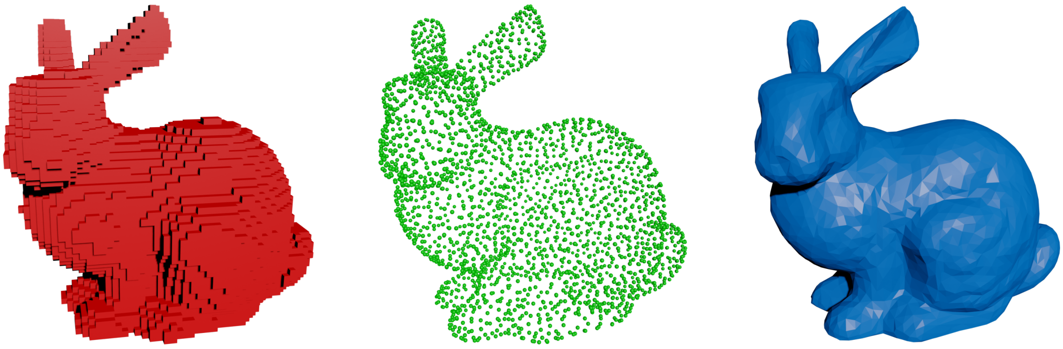
To predict 3D properties from single RGB images, it is mandatory to first understand how 3D models are represented digitally, how 2D images are formed from 3D scenes, and how geometric rules can be exploited to reverse this process. In this context, computer graphics address the process of generating 2D images from 3D information, while 3D vision aims at the inverse process of generating 3D information from 2D images. Thus, 3D vision can be seen as inverse graphics in this case [150, 173].

2.1.1 Computer Graphics

We first discuss the generation of 2D images from 3D information for both real-world scenes and artificial scenes. The presented concepts are also important for computer vision (see Section 2.1.2).

2.1.1.1 3D Models

Digital 3D models resemble three-dimensional entities in the real world. There are multiple different mathematical representations for 3D models that have respective advantages and disadvantages, as shown in Fig. 2.1. Below, we briefly describe the most important representations used in computer graphics and computer vision. For more details, we refer the interested reader to [3].



3D Voxel Grid 3D Point Cloud 3D Triangle Mesh

Figure 2.1: Illustration of different 3D model representations. We show a 3D voxel grid, a 3D point cloud, and a 3D triangle mesh of the Stanford Bunny [280].

3D Voxel Grids: 3D voxel grids are regularly spaced three-dimensional grids, as shown in Fig. 2.1. In most cases, each voxel (*volumentric element*) of a grid takes a binary value indicating if this cell is part of an object or not [4]. This representation is the 3D equivalent to a 2D mask. The level of detail of a 3D voxel grid is limited by the spatial resolution and the memory consumption grows cubically with the spatial resolution. 3D voxel grids can model the interior of objects but otherwise lack flexibility compared to other representations.

3D Point Clouds: 3D point clouds are unordered sets of 3D points, as illustrated in Fig. 2.1. The 3D points usually lie on the surface of objects [95]. Compared to 3D voxel grids, 3D point clouds offer a dynamic level of detail and adaptive memory consumption. However, the 3D points do not form a continuous surface. Thus, generating 2D images from sparse 3D point clouds is difficult.

3D Meshes: 3D meshes can be seen as 3D point clouds with additional connectivity topology to model surfaces, as shown in Fig. 2.1. The simplest and most common form of 3D meshes are triangle meshes [183]. In this case, the connectivity topology defines a set of triangles that form a continuous surface in 3D. Each 3D point, also referred to as vertex, is part of one or more triangles. 3D meshes are extensively used in computer graphics and provide many appealing properties. For example, they are able to compactly represent large flat surfaces, offer a dynamic level of detail, and support shading as well as texturing. In this thesis, we, thus, focus on 3D models in the form of triangle meshes.

2.1.1.2 Projection

Images are 2D projections of the 3D world [65]. In the real world, digital cameras are used to capture images. Digital cameras utilize an array of lenses, an aperture, and a shutter to focus light on a two-dimensional image sensor for a short period of time [126]. Finally, an image signal processing unit computes an image from the measured sensor data [127]. In practice, there are many different types of lens systems, image sensors, and image signal processing units [112]. Thus, physical image acquisition is a complex process.

To model this complex process mathematically, simplified camera models are used in computer graphics and computer vision [95]. A camera model formally describes how 3D points are projected onto the 2D image plane. In this work, we use the commonly employed pinhole camera model, illustrated in Fig. 2.2. This camera model implements a perspective projection with simple mathematics but still accurately approximates many real-world cameras.

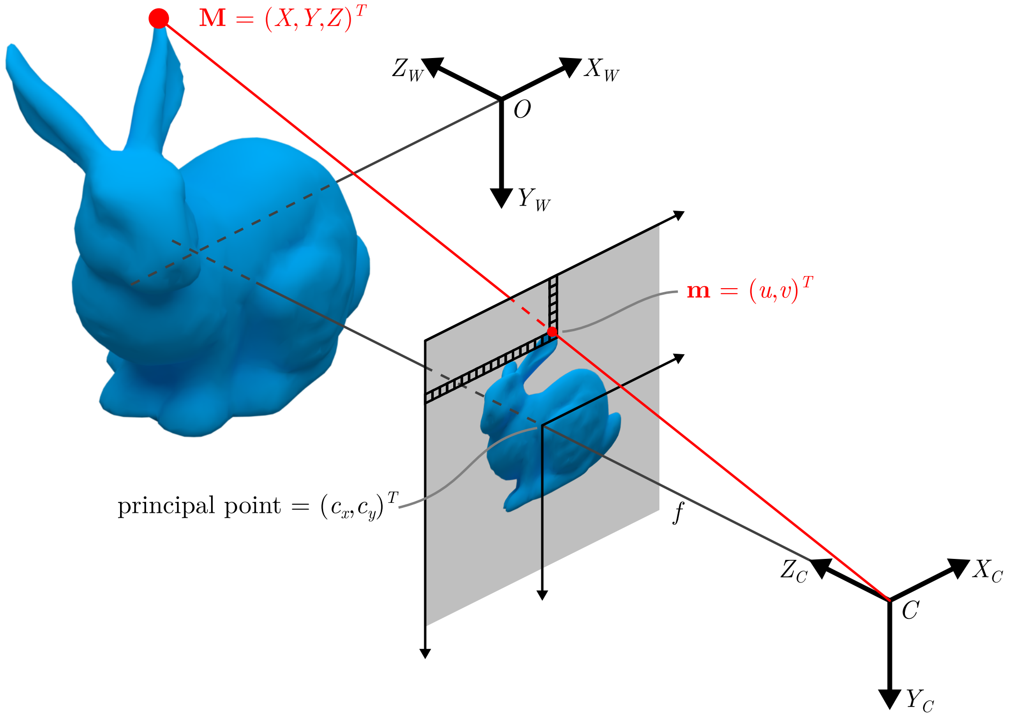


Figure 2.2: Illustration of a pinhole camera model. The 3D point $\mathbf{M}$ is projected to the 2D location $\mathbf{m}$ on the image plane that is located between the 3D scene and the camera eye C. A pinhole camera model implements a perspective projection.

Formally, the perspective projection performed by a pinhole camera model is described as a matrix multiplication

$$\tilde{\mathbf{m}} = \mathbf{P}\tilde{\mathbf{M}}\ , \tag{2.1}$$

where $\tilde{\mathbf{M}} = [\mathbf{M}; 1] = (X, Y, Z, 1)^T$ is a 3D point in homogenous world coordinates and $\tilde{\mathbf{m}} = (U, V, W)^T$ is its 2D projection onto the image plane in homogenous pixel coordinates. To obtain actual pixel coordinates

$$\mathbf{m} = (u, v)^T = (U/W, V/W)^T \tag{2.2}$$

a perspective division of U and V by the homogenous coordinate W is performed. The 3×4 projection matrix

$$\mathbf{P} = \mathbf{K}[\mathbf{R}, \mathbf{t}] \tag{2.3}$$

can be decomposed into a 3×3 intrinsic matrix $\mathbf{K}$ that describes the geometric projection properties of the camera and a 3×4 extrinsic matrix $[\mathbf{R}, \mathbf{t}]$ that describes the 3D position and the 3D rotation of the camera in the world, also referred to as the 3D pose of the camera. In this work, we use a simple parametrization for the intrinsic matrix

$$\mathbf{K} = \begin{bmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{bmatrix}\ , \tag{2.4}$$

where f is the focal length of the camera in pixel units and $(c_x, c_y)^T$ is the principal point of the image that is always assumed at the image center. In this case, f can be interpreted as the distance between the image plane and the camera eye C and the principal point $(c_x, c_y)^T$ lies at the intersection of the camera's view direction ray and the image plane (see Fig. 2.2). Available datasets only provide annotations for this kind of parametrization (see Chapter 4). For completeness, more complex parametrizations of $\mathbf{K}$ including asymmetric focal lengths, skew, or off-center principal point exist. Additionally, there are also non-linear camera models that take lens distortions into account [265].

The extrinsic matrix

$$[\mathbf{R}, \mathbf{t}] = \left[\begin{array}{ccc|c} r_{1,1} & r_{1,2} & r_{1,3} & t_x \\ r_{2,1} & r_{2,2} & r_{2,3} & t_y \\ r_{3,1} & r_{3,2} & r_{3,3} & t_z \end{array}\right] \tag{2.5}$$

concatenates a 3×3 rotation matrix $\mathbf{R}$ and a 3×1 translation vector $\mathbf{t}$. $[\mathbf{R}, \mathbf{t}]$ transforms 3D points in world coordinates to 3D points in camera coordinates. The origin of the camera coordinate system is the camera eye C and the camera is looking in the Z_C direction with $-Y_C$ being the up direction. In this case, $\mathbf{t}$ can be interpreted as the position of the world origin O in camera coordinates and the columns of $\mathbf{R}$ represent the directions of the world axis (X_W, Y_W, Z_W) in camera coordinates.

Three-dimensional rotation matrices have many special properties. Rotation matrices are orthogonal matrices, have a determinant of one, and their transpose is equal to their inverse. Moreover, the set of rotation matrices that describe general rotations around the origin of three-dimensional Euclidean space form the special orthogonal group $SO(3)$, also denoted the 3D rotation group. This implies that the composition of two rotations is also a rotation. While a 3×3 rotation matrix has nine elements, it only has three degrees of freedom. Thus, 3D rotations are often parametrized using more compact representations than rotation matrices:

Euler Angles: In three-dimensional Euclidean space, a general rotation can be described by a series of rotations around the individual axis of the coordinate system. This parametrization, called Euler angles, consists of three rotation angles α, β, and γ, one for each axis. These angles are often referred to as yaw (Z-axis), pitch (Y-axis), and roll (X-axis). The rotation around a specific axis by an angle θ can be implemented as a rotation matrix:

$$\mathbf{R}_X(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\theta) & -sin(\theta) \\ 0 & sin(\theta) & cos(\theta) \end{bmatrix} ,$$

$$\mathbf{R}_Y(\theta) = \begin{bmatrix} cos(\theta) & 0 & sin(\theta) \\ 0 & 1 & 0 \\ -sin(\theta) & 0 & cos(\theta) \end{bmatrix} , \tag{2.6}$$

$$\mathbf{R}_Z(\theta) = \begin{bmatrix} cos(\theta) & -sin(\theta) & 0 \\ sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} .$$

Finally, the matrices for rotations around individual axis can be combined into a general rotation matrix

$$\mathbf{R}(\alpha, \beta, \gamma) = \mathbf{R}_Z(\alpha)\mathbf{R}_Y(\beta)\mathbf{R}_X(\gamma) . \tag{2.7}$$

While the Euler angle representation for three-dimensional rotations has no redundant parameters because it only uses three angles for the three degrees of freedom, it exposes singularities. These singularities occur when the middle rotation aligns the axis of the first and last rotation. In this case, called Gimbal lock, one degree of freedom is lost because the first and last rotation rotate around the same axis [59].

Exponential Maps: Euler's rotation theorem states that any rotation or sequence of rotations around a fixed point is equivalent to a single rotation by an angle θ around an axis running through this fixed point. This is known as an axis-angle representation [265]. One way to parametrize such a representation is a three-dimensional vector $\mathbf{v} = (v_x, v_y, v_z)^T$, where the axis of rotation is given by the direction of the vector and the rotation angle θ is given by the norm of the vector $\|\mathbf{v}\|_2$. Compared to Euler angles, there is no Gimbal lock. However, singularities occur on spheres with radius $2\pi n$ because rotations by a multiple of 2π around any axis are equivalent to no rotation.

In this context, an exponential map transforms a vector $\mathbf{v}$ describing the axis and the magnitude of a 3D rotation to a rotation matrix. This transformation is usually implemented by summing an infinite series of exponentiated skew-symmetric matrices that is evaluated using the compact Rodrigues' formula [89].

Quaternions: Another parametrization for 3D rotations using an axis-angle representation are unit quaternions [4]. A quaternion has four parameters that are represented as

$$\mathbf{q} = a + bi + cj + dk \ , \tag{2.8}$$

where $i^2 = j^2 = k^2 = ijk = -1$. Only quaternions with $\|\mathbf{q}\|_2 = 1$ represent rotations. A quaternion is generated from a rotation axis unit vector $\mathbf{u} = (u_x, u_y, u_z)^T$ and a rotation angle θ as

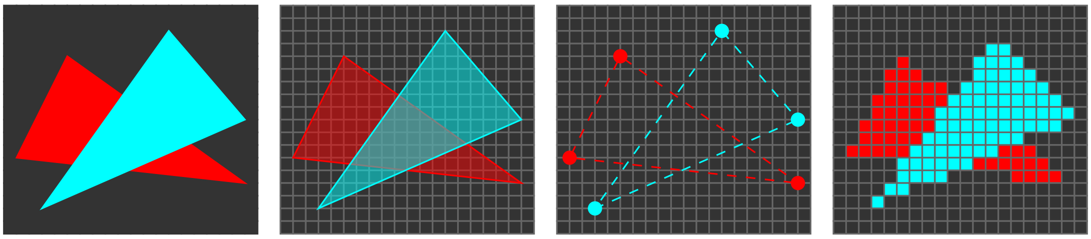
$$\mathbf{q} = cos\left(\frac{\theta}{2}\right) + sin\left(\frac{\theta}{2}\right)(u_x i + u_y j + u_z k) \ . \tag{2.9}$$

In contrast to Euler angles, quaternions do not suffer from Gimbal lock. Compared to axis-angle representations with three parameters, quaternions have four parameters but support algebraic operations on rotations using simple and elegant mathematics. For example, the dot product of two quaternions corresponds to the minimum angular distance between the two respective rotations. Quaternions can also be converted to rotation matrices and vice versa. In this work, we primarily use rotation matrices to execute rotations via matrix multiplications but use the presented parametrizations to efficiently store rotations or evaluate distances between rotations.

2.1.1.3 Rendering

Projecting 3D vertices onto the 2D image plane is not sufficient to generate an image from a 3D model given as a triangle mesh, as illustrated in Fig. 2.3. In particular, the projection only computes a set of 2D locations $\{\mathbf{m}_i\}$ in $\mathbb{R}^2$ from a set of 3D vertices $\{\mathbf{M}_i\}$ in $\mathbb{R}^3$. The so-far unused connectivity topology between the vertices is required to generate an image that correctly shows the surface of a 3D model. Continuous vector primitives in the form of triangles spanned by three 2D locations have to be mapped onto the discretized pixel grid of an image. This process is called rasterization [64].

The rasterization determines which pixels of an image have to be filled and which triangle is visible at a certain pixel. For this purpose, the rasterization algorithm performs pixel tests for each triangle. Specifically, for each triangle the algorithm tests all pixels and identifies if the center of a pixel lies within the triangle. If the center of a pixel lies within a triangle, this pixel has to be filled. However, an issue arises if a pixel is covered by two or more triangles.



Continuous Primitives Discretized Pixel Grid Projected 3D Vertices Shaded Rasterization

Figure 2.3: Illustration of rasterization. Continuous vector primitives in the form of triangles spanned by the 2D locations of three projected 3D vertices have to be mapped onto the discretized pixel grid of an image. The rasterization determines which pixels of an image have to be filled and which triangle is visible at a certain pixel using a depth buffer. Finally, a shading computation is performed to fill the respective pixels.

To correctly deal with overlapping triangles, the rasterization uses a depth buffer [64] The depth buffer has the same spatial resolution as the image and is initialized with a large constant value, e.g., infinity. During rasterization, the depth buffer is continuously updated. If the center of a pixel lies within a triangle, a depth value is computed for this pixel. This depth value is calculated from the depth values of the three projected triangle vertices using perspectively correct interpolation with barycentric coordinates. If the value stored in the depth buffer at this pixel is larger than the computed depth, the depth buffer is updated with the new smaller value and the corresponding pixel of the image is overwritten. The pixels of an image can be filled with arbitrary values like RGB colors, surface normals, or 3D coordinates. The computation of these values is referred to as shading [183]. Shading computations range from the evaluation of simple boolean tests for generating binary masks to complex global illumination [52, 129] equations for generating photo-realistic images.

In practice, rasterization is accelerated by only testing pixels that lie inside the 2D bounding box of a projected triangle. Moreover, the pixel tests for each triangle are parallelized. In the case of parallel triangle processing, it is mandatory to use a mutual exclusion mechanism to control access to both the depth buffer and the image to avoid race conditions [221]. Another common practice is to first solve the visibility of projected triangles and then apply shading for each pixel in a second pass. This technique is known as deferred shading [4]. The entire process of generating a 2D image given a 3D scene and camera parameters is referred to as rendering [183].

Another approach to rendering is ray casting [183]. This technique shoots 3D rays from the camera eye through each pixel center (see Fig. 2.2). For each 3D ray, an intersection test with all 3D triangles is computed. Finally, if there are one or more ray-triangle intersections, the triangle corresponding to the intersection closest to the camera is used to compute a value to fill the respective pixel. However, in most scenarios, projection followed by rasterization is significantly faster than ray casting. Thus, we focus on rendering via projection and rasterization in this work.

Graphics processing units (GPUs) [4] efficiently implement rendering via projection and rasterization in hardware. Modern GPUs even offer the possibility to customize certain stages of the rendering pipeline. For example, stages like the projection or shading can be programmed to perform custom computations. Programs that run on the GPU are called shaders [53]. Shaders are not only useful for rendering but also for general purpose parallel programming [236].

2.1.2 Computer Vision

In Section 2.1.1, we showed how 2D images are generated from 3D information. Now, we discuss strategies to inverse this process and recover 3D information from 2D images.

2.1.2.1 Sparse Correspondence-Based Vision

The geometric relation between a 3D scene and a 2D image is described by the projection matrix $\mathbf{P}$. Thus, many 3D properties can be extracted from $\mathbf{P}$. For example, $\mathbf{P}$ specifies the camera focal length, the 3D pose of the camera in the world, and more.

Camera Calibration: However, in computer vision, $\mathbf{P}$ is usually unknown. Consequently, a popular strategy to recover 3D properties is to compute $\mathbf{P}$ from a set of 2D-3D correspondences $\{\mathbf{m}_i, \mathbf{M}_i\}$ consisting of corresponding 2D locations $\mathbf{m}_i$ and 3D points $\mathbf{M}_i$ [95]. This is also known as camera calibration [276, 317]. Because $\mathbf{P}$ describes a transformation from $\mathbf{M}_i$ to $\mathbf{m}_i$ up to a multiplicative factor, as shown in Eqs. (2.1) and (2.2), a linear system of equations

$$\mathbf{A}\mathbf{p} = \mathbf{0} \tag{2.10}$$

with the parameters of $\mathbf{P}$ as unknowns can be formulated. In this case, $\mathbf{A}$ is a coefficient matrix and $\mathbf{p}$ is a vector with the parameters of $\mathbf{P}$. Each 2D-3D correspondence yields two linearly independent equations. Thus, a minimum of six 2D-3D correspondences is required to solve for the unknowns of $\mathbf{P}$ but more 2D-3D correspondences can be used. A linear least-squares solution that minimizes the algebraic error of this system of equations is obtained via singular value decomposition [80]. This approach is known as the Direct Linear Transformation [2]. Finally, RQ factorization [81] is performed to decompose $\mathbf{P}$ into $\mathbf{K}$, $\mathbf{R}$, and $\mathbf{t}$ to extract different 3D properties. It is also possible to further decompose $\mathbf{R}$ into Euler angles, for example.

In many cases, the initial solution computed by the Direct Linear Transformation is refined using a non-linear optimization [317]. Specifically, the geometric reprojection error between corresponding 2D locations $\mathbf{m}_i$ and projected 3D points $\mathbf{M}_i$ subject to $\mathbf{K}$, $\mathbf{R}$, and $\mathbf{t}$ is minimized as

$$\min_{\mathbf{K},\mathbf{R},\mathbf{t}} \sum_{i=1}^{N} \|\mathbf{m}_i - \text{proj}(\mathbf{M}_i, \mathbf{K}, \mathbf{R}, \mathbf{t})\|_2^2 \, . \tag{2.11}$$

In this case, proj$(\cdot)$ describes the entire projection including matrix multiplication with $\mathbf{P}$ and perspective division. N denotes the number of 2D-3D correspondences. This non-linear least-squares optimization problem is minimized using iterative gradient-based updates, for example, using the Levenberg-Marquardt algorithm [189].

In practice, 2D-3D correspondences are predicted from images [91]. Thus, they are often noisy and contain outliers. While least-squares losses as described above have many beneficial properties in terms of optimization, they are not robust to noise or outliers. To address this issue, robust losses like the Huber loss [118] or the Cauchy loss [274] can be employed instead. These loss functions assign less penalty to large errors, however, their optimization is more complex [40].

Another strategy to deal with outliers is the random sample consensus (RANSAC) [63]. RANSAC is a non-deterministic meta-optimization algorithm that is wrapped around a conventional optimization, e.g., a least-squares optimization, to achieve robustness. In particular, the algorithm first randomly selects a subset of all available data points required to compute a minimal solution, e.g., six 2D-3D correspondences. Second, a solution using this minimal subset is computed. Third, all remaining data points are evaluated against the solution. Depending on a predefined criterion, e.g., a reprojection error threshold, a data point either supports the solution (*inlier*) or not (*outlier*). These steps are repeated until a termination criterion is met, e.g., a maximum number of iterations is reached. In the end, the randomly selected data points of the minimal solution with the largest number of inliers as well as the inlier data points of this solution are used to compute a final solution. Due to random sampling, the computed solution is non-deterministic and can vary between two instantiations of RANSAC. Also, the iterative steps of the algorithm can cause significant computational overhead. Finally, the algorithm is only useful when the total number of data points is larger than the number of data points required for a minimal solution.

Perspective-n-Point: In some cases, the intrinsic matrix $\mathbf{K}$ is known and only the 3D rotation $\mathbf{R}$ and the 3D translation $\mathbf{t}$ need to be computed. This is known as the Perspective-n-Point (PnP) problem [63, 216, 299]. PnP problems are used to calculate the 3D pose of a calibrated camera relative to a scene but also to compute the 3D pose of objects relative to a camera.

In both cases, a minimum of three 2D-3D correspondences is required to solve for the unknowns of $\mathbf{R}$ and $\mathbf{t}$ [68]. However, the minimal solution computed from three 2D-3D correspondences yields up to four geometrically feasible solutions [135]. Thus, an additional constraint, e.g., a fourth 2D-3D correspondence, is required to disambiguate the solutions. Efficient closed-form solutions for the PnP problem in the case of four or more 2D-3D correspondences exist [143, 155].

In practice, PnP approaches use the same high-level optimization strategy as camera calibration approaches. First, a linear solution is computed in closed form. Second, this solution serves as initialization for a non-linear optimization that refines the result using iterative gradient-based updates [155, 176]. Again, robust techniques are utilized to account for noisy 2D-3D correspondences and outliers.

There are also variations of the PnP problem that additionally recover parts of the intrinsic matrix $\mathbf{K}$. For example, the PnPf problem assumes that the camera focal length is unknown [296, 318]. In this case, a minimum of four 2D-3D correspondences is required to solve for the unknowns of $\mathbf{R}$, $\mathbf{t}$, and f. Other variations like the PnPfr problem additionally recover non-linear radial lens distortion parameters [192].

Sparse correspondences have an important role in computer vision. Besides 2D-3D correspondences, 2D-2D correspondences are used to compute stereo depth [292], homographies [95], and fundamental matrices [178]. In this work, we take advantage of the geometric relations encoded by 2D-3D correspondences and solve PnP(f) problems to recover the 3D poses of objects in Chapters 5 and 6.

2.1.2.2 Dense Alignment-Based Vision

The above-described techniques for recovering 3D information from 2D images rely on correspondences that are extracted from images. However, computing robust correspondences from images poses many challenges in practice [265]. For example, homogenous regions or duplicate structures in images make the extraction of correspondences ambiguous. Additionally, specialized system are required to compute 2D-3D correspondences in many cases [22, 217, 317].

Appearance Alignment: One strategy to overcome these limitations is to align 3D models or registered views to images such that their appearance difference in the image space is minimized. In this way, an optimization problem can be formulated directly on an image I opposed to indirectly on a set of 2D-3D correspondences $\{\mathbf{m}_i, \mathbf{M}_i\}$ predicted from an image I. In this context, 3D extensions [37, 113] of the Lucas-Kanade algorithm [177] aim at finding the parameters of a motion model that minimize the discrepancy between an observed image and a stored template to recover 3D pose, for example. In contrast, Active Shape Models [41], 3D Morphable Models [20], and their non-linear extensions [215, 273] optimize for the parameters of trained generative models of statistical variations in shape and appearance to perform image alignment and predict both 3D pose and 3D shape.

Differentiable Rendering: However, a more generic approach to address image alignment is to formulate the entire rendering pipeline as a differentiable function [134, 171]. In this way, a non-linear least-squares optimization problem

$$\min_{\theta} \sum_{x,y} \|I_{x,y} - \mathrm{render}(\theta)_{x,y}\|_2^2 \tag{2.12}$$

in the image space can be efficiently minimized using iterative gradient-based updates that take the entire rendering pipeline into account. In this case, I is an observed image, render$(\cdot)$ is a fully differentiable rendering function that, for example, performs projection and rasterization, (x, y) is a pixel position, and θ is a set of render inputs. In theory, any traditional renderer input can be seen as an optimizable parameter is this formulation. For example, θ can comprise intrinsic parameters like the camera focal length, extrinsic parameters such the camera's 3D pose in the world, 3D scene geometry in the form of different 3D models, multiple light sources, textures, materials, and much more. A differentiable renderer provides a universally applicable inverse graphics tool that avoids the need for complex and specialized correspondence extraction systems. In Chapter 8, we present such a differentiable rendering framework and apply it to 3D pose refinement. However, the performed non-linear optimization requires an initialization just as in the case of correspondences presented above. Such an initialization could be obtained by a deep learning system, for example.

2.2 Deep Learning

Deep learning [82] is a subfield of machine learning [19] which studies algorithms that learn to perform a specific task from data instead of being explicitly programmed. The distinctive characteristic of deep learning systems compared to classic machine learning systems is that they do not only learn a mapping from input to output but also learn an abstract hierarchical representation of the data, as shown in Fig. 2.4.

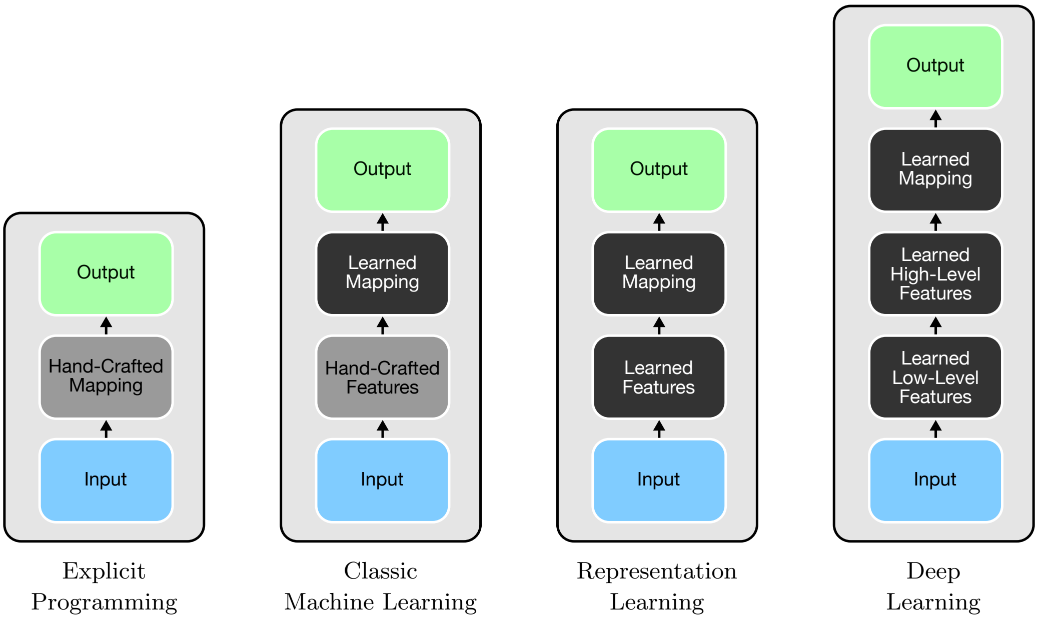


Figure 2.4: Comparison of different machine learning approaches. While classic machine learning systems rely on hand-crafted features which limit performance, deep learning systems learn an abstract hierarchical feature representation of the data. Thus, deep learning is a special form of representation learning.

The performance of learned systems often strongly depends on the representation of the data they are given. For example, directly mapping raw inputs to outputs in a single step is not robust in many cases. Thus, machine learning systems first transform the raw input data to a different representation that eases the task and then perform the mapping to the output (see Fig. 2.4). This intermediate data representation is known as features. While classic machine learning systems rely on hand-crafted features that limit the performance, deep learning systems learn features that are optimized for the task from data. In this context, deep learning refers to the ability to learn an abstract hierarchical feature representation of the input data using a series of transformations, which is a special form of representation learning [82]. Thus, a deep learning system can be seen as a single component that performs a mapping from an arbitrarily shaped input to an arbitrarily shaped output and internally uses a hierarchical feature extraction pipeline optimized for the task to perform robust predictions [146]. This yields an end-to-end trainable system that can be directly optimized for a specific task.

Deep learning techniques have been known for decades [122]. However, only in recent years, the availability of large supervised datasets [165, 233] and advances in both software [1, 5, 203] and hardware [141, 248] have ignited the revolution of the deep learning in practice [245]. Deep learning systems have significantly improved the state of the art across different computing disciplines like speech recognition [107] and image processing [99] and have become the main tool in machine learning today.

2.2.1 Neural Networks

Deep learning systems are typically implemented using artificial neural networks [18]. Neural networks are mathematical models which are loosely inspired by the complex network structures in the human brain. As shown in Fig. 2.5, a neural network represents a directed computational graph consisting of interconnected neurons.

2.2.1.1 Neurons

Neurons are the fundamental structures of a neural network. A neuron typically implements a non-linear transformation

$$y = \sigma(z) = \sigma(\mathbf{w}^T\mathbf{x}) = \sigma\Big(\sum_{n=1}^{N} w_n x_n\Big) \tag{2.13}$$

that computes a weighted linear sum z as the dot product of a vector of inputs $\mathbf{x} = (x_1, x_2, \dots, x_N)^T$ and a corresponding vector of learnable weights $\mathbf{w} = (w_1, w_2, \dots, w_N)^T$ followed by a non-linear activation function $\sigma(\cdot)$. This non-linearity is essential because without it any neuron or neural network would just be a linear regressor. The are many possibilities to design activation functions but they are usually monotonically increasing, continuous, and piecewise differentiable. Popular activation functions are the sigmoid

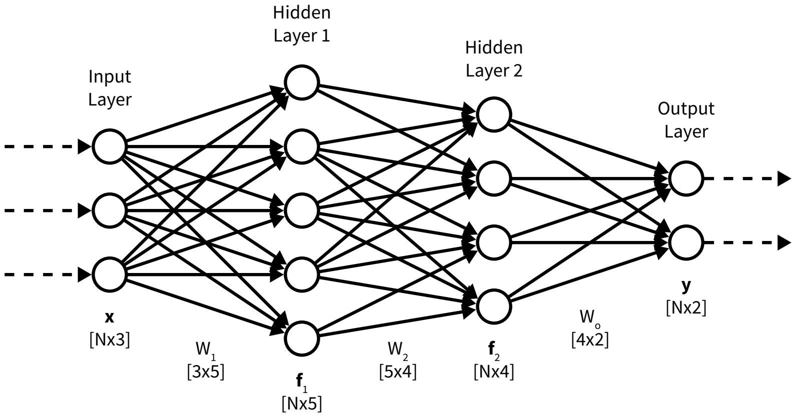


Figure 2.5: Illustration of a neural network with two hidden layers that maps three-dimensional inputs to two-dimensional outputs. All layers of this network are fully connected as each neuron is linked to all neurons in the previous layers. The dimension of the weight matrices $W\_$ and hidden features $\mathbf{f}\_$ are shown for completeness.

function [187] $\sigma(z) = \frac{1}{1+e^{-z}}$, the hyperbolic tangent function [19] $\sigma(z) = tanh(z)$, and the rectifier function [77] $\sigma(z) = \max(0, z)$. Each activation function has unique properties and specific use cases. For more details, we refer the interested reader to [197].

2.2.1.2 Layers

The neurons of a network are organized in layers, as illustrated in Fig. 2.5. Networks typically consist of one input layer, one or more hidden layers, and one output layer. Both the input and the output of a layer can be seen as a vector of features and each hidden layer implements a non-linear feature transformation. In this way, consecutive hidden layers perform a series of multiple non-linear transformations to internally compute an abstract hierarchical feature representation of the input data.

In practice, neurons within a layer compute the same non-linear transformation. As a consequence, the transformation performed by a layer can be formulated as a matrix multiplication of an input vector $\mathbf{x}$ and a weight matrix W followed by an element-wise non-linearity. However, individual layers can implement different transformations. Thus, many networks combine different types of layers that have specific properties and use cases [250, 263].

Fully Connected Layers: A fully connected layer represents the most general type of layer. In this case, each neuron of the layer is connected to all neurons of the previous layer (see Fig. 2.5). This dense connection pattern makes it possible to compute features with global context since every neuron takes all features of the previous layer as inputs. Fully connected layers are often used in the final layer of a network [146].

Convolutional Layers: While fully connected layers offer the beneficial property of computing features with global context, they have a large number of learnable weights and, therefore, are memory inefficient and prone to overfitting. To overcome this limitation, convolutional layers employ a sparse local connectivity pattern opposed to a dense global connectivity pattern. In particular, convolutional layers spatially convolve small filter kernels over their input to compute features. Using filter kernels is a well known and successful strategy in image processing to extract features like edges [281]. Formally, each neuron of a 2D convolutional layer computes

$$y_{i,j} = \sigma\Big(\sum_{m=1}^{M} \sum_{n=1}^{N} \sum_{c=1}^{C} w_{m,n,c} x_{i-\frac{M}{2}+m, j-\frac{N}{2}+n, c} \Big) , \tag{2.14}$$

where $x_{\_,\_,\_}$ is a scalar input sampled from an input tensor with dimensions $H \times W \times C$ like an RGB image, (i, j) is a 2D location in $([1, H], [1, W])$, and $w_{\_,\_,\_}$ is a scalar weight sampled from a filter kernel with dimensions $M \times N \times C$. Typically, the spatial dimensions of the kernel M and N are much smaller than the spatial dimensions of the input H and W. A 2D convolutional layer can also be seen as a special fully connected layer with a weight matrix that only has $M \cdot N \cdot C$ non-zeros values in each row, where each of these values appears in shared form exactly once per row in a different column.

Convolutional layers have three distinctive properties. First, due to the sparse local connectivity pattern convolutional features only consider inputs in a spatial neighborhood. Thus, convolutional layers have a low number of learnable weights. Additionally, they have low computational complexity and can efficiently process large inputs. Second, the same filter kernel is applied at every spatial input location. Hence, the number of learnable weights is further reduced and the implemented transformation can be parallelized for every spatial input location. Third, convolutional layers compute translation equivariant features. This means that if the input spatially shifts in a certain direction, the computed features spatially shift by the same amount. This property is important for localization tasks [198, 225].

While a single convolutional layer can only compute local low-level features, a series of consecutive convolutional layers can compute a hierarchy of features, where the level of abstraction increases with the layer count [152]. Compared to a single convolution, a series of convolutions increases the spatial region of the input that effects a single feature value, also known as the receptive field. In practice, the desire for a large receptive field and a low number of parameters contradict each other and a tradeoff between the two has to be found. The hierarchical feature extraction performed by multiple convolutions layers is inspired by the organization of cells in the first stages of the human visual cortex [117], where early cells respond to simple structures like oriented edges while later cells respond to more complex patterns like faces. Neural networks that employ convolutional layers are referred to as convolutional neural networks (CNNs).

Interpolation Layers: Interpolation layers interpolate features in a local neighborhood. They are used to upsample, downsample, or reshape features. In contrast to fully connected and convolutional layers, interpolation layers do not have learnable weights. Popular interpolation layers are pooling layers [250], nearest-neighbor interpolation layers [72], and bilinear interpolation layers [96].

Other Layers: Apart from the more commonly used layers described above, there are also more specialized layers that, for example, apply a parameter-free non-linearity [82], normalize features [121], or compute an embedding [101].

2.2.1.3 Architectures

Formally, a neural network models a non-linear function

$$\mathbf{y} = f(\mathbf{x}; \theta) \tag{2.15}$$

that maps an arbitrarily shaped input $\mathbf{x}$ to an arbitrarily shaped output $\mathbf{y}$ given a set of parameters θ. In this case, θ comprises the learnable weights of all individual neurons of a network. The implemented function $f(\cdot)$ can be arbitrarily complex and can address both classification and regression problems. In fact, the universal approximation theorem states that a network with a single hidden layer and a finite number of neurons can approximate any continuous function up to a desired precision [114]. Although it is in theory possible to approximate any function using a single hidden layer network up to an epsilon, a large number of neurons in the hidden layer is required in practice. However, research shows that networks with multiple hidden layers empirically require a smaller number of neurons to approximate the same function and often generalize better to unseen data because they internally learn an abstract hierarchy of features [99, 250].

The structure of a network which is also referred to as architecture, i.e., the number of layers and the number of neurons per layer, can be designed arbitrarily depending on the task. In practice, though, many networks show common architecture patterns that can be used to categorize them. For example, many networks are feed-forward networks [315]. Feed-forward networks do not have recurrent connections or form cycles (see Fig. 2.5). Thus, their computational graph is directed and acyclic. This structure eases the optimization compared to networks with recurrent states [107, 139] that are important for dealing with input sequences like videos but not for single image inputs.

2.2.2 Learning

Machine learning systems like neural networks are operated in two phases, i.e., training and inference. During training, a model that learns to perform a specific task from data is trained [234]. For this purpose, the learnable parameters of a model are adjusted to optimize a predefined objective given some training data. In this thesis, we focus on supervised learning [188], where the training data consists of pairs of corresponding inputs and outputs. During inference, the model parameters are frozen and the trained model is used to make predictions, for example, for previously unseen data. In the following sections, we describe the training of neural networks in more detail.

2.2.2.1 Objective

To train a neural network, first an objective also known as loss function is defined [82]. In the case of supervised learning, a loss function

$$L(\mathbf{y}, \hat{\mathbf{y}}) = L\big(\mathbf{y}, f(\mathbf{x}; \theta)\big) \tag{2.16}$$

computes a scalar value that quantizes the error between a ground truth output $\mathbf{y}$ and a prediction $\hat{\mathbf{y}} = f(\mathbf{x}; \theta)$ computed by the model given the corresponding input $\mathbf{x}$ for a single input-output data pair $\{\mathbf{x}, \mathbf{y}\}$. Depending on the task, the loss function can take a variety of different forms. A popular loss function for regression problems is the least-squares loss

$$L(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{2} \|\mathbf{y} - \hat{\mathbf{y}}\|_2^2 \ , \tag{2.17}$$

where the term $\frac{1}{2}$ is used as a scaling constant that reduces the gradient to the residual $\|\mathbf{y} - \hat{\mathbf{y}}\|_2$. However, similar to optimization problems in geometry (see Section 2.1.2.1), robust losses like the Huber loss [118] or the Cauchy loss [274] are often employed to increase the robustness to outliers in the data. A popular loss function for classification problems is the cross-entropy loss

$$L(\mathbf{y}, \hat{\mathbf{y}}) = -\sum_{c=1}^{C} y_c \log(\hat{y}_c) \ , \tag{2.18}$$

where $\mathbf{y}$ is a one-hot vector of length C that represents the ground truth class probability distribution and $\hat{\mathbf{y}}$ is a vector of length C that represents the predicted class probability distribution. In this case, C is the number of classes, while y_c and $\hat{y}_c$ represent the ground truth and predicted probability for a certain class. However, the output of a neural network is typically not a valid class probability distribution in which, for example, all individual class probabilities sum to one and there are no negative values. To overcome this limitation, a softmax normalization

$$\hat{y}_c = \frac{\exp(\tilde{y}_c)}{\sum_{i=1}^{C} \exp(\tilde{y}_i)} \tag{2.19}$$

is applied to all raw network outputs. In this case, $\hat{y}_c$ is a normalized output and $\tilde{y}_c$ is the corresponding raw output. This normalization makes it possible to interpret each element $\hat{y}_c$ of a predicted output $\hat{\mathbf{y}}$ as the probability $p(c|\mathbf{x})$ of a certain class c given the input $\mathbf{x}$.

2.2.2.2 Optimization

In the context of supervised neural network training, the goal of optimization is to find a set of network parameters θ which minimizes a defined loss function given a set of input-output training data pairs $\{\mathbf{x}_i, \mathbf{y}_i\}$. Neural networks implement highly non-linear and non-convex functions. Consequently, there is no closed-form solution to compute an optimal set of parameters θ given a loss function and training data pairs. However, the individual transformations of a neural network are differentiable. Thus, optimization can be performed using iterative gradient-based updates.

Backpropagation: Backpropagation [232] describes an algorithm to compute the gradient of a loss function $L\big(\mathbf{y}, f(\mathbf{x}; \theta)\big)$ with respect to θ for a given training data pair $\{\mathbf{x}, \mathbf{y}\}$. In particular, backpropagation makes use of the chain rule to calculate the gradient for each individual parameter in θ. The algorithm uses dynamic programming to efficiently compute gradients and to avoid redundant calculations of intermediate terms.

Gradient Descent: Backpropagation only computes gradients for the parameters in θ but does not perform optimization. In order to actually optimize the parameters, an iterative gradient-based optimization algorithm is applied. Gradient descent [18] iteratively updates parameters according to the negative direction of the gradient:

$$\theta^{k+1} = \theta^k - \eta \cdot \nabla_\theta L\big(\mathbf{y}, f(\mathbf{x}; \theta)\big) \ . \tag{2.20}$$

In this case, k is the iteration index, η is the update step size also referred to as learning rate, and $\nabla_\theta L(\cdot)$ is the gradient of a loss function with respect to θ. To compute an update direction for multiple training data pairs, the gradients $\nabla_\theta L(\cdot)$ of individual data pairs $\{\mathbf{x}_i, \mathbf{y}_i\}$ are averaged.

Initialization: Iterative optimization algorithms like gradient descent require an initial estimate that serves as a starting point for the optimization. Typically, neural network parameters are initialized using random sampling schemes [76, 97, 242]. In practice, another common strategy is to reuse parts of a network trained for a different task [217, 225, 279]. This is known as transfer learning [314].

2.2.2.3 Regularization

However, learning differs from optimization [82]. While the goal of optimization is to find parameters that minimize a loss function for some given data, the goal of learning is to additionally perform well on previously unseen data in the future. Thus, learning aims at optimizing for the entire underlying data distribution rather than for a finite set of training data pairs.

To achieve learning via optimization, regularization is essential [19]. In fact, regularization is omnipresent in deep learning. For example, in practice, neural networks are optimized using regularized variants of gradient descent. In contrast to standard gradient descent that considers the entire training data set for computing an update direction in each iteration, stochastic gradient descent [21] approximates the true gradient of the training data set by only considering a randomly selected subset of all available data pairs also known as minibatch. This approach has three benefits. First, the approximated updates have a regularizing effect and avoid overfitting. Second, the computational complexity for calculating a single update is significantly reduced. Third, the stochastic updates reduce the risk of getting stuck in poor local minima and make it possible to escape from shallow basins in the high-dimensional loss surface. Additionally, there are extensions to stochastic gradient descent that use heuristics to accelerate the optimization and avoid jittering gradients like Momentum [214] and Nesterov Momentum [260] or dynamically adjust the learning rate like Adam [138].

During optimization, a regularizing effect can also be achieved by early stopping [19], where the training is discontinued ahead of schedule if the error on a validation set increases. Regularization can also be encoded in the loss function by adding Lagrange multipliers to the training objective. For example, weight decay [147] follows the principle of Occam's razor and encourages the norm of learned weights to be small, which results in simpler models.

Another way to achieve regularization is to increase the size and the diversity of the training data set [233]. A larger training data set provides a better approximation of the true underlying data distribution and results in improved generalization. Common tricks to avoid the costly collection of additional real data are data augmentation [88, 145] and synthetic data [257, 271].

Specific structures within a network can also have a regularizing effect. For example, a low number of network parameters decreases the risk of overfitting. Also, parameter sharing as performed in convolutional layers increases generalization. Further, feature normalization is a form of regularization [121, 235, 300].

Finally, different ensemble techniques can be used to achieve regularization. This ranges from ensembles of predictions, e.g., a combination of predictions from multiple random crops of an image, over ensembles of networks, to intra-network ensembles such as dropout [254].

2.2.2.4 Hyperparameters

Neural networks have many hyperparameters that can significantly effect the performance of trained models. Hyperparameters include the loss function, the optimizer, the parameter initialization, the learning rate, the activation functions, the employed regularization techniques, the minibatch size, the number of gradient-based updates to the model, the performed data augmentation, the spatial resolution of input images, the floating-point precision of the model, and much more [82]. A consensus between all hyperparameters is required to achieve high generalization. In practice, this consensus is found via empirical trial-and-error or exhaustive meta optimization.

The architecture of a network can also be seen as a hyperparameter. The structure of a network is typically designed by researchers based on empirical results and provides a strong prior that might be suboptimal for the task. To overcome this limitation, recent approaches perform meta optimization in the form of neural architecture search [54] to learn architectures that are optimized for the task.

CHAPTER 3

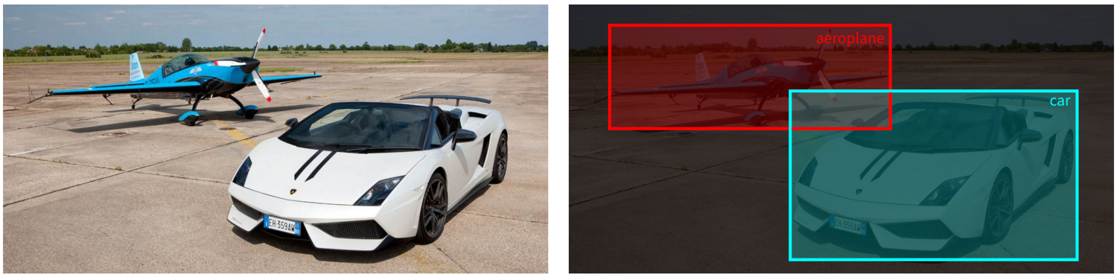
Related Work

Contents

In this chapter, we present an extensive literature review on prior research and related work for different 3D vision tasks. In particular, we formally describe and discuss object detection (see Section 3.1), 3D pose estimation (see Section 3.2), 3D model retrieval (see Section 3.3), the estimation of camera intrinsics (see Section 3.4), and differentiable rendering (see Section 3.5).

3.1 Object Detection

Object detection aims at predicting the presence as well as the 2D location of objects in images. Predictions are typically expressed as 2D bounding boxes (x, y, w, h) with an accompanied class label for each bounding box, as shown in Fig. 3.1. Object detection has many applications in computer vision, including surveillance [45, 246], face detection [284], and general 2D scene understanding [61]. Moreover, object detection is often a first step towards solving higher-level tasks like 3D pose estimation [86] or 3D model retrieval [87].



Image

Object Detection

Figure 3.1: Example of object detection. Predictions are typically expressed as 2D bounding boxes (x, y, w, h) with an accompanied class label for each bounding box. The 2D bounding boxes of individual objects can overlap, which results in ambiguity[1].

Templates: Early approaches to object detection are based on template matching [29]. In this case, a reference pattern is compared to every subregion of an image in a sliding window fashion [127]. If the sum of squared differences between the template and an image window is below a certain threshold, an object is detected. Comparing the template against an image pyramid [182] makes it possible to detect objects at different scales. However, simple template matching in the image space is computationally expensive and not robust to any appearance variations and, thus, performs poorly in practice.

Machine Learning: To overcome these limitations, classic machine learning approaches train a classifier on top of hand-crafted features instead of raw pixel intensities and employ various speed-up mechanisms. The Viola-Jones detector [284], for example, uses an integral image to efficiently compute Haar features [201] and uses Adaboost [67] to both perform feature selection and train an efficient detection cascade for real-time face detection. In contrast, the HOG detector [45] uses a linear SVM [42] on top of more discriminative features in the form of histograms of oriented gradients computed on a dense grid of uniformly spaced cells with overlapping local contrast normalization. Deformable part models [61, 74] extend the HOG detector and use a divide-and-conquer strategy to learn a decomposition of objects into individual parts during training. During inference, root and part detection results for an object are combined using a spatial deformation cost to compute a final detection score. However, the performance of detection approaches based on classic machine learning is limited because they rely on hand-crafted features that are not optimized for the actual task.

Deep Learning: In contrast to classic machine learning approaches, deep learning approaches use convolutional neural networks to learn an abstract hierarchy of features optimized for the task of object detection. In this way, they have significantly improved the state of the art in object detection in recent years. Deep learning approaches to ob-

[1] `https://www.evo.co.uk/lamborghini/gallardo/11857/video-lamborghini-vs-stunt-plane`

ject detection can be categorized into two-stage detectors [72, 73, 98, 225] that implement a coarse-to-fine strategy and one-stage detectors [169, 222–224] that directly aim for a fine-grained prediction.

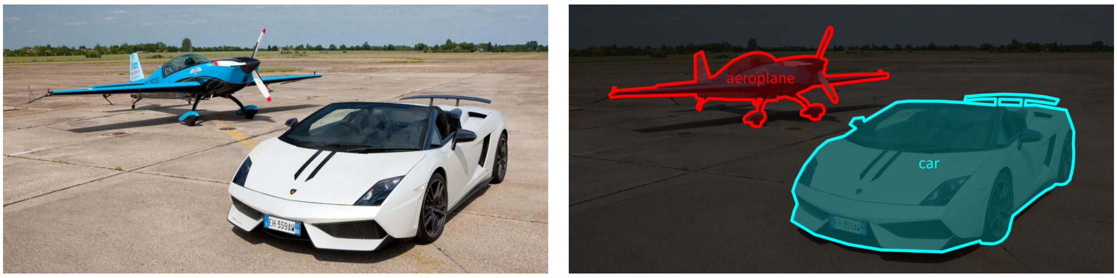
Two-stage detectors typically achieve higher accuracy but also have a higher computational complexity. R-CNN [73], for example, uses selective search [283] to identify regions of interest and classifies each region of interest using a CNN pre-trained for ImageNet classification [146]. To overcome the computational bottleneck of redundant feature calculations for overlapping regions of interest, SPPNet [98] performs a single CNN forward pass to compute features for the entire image and then uses spatial pyramid pooling to extract a fixed-size representation for arbitrarily shaped regions of interest. Fast R-CNN [72] further develops this idea and introduces a region of interest pooling operation to optimize all network layers for object detection. Next, Faster R-CNN [225] introduces a region proposal network which eliminates the need for computing regions of interest in a pre-processing step and provides an end-to-end trainable model for object detection. Finally, Feature Pyramid Networks [163] combine low-level and high-level features from different layers to improve the detection of objects at different scales.

In contrast to two-stage detectors, one-stage detectors compromise in terms of accuracy while significantly improving in terms of speed. For example, YOLO [222] and successors [223, 224] use a single network that simultaneously divides the image into regions and predicts bounding boxes and class probabilities for each region. In this context, SSD [169] presents a multi-resolution technique that detects objects of different scales at different network layers. To address the imbalance of foreground and background regions in training one-stage detectors, RetinaNet [164] presents a modified cross-entropy loss that puts more emphasis on misclassified training samples.

Backbones: The performance of object detectors based on deep learning strongly depends on the used network architecture also referred to as the backbone of a detector. Due to the task similarities between detection and classification, many detection approaches build on networks designed and pre-trained for ImageNet classification [233]. Empirical results show, that architectures that perform well for classification tend to perform well for object detection with a general trend towards more layers and more regularization in networks [99, 100, 146, 250, 262, 307]. To further improve the performance, neural architecture search [54] for object detection is performed [36].

Instance Segmentation: To detect objects even more precisely, recent approaches perform instance segmentation [96, 168]. In contrast to object detection which only provides 2D bounding boxes for detections, instance segmentation additionally computes a binary mask for each detected object, as shown in Fig. 3.2. In this way, objects are localized more accurately and can be detached from their background to ease subsequent tasks [125].

Finally, a recent trend in this field is to directly integrate higher-level tasks with object detection or instance segmentation using a single network [75]. The resulting multi-task



Image

Instance Segmentation

Figure 3.2: Example of instance segmentation. Predictions are typically expressed as binary masks with an accompanied class label for each mask. In this way, objects are localized more accurately compared to object detection and can be detached from their background.

network address various tasks using shared optimized features which increases generalization, reduces the parameter count, and shortens the training time. In this context, we directly integrate 3D pose estimation and 3D model retrieval with object detection to deal with multiple objects in a single image in Chapters 6 and 7.

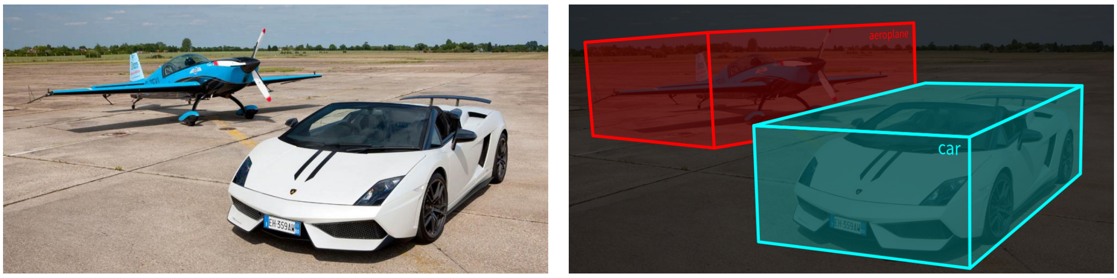
3.2 3D Pose Estimation

3D pose estimation addresses the task of predicting the 3D rotation and 3D translation of objects relative to the camera. In some cases, only the 3D rotation is predicted. This task is referred to as 3D viewpoint estimation. In other cases, the 3D dimensions of an object are additionally predicted. This task is referred to as 3D object detection. An overview of the three tasks and their respective degrees of freedom is presented in Table 3.1. For all three tasks, predictions are typically visualized using 3D bounding boxes, as shown in Fig. 3.3.

Task	3D Rotation	3D Translation	3D Dimensions	Degrees of Freedom
3D Viewpoint Estimation	✓			3
3D Pose Estimation	✓	✓		6
3D Object Detection	✓	✓	✓	9

Table 3.1: Overview of 3D viewpoint estimation, 3D pose estimation, and 3D object detection.

In the literature, most 3D pose estimation approaches are based on some form of 2D-3D correspondences [255]. For example, the 3D pose of an object can be recovered from correspondences between 3D model points and their respective 2D image locations using a geometric optimization, as already discussed in Section 2.1.2.1. However, while solving correspondence-based geometric optimization problems is not an issue in practice, establishing 2D-3D correspondences in the first place is challenging.



Image

3D Pose Estimation

Figure 3.3: Example of 3D pose estimation. Predictions are typically visualized using 3D bounding boxes. Because the 3D configuration of multiple objects is known, overlapping issues between projected 3D bounding boxes can be resolved correctly.

2D-3D Edge Correspondences: Early approaches to 3D pose estimation are based on corresponding edges between rendered 3D models and observed 2D images. In this context, RAPID [93] aligns selected control points on high contrast edges of 3D models to edges in images. Similarly, [174] minimizes the distance from points on image edges to curve segments of projected 3D models and takes inherent inaccuracies in the image measurements into account. [50] performs 3D pose estimation via constrained active contour tracking. [130] uses correspondences between different geometric primitives such as points, lines, and ellipse-circle pairs to improve robustness. However, these methods require an initial 3D pose which is close to the ground truth.

2D-3D Point Correspondences: To overcome this limitation, many approaches establish 2D-3D point correspondences instead. These approaches usually have an offline phase and an online phase. In the offline phase, corresponding 3D model points and 2D image locations are selected in registered images, a 2D keypoint descriptor is computed for each 2D image location, and a database of 2D keypoint descriptors with associated 3D model points is created. In the online phase, 2D keypoints are detected, 2D keypoint descriptors are computed, and 2D-3D correspondences are established by retrieving 3D points from the database. For example, [220] manually selects 2D keypoints in several registered reference images of an object and uses small local image patches as 2D keypoint descriptors. [251] extracts SIFT [175] features from multiple images of an object, uses structure from motion [95] to reconstruct 3D points, and creates a database from these correspondences. In contrast, [270] learns a database of SURF [15] descriptors which are robust to strong perspective changes from a textured 3D model. [154] detects 2D keypoints and performs 2D keypoint classification using randomized trees to predict 2D-3D correspondences instead. [202] uses a combination of 2D keypoint matching and tracking to increase speed. Finally, [282] combines information provided by edges and Harris [94] keypoints to increase robustness. However, methods based on traditional 2D keypoints only perform well for textured objects.

Templates: Thus, another line of research focuses on template-based approaches. In this context, [131] performs 2D template matching under geometric template deformations. [104] represents objects with multiple templates based on gradient orientations computed from registered images, where each template is associated with a 3D pose. In contrast, [105] builds templates from textured 3D models. [205] uses shape templates based on object contours. [162] uses LDA [186] on HOG features [45] computed from local edge maps to predict correspondences and measures global alignment of 3D poses. [209] extends 2D deformable part models [61] to additionally predict 3D viewpoints. [305] combines a deformable part model with a 3D aspect layout model to estimate continuous 3D viewpoints. [161] uses a deformable part model to learn shared 3D parts and predict 3D poses. [293] maps images of objects to a descriptor space that is discriminative in terms of 3D shape and 3D pose and performs retrieval in this space to predict both quantities. [259] trains an augmented auto-encoder and matches latent codes for 3D pose retrieval. However, template-based methods often require large databases of templates and can only interpolate between reference 3D poses.

Direct Estimation: Inspired by the success of end-to-end deep learning for tasks like classification [233], recent approaches directly predict 3D pose parameters from RGB images. In this context, numerous works only perform 3D viewpoint estimation using CNNs. These methods predict the 3D rotation of objects parametrized as three Euler angles, a three-dimensional axis-angle vector, or a quaternion using regression [184, 302], classification [277, 279], or hybrid variants of both [179, 306]. For example, [302] directly regresses Euler angles using a CNN. [184] compares different variants and presents a regression approach which parameterizes each Euler angle using trigonometric functions. [277] and [279] perform 3D viewpoint classification by discretizing the range of each Euler angle into a number of disjoint bins and predicting the most likely bin using a CNN. [257] uses a fine-grained geometric structure-aware classification, which encourages the correlation between bins of nearby 3D viewpoints. [179] formulates the task as a hybrid classification-regression problem using an axis-angle representation: In addition to 3D viewpoint classification, a residual 3D rotation is regressed for each angular bin using a delta-bin approach. [306] also employs a delta-bin approach but estimates Euler angles and additionally leverages a 3D model of the object in the image. However, predicting a full 3D pose opposed to a 3D viewpoint is desirable for many applications.

Thus, many approaches combine the 3D rotation estimation techniques described above with 3D translation estimation. [190] uses a simplified 3D pose parametrization consisting of two Euler angles and three translation parameters in a coarse-to-fine approach that performs 3D pose classification followed by 3D model alignment. [156] performs hybrid classification-regression for both 3D rotation and 3D translation using a delta-bin approach. [191] performs 3D object detection using a delta-bin approach for 3D rotation, a regression approach for 3D dimensions, and geometric constraints on the 2D bounding box to recover 3D translation. However, these approaches cannot deal with multiple objects in a single image nor handle objects at different scales, thus, they require object detection in a preprocessing step.

To overcome these limitations, recent approaches integrate 3D pose estimation techniques into object detection pipelines making the entire system end-to-end trainable. [136] augments SSD [169] with 3D rotation classification and recovers 3D translation from 2D bounding box constraints. Similarly, [151] extends Faster R-CNN [225] with delta-bin 3D rotation estimation and additional 3D shape prediction. [304] performs semantic segmentation for object detection, object center Hough voting [14] for 3D translation estimation, and quaternion regression for 3D rotation estimation using a single network. [288] augments Mask R-CNN [96] to regress a quaternion as well as the object's principal point and distance from the camera for 3D pose estimation. However, end-to-end deep learning for 3D pose estimation has not seen the same success as end-to-end deep learning for 2D recognition tasks like classification [233]. This can be accounted to the lack of supervised training data in the form of 2D images with 3D annotations [258], overfitting to random structures in the data, and network architectures that are not optimized for 3D tasks [56, 277, 295]. Additionally, simple feed-forward CNNs cannot exploit prior knowledge about the task, for example, in the form of geometric rules or information about the camera intrinsics, which is a significant drawback.

Learning and Geometry: To leverage the benefits of both learning and geometry, recent approaches predict 2D-3D point correspondences using learning and recover 3D poses using geometric optimizations (see Section 2.1.2.1). In this context, multiple approaches use CNNs to predict the 2D image locations of sparse category-specific 3D model points from RGB images. For example, [208] recovers the 3D pose of an object from predicted 2D locations of manually selected 3D model points using a PnP algorithm. Similarly, [204] also predicts 2D locations using a CNN but recovers 3D poses using a trained deformable shape model. [279] presents an approach that combines local 2D location estimates with a global 3D viewpoint estimate. [207] uses a pixel-wise voting for generating heatmaps of 2D locations in combination with semantic segmentation to address partial occlusions. However, these approaches rely on category-specific 3D model points which need to be selected and annotated manually for each 3D model.

In contrast, other approaches predict the 2D image locations of sparse category-agnostic virtual 3D points that can be inferred automatically from 3D models. For example, [43] predicts the 2D locations of virtual 3D points to estimate the 3D pose of object parts. [217] predicts the 2D locations of the 3D bounding box corners of an object. [269] integrates the technique presented in [217] with the YOLO [222] object detector to deal with multiple objects in a single image and handle scale variations. [198] also predicts the 2D locations of 3D bounding box corners but uses a patch-based CNN that generates heatmaps to increase the robustness to partial occlusions. However, these approaches require the 3D model of an object to be known during inference to recover the 3D using a PnP algorithm.

To avoid providing ground truth 3D models at runtime, multiple approaches predict both 2D image locations and corresponding 3D model points in the form of dense unsupervised 2D-3D correspondences. In this case, a natural choice is to segment pixels that belong to an object and predict a 3D model point for each segmented pixel. [22] uses random forests [28] to predict such 2D-3D correspondences. [24] extends this approach and uses predicted uncertainties to refine 3D poses. [125] uses Mask R-CNN [96] to segment instances and a second custom CNN to regress 3D model points. In contrast, [288] integrates 3D model point regression with Mask R-CNN, while [285] integrates 3D model point classification with Mask R-CNN. In Chapters 5 and 6, we present methods that also predict 2D-3D correspondences using deep learning and recover 3D poses using geometric optimizations. However, while these feed-forward approaches robustly estimate the coarse high-level 3D rotation and 3D translation of objects, they cannot predict fine-grained 3D poses in many cases [306].

3D Pose Refinement: To improve the 3D pose accuracy, refinement approaches try to align 3D models to objects in RGB images. In particular, 3D pose refinement approaches are based on the assumption that the projection of an object's 3D model aligns with the object's appearance in the image given the correct 3D pose. Thus, they compare renderings under the current 3D pose to the input image to get feedback on the prediction. 3D pose refinement approaches relate back to edge alignment between rendered 3D models and observed 2D images discussed in the beginning of this section [50, 93, 174].

In a similar fashion, region-based 3D pose refinement methods optimize energy functions to refine the alignment of rendered 3D models and objects in 2D images for selected regions. [243] optimizes an energy function defined on the 3D pose parameters to align the contours of rendered 3D models and observed 2D images. [210] minimizes an energy function based on foreground and background membership probabilities of each pixel to refine 3D poses. However, the formulated energy functions are not directly optimizable.

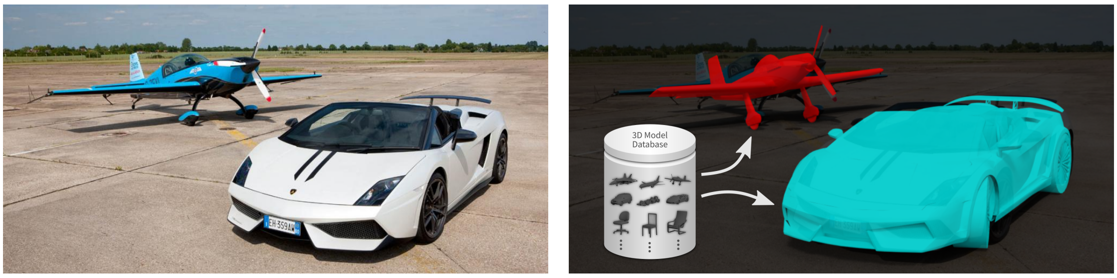
Another strategy to refine 3D poses is to generate many small perturbations and evaluate their accuracy using a scoring function. [311] uses normalized cross-correlation in the intensity domain to evaluate the accuracy of a hypothesis. [166] trains a CNN that scores in-painted 3D bounding boxes. However, this approach is computationally expensive and the design of a robust scoring function is unclear.

Therefore, other approaches try to predict iterative 3D pose updates with deep learning systems instead. These approaches use pairs of RGB images and 3D model renderings as network input. [217] performs refinement by predicting 2D updates to previously predicted 2D locations of 3D bounding box corners using a custom network. [181] directly estimates 3D pose updates using a frozen Inception [262] architecture with custom fine-tuned output layers. [159] predicts 3D pose updates using a network architecture designed for optical flow estimation [48]. In contrast, [313] predicts 3D pose updates using a ResNet-like architecture [99]. In practice, though, the performance of these methods is limited because they cannot generalize to 3D poses or 3D models that have not been seen during training [217].

Recent approaches based on differentiable rendering overcome these limitations. Compared to the methods described above, they analytically propagate error signals backward through the rendering pipeline to compute more accurate 3D pose updates. In this way, they exploit knowledge about the 3D scene geometry and the projection pipeline for the optimization. While [171] relies on comparisons between images and renderings in the RGB space, [200] and [134] rely on comparisons in the mask space. In contrast, we present a refinement method based on differentiable rendering that compares images and renderings in a feature space optimized for 3D pose refinement in Chapter 8.

3.3 3D Model Retrieval

3D model retrieval addresses the task of predicting 3D models for objects in images. In particular, 2D image observations of objects are used to retrieve 3D models which accurately represent the geometry of the object from databases of 3D models [33, 258], as shown in Fig. 3.4. For this task, only the 3D shape of objects is considered, while colors and textures are ignored because color information can be extracted from images and applied to registered 3D models easily [265]. In theory, a ranked list of all available 3D models from a database is generated for each query. In practice, however, only the highest-ranked or approximately highest-ranked 3D model is returned to speed up retrieval for large databases.



Image

3D Model Retrieval

Figure 3.4: Example of 3D model retrieval. For each object, a 3D model which accurately represents the geometry of the respective object is retrieved from a database of 3D models. In this case, the predicted 3D models are projected onto the image using the 3D poses of the objects.

Retrieval versus Reconstruction: 3D model retrieval differs from 3D model reconstruction [95] which also addresses the task of predicting 3D models for objects in images. In particular, 3D reconstruction directly estimates geometry from 2D image observations [255, 265]. In the context of single image 3D model reconstruction for objects, there are approaches that predict voxel grids [144, 278, 297, 316], unstructured 3D point clouds [58, 180, 193], structured 3D point clouds [22, 24, 268, 285], and 3D meshes [75, 90, 252, 286, 308]. Naturally, many concepts used in reconstruction approaches are also found in retrieval approaches. However, because 3D model retrieval databases consist of 3D meshes designed by humans, retrieved 3D models have several advantages compared to reconstructed 3D models in practice. First, retrieved 3D models have a high level of detail while having a low polygon count. Second, they obey symmetries and provide correct geometry even for occluded or truncated object parts. Third, they are often composed of hierarchical parts and model interior object structures. Finally, they have pre-defined materials that can be used to realistically change the appearance of an entire 3D model using color information extracted from images.

3D queries: Many approaches perform 3D model retrieval given a query 3D model [240, 241]. In this case, feature vectors in the form of 3D shape descriptors are computed from both the query 3D model and the 3D models in the retrieval database. Then, a distance between the query descriptor and each database descriptor is computed [149]. Finally, the 3D model from the database corresponding to the smallest computed distance is returned. To improve speed for large databases, also an approximate nearest neighbor in the descriptor space can be returned [6]. In this context, different approaches either directly operate on 3D data, e.g., in the form of 3D voxel grids [211, 298], spherical maps [55], or 3D point clouds [142, 212, 213], or process multi-view renderings of the query 3D model [13, 34, 211, 256] to compute 3D shape descriptors. However, in this work, we focus on the much more challenging task of 3D model retrieval given a single RGB image [85, 133]

Classification: One approach to perform 3D model retrieval given a single RGB image is to train a classifier that provides a 3D model for each class. In this context, [8] uses a linear classifier on top of handcrafted features extracted from an image to predict both 3D shape and 3D viewpoint. In contrast, [190] trains a CNN that performs fine-grained category classification and provides a 3D model for each category. However, 3D model retrieval via classification does not scale and is limited to 3D models that have been seen during training.

Embedding Spaces: A popular strategy to overcome the limitations of retrieval via classification is to map 3D models and RGB images to feature vectors in a common embedding space. In this space, 3D model retrieval is performed using distance-based matching [149] between feature vectors that represent 3D shape descriptors just as described for 3D model queries above. In this case, the mapping, the embedding space, and the distance measure can be designed in a variety of ways.

For example, numerous works match features extracted from an RGB image against features extracted from multi-view RGB renderings to predict both 3D shape and 3D viewpoint. In this context, [10] uses a CNN trained for ImageNet classification [233] to extract features. [185] takes a similar approach but additionally performs non-linear feature adaption to overcome the domain gap between real and rendered RGB images.

[116] and [123] use a CNN trained for object detection as a feature extractor. However, the CNNs used in these approaches are not optimized for 3D model retrieval.

Thus, other approaches train mappings to predefined embedding spaces. [266] trains CNNs to map 3D models, RGB images, depth maps, and sketches to an embedding space based on text for cross-modal retrieval. [119] constructs a low-dimensional embedding space by performing PCA on 3D model points and maps 3D model points predicted using a CNN to that space for retrieval. [158] and [267] train a CNN to map RGB images to an embedding space constructed from pairwise similarities between 3D models.

Instead of handcrafting an embedding space, an embedding space capturing 3D shape properties can also be learned. [278] reconstructs voxel grids from RGB images of objects using CNNs. The low-dimensional bottle-neck 3D shape descriptors are also used for retrieval. [71] combines a 3D voxel encoder and an RGB image encoder with a shared 3D voxel decoder to perform reconstruction from a joint embedding. 3D model retrieval is performed by matching embeddings of 3D voxel grids against those of RGB images.

Finally, recent approaches explicitly learn an embedding space which is optimized for 3D model retrieval. [302] uses a single CNN to map RGB images and RGB renderings to an embedding space which is optimized using a Euclidean distance-based lifted structure loss [199]. At test time, the distances between an embedding of an RGB image and embeddings of multi-view RGB renderings are averaged to compensate for the unknown 3D pose of objects. [153] uses two CNNs to map RGB images and gray-scale renderings to an embedding space and optimizes a Euclidean distance-based Triplet loss [290]. Additionally, cross-view convolutions are employed to aggregate a sequence of multi-view renderings into a single 3D shape descriptor to reduce the matching complexity. In Chapters 5 and 7, we also present methods that explicitly learn an embedding space from data but further perform efficient and scalable 3D model retrieval.

3.4 Estimation of Camera Intrinsics

The estimation of camera intrinsics, also referred to as camera calibration, has a long tradition in computer vision [59, 95]. In this work, we use a simple parametrization for the camera intrinsics consisting of a single parameter, i.e., the camera focal length (see Section 2.1.1.2). This choice is motivated by the circumstance that available datasets consisting of 2D images with 3D annotations only provide annotations for this kind of parametrization (see Chapter 4). Thus, we focus our discussion of related work on approaches that estimate the camera focal length.

2D-3D Point Correspondences: Most approaches for estimating the camera intrinsics are based on 2D-3D point correspondences. In this case, a geometric optimization based on a small number of corresponding 2D locations and 3D points is performed to recover the camera intrinsics, as already discussed in Section 2.1.2.1. In this context, the intrinsic and extrinsic parameters of the camera are often recovered jointly [2]. Moreover, numerous works present geometric optimizations that explicitly estimate the camera focal length and the 3D pose of the camera by solving a PnPf problem [192, 206, 296, 318, 319].

In practice, these methods require precise 2D-3D correspondences, which are often selected manually or using known calibration grids [276, 317]. Many applications, however, require automatic calibration. In specific cases, it is possible to exploit geometric image elements such as lines [51], vanishing points [32, 39], or circles [35] to compute the intrinsics. However, these approaches do not generalize to arbitrary natural images.

Deep Learning: To overcome this limitation, recent approaches directly estimate the camera focal length from RGB images without requiring particular geometric structures using deep learning. For example, [294] uses a CNN pre-trained for ImageNet classification [146] to regress the horizontal field-of-view of the camera that has a one-to-one mapping to the camera focal length. Similarly, [111] trains a CNN to classify the vertical field-of-view of the camera. [172] regresses the vertical field-of-view of the camera as well as radial distortion parameters. In contrast, [288] regresses the camera focal length from regions of interest that contain objects to reduce the impact of homogenous background regions on the prediction. In Chapter 6, we present a method based on deep learning to predict the camera focal length using a logarithmic parametrization.

3.5 Differentiable Rendering

Differentiable rendering [171] is a powerful concept that provides inverse graphics capabilities by computing gradients for 3D scene parameters from 2D image observations, as already discussed in Section 2.1.2.2. This novel technique recently gained popularity for tasks like 3D reconstruction [151, 195], scene lighting estimation [11, 157], and texture prediction [132, 309]. Additionally, differentiable rendering can also be used to refine the 3D pose of objects [200].

However, even simple rendering is a non-differentiable process due to the rasterization operation [134] (see Section 2.1.1.3). In particular, solving the visibility of overlapping triangles is non-differentiable because an argmin operation is required to identify the index of the frontmost triangle for each rasterized pixel. Thus, differentiable rendering approaches either try to mimic rasterization with differentiable operations [167, 200] or use conventional rasterization and provide approximate gradients for the rasterization operation [70, 102, 134, 171].

To mimic rasterization, [200] presents a rastering algorithm formulated with differentiable operations that assigns continuous opposed to binary values to each pixel to generate approximate masks from 3D meshes. [167] formulates rendering as an aggregation function that fuses the probabilistic contributions of all triangles of a 3D mesh with respect to the rendered pixels to generate approximate silhouettes as well as shaded RGB renderings.

To perform differentiable rendering while using conventional rasterization, [151] uses finite differences between forward renderings. In contrast, [171] uses filter-based derivatives to provide approximate gradients for the rasterization operation. [102] extends [171] by computing correctly behaving derivatives in cases of self-occlusion. [70] treats 3D models as locally planar and computes rasterization derivatives for barycentric coordinates but not for visible triangle indices. [134] approximates gradients for rasterization by replacing the sudden intensity changes at triangle boundaries that lead to zero gradients with gradual changes that yield non-zero gradients using linear interpolation. However, differentiable rendering is still considered an unsolved problem. Thus, we present a method that learns to approximate the rasterization backward pass from data in Chapter 8.

CHAPTER 4

Datasets

Contents

In this chapter, we discuss available datasets consisting of 2D images with 3D annotations. Because the methods presented in this thesis are data-driven, labeled datasets are indispensable for training our models in a supervised learning setup. They do not only determine which tasks we can learn but also strongly impact the quality of our learned models. We first give an overview of public datasets that address our 3D vision tasks and then discuss datasets used in our evaluations in more detail.

4.1 Overview

Table 4.1 presents an overview of publicly available datasets consisting of 2D images with 3D annotations. We highlight important differences between existing datasets with respect to specific properties. In the following, we discuss these properties in detail and explain why certain datasets are better suited for training data-driven models for single image 3D vision in the wild than others.

Dataset	Scenario	Images	3D Models	3D Poses	Focal Length
LineMOD [104]	Instance	Real	Fine-grained	Fine-grained	Single camera
YCB [30]	Instance	Real	Fine-grained	Fine-grained	Single camera
T-LESS [108]	Instance	Real	Fine-grained	Fine-grained	Single camera
BOP [110]	Instance	Real	Fine-grained	Fine-grained	Single camera
NOCS [285]	Category	Real	Fine-grained	Fine-grained	Single camera
KITTI [69]	Category	Real	Coarse	Fine-grained	Single camera
ScanNet [44]	Category	Real	Coarse	Fine-grained	Single camera
Pascal3D+ [303]	Category	Real	Coarse	Coarse	Coarse
ObjectNet3D [302]	Category	Real	Coarse	Coarse	Coarse
Houzz [119]	Category	Real	Coarse	Coarse	Fine-grained
Comp/Stanford [288]	Category	Real	Fine-grained	Coarse	Fine-grained
IKEA [162]	Category	Real	Fine-grained	Fine-grained	Fine-grained
Pix3D [258]	Category	Real	Fine-grained	Fine-grained	Fine-grained
P4B [227]	Category	Synthetic	Fine-grained	Fine-grained	Single camera
ShapeNet [33]	Category	Synthetic	Fine-grained	Fine-grained	Fine-grained

Table 4.1: Overview of publicly available datasets consisting of 2D images with 3D annotations.

Scenario: First, existing datasets can be divided into instance-level and category-level scenarios. In instance-level scenarios [30, 104, 108], the same objects are encountered during both training and testing. Additionally, the corresponding datasets provide 3D models with colors and textures that exactly match those of the objects in the RGB images. Typically, the number of different objects is low.

In contrast, category-level scenarios [258, 285, 288, 303] provide a more challenging setup. In this case, the training and test objects are different instances of a common category. Thus, objects seen during training differ from objects seen during testing. For example, objects within a category have varying 3D shapes and textures [302]. As a consequence, they can have significantly different appearances. Additionally, the number of different objects is high. However, common characteristics of objects within a category can be learned to make predictions for previously unseen instances of that category.

Because both scenarios are significantly different, they require specialized approaches. For example, while color information is an important clue in instance-level scenarios, it is mostly irrelevant in category-level scenarios. In instance-level scenarios, exact 3D models are typically available during testing. This is not the case in category-level scenarios. Additionally, category-level approaches need to handle more appearance variations due to the varying 3D shapes and textures of objects. As a consequence, many instance-level approaches are not directly applicable to category-level scenarios. However, any category-level approach can be used in an instance-level scenario but it might not leverage all available information. The two scenarios pose different tasks.

In this thesis, we focus on datasets that address category-level scenarios because they resemble our intended use case of general 3D scene understanding for objects in the wild.

Images: While most datasets provide real-world images, some datasets only provide synthetic images [33, 227]. However, research shows that systems purely trained on synthetic data do not generalize to real data because there is a significant domain gap between real and rendered images [71, 185, 285]. Even techniques like domain randomization [271] and domain adaptation [218, 249] cannot fully bridge the gap between real and synthetic data.

As already discussed in Section 2.1.1.2, the formation of an image in the real world is a complex process that is subject to many factors of variation. For example, physical objects are composed of materials like metal, glass, and fabric that have complex reflection and refraction properties. Moreover, objects are effected by their environment. Real-world scenes consist of many objects that can occlude each other, cast or receive shadows, and form complicated image backgrounds. Additionally, scenes in the real world are globally illuminated and have multiple direct as well as indirect light sources [52, 129]. Finally, artifacts caused by physical cameras and image compression algorithms effect real-world images. As a consequence, generating photo-realistic renderings is hard, computationally expensive, and in many cases even impossible.

In this thesis, we focus on datasets that provide real-world images because the statistics of physically acquired images are different from those of rendered images, and we want to make predictions for the domain of natural images.

3D Models: Many datasets only provide 3D models that coarsely match the geometry of objects in images [44, 119, 302, 303]. Especially in category-level scenarios, it is difficult for annotators to provide fine-grained 3D models for arbitrary objects in images. In many cases, exactly matching 3D models are not available. However, manually constructing them using computer-aided design is a complicated and time-consuming process. Thus, annotators often select approximate 3D models.

In this thesis, we focus on datasets that provide fine-grained 3D model annotations because precise knowledge about the geometry of objects is required to train accurate 3D pose estimation and 3D model retrieval systems in practice.

3D Poses: Similar to the case of 3D models above, many datasets only provide coarse 3D pose annotations [119, 288, 302, 303]. 3D poses have six degrees of freedom that cannot be labeled independently of each other. Hence, their annotation is difficult. The 3D pose of an object is typically annotated based on a 3D model. In this case, annotators either manually optimize the alignment between an object in an image and a rendered 3D model, or first select correspondences between 2D object locations and 3D model points and then perform a geometric optimization to recover the 3D pose (see Section 2.1.2.1). Thus, if the provided 3D model is only a coarse approximation of the object geometry, the annotated 3D pose of an object will also only be a coarse approximation in either labeling approach.

In this thesis, we focus on datasets that provide fine-grained 3D pose annotations because precise ground truth for the 3D rotation and 3D translation of objects is required to train accurate 3D pose estimation and 3D pose refinement systems in practice.

Focal Length: Finally, there are many datasets that have been captured using a single camera [44, 104, 285]. In this case, the camera intrinsics are calibrated and assumed to be known during testing. Additionally, this setup makes it possible to exploit certain geometric constraints. For example, even without explicitly using the camera intrinsics and accurate 3D translation can be estimated based on the 2D bounding box of an object [156]. This is not possible if images are captured with multiple cameras having different focal lengths. Moreover, different focal length images are required to learn to predict the focal length from data.

In this thesis, we focus on datasets that provide fine-grained focal length annotations because in the case of uncalibrated cameras the estimation of an unknown focal length provides an effective prior for 3D pose estimation.

4.2 Pascal3D+

The Pascal3D+ [303] dataset provides in-the-wild RGB images from PascalVOC [57] and ImageNet [233] which are augmented with coarse 3D pose, 3D model, and focal length annotations for objects of twelve categories (*aeroplane*, *bike*, *boat*, *bottle*, *bus*, *car*, *chair*, *table*, *motorbike*, *sofa*, *train*, and *tv*). The images are captured with multiple cameras having different focal lengths and there are annotations for multiple objects in a single image. However, there is only a limited pool of around ten 3D models for each category and each labeled object is assigned one 3D model from the respective category pool. Thus, the 3D model annotations are only approximate. Moreover, the annotated 3D poses are computed from manually selected 2D-3D correspondences. Because the annotated 3D models are only approximate, the computed 3D poses are also only approximate. Additionally, a constant focal length is assumed for images captured with different focal length cameras. The Pascal3D+ datasets already provide a train-test split and we use all available samples for training and evaluation.

4.3 Comp and Stanford

The Comp and Stanford [288] datasets provide in-the-wild RGB images with fine-grained 3D model and focal length but coarse 3D pose annotations for objects of the category *car*. The images are captured with multiple cameras having different focal lengths. Most images show one prominent car which is non-occluded and non-truncated. Only this prominent object is annotated, while possible other objects in the background are ignored. The annotated 3D models are selected based on the known types of the cars in the images. Thus, they are accurate for the majority of samples. However, the 3D poses are manually optimized based on visual alignment by human annotators. As a consequence, the 3D poses of many samples lack accuracy. The two datasets already provide a train-test split. Thus, we use all available samples from Comp and Stanford for training and evaluation.

4.4 Pix3D

The Pix3D [258] dataset extends the IKEA dataset [162] and provides in-the-wild RGB images with fine-grained 3D pose, 3D model, and focal length annotations for objects of different categories. The images are captured with multiple cameras having different focal lengths. Compared to other category-level datasets, all annotations are fine-grained because the annotation process is performed using exact 3D models and accurate 2D-3D correspondences. However, only a single object per image is annotated, even though there are multiple objects in many cases. This dataset provides annotations for multiple object categories but we only train and evaluate on categories which have more than 300 non-occluded and non-truncated samples (*bed*, *chair*, *sofa*, *table*). Further, we restrict the training and evaluation to samples marked as non-occluded and non-truncated because we do not know which objects parts are occluded nor the extent of the occlusion, and many objects are heavily truncated. Because the Pix3D dataset does not provide a train-test split, we randomly select 50% of the samples of each category for training and the other 50% for testing.

4.5 ShapeNet

In contrast to the datasets describes above, the ShapeNet [33] dataset does not provide real-world RGB images but a very large number of textured 3D models. Due to the enormous size of this 3D model database, ShapeNet does not only cover many different object categories but also presents a large variety in 3D model geometry. Synthetic images for the training of data-driven approaches can be rendered from the available 3D models [257] but the significant domain gap between real and rendered images limits the benefit of this strategy in practice. However, the ShapeNet datasets is an excellent candidate for a 3D model retrieval database because 3D models within a category densely sample the underlying true data distribution of objects in the real world. Additionally, all 3D models are consistently oriented, scaled, and translated.

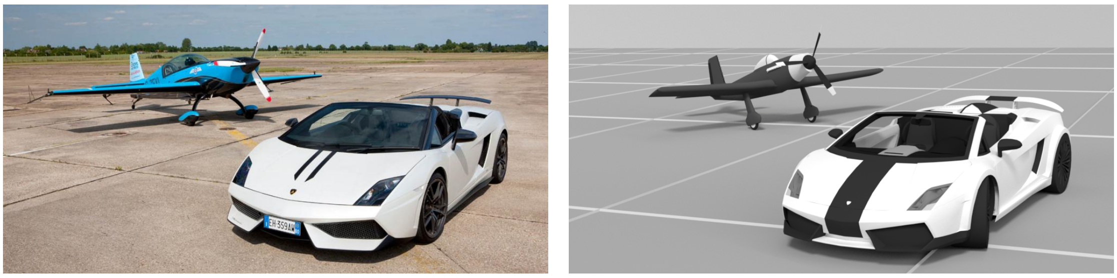
CHAPTER 5

3D Pose Estimation and 3D Model Retrieval in the Wild

Contents

In this chapter, we present the first contribution of this thesis. In particular, we propose a scalable, efficient, and accurate approach to retrieve 3D models for objects of arbitrary categories from single RGB images captured in the wild. Our specific contribution is twofold. First, we present a novel category-level 3D pose estimation approach which significantly outperforms the state of the art for 3D viewpoint estimation on Pascal3D+ [303]. Second, we use our estimated 3D poses as a prior to retrieve 3D models which accurately represent the geometry of objects in RGB images. For this purpose, we compare each object in an RGB image to depth renderings of 3D models from a retrieval database generated under our predicted 3D pose. We perform this comparison by mapping RGB images and depth renderings to feature vectors in the form of 3D shape descriptors in a common embedding space using a CNN-based multi-view metric learning approach. In this embedding space, we match 3D shape descriptors to perform 3D model retrieval. We evaluate our proposed 3D model retrieval approach on Pascal3D+, where we are the first to report quantitative results and our method retrieves the ground truth 3D model for 50% of all validation objects. In addition, we show that our method which was trained purely on Pascal3D+ retrieves rich and accurate 3D models from ShapeNet [33] without retraining given previously unseen RGB images of objects in the wild.

5.1 Introduction



Image

Predicted 3D Poses and 3D Models

Figure 5.1: Given a single RGB image containing one or more objects of arbitrary categories like *car* or *aeroplane*, we predict a 3D pose and a 3D model for each object.

Retrieving 3D models for objects in single 2D images captured in the wild, as shown in Fig. 5.1, is important for applications like augmented reality, robotics, and general 3D scene understanding. Recently, the emergence of large 3D model databases such as ShapeNet [33] initiated substantial interest in this task and motivated research for matching single 2D images of objects against 3D models. However, there is no straightforward approach to compare 2D images and 3D models, since they have considerably different representations and characteristics.

One approach to address this problem is to project 3D models onto 2D images using rendering (see Section 2.1.1). This converts the task to comparing 2D images, which is, however, still challenging because the appearance of objects in real-world images and synthetic renderings can significantly differ. In general, the geometry and textures of available 3D models do not exactly match those of objects in real-world images. Thus, recent approaches [10, 123, 256, 302] use CNNs (see Section 2.2) to extract features which are partly invariant to these variations from images. In particular, these methods compute and match feature vectors from real-world RGB images and synthetic RGB images generated by rendering 3D models under multiple 3D poses to perform retrieval. This strategy allows them to extract features from both real and synthetic RGB images using a single CNN that can be trained purely on synthetic data but can later also be applied to real data. However, there are two main disadvantages:

First, there is a significant domain gap between real and synthetic RGB images (see Section 4.1). Real images are subject to complex lighting, uncontrolled degradation, and natural backgrounds. This makes it hard to generate photo-realistic renderings. Moreover, many 3D models do not even provide color or texture information. In this case, photo-realistic rendering is impossible. As a consequence, using a single CNN for feature extraction from both the real and synthetic image domain is limited in performance. Finally, not even techniques like domain adaption [185] can fully account for the different characteristics of the two domains.

Second, processing renderings from multiple different 3D poses for each 3D model is computationally expensive. However, this step is mandatory because the appearance of an object can significantly change with its 3D pose. Also, mapping renderings from all 3D poses to a common feature vector does not scale to many categories [158]. Thus, knowing the 3D pose of an object significantly eases the task of 3D model retrieval while reducing computational complexity at the same.

To overcome these limitations, we propose to first predict the 3D pose of an object and then use the predicted 3D pose as an effective prior for 3D model retrieval. Inspired by recent work on instance-level 3D pose estimation [43, 217], we present a robust category-level 3D pose estimation approach based on virtual 2D-3D correspondences. More specifically, we use a CNN to predict both virtual 3D points as well as their corresponding 2D image locations from which we recover the 3D pose of each object using a PnP algorithm (see Section 2.1.2). In contrast to previous work [43, 217], our approach does not require ground truth 3D models at runtime and additionally supports category-agnostic predictions.

To perform 3D model retrieval, we then compare each object in an RGB image to renderings of 3D models from a database. However, in contrast to previous work [10, 123, 256, 302], we exploit a 3D pose prior and only use a single rendering generated under our predicted 3D pose per 3D model. This rendering can be pre-computed offline, which makes our approach scalable. Additionally, we propose to use depth renderings instead of RGB renderings and use two different CNNs in a multi-view metric learning approach to map real-world RGB images and synthetic depth renderings to feature vectors in the form of 3D shape descriptors in a common embedding space. In this way, we are not only able to deal with untextured 3D models but also alleviate the domain gap.

We evaluate our proposed 3D pose estimation and 3D model retrieval approach on the challenging Pascal3D+ [303] dataset. In terms of 3D pose estimation, our method outperforms the state of the art for 3D viewpoint estimation. In terms of 3D model retrieval, we are the first to report quantitative results on Pascal3D+ and our method retrieves the ground truth 3D model for 50% of all validation objects. Moreover, we demonstrate that our method which was trained purely on Pascal3D+ retrieves rich and accurate 3D models from the ShapeNet [33] dataset without retraining given previously unseen RGB images of objects in the wild from the Pascal3D+ validation data. To summarize, we make the following contributions:

- We present a category-level 3D pose estimation approach for objects in the wild that significantly outperforms the state of the art on Pascal3D+. Our method predicts virtual 2D-3D correspondences which generalize across different categories.

- We introduce a novel 3D model retrieval approach that utilizes a 3D pose prior. Our method compares objects in single RGB images to 3D model renderings generated under predicted 3D poses and bridges the domain gap between real and rendered images. In this way, we retrieve 3D models which accurately represent the geometry of objects in images in a computationally efficient way, as shown in Fig. 5.1.

5.2 3D Pose Estimation and 3D Model Retrieval

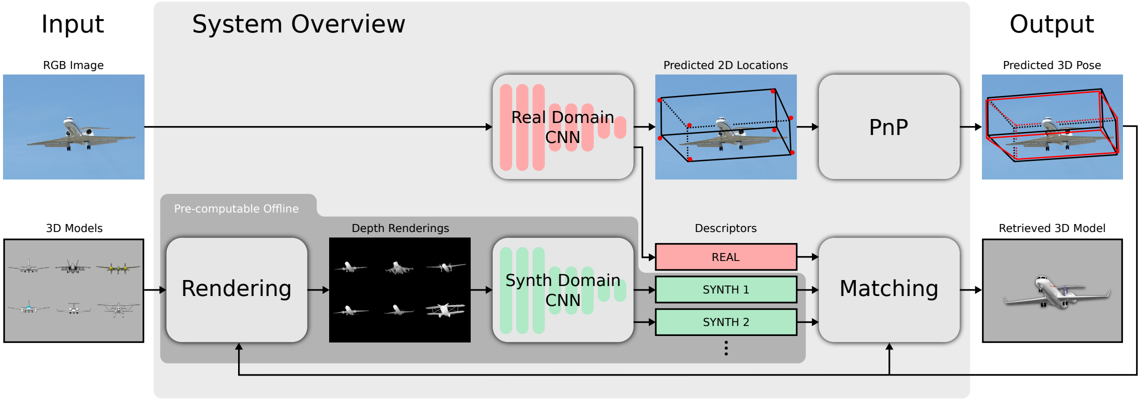


Figure 5.2: Overview of our approach. First row: Given a single RGB image of an object, we first predict its 3D pose. We use a CNN to predict the 2D image locations of the object's 3D bounding box corners (red dots) and the object's 3D dimensions. From these virtual 2D-3D correspondences, we recover the 3D pose of the object using a PnP algorithm. Second row: We generate depth renderings from 3D models under our predicted 3D pose and compute 3D shape descriptors from both the real RGB image as well as the synthetic depth renderings using two different CNNs. Finally, we match the computed descriptors to retrieve an accurate 3D model for the object. Our approach supports offline pre-computed 3D shape descriptors for 3D models.

Given a single RGB image containing one or more objects, we predict a 3D pose and retrieve a 3D model with accurate geometry for each object. Fig. 5.2 presents an overview of our proposed pipeline. We first estimate the 3D pose of each object from an image window roughly centered on the object. In this work, we assume the input image windows are known as in [303] or given by a 2D object detector [225]. Similar to previous work [191, 204, 279], we also assume the object category to be known since it is a useful prior for both 3D pose estimation and 3D model retrieval. However, we also show that this information is not necessarily required in our approach. In fact, we can predict accurate 3D poses with only a marginal loss in accuracy when the object category is unknown.

After we estimated the 3D pose of an object, we perform 3D model retrieval by comparing the object to a number of candidate 3D models rendered under our predicted 3D pose. In particular, we use depth renderings which allow us to deal with untextured 3D models and circumvent the problem of scene lighting. In order to compare the real RGB image to synthetic depth renderings, we map both representations to 3D shape descriptors in a common embedding space using two CNNs, one for each domain. Finally, we match our computed 3D shape descriptors to retrieve a 3D model with accurate geometry.

5.2.1 3D Pose Estimation

The first step in our approach is to robustly compute the 3D pose of an object. For this purpose, we predict the 2D image locations of virtual 3D points [43]. More precisely, we use a CNN to predict the 2D image locations of the projections of the eight 3D bounding box corners for each object [217]. From these 2D-3D correspondences, we recover the 3D pose of an object by solving a PnP problem (see Section 2.1.2.1). This is illustrated in the first row of Fig. 5.2.

However, PnP algorithms require both the 2D locations and 3D points of correspondences to be known. Thus, previous approaches either assume the exact 3D model to be given at runtime [217] or predict the projections of static 3D points [43]. To overcome this limitation, we predict the spatial 3D dimensions $\mathbf{d} = (d_x, d_y, d_z)^T$ of the object's 3D bounding box and use these to scale a unit cube, which approximates the ground truth 3D points. In this way, we are able to make predictions for arbitrary objects without providing ground truth 3D models at runtime.

For this purpose, we introduce a CNN architecture which jointly regresses the 2D image locations of the projections of the eight 3D bounding box corners (16 values) as well as the 3D bounding box dimensions (3 values). As illustrated in Fig. 5.3, we implement this architecture as a single 19 neuron linear output layer, which we apply on top of the penultimate layer of different base networks such as VGG [250] or ResNet [99, 100]. During training, we optimize the pose loss

$$L_{\text{pose}} = L_{\text{proj}} + \alpha L_{\text{dim}} + \beta L_{\text{reg}_1} \tag{5.1}$$

which is a linear combination of the projection loss L_{proj}, the dimension loss L_{dim}, and the regularization L_{reg}. The meta-parameters α and β control the impact of the different loss terms. Let $\mathbf{M}_i$ be the i-th 3D bounding box corner and $\mathbf{m}_i = \text{proj}(\mathbf{M}_i, \mathbf{K}, \mathbf{R}, \mathbf{t})$ its projection onto the image plane using the ground truth camera intrinsics $\mathbf{K}$, 3D rotation $\mathbf{R}$, and 3D translation $\mathbf{t}$. Then the projection loss

$$L_{\text{proj}} = E\left[\sum_{i=1}^{8} \mathcal{L}\Big(\|\mathbf{m}_i - \hat{\mathbf{m}}_i\|_2\Big)\right] \tag{5.2}$$

is the expected value of the distances between the ground truth projections $\mathbf{m}_i$ and the predicted 2D locations of these projections $\hat{\mathbf{m}}_i$ computed by the CNN for the training set. Being aware of inaccurate annotations in datasets such as Pascal3D+ [303], we use the Huber loss [118] $\mathcal{L}(\cdot)$ in favour of the squared loss to be more robust to outliers.

The dimension loss

$$L_{\text{dim}} = E\left[\mathcal{L}\Big(\|\mathbf{d} - \hat{\mathbf{d}}\|_1\Big)\right] \tag{5.3}$$

is the expected value of the distances between the ground truth 3D dimensions $\mathbf{d}$ and the predicted 3D dimensions $\hat{\mathbf{d}}$ computed by the CNN for the training set. Again, we use the Huber loss $\mathcal{L}(\cdot)$ in favour of the squared loss to be more robust to outliers. Finally, to reduce the risk of overfitting, the regularization loss L_{reg_1} in Eq. (5.1) adds weight decay [147] for all CNN parameters.

5.2.2 3D Model Retrieval

Having a robust estimate of the 3D pose of an object, we only consider renderings under our predicted 3D pose for 3D model retrieval instead of considering multiple renderings under different 3D poses [10, 123, 185, 256]. This significantly reduces the computational complexity compared to methods which process multiple renderings for each 3D model and additionally provides a useful prior for retrieval. Moreover, in contrast to many recent approaches [158, 185, 256, 302], we use depth renderings instead of RGB renderings. This allows us to deal with 3D models which do not have material or texture information. Additionally, we circumvent the problem of how to set the scene lighting. Compared to methods that also achieve this effect by directly operating on 3D data, we do not require computationally expensive 3D convolutions for high-resolution 3D voxel grids [71] nor rely on real depth sensors [62, 320, 321].

However, since real RGB images and synthetic depth renderings have considerably different characteristics, we introduce a multi-view metric learning approach which maps images from both domains to a common embedding space. We implement this mapping using a separate CNN for each domain. For real RGB images, we extract 3D shape descriptors from the penultimate layer of our 3D pose estimation CNN (see *Real Domain CNN* in Fig. 5.3). Because these features have already been computed during 3D pose estimation inference, we get the real domain 3D shape descriptor without any additional computational cost. For synthetic depth renderings, we compute 3D shape descriptors using a CNN with the same architecture as our 3D pose estimation CNN, except for the output layer (see *Synth Domain CNN* in Fig. 5.3).

To train the mapping of images from both domains to the common embedding space, we optimize the similarity loss

$$L_{\text{similarity}} = L_{\text{descr}} + \gamma L_{\text{reg}_2} \tag{5.4}$$

which comprises the descriptor loss L_{descr} and the regularization loss L_{reg_2} weighted by the meta-parameter γ. The descriptor loss

$$L_{\text{descr}} = E\left[\max(0, s^+ - s^- + m)\right] \tag{5.5}$$

minimizes the expected value of the Triplet loss [290] for the training set. In this case, s^+ is the Euclidean distance between a real domain descriptor and the corresponding synthetic domain descriptor, s^- is the Euclidean distance between the same real domain descriptor and a negative example synthetic domain descriptor, and m specifies the margin, i.e., the desired minimum difference between s^+ and s^-. To reduce the risk of overfitting, the regularization loss L_{reg_2} in Eq. (5.4) adds weight decay for all CNN parameters.

Finally, we can first train our *Real Domain CNN* using the pose loss L_{pose} and then train our *Synth Domain CNN* using the similarity loss $L_{\text{similarity}}$, or jointly optimize both

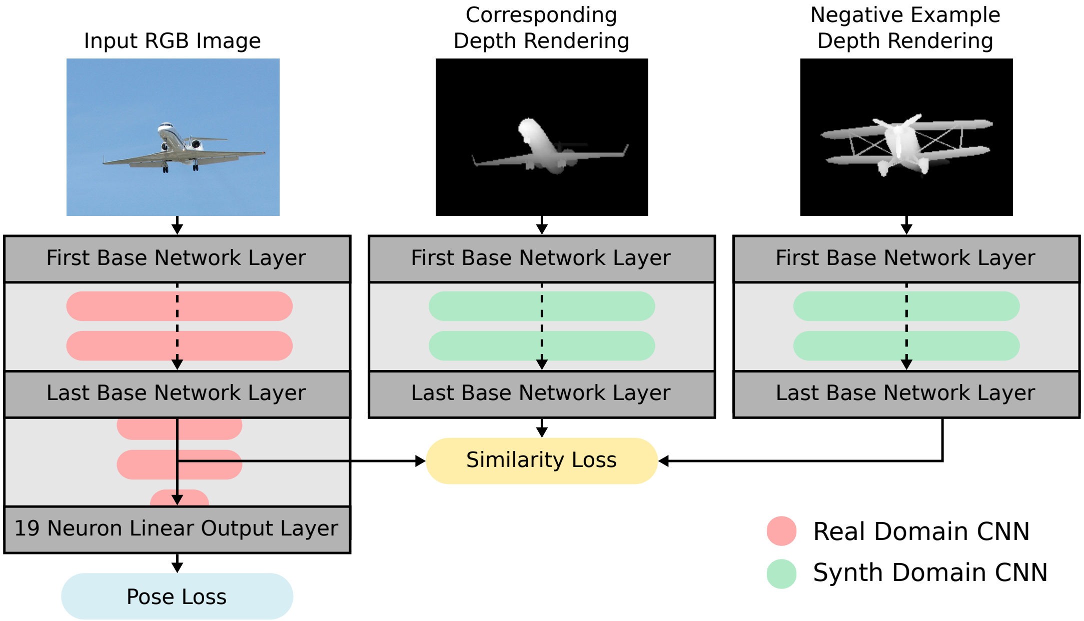


Figure 5.3: Our pose loss is computed on the output of our *Real Domain CNN*. Our similarity loss is computed on features extracted from the last base network layer of both our *Real Domain CNN* and our *Synth Domain CNN*.

CNNs at the same time using the combined loss

$$L_{\text{combined}} = L_{\text{pose}} + \lambda L_{\text{similarity}} \,, \tag{5.6}$$

where λ is weighting meta-parameter.

After the optimization of our synthetic domain CNN, we can pre-compute descriptors for synthetic depth renderings in an offline phase. In this case, we first generate multiple renderings for each 3D model which cover the full 3D pose space. We then compute descriptors for all these renderings and store them in a database. At runtime, we just match descriptors corresponding to the 3D pose closest to our predicted 3D pose, which is fast and scalable but still accurate as shown in our experiments.

5.3 Evaluation

To demonstrate our 3D model retrieval approach for objects in the wild, we evaluate it in a realistic setup, where we retrieve 3D models from ShapeNet [33] given unseen RGB images from Pascal3D+ [303]. In particular, we train our 3D model retrieval approach purely on data from Pascal3D+ but use it to retrieve 3D models from ShapeNet. The corresponding results are detailed in Section 5.3.3. Because estimating accurate 3D poses is essential for our 3D model retrieval approach, we additionally evaluate our 3D pose estimation approach on Pascal3D+ in Section 5.3.2.

5.3.1 Implementation Details

To train our approach, we use specific implementation details, neural network hyperparameters, and other settings described in the following.

Intrinsic Camera Parameters: In Pascal3D+, the ground truth 3D poses are computed from 2D-3D correspondences assuming the same intrinsic camera parameters for all images. The dataset uses a pinhole camera model with a large focal length, as presented in Section 2.1.1.2. We employ the same camera model in our approach.

Data Augmentation: Akin to previous work [191, 204, 257, 279], we perform data augmentation by jittering ground truth 2D detections during training and exclude detections marked as occluded or truncated from the evaluation. Additionally, we augment samples for which the longer edge of the ground truth 2D bounding box is greater than 224 pixels by applying Gaussian blurring with random kernel size and σ. For the training of our *Synth Domain CNN*, we randomly sample negative example 3D models from the available 3D model pool for our Triplet criterion.

Meta-Parameters: Our *Real Domain CNN* is trained to predict normalized 2D locations. During training, we normalize 2D locations such that the image pixel range $[0, 223]$ is mapped to the interval $[0, 1]$ and use the same Huber loss ($\delta = 0.01$) for all 19 estimated values. Empirically, we found $\alpha = 1$, $\beta = 1e^{-5}$, and $\gamma = 1e^{-3}$ to perform well and set $m = 1$. To jointly train both CNNs, we set $\lambda = 1e^{-2}$.

Network Parameters: We use a minibatch size of 50, train both of our CNNs for 100 epochs, decrease the initial learning rate of $1e^{-4}$ by one order of magnitude after 50 and 90 epochs, and employ the Adam [138] optimization algorithm. We initialize our base networks with weights pre-trained for ImageNet [233] classification and fine-tune all layers for our task.

3D Dimensions: For both Pascal3D+ and ShapeNet, 3D models are consistently oriented, scaled, and translated. They are normalized to fit within a unit cube centered at the origin. Thus, we estimate 3D dimensions in the model space in the range $[0, 1]$. Since these dimensions tend to be consistent within a category, estimating them is not a major problem. In fact, we achieve high accuracy across all categories and the mean absolute 3D dimension error is around 0.017 which corresponds to a relative error of only 1.7%.

5.3.2 3D Pose Estimation

In the following, we first perform an ablation study of our 3D pose estimation approach. We then compare it to previous methods and significantly outperform the state of the art for 3D viewpoint estimation on Pascal3D+. Finally, we demonstrate that we are even able

	$MedErr$	$Acc_{\frac{\pi}{6}}$
Ours - VGG	11.7	0.8076
Ours - VGG+blur	11.6	0.8033
Ours - ResNet	**10.9**	0.8341
Ours - ResNet+blur	**10.9**	**0.8392**

Table 5.1: Ablation study of our 3D pose estimation method. We show the 3D viewpoint accuracy using ground truth detections on Pascal3D+ for different setups of our approach. We report the mean performance across all categories. Using ResNet as a base network together with blurring training images improves the performance, especially on objects detected at small scale.

to top the state of the art without providing the correct category prior in some cases. For a fair evaluation, we follow the protocol of [279] which quantifies 3D viewpoint prediction accuracy on Pascal3D+ using the geodesic distance

$$\Delta(\mathbf{R}_{\text{gt}}, \mathbf{R}_{\text{pred}}) = \frac{\| \log(\mathbf{R}_{\text{gt}}^T \mathbf{R}_{\text{pred}}) \|_F}{\sqrt{2}} \tag{5.7}$$

to measure the difference between the ground truth 3D rotation matrix $\mathbf{R}_{\text{gt}}$ and the predicted 3D rotation matrix $\mathbf{R}_{\text{pred}}$. In particular, we report two metrics: $MedErr$ (the median of all geodesic distances) and $Acc_{\frac{\pi}{6}}$ (the percentage of all geodesic distances smaller than $\frac{\pi}{6}$ respectively 30°). Evaluating our approach using the AVP metric [303] which couples 2D object detection and azimuth classification is not meaningful as it is very different from our specific task. Also, there is no 3D pose estimation evaluation protocol for Pascal3D+. However, we demonstrate the benefit of additionally predicting 3D translation information in qualitative 3D model alignment experiments in Section 5.3.3.3.

5.3.2.1 3D Pose Estimation on Pascal3D+

Table 5.1 presents an ablation study our approach that shows quantitative results for 3D viewpoint estimation on Pascal3D+ using different setups. We start from a baseline and additionally present more elaborate versions. For our baseline approach (*Ours - VGG*), we build on a VGG-backend [250] and fine-tune the entire network for our task similar to [191, 277, 279]. This baseline already achieves 3D viewpoint accuracy on par with the state of the art (see Table 5.2).

However, by inspecting the failure cases of this baseline, we observe that many incorrect predictions relate to small objects. In these cases, objects detected at a small scale need to be upsampled to fit the fixed spatial input resolution of the network, i.e., 224 × 224 pixels. In fact, for more than 55% of Pascal3D+ validation objects the longer edge of the ground truth 2D bounding box is smaller than 224 pixels. Thus, the required upsampling results in blurry images with washed-out details, over-smoothed edges, and suppressed corners. As a consequence, VGG which only employs 3 × 3 convolutions performs poorly at extracting features from such upscaled images.

To overcome this limitation, we propose to use a network with larger kernel sizes that performs better at handling over-smoothed input images such as ResNet-50 [99, 100] which uses 7 × 7 kernels in the first convolutional layer. As shown in Table 5.1, our approach with a ResNet-backend (*Ours - ResNet*) significantly outperforms the VGG-based version. In addition, the total number of network parameters is notably lower (VGG: 135M vs. ResNet-50: 24M).

To further improve the performance, we employ data augmentation in the form of image blurring. In particular, we augment samples for which the longer edge of the ground truth 2D bounding box is greater than 224 pixels by applying Gaussian blurring with random kernel size and σ. Using ResNet-50 as a base network together with blurring training images (*Ours ResNet+blur*), we improve on the $Acc_{\frac{\pi}{6}}$ metric while maintaining low $MedErr$ (see Table 5.1). This indicates that we improve the performance on over-smoothed images but do not loose accuracy on sharp images. While our approach with a ResNet-backend shows increased performance in this setup, we do not benefit from training on blurred images using a VGG-backend (*Ours - VGG+blur*). This also confirms that VGG is not suited for feature extraction from over-smoothed images. We also experimented with other forms of data augmentation like in-plane rotation or training without truncated and occluded objects but we did not achieve any performance improvements in these cases. We hypothesize that this is due to the bias towards zero in-plane rotation and little truncations and occlusions in the Pascal3D+ dataset. For all following experiments, we use our best performing setup, i.e., *Ours - ResNet+blur*.

5.3.2.2 Comparison to the State of the Art

Next, we compare our 3D pose estimation approach to state-of-the-art methods on Pascal3D+. Quantitative results are presented in Table 5.2. Our approach significantly outperforms the state of the art in both $MedErr$ and $Acc_{\frac{\pi}{6}}$ considering mean performance across all categories and also shows competitive results for individual categories.

However, the $Acc_{\frac{\pi}{6}}$ scores for two categories, i.e., *boat* and *table*, are significantly below the mean score. We analyze these results in more detail. The category *boat* is the most challenging category due to the large intra-class variability in 3D shape and appearance. Many detections for this category are of low resolution and often objects are barely visible because of fog or mist. Additionally, there are a lot of ambiguities, e.g., even a human cannot distinguish between the front and the back of a symmetric canoe. Nevertheless, we outperform the state of the art for this challenging category.

The low $Acc_{\frac{\pi}{6}}$ scores for the category *table* can be explained by three factors. First, many tables are partly occluded by chairs (see *table* in Fig. 5.4). Second, the evaluation protocol does not take into account that many tables are ambiguous with respect to a rotation of π, $\frac{\pi}{2}$, or even have an axis of symmetry like a round table. In some cases, our system predicts an ambiguous 3D pose instead of the ground truth 3D pose, while it is

	category-specific												
	aero	bike	boat	bottle	bus	car	chair	table	mbike	sofa	train	tv	mean
$MedErr$([204])	11.2	15.2	37.9	13.1	4.7	6.9	12.7	N/A	N/A	21.7	9.1	38.5	N/A
$MedErr$([204]*)	**8.0**	13.4	40.7	11.7	**2.0**	5.5	10.4	N/A	N/A	9.6	8.3	32.9	N/A
$MedErr$([279])	13.8	17.7	21.3	12.9	5.8	9.1	14.8	15.2	14.7	13.7	8.7	15.4	13.6
$MedErr$([191])	13.6	12.5	22.8	**8.3**	3.1	5.8	11.9	12.5	12.3	12.8	6.3	11.9	11.1
$MedErr$([257]**)	15.4	14.8	25.6	9.3	3.6	6.0	**9.7**	**10.8**	16.7	**9.5**	**6.1**	12.6	11.7
$MedErr$(Ours)	10.0	15.6	**19.1**	8.6	3.3	**5.1**	13.7	11.8	**12.2**	13.5	6.7	**11.0**	**10.9**
$Acc_{\frac{\pi}{6}}$([279])	0.81	0.77	0.59	0.93	**0.98**	0.89	0.80	0.62	0.88	0.82	0.80	0.80	0.8075
$Acc_{\frac{\pi}{6}}$([191])	0.78	**0.83**	0.57	0.93	0.94	0.90	0.80	0.68	0.86	0.82	0.82	0.85	0.8103
$Acc_{\frac{\pi}{6}}$([257]**)	0.74	**0.83**	0.52	0.91	0.91	0.88	**0.86**	**0.73**	0.78	**0.90**	**0.86**	**0.92**	0.8200
$Acc_{\frac{\pi}{6}}$(Ours)	**0.83**	0.82	**0.64**	**0.95**	0.97	**0.94**	0.80	0.71	0.88	0.87	0.80	0.86	**0.8392**
	category-agnostic												
	aero	bike	boat	bottle	bus	car	chair	table	mbike	sofa	train	tv	mean
$MedErr$(Ours)	10.9	**12.2**	23.4	9.3	3.4	5.2	15.9	16.2	**12.2**	11.6	6.3	11.2	11.5
$Acc_{\frac{\pi}{6}}$(Ours)	0.80	0.82	0.57	0.90	0.97	**0.94**	0.72	0.67	**0.90**	0.80	0.82	0.85	0.8133

Table 5.2: Quantitative results for 3D viewpoint estimation using ground truth detections on Pascal3D+. [204]* requires the ground truth 3D models to be known at runtime. [257]** was trained on vast amounts of RGB renderings from ShapeNet instead of Pascal3D+ training data.

not possible to differentiate between the two 3D poses visually. The evaluation protocol needs to be changed to take this into account. Finally, the number of validation samples for the category *table* is very small, i.e., 21, and, therefore, the reported results for this category are highly biased.

5.3.2.3 Category-Agnostic 3D Pose Estimation

So far, the discussed results are category-specific. This means that the ground truth category must be provided at runtime. In fact, all evaluated methods use a separate output layer for each category. However, our approach is able to make category-agnostic predictions which generalize across different categories. In this case, we use a single 19 neuron output layer which is shared for all categories making our approach scalable. Our category-agnostic 3D pose estimation even outperforms the previous category-specific state of the art for some categories (see Table 5.2) because it fully leverages the mutual information between similar categories like *bike* and *motorbike*, for example.

5.3.3 3D Model Retrieval

Now, we demonstrate our 3D model retrieval approach which uses our predicted 3D poses as a prior. First, we present a quantitative evaluation of our approach on Pascal3D+. Second, we show qualitative results for 3D model retrieval from ShapeNet given previously unseen images from Pascal3D+. Finally, we use our predicted 3D poses and 3D dimensions to precisely align retrieved 3D models to objects in real-world images.

	aero	bike	boat	bottle	bus	car	chair	table	mbike	sofa	train	tv	mean
Top-1-Acc (Rand)	0.15	0.21	0.36	0.25	0.25	0.10	0.15	0.10	0.28	0.31	0.27	0.27	0.2250
Top-1-Acc (Cano)	0.12	0.25	0.38	0.35	0.45	0.21	0.20	0.15	0.20	0.21	0.49	0.50	0.2925
Top-1-Acc (Off)	0.48	0.33	0.58	**0.41**	0.75	0.35	0.28	0.10	0.44	0.28	0.62	0.63	0.4375
Top-1-Acc (Pred)	0.48	0.31	**0.60**	**0.41**	0.78	0.41	0.29	0.19	0.43	**0.36**	0.65	0.61	0.4600
Top-1-Acc (GT)	**0.53**	**0.38**	0.51	0.37	**0.79**	**0.44**	**0.32**	**0.43**	**0.48**	0.33	**0.66**	**0.72**	**0.4967**

Table 5.3: Quantitative results for 3D model retrieval using ground truth detections on Pascal3D+.

5.3.3.1 3D Model Retrieval from Pascal3D+

Since Pascal3D+ provides 3D pose and 3D model annotations for objects in RGB images, we can train our entire approach on this dataset. In particular, we train our proposed two CNN architectures for computing descriptors from real RGB images and synthetic depth renderings. For our *Synth Domain CNN*, we employ the same data augmentation as for our *Real Domain CNN*, except for blurring because we do not need to rescale low-resolution depth renderings. The renderings are always generated at the desired resolution.

In fact, we are the first to report quantitative results for 3D model retrieval on Pascal3D+. For this purpose, we compute the top-1-accuracy (*Top-1-Acc*), i.e., the percentage of evaluated samples for which the top retrieved 3D model equals the ground truth 3D model. This task is not trivial because many 3D models in Pascal3D+ have similar geometry and are hard to distinguish. We evaluate our approach using five different 3D pose setups: (1) the ground truth 3D pose (*GT*), (2) our predicted 3D pose (*Pred*), (3) our predicted 3D pose with offline pre-computed descriptors (*Off*), (4) a canonical 3D pose (*Cano*), and (5) a random 3D pose (*Rand*). Table 5.3 compares quantitative retrieval results for all five setups.

As expected, we achieve the highest accuracy if we provide the ground truth 3D pose of an object (*GT*). In this case, our approach chooses the same 3D models as human annotators for 50% of all validation objects. However, if we use 3D model renderings generated under our predicted 3D pose (*Pred*), we almost match the accuracy of the ground truth 3D pose setup. For some categories, we observe even better accuracy when using our predicted 3D pose. This proves the high quality of our predicted 3D poses.

Moreover, our approach is fast and scalable at runtime while almost maintaining accuracy by using offline pre-computed descriptors (*Off*). For this experiment, we discretize the 3D pose space in intervals of 10° and pre-compute descriptors for all 3D models. At runtime, we only match pre-computed descriptors from the discretized 3D pose which is closest to our predicted 3D pose and do not have to render 3D models or compute descriptors online.

In contrast, if we just render the 3D models under a random 3D pose (*Rand*) the performance decreases significantly. Nevertheless, even in this case, where we do not have any information about the 3D pose, our approach performs better than random guessing. Rendering models under a canonical 3D pose (*Cano*), e.g., a frontal view, provides a useful bias for objects of the categories *train*, *bus*, and *tv monitor* because these objects are frequently seen from an almost frontal view in the Pascal3D+ dataset. Finally, these results confirm the importance of fine-grained 3D pose estimation in our approach.

5.3.3.2 3D Model Retrieval from ShapeNet

In contrast to Pascal3D+, ShapeNet provides a significantly larger spectrum of 3D models. Thus, we now evaluate our retrieval approach trained purely on Pascal3D+ for 3D model retrieval from ShapeNet without retraining given previously unseen images from Pascal3D+. Fig. 5.4 shows qualitative retrieval results for all twelve categories in Pascal3D+. Our approach predicts accurate 3D poses and 3D models for objects of different categories. In some cases, our predicted 3D pose (see *sofa* in Fig. 5.4) or our retrieved 3D model from ShapeNet (see *aeroplane* and *chair* in Fig. 5.4) is even more accurate than the annotated ground truth from Pascal3D+. While the geometry of the retrieved 3D models corresponds well to the objects in the query images, the materials and textures typically do not. The reason for this is that we use depth renderings for retrieval which do not include color information. This issue can be addressed by extracting texture information from the query RGB image or by performing retrieval with RGBD images. However, this is up to future research.

5.3.3.3 3D Model Alignment

Next, we use our predicted 3D pose and 3D dimensions to precisely align retrieved 3D models to objects in real-world images. Fig. 5.5 shows how we improve the 2D object localization and the alignment between the object and a rendering using our predicted 3D pose and 3D dimensions. This is especially useful if the object detection is not fully accurate, which is true in almost all situations. In the presented cases, the detected 2D bounding boxes are too small and the objects are not centered in the image windows. Thus, if we just render a 3D model under our predicted 3D rotation, rescale it to tightly fit into the 2D image window, and center it in the 2D image window, the alignment is poor. However, if we additionally use our predicted 3D translation and 3D dimensions for scaling and positioning, we significantly improve the alignment between the object and the rendering. This is important for applications like robotics and augmented reality.

5.3.4 Failure Modes

Finally, we discuss failure modes of our approach. If our 3D pose estimation fails, the 3D model retrieval becomes even more difficult. This is also reflected in Table 5.3, where we observe a strong decrease in performance if we render 3D models without 3D pose information (*Rand* and *Cano*).

Most failure cases of our 3D pose estimation on Pascal3D+ relate to low-resolution images or ambiguous objects. Fig. 5.6 shows four examples in which our 3D pose estimation

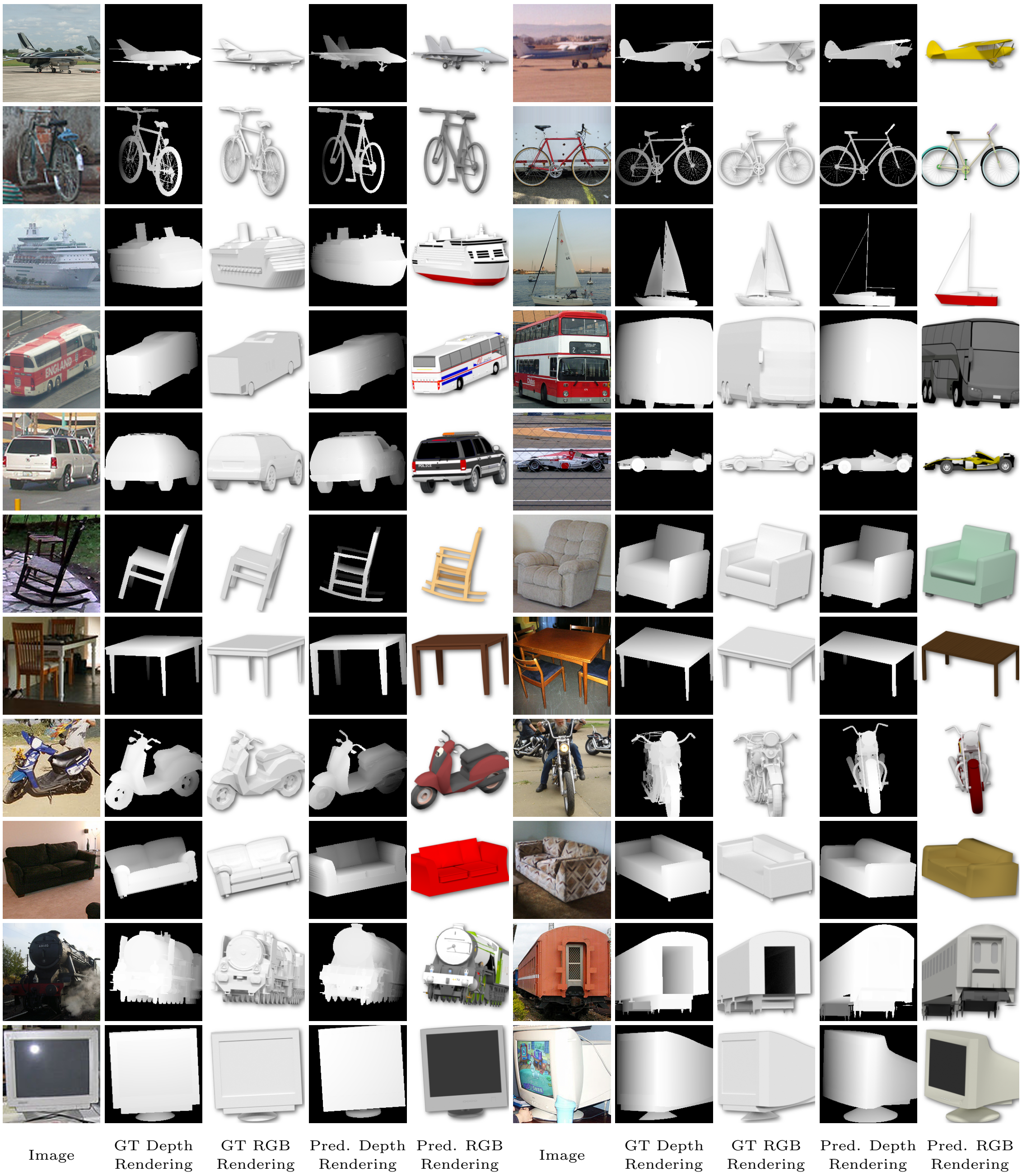


Figure 5.4: Qualitative results for 3D pose estimation and 3D model retrieval from ShapeNet given previously unseen RGB images from Pascal3D+ for all twelve categories. For each category, we show two examples consisting of: the query RGB image; a depth rendering and an RGB rendering of the ground truth 3D model from Pascal3D+ under the ground truth 3D pose from Pascal3D+; a depth rendering and an RGB rendering of our retrieved 3D model from ShapeNet under our predicted 3D pose.

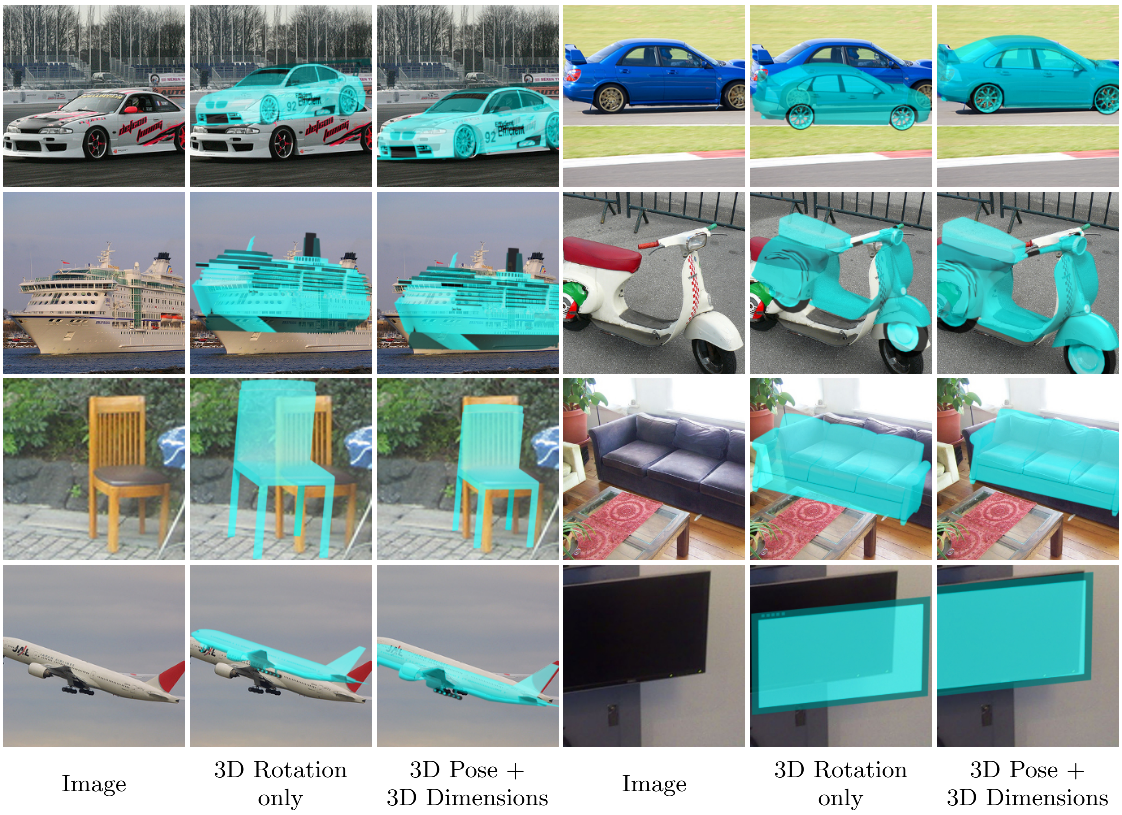


Figure 5.5: We use our predicted 3D pose and 3D dimensions to refine the alignment between the object and a rendering. For each example, we show: The detected object which is not centered on the image window; A rendering which only uses our predicted 3D rotation; A rendering which uses our predicted 3D pose and 3D dimensions.

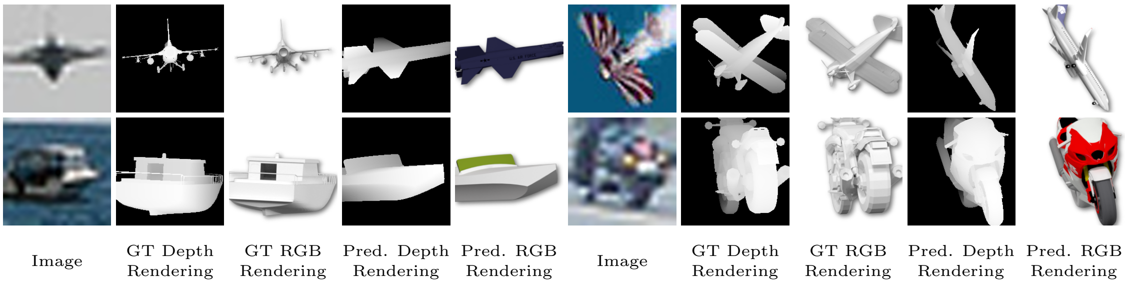


Figure 5.6: Many failure cases of our 3D pose estimation relate to low-resolution images. In fact, for more than 55% of Pascal3D+ validation objects the longer edge of the ground truth 2D bounding box is smaller than 224 pixels which is the fixed spatial input size of pre-trained CNNs like VGG or ResNet. If the input resolution is too low, we cannot predict an accurate 3D pose.

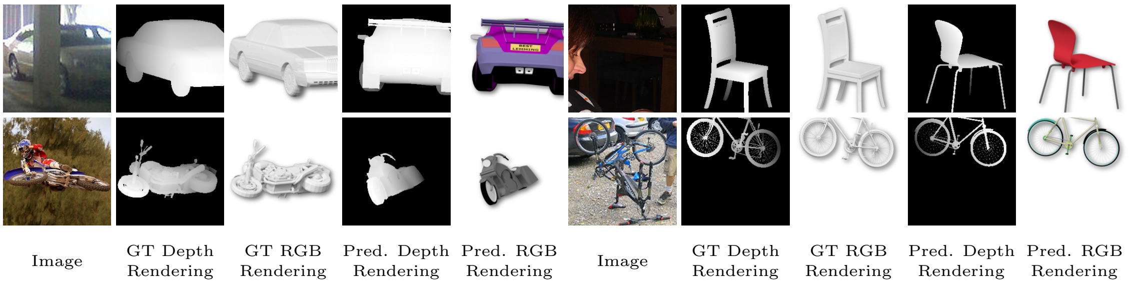


Figure 5.7: Our 3D pose estimation sometimes fails in challenging conditions. We observe that heavy occlusions (top left), bad illumination conditions (top right), and difficult 3D poses (bottom) which are far from the 3D poses seen during training result in incorrect 3D pose predictions. For the last example, not even the annotated ground truth 3D pose is correct.

fails due to low-resolution inputs. After upsampling, the over-smoothed input RGB images lack details and sharp discontinuities, which results in incorrect 3D pose predictions. In fact, even for a human it is difficult to identify the correct 3D poses of the objects in these examples. Fig. 5.7 shows additional failure cases, where heavy occlusions, bad illumination conditions, and difficult 3D poses which are far from the 3D poses seen during training result in incorrect 3D pose predictions.

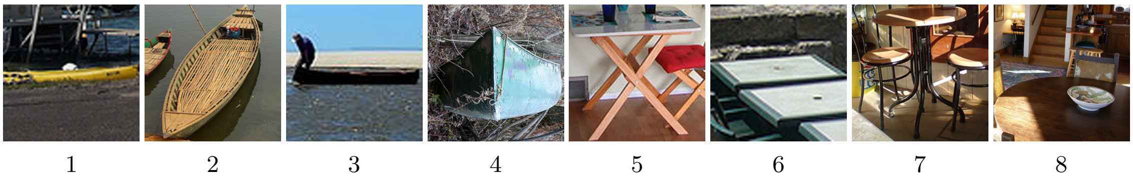


Figure 5.8: Example images of objects with ambiguous 3D poses from the Pascal3D+ validation data. It is impossible to differentiate between the front and back of symmetric boats (1–4). Additionally, many tables are ambiguous with respect to a rotation of π (5), $\frac{\pi}{2}$ (6 and 7), or even have an axis of symmetry like a round table (8).

Next, Fig. 5.8 shows examples for ambiguous objects from Pascal3D+. Many objects have symmetries which make their 3D pose not well defined. For example, it is impossible to differentiate between the front and back of a symmetric boat. This issue is even more apparent for objects of the category *table*: Many tables are ambiguous with respect to a rotation of π, $\frac{\pi}{2}$, or even have an axis of symmetry like a round table. If our approach predicts one of the possible 3D poses that is not the annotated ground truth 3D pose, this is considered as a mistake by the commonly used evaluation protocol [279]. Even though it is not possible to differentiate between the two 3D poses visually.

Finally, our 3D model retrieval sometimes fails in challenging cases. If the environmental conditions in the query RGB image are too difficult, we cannot retrieve an accurate 3D model, even if our predicted 3D pose is accurate. For example, visual distortions due to wide-angle lenses, deformed or demolished objects, and heavy occlusions disturb our 3D model retrieval, as shown in Fig. 5.9.

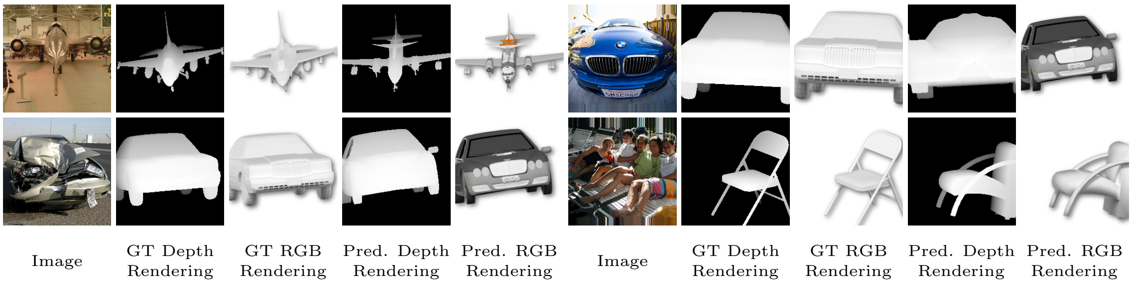


Figure 5.9: Failure cases of our 3D model retrieval for challenging cases, where our 3D pose estimation was successful. The images exhibit fish-eye effects due to wide-angle lenses (top), contain deformed or demolished objects (bottom left), and show heavy occlusions (bottom right) which disturb our 3D model retrieval. However, the annotated ground truth 3D models are also not accurate for these examples.

5.4 Conclusion

3D model retrieval from single RGB images in the wild is an important but challenging task. Existing approaches address this problem by training on vast amounts of synthetic data. However, there is a significant domain gap between real images and synthetic renderings which limits performance in practice. For this reason, we learn to map real RGB images and synthetic depth renderings to a common embedding space. Additionally, we show that estimating the 3D pose of an object is a useful prior for 3D model retrieval. Our approach is scalable as it supports category-agnostic predictions and offline pre-computed descriptors. We do not only outperform the state of the art for 3D viewpoint estimation on Pascal3D+ but also retrieve accurate 3D models from ShapeNet given previously unseen RGB images from Pascal3D+. In Chapters 6 and 7, we will present further improvements to both the 3D pose estimation and the 3D model retrieval part of our proposed system.

CHAPTER 6

3D Pose Estimation with Geometric Projection Parameter Consensus

Contents

In the previous chapter, we presented a novel category-level 3D pose estimation approach for objects in the wild. While this approach is able to make category-agnostic predictions without providing ground truth 3D models at runtime, it has three shortcomings. First, it requires given 2D bounding boxes for the objects in the images. Second, it assumes the ground truth camera focal length is given at runtime. Third, it relies on the prediction of the 2D locations of an object's 3D bounding box corner projections. However, these 2D locations can be far from the actual object and can even lie outside of the image area, which makes them difficult to predict in some cases.

To overcome these limitations, we integrate our 3D pose estimation approach presented in Chapter 5 into an object detection pipeline [96] to deal with multiple objects at different scales in a single image. Additionally, we augment our system to estimate the camera focal length. For this purpose, we combine deep learning techniques and geometric algorithms in a two-stage approach: First, we estimate an initial focal length and establish 2D-3D correspondences from a single RGB image using a deep network. Second, we recover 3D poses and refine the focal length by minimizing the reprojection error of the predicted correspondences. In this way, we exploit the geometric prior given by the focal length for 3D pose estimation. This results in two advantages: First, we achieve significantly improved 3D translation and 3D pose accuracy compared to existing methods. Second, our approach finds a geometric consensus between the individual projection parameters, which is required for precise 2D-3D alignment. Finally, we explore a different form of 2D-3D correspondences for 3D pose estimation. In addition to sparse 2D-3D bounding box corner correspondences, as presented in Chapter 5, we predict dense 2D-3D correspondences by regressing a 3D point for each object pixel. We evaluate our proposed approach on three challenging real-world datasets (Pix3D, Comp, and Stanford) with different object categories and significantly outperform the state of the art by up to 20% absolute in multiple different metrics.

6.1 Introduction

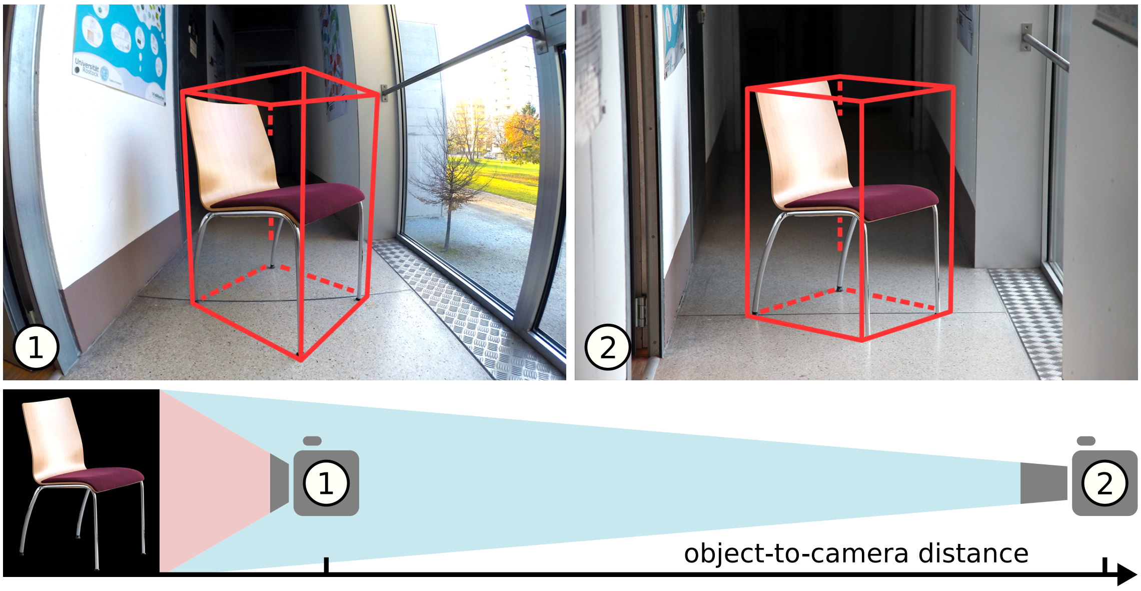


Figure 6.1: Two images of the same scene captured with two cameras having different focal lengths. The appearance of the chair is similar in both images, but the 3D poses are significantly different due to the distinct focal lengths and object-to-camera distances.

3D object pose estimation aims at predicting the 3D rotation and 3D translation of objects relative to the camera. It is a fundamental yet unsolved computer vision problem with many applications including augmented reality, robotics, and 3D scene understanding. Recently, there have been significant advances in 3D object pose estimation from single RGB images on the category level [85, 191, 225, 279], thanks to the revolution of deep learning and the creation of large-scale datasets providing 3D annotations for 2D images [302, 303].

While recent approaches achieve high accuracy in terms of 3D rotation, their accuracy in terms of 3D translation is often low [190, 288]. The main reason for this discrepancy is illustrated in Fig. 6.1, where we compare two images of an object captured with cameras having different focal lengths. The appearance of the object is similar in both images, even though the 3D poses are significantly different. In fact, the appearance of an object in an image is not only determined by the 3D pose, but also by the camera intrinsics. While changes in the 3D rotation always significantly effect the appearance, changes in the 3D translation do not if the translation direction and the ratio between the object-to-camera distance and the focal length remain constant. Thus, estimating the 3D translation of objects from RGB images in the case of unknown intrinsics is highly ambiguous.

Existing approaches either assume that the focal length is given at runtime [85, 191], that the focal length is constant for all images [156], or that the 3D pose estimation method will implicitly learn the subtle appearance variations caused by different focal lengths from the data and adapt the prediction accordingly [190, 288]. In practice, however, none of these assumptions holds. In the wild, the focal length is unknown, images are captured with different cameras, and deep networks do not discover the underlying geometric relations of the problem without explicit guidance.

To overcome these limitations, we propose to explicitly estimate and integrate the focal length into the 3D pose estimation. For this purpose, we introduce a two-stage approach that combines deep learning techniques and geometric algorithms. In the first stage, we estimate an initial focal length and establish 2D-3D correspondences from a single RGB image using a deep network. In the second stage, we perform a geometric optimization on the predicted correspondences to recover 3D poses and refine the focal length. In particular, we minimize the reprojection error between predicted 2D locations and 3D points subject to the 3D rotation, 3D translation, and focal length by solving a PnPf problem (see Section 2.1.2). In this way, we exploit the geometric prior given by the focal length for 3D pose estimation.

In contrast to existing approaches, which also predict 3D poses and the focal length but only perform an independent estimation of the individual parameters [288], our approach has two main advantages: First, explicitly modeling the focal length in the 3D pose estimation yields significantly improved 3D translation and 3D pose accuracy. Second, our approach finds a geometric consensus between 3D poses and the focal length. This results in a significantly improved 2D-3D alignment when projecting 3D models of objects back onto the image, which is important for many applications like augmented reality. Thus, we call our method *Geometric Projection Parameter Consensus* (GP^{2}C).

In addition, we explore two possible methods for establishing 2D-3D correspondences from single RGB images which approach the task from different directions. Our first method predicts 3D points for known 2D locations by estimating a 3D point for each object pixel [22, 24, 125]. Our second method predicts 2D locations for known 3D points by estimating the 2D locations of the object's 3D bounding box corner projections [85, 217, 269], as presented in Chapter 5. Our experiments show that both 2D-3D correspondence estimation methods achieve comparable overall accuracy, but each method has its respective advantages and disadvantages. Thus, we provide a detailed discussion comparing the two methods in our evaluation.

Finally, we integrate our proposed method into an object detection pipeline [96]. In this way, we predict 2D-3D correspondences for multiple objects at different scales in a single image and estimate the focal length using a single deep network. Both of our 2D-3D correspondence estimation methods elegantly fit into this framework. As a consequence, we avoid computationally expensive redundant feature calculations for individual tasks and speed up training.

To demonstrate the benefits of our joint 3D pose and focal length estimation approach, we evaluate it on three challenging real-world datasets with different object categories: Pix3D [258] (*bed*, *chair*, *sofa*, *table*), Comp [288] (*car*), and Stanford [288] (*car*). We present quantitative as well as qualitative results and significantly outperform the state of the art. To summarize, our main contributions are:

- We present the first method for joint 3D pose and focal length estimation that enforces a geometric consensus between 3D poses and the focal length.

- We outperform the state of the art on three challenging real-world datasets by up to 20% absolute in multiple metrics covering different aspects of projective geometry including 3D translation, 3D pose, focal length, and projection accuracy.

6.2 Joint 3D Pose and Focal Length Estimation

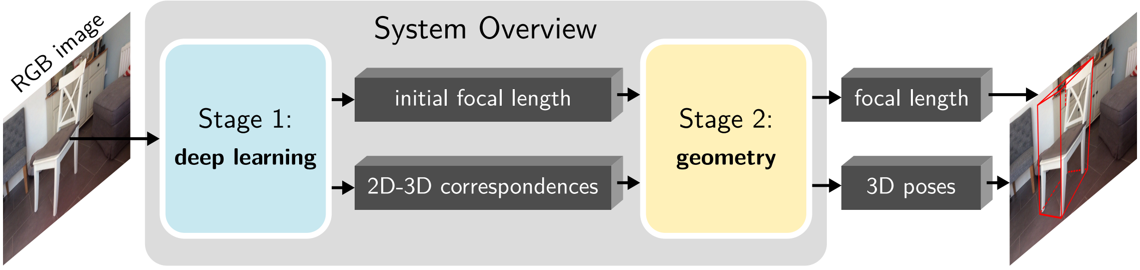


Figure 6.2: Overview of our proposed two-stage approach. **Stage 1:** We first predict an initial focal length and establish 2D-3D correspondences for each object using deep learning. **Stage 2:** We then perform a geometric optimization on the predicted correspondences to recover 3D poses and refine the initial focal length.

Given a single RGB image, we want to predict the camera focal length and the 3D pose of each object in an image. For this purpose, we introduce a two-stage approach that combines deep learning techniques and geometric algorithms, as shown in Fig. 6.2. In the first stage, we predict an initial focal length and establish 2D-3D correspondences using deep learning (see Section 6.2.1). In the second stage, we perform a geometric optimization on our predicted correspondences to recover 3D poses and refine the initial focal length (see Section 6.2.2).

6.2.1 Deep Focal Length and 2D-3D Correspondence Estimation

To predict the focal length as well as 2D-3D correspondences for multiple objects with a single deep network, we extend the generalized Faster/Mask R-CNN framework [96, 225]. This generic multi-task framework includes a 2D object detection pipeline to perform per-image and per-object computations. In this way, we address multiple different tasks using a single end-to-end trainable network. For our implementation, we use a Feature Pyramid Network [163] on top of a ResNet-101 backbone [99, 100] and fine-tune a model pre-trained for instance segmentation on COCO [165].

In the context of the generalized Faster/Mask R-CNN framework, an output branch provides one or more subnetworks with different structure and functionality. We introduce two dedicated output branches for estimating the focal length and 2D-3D correspondences alongside the existing object detection branches.

Focal Length: The focal length branch provides one subnetwork which performs a per-image computation, as illustrated in Fig. 6.4. In this case, we regress a scalar for each image from the entire spatial resolution of the shared feature maps computed by the convolutional network backbone. In contrast to previous work, we propose to regress a logarithmic parametrization of the focal length

$$y_f = \ln(f), \tag{6.1}$$

instead of predicting the focal length f directly [288], which has two advantages: First, the logarithmic parametrization reduces the bias towards minimizing the error on long focal lengths during the optimization of the network. This is meaningful because, regarding the estimation of the focal length, the relative error is more important than the absolute error. Second, the logarithmic parametrization achieves a more balanced sensitivity across the entire range of the focal length. Otherwise, the sensitivity is significantly higher for short focal lengths than for long focal lengths. During training, we optimize y_f in a supervised manner using the Huber loss [118].

2D-3D correspondences: For establishing 2D-3D correspondences, we explore two distinct methods. Both methods approach the problem from different directions and produce significantly different correspondences and representations, as shown in Fig. 6.3. However, our overall approach works with any kind of 2D-3D correspondences and does not depend on a specific format. Thus, the method for establishing correspondences can be exchanged. This is extremely useful because different methods have their respective advantages and disadvantages which we discuss in our experiments in Section 6.3.5.

Our first method predicts 3D points for known 2D locations. In particular, we establish correspondences between 2D image pixels which belong to the object and 3D coordinates on the surface of the object. We represent these correspondences in the form of a location field (LF) [288], which provides dense 2D-3D correspondences in an image-like format, as

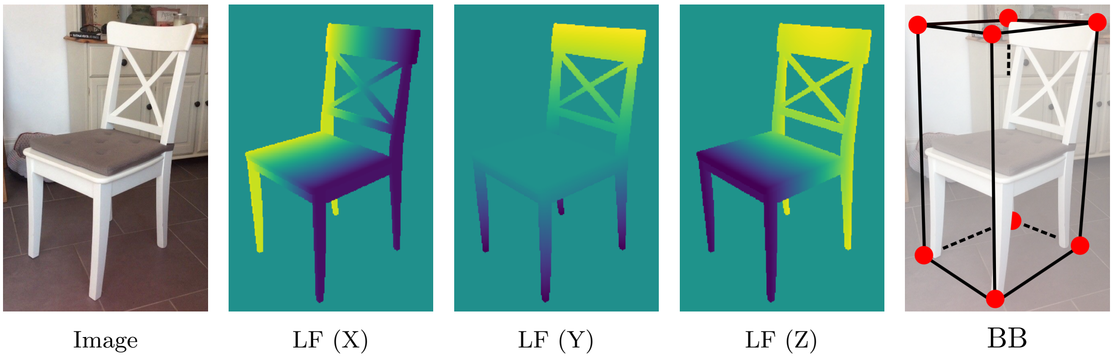


Figure 6.3: Visualization of two different forms of 2D-3D correspondences for an object: A location field which encodes XYZ 3D coordinates for each pixel (LF) and the 2D projections of the object's 3D bounding box corners (BB).

shown in Fig. 6.3. A location field has the same size and spatial resolution as its reference RGB image, but the three channels encode XYZ 3D coordinates in the object coordinate system instead of RGB colors. Due to its image-like structure, this representation is well-suited for regression with a CNN.

Our second method predicts 2D locations for known 3D points. In this case, we predict the 2D projections of the object's 3D bounding box corners (BB) [217], as shown in Fig. 6.3. Since the 3D coordinates of the bounding box corners are unknown during inference, we also predict the 3D dimensions of the object along the XYZ axis [85] from which we derive the required 3D points. We represent these sparse 2D-3D correspondences in the form of a 19-dimensional vector, which consists of the 2D locations of the eight bounding box corners (16 values) and the 3D dimensions of the object (3 values), as already explained in Section 5.2.1.

As shown in Fig. 6.4, we implement a separate 2D-3D correspondences branch for each method. In contrast to the focal length branch, both branches perform region-based per-object computations: For each detected object, an associated spatial region of interest in the feature maps is aligned to a fixed size feature representation with a low spatial resolution, e.g., 14 × 14. These aligned features serve as an input to one of our two proposed correspondences branches. Thus, the chosen 2D-3D correspondences branch is evaluated N times for each image, where N is the number of detected objects. We identify the chosen 2D-3D correspondences method by adding a suffix: Ours-LF or Ours-BB.

For the LF method, the correspondences branch provides two different fully convolutional subnetworks to predict a tensor of 3D points and a 2D object mask at a spatial resolution of 28 × 28. The 2D mask is then applied to the tensor of 3D points to get a low-resolution location field. We found this approach to produce significantly higher accuracy compared to directly regressing a low-resolution location field which tends to predict over-smoothed 3D coordinates around the object silhouette.

The resulting low-resolution location field can be upscaled and padded to obtain a high-

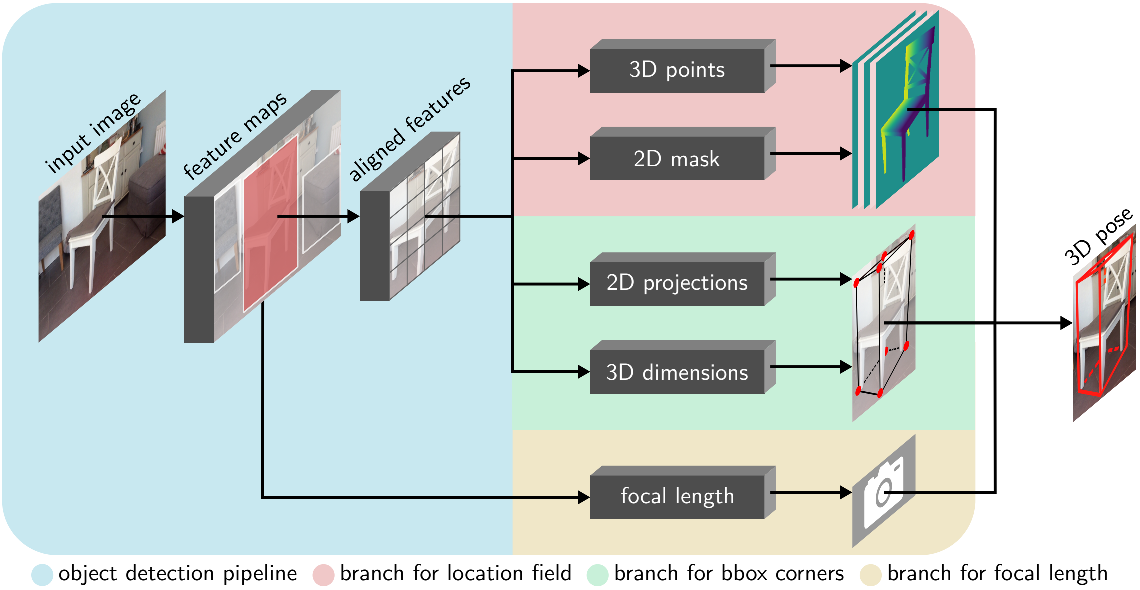

input image
feature maps
aligned features
3D points
2D mask
2D projections
3D dimensions
focal length
3D pose
● object detection pipeline
● branch for location field
● branch for bbox corners
● branch for focal length

Figure 6.4: Illustration of our network architecture. Our focal length branch takes the entire shared feature maps as an input to predict an initial focal length and is evaluated once per image. In contrast, our two correspondences branches take aligned features for each object as an input to predicted 2D-3D correspondences (LF or BB) and are evaluated once for each detected object.

resolution location field with the same spatial resolution as the input image. However, we sample 2D-3D correspondences from the low-resolution location field and only adjust their 2D locations to match the input image resolution. In this way, we avoid generating a large number of 2D-3D correspondences without providing additional information.

For the BB method, the correspondences branch also provides two subnetworks, but this time with fully connected output layers. One subnetwork predicts the 2D locations of the object's 3D bounding box corners, the other subnetwork estimates the 3D dimensions of the object. In this case, we regress the 2D location in normalized coordinates relative to the spatial resolution of the aligned features. Again, we adjust the predicted 2D locations to match the input image resolution.

During training, we optimize the 3D points and 2D mask (Ours-LF), or the 2D projections and 3D dimensions (Ours-BB) in a supervised manner using the Huber loss [118]. The final network loss is a combination of our focal length loss, our chosen 2D-3D correspondences loss, and the 2D object detection losses of the generalized Faster/Mask R-CNN framework [96, 225].

6.2.2 Geometric Optimization

Once we established correspondences between 2D locations and 3D points and predicted an initial focal length, we use the same geometric optimization for all methods. In this case, we perform a non-linear optimization of the PnPf problem [192] which finds a geometric consensus between the individual projection parameters. In particular, we optimize

$$\min_{\mathbf{K},\mathbf{R},\mathbf{t}} \sum_{i=1}^{N} \mathcal{L}\Big(\|\mathbf{m}_i - \mathrm{proj}(\mathbf{M}_i, \mathbf{K}, \mathbf{R}, \mathbf{t})\|_2\Big) \tag{6.2}$$

which minimizes the reprojection error between corresponding 3D points $\mathbf{M}_i$ and 2D locations $\mathbf{m}_i$. In this case, $\mathrm{proj}(\cdot)$ performs the projection of 3D points in the object coordinate system to 2D locations in the image plane with respect to the rotation $\mathbf{R}$, translation $\mathbf{t}$, and camera intrinsics $\mathbf{K}$. As presented in Section 2.1.1.2, $\mathbf{K}$ is parametrized only by the focal length f. Finally, $\mathcal{L}(\cdot)$ is a loss function, such as the standard squared loss $\mathcal{L}(x) = x^2$ or the more robust Cauchy loss [274] $\mathcal{L}(x) = \ln(1 + x^2)$, and N denotes the number of correspondences.

In this way, we perform an optimization over both the 3D pose and the focal length. In this case, a minimum of four 2D-3D correspondences is needed to find a unique solution [299] because each correspondence gives two independent equations and we optimize seven parameters: the 3-DoF rotation, the 3-DoF translation, and the 1-DoF focal length. In practice, however, it is important to use more 2D-3D correspondences to compensate for the presence of noise.

Following the strategy of previous PnP(f) approaches [103, 155, 206], we compute an initial solution in $O(n)$ time followed by an iterative refinement technique. For our initial solution, we compute the 3D rotation and 3D translation using EPnP [155] with our predicted focal length. Providing a good initial focal length is a key factor in achieving high accuracy in terms of 3D translation. In theory, it is also possible to recover the focal length using 2D-3D correspondences from scratch [192, 206] but this requires extremely accurate and clean correspondences. For 2D-3D correspondence estimation on the category level in the wild, however, we are facing fuzzy and noisy predictions. In this case, a low reprojection error is achieved by finding the correct ratio between the object-to-camera distance and the focal length. Thus, we cannot assume that the geometric optimization will find the correct absolute focal length from scratch.

Taking this into account, we jointly optimize the 3D rotation, 3D translation, and focal length during our iterative refinement. For this purpose, we employ a Newton-Step-based optimization [40] depending on the loss function $\mathcal{L}(\cdot)$, i.e., Levenberg-Marquardt [189] (squared loss) or Subspace Trust-Region Interior-Reflective [27] (Cauchy loss).

Our approach naturally handles different projection models (egocentric or allocentric) [151]. Additionally, jointly optimizing the 3D poses of multiple objects in an image together with the focal length is straightforward. In this case, we compute the initial solution as before but perform our iterative refinement for $1 + 6N$ parameters where N is the number of detected objects. We did not evaluate this joint refinement though because available category-level datasets with focal length annotations just provide 3D annotations for one object per image, even if there are multiple objects in the image [258, 288]. In most cases, we are still able to detect the other objects but do not have ground truth annotations to evaluate them, as shown in our qualitative results in Section 6.3.2. Moreover, our approach can readily be extended to deal with more complex camera models including skew, off-center principal point, asymmetric aspect ratio, or lens distortions [192]. However, currently there are no datasets with this kind of annotations, as discussed in Chapter 4.

6.3 Evaluation

To demonstrate the benefits of our joint 3D pose and focal length estimation approach (GP^{2}C), we evaluate it on three challenging real-world datasets with different object categories: Pix3D [258] (*bed*, *chair*, *sofa*, *table*), Comp [288] (*car*), and Stanford [288] (*car*). Details on the evaluated datasets are presented in Chapter 4. In particular, we provide a quantitative and qualitative evaluation of our approach in comparison to the state of the art in Section 6.3.2, analyze important aspects in Section 6.3.3, investigate failure modes in Section 6.3.4, and discuss advantages and disadvantages of our two presented methods for establishing 2D-3D correspondences in Section 6.3.5. To cover different aspects of projective geometry in our evaluation, we use the following well-established metrics:

Detection: We report the detection accuracy $Acc_{D_{0.5}}$ which gives the percentage of objects for which the intersection over union between the ground truth 2D bounding box and the predicted 2D bounding box is larger than 50% [303]. This metric is an upper bound for other Acc metrics since we do not make blind predictions.

Rotation: We compute the geodesic distance

$$e_{\mathbf{R}} = \frac{\| \log(\mathbf{R}_{\text{gt}}^T \mathbf{R}_{\text{pred}}) \|_F}{\sqrt{2}} \tag{6.3}$$

between the ground truth 3D rotation matrix $\mathbf{R}_{\text{gt}}$ and the predicted 3D rotation matrix $\mathbf{R}_{\text{pred}}$ which gives the minimal angular distance. We report the median of this distance for all objects ($MedErr_{\mathbf{R}}$) and the percentage of objects for which the distance is below the threshold of $\frac{\pi}{6}$ or 30° ($Acc_{\mathbf{R}\frac{\pi}{6}}$) [279].

Translation: We calculate the relative translation distance

$$e_{\mathbf{t}} = \frac{\|\mathbf{t}_{\text{gt}} - \mathbf{t}_{\text{pred}}\|_2}{\|\mathbf{t}_{\text{gt}}\|_2} \tag{6.4}$$

between the ground truth 3D translation $\mathbf{t}_{\text{gt}}$ and the predicted 3D translation $\mathbf{t}_{\text{pred}}$ [109]. We report the median of this distance for all objects ($MedErr_{\mathbf{t}}$).

Pose: We compute the average normalized distance of all transformed 3D model points in 3D space

$$e_{\mathbf{R},\mathbf{t}} = \operatorname*{avg}_{\mathbf{M}\in\mathcal{M}} \frac{d_{\text{bbox}}}{d_{\text{img}}} \cdot \frac{\|\text{transf}(\mathbf{M},\mathbf{R}_{\text{gt}},\mathbf{t}_{\text{gt}}) - \text{transf}(\mathbf{M},\mathbf{R}_{\text{pred}},\mathbf{t}_{\text{pred}})\|_2}{\|\mathbf{t}_{\text{gt}}\|_2} \tag{6.5}$$

to evaluate 3D pose accuracy [105, 109]. In this case, transf$(\cdot)$ describes a transformation in three-dimensional Euclidean space subject to a 3D rotation and a 3D translation. Thus, each 3D point $\mathbf{M}$ of the ground truth 3D model $\mathcal{M}$ is transformed using the ground truth 3D pose ($\mathbf{R}_{\text{gt}}$ and $\mathbf{t}_{\text{gt}}$) and the predicted 3D pose ($\mathbf{R}_{\text{pred}}$ and $\mathbf{t}_{\text{pred}}$). We normalize this distance by the relative size of the object in the image using the ratio between the ground truth 2D bounding box diagonal d_{bbox} and the image diagonal d_{img}, and the L2-norm of the ground truth translation $\|\mathbf{t}_{\text{gt}}\|_2$. This normalization provides an unbiased metric for 3D pose evaluation in the case of unknown intrinsics. We report the median of this distance for all objects ($MedErr_{\mathbf{R},\mathbf{t}}$).

Focal Length: We calculate the relative focal length error

$$e_f = \frac{|f_{\text{gt}} - f_{\text{pred}}|}{f_{\text{gt}}} \tag{6.6}$$

between the ground truth focal length f_{gt} and the predicted focal length f_{pred} [206, 296]. We report the median of this error for all objects ($MedErr_f$).

Projection: To evaluate all projection parameters, we compute the average normalized reprojection distance

$$e_P = \operatorname*{avg}_{\mathbf{M}\in\mathcal{M}} \frac{\|\text{proj}(\mathbf{M},\mathbf{K}_{\text{gt}},\mathbf{R}_{\text{gt}},\mathbf{t}_{\text{gt}}) - \text{proj}(\mathbf{M},\mathbf{K}_{\text{pred}},\mathbf{R}_{\text{pred}},\mathbf{t}_{\text{pred}})\|_2}{d_{\text{bbox}}} . \tag{6.7}$$

In this case, proj$(\cdot)$ describes a projection from 3D to 2D subject to a 3D rotation, a 3D translation, and a focal length. Thus, each 3D point $\mathbf{M}$ of the ground truth 3D model $\mathcal{M}$ is projected to a 2D location using the ground truth projection parameters ($\mathbf{K}_{\text{gt}}$, $\mathbf{R}_{\text{gt}}$, and $\mathbf{t}_{\text{gt}}$) and the predicted projection parameters ($\mathbf{K}_{\text{pred}}$, $\mathbf{R}_{\text{pred}}$, and $\mathbf{t}_{\text{pred}}$). As presented in Section 2.1.1.2, $\mathbf{K}$ is parametrized only by the focal length f. Finally, d_{bbox} is the ground truth 2D bounding box diagonal. We report the median of this distance for all objects ($MedErr_P$) and the percentage of objects for which the distance is below the threshold of 0.1 ($Acc_{P_{0.1}}$) [288].

6.3.1 Implementation Details

For our implementation, we resize and pad images to a spatial resolution of 512 × 512 maintaining the aspect ratio. In this way, we are able to use a minibatch size of 6 on a 12GB GPU. We train our networks for 200 epochs and employ a staged training strategy for fine-tuning a model pre-trained on COCO [165]: First, we freeze all pre-trained weights and only train our focal length and 2D-3D correspondences branches using a learning rate of $1e^{-3}$. During training, we gradually unfreeze all network layers and, finally, train the entire model using a learning rate of $1e^{-4}$.

We employ different forms of data augmentation commonly used in object detection [96]. In this case, some techniques like mirroring or jittering of location, scale, and rotation require adjusting the training targets accordingly, while independent pixel augmentations like additive noise do not.

Balancing individual loss terms is crucial for training a multi-task network. We weight the focal loss with 0.1, the 2D-3D correspondences loss with 10.0, and the object detection loss with 1.0, however, the specific numbers are highly dependent on the implementation.

6.3.2 Comparison to the State of the Art

We first present quantitative results of our approach using our two different methods for establishing 2D-3D correspondences (Ours-LF and Ours-BB) and compare them to the state of the art. In fact, there is little research on joint 3D pose and focal length estimation in the wild because datasets with this kind of annotations have only been available recently (see Chapter 4). Existing 3D pose estimation methods either assume that the ground truth focal length is given at runtime [85, 191] or evaluate on datasets which were captured using a single camera with constant focal length [156]. However, we reimplement the approach of [288] which also targets 3D pose estimation from uncalibrated single RGB images and achieve comparable results, even outperforming their reported $MedErr_P$ and $Acc_{P_{0.1}}$ scores due to our improved backbone architecture and initialization. The results are summarized in Table 6.1. We achieve consistent results across all datasets and categories, thus, we provide a joint discussion based on the evaluated metrics:

Detection: All methods achieve high detection accuracy ($Acc_{D_{0.5}}$). This is not surprising because we fine-tune a model pre-trained for instance segmentation on COCO [165]. In fact, all evaluated categories are also present in COCO.

Rotation: Also, all methods achieve high 3D rotation accuracy ($MedErr_{\mathbf{R}}$ and $Acc_{\mathbf{R}\frac{\pi}{6}}$). Our reported numbers are in line with the results of previous work on 3D rotation estimation in the wild [85, 279, 288] and confirm that 3D rotation can robustly be recovered from 2D observations up to a certain precision. Only for the category *table*, we observe sub-average accuracy. In fact, almost all tables have symmetries, as can be seen in Fig. 6.6. The resulting ambiguous object appearances sometimes confuse all evaluated methods because they predict a single 3D pose rather than a distribution (see *table* in Fig. 6.11).

Translation: In terms of 3D translation accuracy ($MedErr_{\mathbf{t}}$), our approach significantly outperforms the state of the art. Directly predicting the 3D translation from a local image

			Detection	Rotation		Translation	Pose	Focal	Projection	
Method	Dataset	Class	$Acc_{D_{0.5}}$	$MedErr_{\mathbf{R}}$ $\cdot 1$	$Acc_{\mathbf{R}_{\frac{\pi}{6}}}$	$MedErr_{\mathbf{t}}$ $\cdot 10^1$	$MedErr_{\mathbf{R},\mathbf{t}}$ $\cdot 10^1$	$MedErr_f$ $\cdot 10^1$	$MedErr_P$ $\cdot 10^2$	$Acc_{P_{0.1}}$
[288]			98.4%	5.82	95.3%	1.95	1.56	2.22	6.05	74.9%
Ours-LF	Pix3D	bed	99.0%	**5.13**	96.3%	**1.41**	**1.04**	**1.43**	**3.52**	90.6%
Ours-BB			**99.5%**	5.40	**97.9%**	1.66	1.17	1.59	3.55	**93.2%**
[288]			94.9%	7.52	88.0%	2.69	1.58	1.98	6.04	75.3%
Ours-LF	Pix3D	chair	95.2%	7.52	88.8%	1.92	1.21	1.62	3.41	88.2%
Ours-BB			**97.3%**	**6.95**	**91.0%**	**1.68**	**1.08**	**1.58**	**3.24**	**90.9%**
[288]			96.5%	4.73	94.8%	2.28	1.62	2.42	4.33	82.2%
Ours-LF	Pix3D	sofa	96.5%	4.49	95.0%	1.92	1.33	1.79	2.56	93.7%
Ours-BB			**98.3%**	**4.40**	**97.0%**	**1.63**	**1.16**	**1.73**	**2.13**	**95.6%**
[288]			94.0%	10.94	72.9%	3.16	2.28	3.03	8.90	53.6%
Ours-LF	Pix3D	table	94.0%	**10.53**	73.5%	**2.16**	**1.62**	**2.05**	5.92	69.5%
Ours-BB			**95.7%**	10.80	**77.2%**	2.81	1.78	2.10	**5.74**	**72.4%**
[288]			96.0%	7.25	87.8%	2.52	1.76	2.41	6.33	71.5%
Ours-LF	Pix3D	*mean*	96.2%	6.92	88.4%	**1.85**	**1.30**	**1.72**	3.85	85.5%
Ours-BB			**97.7%**	**6.89**	**90.8%**	1.94	**1.30**	1.75	**3.66**	**88.0%**
[288]			**98.9%**	5.24	97.6%	3.30	2.35	3.23	7.85	73.7%
Ours-LF	Comp	car	98.8%	5.23	97.9%	2.61	1.86	2.97	4.21	95.1%
Ours-BB			**98.9%**	**4.87**	**98.1%**	**2.55**	**1.84**	**2.95**	**3.87**	**95.7%**
[288]			**99.6%**	5.43	98.0%	2.33	1.80	2.34	7.46	76.4%
Ours-LF	Stanford	car	**99.6%**	5.38	**98.3%**	1.93	1.51	**2.01**	3.72	96.2%
Ours-BB			**99.6%**	**5.24**	**98.3%**	**1.92**	**1.47**	2.07	**3.25**	**96.5%**

Table 6.1: Quantitative results on the Pix3D, Comp, and Stanford datasets. We significantly outperform the state of the art in the 3D translation, 3D pose, focal length, and projection metrics. We explain the reported numbers for each metric in detail in Section 6.3.2.

window of an object is highly ambiguous in the case of unknown intrinsics. By explicitly estimating and integrating the focal length into the 3D pose estimation, we exploit a geometric prior and achieve a relative improvement of 20%.

Pose: In the case of unknown intrinsics, the 3D pose accuracy ($MedErr_{\mathbf{R},\mathbf{t}}$) is primarily governed by the 3D translation accuracy. Hence, we also observe a relative improvement of 20% compared to the state of the art.

Focal Length: Considering the focal length accuracy ($MedErr_f$), our approach outperforms the state of the art by a relative improvement of 10% due to our logarithmic parametrization and correspondence-based refinement.

Projection: Finally, we report the projection metrics ($MedErr_P$ and $Acc_{P_{0.1}}$) which evaluate all predicted parameters. In these metrics, we achieve the largest improvement compared to the state of the art: 20% absolute in $Acc_{P_{0.1}}$ and 40% relative in $MedErr_P$ across all datasets. In contrast to an independent estimation of the individual projection parameters, our approach finds a geometric consensus which results in improved 2D-3D alignment and reprojection error. This quantitative improvement is also reflected in our qualitative results shown in Figs. 6.5 and 6.6. In this experiment, our approach consistently

		Projection	
Method	PnP Strategy	$MedErr_P \cdot 10^2$	$Acc_{P_{0.1}}$
Ours-LF	Standard	3.88	85.3%
	RANSAC	3.87	85.4%
	Cauchy	**3.85**	**85.5%**
Ours-BB	Standard	3.68	87.5%
	RANSAC	3.68	87.6%
	Cauchy	**3.66**	**88.0%**

Table 6.2: Evaluation of different PnP strategies. The results show that our predicted 2D-3D correspondences are reliable and do not contain single extreme outliers.

produces a higher quality 2D-3D alignment compared to the state of the art for objects of different categories. This significant improvement can be accounted to the fact that we minimize the reprojection error during inference. However, we want to emphasize that the 3D model is only used for the evaluation. The 3D poses and focal length are solely computed from a single RGB image in our approach.

6.3.3 Analysis

Next, we analyze two important aspects of our approach: (1) the robustness of our predicted 2D-3D correspondences and (2) the importance of the focal length for estimating 3D poses from these correspondences. For this purpose, we perform experiments on Pix3D, which is the most challenging dataset because it provides multiple object categories and has the largest variation in object scale. However, the reported results and insights also transfer to other datasets.

2D-3D Correspondences: First, we run our approaches using different PnP strategies and compare the obtained results using the projection metrics ($MedErr_P$ and $Acc_{P_{0.1}}$) in Table 6.2. In particular, we compare the standard approach, which is sensitive to outliers due to the squared loss $\mathcal{L}(x) = x^2$, to the more robust RANSAC scheme and Cauchy loss [274] $\mathcal{L}(x) = \ln(1 + x^2)$.

All three PnP strategies achieve similar performance for both Ours-LF and Ours-BB. This experiment shows that our predicted 2D-3D correspondences do not contain single extreme outliers which are often present in traditional approaches based on 2D keypoints (see Section 3.2). This is due to the fact that all 2D-3D correspondences are computed from a low dimensional feature embedding which produces consistent predictions. Qualitative examples of our predicted 2D-3D correspondences are presented in Figs. 6.7 and 6.8.

Considering our predicted location fields, we observe that the overall shape of the object is recovered very accurately. In specific cases, thin object parts and details are not detected, e.g., the skinny legs of a table or the ornaments of a bed as shown in Fig. 6.8. To

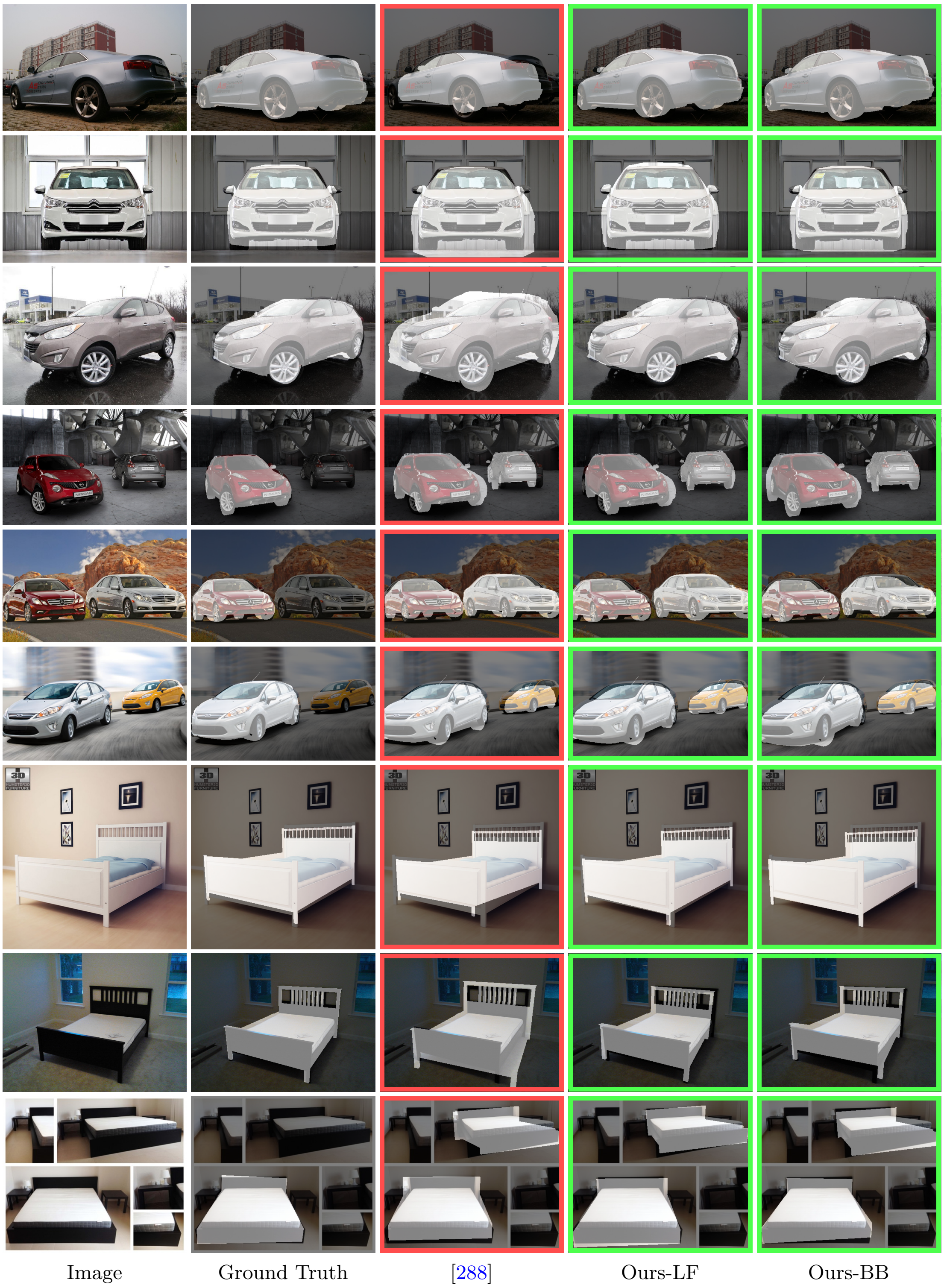


Figure 6.5: Qualitative 3D pose and focal length estimation results for all evaluated datasets and categories. We project the ground truth 3D model onto the image using the 3D pose and focal length predicted by different approaches. In contrast to [288] (red frames), our methods find a geometric consensus between the parameters which results in improved 2D-3D alignment, e.g., the scale of the rendering is more accurate (green frames). All images just provide 3D annotations for one object per image. However, we are able to detect and predict 3D poses for multiple objects.

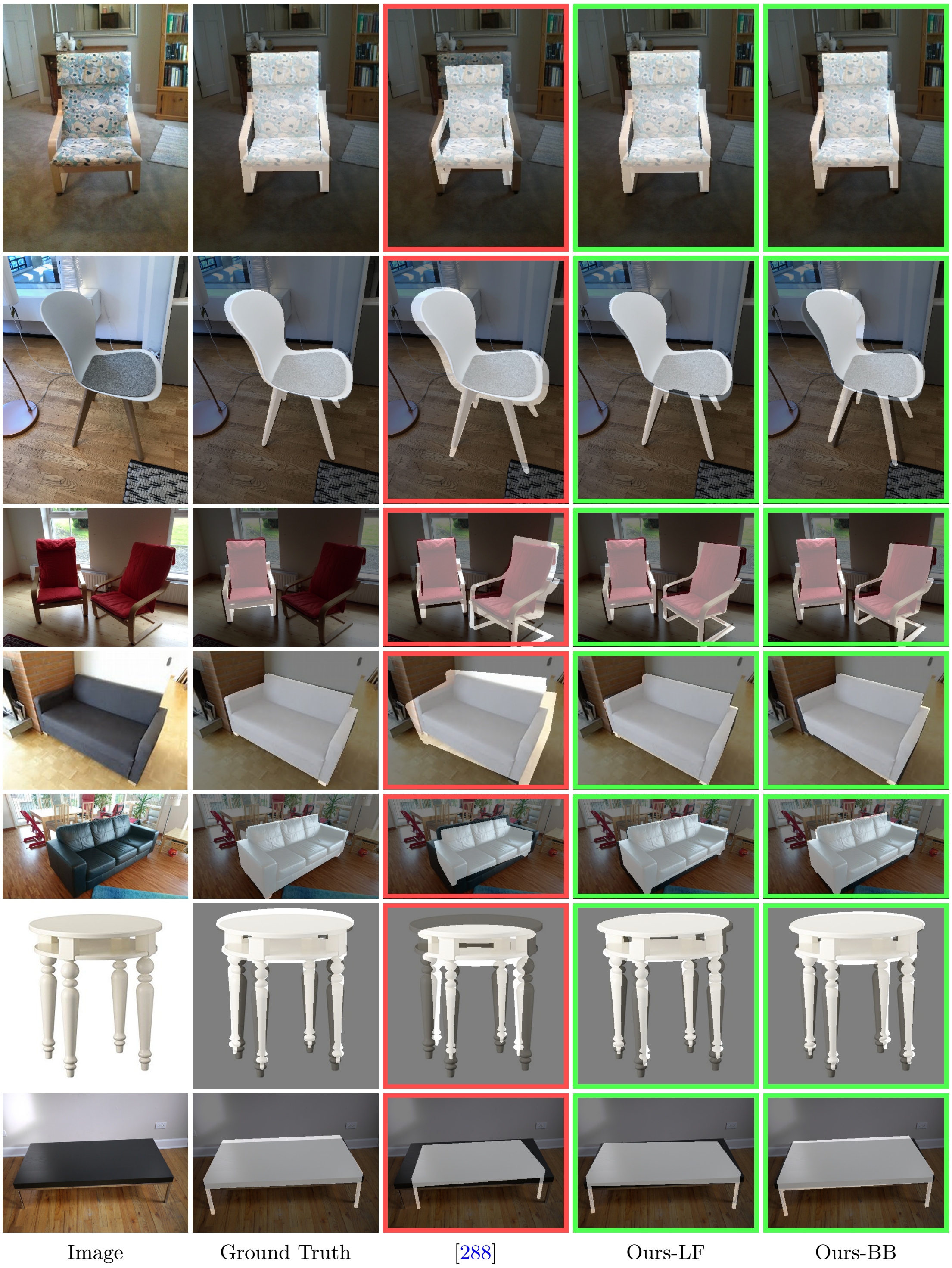


Figure 6.6: Qualitative 3D pose and focal length estimation results for all evaluated datasets and categories. We project the ground truth 3D model onto the image using the 3D pose and focal length predicted by different approaches. In contrast to [288] (red frames), our methods find a geometric consensus between the parameters which results in improved 2D-3D alignment, e.g., the scale of the rendering is more accurate (green frames). All images just provide 3D annotations for one object per image. However, we are able to detect and predict 3D poses for multiple objects.

address this issue, the spatial resolution of the predicted location field can be increased. In this work, we follow the architecture of Mask R-CNN's mask branch and use a spatial resolution of 28×28 [96].

Considering our 3D bounding box corner projections, we observe that the predicted 2D locations are close to the ground truth 2D locations. Also, the perspective box shape is well recovered and there is a consensus between the individual points. In some cases, the predictions are even accurate for corners which project outside the image area, as shown in Fig. 6.7. Also the 3D dimensions are predicted accurately, as already discussed in Section 5.3.1.

However, visually looking at the raw 2D-3D correspondences, we observe that basically all of the correspondences are noisy but the magnitude of the noise is rather low. If our prediction fails, e.g., for an object with symmetries, entire regions of 2D-3D correspondences are corrupt. In such cases, we cannot estimate an accurate 3D pose, not even with robust methods.

Focal Length: Second, to demonstrate the importance of the focal length for estimating 3D poses from 2D-3D correspondences, we initialize our geometric optimization with three different focal lengths and compare the results using the 3D pose distance in Fig. 6.9. In this experiment, we plot the percentage of objects for which the 3D pose distance is below a threshold varying in the range [0,1] ($Acc_{\mathbf{R},\mathbf{t}}$).

As expected, if we initialize the geometric optimization with the ground truth focal length, we achieve the highest 3D pose accuracy. However, for 3D pose estimation in the wild, the focal length is unknown at runtime. In this case, we can use a constant or a predicted focal length for initialization. Even if we use the best possible constant focal length, which is the mode of the focal length distribution of the training dataset, the accuracy drops significantly. This case is representative of the performance of our 3D pose estimation approach presented in Chapter 5 which assumes the focal length is given at runtime. Instead, if we initialize our geometric optimization using our predicted focal length, we achieve improved 3D pose accuracy. However, there is still a gap in the accuracy compared to using the ground truth focal length.

This performance gap results from appearance ambiguities caused by different focal lengths, as emphasized by a qualitative example shown in Fig. 6.10. In this experiment, we again initialize our geometric optimization with three different focal lengths (ground truth, predicted, and constant). We use the predicted 3D pose and focal length to project the ground truth 3D model onto the image and additionally visualize the predicted and ground truth object-to-camera distance.

Our geometric optimization finds a consensus between the individual projection parameters, which results in a precise 2D-3D alignment for any initial focal length because we optimize the reprojection error during inference. However, the 3D pose of an object is ambiguous in the case of unknown intrinsics. Thus, a good initial focal length is a key factor in achieving high accuracy in terms of 3D translation, as can be seen from the

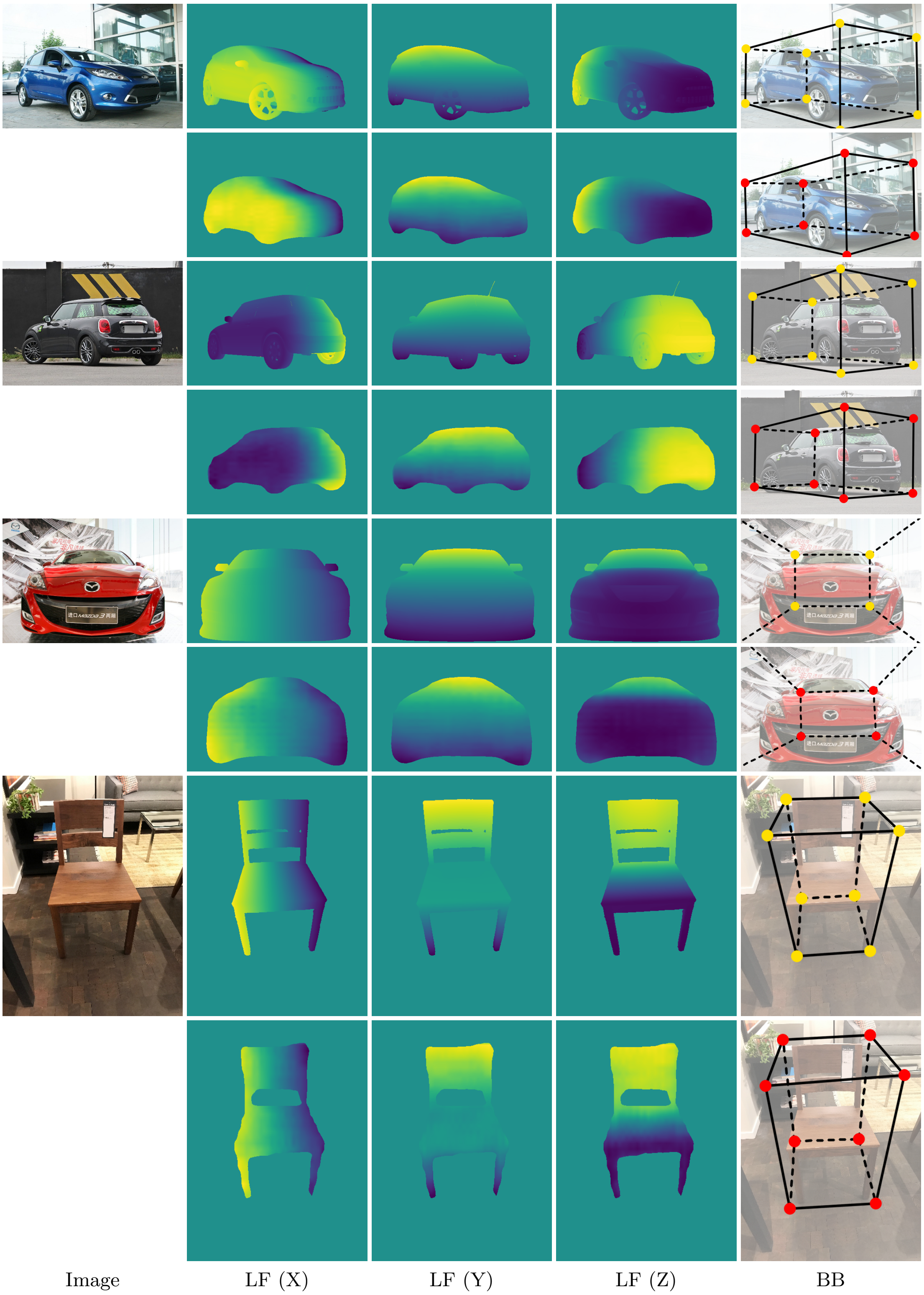


Figure 6.7: Qualitative examples of our predicted 2D-3D correspondences. For each object, we show two forms of 2D-3D correspondences: the location field (LF) and the projections of the object's 3D bounding box corners (BB). For each example image, the top row shows the ground truth and the bottom row shows our predictions.

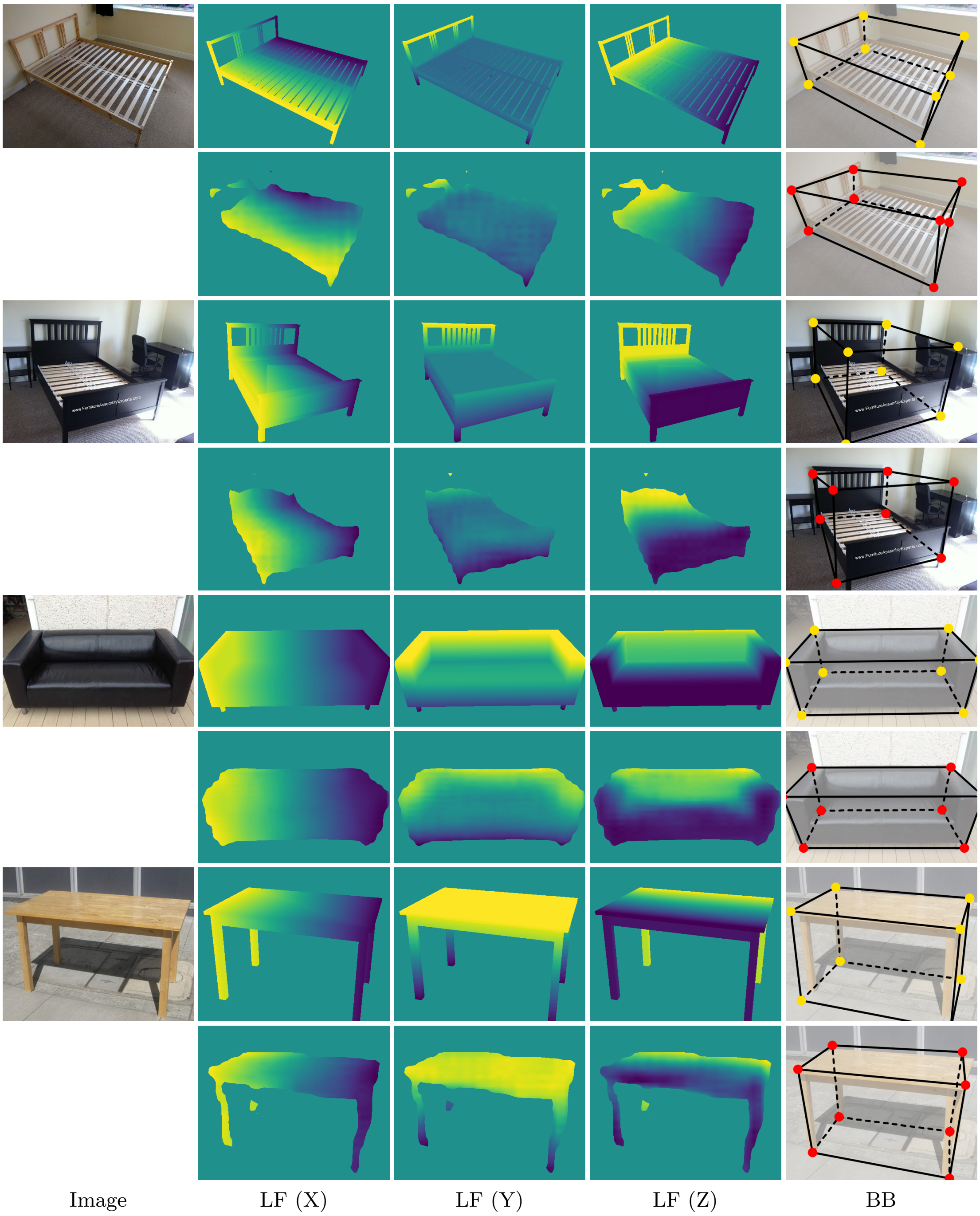
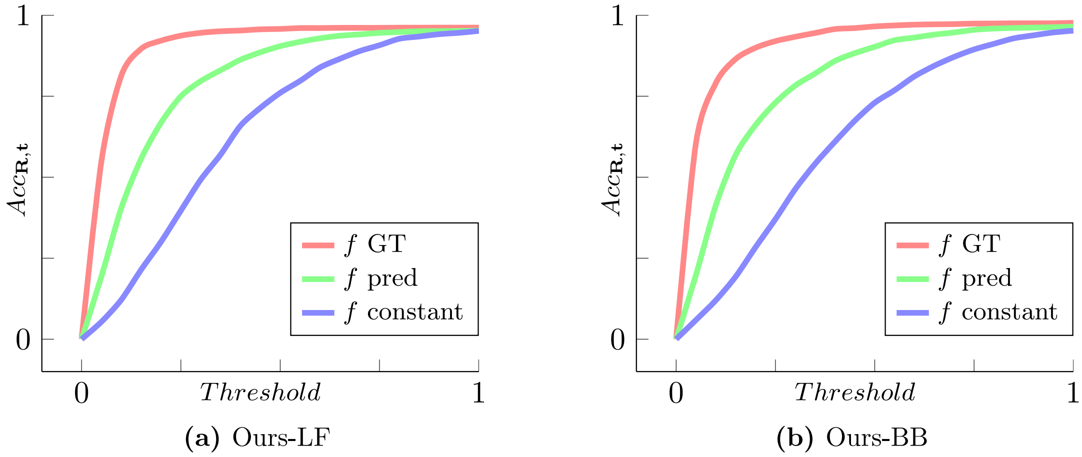


Figure 6.8: Qualitative examples of our predicted 2D-3D correspondences. For each object, we show two forms of 2D-3D correspondences: the location field (LF) and the projections of the object's 3D bounding box corners (BB). For each example image, the top row shows the ground truth and the bottom row shows our predictions.



(a) Ours-LF

(b) Ours-BB

Figure 6.9: Evaluation of 3D pose accuracy for different initial focal lengths. The results show that a good initial estimate of the focal length is a key factor for achieving high 3D pose accuracy.

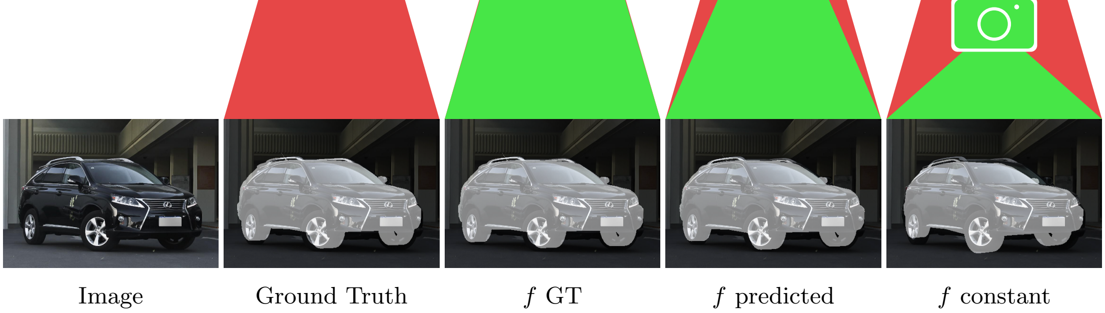


Figure 6.10: In the case of unknown intrinsics, the 3D pose of an object is ambiguous. Our approach finds a geometric consensus between all projection parameters, which results in a precise 2D-3D alignment for any initial focal length. However, a good initial focal length is required to compute an accurate 3D pose as illustrated by the visualization of the object-to-camera distance.

			Rotation		**Translation**	**Pose**	**Focal**	**Projection**	
Method	Dataset	Class	$MedErr_{\mathbf{R}}$ $\cdot 1$	$Acc_{\mathbf{R}_{\frac{\pi}{6}}}$	$MedErr_{\mathbf{t}}$ $\cdot 10^1$	$MedErr_{\mathbf{R},\mathbf{t}}$ $\cdot 10^1$	$MedErr_f$ $\cdot 10^1$	$MedErr_P$ $\cdot 10^2$	$Acc_{P_{0.1}}$
Ours-LF *initial*	Pix3D	*mean*	7.10	87.9%	1.89	1.32	1.73	3.98	84.7%
Ours-LF *refined*			**6.92**	**88.4%**	**1.85**	**1.30**	**1.72**	**3.85**	**85.5%**
Ours-BB *initial*	Pix3D	*mean*	7.04	90.1%	1.98	1.33	1.77	3.87	86.8%
Ours-BB *refined*			**6.89**	**90.8%**	**1.94**	**1.30**	**1.75**	**3.66**	**88.0%**

Table 6.3: Ablation study on joint 3D pose and focal length refinement. We compare our initial solution to the final solution obtained by our joint refinement. Jointly optimizing all parameters results in an improvement across all metrics.

visualization of the object-to-camera distance in Fig. 6.10. Our predicted focal length is significantly more accurate than the best possible constant focal length, i.e., the mode of the focal length distribution of the training dataset.

Finally, Table 6.3 presents quantitative results of our approach with and without joint 3D pose and focal length refinement. For this purpose, we compare our initial solution obtained by EPnP [155] with our predicted focal length to the final solution computed by our joint 3D pose and focal length refinement. Jointly optimizing all parameters results in an improvement across all metrics. In fact, the initial solution already outperforms the state of the art by a large margin. Our geometric optimization is fast and efficient. In our implementation, the geometric optimization with joint refinement (Stage 2) takes only 5 ms, while the CNN forward pass (Stage 1) takes 60 ms per image on average.

6.3.4 Failure Modes

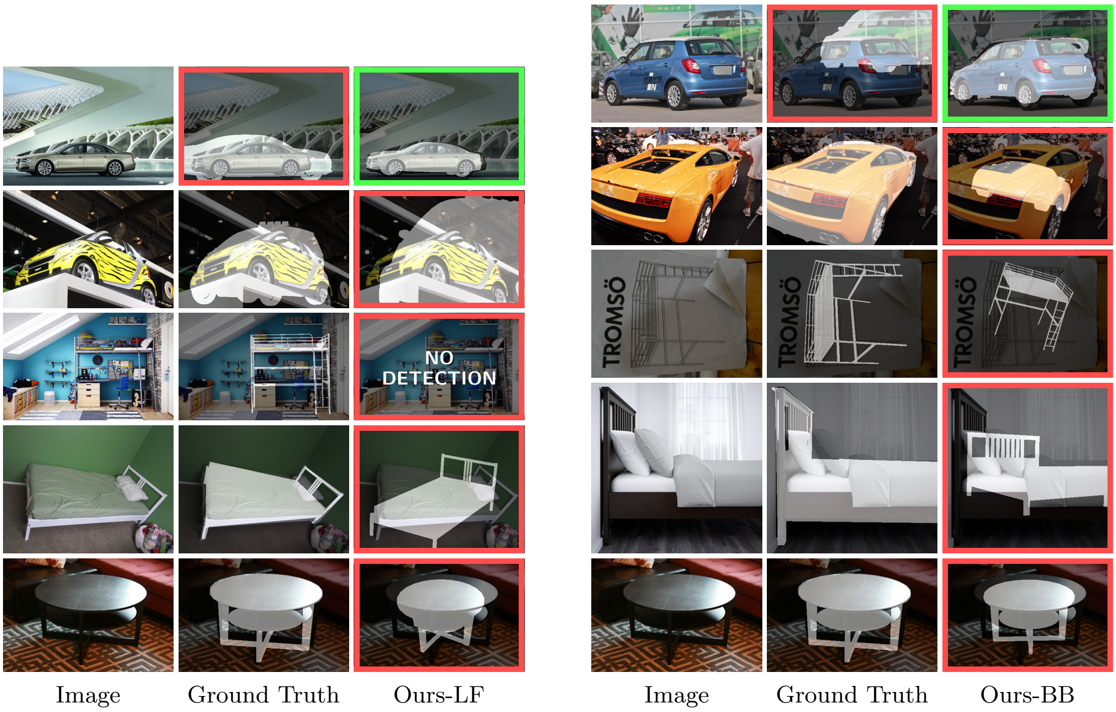
Fig. 6.11 shows failure modes of our approach using our two different methods for establishing 2D-3D correspondences (Ours-LF and Ours-BB). Most failure cases relate to strong truncations, heavy occlusions, object symmetries and ambiguities, or 3D poses which are far from the 3D poses seen during training. Interestingly, there is a large overlap between the failure cases of both methods, which indicates that the respective samples are significantly different from the samples seen during training.

Naturally, the annotations are not perfect and some occluded or truncated samples are marked as non-occluded and non-truncated, or the 3D pose annotation is incorrect. In some cases, our methods make a correct prediction but this prediction is considered wrong because of an erroneous ground truth 3D pose annotation, as shown in Fig. 6.11.

6.3.5 Discussion

So far, our results show that both presented 2D-3D correspondence estimation methods (Ours-LF and Ours-BB) achieve a similar level of overall accuracy across different metrics. However, each method has specific characteristics advantageous for different use cases.

For example, location fields implicitly handle truncations and occlusions because they estimate 3D points for visible object parts and resolve occlusions using the 2D mask.



(a) Failure cases of Ours-LF

(b) Failure cases of Ours-BB

Figure 6.11: Example failure cases for both presented 2D-3D correspondence estimation methods (Ours-LF and Ours-BB). Most failure cases relate to strong truncations, heavy occlusions, object symmetries and ambiguities, or 3D poses which are far from the 3D poses seen during training. However, in some cases, our methods make a correct prediction but the ground truth 3D pose annotation is corrupt, e.g., the annotator confused the back and the front of a car or mislabeled the location of the object in the image, as shown in the first rows of both (a) and (b). We highlight incorrect predictions and erroneous ground truth annotations with red frames and correct predictions with green frames.

Moreover, the predicted dense 2D-3D correspondences might also be useful for other tasks like dense depth estimation or 3D reconstruction. However, predicting location fields for objects with thin structures is challenging. Additionally, this method requires detailed 3D models for training.

In contrast, 3D bounding box corner correspondences only require accurate 3D bounding boxes for training. The overall design of this method is simpler and more lightweight, which makes it easier to implement and train. This is also reflected in our reported numbers, which show a slight advantage compared to location fields. Additionally, 3D bounding box corner correspondences always give a fixed number of sparse 2D-3D correspondences. This results in fast inference, which is beneficial for real-time applications, for example. However, while this method is well-suited for dealing with box-shaped objects like cars, other approaches might perform better on highly non-box-shaped objects.

6.4 Conclusion

Estimating the 3D poses of objects in the wild is an important but challenging task. In particular, predicting the 3D translation is difficult due to ambiguous appearances resulting from images captured with different but unknown focal lengths. For this purpose, we present the first joint 3D pose and focal length estimation approach that enforces a geometric consensus between 3D poses and the focal length. Our approach combines deep learning techniques and geometric algorithms to explicitly estimate and integrate the focal length into the 3D pose estimation. We evaluate our approach on three challenging real-world datasets (Pix3D, Comp, and Stanford) and significantly outperform the state of the art by up to 20%.

CHAPTER 7

3D Model Retrieval with Location Field Descriptors

Contents

In Chapter 5, we presented a novel approach to retrieve 3D models for objects of arbitrary categories from single RGB images captured in the wild. While this approach is computationally efficient at runtime because it utilizes a 3D pose prior to only match a single offline pre-computed descriptor for each 3D model, it has three shortcomings. First, it requires a large database of pre-computed descriptors which densely sample the 3D pose space. Depending on the sampling, this database can become very large and can easily exceed a thousand pre-computed descriptors for each 3D model, which is memory inefficient. Second, the descriptors trained in Chapter 5 have to be discriminative in terms of both 3D pose and 3D shape as a consequence of the architectural design. This is suboptimal for the task of 3D model retrieval. Finally, our previous approach requires given 2D bounding boxes for the objects in the images.

To overcome these limitations, we propose a novel approach for single image 3D model retrieval in the wild. In contrast to our previous method that directly maps RGB images and 3D model renderings to an embedding space (see Section 5.2.2), we now establish a common low-level representation in the form of location fields (see Section 6.2.1) from which we compute pose invariant 3D shape descriptors. Location fields encode dense correspondences between 2D object pixels and 3D model points and, thus, explicitly capture

3D shape and 3D pose information without appearance variations which are irrelevant for the task. This early fusion of RGB images and 3D model renderings results in six main advantages:

First, our learned 3D shape descriptors only have to be discriminative in terms of 3D shape because we can recover 3D poses from location fields using a geometric optimization (see Section 6.2.2). Second, we only store a single pose invariant 3D shape descriptor for each 3D model in our offline pre-computed database. In this way, our 3D model retrieval is as computationally efficient as before but the memory consumption of the database is significantly reduced. Third, predicting per-object location fields allows us to perform 3D model retrieval for multiple objects at different scales in a single image. Fourth, our bottleneck location field prediction acts as a regularizer during training. Fifth, major parts of our system benefit from training on a virtually infinite amount of synthetic data. Finally, our predicted location fields are visually interpretable and unblackbox the system. We evaluate our proposed approach on three challenging real-world datasets (Pix3D, Comp, and Stanford) with different object categories and significantly outperform the state of the art by up to 20% absolute in multiple 3D retrieval metrics.

7.1 Introduction

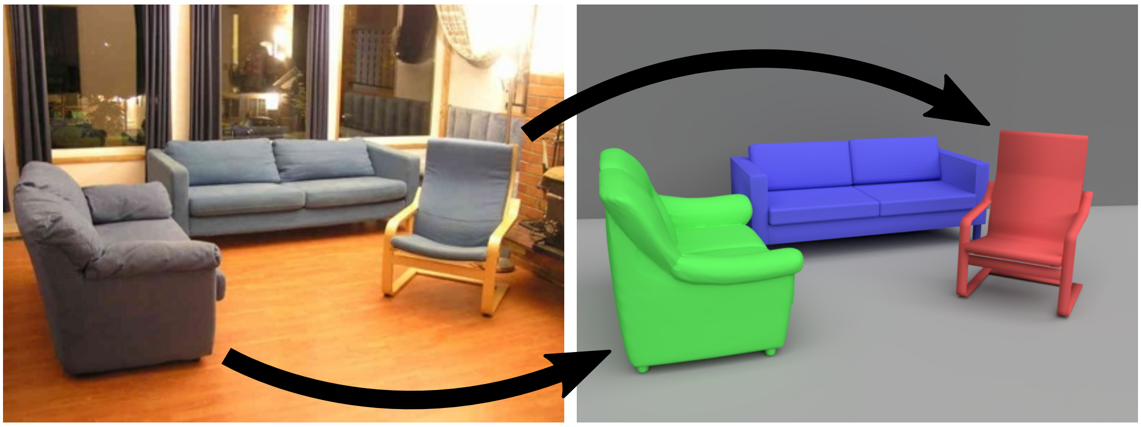


Figure 7.1: Given a single RGB image, we retrieve a 3D model with accurate geometry for each object in the image from a previously seen or unseen 3D model database.

3D model retrieval from a single RGB image, as shown in Fig. 7.1, is a challenging but important task with applications in augmented reality, robotics, 3D printing, 3D scene understanding, and 3D scene modeling. Compared to reconstruction [71, 278], retrieval provides 3D models designed by humans which are rich in detail. Due to the growing number of large-scale 3D model databases, like ShapeNet [33] or 3D Warehouse[1], efficient image-based retrieval approaches have become a fundamental requirement.

[1] `https://3dwarehouse.sketchup.com`

Recent works address the retrieval task by directly mapping RGB images and 3D models to a common embedding space [158, 266]. However, previous approaches have a number of limitations in practice. First, the learned mapping is highly prone to overfitting because training data in the form of RGB images with 3D model annotations is scarce [258, 288]. Second, systems purely trained on synthetic data do not generalize to real data due to the domain gap between RGB images and RGB renderings [71, 185]. Finally, the black box characteristic of these systems makes it hard to understand why the approaches fail in certain scenarios.

To overcome these limitations, we map RGB images and 3D models to a common low-level representation in the form of location fields from which we compute pose invariant 3D shape descriptors. Location fields [86, 288] are image-like representations that encode a 3D model point for each 2D object pixel (see Fig. 7.3). In particular, we predict location fields from RGB images using a CNN and render location fields from 3D models, as already described in Section 6.2.1. Then, instead of exhaustively comparing location fields from different 3D poses [123, 185], we compute pose invariant 3D shape descriptors in an embedding space optimized for 3D model retrieval from the location fields. Thus, we call our approach *Location Field Descriptors*.

Regarding 3D model retrieval, location fields have several advantages compared to other rendered representations: RGB renderings [71, 302] are subject to appearance variations which are irrelevant for the task caused by material, texture, and lighting. Textureless gray-scale renderings [153, 185] are still effected by the scene lighting. Silhouettes [34] are not effected by such appearance variations but discard valuable 3D shape information. Depth [85, 316] and normal renderings [287, 301] capture 3D geometry but lose the relation to the 3D pose in the object's canonical coordinate system. In contrast, location fields explicitly present 3D shape and 3D pose information as they establish correspondences between 2D object pixels and 3D model points on the object surface. Considering 3D shape, the dense 3D points provide a partial reconstruction of the object geometry. Considering 3D pose, the object's 3D rotation and 3D translation can be geometrically recovered from the 2D-3D correspondences using a PnP algorithm [24, 125].

The benefits of our approach are threefold: First, compared to directly mapping to an embedding space, our intermediate location field prediction serves as a regularizing bottleneck which reduces the risk of overfitting in the case of limited training data. Second, major parts of our system benefit from training on a virtually infinite amount of synthetic data due to the early fusion of RGB images and 3D models. Third, our predicted location fields are visually interpretable, unblackbox our system, and offer valuable insights in cases where our approach fails.

Finally, to demonstrate the benefits of our novel 3D model retrieval approach, we evaluate it on three challenging real-world datasets with different object categories: Pix3D [258] (*bed*, *chair*, *sofa*, *table*), Comp [288] (*car*), and Stanford [288] (*car*). We present quantitative as well as qualitative results and significantly outperform the state of the art. To summarize, our main contributions are:

- We present the first method that uses location fields for pose invariant 3D model retrieval. Our approach is accurate, scalable, and interpretable.
- We outperform the state of the art by up to 20% absolute in multiple 3D retrieval metrics given both previously seen and unseen 3D model databases.

7.2 Location Field Descriptors

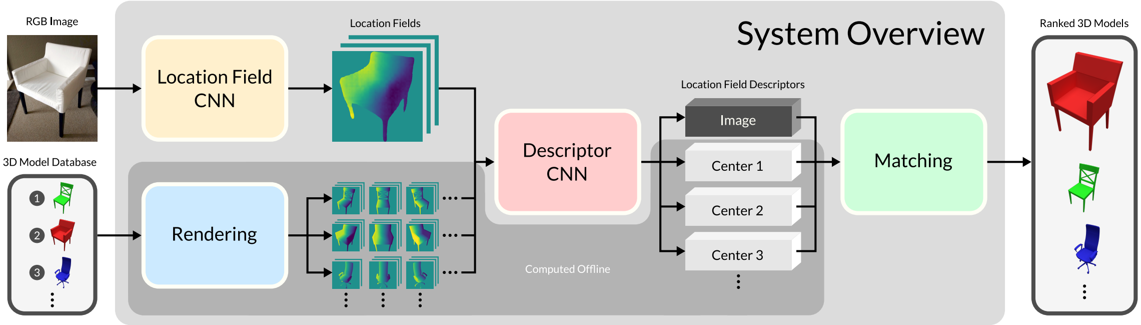


Figure 7.2: Overview of our approach. Given a single RGB image and a 3D model database, we use CNNs to predict a location field and a pose invariant location field descriptor for each object in the image. For each 3D model in the database, we learn a single center descriptor from multi-view location fields offline during training. Finally, we match location field descriptors predicted from the RGB image against offline computed center descriptors to retrieve a ranked list of the 3D models in the database.

Given a single RGB image and a 3D model database, we retrieve a 3D model for each object in the image, as shown in Fig. 7.2. For this purpose, we first generate location fields from both RGB images and 3D models (see Section 7.2.1). We then compute pose invariant 3D shape descriptors from the locations fields (see Section 7.2.2). Finally, we match descriptors predicted from the RGB image against descriptors computed from the 3D models to retrieve the highest-ranked 3D model from the database for each object.

7.2.1 Location Field Generation

The first step in our approach is to map RGB images and 3D models to a common low-level representation in the form of location fields. As illustrated in Fig. 7.3, a location field [288] is an image-like representation that encodes a 3D model point for each 2D object pixel. Compared to its reference RGB image, a location field has the same size and spatial resolution but the three channels encode XYZ 3D coordinates in the canonical object coordinate system instead of RGB colors. Locations fields explicitly present 3D shape and 3D pose information because they encode dense correspondences between 2D object pixels and 3D model points. From these 2D-3D correspondences, the 3D pose can be geometrically recovered using a PnP algorithm [24, 125], as already presented

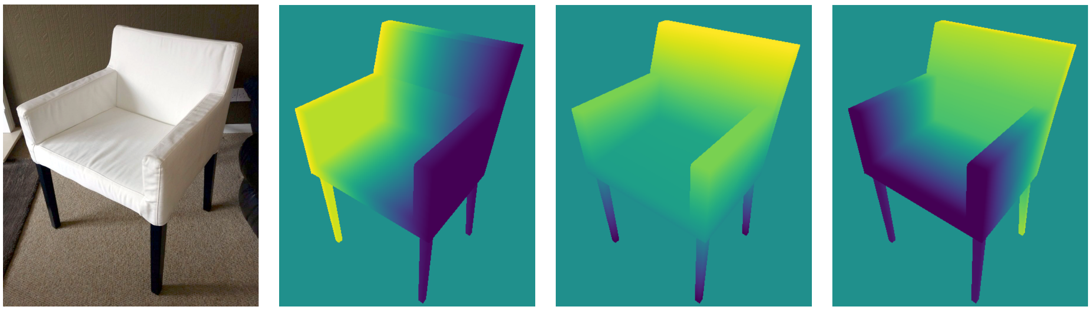


Image LF (X) LF (Y) LF (Z)

Figure 7.3: An image of an object and its location field (LF). Location fields encode a 3D model point in the canonical object coordinate system for each 2D object pixel. We show the three channels which correspond to the X, Y, and Z values of the 3D coordinates in separate images.

in Chapter 6. Additionally, location fields can also be interpreted as structured partial 3D point clouds.

Location fields can be directly rendered from 3D models. For this purpose, we rasterize 3D meshes using OpenGL and implement a custom fragment shader (see Section 2.1.1.3) which linearly interpolates per-vertex 3D coordinates along the triangles of a 3D mesh. Because the interpolated values describe 3D coordinates in the canonical object coordinate system, the relation to the object's inherent 3D pose is preserved.

In order to generate location fields from RGB images, we need to detect objects in 2D and predict a location field of each object. For this purpose, we introduce a Location Field CNN (see Fig. 7.2) which extends the generalized Faster/Mask R-CNN framework [96, 225]. This generic multi-task framework includes a 2D object detection pipeline to perform per-image and per-object computations. In this way, we address multiple different tasks using a single end-to-end trainable network.

In the context of the generalized Faster/Mask R-CNN framework, each output branch provides a task-specific subnetwork with different structure and functionality. We introduce a dedicated output branch for estimating location fields alongside the existing object detection branches, similar as presented in Section 6.2.1. Like the mask branch [96], the location field branch performs region-based per-object computations: For each detected object, an associated spatial region of interest in the feature maps is aligned to a fixed size feature representation with a low spatial but high channel resolution using linear interpolation, e.g., 14 × 14 × 256. These aligned features serve as a shared input to the classification, mask, and location field branches. Each branch is evaluated N times per image, where N is the number of detected objects.

Our location field branch uses a fully convolutional subnetwork to predict a tensor of 3D points at a resolution of 56 × 56 × 3 from the shared aligned features. We also modify the mask branch to predict 2D masks at the same spatial resolution and use the predicted masks to threshold the tensor of 3D points to get low-resolution location fields.

We experimentally found this approach to generate significantly higher accuracy location fields compared to directly regressing low-resolution location fields, which tends to predict over-smoothed 3D coordinates around the object silhouette. During training, we optimize our predicted location fields in a supervised manner using the Huber loss [118].

The resulting low-resolution location fields can be upscaled and padded to obtain high-resolution location fields with the same spatial resolution as the input image. This is especially helpful in cases where our approach fails. A visual overlay of the input image and the predicted location fields is intuitively interpretable and offers valuable insights to why the system fails.

However, we compute pose invariant 3D shape descriptors from the low-resolution location fields because upscaling does not provide additional information but merely increases the computational workload. Moreover, the predicted low-resolution location fields are tightly localized crops, which reduces the complexity of the descriptor computation.

7.2.2 3D Shape Descriptors

Instead of exhaustively comparing each location field predicted from an RGB image to multiple location fields of different 3D models rendered under different 3D poses [123, 185], we map location fields to pose invariant 3D shape descriptors in an embedding space. We refer to descriptors in this space as *Location Field Descriptors*.

For this purpose, we introduce a Descriptor CNN (see Fig. 7.2) which utilizes a dense connection pattern [115]. Similar to ResNets [99, 100], DenseNets [115] introduce skip-connections in the computational graph but concatenate feature maps instead of adding them. The dense connection pattern encourages feature reuse throughout the network and leads to compact but expressive models. This architecture is well-suited for computing descriptors from location fields because they already provide a high level of abstraction. Location fields are not effected by task-irrelevant appearance variations caused by color, material, texture, or lighting. The object is already segmented from the background and occlusions in the predicted location fields are resolved by the 2D mask used for thresholding the tensor of 3D points. In fact, even the raw 3D coordinates provide useful matching attributes, for example by aligning query and test 3D point clouds using ICP [16]. Thus, extensive feature reuse within the Descriptor CNN is rational.

In order to learn an embedding space which is optimized for 3D model retrieval, we need to address two requirements. First, the embedding space has to be discriminative in terms of 3D shape. Second, the computed descriptors have to be invariant to the 3D pose of the object in the location field. We jointly address both requirements by learning a representative center descriptor for each 3D model, as shown in Fig. 7.2. For this purpose, we train the Descriptor CNN to map location fields of a 3D model rendered under different 3D poses close to its corresponding center descriptor. At the same time, we make sure that all center descriptors are discriminatively distributed in the embedding space. Thus, during training, we penalize the distances between location field descriptors and center descriptors in a way that each location field descriptor and its corresponding center descriptor are pulled closer together, while all center descriptors are pulled further apart. This approach resembles a non-linear discriminant analysis [237], in which the intra-class variance is minimized, while the inter-class variance is maximized to train more discriminative embeddings.

In particular, we build on the ideas of Center loss [291] and Triplet-Center loss [101] to optimize our embedding space. The Center loss

$$L_{\mathrm{C}} = \sum_{i=1}^{N} D(\mathbf{f}_i, \mathbf{c}_{y_i}) \tag{7.1}$$

minimizes the distance $D(\mathbf{f}_i, \mathbf{c}_{y_i})$ between a location field descriptor $\mathbf{f}_i$ and its corresponding center descriptor $\mathbf{c}_{y_i}$. In this case, y_i is the index of the corresponding 3D model and N denotes the number of samples in the minibatch. In contrast, the Triplet-Center loss

$$L_{\mathrm{TC}} = \sum_{i=1}^{N} \max\Big(0\ , D(\mathbf{f}_i, \mathbf{c}_{y_i}) + m - \min_{j \neq y_i} D(\mathbf{f}_i, \mathbf{c}_j)\Big) \tag{7.2}$$

enforces the same distance $D(\mathbf{f}_i, \mathbf{c}_{y_i})$ to be smaller than the distance between a location field descriptor and its closest non-corresponding center descriptor $\min_{j \neq y_i} D(\mathbf{f}_i, \mathbf{c}_j)$ by at least the margin m. For the distance function $D(\cdot)$, we use $D(\mathbf{a}, \mathbf{b}) = \mathcal{L}(\|\mathbf{a} - \mathbf{b}\|_2)$ which is the Euclidean distance in the embedding space subject to the Huber loss [118] $\mathcal{L}(\cdot)$ in favour of the squared loss to be more robust to outliers.

As a consequence, the Center loss only minimizes intra-class variance, while the Triplet-Center loss aims at both minimizing intra-class variance and maximizing inter-class variance. In many cases, however, the Triplet-Center loss fails to achieve these goals. Instead, it learns degenerated clusterings because the optimization criterion does not guarantee the desired properties [137]. Thus, we employ a combination of Center loss and Triplet-Center loss in our Descriptor loss

$$L_{\mathrm{D}} = L_{\text{cross-entropy}} + \alpha L_{\mathrm{C}} + \beta L_{\mathrm{TC}} \tag{7.3}$$

to achieve both low intra-class variance and high inter-class variance [137, 293]. In practice, these losses are combined with a cross-entropy loss $L_{\text{cross-entropy}}$ computed on top of a classification output layer with softmax normalization to learn more discriminative embeddings than classification alone [101, 291]. The parameters α and β control the impact of the different loss terms.

We want to emphasize that the center descriptors are not fixed but learned during training. In fact, the center descriptors are trainable weights of the Descriptor CNN in our implementation. Also, even though we optimize a triplet criterion, we do not require location field triplets as training input. Only a single location field and its corresponding 3D model index y_i are needed. The center descriptors required for the Triplet loss are sampled within the Descriptor CNN. Additionally, hard triplet mining [244] is less important because we always sample the closest non-corresponding center descriptor and also employ Center and cross-entropy losses.

We jointly train the Descriptor CNN on predicted and rendered location fields. This is a major advantage compared to previous approaches that directly map to an embedding space because training data in the form of RGB images with 3D model annotations is limited (see Chapter 4). In contrast, we benefit from training on a virtually infinite amount of synthetic data. Additionally, the intermediate location field prediction serves as a regularizing bottleneck and reduces the risk of overfitting because regressing location fields is more difficult than computing embeddings.

Since there is a domain gap between predicted and rendered location fields, we perform Feature Mapping [218]. We use a residual block [99, 100] to map location field descriptors from the predicted to the rendered domain. Thus, the training input either consists of pairs of corresponding predicted and rendered location fields or single location fields, and a 3D model index y_i in both cases. In the case of pairs, we compute an additional Feature Mapping loss between corresponding feature-mapped predicted and rendered location field descriptor using the same distance function $D(\cdot)$ as defined above.

To perform retrieval from a previously unseen 3D model database, we generate center descriptors without retraining. For each unseen 3D model, we render 100 location fields under different 3D poses, compute their embeddings using the Descriptor CNN, and average these location field descriptors to obtain a new center descriptor. Alternatively, we can retrain the Descriptor CNN by incorporating the new 3D models as additional rendered location fields. In any case, the center descriptors are computed offline which results in fast inference with a small memory footprint.

During inference, we only need to process RGB images using our CNNs since the center descriptors have already been computed offline. For each RGB image, we evaluate the Location Field CNN once and the Descriptor CNN N times to compute location field descriptors, where N is the number of detected objects. We then match each computed location field descriptors against all center descriptors and generate a ranked list of 3D models based on the Euclidean distance between the descriptors, as shown in Fig. 7.2.

Finally, our entire system, i.e., the Location Field CNN and the Descriptor CNN, is end-to-end trainable. The system loss is a combination of our Location Field loss, our Descriptor loss, our Feature Mapping loss, and the detection losses of the generalized Faster/Mask R-CNN framework.

7.3 Evaluation

To demonstrate the benefits of our novel 3D model retrieval approach, we evaluate it on three challenging real-world datasets with different object categories: Pix3D [258] (*bed*, *chair*, *sofa*, *table*), Comp [288] (*car*), and Stanford [288] (*car*). Details on the evaluated datasets are presented in Chapter 4. In particular, we provide quantitative results for

3D model retrieval from seen and unseen databases in comparison to the state of the art in Section 7.3.2, present qualitative results of our approach in Section 7.3.3, perform an ablation study in Section 7.3.4, and investigate failure modes in Section 7.3.5. For our quantitative evaluation, we use the following well-established metrics:

Detection: We report the detection accuracy $Acc_{D_{0.5}}$ which gives the percentage of objects for which the intersection over union between the ground truth 2D bounding box and the predicted 2D bounding box is larger than 50% [303]. This metric is an upper bound for other Acc metrics since we do not make blind predictions.

Retrieval Accuracy: We evaluate the retrieval accuracies $Acc_{\textit{Top-1}}$ and $Acc_{\textit{Top-10}}$ which give the percentage of objects for which the ground truth 3D model equals the top ranked 3D model (*Top-1*) [85], or is in the top ten ranked 3D models (*Top-10*) [302]. These metrics can only be provided if the ground truth 3D model is in the retrieval database.

Hausdorff Distance: We compute a modified Hausdorff distance [7, 12]

$$d_{\mathrm{H}} = \frac{1}{|\mathcal{A}| + |\mathcal{B}|} \Big(\sum_{\mathbf{A} \in \mathcal{A}} \min_{\mathbf{B} \in \mathcal{B}} \|\mathbf{A} - \mathbf{B}\|_2 + \sum_{\mathbf{B} \in \mathcal{B}} \min_{\mathbf{A} \in \mathcal{A}} \|\mathbf{B} - \mathbf{A}\|_2 \Big) \tag{7.4}$$

between the ground truth 3D model $\mathcal{A}$ and the retrieved 3D model $\mathcal{B}$. For each 3D vertex $\mathbf{A} \in \mathcal{A}$ and $\mathbf{B} \in \mathcal{B}$, we calculate the Euclidean distance to the closest vertex from the other 3D model and compute the mean over both sets. Before computing d_{H}, we regularly resample each 3D model. For this purpose, we render orthographic location fields from six canonical 3D poses (front, left, right, back, top, bottom), sample 3D model points from these location fields, and concatenated all 3D points. For the orthographic projection, we use a resolution of 64×64. In this way, we sample between 5000 and 10000 3D points per 3D model depending on the geometry.

There are many possible strategies for resampling 3D meshes. However, as long as the sampling of 3D points is sufficiently dense, the differences in evaluation will be minimal. We report the mean modified Hausdorff distance for all detected objects (d_{HAU}). Since all 3D models are consistently aligned, the score is in the interval $[0, \sqrt{2}]$ (lower is better).

3D Intersection Over Union: We compute the 3D intersection over union between a voxelization of the ground truth 3D model and a voxelization of the retrieved 3D model [267]. For this purpose, we voxelize 3D models using `binvox` [196] with a resolution of $128 \times 128 \times 128$. We report the mean 3D IOU for all detected objects (d_{IOU}). The score is in the interval $[0, 1]$ (higher is better).

7.3.1 Implementation Details

For our Location Field CNN, we use a Feature Pyramid Network [163] on top of a ResNet-101 backbone [99, 100]. For our Descriptor CNN, we use a DenseNet-50 architecture [115] with 3 dense blocks and a growth rate of 24. For our implementation, we resize and pad RGB images to a spatial resolution of $512 \times 512 \times 3$ maintaining the aspect ratio. For the location fields, we employ a resolution of $56 \times 56 \times 3$. In this configuration, the Descriptor CNN maps low-resolution location fields to a 270-dimensional embedding space.

We initialize the convolutional backbone and the detection branches of the Location Field CNN with weights trained for instance segmentation [96] on COCO [165]. The location field branch and the Descriptor CNN are trained from scratch. We train our networks for 300 epochs using a batch size of 32. The initial learning rate of $1e^{-3}$ is decreased by a factor of 5 after 150 and 250 epochs.

We employ different forms of data augmentation. For RGB images, we use mirroring, jittering of location, scale, and rotation, and independent pixel augmentations like additive noise. For rendered location fields, we additionally use different forms of blurring to simulate predicted location fields. During training of the Descriptor CNN, we further leverage synthetic data in addition to real data and train on predicted and rendered location fields using a ratio of $1:3$.

To balance the individual terms in the system loss

$$L = L_{\text{det}} + L_{\text{cross-entropy}} + \alpha L_{\text{C}} + \beta L_{\text{TC}} + \gamma L_{\text{LF}} + \delta L_{\text{FM}} \ , \tag{7.5}$$

we assign unit weights to classification losses and non-unit weights to regression losses. Thus, we combine the unmodified Detection losses (L_{det}) of the generalized Faster/Mask R-CNN framework and the Descriptor CNN cross-entropy loss ($L_{\text{cross-entropy}}$) with the weighted Center (L_{C}), Triplet-Center (L_{TC}), Location Field (L_{LF}), and Feature Mapping (L_{FM}) losses. We experimentally set $\alpha = 0.01$, $\beta = 0.1$, $\gamma = 10$, $\delta = 0.01$, and use a margin of $m = 1$.

We consistently orient, scale, and translate all 3D models. In particular, we rotate all 3D models to have common front-facing, up, and starboard directions. Additionally, we scale and translate all 3D models to fit inside a unit cube centered at the coordinate origin $(0, 0, 0)$ while preserving the aspect ratio of the 3D dimensions. This 3D model alignment is not only important for training our approach but also for the evaluated metrics. For example, computing the modified Hausdorff distance and the 3D IOU between two 3D models is only meaningful if they are consistently oriented, scaled, and centered.

7.3.2 Comparison to the State of the Art

We are the first to present results for 3D model retrieval on Pix3D, Comp, and Stanford. For this purpose, we compare our approach to a baseline method [10] and a state-of-the-art method [85] which is our approach presented in Chapter 5. Since [10] and [85] assume that

				seen 3D models				**unseen 3D models**	
Method	Dataset	Category	$Acc_{D_{0.5}}$	$Acc_{\mathit{Top}\text{-}1}$	$Acc_{\mathit{Top}\text{-}10}$	d_{HAU}	d_{IOU}	d_{HAU}	d_{IOU}
[10]	Pix3D	bed	99.0%	19.4%	46.6%	0.0821	0.3397	0.0960	0.2487
[85]				35.1%	83.2%	0.0385	0.5598	0.0577	0.3013
Ours				**64.4%**	**89.0%**	**0.0152**	**0.8074**	**0.0448**	**0.3490**
[10]	Pix3D	chair	91.5%	17.3%	49.1%	0.0559	0.3027	0.0843	0.1334
[85]				41.3%	73.9%	0.0305	0.5469	0.0502	0.1965
Ours				**58.1%**	**81.8%**	**0.0170**	**0.7169**	**0.0375**	**0.2843**
[10]	Pix3D	sofa	96.9%	21.7%	52.2%	0.0503	0.3824	0.0590	0.3493
[85]				44.1%	89.8%	0.0197	0.7762	0.0294	0.6178
Ours				**67.0%**	**94.4%**	**0.0075**	**0.9028**	**0.0178**	**0.7472**
[10]	Pix3D	table	91.2%	12.0%	34.2%	0.1003	0.1715	0.1239	0.1047
[85]				33.9%	66.1%	0.0607	0.4500	0.0753	0.1730
Ours				**53.3%**	**80.1%**	**0.0288**	**0.6383**	**0.0482**	**0.2573**
[10]	Pix3D	*mean*	94.6%	17.6%	45.5%	0.0722	0.2991	0.0908	0.2090
[85]				38.6%	78.3%	0.0374	0.5832	0.0531	0.3222
Ours				**60.7%**	**86.3%**	**0.0171**	**0.7663**	**0.0370**	**0.4095**
[10]	Comp	car	99.9%	2.4%	18.2%	0.0207	0.7224	0.0271	0.6344
[85]				10.2%	36.9%	0.0158	0.7805	0.0194	0.7230
Ours				**20.5%**	**58.0%**	**0.0133**	**0.8142**	**0.0165**	**0.7707**
[10]	Stanford	car	99.6%	3.7%	20.1%	0.0198	0.7169	0.0242	0.6526
[85]				11.3%	42.2%	0.0153	0.7721	0.0183	0.7201
Ours				**29.5%**	**69.4%**	**0.0110**	**0.8352**	**0.0150**	**0.7744**

Table 7.1: Experimental results on the Pix3D, Comp, and Stanford datasets. We provide results for 3D model retrieval from both seen (in training dataset) and unseen (ShapeNet) 3D model databases given unseen test images. We significantly outperform the state of the art in all metrics and datasets. A detailed discussion of the reported numbers is presented in Section 7.3.2.

objects are already detected in 2D, we use the detections given by our approach for a fair comparison. The results are summarized in Table 7.1, where we significantly outperform the state of the art in all metrics and datasets.

First of all, we correctly detect 95% of all objects in the images on average ($Acc_{D_{0.5}}$), since object detection is tightly integrated into our approach. In fact, our Location Field CNN is initialized with weights trained for instance segmentation [96] on COCO [165] and all evaluated categories are present in COCO.

Next, we evaluate two different retrieval setups: First, we use all 3D models from the respective dataset as a 3D model database for retrieval (*seen 3D models*). In this case, we retrieve the correct 3D model ($Acc_{\mathit{Top}\text{-}1}$) for more than 60% of all test samples on average on Pix3D. This is a significant improvement of more than 20% absolute compared to the state of the art. Also, the retrieval accuracy quickly raises if we consider the top ten ranked 3D models ($Acc_{\mathit{Top}\text{-}10}$).

In contrast, the retrieval accuracy on Comp and Stanford is significantly lower for all evaluated methods. This is due to the considerably smaller variation in the overall 3D shape of *cars* compared to *chairs*, for example. Thus, many 3D models of *cars* have a similar appearance in a number of 3D poses and can only be discriminated by extremely fine-grained details like wheel rims or radiator grill structures. Such pixel-level information is usually discarded by CNNs.

However, by analyzing the 3D mesh similarity between the ground truth 3D model and the top retrieved 3D model (d_{HAU} and d_{IOU}), we observe that our approach consistently achieves high performance across all datasets and categories. To put the reported numbers in perspective, we compute the mean of the modified Hausdorff distance (0.1236) and the 3D IOU (0.0772) for all pairs of 3D models in the training datasets. These numbers represent the expected accuracy for picking a random 3D model. For both metrics, the 3D mesh similarity of our retrieved 3D model is around ten times better compared to picking a random 3D model. Additionally, we significantly outperform the state of the art by up to 50% relative considering d_{HAU}.

Second, we perform retrieval using previously unseen 3D models from ShapeNet [33] (*unseen 3D models*). If 3D models are present in the respective dataset and in ShapeNet, we exclude them for retrieval from ShapeNet to evaluated retrieval from an entirely unseen database. Since the correct 3D model is not in the database in this case, the achievable performance is limited. Thus, the reported numbers are slightly worse compared to retrieval from a previously seen 3D model database. Still, the performance is much better compared to picking a random 3D model. In fact, for some categories, e.g., Stanford *cars*, our approach retrieves more accurate 3D models from an unseen database than the state of the art from a database seen during training.

7.3.3 Qualitative Results

The quantitative performance of our approach is also reflected in our qualitative results. First, Fig. 7.4 shows examples of our predicted location fields. We upscale and pad the predicted location fields to match the input image resolution. The overall 3D shape is recovered well in the location fields but fine-grained details like the side mirrors of the car or thin structures like the frame ornaments of tables and beds are missed.

Next, Fig. 7.5 shows qualitative results for 3D model retrieval from ShapeNet. Considering the top ten ranked 3D models, we observe that the retrieved 3D models have a consistent and accurate overall 3D shape and geometry.

Finally, Figs. 7.6 and 7.7 present examples for 3D model retrieval from both seen and unseen databases. In addition, we show that location fields provide all relevant information to also compute the 3D pose of objects. For this purpose, we sample 2D-3D correspondences from the location fields and solve a PnP problem during inference. The projections onto the image show that both our retrieved 3D models and our computed 3D poses are highly accurate. Our approach naturally handles multiple objects in a single image. However, all evaluated datasets only provide annotations for a single object per image, as shown in Fig. 7.7.

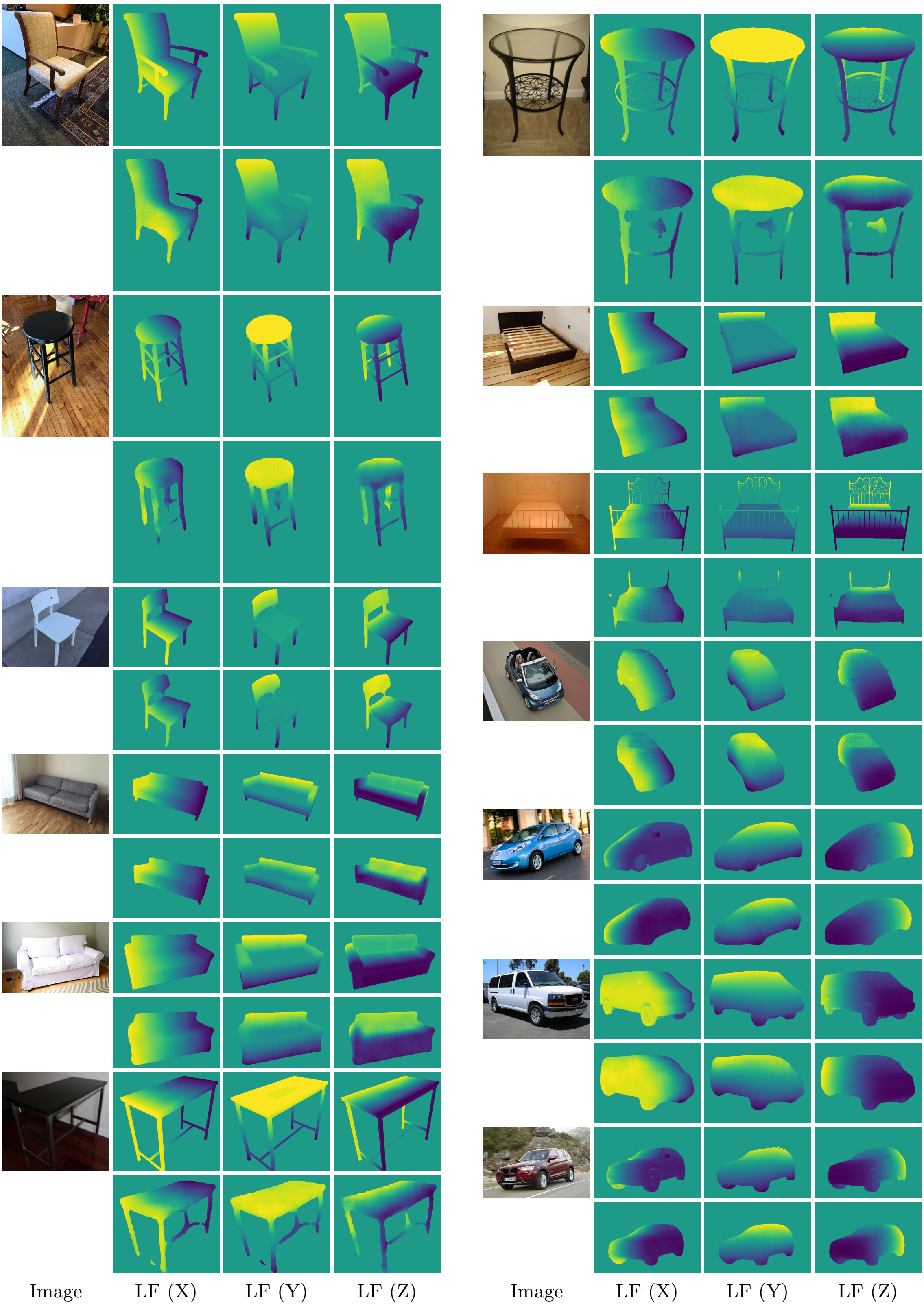


Figure 7.4: Qualitative examples of our predicted location fields. For each example image, the top row shows the ground truth and the bottom row shows our prediction. The overall 3D shape is recovered well but fine-grained details like the side mirrors of cars or thin structures like the frame ornaments of tables and beds are missed.

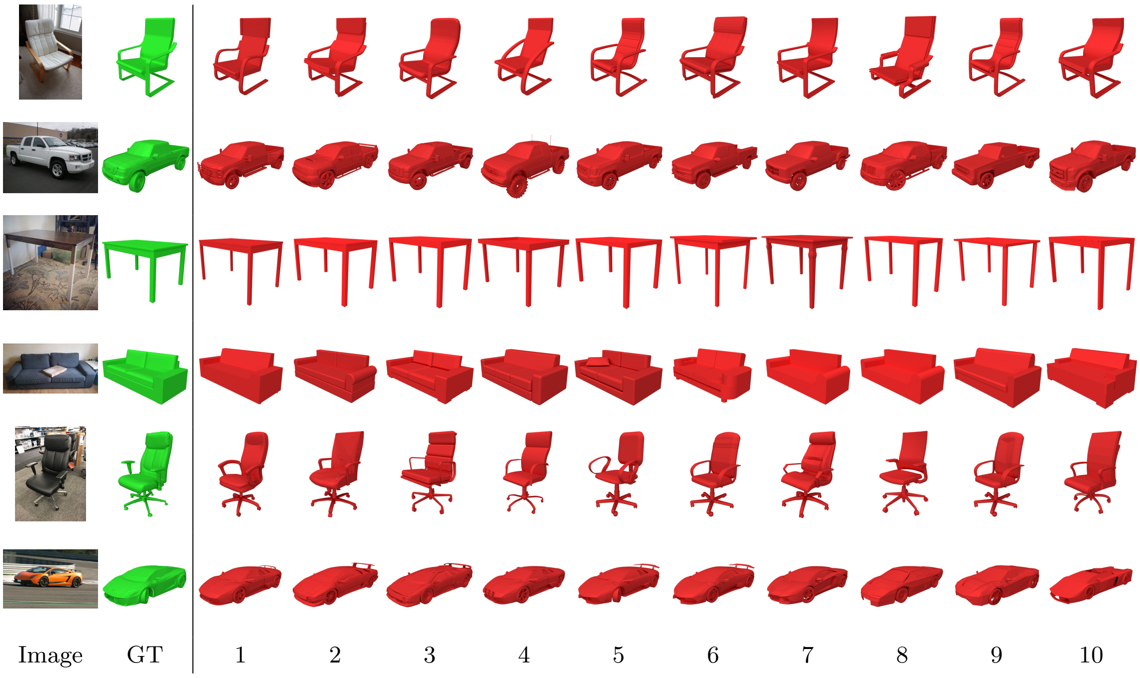


Figure 7.5: Qualitative results for 3D model retrieval from ShapeNet. From left to right, we show the input image, the ground truth 3D model, and the top ten ranked 3D models. The overall 3D shape of the retrieved 3D models is consistent and accurate.

7.3.4 Ablation Study

To understand which aspects of our approach are crucial for performance, we conduct an ablation study. For this purpose, we perform experiments on Pix3D which is the most challenging dataset because it provides multiple categories and has the largest variation in object scales and 3D poses. In particular, we train our approach using different setups and evaluate the mean performance across all categories. The results of this experiment are summarized in Table 7.2.

If we train our approach without synthetic data, i.e., train our Descriptor CNN purely on predicted location fields, the performance decreases significantly. Since training data is limited, we do not see location fields from many different 3D poses during training in this setup. As a consequence, the Descriptor CNN suffers from overfitting.

Next, if we predict location fields at half of our proposed resolution ($28 \times 28 \times 3$) the performance drops significantly. In this case, fine-grained structures, e.g., thin legs of a chair, cannot be recovered due to the limited spatial resolution.

Optimizing a pure cross-entropy loss without our proposed combination of Center loss [291] and Triplet-Center loss [101] results in a small performance decrease. This shows that our proposed Descriptor loss presented in Eq. (7.3) indeed learns more discriminative embeddings than classification alone since it explicitly minimizes intra-class variance and maximizes inter-class variance.

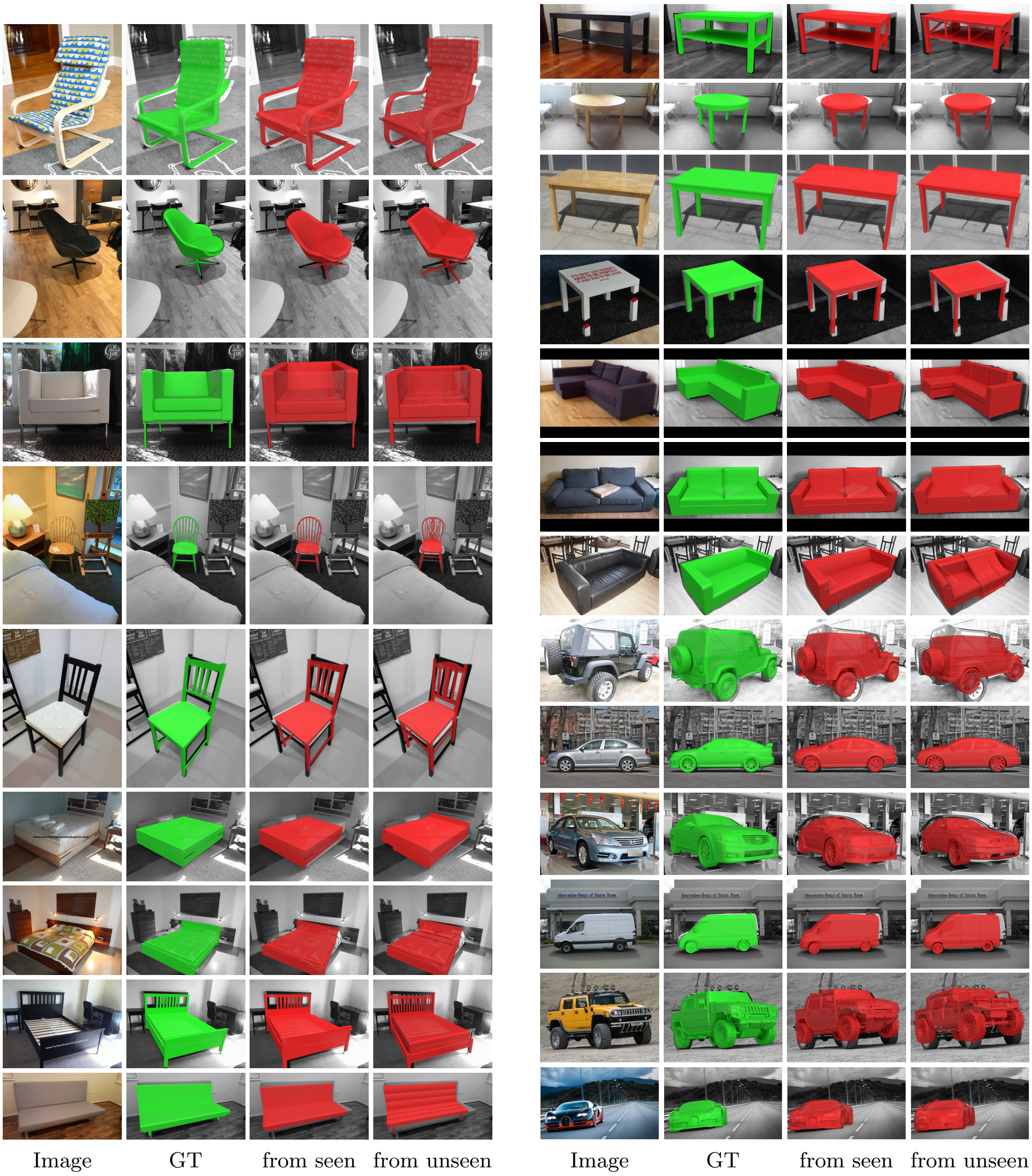


Figure 7.6: Qualitative results for 3D pose estimation and 3D model retrieval from both seen and unseen databases. We project the retrieved 3D model onto the image using a 3D pose computed from the predicted location field by solving a PnP problem. For the ground truth 3D model, we use the ground truth 3D pose. In fact, location fields provide all relevant information to jointly address both tasks.

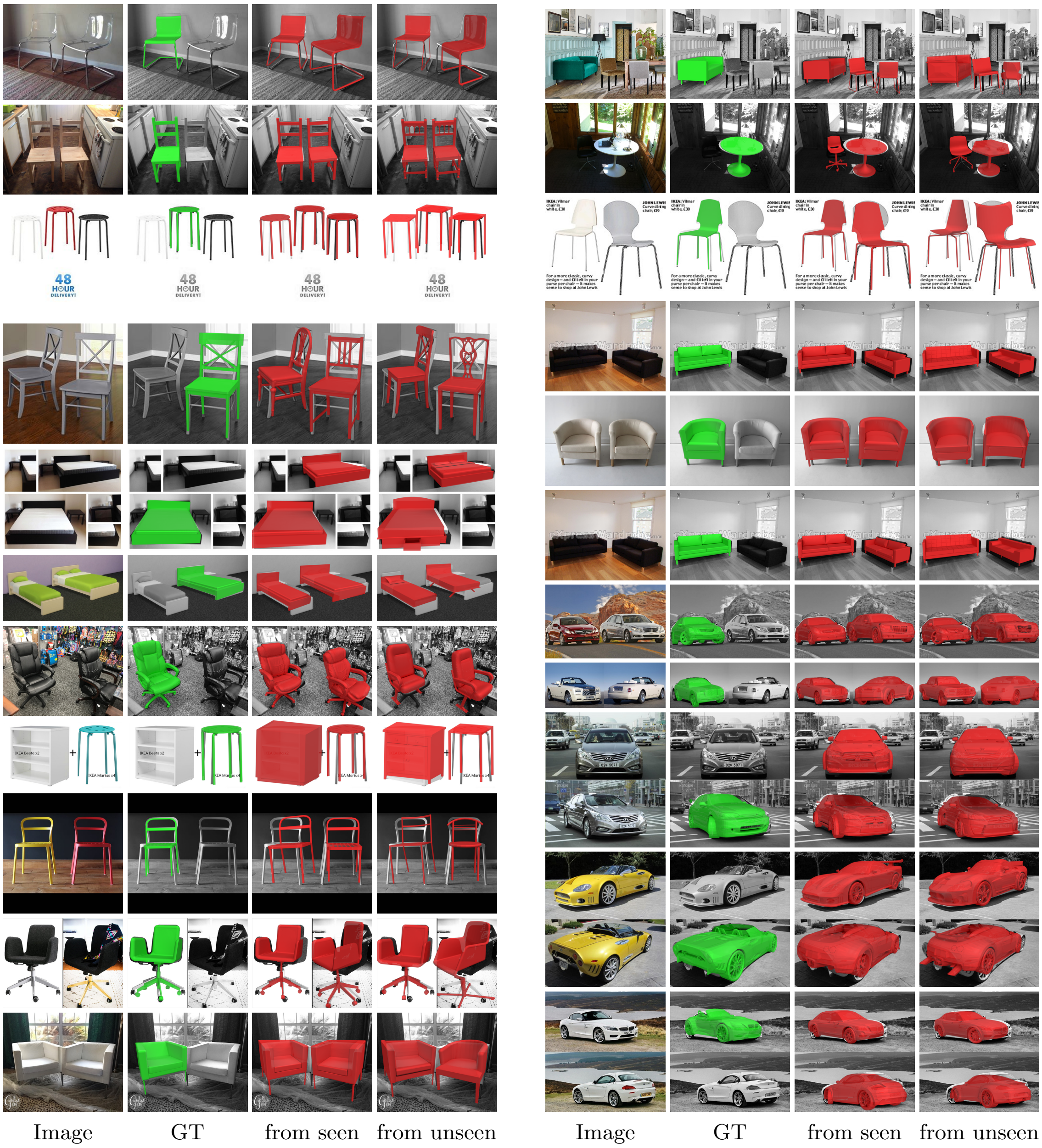


Figure 7.7: Qualitative results for 3D pose estimation and 3D model retrieval from both seen and unseen databases. We project the retrieved 3D model onto the image using a 3D pose computed from the predicted location field by solving a PnP problem. For the ground truth 3D model, we use the ground truth 3D pose. In fact, location fields provide all relevant information to jointly address both tasks. Our approach naturally handles multiple objects in a single image. However, all evaluated datasets only provide annotations for a single object per image.

Method	$Acc_{Top\text{-}1}$	d_{HAU}	d_{IOU}
Ours w/o synthetic data	55.0%	0.0219	0.7156
Ours half-res LFs	58.7%	0.0204	0.7370
Ours w/o (T)CL [101, 291]	59.9%	0.0175	0.7621
Ours w/o Mapping [218]	60.0%	0.0174	0.7630
Ours multi-view	**60.9%**	0.0173	**0.7686**
Ours	60.7%	**0.0171**	0.7663

Table 7.2: Ablation study of our method on the Pix3D dataset. Exploiting synthetic data in the form of rendered location fields during training and employing location fields with sufficient resolution to capture thin structures are the most important aspects for increasing performance.

Training without Feature Mapping [218] only slightly decreases performance. This is in part due to the fact that we also address the domain gap by aggressively augmenting and degenerating rendered location fields during training of our Descriptor CNN to simulate predicted location fields.

Finally, if we do not use our learned center descriptors but multi-view descriptors for matching, the performance almost remains the same. In this case, we match against 100 descriptors computed from location fields rendered under different 3D poses instead of a single center descriptor for each 3D model. This exhaustive comparison has a much higher computational complexity than our proposed approach and considerably increases the size of the retrieval database. In fact, using center descriptors is not only significantly faster but also achieves better performance considering d_{HAU}. This experiment confirms that our approach indeed learns pose invariant 3D shape descriptors.

7.3.5 Failure Modes

Fig. 7.8 shows failure modes of our approach. Most failure cases relate to incorrect location field predictions. For example, if the 3D pose of the object in the image is far from the 3D poses seen during training, or if multiple objects are detected as a single object in a complex occlusion scenario, we cannot predict an accurate location field. In other failure cases, we predict an accurate location field but retrieve a 3D model from a different category due to ambiguous geometries in different categories.

While the detection branch of the generalized Faster/Mask R-CNN framework predicts a category for each detected object in an RGB image, we do not use this information during 3D model retrieval because there is a category ambiguity for many objects. For example, it is unclear if a couch with sleeping functionality is a *sofa* or a *bed* (see Fig. 7.6, left column, last example). We want to retrieve 3D models with accurate geometry without category restrictions. However, the predicted category can also be used in our approach to narrow the retrieval down to 3D models from a single category and speed up the matching.

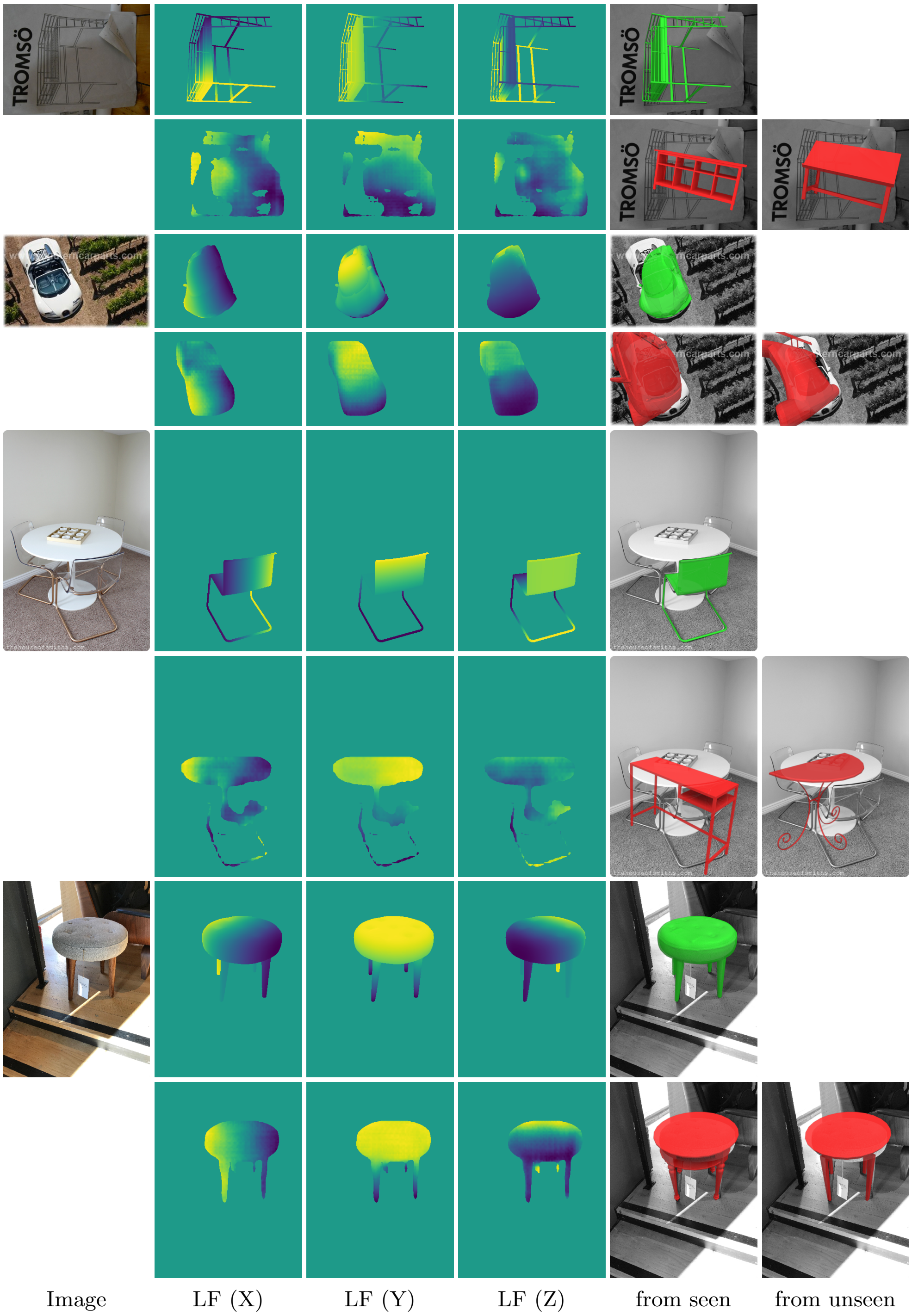


Figure 7.8: Failure cases of our approach. For each example image, the top row shows the ground truth and the bottom row shows our prediction. Most failure cases relate to incorrect location field predictions. If the 3D pose of the object in the image is far from the 3D poses seen during training (first and second example), we cannot predict an accurate location field. The location field prediction can also fail in complex occlusion scenarios if multiple objects are detected as a single object (third example). In other cases, we predict an accurate location field, but retrieve a 3D model from a different category due to ambiguous geometries in different categories, e.g., *table* instead of *chair* (last example).

7.4 Conclusion

Learning a common embedding of RGB images and 3D models for single image 3D model retrieval is difficult due to limited training data and the domain gap between real and synthetic data. For this purpose, we map RGB images and 3D models and to a common low-level representation in the form of location fields from which we compute pose invariant 3D shape descriptors. In this way, we bridge the domain gap and benefit from training on synthetic data. We evaluate our proposed approach on three challenging real-world datasets (Pix3D, Comp, and Stanford) and significantly outperform the state of the art by up to 20% absolute.

CHAPTER 8

3D Pose Refinement with Geometric Correspondence Fields

Contents

In the previous chapters, we presented methods for estimating 3D poses and retrieving 3D models for objects given single RGB images in the wild. The quality of these predictions can be visually assessed by reprojecting retrieved 3D models under estimated 3D poses back onto the input RGB image, as shown in the evaluation sections of the previous chapters. By investigating our predictions in this way, we observe that the overall quality of our reprojections is high. However, looking closely at the examples, we observe that our renderings do not precisely align to the objects in the RGB images because our predictions are not accurate enough in many cases.

To overcome this limitation, we present a novel 3D pose refinement approach based on differentiable rendering for objects of arbitrary categories in the wild. Our approach establishes a feedback loop between single RGB image inputs and reprojected renderings based on our predicted 3D pose and 3D model outputs. In contrast to previous refinement methods, we make two main contributions: First, instead of comparing real-world images and synthetic renderings in the RGB or mask space, we compare them in a feature space optimized for 3D pose refinement. Second, we introduce a differentiable renderer that learns to approximate the rasterization backward pass from data instead of relying on a hand-crafted algorithm. For this purpose, we predict deep cross-domain correspondences between RGB images and 3D model renderings in the form of what we call geometric correspondence fields. These correspondence fields serve as pixel-level gradients which are analytically propagated backward through the rendering pipeline to perform gradient descent directly on the 3D pose. In this way, we precisely align 3D models to objects in RGB images which results in significantly improved 3D pose estimates. We evaluate our approach on the challenging Pix3D dataset and achieve up to 55% relative improvement compared to state-of-the-art refinement methods in multiple metrics.

8.1 Introduction

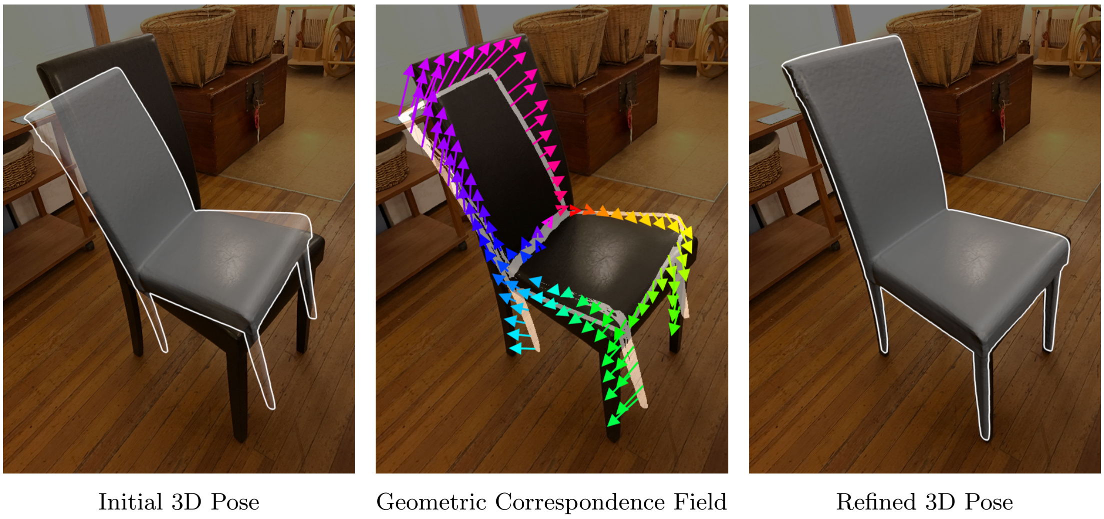


Figure 8.1: Given an initial 3D pose predicted by a feed-forward network, we predict deep cross-domain correspondences between RGB images and 3D model renderings in the form of a geometric correspondence field to refine the 3D pose using differentiable rendering.

Recently, there have been significant advances in single image 3D object pose estimation thanks to deep learning [85, 191, 279]. However, the accuracy achieved by today's feed-forward networks is not sufficient for many applications like augmented reality or robotics [86, 288]. As shown in Fig. 8.1, feed-forward networks robustly estimate the coarse high-level 3D rotation and 3D translation of objects in the wild given a single RGB image but fail to predict fine-grained 3D poses [306].

Thus, it is natural to introduce a refinement stage that improves the accuracy of predicted 3D poses by aligning 3D models to objects in RGB images. In this context, many refinement methods train a network that directly predicts 3D pose updates in a forward pass given the input RGB image and a 3D model rendering under the current 3D pose estimate [159, 217, 313]. In contrast, more recent methods explicitly optimize an objective function that is conditioned on the input RGB image and renderer inputs like the 3D pose of an object instead [134, 200]. These methods require differentiable rendering [171] to backpropagate gradients computed in the image space to render inputs but yield more accurate 3D pose updates because they exploit knowledge about the 3D scene geometry and the projection pipeline for the optimization.

However, existing approaches based on differentiable rendering have significant shortcomings because they rely on comparisons in the RGB or mask space. First, methods which compare images and renderings in the RGB space require photo-realistic 3D model renderings [171]. Generating such renderings is difficult because objects in the real world are subject to complex scene lighting, have unknown reflection properties, and are surrounded by cluttered backgrounds. Moreover, many 3D models only provide geometry but no textures or materials which makes photo-realistic rendering impossible [258]. Second, methods which compare images and renderings in the mask space need to predict accurate masks from RGB images [134, 200]. Generating such masks is difficult even using state-of-the-art approaches like Mask R-CNN [96]. Additionally, masks discard valuable shape information which makes 2D-3D alignment ambiguous. As a consequence, differentiable rendering methods based on comparisons in the RGB or mask space are not robust in practice. Finally, computing gradients for the non-differentiable rasterization operation in rendering is still an open research problem and existing approaches rely on hand-crafted approximations for this task [102, 134, 171].

To overcome these limitations, we compare RGB images and 3D model renderings in a feature space optimized for 3D pose refinement and learn to approximate the rasterization backward pass in differentiable rendering. In particular, we introduce a novel network architecture that jointly performs both tasks. Our network maps real-world images and synthetic renderings to a common feature space and predicts deep cross-domain correspondences in the form of *Geometric Correspondence Fields* (see Fig. 8.1). Geometric correspondence fields hold 2D displacement vectors between corresponding 2D object points in RGB images and 3D model renderings similar to optical flow [48]. These predicted 2D displacement vectors serve as pixel-level gradients that enable us to approximate the rasterization backward pass. By accumulating the pixel-level gradients for the projected geometry and propagating the resulting geometry-level gradients backward through the remaining differentiable parts of the rendering pipeline [134], we compute accurate gradients that minimize an ideal geometric reprojection loss for renderer inputs like the 3D pose, the 3D model, or the camera intrinsics (see Section 2.1.2.2).

Our approach has three main advantages: First, we can leverage depth, normal, and location field [288] renderings which provide 3D pose information more explicitly than RGB and mask renderings [159]. Second, we avoid task-irrelevant appearance variations in the RGB space and 3D pose ambiguities in the mask space [87]. Third, we learn to approximate the rasterization backward pass from data instead of relying on a hand-crafted algorithm [102, 134, 171].

To demonstrate the benefits of our novel 3D pose refinement approach, we evaluate it on the challenging Pix3D [258] dataset with different object categories. We present quantitative as well as qualitative results and significantly outperform state-of-the-art feed-forward 3D pose estimation and competing refinement methods in multiple metrics by up to 70% and 55% relative. Finally, we combine our refinement approach with our feed-forward 3D pose estimation (see Chapter 6) and 3D model retrieval (see Chapter 7) methods to predict fine-grained 3D poses for objects in the wild without providing initial 3D poses or ground truth 3D models at runtime given only a single RGB image. To summarize, our main contributions are:

- We introduce the first refinement method based on differentiable rendering that does not compare RGB images and 3D model renderings in the RGB or mask space but in a feature space optimized for the task at hand.
- We present a novel differentiable renderer that learns to approximate the rasterization backward pass from data instead of relying on a hand-crafted algorithm.

8.2 Learned 3D Pose Refinement

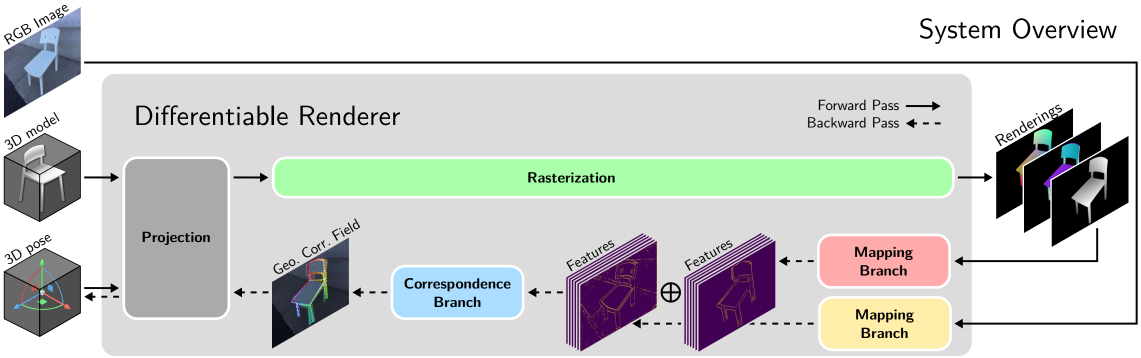


Figure 8.2: Overview of our system. In the forward pass (⟶), we generate 3D model renderings under the current 3D pose. In the backward pass (⇠ - -), we map the RGB image and our renderings to a common feature space and predict a geometric correspondence field that enables us to approximate the rasterization backward pass and compute gradients for the 3D pose that minimize an ideal geometric reprojection loss.

Given a single RGB image, a 3D model, and an initial 3D pose, we compute iterative updates to refine the 3D pose, as shown in Fig. 8.2. For this purpose, we first introduce the objective function that we optimize at runtime (see Section 8.2.1). Then we explain how we compare the input RGB image to renderings under the current 3D pose in a feature space optimized for refinement (see Section 8.2.2), predict pixel-level gradients that minimize an ideal geometric reprojection loss in the form of geometric correspondence fields (see Section 8.2.3), and propagate gradients backward through the rendering pipeline to perform gradient descent directly on the 3D pose (see Section 8.2.4).

8.2.1 Runtime Object Function

Our approach to refine the 3D pose of an object is based on the numeric optimization of an objective function at runtime. In particular, we seek to minimize an ideal geometric reprojection loss

$$e(\mathcal{P}) = \frac{1}{2} \sum_{i} \|\mathrm{proj}(\mathbf{M}_i, \mathcal{P}_{\mathrm{gt}}) - \mathrm{proj}(\mathbf{M}_i, \mathcal{P})\|_2^2 \tag{8.1}$$

for all provided 3D model vertices $\mathbf{M}_i$. In this case, $\mathrm{proj}(\cdot)$ is an abbreviated notation of the projection function $\mathrm{proj}(\mathbf{M}_i, \mathbf{K}, \mathbf{R}, \mathbf{t})$ (see Section 2.1.2.1) that assumes $\mathbf{K}$ is constant and denotes the 3D pose parameters, i.e., $\mathbf{R}$ and $\mathbf{t}$, as $\mathcal{P}$ for simplicity. Hence, $\mathcal{P}_{\mathrm{gt}}$ represents the ground truth 3D pose.

It is clear that $\arg\min e(\mathcal{P}) = \mathcal{P}_{\mathrm{gt}}$. Thus, our goal is to efficiently minimize $e(\mathcal{P})$ using a gradient-based optimization starting from an initial 3D pose estimate. To compute gradients for the 3D pose parameters, we calculate the Jacobian of $e(\mathcal{P})$ with respect to $\mathcal{P}$ and apply the chain rule, which yields the expression

$$\left(\frac{\partial e(\mathcal{P})}{\partial \mathcal{P}}\right)(\mathcal{P}_{\mathrm{curr}}) = \sum_{i} \left[\frac{\partial \mathrm{proj}(\mathbf{M}_i, \mathcal{P})}{\partial \mathcal{P}}\right]^T \Big(\mathrm{proj}(\mathbf{M}_i, \mathcal{P}_{\mathrm{gt}}) - \mathrm{proj}(\mathbf{M}_i, \mathcal{P}_{\mathrm{curr}})\Big) , \tag{8.2}$$

where $\mathcal{P}_{\mathrm{curr}}$ is the current 3D pose estimate and the point where the Jacobian is evaluated. In this case, the term $\left[\frac{\partial \mathrm{proj}(\mathbf{M}_i, \mathcal{P})}{\partial \mathcal{P}}\right]^T$ can be computed analytically because it is simply a sequence of differentiable operations. However, the term $(\mathrm{proj}(\mathbf{M}_i, \mathcal{P}_{\mathrm{gt}}) - \mathrm{proj}(\mathbf{M}_i, \mathcal{P}_{\mathrm{curr}}))$ cannot be computed analytically because the ground truth projections $\mathrm{proj}(\mathbf{M}_i, \mathcal{P}_{\mathrm{gt}})$ are unknown at runtime and can only be observed indirectly via the input RGB image.

However, for visible vertices, this term can be computed from a geometric correspondence field (see Section 8.2.4). Thus, we introduce a novel network architecture that learns to predict geometric correspondence fields given an RGB image and 3D model renderings under the current 3D pose estimate in the following sections. Moreover, we embed this network in a differentiable rendering framework to approximate the rasterization backward pass and to compute gradients for renderer inputs like the 3D pose of an object in an end-to-end manner, as shown in Fig. 8.2.

8.2.2 Refinement Feature Space

The first step in our refinement approach is to render the provided 3D model under the current 3D pose using the forward pass of our differentiable renderer (see Fig. 8.2). In particular, we generate depth, normal, and location field [288] renderings. These representations provide 3D pose and 3D shape information more explicitly than RGB or mask renderings which makes them particularly useful for 3D pose refinement [87]. By concatenating the different renderings along the channel dimension, we leverage complementary information from different representations in the backward pass rather than relying on a single manually selected type of rendering [171].

Next, we begin the backward pass of our differentiable renderer by mapping the input RGB image and our multi-representation renderings to a common feature space. For this task, we use two different network branches that bridge the gap between the real-world image domain and the rendered image domain, as shown in Fig. 8.2. Our mapping branches use a custom architecture based on task-specific design choices:

First, we want to predict local cross-domain correspondences under the assumption that the initial 3D pose is close to the ground truth 3D pose. Thus, we do not require features with global context but features with local discriminability. For this reason, we use small fully convolutional networks which are fast, memory-efficient, and learn low-level features that generalize well across different objects. Because the low-level structures appearing across different objects are similar, we do not require a different network for each object [217] but address objects of all categories with a single class-agnostic network for each domain.

Second, we want to predict correspondences with maximum spatial accuracy. Thus, we do not use pooling or downsampling but maintain the spatial resolution throughout the network. In this configuration, consecutive convolutions provide sufficient receptive field to learn advanced shape features which are superior to simple edge intensities [136], while higher layers benefit from full spatial parameter sharing during training which increases generalization. As a consequence, the effective minibatch size during training is higher than the number of images per minibatch because only a subset of all image pixels contributes to each computed feature. In addition, the resulting high spatial resolution feature space provides an optimal foundation for computing spatially accurate correspondences.

For the implementation of our mapping branches, we use fully convolutional networks consisting of an initial 7×7 Conv-BN-ReLU block, followed by three residual blocks [99, 100], and a 1×1 Conv-BN-ReLU block for dimensionality reduction. In this way, we map both RGB images and multi-representation renderings to $W \times H \times 64$ feature maps, where W and H are the spatial dimensions of the respective input image. This architecture enforces two properties which are crucial for 3D pose refinement in the resulting feature space: Local discriminability and high spatial resolution.

8.2.3 Geometric Correspondence Fields

After mapping RGB images and 3D model renderings to a common feature space, we compare their feature maps and predict cross-domain correspondences. For this purpose, we concatenate their feature maps and use another fully convolutional network branch to predict correspondences, as shown in Fig. 8.2.

In particular, we regress per-pixel correspondence vectors in the form of a geometric correspondence field (see Fig. 8.1). Geometric correspondence fields hold 2D displacement vectors between corresponding 2D object points in RGB images and 3D model renderings similar to optical flow [48]. These displacement vectors represent the projected relative 2D motion of individual 2D object points that is required to minimize the reprojection error and refine the 3D pose. A geometric correspondence field has the same spatial resolution as the respective input RGB image and two channels, i.e., $W \times H \times 2$.

If the ground truth 3D model and 3D pose are known, we can render the ground truth geometric correspondence field for an arbitrary 3D pose. For this task, we first compute the 2D displacement

$$\nabla \mathbf{m}_i = \mathrm{proj}(\mathbf{M}_i, \mathcal{P}_{\mathrm{gt}}) - \mathrm{proj}(\mathbf{M}_i, \mathcal{P}_{\mathrm{curr}}) \tag{8.3}$$

between the projection under the ground truth 3D pose $\mathcal{P}_{\mathrm{gt}}$ and the current 3D pose $\mathcal{P}_{\mathrm{curr}}$ for each 3D model vertex $\mathbf{M}_i$. We then generate a ground truth geometric correspondence field $G(\mathcal{P}_{\mathrm{curr}}, \mathcal{P}_{\mathrm{gt}})$ by rendering the 3D model using a shader that interpolates the per-vertex 2D displacements $\nabla \mathbf{m}_i$ across the projected triangle surfaces using barycentric coordinates (see Section 2.1.1.3).

In our scenario, predicting correspondences using a network has two advantages compared to traditional correspondence matching [95]. First, predicting correspondences with convolutional kernels is significantly faster than exhaustive feature matching during both training and testing [312]. This is especially important in the case of dense correspondences. Second, training explicit correspondences can easily result in degenerated feature spaces and requires tedious regularization and hard negative sample mining [38].

However, in many cases local correspondence prediction is ambiguous. For example, many objects are untextured and have homogeneous surfaces, e.g., the backrest and the seating surface of the chair in Fig. 8.1, which cause unreliable correspondence predictions. To address this problem, we additionally employ a geometric attention module which restricts the correspondence prediction to visible object regions with significant geometric discontinuities, as outlined in white underneath the 2D displacement vectors in Fig. 8.1. We identify these regions by finding local variations in our depth, normal, and location field renderings.

For this task, we slide a small kernel across the renderings and detect rendering-specific intensity changes larger than a certain threshold within a local 5×5 window to construct a geometric attention mask w^a. In particular, we compare the window center to all neighboring window locations to compute the geometric attention weight

$$w^a_{x,y} = \max_{u,v \in W} \left(\delta^R \Big(R(\mathcal{P}_{\mathrm{curr}})_{x,y}, R(\mathcal{P}_{\mathrm{curr}})_{x-u,y-v} \Big) \right) > t^R \tag{8.4}$$

for each pixel location. In this case, $R(\mathcal{P}_{\mathrm{curr}})$ is a concatenation of depth, normal, and location field renderings under the current 3D pose $\mathcal{P}_{\mathrm{curr}}$, (x, y) is a pixel location, and (u, v) are pixel offsets within the window W. The comparison function $\delta^R(\cdot)$ and the threshold t^R are different for each type of rendering. For depth renderings, we compute the absolute difference between normalized depth values and use a threshold of 0.1. For normal renderings, we compute the angle between normals and use a threshold of 15°. For location field renderings, we compute the Euclidean distance between 3D points and use a threshold of 0.1. If any of these thresholds applies, the corresponding pixel (x, y) in our geometric attention mask w^a becomes active. Because we already generated these renderings before, our geometric attention mechanism requires almost no additional computations and is available during both training and testing.

For the implementation of our correspondence branch, we use three consecutive 7 × 7 Conv-BN-ReLU blocks followed by a final 7 × 7 convolution which reduces the channel dimensionality to two. For this network, a large receptive field is crucial to cover correspondences with high spatial displacement.

During training of our system, we optimize the learnable part of our differentiable renderer, i.e., a joint network $f(\cdot)$ consisting of our two mapping branches and our correspondence branch with parameters θ (see Fig. 8.2). Formally, we minimize the error between predicted $f(\cdot)$ and ground truth $G(\cdot)$ geometric correspondence fields as

$$\min_{\theta} \sum_{x,y} w^a_{x,y} \| f(I, R(\mathcal{P}_{\text{curr}}); \theta)_{x,y} - G(\mathcal{P}_{\text{curr}}, \mathcal{P}_{\text{gt}})_{x,y} \|_2^2 \tag{8.5}$$

In this non-linear least-squares optimization problem, w^a is a geometric attention mask, I is an RGB image, $R(\mathcal{P}_{\text{curr}})$ is a concatenation of depth, normal, and location field renderings generated under a random 3D pose $\mathcal{P}_{\text{curr}}$ produced by perturbing the ground truth 3D pose $\mathcal{P}_{\text{gt}}$, $G(\mathcal{P}_{\text{curr}}, \mathcal{P}_{\text{gt}})$ is the ground truth geometric correspondence field, and (x, y) is a pixel location.

In particular, we first generate a random 3D pose $\mathcal{P}_{\text{curr}}$ around the ground truth 3D pose $\mathcal{P}_{\text{gt}}$ for each training sample in each iteration. For this purpose, we sample 3D pose perturbations from normal distributions and apply them to $\mathcal{P}_{\text{gt}}$ to generate $\mathcal{P}_{\text{curr}}$. For 3D rotations, we use absolute perturbations with $\sigma = 5°$. For 3D translations, we use relative perturbations with $\sigma = 0.1$. We then render the ground truth geometric correspondence field $G(\mathcal{P}_{\text{curr}}, \mathcal{P}_{\text{gt}})$ between the perturbed 3D pose $\mathcal{P}_{\text{curr}}$ and the ground truth 3D pose $\mathcal{P}_{\text{gt}}$ as described above, generate concatenated depth, normal, and location field renderings $R(\mathcal{P}_{\text{curr}})$ under the perturbed 3D pose $\mathcal{P}_{\text{curr}}$, and compute the geometric attention mask w^a. Finally, we predict a geometric correspondence field using our network $f(I, R(\mathcal{P}_{\text{curr}}); \theta)$ given the RGB image I and the renderings $R(\mathcal{P}_{\text{curr}})$, and optimize for the network parameters θ.

In this way, we train a network that performs three tasks: First, it maps RGB images and multi-representation 3D model renderings to a common feature space. Second, it compares features in this space. Third, it predicts geometric correspondence fields which serve as pixel-level gradients that enable us to approximate the rasterization backward pass of our differentiable renderer.

8.2.4 Learned Differentiable Rendering

In the classic rendering pipeline, the only non-differentiable operation is the rasterization [171] that determines which pixels of a rendering have to be filled, solves the visibility

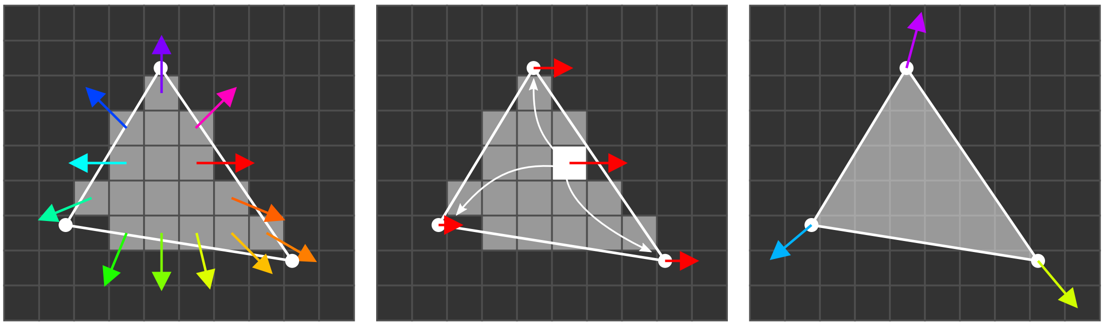


Figure 8.3: To approximate the rasterization backward pass, we predict a geometric correspondence field (*left*), disperse the predicted 2D displacement of each pixel among the vertices of its corresponding triangle (*middle*), and normalize the contributions of all pixels. In this way, we obtain motion gradients for projected 3D model vertices (*right*).

of projected triangles, and fills the respective pixels according to a shading computation (see Section 2.1.1.3). This operation raises one main challenge: There is no inherent relationship between the vertex locations of a projected triangle and the attribute used to fill the pixels covered by this triangle. The rasterization backward pass must convert pixel intensity gradients to vertex motion gradients [134]. However, this conversion has no closed solution and can only be approximated.

We solve this problem using geometric correspondence fields. Instead of actually differentiating a loss in the image space which yields per-pixel *intensity* gradients, we use a network to predict per-pixel *motion* gradients in the form of a geometric correspondence field, as shown in Fig. 8.2. To obtain per-vertex motion gradients, we simply accumulate all per-pixel motion gradients for the projected triangle vertices of the 3D model, as illustrated in Fig. 8.3. Formally, we compute the motion gradient of a projected 3D model vertex $\mathbf{m}_i$ as

$$\nabla\mathbf{m}_i = \frac{1}{\sum\limits_{x,y} w^a_{x,y} w^i_{x,y}} \sum_{x,y} w^a_{x,y} w^i_{x,y}\, f(I, R(\mathcal{P}_{\text{curr}}); \theta)_{x,y} \ . \qquad (8.6)$$
$$\forall x, y : \mathbf{m}_i \in \triangle_{\text{IndexMap}_{x,y}}$$

In this case, $f(I, R(\mathcal{P}_{\text{curr}}); \theta)$ is a geometric correspondence field predicted by our network $f(\cdot)$ with frozen parameters θ given an RGB image I and concatenated 3D model renderings $R(\mathcal{P}_{\text{curr}})$ under the current 3D pose estimate $\mathcal{P}_{\text{curr}}$, w^a is a geometric attention mask, w^i is a barycentric weight for $\mathbf{m}_i$, and (x, y) is a pixel position. We accumulate the 2D displacement vectors for all positions (x, y) for which $\mathbf{m}_i$ is a part of the triangle $\triangle_{\text{IndexMap}_{x,y}}$ that is visible at this position. For this task, we generate an IndexMap which stores the index of the visible triangle for each pixel during the forward pass of our differentiable renderer.

Our predicted per-vertex motion gradients $\nabla\mathbf{m}_i$ approximate the second term in Eq. (8.2) that cannot be computed analytically (also see Eq. (8.3)). In this way, our approach combines many local per-pixel 2D displacement vectors into per-vertex motion gradients and further uses these to compute accurate global 3D pose gradients considering the 3D model geometry and the rendering pipeline. As our experiments in Section 8.3 show, predicting local 2D displacement vectors is easier than directly predicting global 3D pose updates and generalizes better to unseen data [159, 313].

Finally, during inference of our system, we perform iterative updates to refine $\mathcal{P}_{\text{curr}}$. In each iteration, we compute a 3D pose gradient by evaluating our refinement loop presented in Fig. 8.2. For our implementation, we use the Adam optimizer [138] with a small learning rate and perform multiple updates to account for noisy correspondences and achieve the best accuracy.

8.3 Evaluation

To demonstrate the benefits of our 3D pose refinement approach, we evaluate it on the challenging Pix3D [258] dataset which provides in-the-wild images for objects of different categories. Details on the evaluated dataset are presented in Chapter 4. In particular, we quantitatively and qualitatively compare our approach to state-of-the-art refinement methods in Section 8.3.2, perform an ablation study in Section 8.3.3, combine our refinement approach with our feed-forward 3D pose estimation (see Chapter 6) and 3D model retrieval (see Chapter 7) methods to predict fine-grained 3D poses without providing initial 3D poses or ground truth 3D models in Section 8.3.4, and investigate failure modes in Section 8.3.5. We follow the evaluation protocol of previous work [86] and report the median error ($MedErr$) of multiple geometric distances:

Rotation: The 3D rotation distance

$$e_{\mathbf{R}} = \frac{\| \log(\mathbf{R}_{\text{gt}}^T \mathbf{R}_{\text{pred}}) \|_F}{\sqrt{2}} \tag{8.7}$$

represents the minimal angle between the ground truth rotation matrix $\mathbf{R}_{\text{gt}}$ and the predicted rotation matrix $\mathbf{R}_{\text{pred}}$ [279].

Translation: The 3D translation distance

$$e_{\mathbf{t}} = \frac{\|\mathbf{t}_{\text{gt}} - \mathbf{t}_{\text{pred}}\|_2}{\|\mathbf{t}_{\text{gt}}\|_2} \tag{8.8}$$

gives the relative error between the ground truth translation $\mathbf{t}_{\text{gt}}$ and the predicted translation $\mathbf{t}_{\text{pred}}$ [109].

Pose: The 3D pose distance

$$e_{\mathbf{R},\mathbf{t}} = \operatorname*{avg}_{\mathbf{M}\in\mathcal{M}} \frac{d_{\text{bbox}}}{d_{\text{img}}} \cdot \frac{\|\text{transf}(\mathbf{M}, \mathbf{R}_{\text{gt}}, \mathbf{t}_{\text{gt}}) - \text{transf}(\mathbf{M}, \mathbf{R}_{\text{pred}}, \mathbf{t}_{\text{pred}})\|_2}{\|\mathbf{t}_{\text{gt}}\|_2} \tag{8.9}$$

represents the average normalized Euclidean distance of all transformed 3D model points in 3D space [105, 109]. Each 3D point $\mathbf{M}$ of the ground truth 3D model $\mathcal{M}$ is transformed using the ground truth 3D pose ($\mathbf{R}_{\text{gt}}$ and $\mathbf{t}_{\text{gt}}$) and the predicted 3D pose ($\mathbf{R}_{\text{pred}}$ and $\mathbf{t}_{\text{pred}}$). This distance is normalized by the relative size of the object in the image using the ratio between the ground truth 2D bounding box diagonal d_{bbox} and the image diagonal d_{img}, and the L2-norm of the ground truth translation $\|\mathbf{t}_{\text{gt}}\|_2$.

Projection: The 2D projection distance

$$e_P = \operatorname*{avg}_{\mathbf{M}\in\mathcal{M}} \frac{\|\text{proj}(\mathbf{M}, \mathbf{K}, \mathbf{R}_{\text{gt}}, \mathbf{t}_{\text{gt}}) - \text{proj}(\mathbf{M}, \mathbf{K}, \mathbf{R}_{\text{pred}}, \mathbf{t}_{\text{pred}})\|_2}{d_{\text{bbox}}} \tag{8.10}$$

is the average reprojection error normalized by the ground truth 2D bounding box diagonal d_{bbox} [288]. In this case, each 3D point $\mathbf{M}$ of the ground truth 3D model $\mathcal{M}$ is projected to a 2D location using the ground truth projection parameters ($\mathbf{K}$, $\mathbf{R}_{\text{gt}}$, and $\mathbf{t}_{\text{gt}}$) and the predicted projection parameters ($\mathbf{K}$, $\mathbf{R}_{\text{pred}}$, and $\mathbf{t}_{\text{pred}}$). In this work, we assume the intrinsic matrix $\mathbf{K}$ to be known.

8.3.1 Implementation Details

To train our refinement network, we resize and pad images to a spatial resolution of 256 × 256 while maintaining the aspect ratio. In this way, we are able to combine images with different aspect ratios in the same training batch. For our mapping branches, we adapt a ResNet-50 [99, 100] architecture. We utilize all layers up to the end of the first stage but use a stride of 1 for all convolutional layers and discard max pooling layers. For our correspondence branch, we use a channel dimensionality of 64 for all convolutional layers except the output layer.

During training of our system, we optimize our joint network $f(\cdot)$ consisting of our two mapping braches and our correspondence branch on random 3D pose perturbations, as described in Section 8.2.3. In addition to these artificial 3D pose perturbations, we employ different forms of data augmentation like mirroring, affine transformations, and independent pixel augmentations like additive or multiplicative noise for the RGB image.

To regularize this loss, we use L2 weight decay with a factor of $1e^{-5}$. We train our network $f(\cdot)$ for 1500 epochs using the Adam optimizer [138] with an initial learning rate of $\eta = 1e^{-3}$. We use a minibatch size of 8 and decrease the learning rate by a factor of 5 after 1000 and 1400 epochs.

3D model vertices belonging to self-occluded triangles [134] or to visible triangles which are masked out by our geometric attention module do not receive gradients. However, since the geometry is fixed, providing gradients for a subset of all 3D model vertices is sufficient to perform 3D pose refinement.

8.3.2 Comparison to the State of the Art

We compare our approach to state-of-the-art refinement methods. For this purpose, we perform 3D pose refinement on top of an initial 3D pose estimation baseline. In particular, we predict initial 3D poses using our feed-forward approach presented in Chapter 6 (*Baseline*) which is the state of the art for single image 3D pose estimation on the Pix3D dataset. We compare our refinement approach to traditional image-based refinement without differentiable rendering [313] (*Image Refinement*) and mask-based refinement with differentiable rendering [134] (*Mask Refinement*).

RGB-based refinement with differentiable rendering [171] is not possible in our setup because all available 3D models lack textures and materials. This approach even fails if we compare grey-scale images and renderings because the monochrome image intensities at corresponding locations do not match. As a consequence, 2D-3D alignment using a photo-metric loss is impossible.

For *Image Refinement*, we use grey-scale instead of RGB renderings because all available 3D models lack textures and materials. In addition, we do not perform a single full update [313] but perform up to 1000 iterative updates with a small learning rate of $\eta = 0.05$ using the Adam optimizer [138] for all methods.

For *Mask Refinement*, we predict instance masks from the input RGB image using Mask R-CNN [96]. To achieve maximum accuracy, we internally predict masks at four times the original spatial resolution proposed in Mask R-CNN and fine-tune a model pre-trained on COCO [165] on Pix3D.

8.3.2.1 Quantitative Results

First, Table 8.1 presents quantitative 3D pose refinement results for individual categories on Pix3D. In this experiment, we provide the ground truth 3D model of the object in the image for refinement. For completeness, we additionally report the detection accuracy $Acc_{D_{0.5}}$ which gives the percentage of objects for which the intersection over union between the ground truth 2D bounding box and the predicted 2D bounding box is larger than 50% [303]. We do not make 3D pose predictions or perform 3D pose refinement for objects which are not detected by the baseline [86]. However, the reported $MedErr$ metrics are computed over all samples, both detected and not detected.

Compared to the baseline, *Image Refinement* only achieves a small improvement in the 3D rotation, 3D translation, and 3D pose metrics. There is almost no improvement in the projection metric ($MedErr_P$) as this method does not minimize the reprojection error. Traditional refinement methods are not aware of the rendering pipeline and the underlying 3D scene geometry and can only provide coarse 3D pose updates [313]. In our in-the-wild scenario, the number of 3D models, possible 3D pose perturbations, and category-level

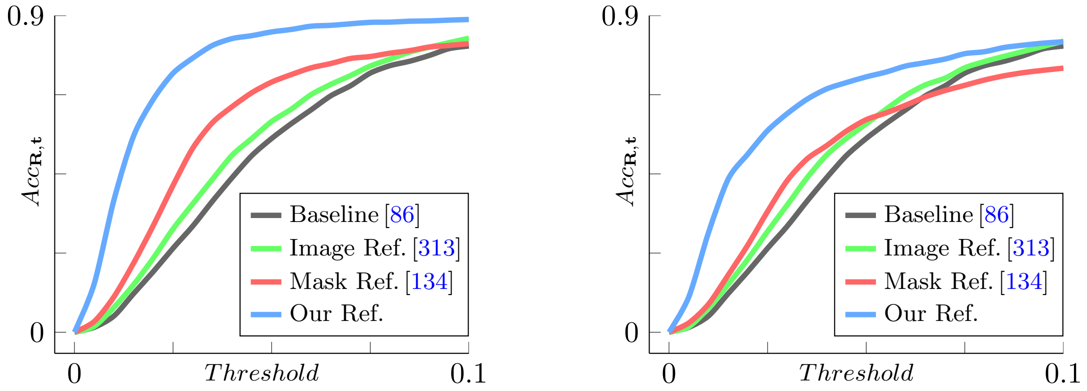
		Detection	Rotation	Translation	Pose	Projection
Method	Category	$Acc_{D_{0.5}}$	$MedErr_{\mathbf{R}}$ $\cdot 1$	$MedErr_{\mathbf{t}}$ $\cdot 10^2$	$MedErr_{\mathbf{R},\mathbf{t}}$ $\cdot 10^2$	$MedErr_{P}$ $\cdot 10^2$
Baseline [86]	bed	99.0	5.07	6.68	5.18	3.42
Image Refinement [313]			4.65	5.45	4.60	3.38
Mask Refinement [134]			3.03	4.04	3.07	2.00
Our Refinement			**2.40**	**1.84**	**1.45**	**1.28**
Baseline [86]	chair	95.2	7.36	5.49	3.90	3.32
Image Refinement [313]			7.10	5.31	3.68	3.34
Mask Refinement [134]			4.42	4.89	3.17	1.79
Our Refinement			**2.96**	**1.77**	**1.23**	**1.17**
Baseline [86]	sofa	96.5	4.40	4.96	3.78	2.57
Image Refinement [313]			4.30	3.87	3.15	2.54
Mask Refinement [134]			2.97	2.89	2.25	1.54
Our Refinement			**2.28**	**1.36**	**1.19**	**1.08**
Baseline [86]	table	94.0	10.18	7.72	6.17	5.54
Image Refinement [313]			9.81	7.07	5.80	5.40
Mask Refinement [134]			3.81	4.44	3.34	2.27
Our Refinement			**2.59**	**2.00**	**1.48**	**1.55**
Baseline [86]	*mean*	96.2	6.75	6.21	4.76	3.71
Image Refinement [313]			6.46	5.43	4.31	3.67
Mask Refinement [134]			3.56	4.06	2.96	1.90
Our Refinement			**2.56**	**1.74**	**1.34**	**1.27**

Table 8.1: Quantitative 3D pose refinement results for individual categories on the Pix3D dataset. In this experiment, we provide the ground truth 3D model for refinement. We outperform existing methods across all categories and all metrics by a large margin.

appearance variations is too large to simulate all permutations during training. As a consequence, this method cannot generalize to examples which are far from the ones seen during training and only achieves small improvements.

Next, *Mask Refinement* improves upon *Image Refinement* by a large margin across all metrics. Due to the 2D-3D alignment with differentiable rendering, this method computes more accurate 3D pose updates and additionally reduces the reprojection error ($MedErr_P$). However, we observe that *Mask Refinement* fails in two common situations: First, when the object has holes and the mask is not a single blob the refinement fails (see Fig. 8.7, 2nd and 3rd row). In the presence of holes, the hand-crafted approximation for the rasterization backward pass accumulates gradients with alternating signs while traversing the image. This results in unreliable per-vertex motion gradients. Second, simply aligning the silhouette of objects is ambiguous as renderings under different 3D poses can have similar masks. The interior structure of the object is completely ignored. As a consequence, the refinement gets stuck in poor local minima. Finally, the performance of *Mask Refinement* is limited by the quality of the predicted masks [96].

In contrast, our refinement overcomes these limitations and significantly outperforms the baseline as well as competing refinement methods across all metrics by up to 70% and



(a) Ground truth 3D models

(b) Retrieved 3D models

Figure 8.4: Evaluation on 3D pose accuracy at different thresholds. We significantly outperform other methods on strict thresholds using both ground truth and retrieved 3D models. The curves show that our approach performs especially well in the fine-grained regime.

55% relative. In fact, we do not only increase the *mean* performance over all categories but also achieve state-of-the-art results for each individual category. Using our geometric correspondence fields, we bridge the domain gap between real-world images and synthetic renderings, compute accurate global 3D pose updates, and precisely align both the object outline as well as interior structures.

Our approach performs especially well in the fine-grained regime, as shown in Fig. 8.4a. In this experiment, we plot the 3D pose accuracy $Acc_{\mathbf{R},\mathbf{t}}$ which gives the percentage of samples for which the 3D pose distance $e_{\mathbf{R},\mathbf{t}}$ is smaller than a varying threshold. For strict thresholds close to zero, our approach outperforms other refinement methods by a large margin. For example, at the threshold 0.015, we achieve more than 55% accuracy while the runner-up *Mask Refinement* achieves only 19% accuracy.

8.3.2.2 Iterative Refinement

In this section, we analyze the iterative refinement behavior of different methods. Fig. 8.5 shows the performance of different refinement methods after varying numbers of iterations. In this experiment, we report the 3D pose metric $MedErr_{\mathbf{R},\mathbf{t}}$ as a function of the number of 3D pose updates. The baseline does not perform iterative updates, thus, the $MedErr_{\mathbf{R},\mathbf{t}}$ is constant for different iterations.

For *Image Refinement* [313], the accuracy increases until 20 iterations but then starts to decrease. After the first couple of coarse refinement steps, the predicted updates are not accurate enough to refine the 3D pose but start to jitter without further improving the 3D pose. Moreover, for many objects the prediction fails and the iterative updates cause the 3D pose to drift off which results in high $MedErr_{\mathbf{R},\mathbf{t}}$ for large numbers of iterations. Empirically, we obtain the best overall results for this method using only 20 iterations.

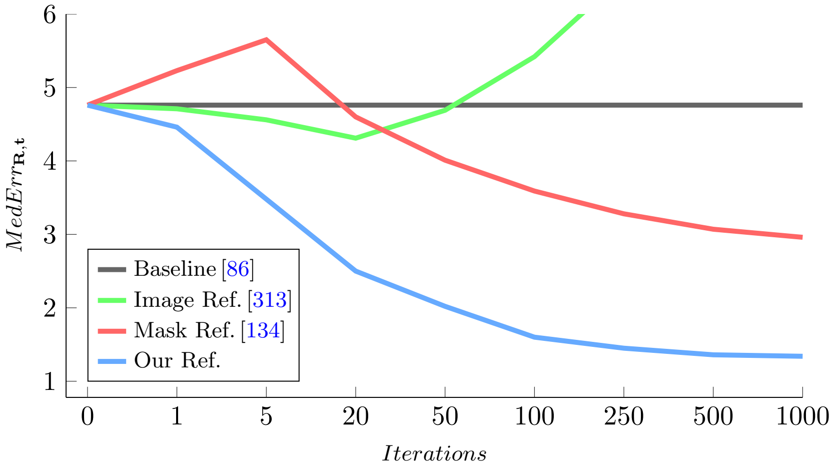
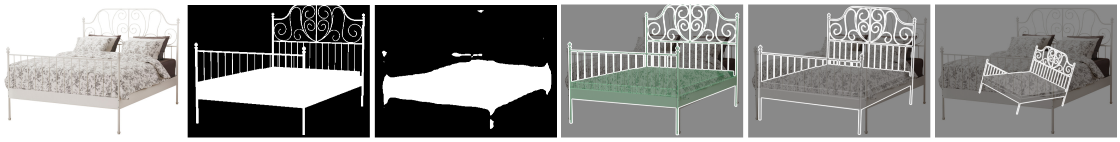


Figure 8.5: Evaluation on 3D pose refinement after varying numbers of iterations. In contrast to other methods, our refinement achieves a consistent improvement upon the baseline, increasing with the number of iterations, as discussed in Section 8.3.2.2.

Thus, we only perform *Image Refinement* with 20 iterative 3D pose updates in all other experiments. For all other methods based on differentiable rendering, we achieve the best accuracy after the full 1000 iterations.

For *Mask Refinement* [134], we observe an opposite effect. In the beginning, the accuracy decreases but then the performance increases. This is due to degenerated masks predicted from RGB images by Mask R-CNN [96]. The mask prediction often fails to capture fine-grained and thin structures, e.g., ornaments and legs of a bed (see Fig. 8.6). These degenerated masks cause large gradients during refinement and quickly pull the 3D pose away from the reasonable initial estimate predicted by the baseline (also see Fig. 8.7, 2nd and 3rd row). After five iterations the refinement on samples with correctly predicted masks counteracts this effect and the performance improves.



Image

Ground Truth Mask

Predicted Mask

Ground Truth 3D Pose

Baseline [86] 3D Pose

Mask Ref. [134] 3D Pose

Figure 8.6: Typical failure case of *Mask Refinement* [134]. Degenerated predicted masks cause large gradients during refinement and quickly pull the 3D pose away from reasonable initial estimates by the baseline. As a consequence, the 3D pose refinement using *Mask Refinement* fails (also see Fig. 8.7, 3rd row).

In contrast to other refinement methods, our approach achieves a consistent improvement upon the baseline, monotonically increasing with the number of iterations. As expected, the accuracy saturates for large numbers of iterations. Empirically, we achieve

Method	Time per Iteration
Image Refinement [313]	86.5 ms
Mask Refinement [134]	29.7 ms
Our Refinement	36.0 ms

Table 8.2: Average computation times of different refinement methods for a single iteration using a Titan X GPU. For all evaluated methods, the execution time for computing a single 3D pose update is within the same order of magnitude.

maximum accuracy by performing 1000 3D pose updates using the Adam optimizer [138] with a learning rate of $\eta = 0.05$.

Finally, Table 8.2 compares the computation times of different refinement methods. *Image Refinement* evaluates two large ResNet-style networks [99] during inference and, thus, is the slowest of our evaluated refinement methods.

Mask Refinement generates a mask rendering, computes a loss in the mask space, and performs the backward pass of its differentiable renderer in each iteration. This method is the fastest in our evaluation since the target mask predicted from the input RGB image by Mask R-CNN [96] does not change during refinement. It is only predicted once before the iterative process and the inference time of Mask R-CNN is not considered in this experiment.

Our refinement is only marginally slower than *Mask Refinement* because we just evaluate our efficient network branches in addition to the forward and backward pass of our differentiable renderer in each iteration. However, all evaluated refinement methods show comparable execution time within the same order of magnitude for computing a single 3D pose update.

8.3.2.3 Qualitative Results

The significant quantitative performance improvement of our refinement method compared to the state of the art is also reflected in numerous qualitative examples presented in Figs. 8.7 to 8.9. Our approach precisely aligns 3D models to objects in RGB images and computes 3D poses which are in many cases visually indistinguishable from the ground truth. Even if the initial 3D pose estimate (*Baseline*) is significantly off, our method can converge towards the correct 3D pose (see Fig. 8.7, 1st row).

Finally, Fig. 8.10 illustrates the high quality of our predicted geometric correspondence fields. Our predicted 2D displacement vectors are highly accurate for many different objects and scales, even though we address objects of all categories with a single class-agnostic network. We restrict the correspondence prediction to visible object surface regions with significant geometric discontinuities which are identified by our geometric attention module. The examples also show our computed geometric attention masks outlined in white underneath the 2D displacement vectors.

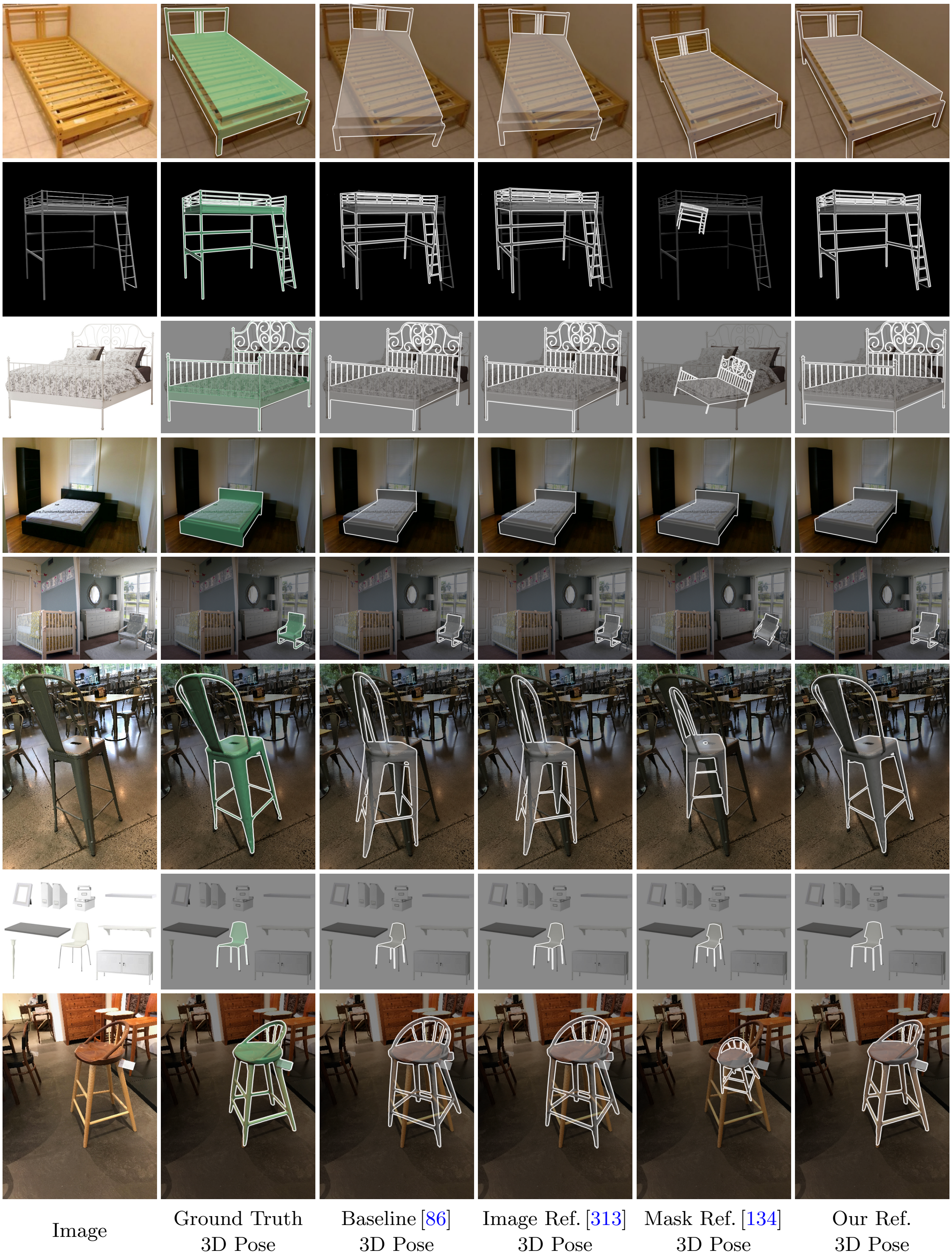


Figure 8.7: Qualitative 3D pose refinement results for objects of different categories. We project the ground truth 3D model onto the image using the 3D pose predicted by different methods. Our approach overcomes the limitations of previous methods and predicts fine-grained 3D poses which are in many cases visually indistinguishable from the ground truth.

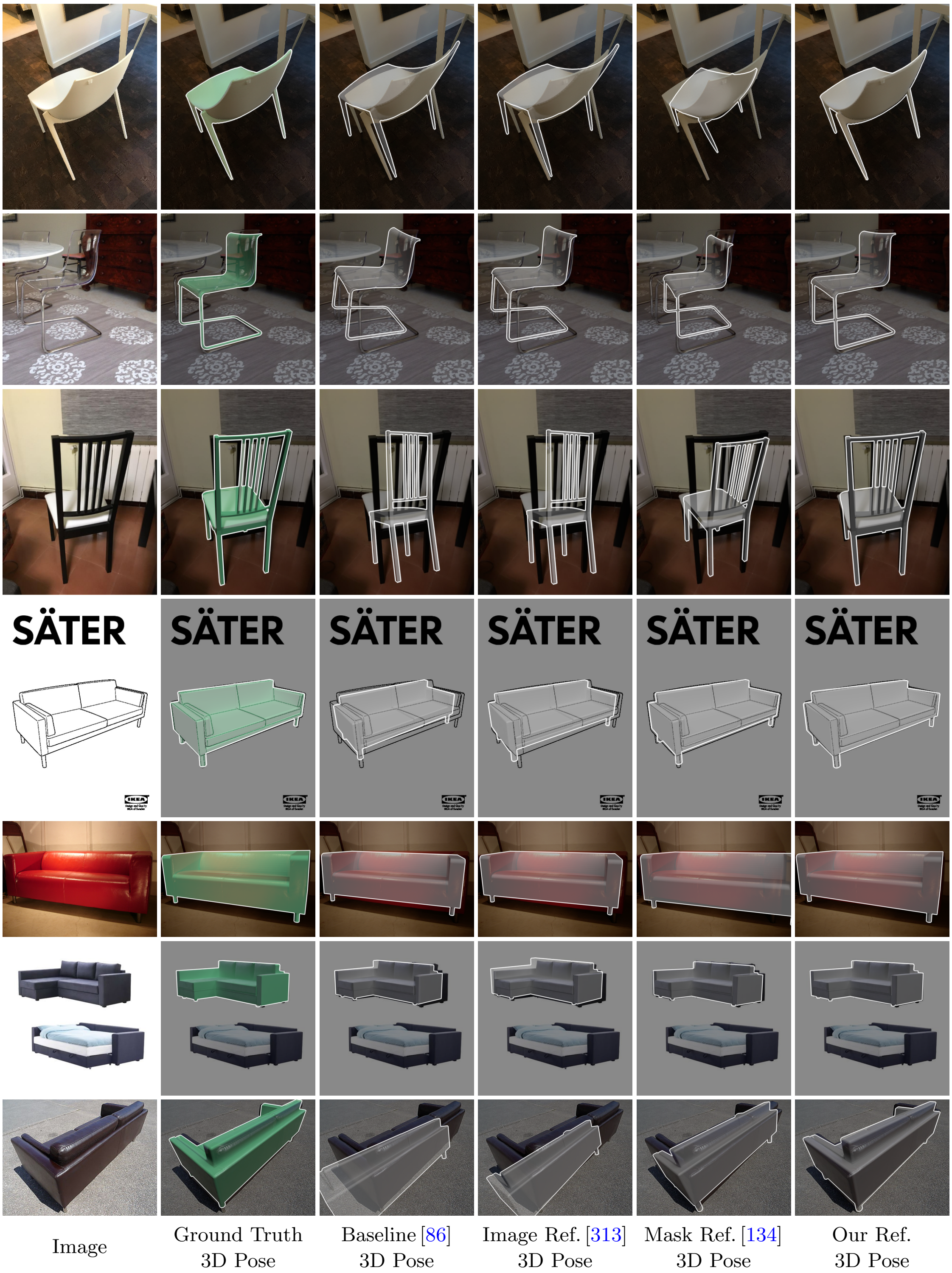


Figure 8.8: Qualitative 3D pose refinement results for objects of different categories. We project the ground truth 3D model onto the image using the 3D pose predicted by different methods. Our approach overcomes the limitations of previous methods and predicts fine-grained 3D poses which are in many cases visually indistinguishable from the ground truth.

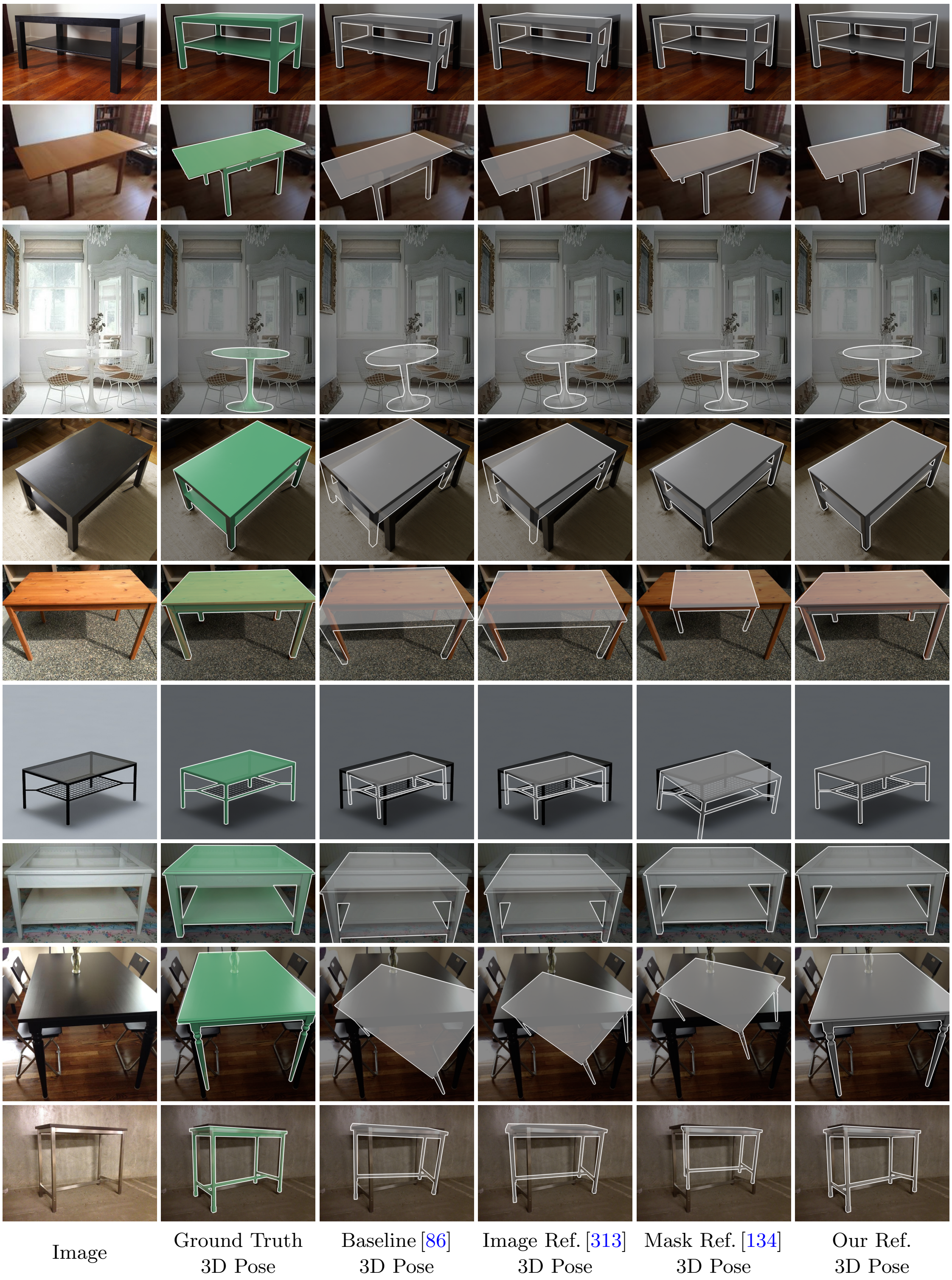


Figure 8.9: Qualitative 3D pose refinement results for objects of different categories. We project the ground truth 3D model onto the image using the 3D pose predicted by different methods. Our approach overcomes the limitations of previous methods and predicts fine-grained 3D poses which are in many cases visually indistinguishable from the ground truth.

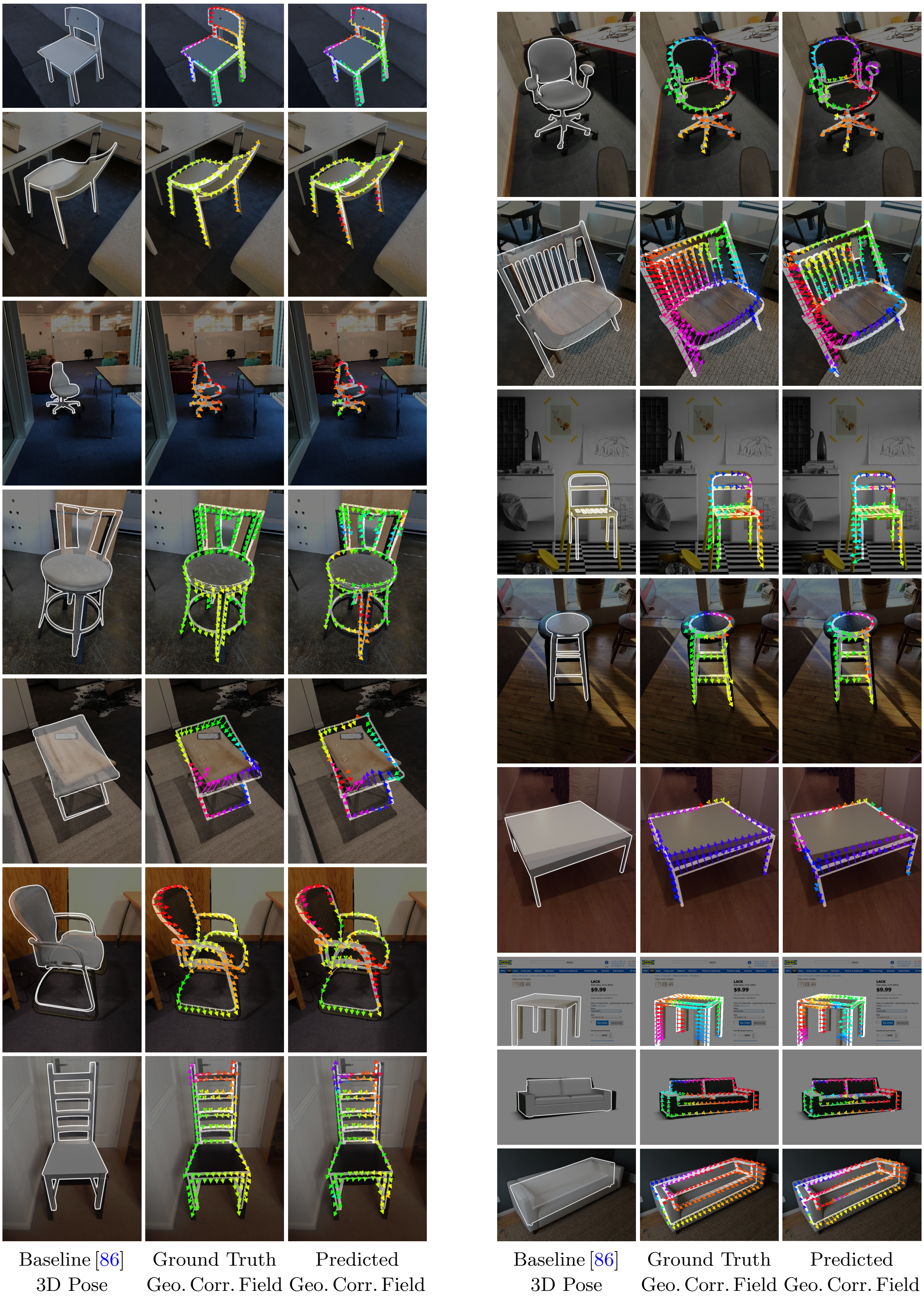


Figure 8.10: Qualitative examples of our predicted geometric correspondence fields. Our predicted 2D displacement vectors are highly accurate. Our computed geometric attention masks are outlined in white underneath the 2D displacement vectors.

	Rotation	Translation	Pose	Projection
Method	$MedErr_{\mathbf{R}}$ $\cdot 1$	$MedErr_{\mathbf{t}}$ $\cdot 10^2$	$MedErr_{\mathbf{R},\mathbf{t}}$ $\cdot 10^2$	$MedErr_{P}$ $\cdot 10^2$
Ours less receptive	3.01	2.12	1.75	1.56
Ours fewer layers	2.84	1.98	1.58	1.41
Ours shallow features	2.76	2.00	1.55	1.42
Ours without attention	2.70	1.92	1.45	1.37
Ours only MASK	2.98	1.85	1.44	1.41
Ours only LF	2.57	1.80	1.39	1.31
Ours only DEPTH	2.60	1.77	1.38	1.29
Ours only NORMAL	2.58	1.76	1.36	1.30
Ours	**2.56**	**1.74**	**1.34**	**1.27**

Table 8.3: Ablation study of our method. Using components which increase the discriminability of learned features is most important for the performance of our approach.

8.3.3 Ablation Study

To understand the importance of individual components in our system, we conduct an ablation study in Table 8.3. For this purpose, we modify a specific system component, retrain our approach, and evaluate the performance impact.

If we use smaller kernels with less receptive field (3 × 3 versus 7 × 7) or fewer layers (2 versus 4) in our correspondence branch, the performance drops significantly. Also, using shallow mapping branches which only employ a single Conv-BN-ReLU block to simulate simple edge and ridge features results in low accuracy because the computed features are not discriminative enough. If we perform refinement without our geometric attention mechanism, the accuracy degrades due to unreliable correspondence predictions in homogeneous regions.

Next, the choice of the rendered representation is important for the performance of our approach. Using masks only performs poorly because masks discard valuable shape information which makes 2D-3D alignment ambiguous. In contrast, depth, normal, and location field renderings increase the accuracy. Finally, we achieve the best accuracy by exploiting complementary information from multiple different renderings by concatenating depth, normal, and location field renderings.

8.3.4 3D Model Retrieval

So far, we assumed that the ground truth 3D model required for 3D pose refinement is given. However, we can overcome this limitation by automatically retrieving 3D models from single RGB images. For this purpose, we combine all refinement approaches with our 3D model retrieval method presented in Chapter 7, where the 3D model database essentially becomes a part of the trained model. In this way, we perform initial 3D pose

Method	Category	**Detection**	**Rotation**	**Translation**	**Pose**	**Projection**
		$Acc_{D_{0.5}}$	$MedErr_{\mathbf{R}}$ $\cdot 1$	$MedErr_{\mathbf{t}}$ $\cdot 10^2$	$MedErr_{\mathbf{R},\mathbf{t}}$ $\cdot 10^2$	$MedErr_P$ $\cdot 10^2$
Baseline [86]	bed	99.0	5.07	6.68	5.18	3.42
Image Refinement [313]			4.65	5.86	4.41	3.38
Mask Refinement [134]			4.40	5.32	4.21	2.69
Our Refinement			**2.95**	**2.75**	**2.18**	**1.83**
Baseline [86]	chair	95.2	7.36	5.49	3.90	3.32
Image Refinement [313]			7.15	5.21	3.67	3.35
Mask Refinement [134]			7.22	6.32	4.53	3.31
Our Refinement			**4.89**	**2.87**	**2.04**	**2.19**
Baseline [86]	sofa	96.5	4.40	4.96	3.78	2.57
Image Refinement [313]			4.34	3.75	3.02	2.54
Mask Refinement [134]			3.33	3.00	2.34	1.63
Our Refinement			**2.60**	**1.60**	**1.42**	**1.19**
Baseline [86]	table	94.0	10.18	7.72	6.17	5.54
Image Refinement [313]			9.73	7.23	6.22	5.68
Mask Refinement [134]			6.92	6.34	5.52	4.85
Our Refinement			**4.73**	**3.38**	**2.93**	**3.49**
Baseline [86]	*mean*	96.2	6.75	6.21	4.76	3.71
Image Refinement [313]			6.47	5.51	4.33	3.74
Mask Refinement [134]			5.47	5.25	4.15	3.12
Our Refinement			**3.79**	**2.65**	**2.14**	**2.18**

Table 8.4: Quantitative 3D pose refinement results for individual categories on the Pix3D dataset. In this experiment, we retrieve 3D models for refinement using our method presented in Chapter 7. We outperform existing methods across all categories and all metrics by a large margin.

estimation, 3D model retrieval, and 3D pose refinement given only a single RGB image at runtime. This setting allows us to benchmark refinement methods against feed-forward baselines in a fair comparison.

The corresponding results are presented in Table 8.4 and Fig. 8.4b. Because the retrieved 3D models often differ from the ground truth 3D models, the refinement performance decreases compared to the case of given ground truth 3D models. Differentiable rendering methods lose more accuracy than traditional refinement methods because they require 3D models with accurate geometry.

Still, all refinement approaches perform remarkably well with retrieved 3D models. As long as the retrieved 3D model is reasonably close to the ground truth 3D model in terms of geometry, our refinement succeeds. Our method achieves even lower 3D pose error ($MedErr_{\mathbf{R},\mathbf{t}}$) with retrieved 3D models than *Mask Refinement* with ground truth 3D models. Finally, using our joint 3D pose estimation-retrieval-refinement pipeline, we reduce the 3D pose error ($MedErr_{\mathbf{R},\mathbf{t}}$) compared to the state of the art for single image 3D pose estimation on Pix3D (*Baseline*) **by 55%** relative without using additional inputs. A summary on the refinement performance of different methods using both ground truth 3D models and retrieved 3D models is shown in Table 8.5.

		Rotation	**Translation**	**Pose**	**Projection**
Method	Input	$MedErr_{\mathbf{R}}$ $\cdot 1$	$MedErr_{\mathbf{t}}$ $\cdot 10^2$	$MedErr_{\mathbf{R},\mathbf{t}}$ $\cdot 10^2$	$MedErr_P$ $\cdot 10^2$
Baseline [86]	RGB Image	6.75	6.21	4.76	3.71
Image Refinement [313]	RGB Image + 3D Model	6.46	5.43	4.31	3.67
Mask Refinement [134]	RGB Image + 3D Model	3.56	4.06	2.96	1.90
Our Refinement	RGB Image + 3D Model	**2.56**	**1.74**	**1.34**	**1.27**
Image Refinement [313] + Retrieval [87]	RGB Image	6.47	5.51	4.33	3.74
Mask Refinement [134] + Retrieval [87]	RGB Image	5.47	5.25	4.15	3.12
Our Refinement + Retrieval [87]	RGB Image	**3.79**	**2.65**	**2.14**	**2.18**

Table 8.5: Summary of our quantitative results on the Pix3D dataset. In the case of provided ground truth 3D models, our refinement significantly outperforms previous refinement methods across all metrics by up to 55% relative. In the case of automatically retrieved 3D models (+ Retrieval [87]), we reduce the 3D pose error ($MedErr_{\mathbf{R},\mathbf{t}}$) compared to the state of the art for single image 3D pose estimation on Pix3D (*Baseline*) by 55% relative without using additional inputs at runtime.

8.3.5 Failure Modes

Finally, Fig. 8.11 shows failure modes of our approach. We observe that our refinement method does not converge if the initial 3D pose is too far from the ground truth 3D pose. In this case, we cannot predict accurate correspondences because our computed features are not robust to large viewpoint changes and the receptive field of our correspondence branch is limited. Also, strong image noise and duplicate or ambiguous structures in the image confuse our refinement. In these cases, our method sometimes fails and aligns the 3D model to wrong parts of the RGB image.

In addition, occlusions cause our refinement to fail because there are no explicit mechanisms to address them. Neither traditional refinement methods [159, 181, 313], nor differentiable rendering based refinement methods [134, 171, 200], nor our approach employ explicit mechanisms to deal with occlusions. For example, one issue across all evaluated methods is that they align renderings to real-world images but occlusions are only present in the real-world images while the renderings are always un-occluded. Also, simply training methods based on feed-forward CNNs on occluded objects is in practice not sufficient to handle occlusions [198]. However, we specifically plan to address occlusions in the future by predicting occlusion masks and correspondence confidences.

8.4 Conclusion

Aligning 3D models to objects in RGB images is the most accurate way to predict 3D poses. However, there is a domain gap between real-world images and renderings which makes this alignment challenging in practice. To address this problem, we predict deep cross-domain correspondences in a feature space optimized for 3D pose refinement and use differentiable rendering to combine local 2D correspondences into a global 3D pose update. Our method outperforms existing refinement approaches by up to 55% relative

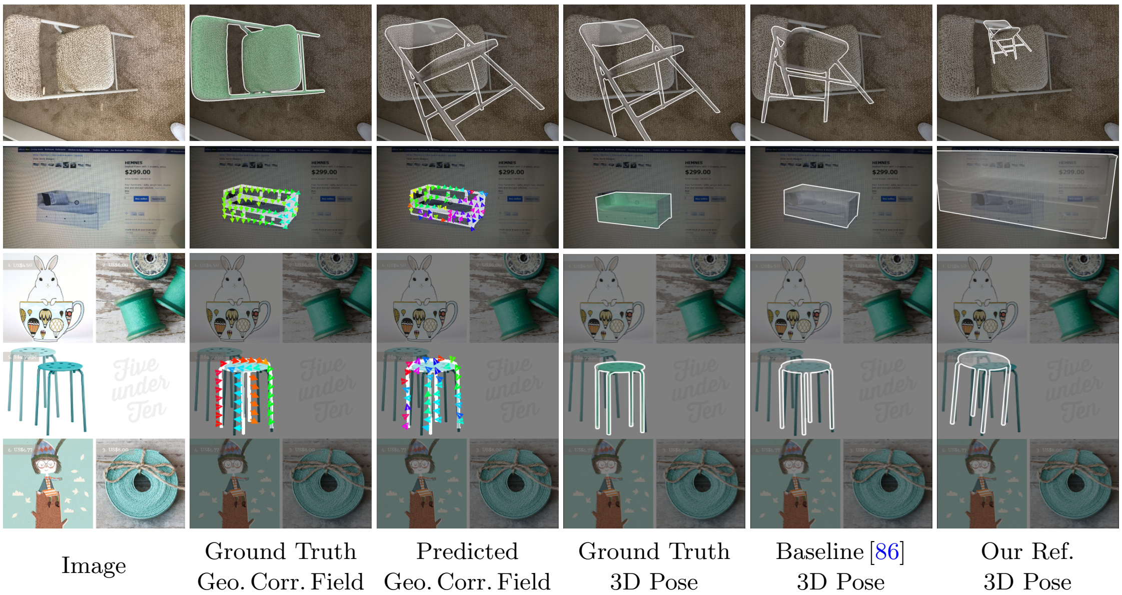


Figure 8.11: Failure cases of our approach. If the initial 3D pose is too far from the ground truth 3D pose (1st row), if the image is degraded by strong noise (2nd row), or if there are duplicate or ambiguous structures in the image (3rd row), our refinement does not converge. In these cases, we cannot predict accurate correspondences in the form of geometric correspondence fields and align the 3D model to wrong parts of the RGB image.

and can be combined with feed-forward 3D pose estimation and 3D model retrieval to predict fine-grained 3D poses given only a single RGB image. Finally, our novel learned differentiable rendering framework can be used for other tasks in the future.

CHAPTER 9

Conclusion

Contents

In this final chapter, we draw conclusions by summarizing our contributions presented in Chapters 5 to 8 and discussing future research directions.

9.1 Summary

In this thesis, we advanced the state of the art in single image 3D vision for objects in the wild. We presented novel methods for estimating different 3D properties which contribute to a holistic 3D scene understanding. Our methods perform object detection, 3D pose estimation, 3D dimension estimation, 3D model retrieval, and estimation of camera intrinsics given only a single RGB image. All of our proposed methods use sophisticated combinations of deep learning and geometry to leverage prior information via learning while exploiting geometric constraints on the desired solutions.

We first introduced a novel 3D pose estimation and 3D model retrieval approach for objects of arbitrary categories in Chapter 5. This method estimates the 3D pose of an object from predicted virtual 2D-3D correspondences in a category-agnostic way. In particular, our method predicts the 3D coordinates of the 3D bounding box corners of an object as well as their 2D image locations and recovers the 3D pose using a PnP algorithm. We then showed that the 3D pose of an object provides an effective prior for 3D model retrieval by comparing RGB images of objects only to 3D model renderings under our predicted 3D pose. Utilizing this prior significantly increases the retrieval performance while reducing the computational complexity. Further, we showed that our approach is efficient at runtime as it supports offline pre-computed 3D shape descriptors for 3D model retrieval.

In Chapter 6, we proposed improvements to the 3D pose estimation part of our approach introduced in Chapter 5. In particular, we integrated our 3D pose estimation method into an object detection pipeline to deal with multiple objects at different scales in a single image. Additionally, we augmented our system to estimate the camera focal length and presented a joint optimization which enforces a geometric consensus between predicted 3D poses and the focal length. In this way, we exploited the geometric prior given by the focal length for 3D pose estimation and significantly improved our 3D pose estimates. Finally, we compared different forms of 2D-3D correspondences for 3D pose estimation. In addition to sparse 2D-3D bounding box corner correspondences, as presented in Chapter 5, we evaluated dense 2D-3D correspondences in the form of location fields.

In Chapter 7, we proposed improvements to the 3D model retrieval part of our approach introduced in Chapter 5. In particular, we presented a novel approach to compute 3D shape descriptors. This method first maps RGB images and 3D models to a common pose aware low-level representation in the form of location fields and then computes pose invariant 3D shape descriptors for retrieval. We showed that this method learns more discriminative 3D shape descriptors, effectively exploits training on a virtually infinite amount of synthetic data, and yields visually interpretable intermediate predictions that unblackbox our system. This results in increased 3D model retrieval performance while significantly reducing the size of the offline pre-computed 3D shape descriptor database compared to our previous approach presented in Chapter 5.

Finally, we presented a novel 3D pose refinement strategy based on differentiable rendering in Chapter 8. By reprojecting our retrieved 3D models under our predicted 3D poses computed as described in Chapters 5 to 7, we established a feedback loop which we used to improve our predicted 3D poses. Our method compares RGB images and 3D model renderings in a feature space which is optimized for 3D pose refinement and additionally learns to approximate the non-differentiable rasterization backward pass in differentiable rendering. We showed that this method precisely aligns 3D models to objects in RGB images and computes 3D poses which are in many cases visually indistinguishable from ground truth 3D poses.

Our methods gradually contributed to a joint system that infers a semantic 3D reconstruction of a scene given only a single RGB image. This system is able to detect multiple objects of arbitrary categories at different scales, estimate a 3D pose for each object, retrieve a 3D model for each object, predict the focal length of the camera, and jointly refine all predictions using differentiable rendering. In this way, we achieve a compact but detailed high-level 3D scene understanding with rich semantics and object hierarchies that paves the way for a host of applications.

9.2 Future Directions

While our proposed methods achieve remarkable performance in many scenarios, there is always room for improvement. In the following, we discuss possible future research directions to improve the accuracy and robustness of single image 3D vision for objects in the wild.

Large-Scale Datasets: Today, state-of-the-art approaches in almost all computer vision disciplines are data-driven [96]. Many of these approaches including our proposed methods internally rely on deep learning systems. However, deep learning systems require massive amounts of data to achieve high performance. Data is the fuel for their success. In fact, the simplest and most effective strategy to improve the accuracy and robustness of learning-based systems is to provide them with more data during training.

While there are large-scale datasets for 2D object recognition tasks like classification [233] or detection [165], data in the form of 2D images with 3D annotations is scarce, as discussed in Chapter 4. One reason for this dilemma is that labeling 3D properties for 2D images is hard and time-consuming [288, 303]. Labeling 3D properties requires domain expertise and numerous 3D labels have many interacting degrees of freedom that cannot be annotated independently of each other. For example, to provide a complete 3D labeling for an arbitrary image captured in the wild, an annotator needs to reconstruct the entire scene in 3D which possibly requires crafting a 3D model for each object from scratch using computer-aided design.

As a consequence of this cumbersome annotation process, not only the amount of available images with 3D labels is limited but also the quality of the annotations is insufficient [302]. In many cases, images are only partially labeled, inaccurately labeled, or even mislabeled such that there is a high degree of label noise in the data. Due to this lack of data, deep learning systems for 3D tasks have not seen the same success as deep learning systems for 2D tasks. Thus, efficient but accurate 3D labeling techniques are a fundamental future requirement. Ideally, automated or semi-automated labeling of 3D properties should be exploited to speed up the annotation process.

However, *large-scale* datasets are not enough. In fact, we require datasets that densely sample and accurately represent the distribution of natural images encountered in specific tasks. For example, a dataset consisting of a billion 2D images with perfect 3D annotations showing the same dining room with a table and four chairs is not sufficient to train a model that accurately and robustly detects arbitrary chairs in the wild. In this context, our experiments showed that deep learning systems fail if the examples encountered during testing are far from the examples seen during training. Thus, we require datasets that resemble the diversity of the actual problem.

Synthetic Data: One strategy to satisfy the thirst of deep learning systems for data is to rely on synthetic data during training [257]. Generating RGB images by rendering 3D scenes provides perfect and free 3D annotations. However, there is a significant domain gap between real-world RGB images and rendered RGB images which limits the benefit of synthetic data in practice, as discussed in Chapter 4.

This domain gap is caused by a variety of different reasons. For example, many 3D models only provide geometry but no textures or materials [302, 303]. In contrast, 3D models with color information often provide unrealistic textures and materials [33]. Many rendering pipelines use simple shading computations based on naive reflection assumptions, rely on primitive lighting, and ignore real camera effects [4]. Moreover, many 3D models are rendered in isolation. Thus, the rendered 3D scene only consists of a single 3D model and there are no geometrically plausible occlusions, shadows, or indirect light sources. Further, many renderings are poorly integrated into natural image backgrounds in a post-processing step [257] or entirely lack backgrounds [185]. As a consequence of these naive rendering setups and over-simplified assumptions, many rendered RGB images are not photo-realistic and do not mirror the distribution of natural images.

However, there is a large amount of 3D content with a high degree of realism but it is currently too difficult to access or not effectively used. For example, the CARLA [49] simulator provides an open-source platform for autonomous driving research in which many objects interact in physically plausible ways in a realistic setup. Similarly, Habitat [239] is a simulation platform for research in embodied artificial intelligence. Moreover, modern video games provide complex simulations with sophisticated shading and advanced physics. While many top titles are closed source, some games offer APIs to extract information[1] and there are even successful attempts to extract 3D labels during the simulation [227, 228].

Finally, many 3D artists target hyper-realism. They use complex 3D scenes, physically-based materials, realistic textures, full global illumination [52, 129], and advanced compositing to generate renderings which are visually indistinguishable from real-world images. This kind of 3D content is typically generated for advertisements, movies, and architectural scenes[2]. However, gathering photo-realistic renderings with perfect and free 3D annotations from existing 3D content is a promising future direction to efficiently collect data and improve the accuracy and robustness of learning-based approaches.

Beyond Supervised Learning: In this work, we focused on supervised learning, where pairs of corresponding inputs and outputs are available during training. However, there are techniques that do not require manually assigned output labels for learning. For example, self-supervised learning methods [78, 79, 289] automatically generate labels for training data without human intervention. In this case, known constraints between multiple cameras [219], fixed objects in video streams [160], or defocus cues [92] can be exploited to generate supervision. Further, unsupervised learning [106] is a way to discover patterns without providing labels. In this way, concepts like 3D pose can be learned from constrained collections of data [120, 133, 226]. Finally, reinforcement learning [261] makes

[1] `https://www.projectcarsgame.com`

[2] `https://www.cgarchitect.com`

it possible to learn an algorithm to address tasks like 3D pose estimation [148]. In summary, learning beyond supervised samples is a promising future direction to increase the generalization of single image 3D vision systems.

Network Architectures for 3D Tasks: In recent years, deep learning systems have significantly improved the state of the art across different computing disciplines like speech recognition [107] and image processing [99]. In this context, many breakthroughs arose from the active participation of lots of researchers in extremely competitive benchmark breaking challenges like ImageNet classification [233]. Specifically, there has been a run on designing neural network architectures that increase classification performance.

However, the success of certain architectures in popular benchmarks has led to a paradox transfer learning trend in computer vision. In particular, many researchers adopt network architectures trained for one task and only make minimal changes to address another task, e.g., only adjust the output layer dimensionality. In this way, they hope that (1) architectural advances on the original task transfer to improved performance on the new task, (2) feature extraction learned on the original task compensates for limited training data on the new task, and (3) a pre-trained model reduces the training time and makes it possible to train a large network with limited computational resources by freezing a significant number of weights in early layers. For example, many 3D pose estimation approaches [85, 217, 257, 279] build on network architectures that are over-engineered for the task of ImageNet classification [99, 146, 250].

While the capacity of these network architectures is large enough to approximate many functions well, this strategy is not optimal. In fact, the architecture of a network provides a strong prior but different tasks have different requirements. For example, translation invariance, e.g., encouraged by pooling layers, is a beneficial property for classification tasks but not for detection tasks, where the precise spatial location of features is important. Object detection, in turn, benefits from translation equivariance, where spatial input translations produce proportional spatial features translations. Similarly, network architectures that enforce rotation equivariance could provide a useful prior for 3D tasks [55, 56, 295]. Thus, specifically designing architectures for 3D tasks has the potential to significantly improve the accuracy and robustness of single image 3D vision in the wild.

However, instead of hand-crafting network architectures based on empirical trial and error, the architecture of a network can also be seen as an optimizable hyperparameter. In this way, neural architecture search [54] can be performed to learn architectures that are optimized for a particular task.

Differentiable Geometric Algorithms: Continuing the idea of the previous paragraph, a deep learning system is not limited to a sequence of convolutional and fully connected layers. In fact, a series of diverse differentiable modules can implement more complex algorithms while exploiting domain expertise. In this context, formulating geometric algorithms in a differentiable way makes it possible to effectively address problems which are inconvenient to solve using conventional interconnected neurons. For example, differentiable implementations of geometric transformations [96, 124], 2D keypoint detection and description [38, 47, 310], or robust estimation techniques like RANSAC [23, 25, 26] provide advanced functionality for complex but end-to-end trainable systems. While there have been successful attempts to make many geometric algorithms differentiable and integrate them into deep learning systems, there is still plenty of room for improvement.

Finally, differentiable rendering [171] is a powerful concept that provides inverse graphics capabilities by computing gradients for 3D scene parameters from 2D image observations, as shown in Chapter 8. A differentiable renderer provides a universally applicable tool for single image 3D vision that avoids the need for complex and specialized handcrafted systems. However, even though numerous approximations for differentiable rendering have been proposed [102, 134, 167], the problem is still considered unsolved.

Bibliography

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems, 2015, `https://www.tensorflow.org/`. (page 28)

[2] Y. I. Abdel-Aziz, H. M. Karara, and M. Hauck. Direct Linear Transformation from Comparator Coordinates into Object Space in Close-Range Photogrammetry. In *ASP Symposium on Close-Range Photogrammetry*, pages 1–18, 1971. (page 24, 47)

[3] E. Ahmed, A. Saint, A. E. R. Shabayek, K. Cherenkova, R. Das, G. Gusev, D. Aouada, and B. Ottersten. A Survey on Deep Learning Advances on Different 3D Data Representations. *arXiv:1808.01462*, 2018. (page 18)

[4] T. Akenine-Möller, E. Haines, and N. Hoffman. *Real-time Rendering*. CRC Press, 2019. (page 18, 22, 23, 24, 144)

[5] R. Al-Rfou, G. Alain, A. Almahairi, C. Angermueller, D. Bahdanau, N. Ballas, F. Bastien, J. Bayer, A. Belikov, A. Belopolsky, Y. Bengio, A. Bergeron, J. Bergstra, V. Bisson, J. Bleecher Snyder, N. Bouchard, N. Boulanger-Lewandowski, X. Bouthillier, A. De Brébisson, O. Breuleux, P.-L. Carrier, K. Cho, J. Chorowski, P. Christiano, T. Cooijmans, M.-A. Côté, M. Côté, A. Courville, Y. N. Dauphin, O. Delalleau, J. Demouth, G. Desjardins, S. Dieleman, L. Dinh, M. Ducoffe, V. Dumoulin, S. Ebrahimi Kahou, D. Erhan, Z. Fan, O. Firat, M. Germain, X. Glorot, I. Goodfellow, M. Graham, C. Gulcehre, P. Hamel, I. Harlouchet, J.-P. Heng, B. Hidasi, S. Honari, A. Jain, S. Jean, K. Jia, M. Korobov, V. Kulkarni, A. Lamb, P. Lamblin, E. Larsen, C. Laurent, S. Lee, S. Lefrancois, S. Lemieux, N. Léonard, Z. Lin, J. A. Livezey, C. Lorenz, J. Lowin, Q. Ma, P.-A. Manzagol, O. Mastropietro, R. T. McGibbon, R. Memisevic, B. Van Merriënboer, V. Michalski, M. Mirza, A. Orlandi, C. Pal, R. Pascanu, M. Pezeshki, C. Raffel, D. Renshaw, M. Rocklin, A. Romero, M. Roth, P. Sadowski, J. Salvatier, F. Savard, J. Schlüter, J. Schulman, G. Schwartz, I. V. Serban, D. Serdyuk, S. Shabanian, E. Simon, S. Spieckermann, S. R. Subramanyam, J. Sygnowski, J. Tanguay, G. Van Tulder, J. Turian, S. Urban, P. Vincent, F. Visin, H. De Vries, D. Warde-Farley, D. J. Webb, M. Willson, K. Xu, L. Xue, L. Yao, S. Zhang, and Y. Zhang. Theano: A Python Framework for Fast Computation of Mathematical Expressions. *arXiv:1605.02688*, 2016. (page 28)

[6] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Y. Wu. An Optimal

Algorithm for Approximate Nearest Neighbor Searching Fixed Dimensions. *Journal of the ACM*, 45(6):891–923, 1998. (page 46)

[7] N. Aspert, D. Santa-Cruz, and T. Ebrahimi. Mesh: Measuring Errors between Surfaces using the Hausdorff Distance. In *International Conference on Multimedia and Expo*, pages 705–708, 2002. (page 105)

[8] M. Aubry, D. Maturana, A. Efros, B. Russell, and J. Sivic. Seeing 3D Chairs: Exemplar Part-Based 2D-3D Alignment Using a Large Dataset of CAD Models. In *Conference on Computer Vision and Pattern Recognition*, pages 3762–3769, 2014. (page 46)

[9] M. Aubry, D. Maturana, A. Efros, B. Russell, and J. Sivic. Strike (with) a Pose: Neural Networks are Easily Fooled by Strange Poses of Familiar Objects. In *Conference on Computer Vision and Pattern Recognition*, pages 4845–4854, 2019. (page 11)

[10] M. Aubry and B. Russell. Understanding Deep Features with Computer-Generated Imagery. In *Conference on Computer Vision and Pattern Recognition*, pages 2875–2883, 2015. (page 46, 58, 59, 62, 106, 107)

[11] D. Azinovic, T.-M. Li, A. Kaplanyan, and M. Niessner. Inverse Path Tracing for Joint Material and Lighting Estimation. In *Conference on Computer Vision and Pattern Recognition*, pages 2447–2456, 2019. (page 48)

[12] S. Bai, X. Bai, W. Liu, and F. Roli. Neural Shape Codes for 3D Model Retrieval. *Pattern Recognition Letters*, 65(1):15–21, 2015. (page 105)

[13] S. Bai, X. Bai, Z. Zhou, Z. Zhang, and L. Jan Latecki. Gift: A Real-Time and Scalable 3D Shape Search Engine. In *Conference on Computer Vision and Pattern Recognition*, pages 5023–5032, 2016. (page 46)

[14] D. H. Ballard. Generalizing the Hough Transform to Detect Arbitrary Shapes. In *Readings in Computer Vision*, pages 714–725. Elsevier, 1987. (page 43)

[15] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. Speeded-Up Robust Features (SURF). *Computer Vision and Image Understanding*, 110(3):346–359, 2018. (page 10, 41)

[16] P. Besl and N. McKay. Method for Registration of 3-D Shapes. In *Sensor Fusion IV: Control Paradigms and Data Structures*, pages 586–607. International Society for Optics and Photonics, 1992. (page 102)

[17] X. Bian, S. N. Lim, and N. Zhou. Multiscale Fully Convolutional Network with Application to Industrial Inspection. In *IEEE Winter Conference on Applications of Computer Vision*, pages 1–8, 2016. (page 2)

[18] C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995. (page 28, 33)

[19] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006. (page 27, 29, 34)

[20] V. Blanz and T. Vetter. A Morphable Model for the Synthesis of 3D Faces. In *ACM SIGGRAPH*, pages 187–194, 1999. (page 26)

[21] L. Bottou. Large-Scale Machine Learning with Stochastic Gradient Descent. In *Computational Statistics*, pages 177–186. Springer, 2010. (page 34)

[22] E. Brachmann, A. Krull, F. Michel, S. Gumhold, J. Shotton, and C. Rother. Learning 6D Object Pose Estimation using 3D Object Coordinates. In *European Conference on Computer Vision*, pages 536–551, 2014. (page 26, 44, 45, 77)

[23] E. Brachmann, A. Krull, S. Nowozin, J. Shotton, F. Michel, S. Gumhold, and C. Rother. DSAC - Differentiable RANSAC for Camera Localization. In *Conference on Computer Vision and Pattern Recognition*, pages 6684–6692, 2017. (page 146)

[24] E. Brachmann, F. Michel, A. Krull, M. Ying Yang, S. Gumhold, and C. Rother. Uncertainty-Driven 6D Pose Estimation of Objects and Scenes from a Single RGB Image. In *Conference on Computer Vision and Pattern Recognition*, pages 3364–3372, 2016. (page 44, 45, 77, 99, 100)

[25] E. Brachmann and C. Rother. Expert Sample Consensus Applied to Camera Re-Localization. In *International Conference on Computer Vision*, pages 7525–7534, 2019. (page 146)

[26] E. Brachmann and C. Rother. Neural-Guided RANSAC: Learning Where to Sample Model Hypotheses. In *International Conference on Computer Vision*, pages 4322–4331, 2019. (page 146)

[27] M. A. Branch, T. F. Coleman, and Y. Li. A Subspace, Interior, and Conjugate Gradient Method for Large-Scale Bound-Constrained Minimization Problems. *SIAM Journal on Scientific Computing*, 21(1):1–23, 1999. (page 82)

[28] L. Breiman. Random Forests. *Machine Learning*, 45(1):5–32, 2001. (page 44)

[29] R. Brunelli. *Template Matching Techniques in Computer Vision: Theory and Practice*. John Wiley & Sons, 2009. (page 38)

[30] B. Calli, A. Singh, A. Walsman, S. Srinivasa, P. Abbeel, and A. Dollar. The YCB Object and Model Set: Towards Common Benchmarks for Manipulation Research. In *International Conference on Advanced Robotics*, pages 510–517, 2015. (page 52)

[31] M. Calonder, V. Lepetit, C. Strecha, and P. Fua. BRIEF: Binary Robust Independent Elementary Features. In *European Conference on Computer Vision*, pages 778–792, 2010. (page 10)

[32] Caprile, Bruno and Torre, Vincent. Using Vanishing Points for Camera Calibration. *International Journal of Computer Vision*, 4(2):127–139, 2010. (page 48)

[33] A. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu. ShapeNet: An Information-Rich 3D Model Repository. Technical report, Stanford University — Princeton University — Toyota Technological Institute at Chicago, 2015. (page 8, 45, 52, 53, 55, 57, 58, 59, 63, 98, 108, 144)

[34] D.-Y. Chen, X.-P. Tian, Y.-T. Shen, and M. Ouhyoung. On Visual Similarity Based 3D Model Retrieval. In *Computer Graphics Forum*, pages 223–232, 2003. (page 46, 99)

[35] Q. Chen, H. Wu, and T. Wada. Camera Calibration with Two Arbitrary Coplanar Circles. In *European Conference on Computer Vision*, pages 521–532, 2004. (page 48)

[36] Y. Chen, T. Yang, X. Zhang, G. Meng, X. Xiao, and J. Sun. DetNAS: Backbone Search for Object Detection. In *Advances in Neural Information Processing Systems*, pages 6638–6648, 2019. (page 39)

[37] Chen, Zhih-Wei and Chiang, Cheng-Chin and Hsieh, Zi-Tian. Extending 3D Lucas–Kanade Tracking with Adaptive Templates for Head Pose Estimation. *Machine Vision and Applications*, 21(6):889–903, 2010. (page 26)

[38] C. B. Choy, J. Gwak, S. Savarese, and M. Chandraker. Universal Correspondence Network. In *Advances in Neural Information Processing Systems*, pages 2414–2422, 2016. (page 123, 146)

[39] R. Cipolla, T. W. Drummond, and D. P. Robertson. Camera Calibration from Vanishing Points in Image of Architectural Scenes. In *British Machine Vision Conference*, pages 382–391, 1999. (page 48)

[40] A. R. Conn, N. I. Gould, and P. L. Toint. *Trust Region Methods*. SIAM, 2000. (page 25, 82)

[41] T. F. Cootes, G. J. Edwards, and C. J. Taylor. Active Appearance Models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(6):681–685, 2001. (page 26)

[42] C. Cortes and V. Vapnik. Support-Vector Networks. *Machine Learning*, 20(3):273–297, 1995. (page 38)

[43] A. Crivellaro, M. Rad, Y. Verdie, K. Moo Yi, P. Fua, and V. Lepetit. A Novel Representation of Parts for Accurate 3D Object Detection and Tracking in Monocular Images. In *International Conference on Computer Vision*, pages 4391–4399, 2015. (page 43, 59, 60, 61)

[44] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner. ScanNet: Richly-Annotated 3D Reconstructions of Indoor Scenes. In *Conference on Computer Vision and Pattern Recognition*, pages 5828–5839, 2017. (page 52, 53, 54)

[45] N. Dalal and B. Triggs. Histograms of Oriented Gradients for Human Detection. In *Conference on Computer Vision and Pattern Recognition*, pages 886–893, 2005. (page 10, 37, 38, 42)

[46] E. R. Davies. *Computer Vision: Principles, Algorithms, Applications, Learning*. Academic Press, 2017. (page 1)

[47] D. DeTone, T. Malisiewicz, and A. Rabinovich. SuperPoint: Self-Supervised Interest Point Detection and Description. In *Conference on Computer Vision and Pattern Recognition Workshops*, pages 224–236, 2018. (page 146)

[48] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. Van Der Smagt, D. Cremers, and T. Brox. FlowNet: Learning Optical Flow with Convolutional Networks. In *Conference on Computer Vision and Pattern Recognition*, pages 2758–2766, 2015. (page 44, 119, 122)

[49] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun. CARLA: An Open Urban Driving Simulator. In *Conference on Robot Learning*, pages 1–16, 2017. (page 144)

[50] T. Drummond and R. Cipolla. Real-Time Visual Tracking of Complex Structures. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):932–946, 2002. (page 41, 44)

[51] M. Dubská, A. Herout, R. Juránek, and J. Sochor. Fully Automatic Roadside Camera Calibration for Traffic Surveillance. *IEEE Transactions on Intelligent Transportation Systems*, 16(3):1162–1171, 2015. (page 48)

[52] P. Dutre, P. Bekaert, and K. Bala. *Advanced Global Illumination*. CRC Press, 2018. (page 9, 23, 53, 144)

[53] D. S. Ebert, F. K. Musgrave, D. Peachey, K. Perlin, and S. Worley. *Texturing & Modeling: A Procedural Approach*. Morgan Kaufmann, 2003. (page 24)

[54] T. Elsken, J. H. Metzen, and F. Hutter. Neural Architecture Search: A Survey. *arXiv:1808.05377*, 2018. (page 35, 39, 145)

[55] C. Esteves, C. Allen-Blanchette, A. Makadia, and K. Daniilidis. Learning SO(3) Equivariant Representations with Spherical CNNs. In *European Conference on Computer Vision*, pages 52–68, 2018. (page 46, 145)

[56] C. Esteves, A. Sud, Z. Luo, K. Daniilidis, and A. Makadia. Cross-Domain 3D Equivariant Image Embeddings. In *International Conference on Machine Learning*, pages 1812–1822, 2019. (page 43, 145)

[57] M. Everingham, L. Van Gool, C. Williams, J. Winn, and A. Zisserman. The Pascal Visual Object Classes (VOC) Challenge. *International Journal of Computer Vision*, 88(2):303–338, 2010. (page 54)

[58] H. Fan, H. Su, and L. Guibas. A Point Set Generation Network for 3D Object Reconstruction from a Single Image. In *Conference on Computer Vision and Pattern Recognition*, pages 605–613, 2017. (page 45)

[59] O. D. Faugeras. *Three-Dimensional Computer Vision: A Geometric Viewpoint*. MIT Press, 1993. (page 21, 47)

[60] L. Fei-Fei and P. Perona. A Bayesian Hierarchical Model for Learning Natural Scene Categories. In *Conference on Computer Vision and Pattern Recognition*, pages 524–531, 2005. (page 10)

[61] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. Object Detection with Discriminatively Trained Part-Based Models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1627–1645, 2009. (page 10, 37, 38, 42)

[62] J. Feng, Y. Wang, and S.-F. Chang. 3D Shape Retrieval Using Single Depth Image from Low-Cost Sensors. In *IEEE Winter Conference on Applications of Computer Vision*, pages 1–9, 2016. (page 62)

[63] M. A. Fischler and R. C. Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Communications of the ACM*, 24(6):381–395, 1981. (page 10, 25)

[64] J. D. Foley, A. Van Dam, S. K. Feiner, J. F. Hughes, and E. Angel. *Computer Graphics: Principles and Practice*. Prentice Hall Professional Technical Reference, 2002. (page 22, 23)

[65] D. A. Forsyth and J. Ponce. *Computer Vision: A Modern Approach*. Prentice Hall Professional Technical Reference, 2002. (page 7, 19)

[66] K. Frankish and W. M. Ramsey. *The Cambridge Handbook of Artificial Intelligence*. Cambridge University Press, 2014. (page 1)

[67] Y. Freund and R. E. Schapire. A Desicion-Theoretic Generalization of On-Line Learning and an Application to Boosting. In *European Conference on Computational Learning Theory*, pages 23–37, 1995. (page 38)

[68] X.-S. Gao, X.-R. Hou, J. Tang, and H.-F. Cheng. Complete Solution Classification for the Perspective-Three-Point Problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(8):930–943, 2003. (page 10, 25)

[69] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision Meets Robotics: The KITTI Dataset. *International Journal of Robotics Research*, 32(11):1231–1237, 2013. (page 52)

[70] K. Genova, F. Cole, A. Maschinot, A. Sarna, D. Vlasic, and W. T. Freeman. Unsupervised Training for 3D Morphable Model Regression. In *Conference on Computer Vision and Pattern Recognition*, pages 8377–8386, 2018. (page 48, 49)

[71] R. Girdhar, D. Fouhey, M. Rodriguez, and A. Gupta. Learning a Predictable and Generative Vector Representation for Objects. In *European Conference on Computer Vision*, pages 484–499, 2016. (page 47, 53, 62, 98, 99)

[72] R. Girshick. Fast R-CNN. In *International Conference on Computer Vision*, pages 1440–1448, 2015. (page 31, 39)

[73] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. In *Conference on Computer Vision and Pattern Recognition*, pages 580–587, 2014. (page 39)

[74] R. Girshick, P. F. Felzenszwalb, and D. A. Mcallester. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. In *Advances in Neural Information Processing Systems*, pages 442–450, 2011. (page 38)

[75] G. Gkioxari, J. Malik, and J. Johnson. Mesh R-CNN. In *Conference on Computer Vision and Pattern Recognition*, pages 9785–9795, 2019. (page 11, 39, 45)

[76] X. Glorot and Y. Bengio. Understanding the Difficulty of Training Deep Feedforward Neural Networks. In *International Conference on Artificial Intelligence and Statistics*, pages 249–256, 2010. (page 33)

[77] X. Glorot, A. Bordes, and Y. Bengio. Deep Sparse Rectifier Neural Networks. In *International Conference on Artificial Intelligence and Statistics*, pages 315–323, 2011. (page 29)

[78] C. Godard, O. Mac Aodha, and G. J. Brostow. Unsupervised Monocular Depth Estimation with Left-Right Consistency. In *Conference on Computer Vision and Pattern Recognition*, pages 270–279, 2017. (page 144)

[79] C. Godard, O. Mac Aodha, M. Firman, and G. J. Brostow. Digging into Self-Supervised Monocular Depth Estimation. In *International Conference on Computer Vision*, pages 3828–3838, 2019. (page 144)

[80] G. H. Golub and C. Reinsch. Singular Value Decomposition and Least Squares Solutions. In *Linear Algebra*, pages 134–151. Springer, 1971. (page 24)

[81] G. H. Golub and C. F. Van Loan. *Matrix Computation*. The John Hopkins University Press, 1996. (page 24)

[82] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. (page 2, 10, 27, 28, 31, 32, 33, 35)

[83] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative Adversarial Nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680, 2014. (page 2)

[84] I. Goodfellow, J. Shlens, and C. Szegedy. Explaining and Harnessing Adversarial Examples. In *International Conference for Learning Representations*, pages 1–8, 2015. (page 2)

[85] A. Grabner, P. M. Roth, and V. Lepetit. 3D Pose Estimation and 3D Model Retrieval for Objects in the Wild. In *Conference on Computer Vision and Pattern Recognition*, pages 3022–3031, 2018. (page 46, 76, 77, 80, 85, 99, 105, 106, 107, 118, 145)

[86] A. Grabner, P. M. Roth, and V. Lepetit. GP^{2}C: Geometric Projection Parameter Consensus for Joint 3D Pose and Focal Length Estimation in the Wild. In *International Conference on Computer Vision*, pages 2222–2231, 2019. (page 8, 9, 37, 99, 118, 126, 128, 129, 130, 131, 133, 134, 135, 136, 138, 139, 140)

[87] A. Grabner, P. M. Roth, and V. Lepetit. Location Field Descriptors: Single Image 3D Model Retrieval in the Wild. In *International Conference on 3D Vision*, pages 583–593, 2019. (page 37, 119, 121, 139)

[88] B. Graham. Fractional Max-Pooling. *arXiv:1412.6071*, 2014. (page 34)

[89] F. S. Grassia. Practical Parameterization of Rotations using the Exponential Map. *Journal of Graphics Tools*, 3(3):29–48, 1998. (page 22)

[90] T. Groueix, M. Fisher, V. G. Kim, B. C. Russell, and M. Aubry. A Papier-Mâché Approach to Learning 3D Surface Generation. In *Conference on Computer Vision and Pattern Recognition*, pages 216–224, 2018. (page 45)

[91] A. Gruen and T. S. Huang. *Calibration and Orientation of Cameras in Computer Vision*. Springer Science & Business Media, 2013. (page 25)

[92] S. Gur and L. Wolf. Single Image Depth Estimation Trained via Depth from Defocus Cues. In *Conference on Computer Vision and Pattern Recognition*, pages 7683–7692, 2019. (page 144)

[93] C. Harris and C. Stennett. RAPID - A Video Rate Object Tracker. In *British Machine Vision Conference*, pages 73–77, 1990. (page 41, 44)

[94] C. Harris and M. Stephens. A Combined Corner and Edge Detector. In *Alvey Vision Conference*, pages 147–151, 1988. (page 10, 41)

[95] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2003. (page 9, 10, 18, 19, 24, 26, 41, 45, 47, 123)

[96] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask R-CNN. In *International Conference on Computer Vision*, pages 2980–2988, 2017. (page 2, 31, 39, 43, 44, 75, 78, 79, 81, 85, 90, 101, 106, 107, 119, 128, 129, 131, 132, 143, 146)

[97] K. He, X. Zhang, S. Ren, and J. Sun. Delving Deep Into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. In *Conference on Computer Vision and Pattern Recognition*, pages 1026–1034, 2015. (page 2, 33)

[98] K. He, X. Zhang, S. Ren, and J. Sun. Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(9):1904–1916, 2015. (page 39)

[99] K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. In *Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016. (page 7, 28, 31, 39, 44, 61, 66, 79, 102, 104, 106, 122, 127, 132, 145)

[100] K. He, X. Zhang, S. Ren, and J. Sun. Identity Mappings in Deep Residual Networks. In *European Conference on Computer Vision*, pages 630–645, 2016. (page 39, 61, 66, 79, 102, 104, 106, 122, 127)

[101] X. He, Y. Zhou, Z. Zhou, S. Bai, and X. Bai. Triplet-Center Loss for Multi-View 3D Object Retrieval. In *Conference on Computer Vision and Pattern Recognition*, pages 1945–1954, 2018. (page 31, 103, 110, 113)

[102] P. Henderson and V. Ferrari. Learning to Generate and Reconstruct 3D Meshes with only 2D Supervision. In *British Machine Vision Conference*, pages 139:1–139:13, 2018. (page 48, 49, 119, 146)

[103] J. A. Hesch and S. I. Roumeliotis. A Direct Least-Squares (DLS) Method for PnP. In *International Conference on Computer Vision*, pages 383–390, 2011. (page 82)

[104] S. Hinterstoisser, C. Cagniart, S. Ilic, P. Sturm, N. Navab, P. Fua, and V. Lepetit. Gradient Response Maps for Real-Time Detection of Textureless Objects. *IEEE*

Transactions on Pattern Analysis and Machine Intelligence, 34(5):876–888, 2011. (page 42, 52, 54)

[105] S. Hinterstoisser, V. Lepetit, S. Ilic, S. Holzer, G. Bradski, K. Konolige, and N. Navab. Model Based Training, Detection and Pose Estimation of Texture-Less 3D Objects in Heavily Cluttered Scenes. In *Asian Conference on Computer Vision*, pages 548–562, 2012. (page 42, 84, 127)

[106] G. E. Hinton, T. J. Sejnowski, and T. A. Poggio. *Unsupervised Learning: Foundations of Neural Computation*. MIT Press, 1999. (page 144)

[107] S. Hochreiter and J. Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 1997. (page 28, 31, 145)

[108] T. Hodan, P. Haluza, Š. Obdržálek, J. Matas, M. Lourakis, and X. Zabulis. T-LESS: An RGB-D Dataset for 6D Pose Estimation of Texture-Less Objects. In *IEEE Winter Conference on Applications of Computer Vision*, pages 880–888, 2017. (page 8, 52)

[109] T. Hodaň, J. Matas, and Š. Obdržálek. On Evaluation of 6D Object Pose Estimation. In *European Conference on Computer Vision*, pages 606–619, 2016. (page 83, 84, 126, 127)

[110] T. Hodan, F. Michel, E. Brachmann, W. Kehl, A. Glent Buch, D. Kraft, B. Drost, J. Vidal, S. Ihrke, X. Zabulis, C. Sahin, F. Manhardt, F. Tombari, T.-K. Kim, J. Matas, and C. Rother. BOP: Benchmark for 6D Object Pose Estimation. In *European Conference on Computer Vision*, pages 19–34, 2018. (page 52)

[111] Y. Hold-Geoffroy, K. Sunkavalli, J. Eisenmann, M. Fisher, E. Gambaretto, S. Hadap, and J.-F. Lalonde. A Perceptual Measure for Deep Single Image Camera Calibration. In *Conference on Computer Vision and Pattern Recognition*, pages 2354–2363, 2018. (page 48)

[112] G. C. Holst. *CCD Arrays, Cameras, and Displays*. Citeseer, 1998. (page 9, 19)

[113] H. S. Hong and M. J. Chung. 3D Pose and Camera Parameter Tracking Algorithm based on Lucas-Kanade Image Alignment Algorithm. In *International Conference on Control, Automation and Systems*, pages 548–551, 2007. (page 26)

[114] K. Hornik. Approximation Capabilities of Multilayer Feedforward Networks. *Neural Networks*, 4(2):251–257, 1991. (page 31)

[115] G. Huang, Z. Liu, L. Van Der Maaten, and K. Weinberger. Densely Connected Convolutional Networks. In *Conference on Computer Vision and Pattern Recognition*, pages 2261–2269, 2017. (page 102, 106)

[116] S. Huang, S. Qi, Y. Zhu, Y. Xiao, Y. Xu, and S.-C. Zhu. Holistic 3D Scene Parsing and Reconstruction from a Single RGB Image. In *European Conference on Computer Vision*, pages 187–203, 2018. (page 2, 47)

[117] D. H. Hubel. Exploration of the Primary Visual Cortex. *Nature*, 299(5883):515–524, 1982. (page 30)

[118] P. Huber. Robust Estimation of a Location Parameter. *The Annals of Mathematical Statistics*, 35(1):73–101, 1964. (page 25, 32, 61, 79, 81, 102, 103)

[119] M. Hueting, P. Reddy, E. Yumer, V. Kim, N. Carr, and N. Mitra. SeeThrough: Finding Objects in Heavily Occluded Indoor Scene Images. In *International Conference on 3D Vision*, pages 267–276, 2018. (page 8, 47, 52, 53)

[120] E. Insafutdinov and A. Dosovitskiy. Unsupervised Learning of Shape and Pose with Differentiable Point Clouds. In *Advances in Neural Information Processing Systems*, pages 2802–2812, 2018. (page 144)

[121] S. Ioffe and C. Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *International Conference on Machine Learning*, pages 448–456, 2015. (page 31, 34)

[122] A. G. Ivakhnenko and V. G. Lapa. *Cybernetics and Forecasting Techniques*. American Elsevier Publishing Company, 1967. (page 28)

[123] H. Izadinia, Q. Shan, and S. Seitz. IM2CAD. In *Conference on Computer Vision and Pattern Recognition*, 2017. (page 2, 47, 58, 59, 62, 99, 102)

[124] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu. Spatial Transformer Networks. In *Advances in Neural Information Processing Systems*, pages 2017–2025, 2015. (page 146)

[125] O. H. Jafari, S. K. Mustikovela, K. Pertsch, E. Brachmann, and C. Rother. iPose: Instance-Aware 6D Pose Estimation of Partly Occluded Objects. In *Asian Conference on Computer Vision*, pages 477–492, 2018. (page 39, 44, 77, 99, 100)

[126] B. Jähne, H. Haussecker, and P. Geissler. *Handbook of Computer Vision and Applications*. Citeseer, 1999. (page 19)

[127] A. K. Jain. *Fundamentals of Digital Image Processing*. Prentice Hall Professional Technical Reference, 1989. (page 19, 38)

[128] Japan Electronic and Information Technology Industries Association. Exchangeable Image File Format for Digital Still Cameras. 2010. (page 9)

[129] H. W. Jensen. *Realistic Image Synthesis using Photon Mapping*. CRC Press, 2001. (page 23, 53, 144)

[130] Q. Ji, M. S. Costa, R. M. Haralick, and L. G. Shapiro. An Integrated Linear Technique for Pose Estimation from Different Geometric Features. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(5):705–733, 1999. (page 41)

[131] F. Jurie and M. Dhome. Hyperplane Approximation for Template Matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):996–1000, 2002. (page 42)

[132] A. Kanazawa, S. Tulsiani, A. A. Efros, and J. Malik. Learning Category-Specific Mesh Reconstruction from Image Collections. In *European Conference on Computer Vision*, pages 371–386, 2018. (page 48)

[133] A. Kanezaki, Y. Matsushita, and Y. Nishida. RotationNet: Joint Object Categorization and Pose Estimation using Multiviews from Unsupervised Viewpoints. In *Conference on Computer Vision and Pattern Recognition*, pages 5010–5019, 2018. (page 46, 144)

[134] H. Kato, Y. Ushiku, and T. Harada. Neural 3D Mesh Renderer. In *Conference on Computer Vision and Pattern Recognition*, pages 3907–3916, 2018. (page 26, 45, 48, 49, 118, 119, 125, 127, 128, 129, 130, 131, 132, 133, 134, 135, 138, 139, 146)

[135] T. Ke and S. I. Roumeliotis. An Efficient Algebraic Solution to the Perspective-Three-Point Problem. In *Conference on Computer Vision and Pattern Recognition*, pages 7225–7233, 2017. (page 25)

[136] W. Kehl, F. Manhardt, F. Tombari, S. Ilic, and N. Navab. SSD-6D: Making RGB-Based 3D Detection and 6D Pose Estimation Great Again. In *International Conference on Computer Vision*, pages 1530–1538, 2017. (page 43, 122)

[137] M. Keller, Z. Chen, F. Maffra, P. Schmuck, and M. Chli. Learning Deep Descriptors with Scale-Aware Triplet Networks. In *Conference on Computer Vision and Pattern Recognition*, pages 2762–2770, 2018. (page 103)

[138] D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. *arXiv:1412.6980*, 2014. (page 34, 64, 126, 127, 128, 132)

[139] D. P. Kingma and J. Ba. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. *arXiv:1406.1078*, 2014. (page 31)

[140] A. Kirillov, R. Girshick, K. He, and P. Dollár. Panoptic Feature Pyramid Networks. In *Conference on Computer Vision and Pattern Recognition*, pages 6399–6408, 2019. (page 11)

[141] D. Kirk. NVIDIA Cuda Software and GPU Parallel Computing Architecture. In *International Symposium on Memory Management*, pages 103–104, 2007. (page 28)

[142] R. Klokov and V. Lempitsky. Escape from Cells: Deep KD-Networks for the Recognition of 3D Point Cloud Models. In *International Conference on Computer Vision*, pages 863–872, 2017. (page 46)

[143] L. Kneip, H. Li, and Y. Seo. UPnP: An Optimal O(n) Solution to the Absolute Pose Problem with Universal Applicability. In *European Conference on Computer Vision*, pages 127–142, 2014. (page 25)

[144] V. A. Knyaz, V. V. Kniaz, and F. Remondino. Image-to-Voxel Model Translation with Conditional Adversarial Networks. In *European Conference on Computer Vision Workshops*, pages 1–14, 2018. (page 45)

[145] A. Krizhevsky. Learning Multiple Layers of Features from Tiny Images. pages 1–58, 2009. (page 34)

[146] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012. (page 2, 11, 28, 29, 39, 48, 145)

[147] A. Krogh and J. A. Hertz. A Simple Weight Decay Can Improve Generalization. In *Advances in Neural Information Processing Systems*, pages 950–9575, 1992. (page 34, 61)

[148] A. Krull, E. Brachmann, S. Nowozin, F. Michel, J. Shotton, and C. Rother. PoseAgent: Budget-Constrained 6D Object Pose Estimation via Reinforcement Learning. In *Conference on Computer Vision and Pattern Recognition*, pages 6702–6710, 2017. (page 145)

[149] B. Kulis. Metric Learning: A Survey. *Foundations and Trends in Machine Learning*, 5(4):287–364, 2013. (page 46)

[150] T. D. Kulkarni, W. F. Whitney, P. Kohli, and J. Tenenbaum. Deep Convolutional Inverse Graphics Network. In *Advances in Neural Information Processing Systems*, pages 2539–2547, 2015. (page 17)

[151] A. Kundu, Y. Li, and J. M. Rehg. 3D-RCNN: Instance-Level 3D Object Reconstruction via Render-and-Compare. In *Conference on Computer Vision and Pattern Recognition*, pages 3559–3568, 2018. (page 8, 43, 48, 82)

[152] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-Based Learning Applied to Document Recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. (page 30)

[153] T. Lee, Y.-L. Lin, H. Y. Chiang, M.-W. Chiu, W. Hsu, and P. Huang. Cross-Domain Image-Based 3D Shape Retrieval by View Sequence Learning. In *International Conference on 3D Vision*, pages 258–266, 2018. (page 47, 99)

[154] V. Lepetit, P. Lagger, and P. Fua. Randomized Trees for Real-Time Keypoint Recognition. In *Conference on Computer Vision and Pattern Recognition*, pages 775–781, 2005. (page 41)

[155] V. Lepetit, F. Moreno-Noguer, and P. Fua. EPnP: An Accurate $O(n)$ Solution to the PnP Problem. *International Journal of Computer Vision*, 81(2):155–166, 2009. (page 10, 25, 26, 82, 94)

[156] C. Li, J. Bai, and G. D. Hager. A Unified Framework for Multi-View Multi-Class Object Pose Estimation. In *European Conference on Computer Vision*, pages 1–16, 2018. (page 42, 54, 77, 85)

[157] T.-M. Li, M. Aittala, F. Durand, and J. Lehtinen. Differentiable Monte Carlo Ray Tracing through Edge Sampling. In *ACM SIGGRAPH Asia*, pages 222:1–222:11, 2018. (page 48)

[158] Y. Li, H. Su, C. R. Qi, N. Fish, D. Cohen-Or, and L. Guibas. Joint Embeddings of Shapes and Images via CNN Image Purification. *ACM Transactions on Graphics*, 34(6):234, 2015. (page 47, 59, 62, 99)

[159] Y. Li, G. Wang, X. Ji, Y. Xiang, and D. Fox. DeepIM: Deep Iterative Matching for 6D Pose Estimation. In *European Conference on Computer Vision*, pages 683–698, 2018. (page 44, 118, 119, 126, 139)

[160] Z. Li, T. Dekel, F. Cole, R. Tucker, N. Snavely, C. Liu, and W. T. Freeman. Learning the Depths of Moving People by Watching Frozen People. In *Conference on Computer Vision and Pattern Recognition*, pages 4521–4530, 2019. (page 144)

[161] J. J. Lim, A. Khosla, and A. Torralba. FPM: Fine Pose Parts-Based Model with 3D CAD Models. In *European Conference on Computer Vision*, pages 478–493, 2014. (page 42)

[162] J. J. Lim, H. Pirsiavash, and A. Torralba. Parsing IKEA Objects: Fine Pose Estimation. In *International Conference on Computer Vision*, pages 2992–2999, 2013. (page 42, 52, 55)

[163] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. J. Belongie. Feature Pyramid Networks for Object Detection. In *Conference on Computer Vision and Pattern Recognition*, pages 2117–2125, 2017. (page 39, 79, 106)

[164] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal Loss for Dense Object Detection. In *International Conference on Computer Vision*, pages 2980–2988, 2017. (page 39)

[165] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: Common Objects in Context. In *European Conference on Computer Vision*, pages 740–755, 2014. (page 2, 9, 28, 79, 85, 106, 107, 128, 143)

[166] L. Liu, J. Lu, C. Xu, Q. Tian, and J. Zhou. Deep Fitting Degree Scoring Network for Monocular 3D Object Detection. In *Conference on Computer Vision and Pattern Recognition*, pages 1057–1066, 2019. (page 44)

[167] S. Liu, T. Li, W. Chen, and H. Li. Soft Rasterizer: A Differentiable Renderer for Image-Based 3D Reasoning. In *International Conference on Computer Vision*, pages 7708–7717, 2019. (page 48, 146)

[168] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia. Path Aggregation Network for Instance Segmentation. In *Conference on Computer Vision and Pattern Recognition*, pages 8759–8768, 2018. (page 39)

[169] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. SSD: Single Shot Multibox Detector. In *European Conference on Computer Vision*, pages 21–37, 2016. (page 39, 43)

[170] J. Long, E. Shelhamer, and T. Darrell. Fully Convolutional Networks for Semantic Segmentation. In *Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015. (page 7)

[171] M. M. Loper and M. J. Black. OpenDR: An Approximate Differentiable Renderer. In *European Conference on Computer Vision*, pages 154–169, 2014. (page 26, 45, 48, 49, 119, 121, 124, 128, 139, 146)

[172] M. Lopez, R. Mari, P. Gargallo, Y. Kuang, J. Gonzalez-Jimenez, and G. Haro. Deep Single Image Camera Calibration with Radial Distortion. In *Conference on Computer Vision and Pattern Recognition*, pages 11817–11825, 2019. (page 48)

[173] D. Lowe. Three-Dimensional Object Recognition from Single Two-Dimensional Images. *Artificial Intelligence*, 31(3):355–395, 1987. (page 17)

[174] D. Lowe. Fitting Parameterized Three-Dimensional Models to Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(5):441–450, 1991. (page 41, 44)

[175] D. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004. (page 10, 41)

[176] C.-P. Lu, G. D. Hager, and E. Mjolsness. Fast and Globally Convergent Pose Estimation from Video Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(6):610–622, 2000. (page 26)

[177] B. D. Lucas and T. Kanade. An Iterative Image Registration Technique with an Application to Stereo Vision. In *International Joint Conference on Artificial Intelligence*, pages 674–679, 1981. (page 26)

[178] Q.-T. Luong and O. D. Faugeras. The Fundamental Matrix: Theory, Algorithms, and Stability Analysis. *International Journal of Computer Vision*, 17(1):43–75, 1996. (page 26)

[179] S. Mahendran, H. Ali, and R. Vidal. A Mixed Classification-Regression Framework for 3D Pose Estimation from 2D Images. In *British Machine Vision Conference*, pages 238:1–238:12, 2018. (page 42)

[180] P. Mandikal, K. L. Navaneet, M. Agarwal, and R. V. Babu. 3D-LMNet: Latent Embedding Matching for Accurate and Diverse 3D Point Cloud Reconstruction from a Single Image. In *British Machine Vision Conference*, pages 55:1–55:12, 2018. (page 45)

[181] F. Manhardt, W. Kehl, N. Navab, and F. Tombari. Deep Model-Based 6D Pose Refinement in RGB. In *European Conference on Computer Vision*, pages 800–815, 2018. (page 44, 139)

[182] D. Marr. *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information*. CumInCAD, 1982. (page 1, 38)

[183] S. Marschner and P. Shirley. *Fundamentals of Computer Graphics*. CRC Press, 2015. (page 18, 23)

[184] F. Massa, R. Marlet, and M. Aubry. Crafting a Multi-Task CNN for Viewpoint Estimation. In *British Machine Vision Conference*, pages 91:1–91:12, 2016. (page 42)

[185] F. Massa, B. Russell, and M. Aubry. Deep Exemplar 2D-3D Detection by Adapting from Real to Rendered Views. In *Conference on Computer Vision and Pattern Recognition*, pages 6024–6033, 2016. (page 10, 46, 53, 58, 62, 99, 102, 144)

[186] G. J. McLachlan. *Discriminant Analysis and Statistical Pattern Recognition*. John Wiley & Sons, 2004. (page 42)

[187] T. M. Mitchell. *Machine Learning*. WCB–McGraw–Hill, 1997. (page 29)

[188] M. Mohri, A. Rostamizadeh, and A. Talwalkar. *Foundations of Machine Learning*. MIT Press, 2018. (page 32)

[189] J. J. Moré. The Levenberg-Marquardt Algorithm: Implementation and Theory. In *Numerical Analysis*, pages 105–116, 1978. (page 25, 82)

[190] R. Mottaghi, Y. Xiang, and S. Savarese. A Coarse-To-Fine Model for 3D Pose Estimation and Sub-Category Recognition. In *Conference on Computer Vision and Pattern Recognition*, pages 418–426, 2015. (page 42, 46, 76, 77)

[191] A. Mousavian, D. Anguelov, J. Flynn, and J. Kosecka. 3D Bounding Box Estimation Using Deep Learning and Geometry. In *Conference on Computer Vision and Pattern Recognition*, pages 7074–7082, 2017. (page 42, 60, 64, 65, 67, 76, 77, 85, 118)

[192] G. Nakano. A Versatile Approach for Solving PnP, PnPf, and PnPfr Problems. In *European Conference on Computer Vision*, pages 338–352, 2016. (page 26, 48, 81, 82, 83)

[193] K. L. Navaneet, P. Mandikal, M. Agarwal, and V. Babu. CAPNet: Continuous Approximation Projection for 3D Point Cloud Reconstruction using 2D Supervision. In *AAAI Conference on Artificial Intelligence*, 2019. (page 45)

[194] A. Nguyen, J. Yosinski, and J. Clune. Deep Neural Networks are Easily Fooled: High Confidence Predictions for Unrecognizable Images. In *Conference on Computer Vision and Pattern Recognition*, pages 427–436, 2015. (page 2)

[195] T. H. Nguyen-Phuoc, C. Li, S. Balaban, and Y. Yang. RenderNet: A Deep Convolutional Network for Differentiable Rendering from 3D Shapes. In *Advances in Neural Information Processing Systems*, pages 7891–7901, 2018. (page 48)

[196] F. Nooruddin and G. Turk. Simplification and Repair of Polygonal Models Using Volumetric Techniques. *IEEE Transactions on Visualization and Computer Graphics*, 9(2):191–205, 2003. (page 105)

[197] C. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall. Activation Functions: Comparison of Trends in Practice and Research for Deep Learning. *arXiv:1811.03378*, 2018. (page 29)

[198] M. Oberweger, M. Rad, and V. Lepetit. Making Deep Heatmaps Robust to Partial Occlusions for 3D Object Pose Estimation. In *European Conference on Computer Vision*, pages 119–134, 2018. (page 8, 30, 43, 139)

[199] H. Oh Song, Y. Xiang, S. Jegelka, and S. Savarese. Deep Metric Learning via Lifted Structured Feature Embedding. In *Conference on Computer Vision and Pattern Recognition*, pages 4004–4012, 2016. (page 47)

[200] A. Palazzi, L. Bergamini, S. Calderara, and R. Cucchiara. End-to-End 6-DoF Object Pose Estimation through Differentiable Rasterization. In *European Conference on Computer Vision Workshops*, pages 1–14, 2018. (page 45, 48, 118, 119, 139)

[201] C. P. Papageorgiou, M. Oren, and T. Poggio. A General Framework for Object Detection. In *International Conference on Computer Vision*, pages 555–562, 1998. (page 38)

[202] Y. Park, V. Lepetit, and W. Woo. Multiple 3D Object Tracking for Augmented Reality. In *International Symposium on Mixed and Augmented Reality*, pages 117–120, 2008. (page 10, 41)

[203] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems*, pages 8024–8035. Curran Associates Inc., 2019. (page 28)

[204] G. Pavlakos, X. Zhou, A. Chan, K. Derpanis, and K. Daniilidis. 6-DoF Object Pose from Semantic Keypoints. In *International Conference on Robotics and Automation*, pages 2011–2018, 2017. (page 43, 60, 64, 67)

[205] N. Payet and S. Todorovic. From Contours to 3D Object Detection and Pose Estimation. In *International Conference on Computer Vision*, pages 983–990, 2011. (page 42)

[206] A. Penate-Sanchez, J. Andrade-Cetto, and F. Moreno-Noguer. Exhaustive Linearization for Robust Camera Pose and Focal Length Estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(10):2387–2400, 2013. (page 48, 82, 84)

[207] S. Peng, Y. Liu, Q. Huang, X. Zhou, and H. Bao. 3D Object Class Detection in the Wild. In *Conference on Computer Vision and Pattern Recognition*, pages 4561–4570, 2019. (page 43)

[208] B. Pepik, M. Stark, P. Gehler, T. Ritschel, and B. Schiele. 3D Object Class Detection in the Wild. In *Conference on Computer Vision and Pattern Recognition Workshops*, pages 1–10, 2015. (page 43)

[209] B. Pepik, M. Stark, P. Gehler, and B. Schiele. Teaching 3D Geometry to Deformable Part Models. In *Conference on Computer Vision and Pattern Recognition*, pages 3362–3369, 2012. (page 42)

[210] V. A. Prisacariu and I. D. Reid. PWP3D: Real-Time Segmentation and Tracking of 3D Objects. *International Journal of Computer Vision*, 98(3):335–354, 2012. (page 44)

[211] C. Qi, H. Su, M. Nießner, A. Dai, M. Yan, and L. Guibas. Volumetric and Multi-View CNNs for Object Classification on 3D Data. In *Conference on Computer Vision and Pattern Recognition*, pages 5648–5656, 2016. (page 46)

[212] C. R. Qi, H. Su, K. Mo, and L. Guibas. Pointnet: Deep Learning on Point Sets for 3D Classification and Segmentation. In *Conference on Computer Vision and Pattern Recognition*, pages 652–660, 2017. (page 46)

[213] C. R. Qi, L. Yi, H. Su, and L. Guibas. Pointnet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. In *Advances in Neural Information Processing Systems*, pages 5099–5108, 2017. (page 46)

[214] N. Qian. On the Momentum Term in Gradient Descent Learning Algorithms. *Neural Networks*, 12(1):145–151, 1999. (page 34)

[215] K. G. Quach, C. N. Duong, K. Luu, and T. D. Bui. Robust Deep Appearance Models. In *International Conference on Pattern Recognition*, pages 390–395, 2016. (page 26)

[216] L. Quan and Z. Lan. Linear N-Point Camera Pose Determination. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(8):774–780, 1999. (page 10, 25)

[217] M. Rad and V. Lepetit. BB8: A Scalable, Accurate, Robust to Partial Occlusion Method for Predicting the 3D Poses of Challenging Objects Without Using Depth. In *International Conference on Computer Vision*, pages 3828–3836, 2017. (page 26, 33, 43, 44, 59, 61, 77, 80, 118, 122, 145)

[218] M. Rad, M. Oberweger, and V. Lepetit. Feature Mapping for Learning Fast and Accurate 3D Pose Inference from Synthetic Images. In *Conference on Computer Vision and Pattern Recognition*, pages 4663–4672, 2018. (page 10, 53, 104, 113)

[219] R. Ranftl, K. Lasinger, D. Hafner, K. Schindler, and V. Koltun. Towards Robust Monocular Depth Estimation: Mixing Datasets for Zero-Shot Cross-Dataset Transfer. *arXiv:1907.01341*, 2019. (page 144)

[220] S. Ravela, B. Draper, J. J. Lim, and R. Weiss. Adaptive Tracking and Model Registration Across Distinct Aspects. In *International Conference on Intelligent Robots and Systems*, pages 174–180, 1995. (page 41)

[221] M. Raynal. *Algorithms for Mutual Exclusion*. MIT Press, 1986. (page 23)

[222] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You Only Look Once: Unified, Real-Time Object Detection. In *Conference on Computer Vision and Pattern Recognition*, pages 779–788, 2016. (page 39, 43)

[223] J. Redmon and A. Farhadi. YOLO9000: Better, Faster, Stronger. In *Conference on Computer Vision and Pattern Recognition*, pages 7263–7271, 2017. (page 39)

[224] J. Redmon and A. Farhadi. YOLOv3: An Incremental Improvement. *arXiv:1804.02767*, 2018. (page 39)

[225] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In *Advances in Neural Information Processing Systems*, pages 91–99, 2015. (page 7, 30, 33, 39, 43, 60, 76, 79, 81, 101)

[226] H. Rhodin, M. Salzmann, and P. Fua. Unsupervised Geometry-Aware Representation for 3D Human Pose Estimation. In *European Conference on Computer Vision*, pages 750–767, 2018. (page 144)

[227] S. R. Richter, Z. Hayder, and V. Koltun. Playing for Benchmarks. In *International Conference on Computer Vision*, pages 2213–2222, 2017. (page 52, 53, 144)

[228] S. R. Richter, V. Vineet, S. Roth, and V. Koltun. Playing for Data: Ground Truth from Computer Games. In *European Conference on Computer Vision*, pages 102–118, 2016. (page 144)

[229] B. M. H. Romeny. *Front-End Vision and Multi-Scale Image Analysis: Multi-Scale Computer Vision Theory and Applications, written in Mathematica*. Springer Science & Business Media, 2008. (page 1)

[230] E. Rosten and T. Drummond. Machine Learning for High-Speed Corner Detection. In *European Conference on Computer Vision*, pages 430–443, 2006. (page 10)

[231] F. E. Ruiz, P. S. Pérez, and B. I. Bonev. *Information Theory in Computer Vision and Pattern Recognition*. Springer Science & Business Media, 2009. (page 7)

[232] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning Representations by Back-Propagating Errors. *Nature*, 323(6088):533–536, 1986. (page 33)

[233] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015. (page 2, 28, 34, 39, 42, 43, 46, 54, 64, 143, 145)

[234] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall Professional Technical Reference, 2010. (page 31)

[235] T. Salimans and D. P. Kingma. Weight Normalization: A Simple Reparameterization to Accelerate Training of Deep Neural Networks. In *Advances in Neural Information Processing Systems*, pages 901–909, 2016. (page 34)

[236] J. Sanders and E. Kandrot. *CUDA by Example: An Introduction to General-Purpose GPU Programming*. Addison-Wesley Professional, 2010. (page 24)

[237] C. Santa Cruz and J. Dorronsoro. A Nonlinear Discriminant Algorithm for Feature Extraction and Data Classification. *IEEE Transactions on Neural Networks*, 9(6):1370–1376, 1998. (page 103)

[238] T. Sattler, Q. Zhou, M. Pollefeys, and L. Leal-Taixe. Understanding The Limitations of CNN-Based Absolute Camera Pose Regression. In *Conference on Computer Vision and Pattern Recognition*, pages 3302–3312, 2019. (page 11)

[239] M. Savva, A. Kadian, O. Maksymets, Y. Zhao, E. Wijmans, B. Jain, J. Straub, J. Liu, V. Koltun, and J. Malik. Habitat: A Platform for Embodied AI Research. In *International Conference on Computer Vision*, pages 9339–9347, 2019. (page 144)

[240] M. Savva, F. Yu, H. Su, et al. SHREC'16 Track Large-Scale 3D Shape Retrieval from ShapeNet Core55. In *Eurographics Workshop on 3D Object Retrieval*, 2016. (page 46)

[241] M. Savva, F. Yu, H. Su, et al. SHREC'17 Track Large-Scale 3D Shape Retrieval from ShapeNet Core55. In *Eurographics Workshop on 3D Object Retrieval*, 2017. (page 46)

[242] A. M. Saxe, J. L. Mcclelland, and S. Ganguli. Exact Solutions to the Nonlinear Dynamics of Learning in Deep Linear Neural Network. In *International Conference for Learning Representations*, pages 3302–3312, 2014. (page 33)

[243] C. Schmaltz, B. Rosenhahn, T. Brox, D. Cremers, J. Weickert, L. Wietzke, and G. Sommer. Region-Based Pose Tracking. In *Iberian Conference on Pattern Recognition and Image Analysis*, pages 56–63, 2007. (page 44)

[244] F. Schroff, D. Kalenichenko, and J. Philbin. FaceNet: A Unified Embedding for Face Recognition and Clustering. In *Conference on Computer Vision and Pattern Recognition*, pages 815–823, 2015. (page 104)

[245] T. J. Sejnowski. *The Deep Learning Revolution*. MIT Press, 2018. (page 28)

[246] P. Sermanet, K. Kavukcuoglu, S. Chintala, and Y. LeCun. Pedestrian Detection with Unsupervised Multi-Stage Feature Learning. In *Conference on Computer Vision and Pattern Recognition*, pages 3626–3633, 2013. (page 37)

[247] J. Shi and C. Tomasi. Good Features to Track. In *Conference on Computer Vision and Pattern Recognition*, pages 593–600, 1994. (page 10)

[248] S. Shi, Q. Wang, P. Xu, and X. Chu. Benchmarking State-Of-The-Art Deep Learning Software Tools. In *International Conference on Cloud Computing and Big Data*, pages 99–104, 2016. (page 2, 28)

[249] A. Shrivastava, T. Pfister, O. Tuzel, J. Susskind, W. Wang, and R. Webb. Learning from Simulated and Unsupervised Images through Adversarial Training. In *Conference on Computer Vision and Pattern Recognition*, pages 2107–2116, 2017. (page 10, 53)

[250] K. Simonyan and A. Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv:1409.1556*, 2014. (page 29, 31, 39, 61, 65, 145)

[251] I. Skrypnyk and D. Lowe. Scene Modelling, Recognition and Tracking with Invariant Image Features. In *International Symposium on Mixed and Augmented Reality*, pages 110–119, 2004. (page 10, 41)

[252] E. Smith, S. Fujimoto, A. Romero, and D. Meger. GEOMetrics: Exploiting Geometric Structure for Graph-Encoded Objects. In *International Conference on Machine Learning*, pages 5866–5876, 2019. (page 45)

[253] M. Sonka, V. Hlavac, and R. Boyle. *Image Processing, Analysis, and Machine Vision*. Cengage Learning, 2014. (page 1, 7)

[254] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014. (page 34)

[255] G. Stockman and L. Shapiro. *Computer Vision*. Prentice Hall Professional Technical Reference, 2001. (page 40, 45)

[256] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller. Multi-View Convolutional Neural Networks for 3D Shape Recognition. In *Conference on Computer Vision and Pattern Recognition*, pages 945–953, 2015. (page 46, 58, 59, 62)

[257] H. Su, C. Qi, Y. Li, and L. Guibas. Render for CNN: Viewpoint Estimation in Images Using CNNs Trained with Rendered 3D Model Views. In *International Conference on Computer Vision*, pages 2686–2694, 2015. (page 10, 34, 42, 55, 64, 67, 143, 144, 145)

[258] X. Sun, J. Wu, X. Zhang, Z. Zhang, C. Zhang, T. Xue, J. Tenenbaum, and W. T. Freeman. Pix3D: Dataset and Methods for Single-Image 3D Shape Modeling. In *Conference on Computer Vision and Pattern Recognition*, pages 2974–2983, 2018. (page 9, 11, 43, 45, 52, 55, 78, 82, 83, 99, 104, 119, 126)

[259] M. Sundermeyer, Z.-C. Marton, M. Durner, M. Brucker, and R. Triebel. Implicit 3D Orientation Learning for 6D Object Detection from RGB Images. In *European Conference on Computer Vision*, pages 699–715, 2018. (page 42)

[260] I. Sutskever, J. Martens, G. Dahl, and G. E. Hinton. On the Importance of Initialization and Momentum in Deep Learning. In *International Conference on Machine Learning*, pages 1139–1147, 2013. (page 34)

[261] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction.* MIT Press, 1998. (page 144)

[262] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi. Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. In *American Association for Artificial Intelligence Conference*, pages 4278–4284, 2017. (page 39, 44)

[263] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going Deeper with Convolutions. In *Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015. (page 29)

[264] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing Properties of Neural Networks. *arXiv:1312.6199*, 2013. (page 11)

[265] R. Szeliski. *Computer Vision: Algorithms and Applications.* Springer Science & Business Media, 2010. (page 1, 20, 21, 26, 45)

[266] F. P. Tasse and N. Dodgson. Shape2Vec: Semantic-Based Descriptors for 3D Shapes, Sketches and Images. *ACM Transactions on Graphics*, 35(6):208, 2016. (page 47, 99)

[267] M. Tatarchenko, S. Richter, R. Ranftl, Z. Li, V. Koltun, and T. Brox. What Do Single-View 3D Reconstruction Networks Learn? In *Conference on Computer Vision and Pattern Recognition*, pages 3405–3414, 2019. (page 47, 105)

[268] J. Taylor, J. Shotton, T. Sharp, and A. Fitzgibbon. The Vitruvian Manifold: Inferring Dense Correspondences for One-Shot Human Pose Estimation. In *Conference on Computer Vision and Pattern Recognition*, pages 103–110, 2012. (page 45)

[269] B. Tekin, S. N. Sinha, and P. Fua. Real-Time Seamless Single Shot 6D Object Pose Prediction. In *Conference on Computer Vision and Pattern Recognition*, pages 292–301, 2018. (page 43, 77)

[270] D. Thachasongtham, T. Yoshida, F. De Sorbier, and H. Saito. 3D Object Pose Estimation using Viewpoint Generative Learning. In *Scandinavian Conference on Image Analysis*, pages 512–521, 2013. (page 41)

[271] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel. Domain Randomization for Transferring Deep Neural Networks from Simulation to The Real World. In *International Conference on Intelligent Robots and Systems*, pages 23–30, 2017. (page 34, 53)

[272] R. Toldo, U. Castellani, and A. Fusiello. A Bag of Words Approach for 3D Object Categorization. In *International Conference on Computer Vision*, pages 116–127, 2009. (page 10)

[273] L. Tran and X. Liu. Nonlinear 3D Face Morphable Model. In *Conference on Computer Vision and Pattern Recognition*, pages 7346–7355, 2018. (page 26)

[274] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon. Bundle Adjustment – A Modern Synthesis. In *International Workshop on Vision Algorithms*, pages 298–372, 1999. (page 10, 25, 32, 82, 87)

[275] J. Trussell and M. Vrhel. *Fundamentals of Digital Imaging*. Cambridge University Press, 2008. (page 7)

[276] R. Tsai. A Versatile Camera Calibration Technique for High-Accuracy 3D Machine Vision Metrology Using Off-The-Shelf TV Cameras and Lenses. *Journal of Robotics and Automation*, 3(4):323–344, 1987. (page 10, 24, 48)

[277] S. Tulsiani, J. Carreira, and J. Malik. Pose Induction for Novel Object Categories. In *International Conference on Computer Vision*, pages 64–72, 2015. (page 42, 43, 65)

[278] S. Tulsiani, S. Gupta, D. Fouhey, A. Efros, and J. Malik. Factoring Shape, Pose, and Layout from the 2D Image of a 3D Scene. In *Conference on Computer Vision and Pattern Recognition*, pages 302–310, 2018. (page 2, 45, 47, 98)

[279] S. Tulsiani and J. Malik. Viewpoints and Keypoints. In *Conference on Computer Vision and Pattern Recognition*, pages 1510–1519, 2015. (page 11, 33, 42, 43, 60, 64, 65, 67, 72, 76, 83, 85, 118, 126, 145)

[280] G. Turk and M. Levoy. Zippered Polygon Meshes from Range Images. In *ACM SIGGRAPH*, pages 311–318, 1994. (page 18)

[281] S. E. Umbaugh. *Digital Image Processing and Analysis*. CRC Press, 2010. (page 30)

[282] L. Vacchetti, V. Lepetit, and P. Fua. Combining Edge and Texture Information for Real-Time Accurate 3D Camera Tracking. In *International Symposium on Mixed and Augmented Reality*, pages 48–56, 2004. (page 41)

[283] K. E. A. Van de Sande, J. R. R. Uijlings, T. Gevers, and A. W. M. Smeulders. Segmentation as Selective Search for Object Recognition. In *International Conference on Computer Vision*, pages 1879–1886, 2011. (page 39)

[284] P. Viola and M. Jones. Rapid Object Detection using a Boosted Cascade of Simple Features. In *Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2001. (page 37, 38)

[285] H. Wang, S. Sridhar, J. Huang, J. Valentin, S. Song, and L. Guibas. Normalized Object Coordinate Space for Category-Level 6D Object Pose and Size Estimation. In *Conference on Computer Vision and Pattern Recognition*, pages 2642–2651, 2019. (page 44, 45, 52, 53, 54)

[286] N. Wang, Y. Zhang, Z. Li, Y. Fu, W. Liu, and Y.-G. Jiang. Pixel2Mesh: Generating 3D Mesh Models from Single RGB Images. In *European Conference on Computer Vision*, pages 52–67, 2018. (page 45)

[287] S. Wang, J. Wu, X. Sun, W. Yuan, W. T. Freeman, J. Tenenbaum, and E. Adelson. 3D Shape Perception from Monocular Vision, Touch, and Shape Priors. In *International Conference on Intelligent Robots and Systems*, pages 1606–1613, 2018. (page 99)

[288] Y. Wang, X. Tan, Y. Yang, X. Liu, E. Ding, F. Zhou, and L. S. Davis. 3D Pose Estimation for Fine-Grained Object Categories. In *European Conference on Computer Vision Workshops*, pages 1–13, 2018. (page 10, 43, 44, 48, 52, 53, 54, 76, 77, 78, 79, 82, 83, 84, 85, 86, 88, 89, 99, 100, 104, 118, 119, 121, 127, 143)

[289] J. Watson, M. Firman, G. J. Brostow, and D. Turmukhambetov. Self-Supervised Monocular Depth Hints. In *International Conference on Computer Vision*, pages 2162–2171, 2019. (page 144)

[290] K. Weinberger and L. Saul. Distance Metric Learning for Large Margin Nearest Neighbor Classification. *Journal of Machine Learning Research*, 10:207–244, 2009. (page 47, 62)

[291] Y. Wen, K. Zhang, Z. Li, and Y. Qiao. A Discriminative Feature Learning Approach for Deep Face Recognition. In *European Conference on Computer Vision*, pages 499–515, 2016. (page 103, 110, 113)

[292] C. Wöhler. *3D Computer Vision: Efficient Methods and Applications*. Springer Science & Business Media, 2012. (page 2, 26)

[293] P. Wohlhart and V. Lepetit. Learning Descriptors for Object Recognition and 3D Pose Estimation. In *Conference on Computer Vision and Pattern Recognition*, pages 3109–3118, 2015. (page 42, 103)

[294] S. Workman, C. Greenwell, M. Zhai, R. Baltenberger, and N. Jacobs. DEEPFOCAL: A Method for Direct Focal Length Estimation. In *International Conference on Image Processing*, pages 1369–1373, 2015. (page 48)

[295] D. E. Worrall, S. J. Garbin, D. Turmukhambetov, and G. J. Brostow. Harmonic Networks: Deep Translation and Rotation Equivariance. In *Conference on Computer Vision and Pattern Recognition*, pages 5028–5037, 2017. (page 43, 145)

[296] C. Wu. P3.5P: Pose Estimation with Unknown Focal Length. In *Conference on Computer Vision and Pattern Recognition*, pages 2440–2448, 2015. (page 26, 48, 84)

[297] J. Wu, Y. Wang, T. Xue, X. Sun, W. T. Freeman, and J. Tenenbaum. MarrNet: 3D Shape Reconstruction via 2.5D Sketches. In *Advances in Neural Information Processing Systems*, pages 540–550, 2017. (page 45)

[298] J. Wu, C. Zhang, X. Zhang, Z. Zhang, W. T. Freeman, and J. Tenenbaum. 3D ShapeNets: A Deep Representation for Volumetric Shapes. In *Conference on Computer Vision and Pattern Recognition*, pages 1912–1920, 2015. (page 46)

[299] Y. Wu and Z. Hu. PnP Problem Revisited. *Journal of Mathematical Imaging and Vision*, 24(1):131–141, 2006. (page 25, 82)

[300] Y. Wu and K. He. Group Normalization. In *European Conference on Computer Vision*, pages 3–19, 2018. (page 34)

[301] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. Learning Shape Priors for Single-View 3D Completion and Reconstruction. In *European Conference on Computer Vision*, pages 646–662, 2018. (page 99)

[302] Y. Xiang, W. Kim, W. Chen, J. Ji, C. Choy, H. Su, R. Mottaghi, L. Guibas, and S. Savarese. ObjectNet3D: A Large Scale Database for 3D Object Recognition. In *European Conference on Computer Vision*, pages 160–176, 2016. (page 9, 10, 42, 47, 52, 53, 58, 59, 62, 76, 99, 105, 143, 144)

[303] Y. Xiang, R. Mottaghi, and S. Savarese. Beyond Pascal: A Benchmark for 3D Object Detection in the Wild. In *IEEE Winter Conference on Applications of Computer Vision*, pages 75–82, 2014. (page 9, 10, 52, 53, 54, 57, 59, 60, 61, 63, 65, 76, 83, 105, 128, 143, 144)

[304] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox. PoseCNN: A Convolutional Neural Network for 6D Object Pose Estimation in Cluttered Scenes. In *Robotics: Science and Systems Conference*, pages 1–10, 2018. (page 43)

[305] Y. Xiang, C. Song, R. Mottaghi, and S. Savarese. Monocular Multiview Object Tracking with 3D Aspect Parts. In *European Conference on Computer Vision*, pages 220–235, 2014. (page 42)

[306] Y. Xiao, X. Qiu, P.-A. Langlois, M. Aubry, and R. Marlet. Pose from Shape: Deep Pose Estimation for Arbitrary 3D Objects. In *British Machine Vision Conference*, pages 120:1–120:14, 2019. (page 42, 44, 118)

[307] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He. Aggregated Residual Transformations for Deep Neural Networks. In *Conference on Computer Vision and Pattern Recognition*, pages 1492–1500, 2017. (page 39)

[308] Y. Yang, C. Feng, Y. Shen, and D. Tian. FoldingNet: Point Cloud Auto-Encoder via Deep Grid Deformation. In *Conference on Computer Vision and Pattern Recognition*, pages 206–215, 2018. (page 45)

[309] S. Yao, T. M. Hsu, J.-Y. Zhu, J. Wu, A. Torralba, W. T. Freeman, and J. Tenenbaum. 3D-Aware Scene Manipulation via Inverse Graphics. In *Advances in Neural Information Processing Systems*, pages 1887–1898, 2018. (page 48)

[310] K. M. Yi, E. Trulls, V. Lepetit, and P. Fua. LIFT: Learned Invariant Feature Transform. In *European Conference on Computer Vision*, pages 467–483, 2016. (page 146)

[311] X. Zabulis, M. I. A. Lourakis, and S. S. Stefanou. 3D Pose Refinement Using Rendering and Texture-Based Matching. In *International Conference on Computer Vision and Graphics*, pages 672–679, 2014. (page 44)

[312] S. Zagoruyko and N. Komodakis. Learning to Compare Image Patches via Convolutional Neural Networks. In *Conference on Computer Vision and Pattern Recognition*, pages 4353–4361, 2015. (page 123)

[313] S. Zakharov, I. Shugurov, and S. Ilic. DPOD: Dense 6D Pose Object Detector in RGB Images. In *International Conference on Computer Vision*, pages 1941–1950, 2019. (page 44, 118, 126, 128, 129, 130, 131, 132, 133, 134, 135, 138, 139)

[314] A. R. Zamir, A. Sax, W. Shen, L. J. Guibas, J. Malik, and S. Savarese. Taskonomy: Disentangling Task Transfer Learning. In *Conference on Computer Vision and Pattern Recognition*, pages 3712–3722, 2018. (page 33)

[315] A. Zell. *Simulation of Neural Networks.* Addison-Wesley, 1994. (page 31)

[316] X. Zhang, Z. Zhang, C. Zhang, J. Tenenbaum, W. T. Freeman, and J. Wu. Learning to Reconstruct Shapes from Unseen Classes. In *Advances in Neural Information Processing Systems*, pages 2257–2268, 2018. (page 45, 99)

[317] Z. Zhang. A Flexible New Technique for Camera Calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1330–1334, 2000. (page 10, 24, 26, 48)

[318] Y. Zheng and L. Kneip. A Direct Least-Squares Solution to the PnP Problem with Unknown Focal Length. In *Conference on Computer Vision and Pattern Recognition*, pages 1790–1798, 2016. (page 26, 48)

[319] Y. Zheng, S. Sugimoto, I. Sato, and M. Okutomi. A General and Simple Method for Camera Pose and Focal Length Determination. In *Conference on Computer Vision and Pattern Recognition*, pages 430–437, 2014. (page 48)

[320] J. Zhu, F. Zhu, E. Wong, and Y. Fang. Learning Pairwise Neural Network Encoder for Depth Image-Based 3D Model Retrieval. In *ACM International Conference on Multimedia*, pages 1227–1230, 2015. (page 62)

[321] Z. Zhu, X. Wang, S. Bai, C. Yao, and X. Bai. Deep Learning Representation Using Autoencoder for 3D Shape Retrieval. *Neurocomputing*, 204:41–50, 2016. (page 62)