TECHNISCHE UNIVERSITÄT DRESDEN

Faculty of Computer Science Institute of Software and Multimedia Technology

LEARNING TO PREDICT DENSE CORRESPONDENCES FOR 6D POSE ESTIMATION

Eric Brachmann

DISSERTATION

TECHNISCHE UNIVERSITÄT DRESDEN

Faculty of Computer Science Institute of Software and Multimedia Technology

LEARNING TO PREDICT DENSE CORRESPONDENCES FOR 6D POSE ESTIMATION

Eric Brachmann
Born on: 13th May 1987 in Chemnitz

DISSERTATION

to achieve the academic degree

DOKTOR RERUM NATURALIUM (DR. RER. NAT.)

First referee
Prof. Dr. Gumhold

Second referee
Prof. Ing. PhD. Matas

Advisor
Prof. PhD. Rother

Supervisor
Prof. Dr. Gumhold

Submitted on: 5th September 2017
Defended on: 17th January 2018

TECHNISCHE UNIVERSITÄT DRESDEN

Faculty of Computer Science Institute of Software and Multimedia Technology

ABSTRACT

Object pose estimation is an important problem in computer vision with applications in robotics, augmented reality and many other areas. An established strategy for object pose estimation consists of, firstly, finding correspondences between the image and the object's reference frame, and, secondly, estimating the pose from outlier-free correspondences using Random Sample Consensus (RANSAC). The first step, namely finding correspondences, is difficult because object appearance varies depending on perspective, lighting and many other factors. Traditionally, correspondences have been established using handcrafted methods like sparse feature pipelines.

In this thesis, we introduce a dense correspondence representation for objects, called *object coordinates*, which can be learned. By learning object coordinates, our pose estimation pipeline adapts to various aspects of the task at hand. It works well for diverse object types, from small objects to entire rooms, varying object attributes, like textured or texture-less objects, and different input modalities, like RGB-D or RGB images. The concept of object coordinates allows us to easily model and exploit uncertainty as part of the pipeline such that even repeating structures or areas with little texture can contribute to a good solution. Although we can train object coordinate predictors independent of the full pipeline and achieve good results, training the pipeline in an end-to-end fashion is desirable. It enables the object coordinate predictor to adapt its output to the specificities of following steps in the pose estimation pipeline. Unfortunately, the RANSAC component of the pipeline is non-differentiable which prohibits end-to-end training. Adopting techniques from reinforcement learning, we introduce Differentiable Sample Consensus (DSAC), a formulation of RANSAC which allows us to train the pose estimation pipeline in an end-to-end fashion by minimizing the expectation of the final pose error.

Statement of authorship

I hereby certify that I have authored this Dissertation entitled *Learning to Predict Dense Correspondences for 6D Pose Estimation* independently and without undue assistance from third parties. No other than the resources and references indicated in this thesis have been used. I have marked both literal and accordingly adopted quotations as such. There were no additional persons involved in the intellectual preparation of the present thesis. I am aware that violations of this declaration may lead to subsequent withdrawal of the degree.

Dresden, 5th September 2017

Eric Brachmann

ACKNOWLEDGEMENT

Although the following text is among the first in this thesis, it is one of the last to be written. It is an opportunity to look back at my time as a PhD student, and it has been a wonderful time. This is due to many great people that shaped my experience of the last years, and while I cannot thank all of them I shall use the opportunity to thank a few very important people, here.

Most of the time that I have worked on the topics of this thesis I have worked together with Frank and Alex. I could not have asked for better colleagues. We began our journey into the realm of computer vision in a citadel of solitude called Falkenbrunnen where we, still accompanied by André and Sebastian, kept a list of our bad habits - a ritual which made us all a better person, I think. Later, when our work on pose estimation gained momentum we could always count on each other. I think our individual theses could not have happened so successfully in any other team.

I also want to thank my main supervisor Stefan Gumhold, who I have met long before the work on this thesis began. You helped me in countless ways during my time as a student (both undergrad and PhD) – beginning with the support for getting a large scholarship when we first met, to offering me my PhD position 5 years back which set my path to where I am now. During my time as a PhD student, you let me have all the freedom to explore any direction I found interesting, you critically questioned weaknesses of my work making it stronger, and you encouraged me to develop my ideas.

"Let's do pose estimation." – A sentence having far-reaching consequences. It was said at some point in 2013 by Carsten, and lead more or less directly to the following document. Without your knowledge of the field, your connections and advice, this thesis would have gone a different way, and would probably not have been so successful. You always contributed passionately to our effort, and it was a pleasure working with you.

Of course I also want to thank my family and friends who are probably still wondering what *exactly* I'm doing. But it's still nice that you ask from time to time, and stoically endure my lengthy explanations.

The time I started my PhD position coincided with something even more important. I met Anja. While you always supported me in anything I wanted to do, you also always kept me down-to-earth. You remind me of what's important in life, and that pose estimation might not be the center of the universe. Without you my views would be very narrow, and life would be boring. I enjoy every step of the way we go together. It looks exciting ahead.

CONTENTS

1. INTRODUCTION

Contents

The last centuries have seen an accelerating development in technical achievements which has enhanced the living quality of many people around the globe. Until 50 years back, these technical achievements have been foremost mechanical in nature, e.g. in the form of trains, cars or factories. With the following invention and rapid development of computers, machines were able to make fast computations, store and process large amounts of data and connect people via the Internet. Lately, we have seen a new generation of machines, driven by advances in artificial intelligence, which can solve more and more complex tasks more and more autonomously. Smart home assistants, fully automated ware houses, computer-assisted surgery and autonomous driving are all within reach. This progressing automation has the potential to increase life expectancy, e.g. by reducing the number of traffic accidents, and to lift the burden of tedious tasks, e.g. by robotic assembly and packaging. At the same time, artificial intelligence challenges our idea of human intelligence, and our social perception and status of labor.

If a machine should act autonomously in a complex environment it has to sense and understand this environment. From the many sensor types available, ordinary cameras are among the most important since they are readily available, cheap and capture rich, visual representations of the world. Computer vision is a discipline which is concerned with the processing of visual data. More specifically, image understanding seeks to extract semantic knowledge

given an image or a video. Image understanding can be divided into many sub-tasks, and in this thesis, we consider one of them: Accurate pose estimation of specific objects from visual data. Recognizing objects and comprehending their state in the environment is a key component in many autonomous systems, like autonomous robotics. It is a precondition of many interactions of the system with its surroundings, e.g. grasping, moving or avoiding certain objects.

Humans perceive their environment with their eyes to a large extent. Understanding images seems natural and easy but for a computer it is not. The brain, a very powerful processing unit, has areas specifically designated for visual processing The intricacies of any high level processing in the brain are not yet well understood but the brain is able to make sense of even very crude data by combining partial information with world knowledge. For computers, even a structured, well defined task like recognizing a specific, known object involves searching for complex patterns in the image data. The task is non-trivial because object appearance changes tremendously depending on perspective, lighting conditions or occlusion.

Early attempts to solve visual recognition tasks consisted of searching for expressive image features, like corners and edges, and describing patterns of these features unique to the object in question. Efficient, statistical analysis of an image could then yield the presence of these patterns, and hence the object itself. Finding a feature pattern that identifies an object from all possible views and under all imaging conditions is a challenging task. It was soon evident, that this process should be automated, since it has to be repeated for every new object to be recognized. By means of machine learning, a field related to statistics, the computer discovers object patterns autonomously on the basis of example views of the object, so called training data.

Until the mid 2000s, computer vision was dominated by pipelines where a set of features was extracted from the input image, and subsequently a learned classifier, e.g. a support vector machine, decides on object presence resp. location. Although containing some learned components, these pipelines where largely handcrafted, meaning that a human designed the overall strategy and the image features to solve the task at hand. Since AlexNet [KSH12] won the ImageNet classification challenge in 2012, deep learning has advanced computer vision achievements massively in very diverse tasks such as classification [KSH12], detection [Gir15], segmentation [LSD15] or human pose estimation [TS14]. The main advantage of deep learning is that a neural net, in computer vision often a convolutional neural net (CNN), can combine feature extraction and classification resp. regression within one flexible yet powerful architecture. A CNN can adapt to a task by end-to-end learning, i.e. choosing millions of free parameters based on pairs of inputs and desired outputs. All components of the system, features and classifier, can optimally adapt to each other. Within the last 5 years, new computational capabilities and vast sets of labeled data enabled computer vision algorithms to even exceed human accuracy on certain tasks like image classification [RDS+15], face recognition [HRBLM07] or traffic sign recognition [SSSI12].

Neural networks can be constructed for a wide range of (structured) inputs and outputs, hence they can be applied to many problems in computer vision. Since neural networks are end-to-end systems, trainable from data, the need for pipeline design seems largely expendable. Consequently, research in deep learning has focused, for a while, on better optimization methods for learning [KB14], building blocks like activation functions [HZRS15b] or memory units [GWR+16], and neural net architectures [SZ14, LSD15, HZRS15a]. However, to train a neural network end-to-end large amounts of training data are needed which is a problem for some applications. For example, for object detection and pose estimation it is in general not feasible to collect thousands of images, and to annotate them with the desired prediction

for each new object. However, we can exploit prior knowledge about the task to reduce the necessary amount of training data, tremendously. For example, if we would want to estimate an object rotation from data, we could define the output as a 3×3 rotation matrix. In this case, the CNN would have to learn which constraints a matrix has to fulfill to represent a rotation. We could also encode this knowledge directly in the architecture by choosing a rotation representation, e.g. an axis-angle vector, where no additional constraints have to be learned.

Hence, it can be beneficial to decompose a task into components, some of which should be learned, while some others can be constructed based on task knowledge. For instance, in most object detection systems, the knowledge that an object can appear anywhere in an image is encoded within a sliding window component. The image features and object classifier, on the other hand, are learned. In this work, we follow the same strategy for object pose estimation: The system we propose is a combination of learned components and fixed components, modeled according to extensive prior knowledge about the task. As we will show throughout this thesis, this strategy does not prevent end-to-end learning which sets it apart from traditional handcrafted pipelines with e.g. a classifier learned separately.

The knowledge we specifically exploit for pose estimation is that of geometry. Geometry is an area of mathematics whose relation with computer vision was extensively studied from the 1980s to the early 2000s. It describes the relationship of the 3D world and its depiction in 2D images. We know that the position and orientation of objects within the world can be modeled by rigid body transformations, and we know how a camera projects 3D structures onto the image plane. This knowledge enables us to reason about the pose of an object relative to the camera given an image.

The problem of object pose estimation is particularly interesting because it is affected by the major challenges of computer vision, namely extremely variable object appearance subject to imaging conditions like view point, lighting etc. At the same time, training data is usually limited. Object pose estimation has many applications, some of which, like augmented reality, require a high level of accuracy while allowing only a low computational budget.

To summarize: In this thesis, we deal with the problem of pose estimation of known object instances from a single camera image. We solve this problem using a combination of machine learning and principles from geometry. Our solution is versatile and can be applied to many variants of the problem, yet is highly precise. In the remainder of this section, we introduce the task of 6D pose estimation of object instances more formally, see Sec. 1.1. We start with a simple, restricted setup, followed by more complex variants. We also discuss challenges and applications of pose estimation. Then, we give an overview of the main concept of the system presented, including the main contributions and a list of associated publications, see Sec. 1.2. In Sec. 1.3, we review the extensive related work concerning object pose estimation. We close the introductory chapter by outlining the structure of the remaining document in Sec. 1.4.

1.1. OBJECT POSE ESTIMATION

Object pose estimation can come in many different forms. As explained in the introduction, we are concerned with pose estimation of rigid object instances. An object instance means a specific object of the real world which is different from all other objects in terms of shape and material. This is in contrast to an object class, which includes all objects of a certain type. For example, the object class *car* could contain all four-wheeled vehicles. A specific car model from that class would be an *instance*, e.g. a red VW Golf VII. The definition of the

instance concept can vary slightly. Some define an instance as being associated with exactly one physical object. Others would argue that two cars of the exact same model, including attributes like color, belong to the same instance since they cannot be distinguished based on appearance. In this work, we follow the latter view.

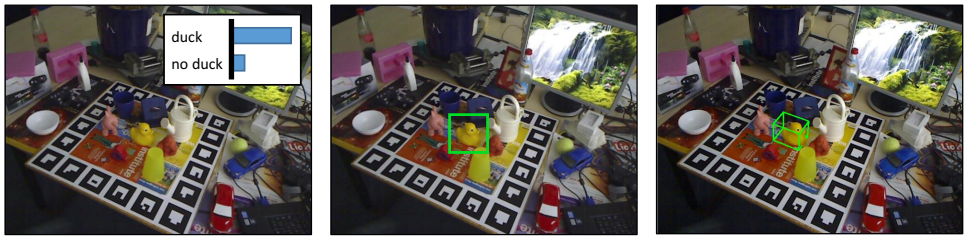

Figure 1.1.: **Recognition vs. Detection vs. Pose Estimation. Left:** A recognition system decides (or gives probabilities) whether an object is visible within the image or not. **Center:** A detection system estimates the 2D location of an object. **Right:** A 6D pose estimation system estimates the 3D position and 3D orientation of an object relative to the camera.

Furthermore, we are concerned with the estimation of *6D poses* of objects. This is in contrast to the related tasks of *recognition* and *detection*, see Fig. 1.1. Recognition is the task of deciding whether an image shows an object or not, i.e. it is a binary classification problem. Detection, as we understand it here, is the task of localizing the 2D position of an object within the image. Sometimes, detection includes estimation of object scale or the full 2D bounding box making it a 2D to 4D regression problem. Going beyond 2D detection, we reason about the location of the object in the 3D world. This involves the estimation of the 3D object position and the 3D object orientation. Note that we estimate continuous, accurate poses as opposed to quantized viewpoints like *front* or *side* which is sometimes done for class-based pose estimation.

Finally, we consider *rigid* objects, i.e. we assume that objects have no moving parts or deformation parameters. In this case, a rigid body transformation is an adequate representation an object's pose.

Formal Definition. The basic setup of pose estimation considered in this work is the following. Variants and extensions of the basic setup will be discussed below. Given a single RGB-D camera frame I, we estimate the rigid body transformation $\mathbf{h}$ of an object instance relative to the camera. Transformation $\mathbf{h}$ is a 6D vector which consists of a 3D translation $\mathbf{t}$ and a 3D orientation $\boldsymbol{\theta}$, e.g. in axis-angle representation. In the basic setup, the object is known to be present in the image.

The depth channel simplifies the pose estimation problem, greatly. It contains rich cues about the shape of the object, discontinuities in depth are a strong indicator of object boundaries, and depth measurements are independent of lighting conditions. Furthermore, image depth constrains the z-component of the unknown object translation. Many RGB-D methods (see Sec. 1.3) include a final refinement that aligns a 3D model of the object with the local depth at the estimated position, e.g. by iterative closest point (ICP) variants. Therefore, the initial pose estimation is allowed to be somewhat coarse as it is just an initialization for refinement.

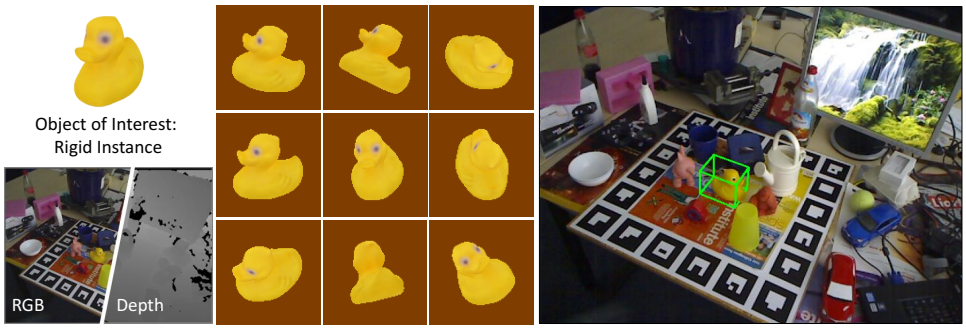

Figure 1.2.: **Basic Setup of Object Pose Estimation.** We want to estimate the 3D location and 3D orientation of a rigid object instance, a toy duck in this example, given an RGB-D image. The image is guaranteed to contain the object. Training data shows the object from different view points where the ground truth pose was annotated for each image. The result is shown as a green bounding box rotated and translated by the estimated pose.

We assume that training data of the specific object is given, i.e. we treat pose estimation as a supervised learning problem. Training data consists of a set of RGB-D images with ground truth pose annotations $\mathbf{h}^*$. Training images can be real images, i.e. captured by a camera, or synthetic images rendered using a 3D model of the object. The training data should cover the same view range as expected during test time. Usually, the training data contains a few hundred images for each object. The basic setup is illustrated in Fig. 1.2, we address this setup mainly in Chapter 2.

1.1.1. VARIANTS

Many variants of the aforementioned basic setup exist which we will briefly explain below. We will address some of these directly in the following chapters of this thesis. Other variants were addressed by extensions of the system we propose here.

Pose Estimation from RGB Only. This variant is identical to the basic setup but the input is an RGB image instead of an RGB-D image. RGB inputs are less reliable as they are directly affected by lighting changes, and object boundaries are less prominent, depending on the background. Furthermore, the distance of the object w.r.t. to the camera is hard to estimate accurately, because the projected size changes only slightly with varying distance. Refinement cannot exploit the local scene geometry, e.g. by using ICP. Therefore, accurate pose estimates are harder to achieve. On the other hand, in many applications, only RGB inputs are available since RGB-D cameras are limited to a few consumer products, e.g. to Kinect style cameras [WA], many of which also do not work well outdoors. We show how to estimate accurate 6D poses from single RGB images in Chapter 3.

Multi-Object Pose Estimation. Instead of estimating the pose of one object which is known to be in the image, the system is trained with multiple objects simultaneously. Given

an RGB or RGB-D image, the system should estimate the poses of all objects present. Not all objects known to the system appear in the input image. Hence, additional to pose estimates, the system has to predict indicators (e.g. probabilities) of whether an object is visible in the image or not, see Fig. 1.3, left. Apart from accuracy, pose estimation systems are assessed by their scalability, i.e. their capability to handle many objects in reasonable runtime. A special case of multi-object pose estimation is multi-instance pose estimation, i.e. one instance can appear in an image multiple times. We do not cover multi-instance pose estimation in this work, i.e. every object is assumed to appear at most once in an image. Our system is natively trained for multiple objects jointly, and can decide whether an object is present in the image or not, see Chapter 2. We show how we efficiently infer the poses of many objects simultaneously in Chapter 3.

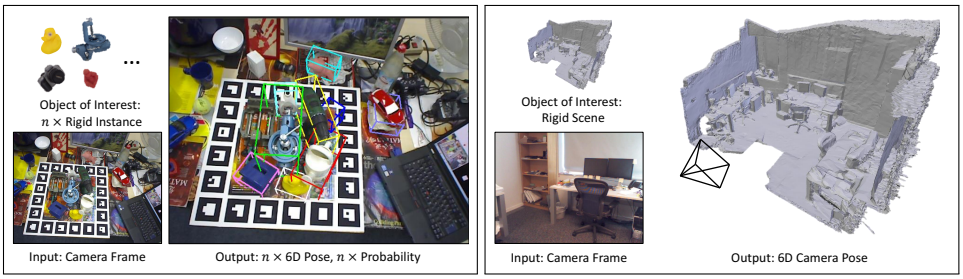

Figure 1.3.: **Multiple Objects and Camera Localization. Left:** In multi-object pose estimation we train the system with many objects. Given an input image, it has to decide (or give probabilities) whether an object appears or not. For example, the system was trained with the camera object, but it is not present in the image. For all objects that do appear, 6D poses should be accurately estimated. Estimated poses are shown using bounding boxes with the color indicating the object ID. **Right:** Camera Localization. Given an image of a known scene, the system estimates the 6D pose of the corresponding camera. The result is shown as a black camera frustum w.r.t. a 3D model of the scene.

Camera (Re-)Localization. So far we have considered pose estimation of small objects which appear in an unknown environment. However, the environment, or *scene*, itself can be considered a rigid object and its pose can be estimated. This problem is called camera localization or re-localization, because usually the pose of the camera is estimated which is the inverse of the scene pose. Hence, we are given an image, RGB or RGB-D, of a known environment, and the system should estimate the 6D pose of the camera which took the image, see also Fig. 1.3, right for an illustration. Conceptually, this problem is easier than object pose estimation because the object does not have to be located as it occupies the whole image. In practice, camera localization is still very challenging, because scenes are usually not completely static, they often contain large, planar, texture-less areas, and repeating structures like windows. Furthermore, in some applications, scenes can reach vast scales, e.g. city-scale image based localization [CBK+11]. We apply our system to the camera localization problem in Chapter 3 and 4.

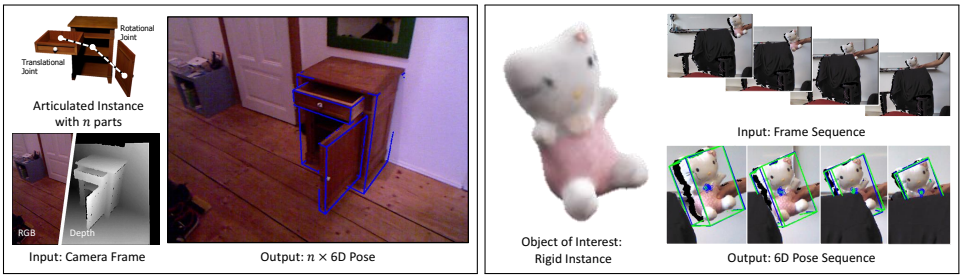

Figure 1.4.: **Articulated Objects and Pose Tracking. Left:** The object of interest is articulated, i.e. it is composed of multiple rigid parts connected by joints. The system should estimate a 6D pose for each part, but respect connectivity constraints as well as the degrees of freedom allowed by the joints. The result is shown as one blue bounding box per part. **Right:** Pose Tracking. Instead of a single frame, the movement of an object is tracked over a sequence of frames. For each frame, a 6D pose should be estimated, taking into account estimations for the previous frames. We show results as green bounding boxes.

Pose Estimation of Articulated Objects. While many objects in the real world are more or less rigid, some have additional degrees of freedom in the form of joints resp. moving parts, see Fig. 1.4. For example, a cupboard with a door consists of two parts connected by a rotational joint. Articulation increases the dimensionality of the pose estimation problem, i.e. the pose $\mathbf{h}$ consists of a 3D location, a 3D rotation and additional degrees of freedom for every movable part. A naive solution of the problem is to model each part of the articulated object as a separate rigid instance, and to estimate poses individually. However, this usually yields poor results as it neglects the connectivity constraints among parts, and parts often occlude each other severely. Our system was extended to estimate poses of articulated objects in [MKB+15]. Articulated objects are a special case of deformable objects. In general, deformable objects, like organs, are not composed of rigid parts, and often lack a clear parametric deformation model. Articulated and deformable objects are out of scope of this work.

Pose Tracking. In some applications, the pose of a moving object should be estimated over time. In this case, the input is a consecutive sequence of RGB-D or RGB images for each of which a 6D pose should be predicted, see Fig. 1.4, right. While a pose estimation system can be applied to each frame individually, taking into account continuity constraints between frames, i.e. tracking the object, can greatly improve robustness, e.g. w.r.t. occlusion or deformation. For this purpose, a motion model expresses the plausibility of changes in position and orientation of the object between frames. This prior knowledge can be used in the pose estimation to yield a smooth trajectory in 6D pose space. Because the search space in each new frame is greatly reduced, tracking often leads to reduced runtime resp. real-time performance. However, one-shot pose estimation is still needed to initialize a tracking system, and to recover from tracking failure. We do not address pose tracking in this thesis, but [KMB+14] describes a tracking system built upon the pipeline presented in Chapter 2.

1.1.2. CHALLENGES

Object instance pose estimation has been researched for many years, see Sec. 1.3. However, the problem is not yet solved, except for a few specific, simplified scenarios. Instance pose estimation in general comes with many difficulties. We list the main challenges here.

Ambiguous Texture and Shape. Many objects encountered in realistic environments have an undistinctive shape or lack salient texture features. This has lead to the development of methods specialized for texture-less objects in the early 2010s, see also Sec. 1.3. Another difficulty are objects that show some form of symmetry. In that case, different poses produce visually the same image. Such ambiguities are problematic for estimation methods because one input has to produce multiple outputs. Furthermore, it affects also data collection and evaluation. Images showing ambiguous poses should be annotated with all valid solutions, and evaluation should accept any of them. However, this comes with substantial additional effort in case of human annotators, and is non-trivial to achieve also for synthetic data generation methods, like rendering. For instance, a cup is only rotationally symmetric if the handle is occluded. Therefore, most available datasets do not consider ambiguities in their ground truth annotations [HMS+]. Some authors proposed evaluation metrics that are robust to ambiguities to some extent [HLI+12, HMO16].

Lighting. Different lighting conditions can change the appearance of an object to a large extent. It changes the gradients on the object surface and intense light can create hard shadows, i.e. new gradients, on the surface. Furthermore, the color of the light can have large effects on the color of the object. As humans, we do not perceive this effect to the full extent because our brain adjusts for it. Either we know the true color of the object from experience, or we estimate the color of lighting from its effects on the environment. Todays computer vision systems do not have this level of reasoning and world knowledge. Methods that rely solely on depth cameras are invariant to lighting conditions but object color and texture can be important cues that these methods ignore. Also, intense sunlight can interfere with some depth sensor types.

Clutter and Occlusion. Depending on size and distance, an object will often occupy only a small fraction of the input image such that some kind of detection or segmentation is needed. In practice, many similar objects might be present in an image which may confuse a detection method. Also, many machine learning algorithms have a closed world assumption, i.e. they decide among a fixed set of known classes. However, the exact background as well as clutter objects are usually unknown at training time. A further challenge is occlusion which can make the pose estimation arbitrarily difficult depending on the object area which is occluded. But even small occlusions can be problematic if it affects a distinctive feature, like the handle of the aforementioned cup.

Reflections. Some objects are composed of difficult, reflective materials which are a problem for both RGB and RGB-D based methods. Depth sensors will produce either wrong measurements for such materials or none at all. In the RGB image, the texture of the object is overlayed with reflections of the environment. This makes the appearance of an object dependent on the interplay of its surroundings, the lighting and the viewing angle of the camera. The environment is unknown at training time and only partially visible in the image at test time. Hence, physical reasoning about reflections is extremely difficult. All of these challenges also apply to transparent objects like glasses or bottles.

Synthetic Training Data. For many applications, generation of training data should be possible without much human effort. Recording and hand labeling thousands of images with accurate 6D poses by humans is often not feasible. An alternative approach relies on rendering training data based on a CAD model or a scan of the object which is much easier to obtain. Rendered, or synthetic, training data comes in arbitrary amounts and with perfect ground truth annotations. Also, the distribution of viewing angles can be easily controlled. However, because of limitations in the scanning process and simplifications in the rendering pipeline, rendered training data often looks different from the real test data, see also the example on the left. This is a problem because a learning algorithm will adapt to the appearance of the training data and cannot generalize to the test data if it looks very different. A regularization which helps to bridge this gap is usually difficult to design, and limits the expressiveness of a model. For example, one could rely on handcrafted features which are robust to the appearance shift between rendered and real data. However, learning features from data usually leads to better accuracy, subject to generalization.

1.1.3. APPLICATIONS

There is a wide field of diverse applications for object instance pose estimation. We name a few but this list is not exhaustive.

Robotics. There are two main applications for object pose estimation in robotics, namely object grasping and navigation.

Many tasks performed by robots involve grasping specific objects, e.g. in assembly lines or in automated ware houses [LLC]. If a robot wants to grasp an object, it has to localize the object relative to itself, see Fig. 1.5, left. Especially objects with a complex shape might have only certain points on the surface where it can be reliably grasped by an end effector. In this case, the pose of the object has to be estimated precisely.

Camera localization can be used to localize the robot within a known environment, see Fig. 1.5, center. If a robot is deployed in an unknown environment, it usually builds a map itself incrementally, a process called Simultaneous Localization and Mapping (SLAM). However, SLAM systems can encounter tracking failure, e.g. because of camera occlusion. The robot should then re-localize itself w.r.t. to the partial map it has already build as opposed to starting SLAM all over again. This can be accomplished by learning a camera localization

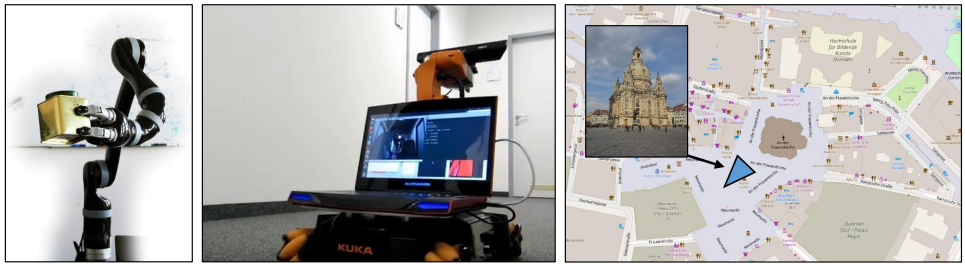

Figure 1.5.: **Applications: Robotics and Localization. Left:** A robotic arm grasping an object. **Center:** A robotic platform navigating in an indoor environment. **Right:** Image based localization where the input is a photograph, e.g. taking with a mobile phone, and the output is a location on OpenStreetMap [Ope17].

component on the fly, parallel to SLAM [CGL+17]. Camera localization is also used to recognize when a robot encounters an area it has seen before. This is called loop closure, and can be used to correct potential drift in the incremental map building process.

Outdoor/Indoor Localization. Similar to navigation in robotics, any device with a camera, e.g. a mobile phone, can localize itself relative to a known environment using pose estimation. For example a mobile phone could localize itself in case GPS (Global Positioning System) is not working or not precise enough. More specifically, using camera localization a user who is lost in a shopping mall could determine his position based on an image of his current surroundings. Or see Fig. 1.5, right for an example of image-based outdoor localization.

Augmented Reality. Augmented reality means overlaying real camera footage with virtual content such that the real and virtual world blend together. For example, it enables engineers to examine a virtual object, e.g. a product concept, in detail in a natural environment. The engineer would wear augmented reality glasses, and could move around the virtual object as if it would be real. This offers a most natural mode of navigation and, potentially, interaction. Camera localization can be used to align a virtual object with a known, real environment. For example, navigation markers can be naturally blended into a live view of the environment using a mobile phone. Object pose estimation can be used to align virtual content with a specific object. Objects can be altered or substituted in an augmented reality view, e.g. in augmented reality games.

Medicine. In modern medicine, surgeries have become very complex procedures. Computer vision can support the staff during operations or detect dangerous situations, a concept called computer aided surgery. For example, in laparoscopic surgery, the operation takes place trough the closed abdominal wall with special instruments based on an endoscopic camera view. While this minimally invasive procedure is beneficial for the patient, it is very involved for the surgeon. He has to navigate inside the body cavity and manipulate organs while taking into account diagnostic information like tumor location. The endoscopic camera view is usually very narrow, and diagnostic information is displayed separately. A vision based pose estimation system, such as the one presented in this work, could help

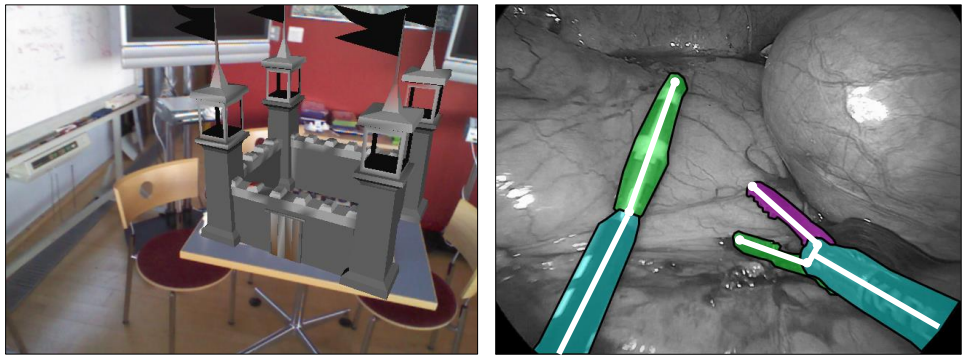

Figure 1.6.: **Applications: Augmented Reality and Medicine. Left:** The system proposed in this thesis was used to estimate the pose of the room, and render a virtual castle on the table in a plausible way. **Right:** Poses of surgical instruments are tracked during a laparoscopic surgery based on an endoscopic camera view.

by estimating the pose of surgical tools or locating the endoscopic camera within the cavity. Augmented reality methods could be used to overlay diagnostic information directly with the endoscopic view.

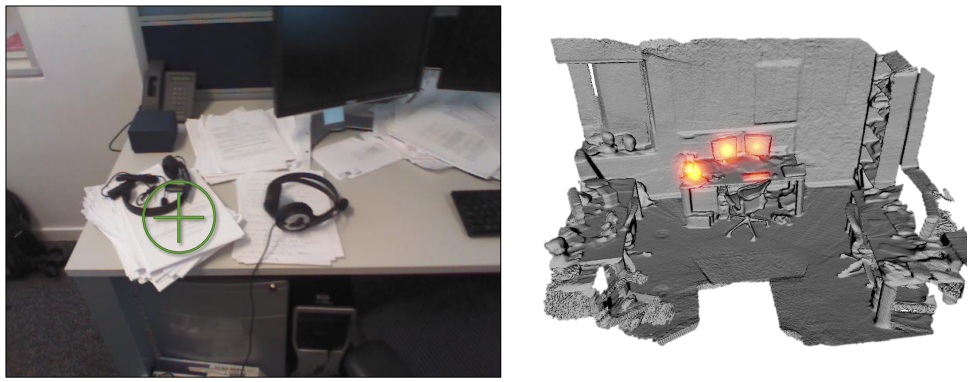

Figure 1.7.: **Applications: Psychology. Left:** View of a person wearing an eye tracker device. Her point of attention is marked with a green cross. **Right:** Camera localization allows the creation of attention maps registered with the 3D world. Colored areas mark zones of high attention.

Psychology. In perceptional studies, psychologist are interested in the parts of the environment which a test subject attends to during a specific task, e.g. in critical situations at the workplace. The test subject usually wears a mobile eye tracker which measures the eye movement relative to a head mounted camera. The camera, in turn, records the environment. Camera localization can be used to align these recordings, and thus the gaze of the subject, with a scene model, see Fig. 1.7. Important areas can be marked beforehand in the

scene model to facilitate automated analysis and creation of gaze statistics. This approach is especially beneficial if the environment must not be altered, e.g. applying fiducial markers in secure areas is usually not possible.

Monitoring. In controlled environments, like a factory, a warehouse or an apron there are defined procedures which involve known, specific objects, e.g. certain airplane types. Often CCTV cameras or depth scanners are available to monitor said facilities. A vision-based pose estimation system, like the one presented in this work, can be used to ensure that operation takes place within recommended parameters. For example, our system has been applied to airplane recognition in apron control [MMDM+16].

1.2. OVERVIEW

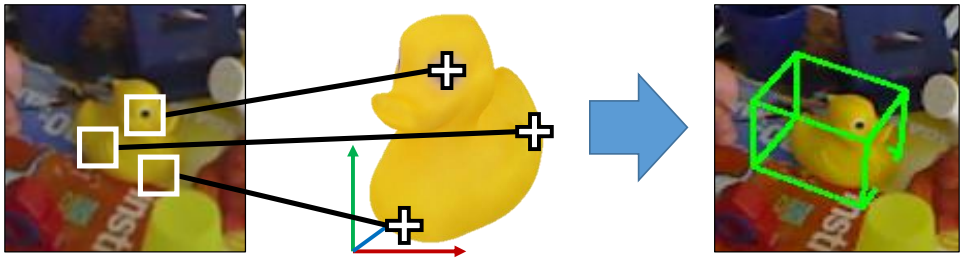

Figure 1.8.: **Pose Estimation from Correspondences.** We estimate an object's pose by first predicting correspondences between the input image and a 3D model of the object. Then, we estimate the rigid body transformation that aligns the 3D model and the input image based on these correspondences. We show the alignment via the bounding box of the 3D model.

The system proposed in this thesis was inspired by an established, general approach to object pose estimation. This approach is two-staged: Firstly, given an image of the object, we establish correspondences between the image and the object. For example, we could recognize certain parts of the object in the image, and assign them to the correct position on a 3D model of the object, see Fig. 1.8, left. Secondly, given sufficiently many such correspondences, we can solve for the rigid body transformation which aligns the 3D model with the input image, see Fig. 1.8, right. This rigid body transformation is the object pose we are looking for.

The question of how to calculate a pose estimate given image-object correspondences has been extensively studied in the earlier years of computer vision. For example, in image-based 3D reconstruction, the calculation of the camera pose (which is the inverse of the object pose) based on estimated correspondences is a fundamental building block [HZ04]. Many algorithms exist that exploit principles of geometry and camera projections. For example, given at least three 3D-3D correspondences, the pose which minimizes the mean-squared error between the aligned, corresponding points can be calculated in closed form with the Kabsch algorithm [Kab76].

In this thesis, we aim at exploiting the geometric knowledge that we have about the task of object pose estimation while learning aspects of the problem which are too complex to model explicitly. Learning the alignment of correspondences, e.g. the Kabsch algorithm, is nonsensical because a machine learning model with limited capacity will probably not generalize as well as an analytical solution. Also, it would require exponential amounts of data to train. A learned model could potentially adapt to systematic errors in the correspondence prediction stage. But even if correspondences are contaminated with outlier predictions, i.e. blatant errors in matching, efficient algorithms from robust optimization, such as Random Sample Consensus (RANSAC), often find a good solution.

On the other hand, predicting correspondences between an image and an object is extremely difficult because of the visual variabilities discussed in Sec. 1.1.2. However, the quality of correspondences is key to accurate and robust pose estimation. In the earlier days of computer vision, researchers tried to hand-engineer image features with the goal to identify objects under varying viewing angles, lighting, etc. Although they were successful in

some restricted scenarios, see also Sec. 1.3, starting from the 2010s, learned features have proven increasingly superior. Thus, in this work, we aim to *learn* to predict correspondences. In particular, we show how to learn to regress for each pixel of an input image the location on the object's surface. For this purpose, we introduce a dense correspondence representation, *object coordinates*, named after the coordinate reference frame of the object, see Fig. 1.9. Based on object coordinate predictions, we find the object pose using robust, geometric optimization. Throughout this thesis, we show how this correspondence prediction can be learned in a supervised fashion from object coordinate ground truth, or from 6D pose ground truth in an end-to-end fashion.

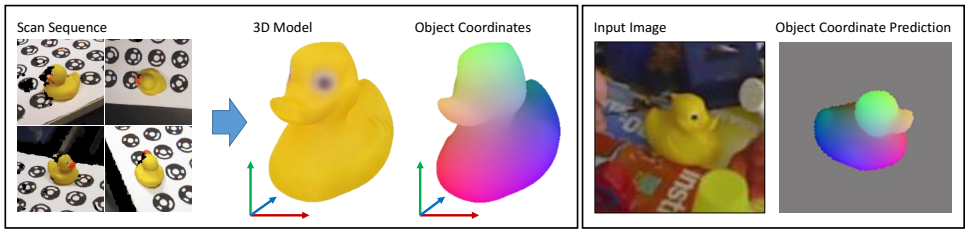

Figure 1.9.: **Object Coordinates. Left:** A 3D model of an object, e.g. created by 3D scanning, defines a coordinate frame, illustrated by three colored axes, specific to the object. Each point on the object surface has a unique coordinate in this frame, a so called *object coordinate*. Throughout this thesis, we visualize object coordinates by mapping their X/Y/Z components to the RGB cube. **Right:** We show a dense object coordinate prediction for each pixel of an input image. Note that each prediction (i.e. each pixel) encodes a correspondence between the input image and the 3D model of the object. Gray pixels in the prediction signify that no correspondence exists.

By learning to predict object coordinates, we can be robust to many sources of visual variability, like lighting changes or occlusion, just by adding them to the training set. The learned model can either develop invariant features, or remember the object appearance under different image conditions. The learned model can also adapt to the specific objects we are interested in. This is opposed to handcrafted features, which, for practical reasons, had to be general purpose solutions. Furthermore, the model can learn to optimally exploit different input modalities. For example, it can make a specific trade-off between color and depth cues based on training data, in case RGB-D images are given. Learning correspondence prediction makes it possible to estimate poses even of difficult, e.g. texture-less, objects under difficult imaging conditions.

In our system, object coordinates are predicted for each pixel based on its local image neighborhood, i.e. based on image patches. While this restricts the available information, depending on the patch size, it has large advantages in terms of generalization capabilities. For example, partial object occlusion will affect some patches but not others, for which object coordinates can still be predicted reliably. Since only a few correspondences suffice to calculate a pose estimate, our system can handle cases where only a small object area remains visible. The same line of argument applies to reflections and, to some extent, to lighting changes. If some local areas are not affected by a specific distortion, accurate poses can still be estimated.

Image patches can be ambiguous, e.g. in texture-less areas or for objects with repeating

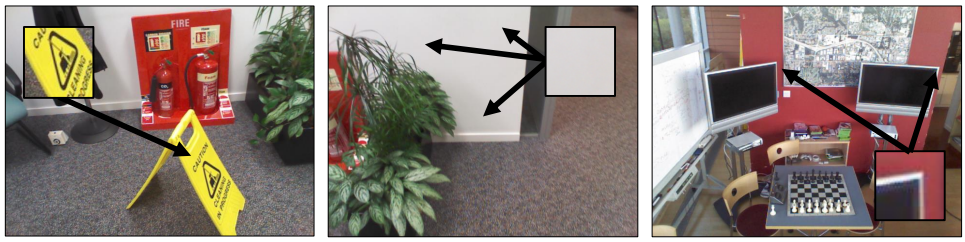

Figure 1.10.: **Patch Ambiguity.** We show three examples of possible correspondences between an image patch, shown as an inlay, and an input image. **Left:** A unique case where we observe only one possible occurrence of the image patch. **Center:** An ambiguous case where the input patch could fit any position on a texture-less area. **Right:** An ambiguous case where the input patch could fit two positions because of a repeating structure.

structures resp. multiple identical parts, see Fig. 1.10. This is a problem when predicting object coordinate point estimates, i.e. when the system is restricted to predict unique correspondences. In Chapter 3, we describe how to predict multi-modal distributions of object coordinates instead of point estimates. The geometric pose optimization can take all uncertainties into account, and hence resolve ambiguities.

1.2.1. CONTRIBUTIONS

In the following, we give the main contributions of this thesis. Note that there are many specific, technical contributions listed in the introductory sections of Chapters 2 to 4.

- We introduce an approach to object pose estimation which combines the benefits of machine learning with principles from geometry.
- We demonstrate the versatility and scalability of our pipeline on a wide range of variants of the task: We handle RGB-D and RGB inputs, we support textured and texture-less objects, and we show results for object pose estimation as well as camera localization.
- We show how to learn specific components of a geometric pipeline individually and end-to-end. For the latter purpose, we introduce a differentiable version of the Random Sample Consensus (RANSAC) algorithm, called Differentiable Sample Consensus (DSAC).

The variants of our pipeline outperformed the respective state-of-the-art at the time of publication. Me made most of our code publicly available to the research community[1], and released two new datasets (see Appendix A.2.2 and A.2.3).

[1] Binaries of the system of Chapter 2 and source code of the systems of Chapters 3 and 4 can be found online.

1.2.2. LIST OF PUBLISHED RESEARCH PAPERS

The following three papers are discussed in detail in the remaining chapters of this thesis.

1. Learning 6D Object Pose Estimation using 3D Object Coordinates
   Eric Brachmann, Alexander Krull, Frank Michel, Stefan Gumhold, Jamie Shotton, Carsten Rother
   ECCV 2014 (Poster Presentation)

2. Uncertainty-Driven 6D Pose Estimation of Objects and Scenes from a Single RGB Image
   Eric Brachmann, Frank Michel, Alexander Krull, Michael Ying Yang, Stefan Gumhold, Carsten Rother
   CVPR 2016 (Poster Presentation)

3. DSAC - Differentiable RANSAC for Camera Localization
   Eric Brachmann, Alexander Krull, Sebastian Nowozin, Jamie Shotton, Frank Michel, Stefan Gumhold, Carsten Rother
   CVPR 2017 (Oral Presentation)

We contributed to the following, additional papers associated with 6D pose estimation. However, these works will not be discussed in this thesis.

4. 6-DOF Model-Based Tracking via Object Coordinate Regression
   Alexander Krull, Frank Michel, Eric Brachmann, Stefan Gumhold, Stephan Ihrke, Carsten Rother
   ACCV 2014 (Oral Presentation, Honorable Mention Demo Award)

5. Pose Estimation of Kinematic Chain Instances via Object Coordinate Regression
   Frank Michel, Alexander Krull, Eric Brachmann, Michael Ying Yang, Stefan Gumhold, Carsten Rother
   BMVC 2015 (Oral Presentation)

6. Learning Analysis-by-Synthesis for 6D Pose Estimation in RGB-D Images
   Alexander Krull, Eric Brachmann, Frank Michel, Michael Ying Yang, Stefan Gumhold, Carsten Rother
   ICCV 2015 (Poster Presentation)

7. Random Forests versus Neural Networks - What's Best for Camera Relocalization?
   Daniela Massiceti, Alexander Krull, Eric Brachmann, Carsten Rother, Philip H.S. Torr
   ICRA 2017 (Oral Presentation)

8. PoseAgent: Budget-Constrained 6D Object Pose Estimation via Reinforcement Learning
   Alexander Krull, Eric Brachmann, Sebastian Nowozin, Frank Michel, Jamie Shotton, Carsten Rother
   CVPR 2017 (Poster Presentation)

9. Global Hypothesis Generation for 6D Object Pose Estimation
   Frank Michel, Alexander Kirillov, Eric Brachmann, Alexander Krull, Stefan Gumhold, Bogdan Savchynskyy, Carsten Rother
   CVPR 2017 (Spotlight Presentation)

The following papers are concerned with topics different from 6D pose estimation but were published during the work on this thesis.

10. Feature Propagation on Image Webs for Enhanced Image Retrieval
    Eric Brachmann, Marcel Spehr, Stefan Gumhold
    ICMR 2013 (Oral Presentation)

11. Simplified Authentication and Authorization for RESTful Services in Trusted Environments
    Eric Brachmann, Gero Dittmann, Klaus-Dieter Schubert
    ESOCC 2012 (Oral Presentation)

1.3. RELATED WORK

In the early 2000s, sparse feature-based methods enabled reliable object detection and pose estimation from RGB images [Low04, GL06]. However, these methods required the objects to be sufficiently textured such that a feature detector could identify reliable sparse feature points. During this period, research focused on improving reliability [YM11] and scalability [NS06, PCI+07] of such methods but general interest in object pose estimation was somewhat limited.

In 2010, the Kinect [WA] depth sensor was introduced for the Microsoft Xbox 360 gaming console. Alongside the success in the consumer marked, researchers started to adopt the device as an affordable RGB-D camera. Among others, the Kinect sparked renewed interest in the problem of object pose estimation [HMS+]. The additional depth channel enabled pose estimation of texture-less objects which were previously difficult to handle. At the same time, machine learning methods became increasingly popular within computer vision with general function approximators like random forests or CNNs pushing accuracy and robustness of object pose estimation. Starting 2015, research focused on difficult cases of object pose estimation from RGB-D images, like objects under severe occlusion [KBM+15, MKB+17, HLRK16] or extremely ambiguous objects [HHO+17]. At the same time, some authors tried to transfer the success of object pose estimation from RGB-D input back to the RGB only setup [CRV+15, WL15, KHKH16].

In the review below, we focus on techniques that specifically address the detection of instances of rigid objects in cluttered scenes and simultaneously infer their 6D pose. We group methods coarsely in four categories: correspondence-based approaches, template-based approaches, voting-based approaches and methods that directly regress the object pose from the input.

1.3.1. CORRESPONDENCES-BASED APPROACHES

Methods within this category follow a two-staged procedure: Firstly, they establish correspondences between the input image and the object. Usually, these correspondences are predicted based on local support, e.g. based on image patches. Secondly, they search for a pose estimate which align the correspondences minimizing some error measure. Usually, the pose search is guided by geometric constraints given by the correspondences. Robust optimization techniques like Random Sample Consensus (RANSAC) or clustering make the pose estimation robust to correspondence prediction errors, see Fig. 1.11. Since only a small

number of correspondences (at least three for RGB-D images and four for RGB images) suffice to recover the pose, correspondence-based methods are typically very robust w.r.t. to occlusion.

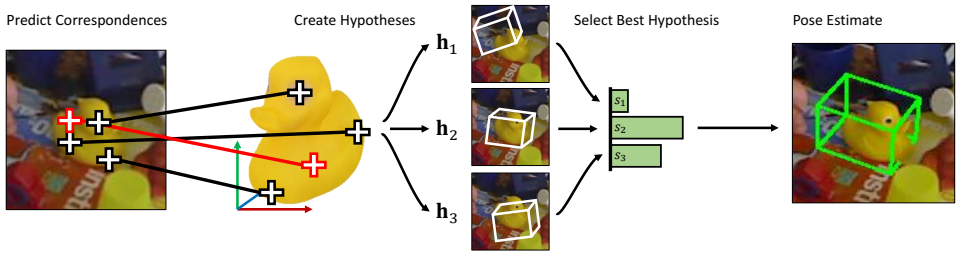

Figure 1.11.: **Sparse Feature-Based Pose Estimation.** From left to right: Sparse features are extracted for an input image and matched to the 3D model of the object. The set of correspondences is likely to contain outliers, i.e. wrong matches shown in red, due to various error sources. For robustness, several hypotheses $\mathbf{h}$ about the object pose are created, using different subsets of correspondences. The quality of hypotheses is assessed w.r.t. to some scoring function s. The final pose estimate is the hypothesis with the largest score, potentially after further refinement. In this thesis, we follow a similar approach, but learn to predict dense correspondences instead of relying on sparse feature detection and matching.

Many variants of correspondence-based approaches rely on sparse feature detectors. These extract points of interest (often scale-invariant) from the image, describe them with local descriptors (often affine and illumination invariant), and match them to a database. For example, Lowe [Low01] used SIFT (scale invariant feature transform) features and clustered images from similar viewpoints into a single model. Another example of a system based on sparse feature clustering is the MOPED framework [MCS10]. Sparse techniques have been shown to scale well to matching across vast vocabularies [NS06, PCI+07], e.g. Sattler et al. demonstrate city scale 6D camera localization based on sparse feature matching [SLK16, STS+17]. Sparse feature pipelines have also been combined with machine learning to improve interest points [RPD10, HSK12], descriptors [WHB09], and matching [LF06, OCLF10, BRF12]. Specifically, Yi et al. [YTLF16] encoded the original SIFT pipeline of Lowe [Low01] within a CNN architecture which they call LIFT (learned invariant feature transform). It consists of feature extraction, rotation estimation and descriptor extraction components which can be learned from data for increased matching performance. However, at time of publication, learned sparse features have not yet been used for object pose estimation. A study by Sattler et al. [SHSP17] has shown that the increased matching performance of learned features does not necessarily lead to increased accuracy in higher level tasks like 3D reconstruction.

Despite their popularity, a major limitation of traditional sparse approaches is that they require sufficiently textured objects. For objects with little or no texture, a feature detector will fail to identify stable feature points. As an alternative, Crivellaro et al. [CRV+15] train a CNN to detect object specific anchor points, a variant of sparse features, like the corners of box-shaped objects. Other authors dispense the feature detector and instead establish correspondences densely for each pixel of the input image. Object coordinates, the approach

presented in this thesis, belongs to this strain of dense correspondence-based methods.

While individual correspondences predicted for areas with little or no texture are likely to be inaccurate, in large numbers they can still contribute to a good solution. In case an RGB-D sensor is available, the additional depth channel can increase the expressiveness of correspondences in texture-less areas. Shape cues in the depth channel are useful for correspondence matching, and allow to utilize stronger geometric constraints during pose estimation. For example, Zach et al. [ZPSP15] use local voxel occupancy information to establish dense correspondences between an RGB-D image and an object. A geometric consistency check discards a large number of outlier correspondences. The remaining correspondences yield a pool of pose hypotheses which are ranked and refined using an ICP-inspired procedure. In our method, we instead *learn* to predict correspondences which are reliable enough to facilitate pose optimization using RANSAC. While the method of Zach et al. [ZPSP15] is limited to RGB-D inputs, our method supports RGB-D and RGB inputs (see Chapter 2 and 3, respectively). Given all available image modalities, it will discover the most appropriate image features to exploit, autonomously.

Scene Coordinate Regression Our method is inspired by the Scene Coordinate Regression Forests (SCoRF) of Shotton et al. [SGZ+13] introduced for camera localization from RGB-D images. As described in Sec. 1.1.1, camera localization is a special case of object pose estimation where the object of interest occupies the complete input image. Shotton et al. learn a mapping from image patches to world coordinates (resp. *scene* coordinates), i.e. 3D coordinates in the reference frame of the scene. Apart from scene coordinates, Shotton et al. also calculate *camera* coordinates, i.e. 3D coordinates in the reference frame of the camera, for each pixel using the depth channel of the input image. The combination of calculated camera coordinates and predicted scene coordinates yields dense 3D-3D correspondences. Based on these correspondences, Shotton et al. fit a rigid body transform using locally refined pre-emptive RANSAC. See Fig. 1.12 for an overview of the SCoRF pipeline.

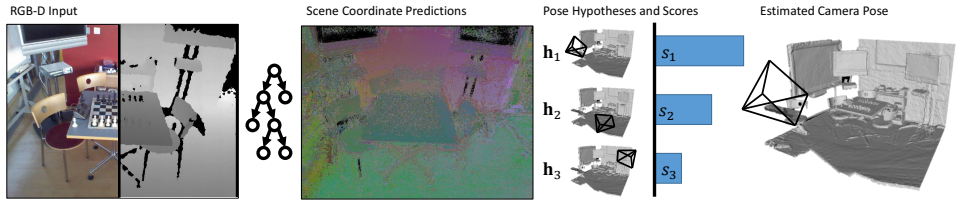

Figure 1.12.: **SCoRF Pipeline [SGZ+13] for Camera Localization.** From left to right: A random forest predicts dense scene coordinates for an RGB-D input image. Hypotheses of camera poses are created, scored and refined in a pre-emptive RANSAC schema. The best hypothesis is returned as the estimated 6D camera pose. In this thesis we generalize the SCoRF framework for object pose estimation and extend it in various ways.

The SCoRF pipeline cannot be directly applied to object pose estimation. If the object of interest occupies only a small fraction of the input image most scene coordinate predictions, specifically those made for the unknown background, will be wrong. This results in an adverse outlier-inlier ratio which RANSAC is unable to handle. In this thesis, we extend the concept of scene coordinate regression to that of object coordinate regression. Jointly with the correspondence prediction, we solve a classification problem that allows us to restrict

RANSAC hypothesis sampling to pixels which likely belong to an object of interest. Thus, we are able to estimate accurate 6D poses of small objects in large, unknown scenes. In contrast to Shotton et al., we also learn object coordinate representations for multiple objects simultaneously leading to a very scalable system (see Chapter 3). Since object pose estimation is a generalization of camera localization, our pipeline can be applied to both problems. Unlike Shotton et al., we demonstrate camera localization from RGB-D *and* RGB images, see Chapter 3. In Chapter 4, we furthermore introduce a modernized variant of the original SCoRF pipeline which can be learned end-to-end with the novel concept of differentiable RANSAC.

The SCoRF pipeline has been extended in multiple follow-up works. Guzman-Rivera et al. [GRKG+14] train a random forest to predict diverse scene coordinates in order to resolve scene ambiguities. Cavallari et al. [CGL+17] show how to adapt pre-trained regression forests for a new scene on the fly, given RGB-D input. Valentin et al. [VNS+15] introduce uncertain correspondence predictions by modelling full distributions of scene coordinates. The uncertainty information leads to a better pose refinement and thus higher pose accuracy. However, their approach cannot be applied to RGB images because it relies on the calculation of camera coordinates from a depth channel. We developed the concept of uncertain correspondence prediction in parallel to Valentin et al. Going beyond their work, we show how to utilize this uncertainty also for RGB input, i.e. even if no camera coordinate can be calculated, see Chapter 3.

Finally, there are approaches for object class detection that use a similar idea as our 3D object coordinate representation. One of the first systems is the 3D LayoutCRF [HRW07] which considers the task of predicting a dense, discrete part-labeling, covering the 3D rigid object, using a decision forest, though they did not attempt to fit a 3D model to those labels. After that, the Vitruvian Manifold [TSSF12] was introduced for human pose estimation. The authors predict a dense, continuous human part labeling based on a segmented depth image of a person. The segmentation is assumed to be given, e.g. by a simple background subtraction on the depth channel. Using the label prediction, the authors solve for the high-dimensional human pose using a robust energy minimization. This work inspired the SCoRF pipeline discussed above in detail. Similar to SCoRF, the Vitruvian Manifold system is limited to RGB-D inputs.

1.3.2. TEMPLATE-BASED APPROACHES

A popular alternative to correspondences-based approaches are templates. A template is a global descriptor that captures the appearance of an object from a certain view point, often in a way that is robust to illumination changes. In practice, a set of templates is extracted for a discrete subset of all possible viewing angles and scales to describe the object as a whole. During test time, each template from the set is compared to the input image at all possible 2D locations. A matching score exceeding a pre-defined threshold will yield a candidate detection. Candidate detections are often pruned, e.g. by further consistency checks and non-maximum suppression. Because each template is associated with a specific view point, a template match does, additionally to 2D location, usually also contain information about object scale and rotation. Therefore, a template match translates to an estimate of the 6D object pose. However, because the set of templates represents only a discretized set of object poses, this estimate is limited in its accuracy. Hence, most template-based methods apply some kind of post-processing, e.g. ICP refinement, to improve the final result. Usually, templates do not require an extensive training phase and have been shown to

be very efficient during test time. Because templates are holistic object descriptions, they tend to be very sensitive to partial occlusion.

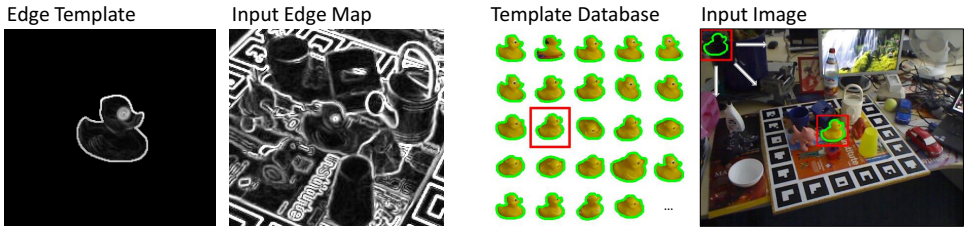

Figure 1.13.: **Template-Based Approaches. Left:** Early works in template matching were concerned with defining robust distance functions to compare clean template edge maps with noisy edge maps of input images [HKR93]. If the template can be matched at a particular image position, the object is localized. **Right:** Templates can be extracted for a dense sampling of viewpoints of an object to create a template database. At test time, all templates a moved over an input image. If the matching score exceeds a threshold, the object is localized, and the viewpoint associated with the matching template can be returned as a pose estimate.

Research on templates dates back far, with methods like [HKR93] or [FTVG06, FJS09] relying on matching contours for object detection. The success of these methods was limited, because contours tend to be unstable when extracted from natural images. Instead, templates based on image gradients have been widely used, often based on the HOG (histogram of oriented gradients) descriptor initially introduced for pedestrian detection [DT05]. For example, Felzenszwalb et al. [FGMR10] describe a HOG-based deformable parts model (DPM) for object class detection which won the PASCAL VOC 2009 detection challenge [EVGW+a].

Although object class templates like HOG or DPM can also detect rigid object instances, they are constructed to be somewhat invariant to object appearance. While this allow them to handle class specific variabilities it also limits their ability to discriminate exact view points of objects as is needed for accurate 6D object pose estimation. That motivated authors to develop instance specific template methods with the LINE templates of Hinterstoisser et al. [HCI+12]. LINE templates come in different variants: LINE2D templates rely on stable RGB gradients for object detection from RGB images. LINE3D templates rely on stable normals for object detection from depth images. LINEMOD templates utilize both cues for object detection from RGB-D images. The templates store and compare only quantized, binarized gradients and normals which makes the computation of matching scores extremely fast. This allows Hinterstoisser et al. to represent an object with 2000+ templates, covering different combinations of scale and viewing angles, and still run in real time. LINEMOD templates have been successfully applied to object pose estimation from RGB-D input in [HLI+12], where the template match gave an initial pose estimate further refined using ICP.

By relying on gradient and normal information, LINE templates are robust to some changes in object appearance. For example, in [HLI+12] templates were extracted using renderings of objects, and applied to real images during test time. This is an advantage in applications were CAD models of objects exist because images with accurate ground truth poses can be generated easily in large amounts. We extensively compare our method to LINEMOD in Sec. 2.3. As we will show, our method is less sensitive to illumination changes and partial

object occlusion. At the same time, we can also learn our object coordinate representation from synthetic (i.e. rendered) training data.

Although the matching function of templates like LINE is very fast to compute, the number of templates needed scales linearly with the number of objects and with the extent of the pose space to cover. In [HLI+12], the discretized pose space was limited to a few discrete distances (resp. scales) and the upper view hemisphere. If a large range of scales and all 360° of viewing angles should be supported, the number of required templates, multiplied by the number of objects, quickly grows tremendously. The scalability issues of template-based methods have been addressed by some authors. Konishi et al. [KHKH16] arrange templates of increasing resolution in an hierarchical pose tree for fast inference. Rios-Cabrera and Tuytelaars [RCT13] use discriminative learning to construct a LINEMOD template cascade which considerably speeds up template matching. Kehl et al. [KTN+16] utilize a LINEMOD inspired hashing function for template matching, resulting in object pose estimation which scales sub-linearly in the number of objects. Similarly Hodan et al. [HZL+15] find candidate templates by hashing features of pixel triplets which also results in sub-linear scaling of the method. As we will demonstrate in Sec. 2.3.6 and Sec. 3.3.2, our method also has an empirical complexity which is sub-linear in the number of objects and in the pose space covered.

While most templates discussed so far have been more or less handcrafted, Wohlhart et al. [WL15] learn an object descriptor from data. The descriptor is the output of a CNN that captures pose and object instance specific appearance in a holistic fashion, hence it is comparable to a template. The CNN is learned such that the descriptors exhibit the following properties: Two descriptors of the same object instance and similar poses should be similar, in any other case they should be dissimilar. At test time, descriptors of the input image and an object database are matched using a nearest neighbor search in Euclidean space. The method can be applied to RGB-D or RGB inputs, and the descriptors can generalize to unseen objects to some extent. While the method has been compared favorably to LINEMOD in a restricted experimental setup, its accuracy and performance within a full object detection or pose estimation pipeline remains unclear.

1.3.3. VOTING-BASED APPROACHES

The notion of voting-based approaches is that local evidence in the image restricts the possible outcome of the desired output. Hence, every image patch can cast a *vote* about the output. For example, if a certain part or feature of an object is detected in the image, this massively restricts the possible position of the object center. In case the object is rigid, the position of the object center relative to an object feature depends only on the object rotation. Furthermore, not all object features are visible under all rotations, and affine distortions of an object feature can yield further information about the object pose. Although individual votes are weak, i.e. noisy resp. uncertain, in summation they can yield a strong prediction, see Fig. 1.14, left.

In the generalized Hough voting scheme, all image patches cast a vote in some quantized prediction space (e.g. 2D object center and scale), and the center of the cell with the most votes is taken as the final prediction. Because the prediction space is quantized, the final prediction is limited in accuracy. As an alternative, clusters of votes can be found in a continuous prediction space using e.g. mean-shift. In this case. the center of the cluster is taken as the final prediction. Voting-based approaches are usually restricted to low dimensional output predictions. In high-dimensional spaces, votes tend to scatter and not form clear clusters.

In [SBXS10, GYR+11], Hough voting was used for object detection, and was shown able to predict coarse object poses. In our work, we borrow an idea from Gall et al. [GYR+11] to jointly train an objective over both Hough votes (resp. object coordinates in our case) and object segmentations. However, in contrast to [GYR+11] we found a simple joint distribution over the outputs (in our case 3D object coordinates and object ID labels) to perform better than the variants suggested in [GYR+11].

Tejani et al. [TTKK14] train a Hough forest for 6D pose estimation from an RGB-D image. Each tree in the forest maps an image patch to a leaf which stores a set of 6D pose votes. Tejani et al. adapt LINEMOD templates [HCI+11], discussed above, to be used as features in each forest split node. Tejani et al. cluster pose votes in three subsequent stages to reduce the dimensionality of the problem: Firstly, they cluster according to the 2D projections of the object center, secondly, according to the 3D translation component and, thirdly, according to the 3D rotation component. Doumanoglou et al. [DKMK16] improved the method by using auto-encoder features in the Hough forest instead of LINEMOD features. Kehl et al. [KMT+16] follow a similar strategy but instead of using a random forest they match image patches to a patch codebook via a nearest neighbor search based on auto-encoder features. Each entry in the codebook is associated with 6D pose votes, which are subsequently clustered in 3 stages similar to [TTKK14]. Our method is conceptually similar to [TTKK14, DKMK16, KMT+16] but instead of casting 6D votes, each image patch instead makes a 3D continuous prediction about only its *local* correspondence to the object surface. This massively reduces the search space, and, for learning a discriminative prediction, allows a much reduced training set since each point on the surface of the object does not need to be seen from every possible angle. We show how these 3D object coordinates can efficiently drive a subsequent model fitting stage to achieve a highly accurate 6D object pose.

Drost et al. [DUNI10] take a two-stage voting approach using point pair features to recover the 6D pose of objects from a depth image. A point pair feature contains information about the distance and the normals of two arbitrary 3D points. In stage one, a reference point of the input image is selected, and possible point pair features are calculated exhaustively by pairing it with all other input points. All these point pair features cast votes for a location of the reference point on the object surface in a 2D voting space. Each peak in the voting space leads to a candidate match between the input and the object. Because a candidate match consists of pairs of points with normal information, it identifies a unique 6D pose of the object. However, due to quantization and sampling artifacts this candidate pose is only a coarse estimate. The voting procedure of stage one is repeated for every input point as the reference point yielding a large number of candidate poses. In a second stage, all pose candidates are clustered and the average pose of a cluster is given as the final output. The method of Drost et al. compared unfavorably to more recent approaches like LINEMOD [HLI+12]. Due to the exhaustive computation and matching of point pair features, the method is susceptible to large amounts of clutter in the input. This also makes the method computationally demanding. These shortcomings were addressed by Hinterstoisser et al. [HLRK16], improving robustness, accuracy and runtime considerably. Foremost, Hinterstoisser et al. avoid the exhaustive pairing of input points by sampling point pairs according to object size. Furthermore, they use soft votes to account for sensor noise, and add post processing to refine final pose estimates. While performing very well, the method is limited to depth inputs, and has been outperformed by Michel et al. [MKB+17] using an approach based on the object coordinate concept presented in this thesis.

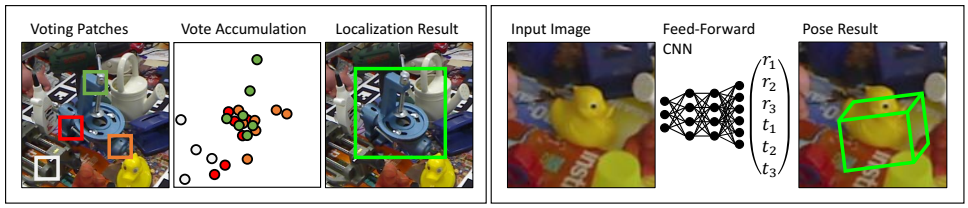

Figure 1.14.: **Voting-Based Approaches and Pose Regression. Left:** Four colored patches vote for the 2D center position of the bench vise object. Every patch casts multiple discrete votes illustrated by colored dots. Votes of the green, red and orange patch form a clear cluster while the white background patch casts random votes that do not contribute to any maximum. The vote cluster yields the 2D position of the object. In high dimensional output spaces votes tend to scatter, and no maximum can be found. **Right:** A CNN was trained to directly regress rotation parameters r and translation parameters t from an input image. Because of limited training data, direct pose regression is usually inaccurate.

1.3.4. POSE REGRESSION

While the methods explained so far rely on multi-staged strategies to estimate object poses from input images, an alternative is to learn the immediate mapping from an input image to a parametric representation of the pose. This idea has been predominately explored for camera localization. For example, Kendall et al. [KGC15] trained a CNN, called PoseNet, to directly regress the 6D pose vector of a scene from an RGB image. Walch et al. [WHL+16] proposed a variant of PoseNet with a LSTM (long short-term memory) component, and Kendall and Cipolla [KC17] evaluated the impact of different loss functions for training PoseNet. However, while PoseNet and its variants are fast and easy to train they fail to predict accurate poses. Specifically, they are by one order of magnitude less accurate than the approaches we present in Chapter 3 and 4. Also, they are less accurate than traditional, handcrafted camera localization approaches based on sparse features. The unconstrained architecture of PoseNet dismisses all geometric knowledge about the task, and the amount of training data is insufficient to discover these concepts purely by end-to-end learning.

Inspired by PoseNet, Doumanoglou et al. [DBKK16] train a CNN to regress the object rotation directly from a window centered at the object position, see Fig. 1.14, right. They conclude, that a direct regression is less accurate than a nearest neighbor matching of mid-level features to a rotation database.

1.4. OUTLINE OF THE THESIS

The remainder of this thesis is structured as follows: Chapter 2 introduces our system for pose estimation of a single object instance from an RGB-D image. Here, we show how to learn to regress object coordinates in a supervised fashion from object coordinate ground truth. In Chapter 3, we extend this system in multiple ways to enable efficient pose estimation of multiple objects from single, RGB images, i.e. without the need for a depth channel. In Chapter 4, we introduce DSAC, Differentiable Sample Consensus, which enables us to learn all components of our pipeline in an end-to-end fashion, i.e. we learn to predict object coordinates that minimize the pose error on the training set. We discuss the status of object pose estimation under the light of this thesis in Chapter 5. This includes open research questions.

2. 6D POSE ESTIMATION USING OBJECT COORDINATES

Contents

2.1. INTRODUCTION

In this chapter, we consider a basic scenario of object pose estimation where the input is a single RGB-D image, and the object in question is a rigid instance that is known to be visible in the image. The ultimate goal is to design a system that is fast, scalable, robust and highly accurate and works well for generic objects (both textured and texture-less) present in challenging real-world settings, such as cluttered environments and with variable lighting conditions. The work associated with this chapter was published in [BKM+14].

As discussed in Sec. 1.3, for many years the main focus in the field of detection and 2D/6D pose estimation of rigid objects has been limited to objects with sufficient amount of texture. Most systems used a sparse representation of local features, either handcrafted, e.g. SIFT features [Low01], or trained from data, e.g. LIFT features [YTLF16]. These systems run typically a two-staged pipeline: Firstly, putative sparse feature matching, and secondly, geometric verification of matched features.

Later, researchers have started to consider the task of object instance detection for texture-less or texture-poor rigid objects, e.g. [HCI+12, HLI+12, RCT13]. For this particular challenge

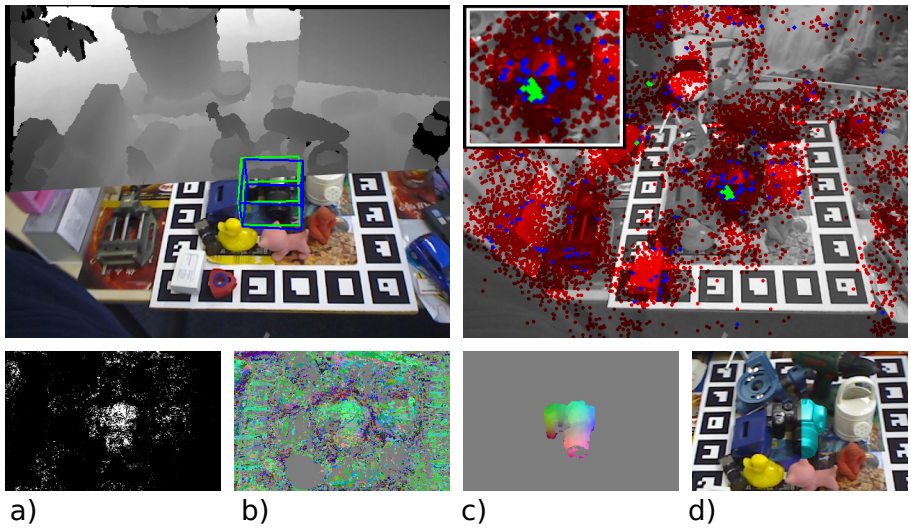

Figure 2.1.: **Overview of our System. Top left:** Results on an RGB-D test image. The upper half shows the depth channel and the lower half shows the RGB image. The estimated 6D pose of the query object (a *Camera*) is illustrated with a blue bounding box, and the respective ground truth with a green bounding box. **Top right:** Visualization of the algorithm's search for the optimal pose, where the inlet is a zoom of the center area. The algorithm optimizes our scoring function in a RANSAC-like fashion over a large, continuous 6D pose space. The 6D poses, projected to the image plane, which are visited by the algorithm are color coded: *red poses* are disregarded in a very fast geometry check; *blue poses* are evaluated using our scoring function during intermediate, fast sampling; *green poses* are subject to the most expensive refinement step. **Bottom, from left to right: (a)** Probability map for the query object, **(b)** predicted 3D object coordinates from a single tree mapped to the RGB cube, **(c)** corresponding ground truth 3D object coordinates, **(d)** overlay of the *3D model in blue* onto the test image, rendered according to the estimated pose.

it has been shown that template-based techniques are superior to sparse feature-based approaches. Nevertheless, template-based techniques are less robust with respect to occlusion because they encode the object in a particular pose with one *global* feature. In contrast to this, sparse feature-based representations for textured objects are *local*, and hence robust to occlusion.

Our approach is motivated by work in the field of articulated human pose estimation [TSSF12] and camera localization [SGZ+13] from a single RGB-D image. Both works do not predict the human resp. camera pose directly from an image, but first regress an intermediate representation. Each pixel in the image votes for a continuous coordinate in a canonical reference frame. In the next step, a *geometric validation* is performed, which optimizes for the desired pose. In spirit, both approaches are akin to the two-staged pipeline of traditional, sparse feature-based techniques but with densely learned features.

Our system is based on these ideas presented in [TSSF12, SGZ+13] and applies them

to the task of estimating the 6D pose of specific objects. An overview of our system is presented in Fig. 2.1. We cannot apply [TSSF12, SGZ+13] directly since we additionally need an object segmentation mask. Note that the method in [TSSF12] can rely on a pre-segmented human shape, and [SGZ+13] does not require a segmentation because the scene occupies the whole image. To achieve object pose estimation, we jointly predict a dense 3D object coordinate labeling and a dense object ID labeling. Another major difference to [SGZ+13] is a guided sampling scheme to avoid the generation of many false hypotheses in the RANSAC-based pose optimization.

Contributions. The contributions of this chapter are:

- We introduce the concept of object coordinates, a dense, continuous correspondence representation, for object instance pose estimation. We train a discriminative model which is able to regress object coordinates and segmentation of multiple objects, simultaneously.
- We formulate a new scoring function which assesses the plausibility of a pose hypothesis w.r.t. the observed image and the discriminative predictions of a random forest. The scoring function utilizes knowledge about the specific object in the form of a 3D model. We utilize the scoring function within a RANSAC-based pose optimization schema.
- We compare with state-of-the-art template-based approaches, and show that our method is on par regarding pose estimation of texture-less object instances. Furthermore, we show superior accuracy on images with severe occlusion and drastic lighting changes.
- We published a new dataset of 20 objects, textured and texture-less, featuring three different lighting conditions. The dataset consists of approximately 10k annotated images, see Appendix A.2.3.
- We published a set of approximately 10k new pose annotations for the dataset of Hinterstoisser et al. [HLI+12] for heavily occluded objects, see Appendix A.2.2.

2.2. METHOD

Given an input image I, we aim at estimating the pose $\mathbf{h}_c$ of an object c, consisting of a 3D rotation $\boldsymbol{\theta}_c$ and a 3D translation $\mathbf{t}_c$. We first describe our decision forest that jointly predicts both 3D object coordinates and object instance probabilities for each pixel of the input image. Then, we will discuss our pose hypothesis scoring function which is based on the forest output. Finally, we will address our RANSAC-based pose optimization scheme.

2.2.1. OBJECT COORDINATE REGRESSION

We use a single decision forest to classify pixels from an RGB-D image. A decision forest is a set $\mathcal{T}$ of decision trees T^j. Each decision tree is a hierarchical structure which consists of two types of nodes, split nodes and leaf nodes. A split node compares a feature response calculated for the incoming pixel i to a threshold, and, depending on the result, passes the pixel to the left or right child node. Split node features access the local neighborhood of the input pixel i to calculate a response, i.e. our forest operates on image patches. At some

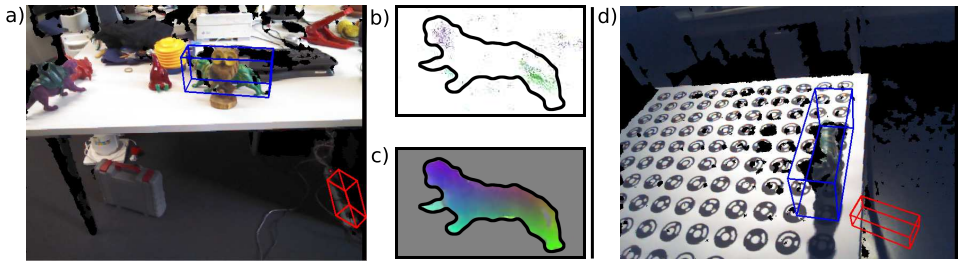

Figure 2.2.: **Pose Estimation Under Severe Occlusion and Lighting Changes.** Our method is able to find the correct pose where a template-based method fails. **(a)** Test image showing a situation with strong occlusion. The pose estimate by our approach is shown in blue. The pose estimated by our re-implementation of the method by Hinterstoisser et al. from [HLI+12] is shown in red. **(b)** The coordinate predictions for *a* from one tree mapped to the RGB cube and weighted by the soft segmentation for that object. **(c)** Ground truth object coordinates for *a* mapped to the RGB cube. **(d)** Test image showing extreme light conditions, different from the training set. Estimated poses are displayed as in *a*.

point, the pixel will reach a leaf node l^j, which stores a prediction. Our forest is trained in a way that allows us to gain information about which object $c \in \mathcal{C}$ the pixel i might belong to, as well as what might be its position on this object. We will denote a pixel's position on the object by $\mathbf{y}_i$ and refer to it as the pixel's *object coordinate*. Each leaf l^j stores a distribution over possible object affiliations $P(c|l^j)$, as well as a set of object coordinates $\mathbf{y}_c(l^j)$ for each possible object affiliation c. The term $\mathbf{y}_c(l^j)$ will be referred to as *coordinate prediction*.

In the following, we describe the training procedure of one randomized decision tree [CS13]. The procedure is the same for all trees of the forest, and consists of two stages. In training stage one, we create the structure of the decision tree, i.e. we select features and thresholds for each split node, and we determine when to terminate growing a tree branch and store a leaf node instead. We also give detailed information on the types of features we use in split nodes. In training stage two, we assign predictions to each leaf node l^j, namely distributions of object affiliations $P(c|l^j)$ and an object coordinate $\mathbf{y}_c(l^j)$ for each object.

For training, we use segmented object images and a set of RGB-D background images. We will treat training images as a combined set of training pixels. Each training pixel i is characterized by its position $\mathbf{p}_i$, its color $\mathbf{x}_i^{\text{rgb}}$, its depth d_i, its object instance label c_i and its object coordinate $\mathbf{y}_i$. Object coordinate $\mathbf{y}_i$ labels will not be used for pixels from background images.

Training of a Decision Tree Structure. Training the tree structure starts at the top node where a feature with parameters $\boldsymbol{\zeta}$ is selected with the goal to reduce the uncertainty in $P(c)$ and $P(\mathbf{y}|c)$ the most, based on the training data. Here, $P(c)$ denotes the distribution over object affiliations c, and $P(\mathbf{y}|c)$ denotes the conditional, continuous 3D distribution over object coordinates $\mathbf{y}$ given an object affiliation c.

The selection of features at each node is based on a split score which should be able to handle well the discrete distribution $P(c)$ and the continuous distribution $P(\mathbf{y}|c)$. We quantize the continuous object coordinate labels $\mathbf{y}$ based on a $5 \times 5 \times 5$ grid to obtain discrete object coordinate labels $\hat{\mathbf{y}}$. The quantization allows us to use the standard information gain

classification objective during training, which has the ability to cope better with the often heavily multi-model distributions $P(\mathbf{y}|c)$ than a regression objective [GSK+11]. We define the information gain over the joint distribution $P(\hat{\mathbf{y}}, c)$ which has potentially $125|\mathcal{C}|+1$ labels, for $|\mathcal{C}|$ object instances and one additional label for a background class. In practice, many bins are empty, and the histograms can be stored sparsely for speed. We denote by $P_\eta(\hat{\mathbf{y}}, c)$ the joint distribution at node η, and by $P_\eta^{\boldsymbol{\zeta},\leftarrow}(\hat{\mathbf{y}}, c)$ resp. $P_\eta^{\boldsymbol{\zeta},\rightarrow}(\hat{\mathbf{y}}, c)$ the distributions in the left resp. the right child of node η, split by parameters $\boldsymbol{\zeta}$. We select parameters $\boldsymbol{\zeta}$ such that the information gain IG in objects and proxy classes is maximized:

$$IG(\eta, \boldsymbol{\zeta}) = \mathcal{H}(P_\eta(\hat{\mathbf{y}}, c)) - \sum_{k \in \{\leftarrow, \rightarrow\}} \Big[\frac{|\eta^k|}{|\eta|} \mathcal{H}(P_\eta^{\boldsymbol{\zeta},k}(\hat{\mathbf{y}}, c)) \Big], \tag{2.1}$$

where $\mathcal{H}(P_\eta(\hat{\mathbf{y}}, c)) = -\sum_c \sum_{\hat{\mathbf{y}}} P_\eta(\hat{\mathbf{y}}, c) \log P_\eta(\hat{\mathbf{y}}, c)$ is the Shannon entropy of the joint distribution, $|\eta|$ is the number of training pixels which arrived at the parent node, and $|\eta^k|, k \in \{\leftarrow, \rightarrow\}$ is the number of pixels split to the left resp. the right child node. We found the suggestion in [GYR+11] to mix two separate information gain criteria, one for $P(c)$ and one for $P(\mathbf{y}|c)$, to be inferior on our data.

The selected feature divides the data to go to the left resp. the right child node, where the feature selection process repeats. This process iterates until a stopping criterion is met. We stop splitting further if a maximum depth has been reached, or not enough training pixels arrived at a node. Due to runtime complexity only a random sub-sampling of both, training pixels and feature parameters $\boldsymbol{\zeta}$, is used. This also introduces variability between trees in the forest, and hence the ability to generalize to unseen data.

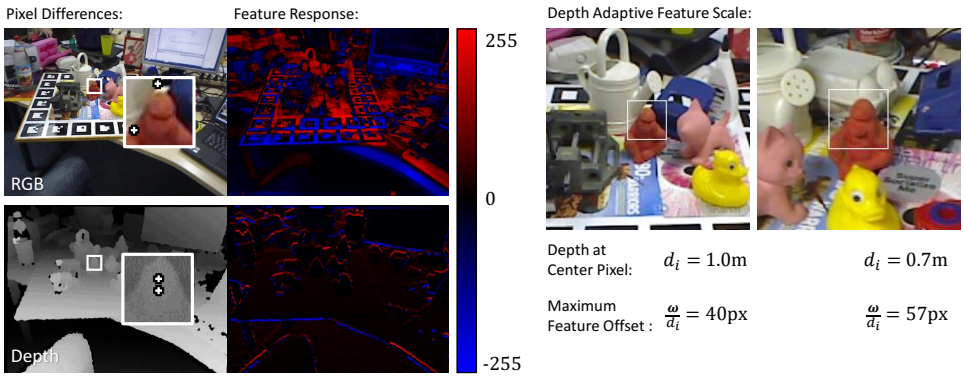

Figure 2.3.: **Depth Adaptive Pixel Difference Features. Left:** Our random forest uses pixel differences on RGB and depth as features. We show the feature response maps for one RGB feature $f_{\text{da-rgb}}$ (top), and for one depth feature $f_{\text{da-d}}$ (bottom). As a magnified inlay, we show on exemplary input patch with feature offsets $\boldsymbol{\omega}_1$ and $\boldsymbol{\omega}_2$ marked by crosses. **Right:** Feature offsets are depth adaptive, i.e. input patches are scale-normalized according to the depth at the center pixel. We mark the patch size for two input images taken with different distance between object and camera. Note how the scale adapted patch covers similar portions of the object.

Split Node Features. The choice of features evaluated in the tree splits is central for accuracy and runtime of the random forest. We experimented with a large number of features, including angles between normals, absolute (L)AB color, Haar features [VJ01], etc. We found that the very simple and fast to compute features from [SGZ+13] performed well, and that adding extra feature types did not appear to give a boost in accuracy but did increase runtime. We assume that the learned combination of simple features in the tree is able to create complex features which specialize for the task defined by the training data and splitting objective. The features in [SGZ+13] consider depth or color differences from pixels in the vicinity of pixel i, and capture local patterns of context, see Fig. 2.3, left. The feature responses can be computed as follows.

$$f_{\text{da-d}}(\boldsymbol{\zeta}, \mathbf{p}_i) = d\left(\mathbf{p}_i + \frac{\boldsymbol{\omega}_1}{d_i}\right) - d\left(\mathbf{p}_i + \frac{\boldsymbol{\omega}_2}{d_i}\right) \tag{2.2}$$

$$f_{\text{da-rgb}}(\boldsymbol{\zeta}, \mathbf{p}_i) = I\left(\mathbf{p}_i + \frac{\boldsymbol{\omega}_1}{d_i}, \gamma_1\right) - I\left(\mathbf{p}_i + \frac{\boldsymbol{\omega}_2}{d_i}, \gamma_2\right) , \tag{2.3}$$

where $I(\mathbf{p}_i, \gamma) = \mathbf{x}_i^{\text{rgb}}[\gamma]$ returns the R, G, or B channel of a pixel according to γ, and $d(\mathbf{p}_i) = d_i$ returns the depth at position $\mathbf{p}_i$. Abbreviations 'da-rgb' and 'da-d' stand for depth adaptive RGB differences and depth adaptive depth differences. $\boldsymbol{\omega}$ indicates a 2D offset. The division by d_i makes the features largely depth invariant, and is similar in spirit to [WCL+08], see also Fig. 2.3, right. In summary, each split node in the forest stores a unique set of parameters $\boldsymbol{\zeta} \subseteq \{\boldsymbol{\omega}_1, \boldsymbol{\omega}_2, \gamma_1, \gamma_2, z, \tau_f\}$, with $z \in \{\text{da-d}, \text{da-rgb}\}$ indicating the type of feature to use. We denote τ_f for the threshold on the feature response that decides whether a pixel goes to the left or the right child of a node. This threshold is also stored per node. As we mentioned above, we use segmented object images for training. If a feature test reaches outside the object mask, we have to model some kind of background to calculate feature responses. In our experiments, we use uniform noise or a simulated plane the object sits on. We found this to work well, and to generalize well to new unseen images. Putting objects on a plane allows the forest to learn contextual information.

Training of Leaf Node Distributions. After constructing the tree structure based on quantized object coordinates, we proceed with stage two of training. We push training pixels from all objects through the tree and record all continuous locations $\mathbf{y}$ for each object c at each leaf l^j. This gives a conditional distribution $P(\mathbf{y}|c, l^j)$ per leaf which we approximate by storing only the mode $\mathbf{y}_c(l^j)$ with most supporting samples. For each object and leaf, we run mean-shift with a Gaussian kernel and a bandwidth of 2.5cm. Note that although we store only one object coordinate mode per object and leaf, the object coordinate prediction per pixel can still be multi-modal by using a forest instead of a single tree. Also note that we did not use quantized object coordinates in training stage two. We furthermore store at each leaf the percentage of pixels coming from each object c to approximate the distribution of object affiliations $P(c|l^j)$ at the leaf. We also store the percentage of pixels from the background set that arrived at l^j, and refer to it as $P(BG|l^j)$. We summarize the forest training procedure in Fig. 2.4.

Using the Forest. At test time, we push all pixels of an RGB-D image through every tree of the forest, thus associating each pixel i with a distribution $P(c|l_i^j)$ and one prediction $\mathbf{y}_c(l_i^j)$ for each tree j and each object c. Here l_i^j is the leaf outcome of pixel i in tree j. The leaf

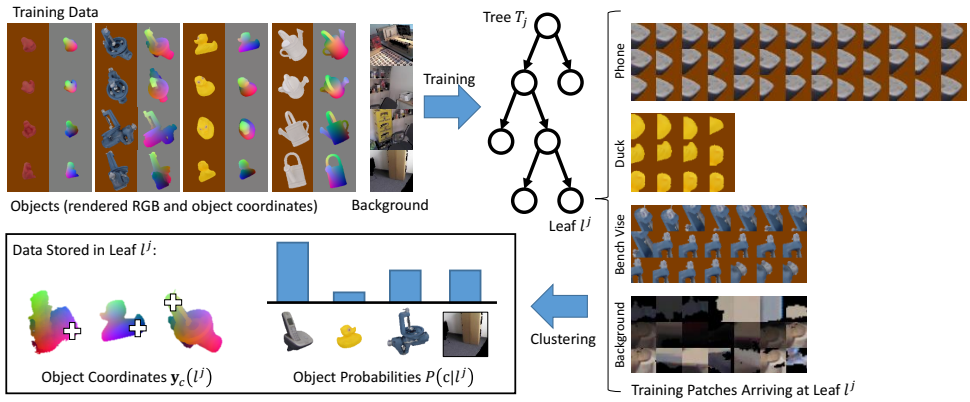

Figure 2.4.: **Training of our Random Forest.** Clockwise: We render RGB-D images and object coordinates ground truth of all objects. We sample image patches from these object images and background images to train the structure of each decision tree in a greedy fashion. After the structure is trained, we collect training patches in tree leafs. Based on these samples, we calculate object probability distributions, shown as a histogram, and object coordinate estimates for each object, shown as crosses on object coordinate renderings.

outcome of all trees for a pixel i is summarized in the vector $\mathbf{l}_i = (l_i^1, \ldots, l_i^j, \ldots, l_i^{|\mathcal{T}|})$. For each pixel i in the image and for each object c, we calculate a weight $P_{c,i}$ by combining the $P(c|l_i^j)$ stored at leafs l_i^j. The weight $P_{c,i}$ can be seen as the approximate probability $P(c|\mathbf{l}_i)$ that a pixel i belongs to object c given it ended up in the combined set of leaf nodes $\mathbf{l}_i = (l_i^1, \ldots, l_i^j, \ldots, l_i^{|\mathcal{T}|})$. We will thus refer to the weight $P_{c,i}$ as the object probability. We calculate the object probability as

$$P_{c,i} = \frac{\prod_{j=1}^{|\mathcal{T}|} P(c|l_i^j)}{\left(\sum_{c' \in \mathcal{C}} \prod_{j=1}^{|\mathcal{T}|} P(c'|l_i^j)\right) + \prod_{j=1}^{|\mathcal{T}|} P(BG|l_i^j)}. \tag{2.4}$$

All object probabilities of an input image yield a soft segmentation for the objects of interest.

Deduction of Eq. 2.4 In the following, we will give a detailed deduction of Eq. 2.4. Our goal is to calculate the approximate probability $P_{c,i} \approx P(c|\mathbf{l}_i)$, that a pixel i belongs to object c given it ended up in the leafs $\mathbf{l}_i = (l_i^1, \ldots, l_i^{|\mathcal{T}|})$ of trees $T^1, \ldots, T^{|\mathcal{T}|}$. Based on Bayes' theorem we can calculate this probability as

$$P(c|\mathbf{l}_i) = \frac{P(c, \mathbf{l}_i)}{P(\mathbf{l}_i)} \tag{2.5}$$

$$= \frac{P(c, \mathbf{l}_i)}{\sum_{c' \in \mathcal{C}} P(c', \mathbf{l}_i) + P(BG, \mathbf{l}_i)}, \tag{2.6}$$

where $P(\mathbf{l}_i)$ is the probability that a pixel ends up in the leafs $\mathbf{l}_i$ regardless the object it belongs to. The expression $P(BG, \mathbf{l}_i)$ denotes the joint probability that the pixel is part of

the background and ends up in leafs $\mathbf{l}_i$. We will first focus on calculating the joint probability $P(c, \mathbf{l}_i)$ that the pixel belongs to object c and ends up in the leafs $\mathbf{l}_i$. It can be calculated as

$$P(c, \mathbf{l}_i) = P(c)P(\mathbf{l}_i|c) \tag{2.7}$$

$$\approx P(c) \prod_{j=1}^{|\mathcal{T}|} P(l_i^j|c), \tag{2.8}$$

where Eq. 2.8 is based on the assumption of conditional independence of the leaf outcomes $\mathbf{l}_i$ given the pixel's object affiliation c. This assumption can be seen as problematic, since the trees were trained to separate pixels not only according to object affiliation but also according to their position in object space, i.e. their object coordinates. Therefore, our calculations are an approximation which work well in practice, however. We can calculate the conditional probability $P(l_i^j|c)$ for a leaf outcome l_i^j given the pixel is part of object c as

$$P(l_i^j|c) = \frac{P(c|l_i^j)P(l_i^j)}{P(c)}, \tag{2.9}$$

where $P(l_i^j)$ is the a priori probability of the leaf outcome l_i^j. We can thus calculate the joint probability $P(c, \mathbf{l}_i)$ for the object affiliation c and leaf outcome $\mathbf{l}_i$ as

$$P(c, \mathbf{l}_i) \approx P(c) \prod_{j=1}^{|\mathcal{T}|} \frac{P(c|l_i^j)P(l_i^j)}{P(c)}. \tag{2.10}$$

In a similar fashion, we can calculate the joint probability $P(c, \mathbf{l}_i)$ for the pixel being part of the background and leaf outcome $\mathbf{l}_i$ as

$$P(BG, \mathbf{l}_i) \approx P(BG) \prod_{j=1}^{|\mathcal{T}|} \frac{P(BG|l_i^j)P(l_i^j)}{P(BG)}, \tag{2.11}$$

where $P(BG)$ is the a priori probability that a pixel belongs to background. By combining Equations 2.6, 2.10 and 2.11, we can calculate the desired probability as

$$P(c|\mathbf{l}_i) \approx \frac{P(c, \mathbf{l}_i)}{P(\mathbf{l}_i)} \tag{2.12}$$

$$= \frac{P(c, \mathbf{l}_i)}{\sum_{c' \in \mathcal{C}} P(c', \mathbf{l}_i) + P(BG, \mathbf{l}_i)} \tag{2.13}$$

$$= \frac{P(c) \prod_{j=1}^{|\mathcal{T}|} \frac{P(c|l_i^j)P(l_i^j)}{P(c)}}{\left(\sum_{c' \in \mathcal{C}} P(c') \prod_{j=1}^{|\mathcal{T}|} \frac{P(c'|l_i^j)P(l_i^j)}{P(c')} \right) + P(BG) \prod_{j=1}^{|\mathcal{T}|} \frac{P(BG|l_i^j)P(l_i^j)}{P(BG)}}. \tag{2.14}$$

If we assume that the a priori probability $P(c)$ that a pixel is part of an object c is the same for each object and identical to the a priori probability $P(BG)$ that a pixel is part of the back-

ground[1], we can simplify:

$$P(c|\mathbf{l}_i) = \frac{\prod_{j=1}^{|\mathcal{T}|} P(c|l_i^j)P(l_i^j)}{\left(\sum_{c'\in\mathcal{C}}\prod_{j=1}^{|\mathcal{T}|} P(c'|l_i^j)P(l_i^j)\right) + \prod_{j=1}^{|\mathcal{T}|} P(BG|l_i^j)P(l_i^j)} \tag{2.15}$$

$$= \frac{\left(\prod_{j=1}^{|\mathcal{T}|} P(l_i^j)\right)\prod_{j=1}^{|\mathcal{T}|} P(c|l_i^j)}{\left(\prod_{j=1}^{|\mathcal{T}|} P(l_i^j)\right)\left(\left(\sum_{c'\in\mathcal{C}}\prod_{j=1}^{|\mathcal{T}|} P(c'|l_i^j)\right) + \prod_{j=1}^{|\mathcal{T}|} P(BG|l_i^j)\right)} \tag{2.16}$$

The factor $\prod_{j=1}^{|\mathcal{T}|} P(l_i^j)$ is present in the numerator and denominator. We can thus finally write:

$$P(c|\mathbf{l}_i) \approx P_{c,i} = \frac{\prod_{j=1}^{|\mathcal{T}|} P(c|l_i^j)}{\left(\sum_{c'\in\mathcal{C}}\prod_{j=1}^{|\mathcal{T}|} P(c'|l_i^j)\right) + \prod_{j=1}^{|\mathcal{T}|} P(BG|l_i^j)}. \tag{2.17}$$

The probabilities $P(c|l_i^j)$ and $P(BG|l_i^j)$ are stored at the leaf l_i^j. In our implementation we add a small constant ($= 10^{-8}$) in the denominator for numerical stability.

2.2.2. SCORING POSE HYPOTHESES

Our goal is to estimate the 6D pose $\mathbf{h}_c$ for an object c. The pose $\mathbf{h}_c$ is defined as the rigid body transformation that maps a point from object space $\mathbf{y}_c$ into camera space $\mathbf{e}$, i.e. $\mathbf{e} = \mathbf{h}_c\mathbf{y}_c$. We formulate pose estimation as an optimization problem w.r.t. to some scoring function s. To calculate the score of a pose hypothesis we compare synthetic images rendered using $\mathbf{h}_c$ with the observed depth values $(d_1, \ldots, d_n)$ and the predictions of the forest $(\mathbf{l}_1, \ldots, \mathbf{l}_n)$. Our scoring function is based on three components:

$$s_c(\mathbf{h}_c) = \lambda^{\text{depth}} s_c^{\text{depth}}(\mathbf{h}_c) + \lambda^{\text{coord}} s_c^{\text{coord}}(\mathbf{h}_c) + \lambda^{\text{seg}} s_c^{\text{seg}}(\mathbf{h}_c). \tag{2.18}$$

Note that the scoring function s_c has a sub-index to denote its dependence on a 3D model of the object c. While the component $s_c^{\text{depth}}(\mathbf{h}_c)$ measures deviations between the observed and rendered depth images, the components $s_c^{\text{coord}}(\mathbf{h}_c)$ and $s_c^{\text{seg}}(\mathbf{h}_c)$ measure consistency of the pose hypothesis and the predictions of the forest, namely the object coordinates and the soft segmentation. Fig. 2.5 visualizes the benefits of each component. The parameters λ^{depth}, λ^{coord} and λ^{seg} reflect the reliability of the different observations. We will now describe the components in detail.

The Depth Component. This component compares observed and rendered depth images. It is defined as

$$s_c^{\text{depth}}(\mathbf{h}_c) = -\frac{\sum_{i\in M_c(\mathbf{h}_c)} f(d_i, d_i^*(\mathbf{h}_c))}{|M_c(\mathbf{h}_c)|}, \tag{2.19}$$

where $M_c(\mathbf{h}_c)$ is the object mask, i.e. the set of pixels belonging to object c. It is derived from the pose $\mathbf{h}_c$ by rendering the object into the image, and excluding pixels with missing depth observations. The term $d_i^*(\mathbf{h}_c)$ is the depth at pixel i produced by rendering the 3D model of object c with pose $\mathbf{h}_c$. The denominator in the definition normalizes the depth component to make it independent of the object's distance to the camera. In order to handle inaccuracies in the 3D model we use a robust error function: $f(d_i, d_i^*(\mathbf{h}_c)) = \min\left(|d_i - d_i^*(\mathbf{h}_c)|, \tau_d\right)/\tau_d$, with cutoff threshold τ_d as free parameter.

[1] Note that a pixel is usually much more likely to belong to the background than to any object. However, experiments with an increased prior probability for background reduced accuracy of our system.

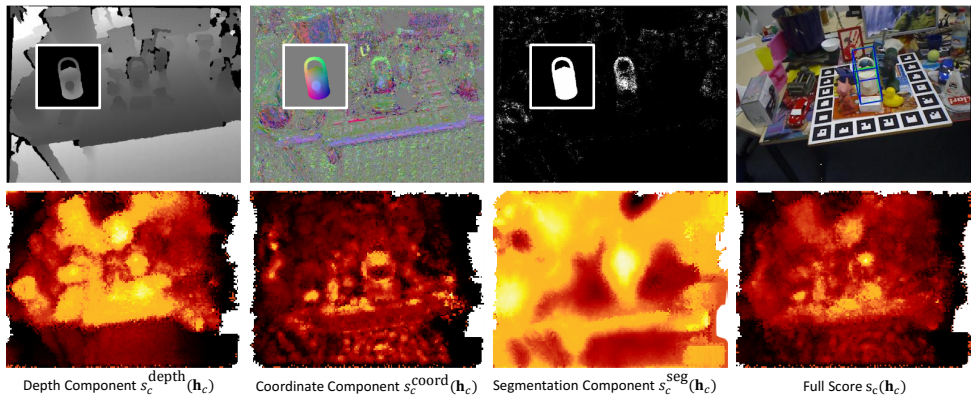

Figure 2.5.: **Components of the Scoring Function.** We visualize the composition of our scoring function for the *Can* object. **Top:** We show the different input channels from which we calculate the individual components of our scoring function. From left to right: The depth channel, object coordinate predictions of one tree, object probability predictions, and (not used for computing scores) the RGB image with the final pose estimate (blue) compared to ground truth (green). As an inlay for each channel, we also show the corresponding object renderings which are checked for consistency by the scoring function. **Bottom:** For visualization, we calculated dense score maps for each score component and the full scoring function. We created a dense sampling of the 6D pose space for the input image, calculated the score for each pose sample, and projected the highest score into the image for each pixel ray. Bright color means high score. While different scoring components display strong local maxima, their combination usually shows the strongest maxima at the correct pose. Note that in our pose optimization we evaluate scores only very sparsely for a few pose hypotheses.

The Coordinate Component. This component measures deviations between the object coordinates predicted by the forest and rendered object coordinates. The component is defined as

$$s_c^{\text{coord}}(\mathbf{h}_c) = -\frac{\sum_{i \in M'_c(\mathbf{h}_c)} \sum_{j=1}^{|\mathcal{T}|} g(\mathbf{y}_c(l_i^j), \mathbf{y}^*_{i,c}(\mathbf{h}_c))}{|M'_c(\mathbf{h}_c)|}. \tag{2.20}$$

where $M'_c(\mathbf{h}_c)$ is the set of pixels belonging to object c excluding pixels with no depth observation and pixels where the object probability is smaller than a threshold, i.e. $P_{c,i} < \tau_p$. The latter is necessary because we find that pixels with small $P_{c,i}$ do not provide reliable object coordinate predictions $\mathbf{y}_c(l_i^j)$. The term $\mathbf{y}^*_{i,c}(\mathbf{h}_c)$ denotes object coordinates rendered using our 3D model of object c with pose $\mathbf{h}_c$. We again use a robust error function $g(\mathbf{y}_c(l_i^j), \mathbf{y}_{i,c}(H_c)) = \min\left(||\mathbf{y}_c(l_i^j) - \mathbf{y}^*_{i,c}(\mathbf{h}_c)||^2, \tau_y\right)/\tau_y$, with cutoff threshold τ_y as free parameter.

The Segmentation Component. This component compares a rendered segmentation with the soft segmentation, i.e. the object probabilities $p(c|l_i^j)$, predicted by the forest. It is defined as

$$s_c^{\text{seg}}(\mathbf{h}_c) = -\frac{\sum_{i \in M_c(\mathbf{h}_c)} \sum_{j=1}^{|\mathcal{T}|} -\log p(c|l_i^j)}{|M_c(\mathbf{h}_c)|}. \tag{2.21}$$

Robustness of the Scoring Function. Since our scoring components are all normalized, stability can be an issue whenever the number of pixels to be considered becomes very small. To address this issue we return $s_c(\mathbf{h}_c) = 0$ whenever $|M'_c(\mathbf{h}_c)| < 100$.

2.2.3. POSE OPTIMIZATION

In order to find the pose which maximizes the score of Eq. 2.18, we use a RANSAC-based algorithm. It samples pose hypotheses based on observed depth values and the coordinate predictions from the forest. Subsequently, these hypotheses are scored and refined. A visualization of the process can be found in Fig. 2.1. We will now describe the procedure in detail.

Sampling of a Pose Hypothesis. We first draw a single pixel i_1 from the image using a weight proportional to the object probability $P_{c,i}$ for each pixel i. We draw two more pixels i_2 and i_3 from a square window around i_1 using the same method. The size of the window is calculated from the diameter of the object δ_c and the observed depth value d_{i_1} of the first pixel, $w = f\delta_c/d_i$ where f is the focal length. Sampling is done efficiently using an integral image of $P_{c,i}$. For each of the three pixels drawn, we can calculate coordinates in camera space $\mathbf{e}_{i_1}, \mathbf{e}_{i_2}$ and $\mathbf{e}_{i_3}$ using observed depth.

Then, we randomly choose tree indices j_1, j_2 and j_3 for the three pixels to read out the three associated object coordinate predictions. Together with the camera coordinates, this yields a set of three 3D-3D correspondences $(\mathbf{e}_{i_1}, \mathbf{y}_c(l_{i_1}^{j_1}))$, $(\mathbf{e}_{i_2}, \mathbf{y}_c(l_{i_2}^{j_2}))$ and $(\mathbf{e}_{i_3}, \mathbf{y}_c(l_{i_3}^{j_3}))$. We use the Kabsch algorithm [Kab76] to calculate the pose hypothesis $\mathbf{h}_c$ that minimizes the squared distance between the three camera coordinate and object coordinate pairs.

The object coordinate predictions of the random forest are very noisy and contain many outliers. Therefore, we perform a geometric check to quickly identify and discard erroneous hypotheses. We map the predicted object coordinate $\mathbf{y}_c(l_{i_1}^{j_1})$ into camera space using $\mathbf{h}_c$

and calculate a transformation error $e_{i_1,j_1}(\mathbf{h}_c) = ||\mathbf{e}_{i_1} - \mathbf{h}_c\mathbf{y}_c(l_{i_1}^{j_1})||$, which is the Euclidean distance to the corresponding camera coordinate. Similarly, we compute errors $e_{i_2,j_2}(\mathbf{h}_c)$ and $e_{i_3,j_3}(\mathbf{h}_c)$ of the remaining pixels. We accept a pose hypothesis $\mathbf{h}_c$ only if none of the three distances is larger than 5% of the object's diameter δ_c. The process is repeated until a fixed number of 210 hypotheses are accepted. We score all accepted hypotheses according to Eq. 2.18.

Refinement. We select the top 25 hypotheses w.r.t. to our scoring function, and refine them. To refine a pose $\mathbf{h}_c$, we iterate over the set of pixels $M_c(\mathbf{h}_c)$ supposedly belonging to the object c as done for score calculation. For every pixel $i \in M_c(\mathbf{h}_c)$, we calculate the error $e_{i,j}(\mathbf{h}_c)$ for all trees j. Let $\tilde{j}$ be the tree with the smallest error $e_{i,\tilde{j}}(\mathbf{h}_c) \leq e_{i,j}(\mathbf{h}_c)|\forall j \in \{1,\ldots,|\mathcal{T}|\}$ for pixel i. Every pixel i where $e_{i,\tilde{j}}(\mathbf{h}_c) < 20\text{mm}$ is considered an inlier. We store the correspondence $(\mathbf{e}_i, \mathbf{y}_c(l_i^{\tilde{j}}))$ for all inlier pixels and use them to re-estimate the pose with the Kabsch algorithm. We repeat the process until the score of the pose according to Eq. 2.18 no longer increases, the number of inlier pixels drops below 3, or a total of 100 iterations is reached.

The Final Estimate. We chose the pose hypothesis with the highest score after refinement as our final estimate. We obtained our estimates in Figs. 2.1 to 2.5 as well as our quantitative results in the experiments section using the exact algorithm described above. Our formulation of the task as an optimization problem, however, allows for the use of any general optimization algorithm to further increase the precision of the estimate.

2.3. EXPERIMENTS

We evaluated our approach on the widely used dataset of Hinterstoisser et al. [HLI+12] and our own datasets. The dataset of Hinterstoisser et al. [HLI+12] provides synthetic training and real test data. Our lighting dataset provides real training and real test data with realistic noise patterns and challenging lighting conditions. To evaluate our method w.r.t. to severe occlusion, we created additional pose annotations for the dataset of Hinterstoisser et al. [HLI+12]. We compare to the template-based method of [HLI+12] on all datasets. We also tested the detection performance and scalability of our method. We present an ablation study w.r.t. to the components of our scoring function, and we comment on running times. We train our decision forest with the following parameters. At each node we sample 500 color features and depth features. In each iteration we choose 1,000 random pixels per training image, collect them in the current leafs and stop splitting if less than 50 pixels arrive. The tree depth is not restricted. We include a complete list of parameter settings for the following experiments in Appendix A.4.1.

2.3.1. OBJECT POSE ESTIMATION

Hinterstoisser et al. [HLI+12] provide colored 3D models of 13 texture-less objects[2] for training, and 1,000+ test images of each object on a cluttered desk together with ground truth poses. The test images cover the upper view hemisphere at different scales and a range of $\pm 45^\circ$ in-plane rotation. More details about the dataset can be found in Appendix A.2.1.

[2] We had to omit 2 objects since proper 3D models were missing.

The goal is to evaluate the accuracy in pose estimation for one object per image. It is known which object is present. We follow exactly the test protocol of [HLI+12] by measuring accuracy as the fraction of test images where the pose of the object was estimated correctly. The tight pose tolerance is defined in Appendix A.3.1. It is based on a threshold on the average distance of transformed 3D points. In [HLI+12], the authors achieve a strong baseline of 96.6% correctly estimated poses, on average. We re-implemented their method and were able to reproduce these numbers. Their pipeline starts with an efficient template matching schema, followed by two outlier removal steps and iterative closest point adjustment. The two outlier removal steps are crucial to achieve the reported results. In essence they comprise of two thresholds on the color and depth difference, respectively, between the current estimate and the test image. Unfortunately, the correct values differ strongly among objects and have to be set by hand for each object[3]. We also compare to [RCT13] who optimize the Hinterstoisser templates in a discriminative fashion to boost performance and speed. They also rely on the same two outlier removal checks but learn the object dependent thresholds discriminatively.

To produce training data for our method we rendered all 13 object models with the same viewpoint sampling as in [HLI+12], but skipped scale variations because of our depth-invariant features. Since our features may reach outside the object segmentation during training we need a background model to compute sensible feature responses. For our color features we use randomly sampled colors from a set of background images. The background set consists of approximately 1500 RGB-D images of cluttered office scenes recorded by ourself. For our depth features, we use an infinite synthetic ground plane as background model. In the test scenes, all objects stand on a table but embedded in dense clutter. Hence, we regard the synthetic plane as an acceptable prior. Additionally, we also show results for a background model of uniform depth noise, and uniform RGB noise. The decision forest is trained for all 13 objects and a background class, jointly. For the background class, we sample RGB-D patches from our office background set. To account for variance in appearance between purely synthetic training images and real test images we add Gaussian noise to the response of color features [SGF+13]. After pose optimizing, we deploy no outlier removal steps or further ICP refinement, in contrast to [HLI+12, RCT13].

Table 2.1.: **Results on the Dataset of Hinterstoisser et al. [HLI+12].** We train with rendered images and different background models (infinite plane and noise). We report the average and median accuracy over all 13 objects. We also report the accuracy for the best (Max.) and worst (Min.) object. We mark the best result per row **bold**. We see that our approach is consistently superior to [HLI+12, RCT13].

	LINEMOD [HLI+12]	DTT-3D [RCT13]	Our (plane)	Our (noise)
Avg.	96.6%	97.2%	**98.3%**	92.6%
Med.	97.1%	97.5%	**98.9%**	92.1%
Max.	99.9%	99.8%	**100.0%**	99.7%
Min.	91.8%	94.2%	**95.8%**	84.4%

Table 2.1 summarizes the results. We score an average of 98.3% with the synthetic plane background model. Hence we improve on both systems of [HLI+12] and [RCT13]. Using uniform noise as background model, we still report accurate results, with 92.6% correctly estimated poses on average. See Fig. 2.6 for qualitative results. We list detailed quantitative results for all objects as well as additional qualitative results in Appendix A.4.1.

[3] We verified this in private communication with the authors. These values are not given in the article.

Table 2.2.: **Synthetic vs. Real Training Data.** We show results on the dataset of Hinterstoisser et al. [HLI+12] using different training data. We achieve high accuracy in all cases.

	Synth. Training Data		Real Training Data	
	Our (plane)	Our (noise)	Our (plane)	Our (noise)
Avg.	**98.3%**	92.6%	98.1%	97.4%
Med.	98.9%	92.1%	**99.6%**	98.8%
Max.	**100.0%**	99.7%	**100.0%**	**100.0%**
Min.	**95.8%**	84.4%	91.1%	89.2%

To verify that our approach is not restricted to synthetic training data, we performed an experiment where we trained with real images for each object. Since the dataset of Hinterstoisser et al. [HLI+12] includes only one sequence per object, we had to split each sequence into training and test. We sampled training images with at least 15° angular distance, to have an approximately regular coverage of the upper hemisphere similar to the synthetic setup. In practice, the maximum distance of training images is $\approx 25^\circ$ making this test slightly harder in terms of generalization to unseen poses. All other images are test images. To remove the background in training images we do an intersection test with the 3D object bounding box. We substitute the background pixels with the two background model variants already discussed above. We do not add noise to the feature responses. In this experiment, we observe high accuracy which is stable even with the simple noise background model (compare right two columns in Table 2.2).

2.3.2. OBJECT DETECTION

Our main experimental setup deals with the task of pose estimation of a known object in an RGB-D image. In many applications, the presence of an object is unknown and has to be established first. This can be done by defining a threshold on the score of the final pose estimate. The system would report a detection only if this score is below the threshold. In the following experiment we evaluate the detection performance of this approach.

We perform this experiment on the dataset of Hinterstoisser et al. [HLI+12] since its images contain dense clutter. In each of the 13 object image sets, we only search for the corresponding object, although other objects might be present. This is because the Hinterstoisser dataset only provides ground truth for one object per set. We run our full pipeline to extract one hypothesis per image. We extract a 2D bounding box based on this hypothesis[4], following the detection evaluation setup in [RCT13]. The bounding boxes of all images of a sequence are ranked according to their hypothesis' score. The ground truth bounding box is extracted using the ground truth pose. As in [RCT13], we consider a detection correct if the intersection over union of detected bounding box and ground truth bounding box is at least 70%. We generated precision-recall curves for each of the 13 Hinterstoisser objects and calculated the average precision[5], AP. See Fig. 2.7 for the precision-recall curves and AP of all objects. The mean AP is 0.88, which proves sensible detection performance on the dataset of Hinterstoisser et al. [HLI+12].

[4] We render the object segmentation mask based on the pose hypothesis and use its bounding box.
[5] We calculate the average precision according to VOC2012 [EVGW+b].

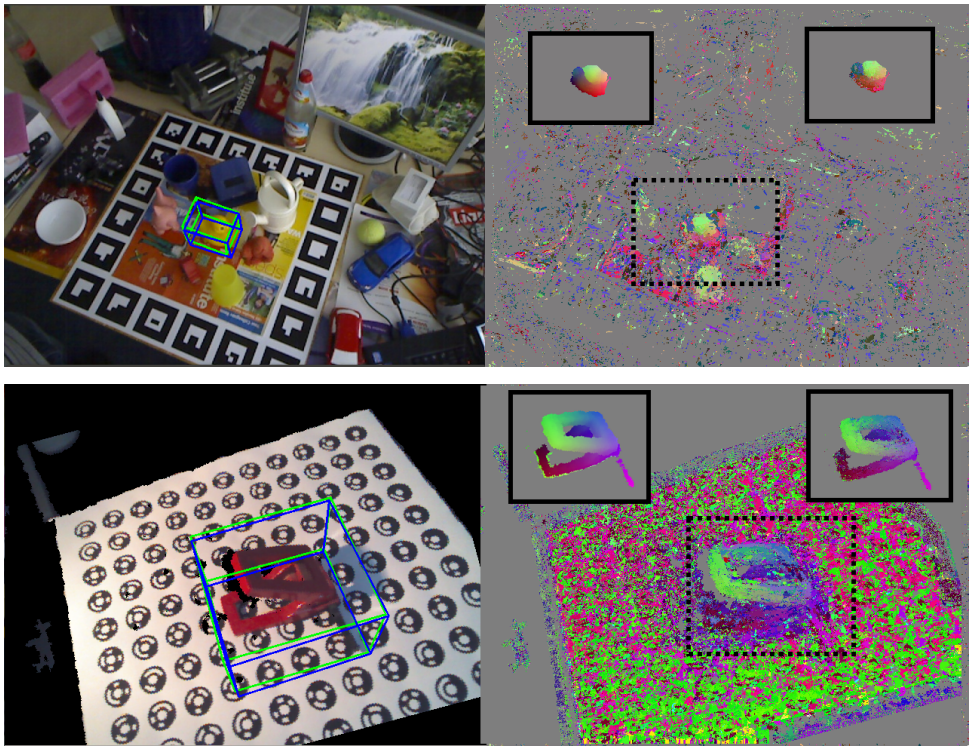

Figure 2.6.: **Examples for Pose Estimation Results.** We visualize our estimates with a blue bounding box and the ground truth pose with a green bounding box. The upper test image shows an object from the dataset of Hinterstoisser et al. [HLI+12], the lower test image shows an object from our lighting dataset. Next to each test image are the predicted object coordinates $\mathbf{y}$ from one tree of the forest. The inlay figures show the ground truth object coordinates (left) and the best object coordinates (right), where "best" is the best prediction of all trees with respect to ground truth (for illustration only).

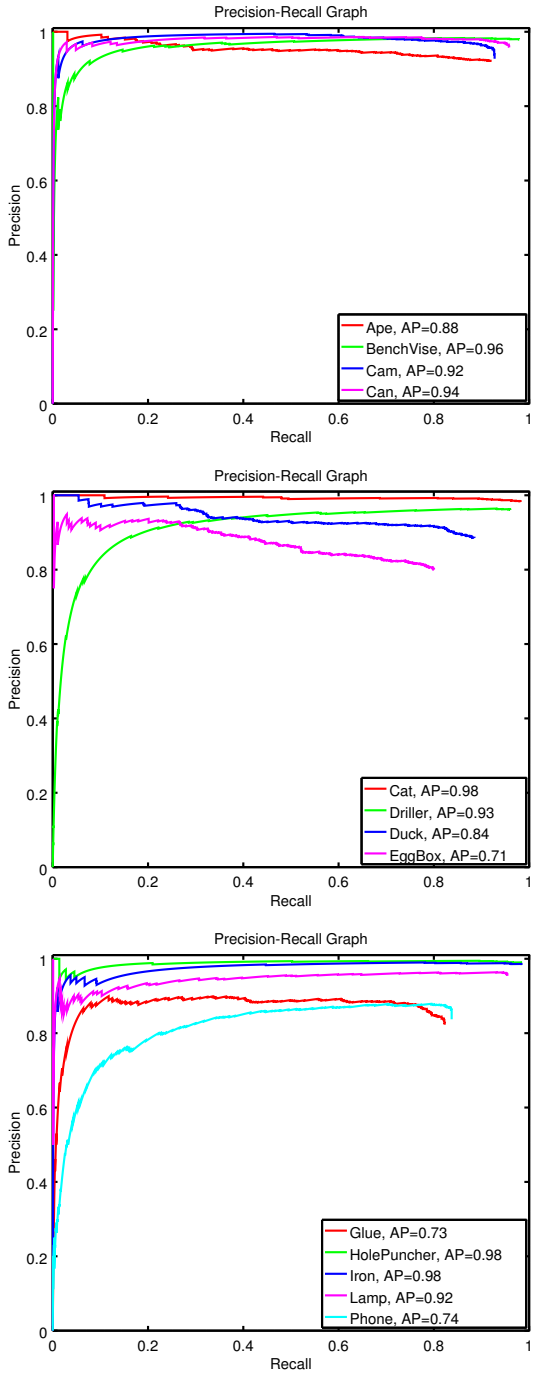

Figure 2.7.: **Detection Performance.** Precision-recall curves of all 13 object of the dataset of Hinterstoisser et al. [HLI+12]. The mean AP of all 13 objects is 0.88

2.3.3. ROBUSTNESS W.R.T. OCCLUSION

While the dataset of Hinterstoisser et al. [HLI+12] is challenging because of the substantial amount of clutter, the object of interest is never occluded, significantly. Other objects in the test images are partially occluded but ground truth pose annotations for these objects are missing. We created these missing annotations for one sequence of the dataset of Hinterstoisser et al. [HLI+12], resulting in approximately 1200 images where all 9 objects present are annotated with accurate 6D poses. More information on this occlusion dataset can be found in Appendix A.2.2.

We applied our full pose estimation approach to the challenging occlusion dataset, resulting in 67.3% average accuracy. Our re-implementation of [HLI+12] achieves only 54.4% average accuracy, see Table 2.3. Consequently, we demonstrate superior robustness to occlusion. In Appendix A.4.1, we show detailed results per object. We attribute our good performance in the presence of occlusion to the fact that our approach relies on local predictions. The random forest is able to predict correct object coordinates for areas not affected by occlusion, and our pose optimization can still find a good solution given a small number of inlier predictions. In contrast, since templates are global descriptions of object appearance, their matching scores quickly deteriorate with growing levels of occlusion. Note that since publication of our approach in [BKM+14], our system has been extended by Krull et al. [KBM+15] and Michel et al. [MKB+17]. Krull et al. [KBM+15] substitute our handcrafted scoring function with a CNN trained to be robust w.r.t. occlusion. Their system scores 70.3% on the occlusion dataset. Michel et al. substitute our RANSAC-based pose optimization schema with a conditional random field (CRF) which identifies sets of object coordinate predictions which are geometrically consistent, similar to our geometric check for pose hypothesis validation. Their system scores 76.7% on the occlusion dataset.

Table 2.3.: **Results on our Occlusion Dataset.** We compare our full scoring function to a score which uses depth only, and to the approach of [HLI+12].

	Full Score	Depth C. Only	LINEMOD [HLI+12]
Avg.	**67.3%**	57.1%	54.4%
Med.	**67.6%**	57.8%	51.2%
Max.	**100.0%**	98.8%	98.7%
Min.	8.5%	2.4%	**23.3%**

2.3.4. ROBUSTNESS W.R.T. LIGHTING CHANGES

We recorded 20 textured and texture-less objects under three different lighting conditions: bright artificial light (*bright*), darker natural light (*dark*), and directional spot light (*spot*). For each light setting we recorded each object on a marker board in a motion that covers its upper view hemisphere. The distance to the object varied during the recording but the in-plane rotation was kept fixed. We added in-plane rotation artificially afterwards in the range of $\pm 45^\circ$. We used KinectFusion [NIH+11, IKH+11] to record the external camera parameters for each frame. This serves as pose ground truth and is used to generate the object coordinates per pixel for training the decision forest. Recordings of the same object but different lighting conditions were registered using the marker board. Images that were used for training were segmented with the 3D object bounding box. An overview over the dataset and details about the recording procedure can be found in Appendix A.2.3. We

sampled training images with at least 15° angular distance. The maximal angular distance of training images is $\approx 25^\circ$. We did not place our objects on a synthetic plane, because they were already recorded on a planar board. Depth features reaching outside the object mask during training will just use the depth in the original training image. For color features we sampled randomly from another set of office backgrounds that do not contain our objects.

To evaluate how well our approach generalizes with respect to varying lighting conditions, we trained our decision forest with the *bright* and *dark* training sets. Again, we added Gaussian noise to the response of the color feature for robustness. In a first run, we tested with images of the *bright* set that were not used for training. Here, the forest did not need to generalize to new lighting conditions but only to unseen views, which it does with excellent accuracy (avg. 95%, see Table 2.4). As before, we measured performance as the percentage of correctly estimated poses of one object per test image which is always present. In a second run, we tested using the complete *spot* set to demonstrate the capability of generalization to a difficult new lighting condition. We report an average rate of correctly estimated poses of 88.2%. See Fig. 2.6 for a qualitative result on our dataset. Complete quantitative results for all of our own objects, as well as additional qualitative results can be found in Appendix A.4.1.

To demonstrate that the template based approach of LINEMOD [HLI$^+$12] does not generalize as well with respect to lighting change we used our re-implementation to extract templates for one object based on the training set described above. Note, that the training set contains each view only with one scale. This can be problematic for LINEMOD if the test data shows scale variation not covered by the training data. Hence, we render each training image from 2 larger and 3 smaller distances in 10cm steps. This gives us 6 different scales for each training image similar to the setup in [HLI$^+$12]. As in [HLI$^+$12], we tuned the outlier removal parameters by hand. However, we found that we had to disable these tests completely to get any detections under new lighting conditions. In a validation run, we extracted templates from the *bright* training set and tested on the *bright* test set. Following the procedure of [HLI$^+$12], we can estimate correct poses in 80.1% of the images. We account the difference to the performance on the dataset of Hinterstoisser et al. [HLI$^+$12] to the fact that the object is textured and that our images contain high levels of noise. If we test with the same templates on the *spot* set, performance drops to 57.1%. Since our tree has seen both *bright* and *dark* in training, we apply the following testing procedure to LINEMOD for a fair comparison. We also extract templates from the *dark* training set and apply it to the *spot* test set, observing 55.3%. For the final score, we consider an image solved by LINEMOD if one of the template sets, *dark* or *bright*, lead to the correct pose. Then we observe an accuracy of 70.2%. Hence, even when selecting the best of multiple hypotheses of LINEMOD, performance drops by 10%. On the same object, we report accuracy of 96.9% on the *bright* test set (a lighting condition our forest has been trained with), and 91.8% on the *spot* test set (not seen in training).

2.3.5. CONTRIBUTION OF SCORING COMPONENTS

We conducted further experiments to reveal the contribution of the individual components of our scoring function for pose hypotheses. We repeated pose estimation experiments on the dataset of Hinterstoisser et al. [HLI$^+$12] but using each scoring component alone. Results are included in Table 2.5, see Appendix A.4.1 for detailed results per object. The coordinate component and the segmentation component alone give clearly inferior results. The depth component alone gives results comparable to the full scoring function. However, we repeated tests on the occlusion dataset discussed above, and observe that, on this

Table 2.4.: **Accuracy on our Lighting Dataset.** We trained our random forest using training images from the *bright* and *dark* lighting condition. We measure accuracy using test images of the *bright* and *spot* lighting condition, separately. We report average and median accuracy for our 20 objects (left). We also compare to LINEMOD [HLI+12] on one object (right). We extract templates from training images of the *bright* and *dark* lighting condition, and measure the respective accuracy. We also calculate accuracy when either template set resulted in the correct pose, denoted *combined*. See the text for details.

	All Objects		Toy (Battle Cat)			
Test Condition	Our, Avg.	Our, Med.	Our	LINEMOD *(dark)*	LINEMOD *(bright)*	LINEMOD *(combined)*
bright	95.6%	97.7%	96.9%	-	80.1%	-
spot	88.2%	93.0%	91.8%	55.3%	57.1%	70.2%

dataset, the depth component alone achieves only 57.1% average accuracy instead of 67.3% with our full scoring function (see third column of Table 2.3). We conclude that the forest predictions are important in handling object occlusion.

Furthermore, we include an additional baseline: Similar to [SGZ+13] we use the percentage of inlier pixels in $M'_c(H_c)$ (see Sec. 2.2.2) instead of our scoring function to rate hypotheses. Inliers are formally defined in Sec.2.2.3 via a threshold on the transformation error of object coordinate predictions. We observe 40.3% average accuracy on the dataset of [HLI+12], which clearly demonstrates that our scoring function formulation is superior.

Table 2.5.: **Evaluating the Scoring Components.** Results on the dataset of Hinterstoisser et al. [HLI+12] with different variants of our scoring function.

	Full Score	Depth C.	Seg. C.	Coord. C.	Inlier Score [SGZ+13]
Avg.	**98.3%**	96.4%	48.9%	77.6%	40.3%
Med.	**98.9%**	97.5%	50.6%	79.8%	40.9%
Max.	**100.0%**	99.8%	82.5%	96.7%	63.8%
Min.	**95.8%**	88.8%	15.1%	49.8%	21.3%

2.3.6. SCALABILITY AND RUN TIMES

Scalability. We show the potential of our method with respect to scalability in two different ways: scalability in the object count, and scalability in the space of poses. The first concerns the number of objects the system is able to identify, while the latter concerns the range of poses it can recognize. We start with a forest that was trained for 5 objects of our lighting dataset, and a set of training images sampled from *dark* and *bright* lighting conditions, with an angular distance of at least $45°$. We add $\pm 45°$ in-plane rotation to each training image. During testing, we consider images of the *spot* set which are at maximum $10°$ apart from the closest training image. This results in the same test difficulty as in the previous experiments. Performance is measured for one object (Stuffed Cat). We modify this setup in two ways. Firstly, we increase the object count to 30 by combining our dataset with real images of the Hinterstoisser dataset [HLI+12]. We sample the Hinterstoisser set to have approximately the same amount of training images for our objects and the additional Hinterstoisser objects.

Secondly, we increase the number of in-plane rotated training images 4-fold to the full $\pm 180°$. The results are shown in Fig. 2.8.

	Default	$6\times$Objects	$4\times$Poses
Forest time	102ms	138ms	113ms
Opt. time	443ms	424ms	517ms
Accuracy	95%	94%	95%

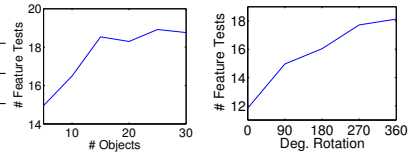

Figure 2.8.: **Scalability. Left:** Running time of our system with increasing object count and pose range. Accuracy stays stable. **Right:** Illustration of the sub-linear growth of the decision forest. We measure the average number of feature tests executed per pixel.

As the number of objects and the range of poses increase, the evaluation time of the tree does increase slightly, but considerably less than $6\times$ resp. $4\times$. The runtime of the pose optimization is effected slightly due to variation in the forest prediction, and the accuracy of the system stays stable. Next to the table in Fig. 2.8, we plot the sub-linear growth in the average number of feature tests per pixel with increasing object number and range of poses. We demonstrate here that with the forest the first essential step of our discriminatively trained method behaves sub-linearly in the number of objects. Our proposed pipeline is still linear in the number of objects because we perform pose optimization for each object individually. In Chapter 3, we will introduce a RANSAC optimization schema, which operates on all objects simultaneously, and scales sub-linearly in the number of objects.

Run Times. The complete run time of our pose estimation approach is the sum of forest prediction time and pose optimization time. Forest predictions are generated once per frame and the results are reused for every object in that frame. Our GPU implementation of the random forests takes 20ms in average per frame on the dataset of Hinterstoisser et al. [HLI$^+$12]. Based on these predictions, pose optimization is done per object. We implemented our scoring function on the GPU and report 398ms avg. per object on the dataset of Hinterstoisser et al. [HLI$^+$12] with the parameter settings suggested above. However, we found that a set of reduced parameters results in a large speed up while maintaining accuracy. We reduced the number of hypotheses from 210 to 42, the number of refinement steps from 100 to 20, and refined only the best 3 hypotheses. This still achieves 96.4% avg. accuracy on the dataset of Hinterstoisser et al. [HLI$^+$12] while reducing the average pose optimization time to 61ms.

The system proposed in this Chapter has been extended by Krull et al. [KMB$^+$14] by embedding it in a Particle Filter framework [GSS93]. Tracking significantly reduces the search space of pose optimization, and they achieve real-time 6D pose estimation with approximately 20 frames per second.

2.4. DISCUSSION

In this chapter, we presented a system which can estimate the 6D pose of a rigid object instance given an RGB-D image. It is based on a random forest which learns to decide for each pixel of the input image whether it belongs to the object and where on the object surface it is located. A subsequent RANSAC-based geometric optimization yields a stable and accurate estimate of the object pose with low runtime. We have shown that the random forest can be trained from synthetic data, that it scales well to many objects, and that it is robust to occlusion and lighting changes.

The main limitation of the system presented in this chapter is its heavy reliance on a depth channel, i.e. it can only be applied to pose estimation from an RGB-D image. Furthermore, pose optimization needs to be performed for each object the forest has been trained with. Objects not present in the image can only be discarded after optimization by thresholding the final pose score. We will extend our method w.r.t. to these aspects in the next chapter.

3. ENHANCED 6D POSE ESTIMATION USING UNCERTAINTY INFORMATION

Contents

3.1. INTRODUCTION

In this chapter, we consider the task of estimating 6D poses of multiple object instances from a single RGB image. The work associated with this chapter was published in [BMK+16].

For sufficiently textured objects, pose estimation is, more or less, considered to be solved using sparse feature pipelines, as discussed in the introduction. The broad availability of consumer depth cameras, has lead to development of RGB-D-based methods for pose estimation of difficult, texture-less object instances. We presented one such system in the previous chapter which is based on a combination of machine learning and geometric processing. In the following, we revisit the scenario of having RGB information only, and show that we can transfer the principles discussed so far to this setting. This is especially interesting for practical applications since many mobile devices are only equipped with a single RGB camera. Furthermore, current consumer depth sensors do not work with direct sunlight and reflective materials. Fig. 3.1 shows a result of our system, with augmented reality. This is a challenging case, and an extension of the method of Chapter 2, developed by Krull et al. [KBM+15], which operates on an RGB-D image, is not able to get a visually pleasing result. We show experimentally that our method can deliver a visually pleasing quality in 74% of test cases. Our approach surpasses competing RGB-based systems. Furthermore, the improvements

presented in this chapter can be applied to the RGB-D setting for increased accuracy. In this chapter, we also demonstrate that our system can be applied to the problem of camera localization where it compares favorably with competitors.

Figure 3.1.: **Pose Estimation Using RGB only. Left:** Result of our method using an RGB image as input. We estimate the pose of the lamp, and use this information to overlay lettering and red markers. The pose of the lamp is estimated sufficiently well for augmented realty. **Right:** Result of a system [KBM+15] that uses an RGB-D input image. The pose is less well suited for augmented reality. Note that the red markers are significantly off on the lamp stand.

As in the previous chapter, we utilize an intermediate representation for objects, namely object coordinates, which is a dense, continuous object-part labeling. We regress object coordinates and object labels jointly for every pixel in the input image, i.e. we encode information about multiple objects within one regressor. In the previous chapter, every tree in the forest predicted an object coordinate point estimate per pixel. In the following, we instead model approximate distributions of object coordinates. We will see that this is crucial for applying our technical contributions, and achieving good results. The idea of modeling object coordinate distributions has been explored in parallel to our own work in [VNS+15], in the context of camera localization from a single RGB-D image. We go beyond their work in various aspects. Firstly, their approach cannot directly be used in an RGB setting, since it relies on the calculation of pixel coordinates in camera space. We solve this problem by efficiently marginalizing object coordinate distributions along the pixel ray. Our approach is suitable for mixtures of anisotropic Gaussians and incorporates perspective effects. Secondly, by exploiting the uncertainty of object labels we are able to process any number of objects, known to the system, with a fixed budget of RANSAC hypotheses. This results in an efficient and scalable system w.r.t. the number of objects. A core part of our pipeline is an efficient regressor which iteratively reduces the uncertainty of object coordinate and class label predictions. Previously, we used a standard random forest for this task. Our extension builds upon the insight that the dense object coordinates contain a substantial amount of "structural" information, i.e. neighboring object coordinate predictions are statistically dependent. This makes it ideal for building the prediction step into an auto-context framework [TB10]. However, directly using an auto-context random forest hampered test performance, due to noisy outputs. We demonstrate that a new, robust regularization of the multi-dimensional, dense labeling gives a major boost in performance.

Contributions. The contributions of this chapter are:

1. We present a generic 6D pose estimation system for both object instances and scenes (textured or texture-less), which only needs a single RGB image as input. Our approach exceeds the state-of-the-art. An RGB-D variant of this system exceeds the state-of-the-art for object pose estimation and is on-par for camera localization.

2. In order to deal with the missing depth information of the sensor, we present an efficient way to marginalize the object coordinate distributions over depth.

3. By exploiting label uncertainty we are able to process multiple object instances jointly with a fixed budget of RANSAC hypotheses.

4. A new, robust (L_1-loss) regularization of the multi-dimensional, continuous output, namely object coordinates and labels, is crucial to get good performance with our auto-context regression forest approach.

The question of how to deal with missing depth information is of general interest to all works that deal with object coordinates, e.g. [KMB+14, SGZ+13, GRKG+14, VNS+15, TSSF12], and would like to use an RGB camera. The idea of applying a fixed-budget RANSAC for multi-object estimation is of interest to all methods where the final optimization depends on variables (explicitly or latent) such as object instance (as in our case), person height [SKS12] or object part [MKB+15]. Our L_1-loss regularization of object coordinates, is of general interest to other auto-context regression frameworks, e.g. auto-context Hough forests [KBC+12].

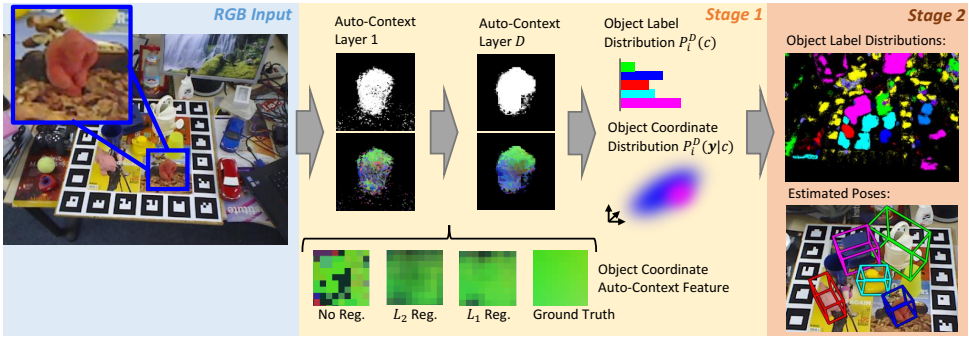

Figure 3.2.: **An RGB Image is Processed by our Pipeline.** Left to right: We visualize two, out of three, stages of our method. In the first stage, the auto-context random forest processes the entire image to predict object labels (shown as probabilities) and object coordinates — only shown for the location of the toy object. The key to make the auto-context forest work well is to apply L_1-loss regularization to the multidimensional data, after each layer. The output of the first stage are pixel-wise distributions of object labels and object coordinates. In the second stage, the distributions over object labels are used to sample hypotheses for all objects at once. Preliminary pose estimates are found with pre-emptive RANSAC. In the third stage, these poses are refined using the object coordinate distributions.

Related Work. We briefly discuss related work relevant for this chapter. Auto-context has been introduced in the work of Tu and Bai [TB10]. The idea is to train stacked classifiers where early classifiers provide input for subsequent classifiers, resulting in a substantial gain in performance. Auto-context has been widely used for segmentation [TB10, SJC08, MSW+11, KKSC13], detection [KBC+12], and human pose estimation [RMH+14, DGLVG13], etc. Montillo et al. [MSW+11] presented an auto-context variant where a random forest has access to its own intermediate predictions for increased efficiency. In [KKSC13], this was extended by smoothing the intermediate predictions. This couples predictors of neighboring pixels, resulting in locally consistent output. We extend this idea further to multi-dimensional continuous data. Auto-context features usually access preliminary probabilities of discrete classes. Kontschieder et al. [KBC+12] showed that auto-context can also be deployed for intermediate multi-dimensional continuous output in a Hough forest for pedestrian detection, but they did not smooth the intermediate output. In this chapter, we show that robust smoothing of the multi-dimensional continuous object coordinates is essential for good performance.

The simultaneous detection of multiple objects (e.g. planes) in data with many outliers is an active field of research [ZKM05, JPF15, MF14]. However, these methods try to fit multiple objects to points in the same coordinate space. While we similarly wish to find multiple objects within the same image, we have a regressor that predicts points in a separate coordinate space for each object. The fitting takes place within these separated spaces. One straight-forward solution for this task is to iterate through all objects and search for the best solution in each object coordinate space, as it has been done in the previous chapter or in [SGZ+13] for camera localization. In contrast to this sequential procedure, we show how RANSAC can efficiently process multiple objects at once according to the evidence in the image.

3.2. METHOD

Our method consists of three individual stages, which are conceptually similar to the previous chapter. In the *first stage*, a random forest predicts object labels and object coordinates jointly for every pixel of the input image. In contrast to Chapter 2, the forest predicts distributions over object labels *and* object coordinates, instead of point estimates (Sec. 3.2.1). In order to reduce uncertainty of the predictions as much as possible, we extend the random forest to an auto-context random forest (Sec. 3.2.2) with L_1-loss regularization. In the *second stage*, we estimate the poses of multiple objects from the predicted 2D-3D correspondences using pre-emptive RANSAC guided by the uncertainty in object labels (Sec. 3.2.3). Finally, in the *third stage*, the poses are refined by exploiting the uncertainty of object coordinate predictions (Sec. 3.2.4). Fig. 3.2 shows an overview of our pipeline.

3.2.1. REGRESSION OF OBJECT COORDINATE DISTRIBUTIONS

For each object $c \in \mathcal{C}$, present in the image, we aim at estimating the 6D pose $\mathbf{h}_c$, i.e. a rigid-body transformation. In our convention, $\mathbf{h}_c$ maps a 3D coordinate in object space $\mathbf{y}$ to a 3D coordinate in camera space $\mathbf{e}$. The set $\mathcal{C}$ contains all objects known to the system, including background. We briefly describe our random forest $\mathcal{T}$ of classification-regression trees T, and focus on differences to the forest used in Chapter 2. The random forest predicts for each pixel i in the image the distribution over object labels, i.e. $P_i(c)$. Furthermore, given an object c and pixel i, we also want to predict a distribution of coordinates in object space

$\mathbf{y} \in \mathcal{Y} \subseteq \mathbb{R}^3$, namely $P_i(\mathbf{y}|c)$. We train both distributions jointly within one classification-regression forest, as in the last chapter. Since our input is RGB only, we do not use features that rely on depth nor depth normalization of feature scale. Instead, we use standard pixel value differences in RGB.

$$f_{\text{rgb}}(\boldsymbol{\zeta}, \mathbf{p}_i) = I\left(\mathbf{p}_i + \boldsymbol{\omega}_1, \gamma_1\right) - I\left(\mathbf{p}_i + \boldsymbol{\omega}_2, \gamma_2\right) \ , \tag{3.1}$$

where $I(\mathbf{p}_i, \gamma) = \mathbf{x}_i^{\text{rgb}}[\gamma]$ returns the R, G, or B channel of a pixel according to γ, and $\boldsymbol{\omega}$ indicates a 2D offset. As before, $\boldsymbol{\zeta}$ denotes split node parameters, i.e. $\boldsymbol{\zeta} \subseteq \{\boldsymbol{\omega}_1, \boldsymbol{\omega}_2, \gamma_1, \gamma_2, z, \tau_f\}$ including feature type z and split threshold τ_f. So far, we introduced only one feature type, i.e. $z \in \{\text{rgb}\}$, but in Sec. 3.2.2 we will define two more feature types.

Because our features are not depth normalized, we train the forest with different scales of training images. We approximate re-scaling training images by scaling the feature offsets instead. The scale factor is chosen randomly per training sample. The scale range depends on the minimal and maximal object scale expected at test time. We use a similar approach when adding in-plane rotation to training images. We do not actually rotate training images but we rotate the feature offset vectors instead. This way, the forest can learn to be rotation invariant to some extent.

To train the forest, we draw training samples uniformly from within the object segmentation and up to a certain distance outside the segmentation. This distance is set to 50% of the maximum feature size. Samples outside the segmentation with sufficient overlap with the object contain valuable information about the object boundary. This has proven important in the application of auto-context discussed later in this section. Furthermore, we draw samples of a background class from a selection of random interior background images. At each tree depth level, we sample a pool of feature parameters $\boldsymbol{\zeta}$, uniformly. Feature thresholds τ_f are also chosen randomly by calculating the feature response at a random training pixel. Should a feature access a pixel outside the object segmentation during training we return uniform color noise.

At each split node, we select the feature which has the highest information gain (see Eq. 2.1) of the joint distribution $P(\hat{\mathbf{y}}, c)$, where $\hat{\mathbf{y}}$ are quantized object coordinates, serving as proxy classes. In the previous chapter, we used a regular grid over the object bounding box for quantization. For some objects, like entire rooms in a camera localization task, this results in a very unbalanced distribution over proxy classes. Therefore, we quantize here by nearest neighbor assignment to random cluster centers, which are randomly selected object coordinates of the training data, see Fig. 3.3 for a visualization.

Each leaf l stores object probability distributions $P_T(c|l)$, and, instead of object coordinate point estimates, object coordinate distributions $P_T(\mathbf{y}|c, l)$. Distributions of object coordinates are stored as Gaussian-Mixture models (GMMs),

$$P_T(\mathbf{y}|c, l) = \sum_{(m, \boldsymbol{\mu}, \boldsymbol{\Sigma}) \in \mathcal{M}} m\mathcal{N}(\mathbf{y}; \boldsymbol{\mu}, \boldsymbol{\Sigma}), \tag{3.2}$$

where $\mathcal{M}$ is the set of mixture components, found via mean-shift. We calculate support weight m, mean $\boldsymbol{\mu}$ and covariance Σ for each mode. We dismiss modes with weights $m < 50\%$ of the highest weight in $\mathcal{M}$. We also dismiss modes with less than 10 samples supporting it. We calculate full covariance matrices for each mode, setting the mean of the distribution to the mode coordinate.

During test time, we merge all object probability predictions $P_T(c|l_i)$ of individual trees at pixel i, as before, according to Eq. 2.4 to yield $P_i(c)$, which results in a high contrast soft segmentation. We average object coordinate distributions $P_T(\mathbf{y}|c, l_i)$ by combining all mixture components within one large GMM to yield $P_i(\mathbf{y}|c)$.

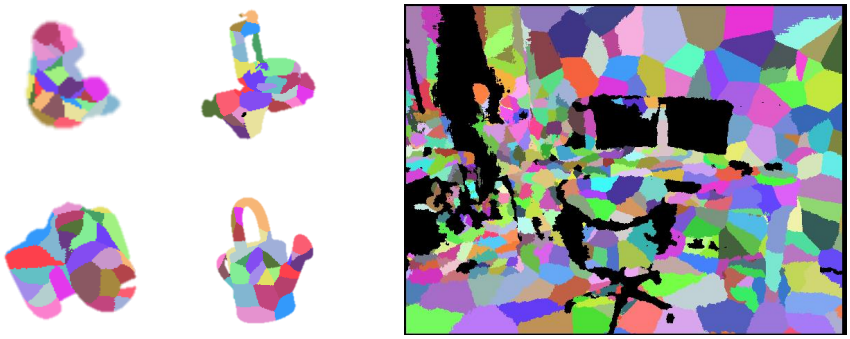

Figure 3.3.: **Proxy Classes.** We show proxy classes used for training the structure of our random forest, for objects (left) and an entire room (right). Class IDs are encoded by color. We create proxy classes by selecting random training pixels as cluster centers, and assigned the remaining training pixels to their nearest cluster in object coordinate space.

3.2.2. OBJECT COORDINATE AUTO-CONTEXT

We now describe the extension of the random forest to an auto-context random forest, and in particular the efficient use of regularized object coordinates as a feature. Instead of one forest, we train a stack of forests $\mathcal{T}^d$, where $d \in \{0, \ldots, D\}$ denotes the stack level. The first level, $\mathcal{T}^0$, is trained exactly as described above. All subsequent forests $\mathcal{T}^{d+1}$ have access to the output of the previous forest $\mathcal{T}^d$, namely object probabilities $P_i^d(c)$ and object coordinate predictions $P_i^d(\mathbf{y}|c)$ of pixel i. Inspired by the "Geodesic Forests" approach [KKSC13], we enforce the coupling of outputs of neighboring pixels by smoothing the predictions before passing them to the next forest. In [KKSC13], a geodesic filter for object probabilities was used because in their application a gradient in the input signal was often a strong indication for an object boundary. We do not observe this correlation in our application scenario, since strong gradients can very well appear within the same object. Instead, we deploy a median filter in a local neighborhood of each pixel. Thus, we define median-smoothed object probabilities $P_i^d(c)$ as

$$g_C^d(c, \mathbf{p}_i) = \underset{\tilde{P}_i \in \mathbb{R}}{\operatorname{argmin}} \sum_{j \in \mathcal{N}_i} |P_j^d(c) - \tilde{P}_i|, \tag{3.3}$$

where $\mathbf{p}_i$ is the position of pixel i and $\mathcal{N}_i$ is a small neighborhood around pixel i. The definition of a feature for this output, to be used in forest $\mathcal{T}^{d+1}$, is straight forward,

$$f_C(\boldsymbol{\zeta}, \mathbf{p}_i) = g_C^d(c, \mathbf{p}_i + \boldsymbol{\delta}). \tag{3.4}$$

The feature parameters $\boldsymbol{\zeta}$ consist of pixel offset $\boldsymbol{\delta}$ and object index c.

We define the smoothing of the object coordinate prediction $P_i^d(\mathbf{y}|c)$ in a similar fashion. The median filter is robust to outliers since it optimizes the L_1-loss. This property is crucial for the object coordinate prediction as well, since outliers are very likely to occur (see Fig. 3.2). If the local smoothing is not robust, outliers will have a strong influence on the result. Unfortunately, the median filter is not directly applicable to data with dimensionality larger than one. However, the optimum under L_1-loss can be calculated in any Euclidean space

resulting in the *geometric median*. Thus, similar to Eq. 3.3, we define our regularized object coordinate output as

$$\mathbf{g}^d_{\mathcal{Y}}(c, \mathbf{p}_i) = \underset{\mathbf{y} \in \mathcal{Y}}{\operatorname{argmin}} \sum_{j \in \mathcal{N}_i} \sum_{T \in \mathcal{T}^d} ||\boldsymbol{\mu}_{j,T} - \mathbf{y}||_2, \tag{3.5}$$

where $\boldsymbol{\mu}_{j,T}$ is the mean with the highest mixture weight of distribution $P^d_{j,T}(\mathbf{y}|c)$ for tree T at pixel j. Hence, $\mathbf{g}^d_{\mathcal{Y}}(c, \mathbf{p}_i)$ is a spatial smoothing, but also a combination of the predictions of trees $T \in \mathcal{T}^d$. Note that the geometric median minimizes the sum of distances as opposed to the sum of squared distances, hence we optimize a L_1-loss. To calculate $\mathbf{g}^d_{\mathcal{Y}}(c, \mathbf{p}_i)$ we use the iterative algorithm of [VZ00]. We define the following feature on the smooth object coordinate output:

$$f_{\mathcal{Y}}(\boldsymbol{\zeta}, \mathbf{p}_i) = [\mathbf{g}^d_{\mathcal{Y}}(c, \mathbf{p}_i + \boldsymbol{\delta})]_j. \tag{3.6}$$

Feature parameters $\boldsymbol{\zeta}$ consist of offset $\boldsymbol{\delta}$, object index c, and dimension index $j \in \{x, y, z\}$. Function $[\mathbf{y}]_j$ returns the entry of $\mathbf{y}$ in dimension j.

The training of forests $\mathcal{T}^d, d \in \{1, \ldots, D\}$ adheres to the same procedure as described in Sec. 3.2.1, but the set of feature types is increased by $f_{\mathcal{C}}$ and $f_{\mathcal{Y}}$. Starting with the second layer of the stack, we calculate the prediction of the previous forest for each training image. Since our training images are segmented and contain no background, we paste each training image into a random image of our background image set at a random position. The resulting montage is most likely physically implausible, but we found this simple strategy sufficient. We calculate the forest prediction on the montage. The resulting auto-context feature channels are stored sub-sampled. This reduces the memory footprint, and increases the effective range of smoothing operations between auto-context layers.

Before training the next layer, we also calculate object probabilities of the previous layer on the background image set. The background images represent our set of negative samples, hence object probabilities should be low. Background regions with high object probability represent hard negatives, i.e. samples where the prediction of the last layer was wrong. When training the new layer, we draw 50% of the background training samples according to object probability. Thus, sampling has a bias towards hard negatives. We found this giving a slight but no substantial gain in segmentation performance of the forest.

3.2.3. MULTI-OBJECT RANSAC

In the second stage of our pipeline, we efficiently find a preliminary pose for all objects present in the image. These poses are refined in stage three (Sec. 3.2.4). We first describe a standard procedure for pose estimation of a single object c', which is a combination of the pre-emptive RANSAC of [SGZ+13] and the hypotheses sampling schema of the previous chapter. One large difference is the use of a scoring function which does not depend on a depth channel but instead relies only on the discriminative predictions of the auto-context forest. Then, we formulate a new approach for handling multiple objects at once with a fixed budget of RANSAC hypotheses.

Single Object RANSAC. The auto-context forest predicts for each pixel i the object label distribution $P^D_i(c')$ (where D is the last auto-context level), and the 2D-3D correspondences $(\mathbf{p}_i, P^D_i(\mathbf{y}|c'))$, i.e. the pixel position $\mathbf{p}_i$ and the uncertain object coordinate $\mathbf{y}$. We start by approximating the distributions $P^D_i(\mathbf{y}|c')$ by their main modes $\{\boldsymbol{\mu}_{i,T}|T \in \mathcal{T}^D\}$. As before, $\boldsymbol{\mu}_{i,T}$

denotes the mean with highest mixture weight of $P^D_{i,T}(\mathbf{y}|c')$ of tree T. Thus, the correspondences simplify to $(\mathbf{p}_i, \boldsymbol{\mu}_{i,T})$. We now define the re-projection error as $||\mathbf{p}_i - C\mathbf{h}_{c'}\boldsymbol{\mu}_{i,T}||_2$ where C is the camera matrix, and $\mathbf{h}_{c'}$ is the pose we are searching. We assume a 4×4 matrix parametrization of hypothesis $\mathbf{h}_{c'}$, and normalization of the homogeneous residual vector before calculating the L_2-norm. A correspondence is an inlier when the re-projection error is below τ_{in}. At this stage, we aim at finding the pose which maximizes the inlier count:

$$\tilde{\mathbf{h}}_{c'} = \underset{\mathbf{h}_{c'}}{\operatorname{argmax}} \sum_{i \in \mathcal{W}(\mathbf{h}_{c'})} \sum_{T \in \mathcal{T}^D} \mathbb{1}[||\mathbf{p}_i - C\mathbf{h}_{c'}\boldsymbol{\mu}_{i,T}||_2 < \tau_{in}], \tag{3.7}$$

where function $\mathbb{1}[\cdot]$ is 1 if the enclosed statement is true, otherwise 0. We restrict the calculation of inliers to window $\mathcal{W}(\mathbf{h}_{c'})$, which is the projection of the 3D bounding box of object c'.

The objective function is maximized using locally optimized pre-emptive RANSAC [SGZ+13]: Firstly, we draw a set of $n_{\mathbf{h}}$ pose hypotheses. A hypothesis $\mathbf{h}_{c'}$ is drawn by choosing four correspondences $(i, \boldsymbol{\mu}_{i,T})$ and solving the perspective-n-point problem (PnP) with the algorithm of Gao et al. [GHTC03]. Note that this substitutes the Kabsch algorithm of the previous chapter, which needed only three correspondences but relied on a depth channel. The tree T from which to use $\boldsymbol{\mu}_{i,T}$ is chosen randomly.

We draw the four initial pixel locations according to probability $P^D_i(c')$, which is an unnormalized distribution over all locations i in the image. Furthermore, pixels 2-4 are drawn within a box centered on the first pixel, which depends on the object size. Since we cannot calculate the actual projected size of the object because of missing depth values we instead use the following heuristic: Firstly, we read out the object coordinate prediction $\boldsymbol{\mu}_{i_1,T}$ associated with the first pixel chosen. We calculate the maximum distance of $\boldsymbol{\mu}_{i_1,T}$ to the 3D object bounding box. We project this distance into the image, assuming a worst case depth of 30cm, which we assume as minimal object distance to the camera. The result is a worst case search window which we relax by shrinking its size to 30%. We reject configurations of 4 pixels which do not pass the following tests: The minimum distance of all pixels in image space should be at least 10px. The minimum distance of all pixels in object space should be at least 10mm. None 3 pixels should be co-linear in object space. Hence, we enforce a minimum distance of 10mm between the line formed by two pixels and the third pixel. Finally, the re-projection error of the 4 correspondences should be below the inlier threshold. If after 1M iterations no valid pixel set has been found, we abort. We discard a hypothesis if the resulting 2D bounding box occupies less than 400 pixels.

For all valid hypotheses, we draw a batch of n_B pixels within their respective windows $\mathcal{W}(\mathbf{h}_{c'})$, and calculate the number of inliers. The hypotheses are ranked according to their inlier count and the lower half is dismissed. The remaining hypotheses are coarsely refined by re-solving PnP on the increased inlier set. Here, we use the PnP algorithm of Lepetit et al. [LMNF09] which is fast even if the set of input correspondences is large. However, we limit the maximal number of inlier correspondences for PnP to 1000. The process of determining inliers, re-solving PnP and discarding half of the hypotheses is repeated with additional batches of pixels drawn in each iteration. Finally, one hypothesis remains which we use as preliminary pose estimate for object c', denoted by $\mathbf{h}'_{c'}$ with inlier set $\mathcal{I}(\mathbf{h}'_{c'})$.

The main advantages of locally refined, pre-emptive RANSAC, as deployed here, compared to the pose optimization described in Chapter 2 are the following. Instead of choosing the n-best, unrefined hypothesis once and refining them to the end, a larger number of hypothesis is partially refined and pruned incrementally. The intuition is that only after some refinement iterations it becomes evident which hypotheses are likely to have high accuracy.

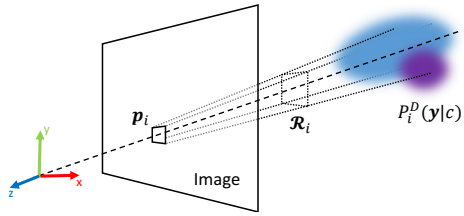

Figure 3.4.: **Exploiting Object Coordinate Uncertainty.** During pose refinement we need to marginalize the 3D object coordinate distribution $P_i^D(\mathbf{y}|c)$ over depth. We show that an approximation can be computed in closed-form for the projection pyramid at position $\mathbf{p}_i$. This is done by integrating along the ray (dashed line) while taking perspective effects into account.

In this context, the advantage of the inlier-based scoring function over the rendering-based score of Chapter 2 is that it can easily be evaluated partially on a limited number of pixels. Only for a few, surviving hypotheses will we calculate the inlier score on a larger number of pixels.

Multi-Object RANSAC. Often, it is unknown which objects are present in the given image. The procedure above could be repeated for all objects $\mathcal{C}$ and the final hypotheses could be filtered based on inlier count, see e.g. the detection experiment in Sec. 2.3.2. However, this scales poorly when the number of potential objects $|\mathcal{C}|$ becomes large. We present a new method for processing all objects at once, within the pre-emptive RANSAC framework. Instead of drawing $n_{\mathbf{h}}$ hypotheses for each object independently, we draw one *shared* set of hypotheses. During the sampling of a hypothesis we decide on the fly which object it belongs to. This decision is based on the object label prediction of the first pixel sampled.

In detail, each hypothesis is created as follows: Assuming background as $c = 0$ we draw the first pixel location i according to $\sum_{c=1}^{|\mathcal{C}|} P_i^D(c)$, which is an unnormalized distribution over i. That is, the first pixel is drawn according to the probability of belonging to *any* object. Only then do we determine the object's identity by drawing the object index c' of the local distribution at pixel i, i.e. $P_i^D(c)$. The remaining three pixel positions are drawn according to the unnormalized distribution $P_i^D(c')$, subject to the bounding box constraint mentioned earlier. Finally, we perform pre-emptive RANSAC separately for each object which was elected by at least one hypothesis.

Note that no hypothesis will be drawn for objects without sufficient evidence in the image. Furthermore, each hypothesis has to pass the initial validity check, which requires some consistency of the object coordinate predictions drawn. Hence, the final distribution of objects c in the hypothesis pool will in general not follow the (unnormalized) distribution $\sum_i P_i^D(c)$ of object probabilities over the image because hypotheses of certain objects might be more likely to fail the hypothesis validity check.

3.2.4. POSE REFINEMENT

The preliminary poses $\mathbf{h}'_c$ of RANSAC minimize the (truncated) re-projection error of a set of inlier correspondences $(i, \boldsymbol{\mu}_{i,T})$, see Eq. (3.7). Note that in the objective function each of

the correspondences is treated with equal weight. However, we have access to full distributions $P_i^D(\mathbf{y}|c)$, which model uncertainty information in the object coordinate prediction. We aim to exploit this information for pose refinement, which is the third stage in our pipeline. To achieve this, we now introduce a novel procedure for efficiently computing the (approximated) marginalized object coordinate distribution in the image.

The basic idea for pose refinement is that we want to maximize the pose probability under the distributions $P_i^D(\mathbf{y}|c)$. In [VNS+15], this idea was explored by scoring hypotheses based on the log-likelihood of inlier correspondences:

$$\tilde{\mathbf{h}}_c = \underset{\mathbf{h}_c}{\operatorname{argmax}} \sum_i \log P_i(\mathbf{h}_c^{-1}\mathbf{e}_i|c), \tag{3.8}$$

where $\mathbf{e}_i$ is the camera coordinate of pixel i. Note, the transformation $\mathbf{h}_c^{-1}\mathbf{e}_i$ yields an object coordinate. However, unlike [VNS+15], we do not have access to depth information, and thus cannot recover $\mathbf{e}_i$. We only know that the true $\mathbf{e}_i$ must lie within the projection volume of pixel i (see Fig. 3.4). We denote this volume by $\mathcal{R}_i$. We can now substitute the likelihood of $\mathbf{e}_i$ in Eq. (3.8) with the probability of $\mathcal{R}_i$:

$$\begin{aligned} \tilde{\mathbf{h}}_c &= \underset{\mathbf{h}_c}{\operatorname{argmax}} \sum_{i \in \mathcal{I}(\mathbf{h}'_c)} \log P^D_{\mathcal{R}_i}, \text{where} \\ P^D_{\mathcal{R}_i} &= \iiint_{\mathcal{R}_i} P_i^D(\mathbf{h}_c^{-1}[x,y,z]^T|c)\, dx\, dy\, dz\ . \end{aligned} \tag{3.9}$$

For robustness, we calculate the likelihood only over the inlier set $\mathcal{I}(\mathbf{h}'_c)$ of the current, preliminary pose $\mathbf{h}'_c$, found according to Sec. 3.2.3.

We now explain how to approximate $P^D_{\mathcal{R}_i}$ efficiently. Instead of integrating over all dimensions, we integrate along the ray cast from the camera origin to the pixel center (see Fig. 3.4). Since the shape of the volume is a pyramid, we add a quadratic factor during integration since the diameter of the pyramid grows with distance z. Because we would like to integrate in camera coordinate space, we first transform the Gaussian mixture components of $P_i^D(\mathbf{y}|c)$, which is defined in terms of object coordinate space, to camera coordinates using $\mathbf{h}_c = [\boldsymbol{\theta}_c|\mathbf{t}_c]$. For notational convenience we assume a matrix parametrization of pose $\mathbf{h}_c$ and rotation $\boldsymbol{\theta}_c$, here. Furthermore, we apply rotation $\boldsymbol{\theta}_{xy}$ which maps the pixel with position $\mathbf{p}_i$ to $(0,0)^T$. Thus, we transform the mean of each mixture component to $\boldsymbol{\mu}_\mathbf{e} = \boldsymbol{\theta}_{xy}(\boldsymbol{\theta}_c\boldsymbol{\mu} + \mathbf{t}_c)$ and the covariance matrices to $\boldsymbol{\Sigma}_\mathbf{e} = \boldsymbol{\theta}_{xy}(\boldsymbol{\theta}_c\boldsymbol{\Sigma}\boldsymbol{\theta}_c^T)\boldsymbol{\theta}_{xy}^T$. After these transformations, the pixel ray aligns with the z-axis, keeping x and y constant in the integral. This allows us to approximate

$$P^D_{\mathcal{R}_i} \approx \sum_{(m,\boldsymbol{\mu},\boldsymbol{\Sigma}) \in \mathcal{M}_i^D} m \int_{-\infty}^{\infty} z^2 \mathcal{N}([0,0,z])^T; \boldsymbol{\mu}_\mathbf{e}, \boldsymbol{\Sigma}_\mathbf{e})\, dz \tag{3.10}$$

where a closed form solution exists for the integral. Note that the factor z^2 is crucial since volume $\mathcal{R}_i$ is a pyramid with the camera origin as its tip. By plugging this approximation into Eq. (3.9) and optimizing, we are able to refine preliminary poses $\mathbf{h}'_c$, to yield our final pose estimates $\tilde{\mathbf{h}}_c$. We run 100 iterations of [NM65] (gradient free) in the implementation of NLopt [Joh]. When calculating the log-likelihood of each pixel, we ignore the contribution of mixture components whose covariance matrix has a determinant of less than 1000, where object coordinates are measured in mm. We threshold the log-likelihood of each pixel between -100 and 100 for robustness.

3.3. EXPERIMENTS

We evaluate our work on three publicly available datasets. First we demonstrate pose estimation performance for a single object given an RGB image (Sec. 3.3.1). Here, we also report results for RGB-D input images. Next, we evaluate our system with respect to detection of multiple objects (Sec. 3.3.2). Finally, we consider the scenario of camera localization (Sec. 3.3.3).

3.3.1. SINGLE OBJECT POSE ESTIMATION

We conduct the following experiments on the dataset of Hinterstoisser et al. [HLI$^+$12], i.e. we estimate poses of texture-less objects in a cluttered scene (see Fig. 3.2 for an example, and Appendix A.2.1 for details). Each image of the dataset is annotated with the ground truth 6D pose of one object and the ID of this object is assumed to be known. Although colored 3D models of the objects are available for the generation of training images, learning from synthetic images is beyond the scope of the approach in this chapter. Instead, we split the real image sequences of the dataset in training and test as explained in the end of Sec. 2.3.1. We select training images such that the associated object poses have a minimum angular distance of 15°, which results in using $\approx 15\%$ of all images for training. The remaining images serve as test set. We segment training images so that the scene context cannot be learned. During training, we sample patches with random scales according to an object distance between 65cm and 115cm, the range given for training in [HLI$^+$12]. Note that this dataset is comprised of RGB-D images. For pose estimation from RGB, we simply omit the depth channel of each test image.

Parameters. We train 3 trees (max. depth 64) per auto-context layer and 3 layers in total. The inlier threshold is set to $\tau_{in} = 3$px and we sample $n_{\mathbf{h}} = 256$ hypotheses during pose optimization. We include a complete list of parameters in Appendix A.4.2.

Metrics. We measure the percentage of images where the object pose was estimated correctly. Different measures have been proposed in the past, see Appendix A.3. Hinterstoisser et al. [HLI$^+$12] define a threshold on the average distance of transformed 3D points. The exact tolerance to translational and rotational error depends on object size and shape. Shotton et al. [SGZ$^+$13] define this tolerance explicitly and accept a pose if the error is below $5cm$ and 5°. While appropriate for some applications, these two measures are not very well suited when applying visual effects to the 2D image, e.g. augmented reality. For example, pose accuracy in z direction is far less important for the visual impression than precision in x and y. Therefore, we additionally propose the following measure, see Appendix A.3.2 for a formal definition. We project the object model into the image using the ground truth pose and the estimated pose. We accept the estimated pose, if the average re-projection error of all model vertices is below 5px. We denote this measure as *2D Projection*. See Fig. 3.1 for a comparison of metrics. With the measure of Hinterstoisser et al. both results are correct, but only the left result is correct with the 2D projection measure. Finally, to evaluate 2D detection performance, we calculate the 2D bounding box overlap and accept it if the intersection over union (IoU) $> 50\%$. We denote this *2D Bounding Box*.

RGB SETTING

Baselines. The template-based approaches LINEMOD (for RGB-D images) and LINE2D (for RGB images) have been introduced in [HCI$^+$11] for object detection. Unfortunately, implementations of LINEMOD and LINE2D are unavailable. We used the open source code in OpenCV which was optimized for synthetic images. To make it work well for real images we activated sampling of strong gradients in the object interior. We verified the detection performance of our LINE2D implementation on the dataset of [RCT13] and achieve comparable results as in [RCT13]. Testing on the original data of [HCI$^+$11] is not possible because it is not publicly available. LINEMOD, but not LINE2D, was extended to perform pose estimation from RGB-D images in [HLI$^+$12]. We created a variant of [HLI$^+$12] based on LINE2D for pose estimation from RGB images. Of the various post-processing steps of [HLI$^+$12] we apply only the color check because the other steps are based on depth.

We were not able to compare to the method of [RCT13]. The authors were not able to provide us source code or binaries, or to run the experiment for us. This method extends [HCI$^+$11, HLI$^+$12] by discriminative learning of templates. Their templates achieve pose estimation accuracy comparable to LINEMOD, while being considerably faster.

Table 3.1.: **Pose Estimation from RGB Images.** Pose estimation results on the dataset of [HLI$^+$12] for a single object where the object ID is known in advance. AC means auto-context.

	Our L_1 reg.	Our L_2 reg.	Our w/o reg.	Our w/o AC	LINE2D [HCI$^+$11]
2D Projection	**73.7%**	68.6%	38.0%	59.3%	20.9%
2D Bounding Box	**97.5%**	97.1%	90.3%	96.2%	86.5%
6D Pose (Metric of [HLI$^+$12])	**50.2%**	46.0%	19.6%	30.1%	24.2%
6D Pose (5cm 5°[SGZ$^+$13])	**40.6%**	34.1%	11.0%	22.6%	8.1%

Results. Results are shown in the left half of Table 3.1. In 73.7% of test images our approach delivers an accuracy which is suitable for visual effects. See Fig. 3.5 for qualitative results. The high accuracy is reflected in low median errors for translation and rotation, i.e. $2.3cm$ and $5.9°$. The translational error occurs predominantly in z-direction. Our final refinement step, using uncertainty (see Sec. 3.2.4), helps reducing this error. Without this step we loose 4.2% with the 2D projection measure and 17.9% with the measure of Hinterstoisser et al.

Our auto-context (AC) framework boosts performance substantially e.g. by 14.4% with the 2D projection measure (see *Ours w/o AC* in Table 3.1). We observe that regularization of the intermediate auto-context feature channels was absolutely essential. Omitting this step leads to unstable results and performance was actually worse than omitting auto-context altogether (see *Ours w/o reg.*). Using L_2 regularization, compared to L_1, results in a loss of 5.1%. Our auto-context framework largely improves correctness of object label and coordinate predictions. For instance the number of inliers within the ground truth segmentation increases from 17.1% to 33.5%. An object coordinate prediction is an inlier if the mean of the distribution falls within 2cm of the ground truth object coordinate at that pixel. Additionally, it also effectively reduces the uncertainty of predictions. The average differential entropy of the object coordinate distributions reduces by 20% and the average number of modes reduces by 10%.

LINE2D detects objects relatively well (see *2D Bounding Box* score) but fails to reliably estimate correct poses. Without depth information it relies mostly on gradients on the object silhouette. This makes accurate estimation of rotation very difficult. Although [RCT13] introduced improved templates, note that even a perfect template based method can at most assign the nearest neighbor in the training set, since refinement using ICP is not possible with RGB images. In our setup, optimal nearest neighbor matching would achieve a median rotation accuracy of $8.3°$ or 15.5% under the 6D pose measure of [SGZ+13] assuming zero translational error. This is substantially less accurate than our results showing that our method successfully interpolates between training images.

Figure 3.5.: **Pose Estimation From RGB Images. Left:** Four objects, partially overlaid with 3D models with the estimated pose. **Right:** Detecting multiple objects at once. Six out of ten objects have been detected. We accept hypotheses with at least 400 inliers. The bounding box color encodes the object ID.

RGB-D SETTING

The pipeline of this chapter can be easily altered to make use of a depth channel if available. In this case, forest features can be depth-normalized, perspective-n-point is substituted by the Kabsch algorithm, inliers are defined in 3D camera space, and final refinement is based on camera coordinates as in [VNS+15].

Baselines. We compare to the RGB-D pose estimation pipeline described in Chapter 2, and denote it as *Render Score*, here. We also compare to the method of Krull et al. [KBM+15] which is an extension of the aforementioned pipeline by utilizing a CNN in the final pose optimization stage.

Results. Results are shown in the right half of Table 3.2. With the measure of Hinterstoisser et al. [HLI+12], multiple methods approach the limit of 100% correctly estimated poses. The measure of Shotton et al. [SGZ+13] is more sensitive with respect to rotation. With this measure, our new approach estimates 82.1% of poses correctly, which is considerably more than Krull et al. (-9%) or the method of Chapter 2, *Render Score*, (-30%). We attribute this to the robust inlier-based pose optimization which is different from the general purpose CNN in [KBM+15] and the handcrafted scoring function of the previous chapter. In

Table 3.2.: **Pose Estimation from RGB-D Images.** Pose estimation results on the dataset of [HLI+12] for a single object where the object ID is known in advance. *Inlier Score* denotes the method presented in this chapter with auto-context and L_1 regularization, but adapted for RGB-D inputs. *Render Score* denotes the method presented in Chapter 2.

	Our *Inlier Score*	Krull et al. [KBM+15]	Our *Render Score*
2D Projection	**95.7%**	82.6%	81.7%
2D Bounding Box	**99.6%**	98.8%	99.1%
6D Pose (Metric of [HLI+12])	**99.0%**	93.9%	97.4%
6D Pose (5cm 5°[SGZ+13])	**82.1%**	73.1%	52.1%

particular, the scoring function of Chapter 2 has a higher emphasis on model fitting with the comparison of rendered and observed images. Here, we instead rely on the discriminative power of the random forest. Note that in the experiments of the previous chapter our *Render Score* was superior to an inlier-based scoring function, which seems to contradict the results here. However, in Chapter 2, the random forest was trained on rendered and tested on real images, hence its predictions were less reliable.

In the RGB-D setting, the use of auto-context results only in a small improvement (+2.4% with L_1 reg. under 5cm 5°, compared to +1.9% with L_2 reg., and -8.9% w/o reg.). With the 2D projection measure, our approach estimates 95.7% of poses correctly, which is again a large improvement compared to the baselines (+13.1%, +14% respectively). This confirms the suitability of our approach for applying visual effects.

We show qualitative results for individual objects in Fig. 3.6, and for multiple objects in Fig. 3.7. These figures also show the intermediate output of the auto-context random forest at different stack levels. In Appendix A.4.2, we report rates of correctly estimated poses per object, and list median pose errors in X/Y/Z and rotation.

3.3.2. MULTI-OBJECT DETECTION

The previous experiments performed pose estimation of a single object. We now evaluate the detection performance of our approach, i.e. we do not know upfront which objects are present in a test image. We use the occlusion dataset, i.e. our additional annotation of the dataset of Hinterstoisser et al. [HLI+12], see Appendix A.2.2 for details. The occlusion dataset consists of one image sequence with pose annotations for all 9 objects present. The amount of occlusion can be very large making this dataset extremely challenging.

We search for all 13 objects, i.e. even those not present in the sequence, in all images, and keep the strongest response per object. We accept a response if the IoU is at least 50%. We rank all results according to response score which is the inlier count for our method and the matching score for LINE2D. We plot precision vs. recall in Fig. 3.8, left. We compare two variants of our method: Firstly, we divide the budget of 256 RANSAC hypotheses evenly among objects, denoted *Ours w/o sharing*. Secondly, we apply the method described in Sec. 3.2.3, i.e. we sample hypotheses according to the object label distribution, denoted *Ours w/ sharing*. It is important to note that the 256 hypotheses have to be valid, see middle of Sec. 3.2.3. Hence, the run-time can vary depending on the difficulty of finding a valid hypothesis.

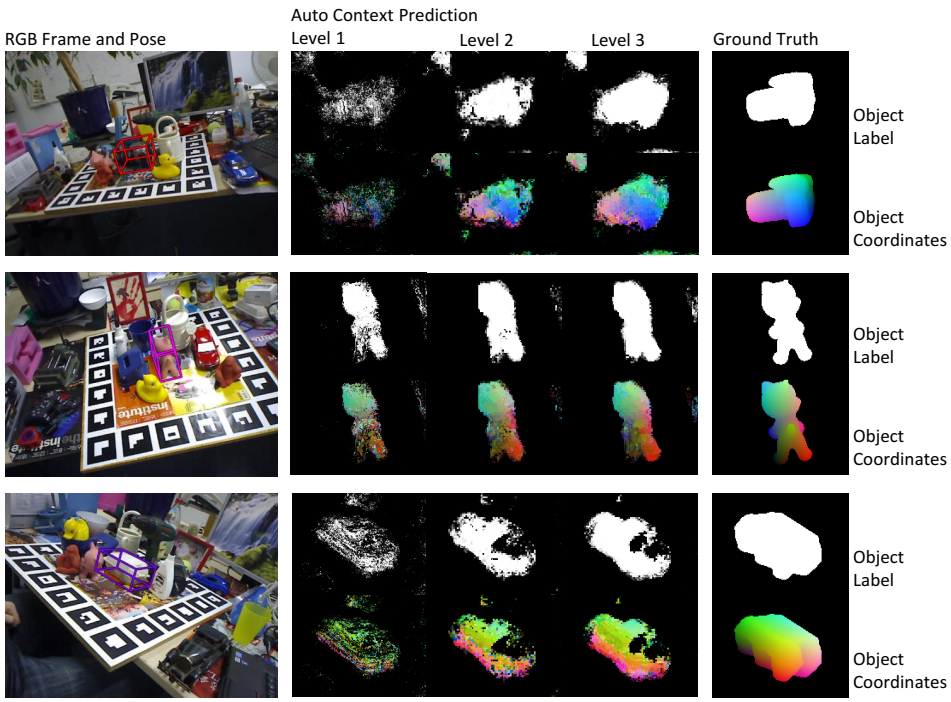

Figure 3.6.: **Single Object Pose Estimation Results. Left:** An RGB frame with the estimated pose of one object. The bounding box color encodes the object ID. **Center:** The prediction of the auto-context random forest at different stack levels zoomed in on the object. The top row shows object probabilities, the bottom row shows object coordinates. Object coordinates are visualized by mapping X/Y/Z to the RGB cube. **Right:** Ground truth segmentation and ground truth object coordinates.

Results. LINE2D achieves 0.21 average precision (AP), due to its sensitivity to occlusion. The two variants of our method score 0.49 AP (*Ours w/o sharing*) and 0.51 AP (*Ours w/ sharing*). As a further experiment, we re-train the auto-context random forest with additional objects to a total of 25 and 50 objects. We repeat the experiment above and measure AP and processing time. Here, we omitted final refinement because it had little impact on the detection performance but would have dominated run time (ca. 100ms per object). See results in Fig. 3.8, right. With more objects, the performance of the schedule w/o sharing drops more rapidly while its run time increases. Processing time is predominately spent on searching for valid hypotheses where objects are occluded or missing. Sampling hypotheses according to the label distribution scales much better. For 50 objects known, on average 21.2 hypotheses are drawn for an object that is present, and only 1.6 hypotheses for an object not visible. In contrast, the naive approach *w/o sharing* allocates 5 hypotheses for each of the 50 objects. Therefore, sampling hypotheses according to the label distribution predicted by our forest lets us concentrate processing time on objects which are likely to be present in the image. Our new, multi-object RANSAC processes all 50 objects in ca. 1s. Note that

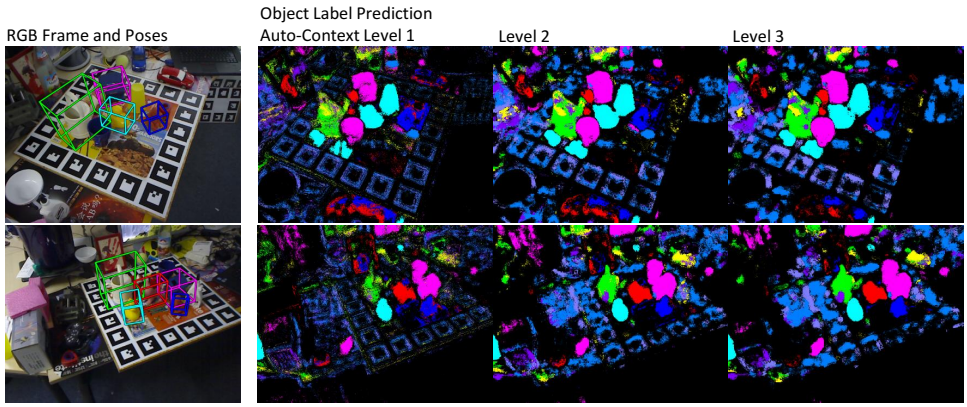

Figure 3.7.: **Multi-Object Pose Estimation Results. Left:** An RGB frame with estimated poses of all objects. Only detections with a minimum number of 400 inliers are shown. **Right:** The label prediction of the auto-context random forest at different stack levels. Bounding box colors encode object IDs.

our implementation runs on CPU only, and is not optimized. We expect a large boost for a GPU port.

3.3.3. CAMERA LOCALIZATION

Camera localization is a variant of object pose estimation where the object of interest is a complete scene, e.g. a room, which occupies the complete image. Shotton et al. [SGZ+13] published an RGB-D dataset of 7 scenes with annotated camera poses, see Appendix A.2.4 for details. Multiple image sequences are given per scene, which were split by [SGZ+13] in training and test. We use the depth channel to calculate object coordinate labels for the training set. This labeling can also be rendered using the available 3D models. Otherwise, we ignored depth channels and estimate the camera pose from RGB only. We kept parameters largely unchanged with respect to Sec. 3.3.1, up to a few exceptions due to the large difference in object size. For example, we increased the inlier threshold τ_{in} to 10px. See Appendix A.4.2 for more details.

Baselines. We compare to a sparse feature-based approach reported in [SGZ+13]. ORB features [RRKB11] are matched to a sparse reconstruction of the scene, and the pose is calculated via perspective-n-point within a RANSAC framework. Furthermore, we compare to PoseNet [KGC15], which is a general purpose CNN architecture, trained to directly regresses the 6D camera pose.

Results. In a first experiment, we follow Shotton et al. and assume that it is known, which of the 7 scenes is present in the test image, see top part of Table 3.3. In 55.2% of test cases we estimate the pose correctly, i.e. within the 5cm 5° threshold. This is an improvement of 14.5% over the sparse feature baseline. Our auto-context framework boosts performance by 3.2% and refinement using uncertainty by 1.0%. Compared to PoseNet, our results

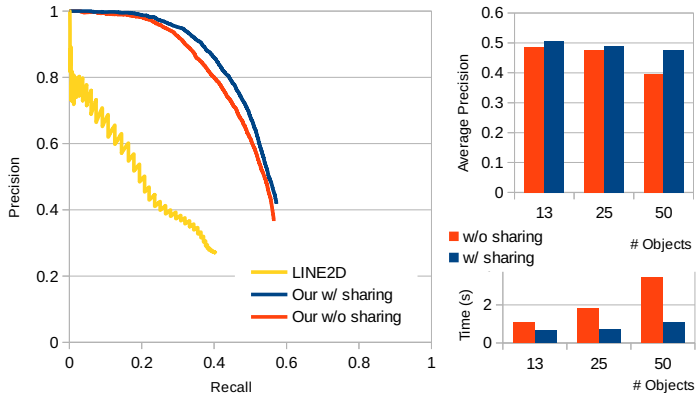

Figure 3.8.: **Detection Experiment on the Occlusion Dataset. Left:** Precision-recall plot for 13 objects. **Right:** Average precision and run time for increasing object count.

Table 3.3.: **Results on the 7-Scenes Dataset Using RGB Only.** The *Avg. Error* is calculated by averaging the median pose error per scene. *Scene Known* denotes the experiment where the scene ID for each image is known in advance. *Scene Unknown* denotes the experiment where the scene ID has to be inferred from the image.

Pose (Scene Known)	5cm 5°	Avg. Error
Sparse RGB [SGZ+13]	40.7%	-
PoseNet [KGC15]	-	46.9cm, 5.4°
Our	**55.2%**	**6.1cm, 2.7°**
Pose (Scene Unknown)		
Our w/ sharing	**50.0%**	**8.5cm, 3.3°**
Our w/o sharing	33.1%	15.0cm, 8.5°

have a translational error that is one order of magnitude smaller. Applying the RGB-D variant of our pipeline, we have 88.1% of correct poses, which is on-par with the results of the state-of-the-art RGB-D method [VNS+15] (89.5%). We observed no improvement when applying auto-context in the RGB-D setting, although the intermediate prediction quality of the forest increased. We show results on individual scenes of the dataset of Shotton et al. [SGZ+13] in Table 3.4. Qualitative results are shown in Fig 3.9.

In a second experiment, we estimate the pose and scene ID jointly from an RGB image, see bottom part of Table 3.3. We train one auto-context random forest for all scenes, and let RANSAC sample 256 object hypotheses per image, according to the object label distributions, see *Our w/ sharing*. Despite the increased difficulty, we observe only a medium loss in performance, i.e. 5.2%. Furthermore, this is substantially better than evenly distributing the budget of hypotheses over the 7 scenes and adjusting parameters to achieve equal runtime, see *Our w/o sharing*. To achieve equal runtime of the two RANSAC variants *w/ sharing* and *w/o sharing* we adjusted parameters in the following way: The final hypothesis of the variant *w/ sharing* will at least pass 8 rounds of pre-emptive RANSAC, i.e. drawing of pixel batches and re-solving PnP. Then, it is refined using uncertainty. For the variant *w/o sharing* there is no minimum number of pre-emptive passes. This results in less pixels drawn and

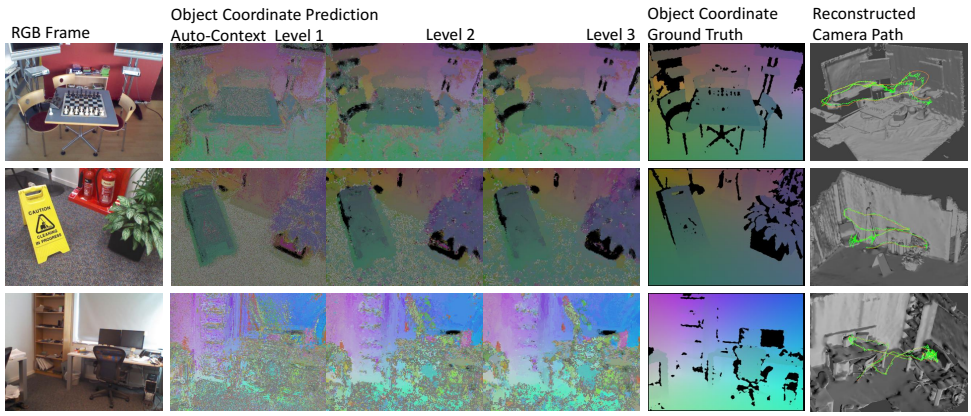

Figure 3.9.: **Camera Localization Results.** We show (from left to right) an RGB frame of a room sequence, the object coordinate prediction of different auto-context levels, the object coordinate ground truth, and the estimated camera path (green) for one complete image sequence. We removed some extreme outlier poses, which results in rare gaps in the estimated path. The ground truth camera path (orange) is also shown for comparison.

less iterations of re-solving PnP. Also, there is no refinement using uncertainty. Using these settings results in a run-time of ca. 700ms for both variants to process all 7 scenes per test image.

3.4. DISCUSSION

In this chapter, we extended our pose estimation system in various ways. We adapted the pipeline to estimate poses from RGB inputs only. Therefore, we substituted the rendering-based scoring function of Chapter 2 by an inlier counting schema which relies only on the discriminative predictions of the random forest. To account for the missing depth cues, we adapted the auto-context algorithm [TB10] for object coordinate regression. The key ingredient to make it work well is a new, robust regularization of the auto-context feature channels. Furthermore, we introduced a refinement schema for extra precision which exploits predicted distributions of object coordinates, and, in contrast to previous work, does not rely on a depth channel. Finally, we presented a multi-object RANSAC schema which lets our system scale to dozens of objects while concentrating computational budget only on those with sufficient evidence in the input image. We demonstrated the capability of our system to estimate poses of single and multiple objects, and to regress accurate camera poses in a camera localization task. The accuracy of our pose estimation system enables the application of convincing visual effects to RGB images, see Fig, 3.10 for some examples.

So far, our pipeline has one learned component: A (auto-context) random forest. While it achieves good results it is unclear how to adjust the training of the random forest for increasing the accuracy of pose estimates. The quality of object coordinate predictions cannot be improved directly, e.g. by stochastic gradient descent (SGD). Furthermore, we measure the performance of our pipeline in terms of accuracy of estimated poses. However, during

Table 3.4.: **Camera Localization Results.** Results of our approach per object on the dataset of Shotton et al. [SGZ+13]. We compare to the sparse feature baseline reported in [SGZ+13], and the state-of-the-art RGB-D-based method of Valentin et al. [VNS+15]. We also compare our median accuracy to PoseNet [KGC15].

	Pose Error <5cm 5°				Median Error	
	RGB		RGB-D		RGB	
Scene	Our	Sparse F. [SGZ+13]	Our	Valentin [VNS+15]	Our	PoseNet [KGC15]
Chess	**94.9%**	70.7%	**99.6%**	99.4%	**1.5cm, 1.3°**	32cm, 3.8°
Fire	**73.5%**	49.9%	94.0%	**94.6%**	**3.0cm, 1.4°**	57cm, 7.0°
Heads	48.1%	**67.6%**	89.3%	**95.9%**	**5.9cm, 3.4°**	30cm, 6.1°
Office	**53.2%**	36.6%	93.4%	**97.0%**	**4.7cm, 1.7°**	48cm, 5.1°
Pumpkin	**54.5%**	21.3%	77.6%	**85.1%**	**4.3cm, 2.1°**	49cm, 4.3°
Kitchen	**42.2%**	29.8%	**91.1%**	89.3%	**5.8cm, 2.2°**	64cm, 4.2°
Stairs	**20.1%**	9.2%	**71.7%**	63.4%	**17.4cm, 7.0°**	48cm, 7.5°
Average	**55.2%**	40.7%	88.1%	**89.5%**	**6.1cm, 2.7°**	46.9cm, 5.4°

Figure 3.10.: **Examples of Visual Effects.** We applied our pose estimation system to a room (left) and the duck object (right). Using our pose estimates, we added a virtual castle resp. sunglasses to the images. The accuracy of our estimated poses, even using inputs which are RGB only, yields high quality results. The result of the LINE2D template-based method [HCI+11] is noticeably off.

training we optimize the information gain of proxy classes per split node of the random forest in a greedy fashion. The relation of this training objective and the pose accuracy is not straight forward, especially because the RANSAC-based pose optimization is robust to errors in the object coordinate prediction. For example, in our experiment regarding camera localization from RGB-D images, we have seen that auto-context improves the quality of object coordinate predictions. However, this did not translate to more accurate pose estimates.

In the next chapter, we will substitute the random forest and our handcrafted scoring function with two CNNs. CNNs, despite having proven powerful machine learning tools in recent years, have the advantage that their predictions can be optimized directly via SGD allowing the use of a wide range of loss functions. We will show how to train a variant of our pose estimation pipeline in an end-to-end fashion, i.e. directly optimizing the accuracy of estimated poses on the training set.

4. END-TO-END LEARNING USING DIFFERENTIAL SAMPLE CONSENSUS (DSAC)

Contents

4.1. INTRODUCTION

In the previous chapters, we have introduced a pose estimation pipeline which combines *learned* components with geometric processing for very accurate and stable results. Recently, deep learning has been shown to be highly successful at image recognition tasks [SZ14, HZRS15a, Gir15, RDGF15], and, increasingly, in other domains including geometry [EF15, KJC16, KGC15, DMR16]. Part of this recent success is the ability to perform end-to-end training, i.e. propagating gradients back through an entire pipeline to allow the direct optimization of a task-specific loss function, see for example [YTLF16, AGT+16, TZTV16].

In this chapter, we are interested in learning components of our pose estimation pipeline such that the accuracy of pose estimates increases. In our approach we predict object coordinates based on local image neighbourhoods, followed by a global model fit. Random Sample Consensus (RANSAC) is an integral component of this wide-spread strategy because it adds robustness to individual errors in the local predictions. We ask the question whether we can train a pipeline like the one presented in this thesis end-to-end. More specifically, we want to learn parameters of a convolutional neural network (CNN) such that models, fit robustly

to its predictions via RANSAC, minimize a task specific loss function. The work associated with this chapter was published in [BKN+17].

Introduced in 1981, RANSAC algorithm [FB81] remains the most important algorithm for robust estimation. It is easy to implement, it can be applied to a wide range of problems, and it is able to handle data with a substantial percentage of outliers, i.e. data points that are not explained by the data model. RANSAC and variants thereof [TZ00, Nis03, CM05] have, for many years, been important tools in computer vision, including multi-view geometry [HZ04], object retrieval [PCI+07], pose estimation [SGZ+13, BKM+14] and simultaneous localization and mapping (SLAM) [MMT15]. Solutions to these diverse tasks often involve a strategy very similar to ours: Local predictions (e.g. feature matches) induce a global model (e.g. a homography). As mentioned before, RANSAC provides robustness to erroneous local predictions. The principles of making RANSAC differentiable which we discuss in this chapter are general and can be transferred to any system with a RANSAC component.

RANSAC works by first creating multiple model hypotheses from small, random subsets of data points. Then it scores each hypothesis by determining its consensus with all data points. Finally, RANSAC selects the hypothesis with the highest consensus as the final output. Unfortunately, this hypothesis selection is non-differentiable meaning that it cannot be used in an end-to-end-trained deep learning pipeline, directly.

A common approach within the deep learning community is to soften non-differentiable operators, e.g. argmax in LIFT [YTLF16] or visual word assignment in NetVLAD [AGT+16]. In the case of RANSAC, the non-differentiable operator is the argmax operator which selects the highest scoring hypothesis. Similar to [YTLF16], we might substitute the argmax for a soft argmax which is a weighted average of arguments [CW10]. We indeed explore this direction but argue that this substitution changes the underlying principle of RANSAC. Instead of learning how to select a good hypothesis, the pipeline learns a (robust) average of hypotheses. We show experimentally that this approach learns to focus on a narrow selection of hypotheses and is prone to overfitting.

Alternatively, we aim to preserve the hard hypothesis selection but treat it as a probabilistic process. We call this approach DSAC – Differentiable Sample Consensus – our new, differentiable counterpart to RANSAC. DSAC allows us to differentiate the *expected* loss of the pipeline w.r.t. to all learnable parameters. This technique is well known in reinforcement learning, for stochastic computation problems like policy gradient approaches [SHWA15].

To demonstrate the principle, we choose the problem of camera localization: From a single RGB image of a known static scene we estimate the 6D camera pose (3D translation and 3D rotation) relative to this scene. Since we are interested in estimating the pose of a single object, a scene, the classification resp. segmentation aspect of our pipeline becomes superfluous, hence we omit it for simplicity in this chapter. We introduce a variant of our system which is end-to-end trainable. In contrast to Chapter 2 and 3, we adopt two CNNs for predicting object coordinates, and for scoring hypotheses. More importantly, the key novelty of this chapter is to replace RANSAC by our new, differentiable DSAC.

Contributions. The contributions of this chapter are:

- We present and discuss two alternative ways of making RANSAC differentiable, by soft argmax selection and probabilistic selection. We call our new RANSAC version, with the latter option, DSAC (Differentiable Sample Consensus).

- We put both options into an end-to-end trainable camera localization pipeline. It follows the same principles discussed so far but contains two separate CNNs, linked by our

new RANSAC.

- We validate experimentally that the option of probabilistic selection is superior, i.e. less sensitive to overfitting, for our application. We conjecture that the advantage of probabilistic selection is allowing hard decisions, and, at the same time, keeping broad distributions over possible decisions.
- We exceed our previous results on camera localization by 11%, which sets a new state-of-the-art at the time of publication of this thesis.

Related Work. Over the last decades, researchers have proposed many variants of the original RANSAC algorithm [FB81]. Most works focus on either or both of two aspects: speed [CMK03, Nis03, CM05], or quality of the final estimate [TZ00, CMK03]. For detailed information about RANSAC variants we refer the reader to [RFP08]. To the best of our knowledge, this work is the first to introduce a differentiable variant of RANSAC for the purpose of end-to-end learning.

In the following, we review previous work on differentiable algorithms. The success of deep learning began with systems in which a CNN processes an image in one forward pass to directly predict the desired output, e.g. class probabilities [KSH12], a semantic segmentation [LSD15] or depth values and normals [EF15]. Given a sufficient amount of training data, CNNs can autonomously discover useful strategies for solving a task at hand, e.g. hierarchical part-structures for object recognition [ZF14].

However, for many computer vision tasks useful strategies have been known for a long time. Recently, researchers started to revisit and encode such strategies explicitly in deep learning pipelines. This can reduce the necessary amount of training data compared to CNNs with an unconstrained architecture [SS16]. Yi et al. [YTLF16] introduced a stack of CNNs that remodels the established sparse feature pipeline of detection, orientation estimation and description originally proposed in [Low04]. Arandjelovic et al. [AGT$^+$16] mapped the Vector of Locally Aggregated Descriptors (VLAD) [AZ13] to a CNN architecture for place recognition. Thewlis et al. [TZTV16] substituted the recursive decoding of Deep Matching [RWHS16] with reverse convolutions for end-to-end trainable dense image matching.

Similar in spirit to these works, we show how to train an established, RANSAC-based computer vision pipeline in an end-to-end fashion. Instead of substituting hard assignments by soft counterparts as in [YTLF16, AGT$^+$16], we enable end-to-end learning by turning the hard selection into a probabilistic process. Thus, we are able to calculate gradients to minimize the expectation of the task loss function [SHWA15].

4.2. METHOD

As a preface to explaining our method, we first briefly review the standard RANSAC algorithm for model fitting, and how it can be applied to the camera localization problem using discriminative object coordinate regression. Then, we explain two strategies of making RANSAC differentiable, using a soft argmax operator and probabilistic selection of hypotheses. Finally, we present a variant of our pose estimation system for camera localization which can be learned end-to-end with the aforementioned strategies.

4.2.1. STANDARD RANSAC

Many problems in computer vision involve fitting a model to a set of data points which in practice usually include outliers due to sensor noise and other factors. The RANSAC algorithm was specifically designed to be able to fit models robustly in the presence of noise [FB81]. Dozens of variations of RANSAC exist [TZ00, CMK03, Nis03, CM05]. We consider a general, basic variant here but the new principles presented in this chapter can be applied to many RANSAC variants, such as to locally-refined pre-emptive RANSAC [SGZ+13]. A basic RANSAC implementation consists of four steps:

1. Generate a set of model hypotheses by sampling minimal subsets of data points.
2. Score hypotheses based on some measure of consensus, e.g. by counting inliers.
3. Select the best scoring hypothesis.
4. Refine the selected hypothesis using additional data points, e.g. the full set of inliers. This step is optional, though in practice important for high accuracy.

We consider an RGB image I consisting of pixels indexed by i. We wish to estimate the parameters $\tilde{\mathbf{h}}$ of a model that explains I. In the camera localization problem this is the 6D camera pose, i.e. the 3D rotation and 3D translation of the camera relative to the scene's coordinate frame. Following the main approach of this thesis, we do not fit model $\tilde{\mathbf{h}}$ directly to image data I, but instead make use of intermediate, noisy 2D-3D correspondences predicted for each pixel: $Y(I) = \{\mathbf{y}(I,i)|\forall i\}$, where $\mathbf{y}(I,i)$ is the object coordinate of pixel i, i.e. a discriminative prediction for where the point imaged at pixel i lives in the 3D object coordinate frame. We will use $\mathbf{y}_i$ as shorthand for $\mathbf{y}(I,i)$. $Y(I)$ denotes the complete set of object coordinate predictions for image I, and we write Y for $Y(I)$. To estimate $\tilde{\mathbf{h}}$ from Y we apply RANSAC as follows:

1. **Generate a Pool of Hypotheses.** Each hypothesis is generated from a subset of correspondences. This subset contains the minimal number of correspondences to compute a unique solution. We call this a minimal set Y_J with correspondence indices $J = \{j_1, ..., j_n\}$, where n is the minimal set size. To create the set, we uniformly sample n correspondence indices: $j_m \in [1, \dots, |Y|]$ to get $Y_J := \{\mathbf{y}_{j_1}, ..., \mathbf{y}_{j_n}\}$. We assume a function $\mathbf{H}$ which generates a model hypothesis as $\mathbf{h}_J = \mathbf{H}(Y_J)$ from the minimal set Y_J. In our application, $\mathbf{H}$ is the perspective-n-point (PnP) algorithm [GHTC03], and $n = 4$.

2. **Score Hypotheses.** The scalar function $s(\mathbf{h}_J, Y)$ measures the consensus or quality of hypothesis $\mathbf{h}_J$, e.g. by counting inlier correspondences. As in Chapter 3, we define an inlier, specific to our application, via the re-projection error of object coordinate $\mathbf{y}_i$:

$$e_i = \|\mathbf{p}_i - C\mathbf{h}_J\mathbf{y}_i\|, \tag{4.1}$$

   where $\mathbf{p}_i$ is the 2D location of pixel i and C is the camera projection matrix. We call $\mathbf{y}_i$ an inlier if $e_i < \tau_{in}$, where τ_{in} is the inlier threshold. In the following, instead of counting inliers, we aim to learn $s(\mathbf{h}_J, Y)$ to directly regress the hypothesis score from re-projection errors e_i, as we will explain shortly.

3. **Select Best Hypothesis.** We take

$$\mathbf{h}_{\mathrm{AM}} = \underset{\mathbf{h}_J}{\mathrm{argmax}}\, s(\mathbf{h}_J, Y), \tag{4.2}$$

   where AM stands for argmax.

4. **Refine Hypothesis.** The selected hypothesis $\mathbf{h}_{\mathrm{AM}}$ is refined using function $\mathbf{R}(\mathbf{h}_{\mathrm{AM}}, Y)$. Refinement may use all correspondences Y. A common approach is to select a set of inliers from Y and recalculate function $\mathbf{H}$ on this set. The refined pose is the output of the algorithm $\tilde{\mathbf{h}}_{\mathrm{AM}} = \mathbf{R}(\mathbf{h}_{\mathrm{AM}}, Y)$.

4.2.2. LEARNING IN A RANSAC PIPELINE

So far, we had a single learned component, namely the regression forest that made the predictions $\mathbf{y}(I, i)$. Krull et al. [KBM+15] extended the approach of Chapter 2 to also learn the scoring function $s(\mathbf{h}_J, Y)$ as a generalization of our handcrafted score. However, these two components have thus far been learned separately.

We instead aim to learn both, the object coordinate prediction and the scoring function, and to do so jointly in an end-to-end fashion within a RANSAC framework. Making the parameterizations explicit, we have $\mathbf{y}(I, i; \mathbf{w})$ and $s(\mathbf{h}_J, Y; \mathbf{v})$. We aim to learn parameters $\mathbf{w}$ and $\mathbf{v}$ where $\mathbf{w}$ affects the quality of poses that we generate, and $\mathbf{v}$ affects the selection process which should choose a good hypothesis. Note that neither $\mathbf{w}$ nor $\mathbf{v}$ influence the initial sampling of minimal sets Y_J which is done uniformly and thus has no learnable parameters. To declutter notation, we denote (transitive) dependence on parameters with a superscript $\mathbf{w}$ resp. $\mathbf{v}$. For example, we write $Y^{\mathbf{w}}$ and $\mathbf{h}_J^{\mathbf{w}}$ to reflect that object coordinate predictions and hypotheses depend on parameters $\mathbf{w}$. Similarly, we write $\mathbf{h}_{\mathrm{AM}}^{\mathbf{w},\mathbf{v}}$ to reflect that the selected hypothesis depends on $\mathbf{w}$ and $\mathbf{v}$. Finally, we abbreviate score $s(\mathbf{h}_J^{\mathbf{w}}, Y^{\mathbf{w}}; \mathbf{v})$ as $s_J^{\mathbf{w},\mathbf{v}}$.

We would like to find parameters $\mathbf{w}$ and $\mathbf{v}$ such that the loss ℓ of the final, refined hypotheses over a training set of images is minimized, i.e.

$$\tilde{\mathbf{w}}, \tilde{\mathbf{v}} = \underset{\mathbf{w},\mathbf{v}}{\operatorname{argmin}} \sum_I \ell(\mathbf{R}(\mathbf{h}_{\mathrm{AM}}^{\mathbf{w},\mathbf{v}}, Y^{\mathbf{w}}), \mathbf{h}^*), \tag{4.3}$$

where $\mathbf{h}^*$ are ground truth model parameters for I. To allow end-to-end learning, we need to differentiate w.r.t. $\mathbf{w}$ and $\mathbf{v}$. We assume a differentiable loss ℓ and a differentiable refinement function $\mathbf{R}$.

One might consider differentiating $\mathbf{h}_{\mathrm{AM}}^{\mathbf{w},\mathbf{v}}$ w.r.t. to $\mathbf{w}$ via the minimal set Y_J of the single selected hypothesis of Eq. 4.2. But learning a RANSAC pipeline in this fashion fails because the selection process itself depends on $\mathbf{w}$ and $\mathbf{v}$ which is not represented in the gradients of the hypothesis selected via argmax. We observed in early experiments that the training loss immediately increases without recovering. Parameters $\mathbf{v}$ influence the selection directly via the scoring function $s(\mathbf{h}, Y; \mathbf{v})$, and parameters $\mathbf{w}$ influence the quality of competing hypotheses $\mathbf{h}$.

We next present two approaches to learn parameters $\mathbf{w}$ and $\mathbf{v}$ – soft argmax selection and probabilistic selection – that do model the dependency of the selection process on the parameters.

SOFT argmax SELECTION (SOFTAM)

To solve the problem of non-differentiability, one can relax the argmax operator of Eq. 4.2 and substitute it for a soft argmax operator [CW10]. The soft argmax turns the hypothesis selection into a weighted average of hypotheses:

$$\mathbf{h}_{\mathrm{SoftAM}}^{\mathbf{w},\mathbf{v}} = \sum_J P(J|\mathbf{v}, \mathbf{w}) \mathbf{h}_J^{\mathbf{w}} \tag{4.4}$$

which averages over candidate hypotheses $\mathbf{h}_J^{\mathbf{w}}$ with

$$P(J|\mathbf{v},\mathbf{w}) = \frac{\exp(s_J^{\mathbf{w},\mathbf{v}})}{\sum_{J'}\exp(s_{J'}^{\mathbf{w},\mathbf{v}})}. \tag{4.5}$$

In this variant, scoring function $s_J^{\mathbf{w},\mathbf{v}}$ has to predict weights that lead to a robust average of hypotheses (i.e. model parameters). This means that model parameters corrupted by outliers should receive sufficiently small weights such that they do not affect the accuracy of $\mathbf{h}_{\mathrm{SoftAM}}^{\mathbf{w},\mathbf{v}}$. Substituting $\mathbf{h}_{\mathrm{AM}}^{\mathbf{w},\mathbf{v}}$ for $\mathbf{h}_{\mathrm{SoftAM}}^{\mathbf{w},\mathbf{v}}$ in Eq. 4.3 allows us to calculate gradients to learn parameters $\mathbf{w}$ and $\mathbf{v}$. More details on the derivatives follow below.

By utilizing the soft argmax operator, we diverge from the RANSAC principle of making one hard decision for a hypothesis. Soft argmax hypothesis selection bears similarity with an independent strain within the field of robust optimization, namely robust averaging, see e.g. the work of Hartley et al. [HAT11]. While we explore soft argmax selection in the experimental evaluation, we introduce an alternative in the next section that preserves the hard hypothesis selection, and is empirically superior for our task.

Further Details on the Derivatives of soft argmax. To learn our camera localization pipeline in an end-to-end fashion, we have to calculate the derivatives of the task loss function $\ell(\mathbf{R}(\mathbf{h}_{\mathrm{SoftAM}}^{\mathbf{w},\mathbf{v}}, Y^{\mathbf{w}}), \mathbf{h}^*)$ w.r.t. to learnable parameters. In the following, we show the derivative w.r.t. parameters $\mathbf{w}$, but derivation w.r.t. parameters $\mathbf{v}$ works similarly.

Applying the chain rule and calculating the total derivative of refinement $\mathbf{R}$, we get:

$$\frac{\partial}{\partial \mathbf{w}}\ell(\mathbf{R}(\mathbf{h}_{\mathrm{SoftAM}}^{\mathbf{w},\mathbf{v}}, Y^{\mathbf{w}}), \mathbf{h}^*) = \frac{\partial \ell}{\partial \mathbf{R}}\left(\frac{\partial \mathbf{R}}{\partial \mathbf{h}_{\mathrm{SoftAM}}^{\mathbf{w},\mathbf{v}}}\frac{\partial \mathbf{h}_{\mathrm{SoftAM}}^{\mathbf{w},\mathbf{v}}}{\partial \mathbf{w}} + \frac{\partial \mathbf{R}}{\partial Y^{\mathbf{w}}}\frac{\partial Y^{\mathbf{w}}}{\partial \mathbf{w}}\right) \tag{4.6}$$

Since $\mathbf{h}_{\mathrm{SoftAM}}^{\mathbf{w},\mathbf{v}}$ is a weighted average of hypotheses (see Eq. 4.4) we can differentiate it as follows:

$$\begin{aligned}\frac{\partial}{\partial \mathbf{w}}\mathbf{h}_{\mathrm{SoftAM}}^{\mathbf{w},\mathbf{v}} &= \frac{\partial}{\partial \mathbf{w}}\sum_J P(J|\mathbf{v},\mathbf{w})\mathbf{h}_J^{\mathbf{w}} \\ &= \sum_J\left(\mathbf{h}_J^{\mathbf{w}}\frac{\partial}{\partial \mathbf{w}}P(J|\mathbf{v},\mathbf{w}) + P(J|\mathbf{v},\mathbf{w})\frac{\partial}{\partial \mathbf{w}}\mathbf{h}_J^{\mathbf{w}}\right)\end{aligned} \tag{4.7}$$

Weights $P(J|\mathbf{v},\mathbf{w})$ follow a softmax distribution of hypothesis scores (see Eq. 4.5). Hence, we can differentiate further as follows:

$$\begin{aligned}\frac{\partial}{\partial \mathbf{w}}P(J|\mathbf{v},\mathbf{w}) &= \frac{\partial}{\partial \mathbf{w}}\frac{\exp(s_J^{\mathbf{w},\mathbf{v}})}{\sum_{J'}\exp(s_{J'}^{\mathbf{w},\mathbf{v}})} \\ &= \frac{\frac{\partial}{\partial \mathbf{w}}\exp(s_J^{\mathbf{w},\mathbf{v}})}{\sum_{J'}\exp(s_{J'}^{\mathbf{w},\mathbf{v}})} - \frac{\exp(s_J^{\mathbf{w},\mathbf{v}})}{(\sum_{J'}\exp(s_{J'}^{\mathbf{w},\mathbf{v}}))^2}\sum_{J'}\frac{\partial}{\partial \mathbf{w}}\exp(s_{J'}^{\mathbf{w},\mathbf{v}}) \\ &= \frac{\exp(s_J^{\mathbf{w},\mathbf{v}})\frac{\partial}{\partial \mathbf{w}}s_J^{\mathbf{w},\mathbf{v}}}{\sum_{J'}\exp(s_{J'}^{\mathbf{w},\mathbf{v}})} - \frac{\exp(s_J^{\mathbf{w},\mathbf{v}})\sum_{J'}\exp(s_{J'}^{\mathbf{w},\mathbf{v}})\frac{\partial}{\partial \mathbf{w}}s_{J'}^{\mathbf{w},\mathbf{v}}}{(\sum_{J''}\exp(s_{J''}^{\mathbf{w},\mathbf{v}}))^2} \\ &= P(J|\mathbf{v},\mathbf{w})\frac{\partial}{\partial \mathbf{w}}s_J^{\mathbf{w},\mathbf{v}} - P(J|\mathbf{v},\mathbf{w})\frac{\sum_{J'}\exp(s_{J'}^{\mathbf{w},\mathbf{v}})\frac{\partial}{\partial \mathbf{w}}s_{J'}^{\mathbf{w},\mathbf{v}}}{\sum_{J''}\exp(s_{J''}^{\mathbf{w},\mathbf{v}})} \\ &= P(J|\mathbf{v},\mathbf{w})\frac{\partial}{\partial \mathbf{w}}s_J^{\mathbf{w},\mathbf{v}} - P(J|\mathbf{v},\mathbf{w})\sum_{J'}P(J'|\mathbf{v},\mathbf{w})\frac{\partial}{\partial \mathbf{w}}s_{J'}^{\mathbf{w},\mathbf{v}} \\ &= P(J|\mathbf{v},\mathbf{w})\left(\frac{\partial}{\partial \mathbf{w}}s_J^{\mathbf{w},\mathbf{v}} - \mathbb{E}_{J'\sim P(J'|\mathbf{v},\mathbf{w})}\left[\frac{\partial}{\partial \mathbf{w}}s_{J'}^{\mathbf{w},\mathbf{v}}\right]\right)\end{aligned} \tag{4.8}$$

Combining this with Eq. 4.7 we have

$$\begin{aligned}\frac{\partial}{\partial \mathbf{w}}\mathbf{h}^{\mathbf{w},\mathbf{v}}_{\mathrm{SoftAM}} &= \sum_J \left(\mathbf{h}^{\mathbf{w}}_J \frac{\partial}{\partial \mathbf{w}} P(J|\mathbf{v},\mathbf{w}) + P(J|\mathbf{v},\mathbf{w}) \frac{\partial}{\partial \mathbf{w}} \mathbf{h}^{\mathbf{w}}_J \right) \\ &= \sum_J \left(P(J|\mathbf{v},\mathbf{w}) \mathbf{h}^{\mathbf{w}}_J \left(\frac{\partial}{\partial \mathbf{w}} s^{\mathbf{w},\mathbf{v}}_J - \mathbb{E}_{J'} \left[\frac{\partial}{\partial \mathbf{w}} s^{\mathbf{w},\mathbf{v}}_{J'} \right] \right) + P(J|\mathbf{v},\mathbf{w}) \frac{\partial}{\partial \mathbf{w}} \mathbf{h}^{\mathbf{w}}_J \right) \\ &= \mathbb{E}_{J\sim P(J|\mathbf{v},\mathbf{w})} \left[\mathbf{h}^{\mathbf{w}}_J \frac{\partial}{\partial \mathbf{w}} s^{\mathbf{w},\mathbf{v}}_J - \mathbf{h}^{\mathbf{w}}_J \mathbb{E}_{J'\sim P(J'|\mathbf{v},\mathbf{w})} \left[\frac{\partial}{\partial \mathbf{w}} s^{\mathbf{w},\mathbf{v}}_{J'} \right] + \frac{\partial}{\partial \mathbf{w}} \mathbf{h}^{\mathbf{w}}_J \right].\end{aligned} \tag{4.9}$$

This formulation is inconvenient to use in combination with backpropagation because the derivatives of the scoring function $s^{\mathbf{w},\mathbf{v}}_J$ appear in two separate places, in an outer loop over J and an inner loop over J'. To avoid calculating backward passes multiple times or caching data, we factor out $s^{\mathbf{w},\mathbf{v}}_J$ as follows. First, to reduce the notational overhead, we write $\mathbb{E}_J[.]$ for $\mathbb{E}_{J\sim P(J|\mathbf{v},\mathbf{w})}[.]$. We also isolate the inner loop by splitting the expectation.

$$\frac{\partial}{\partial \mathbf{w}}\mathbf{h}^{\mathbf{w},\mathbf{v}}_{\mathrm{SoftAM}} = \mathbb{E}_J \left[\mathbf{h}^{\mathbf{w}}_J \frac{\partial}{\partial \mathbf{w}} s^{\mathbf{w},\mathbf{v}}_J \right] - \mathbb{E}_J \left[\mathbf{h}^{\mathbf{w}}_J \mathbb{E}_{J'} \left[\frac{\partial}{\partial \mathbf{w}} s^{\mathbf{w},\mathbf{v}}_{J'} \right] \right] + \mathbb{E}_J \left[\frac{\partial}{\partial \mathbf{w}} \mathbf{h}^{\mathbf{w}}_J \right] \tag{4.10}$$

The nested expectation can be re-structured as follows.

$$\begin{aligned}\mathbb{E}_J \left[\mathbf{h}^{\mathbf{w}}_J \mathbb{E}_{J'} \left[\frac{\partial}{\partial \mathbf{w}} s^{\mathbf{w},\mathbf{v}}_{J'} \right] \right] &= \sum_J P(J) \mathbf{h}^{\mathbf{w}}_J \left(\sum_{J'} P(J') \frac{\partial}{\partial \mathbf{w}} s^{\mathbf{w},\mathbf{v}}_{J'} \right) \\ &= \sum_J \sum_{J'} \mathbf{h}^{\mathbf{w}}_J P(J) P(J') \frac{\partial}{\partial \mathbf{w}} s^{\mathbf{w},\mathbf{v}}_{J'} \\ &= \sum_{J'} \sum_J \mathbf{h}^{\mathbf{w}}_J P(J) P(J') \frac{\partial}{\partial \mathbf{w}} s^{\mathbf{w},\mathbf{v}}_{J'} && \text{(Commuting sums.)} \\ &= \sum_J \sum_{J'} \mathbf{h}^{\mathbf{w}}_{J'} P(J') P(J) \frac{\partial}{\partial \mathbf{w}} s^{\mathbf{w},\mathbf{v}}_J && \text{(Renaming indices.)} \\ &= \mathbb{E}_J \left[\sum_{J'} \mathbf{h}^{\mathbf{w}}_{J'} P(J') \frac{\partial}{\partial \mathbf{w}} s^{\mathbf{w},\mathbf{v}}_J \right]\end{aligned} \tag{4.11}$$

Finally, we insert this back into Eq. 4.10 and isolate the derivatives of the scoring function which allows us to use backpropagation:

$$\frac{\partial}{\partial \mathbf{w}}\mathbf{h}^{\mathbf{w},\mathbf{v}}_{\mathrm{SoftAM}} = \mathbb{E}_J \left[\left(\mathbf{h}^{\mathbf{w}}_J - \sum_{J'} \mathbf{h}^{\mathbf{w}}_{J'} P(J') \right) \frac{\partial}{\partial \mathbf{w}} s^{\mathbf{w},\mathbf{v}}_J \right] + \mathbb{E}_J \left[\frac{\partial}{\partial \mathbf{w}} \mathbf{h}^{\mathbf{w}}_J \right]. \tag{4.12}$$

The derivative of $\mathbf{h}^{\mathbf{w},\mathbf{v}}_{\mathrm{SoftAM}}$ is used in Eq. 4.6 to learn end-to-end with the soft argmax strategy.

PROBABILISTIC SELECTION (DSAC)

As an alternative to the soft argmax selection, we can substitute the deterministic selection of the highest scoring model hypothesis in Eq. 4.2 by a probabilistic selection, i.e. we chose a hypothesis probabilistically according to:

$$\mathbf{h}^{\mathbf{w},\mathbf{v}}_{\mathrm{DSAC}} = \mathbf{h}^{\mathbf{w}}_J, \text{ with } J \sim P(J|\mathbf{v},\mathbf{w}), \tag{4.13}$$

where $P(J|\mathbf{v},\mathbf{w})$ is the softmax distribution of scores predicted by $s_J^{\mathbf{w},\mathbf{v}}$ (see Eq. 4.5).

The inspiration for this approach comes from policy gradient approaches in reinforcement learning that involve the minimization of a loss function defined over a stochastic process [SHWA15]. Similarly, we are able to learn parameters $\mathbf{w}$ and $\mathbf{v}$ that minimize the expectation of the loss of the stochastic process defined in Eq. 4.13:

$$\tilde{\mathbf{w}},\tilde{\mathbf{v}} = \underset{\mathbf{w},\mathbf{v}}{\operatorname{argmin}} \sum_{I\in\mathcal{I}} \mathbb{E}_{J\sim P(J|\mathbf{v},\mathbf{w})} \left[\ell(\mathbf{R}(\mathbf{h}_J^{\mathbf{w}}, Y^{\mathbf{w}}))\right]. \tag{4.14}$$

By optimizing the *expected* loss during training, the system will try to increase the probability of good hypotheses while making them also more accurate. One the other hand, the system will try to decrease the probability of bad hypotheses while the impact of their accuracy is limited.

As shown in [SHWA15], we can calculate the derivative w.r.t. parameters $\mathbf{w}$ by using the identity $\frac{\partial}{\partial \mathbf{w}} P(J|\mathbf{v},\mathbf{w}) = P(J|\mathbf{v},\mathbf{w}) \frac{\partial}{\partial \mathbf{w}} \log P(J|\mathbf{v},\mathbf{w})$ as follows (similarly for parameters $\mathbf{v}$):

$$\begin{aligned}
\frac{\partial}{\partial \mathbf{w}} \mathbb{E}_{J\sim P(J|\mathbf{v},\mathbf{w})} \left[\ell(\cdot)\right] &= \frac{\partial}{\partial \mathbf{w}} \sum_J P(J|\mathbf{v},\mathbf{w}) \ell(\cdot) \\
&= \sum_J \ell(\cdot) \frac{\partial}{\partial \mathbf{w}} P(J|\mathbf{v},\mathbf{w}) + P(J|\mathbf{v},\mathbf{w}) \frac{\partial}{\partial \mathbf{w}} \ell(\cdot) \\
&= \sum_J \ell(\cdot) P(J|\mathbf{v},\mathbf{w}) \frac{\partial}{\partial \mathbf{w}} \log P(J|\mathbf{v},\mathbf{w}) + P(J|\mathbf{v},\mathbf{w}) \frac{\partial}{\partial \mathbf{w}} \ell(\cdot) \\
&= \mathbb{E}_{J\sim P(J|\mathbf{v},\mathbf{w})} \left[\ell(\cdot) \frac{\partial}{\partial \mathbf{w}} \log P(J|\mathbf{v},\mathbf{w}) + \frac{\partial}{\partial \mathbf{w}} \ell(\cdot)\right],
\end{aligned} \tag{4.15}$$

where we use $\ell(\cdot)$ as a stand-in for $\ell(\mathbf{R}(\mathbf{h}_J^{\mathbf{w},\mathbf{v}}, Y^{\mathbf{w}}), \mathbf{h}^*)$. The derivative of the expectation is an expectation over derivatives of the loss and the $\log$ probabilities of model hypotheses. Further derivations follow below.

We call this method of differentiating RANSAC that preserves hard hypothesis selection DSAC – Differentiable Sample Consensus. See Fig. 4.1 for a schematic view of DSAC in comparison to the RANSAC variants introduced at the beginning of this section. While learning parameters with the vanilla RANSAC is not possible, as mentioned before, both new variants, SoftAM and DSAC, are sensible options which we evaluate in the experimental section.

Further Details on the Derivatives of DSAC. Using the DSAC strategy, we learn our pose estimation pipeline by minimizing the expectation of the task loss function, as shown in Eq. 4.15. We differentiate $\frac{\partial}{\partial \mathbf{w}} \ell(\cdot) = \frac{\partial}{\partial \mathbf{w}} \ell(\mathbf{R}(\mathbf{h}_J^{\mathbf{w},\mathbf{v}}, Y^{\mathbf{w}}), \mathbf{h}^*)$ following Eq. 4.6, and log prob-

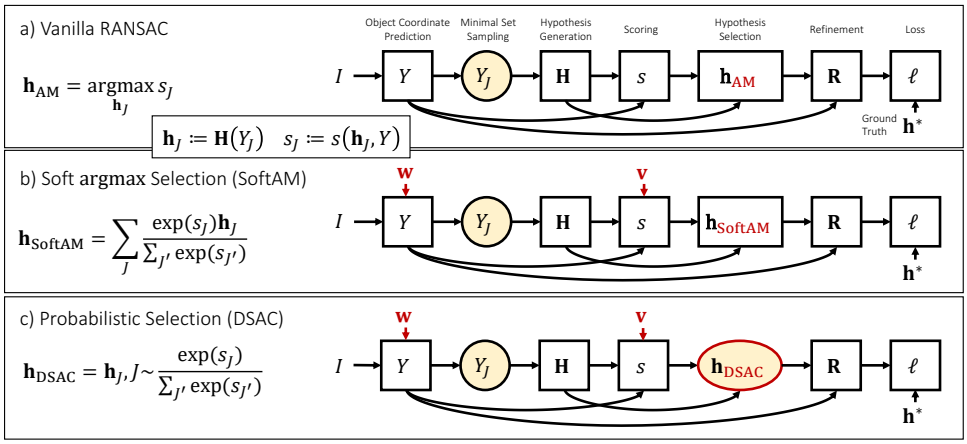

Figure 4.1.: **Stochastic Computation Graphs** [SHWA15]. A graphical representation of three RANSAC variants investigated in this thesis. The variants differ in the way they select the final model hypothesis: **a)** non-differentiable, vanilla RANSAC with hard, deterministic argmax selection; **b)** differentiable RANSAC with deterministic, soft argmax selection; **c)** differentiable RANSAC with hard, probabilistic selection (named DSAC). Nodes shown as boxes represent deterministic functions while circular nodes with yellow background represent probabilistic functions. Arrows indicate dependency in computation. All differences between a), b) and c) are marked in red.

abilities $\log P(J|\mathbf{v}, \mathbf{w})$ as:

$$\begin{aligned}
\frac{\partial}{\partial \mathbf{w}} \log P(J|\mathbf{v}, \mathbf{w}) &= \frac{\partial}{\partial \mathbf{w}} \log \frac{\exp(s_J^{\mathbf{w},\mathbf{v}})}{\sum_{J'} \exp(s_{J'}^{\mathbf{w},\mathbf{v}})} \\
&= \frac{\partial}{\partial \mathbf{w}} s_J^{\mathbf{w},\mathbf{v}} - \frac{\partial}{\partial \mathbf{w}} \log \sum_{J'} \exp(s_{J'}^{\mathbf{w},\mathbf{v}}) \\
&= \frac{\partial}{\partial \mathbf{w}} s_J^{\mathbf{w},\mathbf{v}} - \sum_{J'} \frac{\exp(s_{J'}^{\mathbf{w},\mathbf{v}})}{\sum_{J''} \exp(s_{J''}^{\mathbf{w},\mathbf{v}})} \frac{\partial}{\partial \mathbf{w}} s_{J'}^{\mathbf{w},\mathbf{v}} \\
&= \frac{\partial}{\partial \mathbf{w}} s_J^{\mathbf{w},\mathbf{v}} - \sum_{J'} P(J'|\mathbf{v}, \mathbf{w}) \frac{\partial}{\partial \mathbf{w}} s_{J'}^{\mathbf{w},\mathbf{v}} \\
&= \frac{\partial}{\partial \mathbf{w}} s_J^{\mathbf{w},\mathbf{v}} - \mathbb{E}_{J' \sim P(J'|\mathbf{v},\mathbf{w})} \left[\frac{\partial}{\partial \mathbf{w}} s_{J'}^{\mathbf{w},\mathbf{v}} \right].
\end{aligned} \tag{4.16}$$

Put back in Eq. 4.15, this results in

$$\frac{\partial}{\partial \mathbf{w}} \mathbb{E}_J \left[\ell(\cdot) \right] = \mathbb{E}_J \left[\ell(\cdot) \left(\frac{\partial}{\partial \mathbf{w}} s_J^{\mathbf{w},\mathbf{v}} - \mathbb{E}_{J'} \left[\frac{\partial}{\partial \mathbf{w}} s_{J'}^{\mathbf{w},\mathbf{v}} \right] \right) + \frac{\partial}{\partial \mathbf{w}} \ell(\cdot) \right]. \tag{4.17}$$

As in Eq. 4.9, the term $s_J^{\mathbf{w},\mathbf{v}}$ appears in two separate places. Following Eq. 4.10 et seqq., we simplify further by isolating $s_J^{\mathbf{w},\mathbf{v}}$.

$$\frac{\partial}{\partial \mathbf{w}} \mathbb{E}_J \left[\ell_J(\cdot) \right] = \mathbb{E}_J \left[\left(\ell_J(\cdot) - \sum_{J'} \ell_{J'}(\cdot) P(J') \right) \frac{\partial}{\partial \mathbf{w}} s_J^{\mathbf{w},\mathbf{v}} \right] + \mathbb{E}_J \left[\frac{\partial}{\partial \mathbf{w}} \ell_J(\cdot) \right]. \tag{4.18}$$

Note that we introduced a subindex for loss $\ell(\cdot)$ to keep track of the hypothesis it belongs to.

4.2.3. DIFFERENTIABLE CAMERA LOCALIZATION

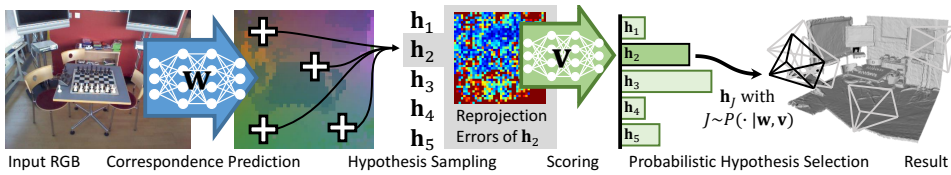

Figure 4.2.: **Differentiable Camera Localization Pipeline.** Given an RGB image, we let a CNN with parameters $\mathbf{w}$ predict object coordinates, i.e. dense 2D-3D correspondences. From these, we sample minimal sets of four object coordinates and create a pool of hypotheses $\mathbf{h}$. For each hypothesis, we create an image of re-projection errors which is scored by a second CNN with parameters $\mathbf{v}$. We select a hypothesis probabilistically according to the score distribution. The selected pose is also refined.

We demonstrate the principles for differentiating RANSAC for the task of one-shot camera localization from an RGB image. In the following, we describe a variant of the pose estimation pipeline of Chapter 3 which can be learned in an end-to-end fashion using the strategies described above.

In Chapter 3, we used an auto-context random forest to predict multi-modal object coordinate distributions per image patch. After that, minimal sets of four object coordinates were randomly sampled and the PnP algorithm [GHTC03] was applied to create a pool of camera pose hypotheses. A pre-emptive RANSAC schema iteratively refined, re-scored and rejected hypotheses until only one remained. The pre-emptive RANSAC scored hypotheses by counting inlier object coordinates, i.e. object coordinates $\mathbf{y}_i$ for which re-projection error $e_i < \tau_{in}$. In a last step, the final, remaining hypothesis was further optimized using the uncertainty of the object coordinate distributions predicted by the auto-context random forest. The differentiable camera localization pipeline of this chapter differs in the following aspects:

- Instead of a random forest, we use a CNN (called 'Coordinate CNN' below) to predict object coordinates. For each 42×42 pixel image patch, it predicts a object coordinate point estimate. We use a VGG style architecture with 13 layers and 33M parameters. To reduce test time we process only 40×40 patches per image which results in 40×40 object coordinate predictions.

- We score hypotheses using a second CNN (called 'Score CNN' below). We took inspiration from the work of Krull et al. [KBM+15] which is an extension of the system described in Chapter 2. Krull et al. learn a CNN to compare rendered and observed images. Instead, our Score CNN predicts hypothesis consensus based on re-projection errors. For each of the 40×40 object coordinate predictions $\mathbf{y}_i$, we calculate the re-projection error e_i for hypothesis $\mathbf{h}_J$ (see Eq. 4.1). This results in a 40×40 re-projection error image, which we feed into the Score CNN, a VGG style architecture with 13 layers and 6M parameters.

- Instead of the pre-emptive RANSAC schema, we score hypotheses only once and select the final pose either by applying the soft argmax operator (SoftAM) or by probabilistic selection according to the softmaxed scores (DSAC).

- Only the final pose is refined. We choose inlier object coordinate predictions, i.e. object coordinates $\mathbf{y}_i$ with re-projection error $e_i < \tau_{in}$, and solve PnP [LMNF09] again using this set. This is iterated multiple times. Since the Coordinate CNN predicts only point estimates we do no further pose optimization using uncertainty.

See Fig. 4.2 for an overview of our differentiable camera localization pipeline. Where applicable we use the same parameter values used in experiments of Chapter 3, e.g. sampling 256 hypotheses, using 8 refinement steps and an inlier threshold of $\tau_{in} = 10$px.

4.3. EXPERIMENTS

For comparability to other methods, we again show results on the 7-Scenes dataset [SGZ+13] which is widely used by camera localization methods. The dataset consists of RGB-D images of 7 indoor environments where each frame is annotated with its 6D camera pose. The data of each scene is comprised of multiple sequences (= independent camera paths) which are assigned either to test or training. The number of images per scene ranges from 1k to 7k for training resp. test. We omit the depth channels and estimate poses using RGB images only. See Appendix A.2.4 for more details on this dataset.

We measure accuracy by the percentage of images for which the pose error is below 5° and 5cm. See Appendix A.3.3 for details on calculating this error. For training, we use the following differentiable loss which is closely correlated with the task loss:

$$\ell_{\text{pose}}(\tilde{\mathbf{h}}, \mathbf{h}^*) = \max(\measuredangle(\tilde{\boldsymbol{\theta}}, \boldsymbol{\theta}^*), \|\tilde{\mathbf{t}} - \mathbf{t}^*\|), \tag{4.19}$$

where $\mathbf{h} = (\boldsymbol{\theta}, \mathbf{t})$, $\boldsymbol{\theta}$ denotes the axis-angle representation of the camera rotation, and $\mathbf{t}$ is the camera translation. We measure angle $\measuredangle(\tilde{\boldsymbol{\theta}}, \boldsymbol{\theta}^*)$ between estimated and ground truth rotation in degree, and distance $\|\tilde{\mathbf{t}} - \mathbf{t}^*\|$ between estimated and ground truth translation in cm.

Since the dataset does not include a designated validation set, we separated multiple blocks of 100 consecutive frames from the training data to be used as validation data (in total 10% per scene). We fixed all learning parameters on the validation set (e.g. learning rate and total amount of parameter updates). Once all hyper parameters were fixed, we re-trained on the full training set.

4.3.1. COMPONENTWISE TRAINING

Our pipeline contains two trainable components, namely the Coordinate CNN and the Score CNN. First, we explain how to train both components using surrogate losses, i.e. train them not in an end-to-end fashion but separately. End-to-end training using differentiable RANSAC will be discussed in Sec. 4.3.2.

Object Coordinate Regression. Similar to our experiments in Chapter 3, we use the depth information of training images to generate object coordinate ground truth. Alternatively, this

ground truth can also be rendered using the available 3D models. We train the Coordinate CNN using the following surrogate loss:

$$\ell_{\text{coord}}(\mathbf{y}, \mathbf{y}^*) = \|\mathbf{y} - \mathbf{y}^*\|, \tag{4.20}$$

where $\mathbf{y}$ is the object coordinate prediction and $\mathbf{y}^*$ is ground truth. We also experimented with other losses including L_2 (squared distance), Huber [Hub64] and Tukey [BT74] which consistently performed worse on the validation set. We train with mini batches of 64 randomly sampled training patches. We use the Adam [KB14] optimizer with a learning rate of 10^{-4}. We cut the learning rate in half after each 50k updates, and train for a total of 300k updates.

Score Regression. We synthetically created data to train the Score CNN in the following way. By adding noise to the ground truth pose of training images, we generated poses above and below the pose error threshold of 5° and 5cm. Using the object coordinate predictions of the trained Coordinate CNN, we compute re-projection error images of these poses. Poses with a large pose error w.r.t. the ground truth pose will lead to large re-projection errors, and we want the Score CNN to predict a small score. Poses close to ground truth will lead to small re-projection errors, and we want the Score CNN to predict a high score. More formally, the pose error $\ell_{\text{pose}}(\mathbf{h}, \mathbf{h}^*)$ of a hypothesis $\mathbf{h}$ should be negatively correlated with the score prediction $s(\mathbf{h}, Y; \mathbf{v})$. Thus, we train the Score CNN to minimize the following loss:

$$\ell_{\text{score}}(s, s^*) = |s - s^*|, \tag{4.21}$$

where: $s^* = -\beta \ell_{\text{pose}}(\mathbf{h}, \mathbf{h}^*)$. Parameter β controls the broadness of the score distribution after applying softmax. We use this distribution for weights in SoftAM (see Eq. 4.5) and to sample a hypothesis in DSAC (see Eq. 4.13). A value of $\beta = 10$ gave reasonable distributions on the validation set, i.e. poses close to ground truth had a high probability to be selected, and poses far away from ground truth had a low probability to be selected. We train the Score CNN with a batch size of 64 re-projection error images of randomly generated poses. We use Adam [KB14] for optimization with a learning rate of 10^{-4}. We train for a total of 2k updates.

Results. We report the accuracy of our pipeline, trained componentwise, in Table 4.1. We present the accuracy per scene and the average over scenes. Since scenes with few test frames like *Stairs* and *Heads* are overrepresented in the average, we additionally show accuracy on the dataset as a whole (denoted *Complete*, i.e. 17,000 test frames). We distinguish between *RANSAC*, i.e. non-differentiable argmax hypothesis selection, *SoftAM*, i.e. differentiable soft argmax hypothesis selection and *DSAC*, i.e. differentiable probabilistic hypothesis selection. As can be seen in Table 4.1, RANSAC, SoftAM and DSAC achieve very similar results when trained componentwise. The probabilistic hypothesis selection of DSAC results in a slightly reduced accuracy of -0.7% on the complete dataset, compared to RANSAC.

4.3.2. END-TO-END TRAINING

In order to facilitate end-to-end learning as described in Sec. 4.2, some parts of the pipeline need to be differentiable which might not be immediately obvious. We already introduced the differentiable loss ℓ_{pose}. Furthermore, we need to derive the model function $\mathbf{H}(Y_J)$ and refinement $\mathbf{R}$ w.r.t. learnable parameters.

Table 4.1.: **Effect of End-to-End Training.** Accuracy measured as the percentage of test images where the pose error is below 5cm and 5°. *Complete* denotes the combined set of frames (17,000) of all scenes. Numbers in green denote improved accuracy after end-to-end training for SoftAM resp. DSAC compared to componentwise training. Similarly, red numbers denote decreased accuracy. **Bold** numbers indicate the best result for each scene.

	Trained Componentwise			Trained End-To-End	
	RANSAC	SoftAM	DSAC	SoftAM	DSAC
Chess	94.9%	94.8%	94.7%	94.2% -0.6%	**95.4%** +0.7%
Fire	75.1%	75.6%	75.3%	**76.9%** +1.3%	75.7% +0.4%
Heads	72.5%	**74.5%**	71.9%	74.0% -0.5%	72.4% +0.5%
Office	70.4%	71.3%	69.2%	56.6% -14.7%	**76.0%** +6.8%
Pumpkin	50.7%	50.6%	50.3%	51.9% +1.3%	**59.7%** +9.4%
Kitchen	47.1%	47.8%	46.2%	46.2% -1.6%	**56.3%** +10.1%
Stairs	6.2%	**6.5%**	5.3%	5.5% -1.0%	6.2% +0.9%
Average	59.5%	60.1%	59.0%	57.9% -2.2%	**63.1%** +4.1%
Complete	61.0%	61.6%	60.3%	57.8% -3.8%	**66.2%** +5.9%

In our application, $\mathbf{H}(Y_J)$ is the PnP algorithm. Off-the-shelf implementations (e.g. [GHTC03, LMNF09]) are fast enough for calculating the derivatives via central differences. Refinement $\mathbf{R}$ involves determining inlier sets and resolving PnP in multiple iterations. This procedure in non-differentiable because of the hard inlier selection procedure. However, because the number of inliers is usually large, refined poses tend to vary smoothly with changes to the input object coordinates. Hence, we treat the refinement procedure as a black box, and calculate derivatives via central differences, as well. For stability, we stop refinement early, in case less than 50 inliers have been found. Because of the large number of inputs and to keep central differences tractable, we subsample the object coordinates for which gradients are calculated (we use 1%), and correct the gradient magnitude accordingly ($\times 100$).

Similar to e.g. [YTLF16] or [KGC15], we found it important to have a good initialization when learning end-to-end. Learning from scratch quickly reached a local minimum. Hence, we initialize the Coordinate CNN and the Score CNN with componentwise training, see Sec. 4.3.1. We optimized training hyperparameters separately for SoftAM and DSAC on the validation set.

For DSAC, we use a fixed learning rate of 10^{-6} for the Coordinate CNN and optimize with Adam. We clamp all gradients to the range of $\pm 10^{-3}$ before passing them to the Coordinate CNN. For the Score CNN, we use a fixed learning rate of 10^{-7} and optimize using stochastic gradient descent with momentum [RHW88] of 0.9. We clamp gradients to the range of $\pm 10^{-1}$ before passing them to the Score CNN.

Choosing learning parameters for the SoftAM strategy was extremely difficult, because training was very unstable. The following setting were working best. We use a fixed learning rate of 10^{-5} for the Coordinate CNN and 10^{-7} for the Score CNN. We optimize both CNNs using stochastic gradient descent with momentum of 0.9, and we clamp all gradients to the range of $\pm 10^{-1}$ before passing them to the CNNs. We train for 5k updates.

Results. See Table 4.1 for results of both strategies. Compared to the initialization (trained componentwise), we observe a significant improvement for DSAC (+5.9% on the complete dataset, standard error of the mean $\pm 0.4\%$). DSAC improves accuracy for all scenes,

with strongest effects for *Pumpkin* (+9.4%) and *Kitchen* (+10.1%). SoftAM significantly decreases accuracy compared to the componentwise initialization (-3.8% on the complete dataset). SoftAM overfits severely on the *Office* scene (-14.7%) and decreases accuracy for most other scenes. Note that these results differ from the results we reported in [BKN+17]. Since the publication of [BKN+17], we improved the learning parameters of DSAC (we state the new parameters above) which increased results from 62.5% to the 66.2% reported here. As mentioned before, learning with SoftAM is unstable and using the updated learning parameters would lead to an accuracy of 33.8% for SoftAM.

Table 4.2.: **Comparison to Previous Work.** Accuracy measured as the percentage of test images where the pose error is below 5cm and 5°. *Complete* denotes the combined set of frames (17,000) of all scenes. **Bold** numbers indicate the best result for each scene. E2E stands for *trained end-to-end*.

	Sparse Features [SGZ+13]	Our	
		Chapter 3	DSAC, E2E
Chess	70.7%	94.9%	**95.4%**
Fire	49.9%	73.5%	**75.7%**
Heads	67.6%	48.1%	**72.4%**
Office	36.6%	53.2%	**76.0%**
Pumpkin	21.3%	54.5%	**59.7%**
Kitchen	29.8%	42.2%	**56.3%**
Strairs	9.2%	**20.1%**	6.2%
Average	40.7%	55.2%	**63.1%**
Complete	38.6%	55.2%	**66.2%**

In Table 4.2, we compare the results of DSAC to the sparse features baseline presented in [SGZ+13] and our results of Chapter 3. Our pipeline, trained with DSAC, surpasses, on most scenes, the accuracy of both competitors substantially. On the complete set, we improve accuracy by 11% compared to the results of Chapter 3. We also measured the median pose error of all frames in the dataset, see Table 4.3. Note that the latest version of PoseNet [KC17] states median translational errors of around 20cm per scene, so it cannot compete in terms of accuracy.

Table 4.3.: **Median Pose Errors.** We show results for the complete 7-Scenes dataset (17,000 frames). Most accurate results are marked **bold**.

	Our, Chapter 3	4.5cm, 2.0°
Ours, Trained Componentwise	RANSAC	4.0cm, 1.6°
	SoftAM	3.9cm, 1.6°
	DSAC	4.0cm, 1.6°
Ours, Trained End-To-End	SoftAM	4.0cm, 1.6°
	DSAC	**3.5cm, 1.6°**

4.3.3. INSIGHTS AND DETAILED STUDIES

Ablation Study. We study the effect of learning the Score CNN and the Coordinate CNN in an end-to-end fashion, individually. We use componentwise training as initialization for both

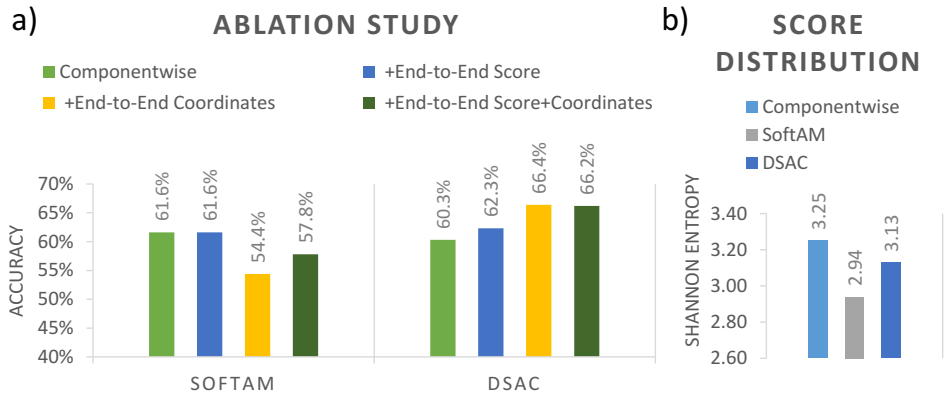

Figure 4.3.: **Ablation Study and Score Distribution Entropy. (a)** Effect of end-to-end learning on pose accuracy w.r.t. the two CNNs in our pipeline. The Coordinate CNN has the main effect on the overall accuracy. **(b)** Effect of end-to-end training on the entropy of the score distribution on the complete dataset. DSAC allows for broader distributions than SoftAM. Set the text for details.

CNNs. See Fig. 4.3 a) for results on the complete dataset. For DSAC, training the Score CNN in an end-to-end fashion increases accuracy slightly, but training the Coordinate CNN is essential for best results. Although training only the Coordinate CNN in an end-to-end fashion increases accuracy by $+0.2\%$ compared to training both CNNs, the effect is not statistically significant. We conclude that the influence of the Score CNN is limited in this pipeline, and the score initialization seems to be sufficient for good results. Similarly, for SoftAM, we see that the bad accuracy is not due to overfitting of the Score CNN, but the instability of learning the Coordinate CNN.

Analysis of Object Coordinate Predictions. In the componentwise training, the Coordinate CNN learned to minimize the surrogate loss ℓ_{coord}, i.e. the distance $\|\mathbf{y}_i - \mathbf{y}_i^*\|$ of object coordinate predictions $\mathbf{y}_i$ w.r.t. ground truth $\mathbf{y}_i^*$. In Fig. 4.4, we visualize how the prediction of the Coordinate CNN changes when trained in an end-to-end fashion, i.e. to minimize the loss ℓ_{pose}. Both end-to-end learning strategies, SoftAM and DSAC, increase the accuracy of object coordinate predictions in some areas of the scene at the cost of decreasing the accuracy in other areas. We observe very extreme changes for the SoftAM strategy, i.e. the increase and decrease in object coordinate accuracy is large in magnitude, and improvements are focused to small scene areas. The DSAC strategy leads to a much more cautious tradeoff, i.e. changes are smaller and widespread. We conclude that SoftAM tends to overfit due to overly aggressive changes in object coordinate predictions.

Score Distribution Entropy. See Fig. 4.3 b) for an analysis of the effect of end-to-end learning on the average entropy of the softmax score distribution (see Eq. 4.5). We observe a clear reduction in entropy for the SoftAM strategy. The larger the pose error of a hypothesis is, the larger is also its influence on the pose average (see Eq. 4.4). SoftAM has to weigh down such poses aggressively for a good average. DSAC can allow for a broader distribution

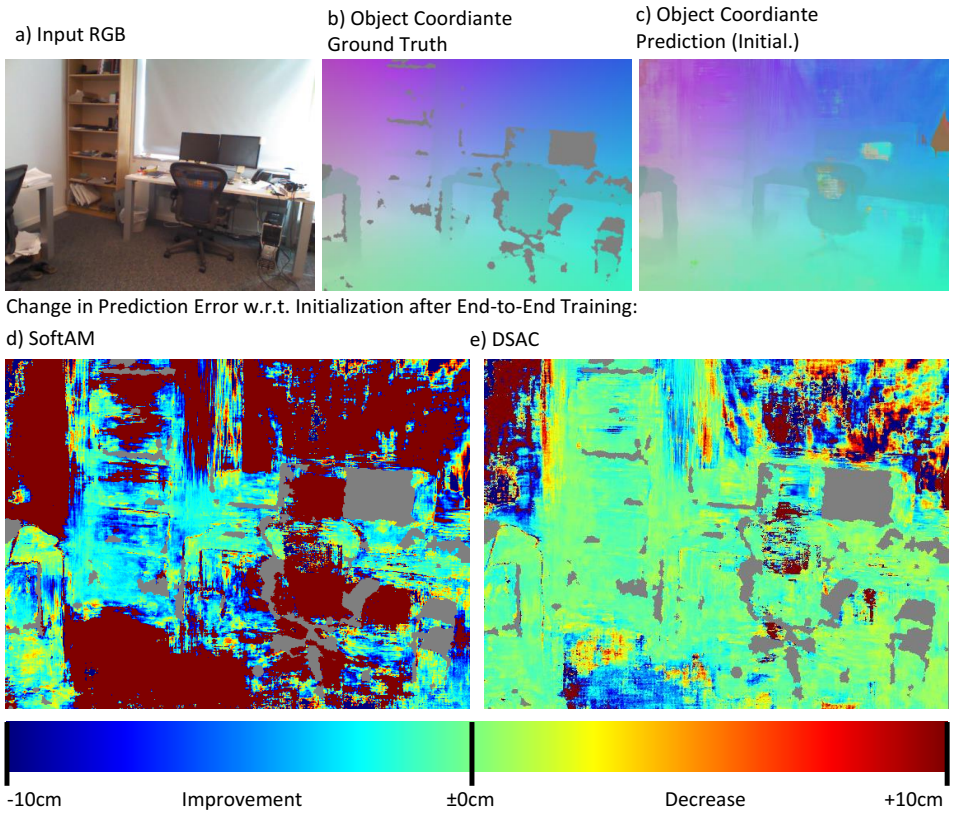

Figure 4.4.: **Prediction Quality.** We analyze object coordinate prediction quality on an *Office* test image **(a)** with ground truth object coordinates **(b)** (X/Y/Z mapped to RGB). The prediction after componentwise training can be seen in **(c)**. We vizualize the relative change of prediction error w.r.t. componentwise training in **(d)** for SoftAM, resp. in **(e)** for DSAC. We observe an aggressive strategy of SoftAM which focuses large improvements on small areas (14% of predictions improve). DSAC shows small improvements but on large areas (38% of predictions improve). Note that DSAC achieves superior pose accuracy on this scene.

because poses which are unlikely to be chosen do not affect the loss of poses which are likely to be chosen. This is an additional factor in the stability of DSAC.

Restoring the argmax Selection. After end-to-end training, one may restore the original RANSAC algorithm, e.g. selecting hypotheses w.r.t. scores via argmax. In this case, the average accuracies stay relatively stable at 65.6% for DSAC resp. 57.2% for SoftAM.

Test Time. The object coordinate prediction takes $\sim$0.5s on a Tesla K80 GPU. Pose optimization takes $\sim$1s. The runtime of argmax hypothesis selection (RANSAC) or probabilistic

selection (DSAC) is identical and negligible.

Difficulty of the 7-Scenes Dataset. Compared to the object coordinate predictions of the random forest in Chapter 3 the predictions of the Coordinate CNN are very smooth, see Fig. 4.4. This raises the question whether robust pose fitting is still necessary for these outputs. Although the CNN predictions look smooth, inlier ratios range from 5% to 85%. See Fig. 4.5, left for the inlier ratio distribution over the complete 7-Scenes dataset. In accordance to [SGZ+13], we consider an object coordinate prediction an inlier if it is within 10cm of the ground truth object coordinate. In Fig. 4.5, right, we plot the performance of DSAC against the ratio of inliers. For comparison we plot the performance of a naive approach without RANSAC where we fit a pose to all object coordinate predictions. Please see Fig. 4.6 for examples of difficult situations in the 7-Scenes dataset which our DSAC pipeline is able to solve.

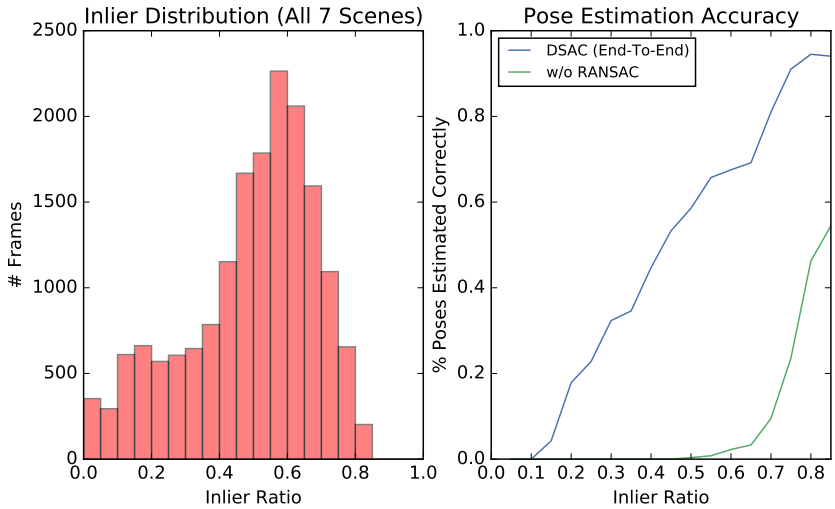

Figure 4.5.: **Importance of RANSAC. Left:** Distribution of inlier ratios of our object coordinate predictions. **Right:** Corresponding pose estimation accuracy of DSAC compared to a naive approach without RANSAC.

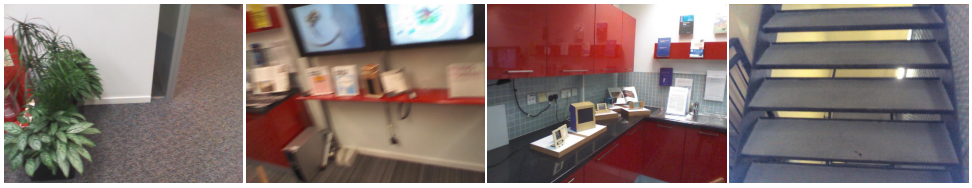

Figure 4.6.: **Difficult Frames within the 7-Scenes Dataset.** From left to right: Texture-less surfaces, motion blur, reflections, and repeating structures. DSAC estimates the correct pose in all 4 cases.

Multi-Modality. Compared to the system presented in Chapter 3, the DSAC pipeline performs not as well on the *Stairs* scene (see Table 4.2). We account this to the fact that the Coordinate CNN predicts only uni-modal point estimates whereas the random forest of Chapter 3 predicts multi-modal object coordinate distributions. The *Stairs* scene contains many repeating structures, so we expect multi-modal predictions to help. We also expect bad performance of the SoftAM strategy in case pose hypothesis distributions are multi-modal because an average is likely to be a bad representation of either mode. In contrast, DSAC can probabilistically select the correct mode. We conclude that multi-modality in object coordinate predictions and pose hypothesis distributions is a promising direction for future work.

4.4. DISCUSSION

We presented two strategies for differentiating the RANSAC algorithm: Using a soft argmax operator, and probabilistic selection. By experimental evaluation we conclude that probabilistic selection is superior and call this approach DSAC. We demonstrated the use of DSAC for learning a variant of our pose estimation pipeline for camera localization in an end-to-end fashion. However, DSAC can be deployed in any deep learning pipeline where robust optimization is beneficial, for example learning structure from motion or SLAM end-to-end.

5. DISCUSSION

Contents

In this thesis, we presented a detection and pose estimation system for object instances which combines machine learning techniques with principles from traditional computer vision, like geometry. This system supports a wide range of object types and can process RGB or RGB-D inputs. We addressed many challenges in this thesis that typically restrict the practical applicability of object pose estimation in realistic scenarios. While we made progress in many aspects, there are still many limitations and open questions for future research. In the following, we discuss both, advantages and limitations, of our proposed solution. Me mark the main keywords **bold** in each limitations section.

5.1. ACCURACY AND VERSATILITY

Our pose estimation pipeline can reliably detect the presence and estimate the poses of object instances from a single input image under varying lighting conditions and severe occlusion. It can handle textured and texture-less objects by discovering reliable features autonomously during training. Several works extended our pipeline. For example, Michel et al. used object coordinates to estimate poses of articulated instances. Our system has been applied to detect small items (see Chapters 2 and 3), furniture [MKB+15], aircrafts [MMDM+16] and whole scenes (see Chapter 3 and 4). All object types can be located with a few centimeters and degrees precision. Visually, the poses estimated and ground truth can hardly be distinguished making it precise enough for augmented reality, even with RGB input. At the time of publication, we set a new state-of-the-art in terms of accuracy on multiple, diverse datasets.

Limitations. While our approach supports a wide variety of object types there are still many objects which are particularly challenging. Specifically, for object composed of **reflective and transparent materials** neither RGB nor RGB-D image cues are reliable. Regarding RGB, the objects may appear highly textured with many distinct features. But those features are not part of the object but of the environment which is reflected or shining through. This environment is usually unknown and only seen partially in the remaining image. This makes reasoning about which cues belong to the object and which to the environment difficult. Regarding depth, most sensors will not provide reliable cues for reflective or transparent materials either. They either provide no measurements or wrong measurements: this applies to structured light, time-of-flight and stereo.

Also, objects of **thin shape** are particularly difficult. They usually do not exhibit expressive texture and most depth sensors have problems to provide reliable measurements. These objects appear in many applications and environments, e.g. in the form of poles or pencils. Tools like cutlery or surgical tools are thin-shaped, and are additionally composed of reflective material making them extra challenging.

As mentioned before, our pipeline has been extended to handle articulated objects instead of rigid objects. Such objects are composed of multiple rigid parts connected by joints, e.g. a cupboard with a moving door. However, other objects, like plush toys or clothing, are freely deformable without such clear constraints. For such objects, the prediction of object coordinates based on image patches is still sensible. The **deformation** is likely to change the global shape of an object but less so the local appearance. However, the subsequent RANSAC-based pose optimization procedure is problematic. Even if a parametric model for the deformable pose can be formulated it is likely to have many free parameters. Every additional free parameter in the pose hypothesis space increases the minimal number of correspondences needed to sample a pose. Hence, it becomes increasingly unlikely that a sampled correspondence set is outlier-free. The number of hypothesis samples needed to counteract this effect grows exponentially. Therefore, to support freely deformable objects, more intelligence in hypothesis sampling is needed or an alternative pose optimization procedure altogether as seen, for example, in [TSSF12] for 60D poses.

The applicability of the object coordinate concept to deformable objects has been demonstrated in the tracking extension of our pipeline [KMB+14], see Fig. 5.1, left. The pose of the cat can be estimated because the object coordinates are correctly predicted for parts not affected by the deformation. The pose optimization does work in this example because of the robustness of tracking. Note, that this system treats the cat as a rigid object, i.e. deformation parameters are not part of the pose space. Tracking restricts the search for a good hypothesis to a small neighborhood in the 6D pose space, based on previous observations. Although the deformed cat does not fit the rigid model very well, it is still the local optimum in this small neighborhood.

We restricted our system to pose estimation of object instances. However, concepts similar to object coordinates have been applied to **object classes** [HRW07, TSSF12]. Specifically, landmarks, which can be thought of as sparse object coordinates, are a recurring concept in class-based pose estimation, see e.g. [TS14, TSLP14]. Object coordinates could be re-thought as a geodesic interpolation of class landmarks, and an object coordinate predictor, e.g. a CNN, could learn to be robust to intra-class variabilities.

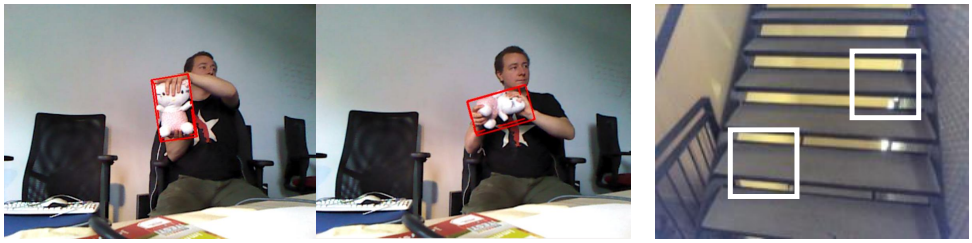

Figure 5.1.: **Deformable Objects and Ambiguities. Left:** In an extension to the system of Chapter 2, Krull et al. [KMB+14] have shown that object coordinate regression can be very robust w.r.t. to object deformation when combined with tracking. Note that deformation parameters are not estimated in their system. **Right:** Although the steps in the image are identical, their appearance differs, see the patches marked. Therefore, learning ambiguities form limited data can be difficult.

5.2. SCALABILITY: OBJECT POSE ESTIMATION

The implementation of our approach takes approximately 100ms to estimate the pose of one object on a single frame, and is has been extended to real time tracking [KMB+14], supporting approximately 20 frames per second on a laptop. Hence, the system is fast enough for practical application such as grasping specific objects by a robot, and even augmented reality when tracking can be applied. In Chapter 2, we discussed how to train a joint classification-regression forest that predicts soft segmentations and object coordinates simultaneously for multiple objects. The runtime of the forest scales logarithmically in the number of objects. In Chapter 3, we discussed a hypothesis sampling schema for RANSAC which decides on the fly, based on predicted object label distributions, for which object a hypothesis should be created. We showed empirically that this lets us perform the pose optimization stage sublinearily in the number of objects. The whole system is able to process 50 objects in approximately 1s with a CPU only implementation.

Limitations. If **thousands or millions of objects** should be supported, a logarithmic complexity might be insufficient. A combination of object coordinate regression with hashing [KTN+16, WL15] could yield better run times. Also, pose optimization can clearly not be performed for all objects in the database. An hierarchical approach which identifies objects first and only optimized poses of objects of high probability could be necessary.

Another aspect, not discussed in this thesis, is **multi-instance pose estimation**, i.e. the case where an object can appear in an image more than once. We achieved some preliminary results by applying mean-shift to identify clusters in the pose hypothesis pool, where subsequently, pose optimization is run per cluster. However, only shortly before the time of publication of this thesis, multi-instance pose estimation datasets were published [HMS+] which would facilitate a quantitative evaluation.

5.3. SCALABILITY: CAMERA LOCALIZATION

In Chapter 4, we discuss a variant of our pipeline for camera localization which used a CNN for object coordinate prediction. In our experiments, we can show that this CNN can handle rooms with a size of a few square meters. In a preliminary experiment, we apply our system to the dataset of Kendall et al. [KGC15] which contains large outdoor scenes extending over hundreds of meters, see Fig. 5.2. We kept the capacity, and hence the run time, of our Coordinate CNN constant in all experiments, therefore it scales constant in the size of the scene.

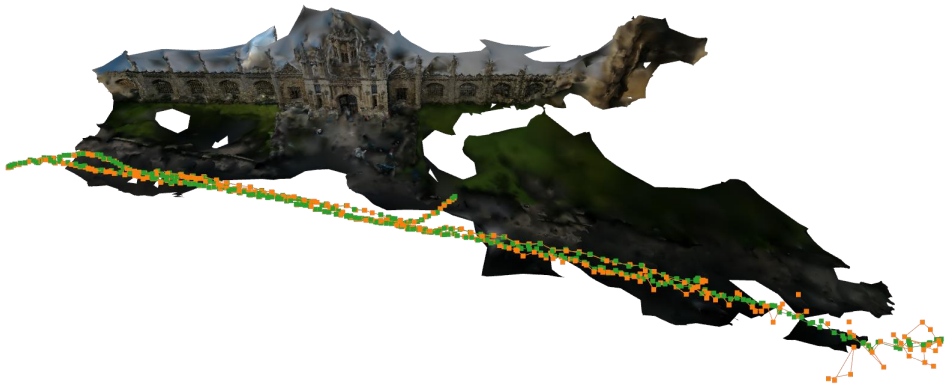

Figure 5.2.: **Outdoor Camera Localization.** We trained a variant of the DSAC system of Chapter 4 for the large scale outdoor camera localization dataset of [KGC15]. We show an exemplary result for the Kings Cross scene, with the ground truth camera path drawn green, and our estimated path drawn orange. Our median accuracy on this scene is 22cm and 0.4°. A state-of-the-art sparse feature based approach [SLK16] has a median accuracy of 42cm and 0.5° on this scene.

Limitations. It is evident that with growing scene size and constant model capacity accuracy of the object coordinate predictions decrease. Unfortunately, inaccurate object coordinates will at some point make pose optimization impossible, because the inherent geometric constraints will be violated. In this case, the system fails to produce even an approximate pose. We have seen in preliminary experiments that this is the case at **district-sized scenes**, specifically the streets scene in the dataset of Kendall et al. [KGC15] which spans over 10,000m^2. Increasing the model capacity has proven ineffective, either because the capacity needed scales exponentially or other aspects play a role. For example, with very large scenes, and specifically with repeating architectural structures, image patches become more and more ambiguous. Object coordinate point estimates, as produced by the CNN of Chapter 4, cannot capture this **ambiguity**. As show in Chapter 3, modeling object coordinate distributions is a sensible strategy and CNNs have been shown capable of making such predictions [GPAM$^+$14, LSZ15, Bis94]. In preliminary experiments, we trained a CNN to model object coordinate distributions. However, this failed to improve results even on highly ambiguous scenes like the Stairs sequence of the 7-Scenes dataset, see Fig. 5.1, right.

Possibly, the amount of training data is insufficient for the CNN to recognize and model ambiguities. A strong regularization at the feature level could help in this case.

An important application of camera localization is to recover from tracking failure in a SLAM system. But to be applied in this scenario, camera localization cannot be pre-trained on the scene since it is discovered on the fly. Cavallari et al. [CGL+17] show how to adapt an object coordinate regression forest online for a unknown scene. However, their systems relies on a depth channel to calculate estimates of object coordinates for incoming frames. Furthermore, the adaptation of the random forest is restricted to updating leaf distributions. A system, like the one presented in Chapter 4, could adapt to a scene with **incremental end-to-end learning**. At its current state however, training is too slow, and end-to-end learning works only with a good initialization. Furthermore, an online learning training schedule has to be developed, e.g. by applying curriculum learning [BLCW09].

5.4. OBJECT POSE ESTIMATION FROM RGB ONLY

There was a renewed interest in object pose estimation after the introduction of the Kinect depth sensor. It enabled detection of objects without sufficient texture cues where previous, feature-based approaches failed. We were among the first who aimed to transfer the success of object pose estimation from RGB-D images back to the RGB only case [BMK+16]. We showed that machine learning approaches are adaptive enough to deal with the few available cues of texture-less objects. We achieved accurate and robust pose estimation results without the need for a depth channel. Since most image data is RGB only, this has been an important achievement.

Limitations. The experiments in Chapter 3 were limited to cases with no or little amount of occlusion. We conducted some experiments on our occlusion dataset (Appendix A.2.2), and while detection of the objects works fairly well, pose estimates are very inaccurate. Preliminary experiments with a CNN architecture instead of a random forest, show some promise. However, to the best of our knowledge, convincing pose estimation from an RGB image and under **severe occlusion** has not yet been demonstrated. The system presented in this thesis has been extended to handle severe occlusion in [KBM+15] and [MKB+17] but both approaches rely on depth information.

5.5. EXPLOITING PREDICTION UNCERTAINTY

In Chapter 3, we showed the benefit of modeling and exploiting uncertainty of object coordinate predictions. We modeled uncertainty by learning multi-modal object coordinate distributions rather than point estimates. This approach can account for varying difficulty in the data. For example, predicting precise object coordinates for highly textured areas is easy and the corresponding distribution will be narrow. For texture-less, uniformly colored areas, the predicted distribution will be broad. Later stages of the pipeline can then trust some predictions more than others. We showed that object coordinate distributions can be exploited even if no depth channel is available at test time.

Limitations. For some symmetric objects, like texture-less bowls, *all* object coordinates are extremely ambiguous. Modeling such extremely broad distributions is possible, although difficult in practice with parametric models like Gaussians. The main problem, however,

is the **sampling of pose hypotheses**. To generate a hypothesis in this case, a minimal set of object coordinates has to be sampled from the predicted distributions. Because the uncertainty is very high, the probability to sample a consistent set is small, at least if the object coordinate samples are drawn independently.

We modeled uncertainty using the distribution of training samples in random forest leaf nodes. In Chapter 4, we used a CNN to predict object coordinates but only point estimates. However, CNNs have been shown to model arbitrarily complex distributions [GPAM+14, LSZ15, Bis94] which we expect to help pose estimation.

5.6. LEARNING OBJECT POSE ESTIMATION

We showed the benefit of carefully choosing aspects of the task at hand which should be learned, and which aspects should be hard coded based on prior knowledge. We chose to learn to predict correspondences from an input image but used principles from geometry to find the final pose estimate. This separation has proven superior to approaches which dismiss all prior knowledge and cast pose estimation as an unconstrained regression problem, like PoseNet [KGC15]. The use of prior knowledge reduces the need for training data which is often limited.

Limitations. In many applications, collection training data with accurate annotations of 6D ground truth poses is connected with substantial human effort and, therefore, a problem. On the other hand, scans of objects are easier to obtain resp. CAD models of objects are available, e.g. from production cycles. Producing synthetic training data based on such 3D models is highly desirable because a rendering engine can produce arbitrary amounts of data and ground truth annotations are highly accurate. However, often there is a shift in appearance between real data of an object and synthetic data, e.g. because the scanning device cannot capture all material properties. Therefore, **learning from synthetic data** is often problematic, especially if features are also learned. The model will adapt to the specificities of the synthetic data including the inaccuracies of the scanning and rendering pipelines. Such models usually perform significantly worse on real data because of lacking generalization capabilities. In Chapter 2, we have shown how to train a random forest from synthetic data and achieve very accurate results on real data. This was possible because the features were not learned from scratch but restricted to combinations of pixel difference features. Furthermore, the random forest could rely on a depth channel where many inaccuracies of synthetic data are less severe. To achieve good performance, we had to apply noise to the responses of RGB features to make them less reliable, and hence the forest chose depth features more often. We expect learning a CNN using similar data to be highly problematic as hierarchical convolution features are much more adaptive, and generalization control by adding noise to responses is more involved. There is, however, a vast literature concerning domain adaptation, see e.g. [BDBC+10], a field which is specifically concerned with bridging the gap between data drawn from different distributions, like synthetic and real data. In recent years, a new generation of generative models, generative adversarial networks (GANs), have been introduced which have been shown to produce increasingly realistic natural images based on random noise samples [GPAM+14, NYB+17]. GANs can be conditioned on existing input data, and change complex image aspects [IZZE17] like time of day and season. GANs have also been successfully applied to domain adaptation tasks [GUA+16, SPT+17]. Therefore, a conditional GAN could potentially adapt the image statistics of rendered training data for object pose estimation making it more natural.

If training data generation using a renderer is an option, one can design an arbitrary training data distribution and generate an infinite number of samples for learning. If no prior knowledge about the test data distribution exists, a uniform prior can be deployed for the training distribution, e.g. by sampling ground truth poses from SE(3) and render the corresponding images. However, it is also possible to **guide the training data sampling** throughout the learning process. A differentiable renderer [LB14], facilitates passing gradients of the training loss through the rendering pipeline directly to the distribution of training data parameters. These parameters control the object pose but also additional aspects like lighting, occlusion and reflections. Changing these parameters in the direction of the training loss gradients will produce more difficult training examples which could potentially improve robustness on the test set.

5.7. END-TO-END LEARNING WITH DSAC

We introduced a differentiable formulation of the RANSAC algortihm which enables us to learn object coordinate regression end-to-end using the full geometric pose estimation pipeline. We therefore bridged a gap between traditional computer vision based in geometry and modern machine learning-based computer vision. However, DSAC is not limited to geometric pipelines. It is a generic, robust optimization method which implements the general strategy of hypothesize and verify. Therefore, DSAC can be used in any learning system where robust selection of model hypotheses is beneficial.

Limitations. In this thesis, end-to-end learning was restricted to a camera localization variant of the pipeline introduced in Chapter 4. Specifically, this pipeline omits the object segmentation step because a scene object does per definition occupy the whole image. However, an adaption of the camera localization pipeline for **object pose estimation** is possible if an additional CNN predicts a soft segmentation similar to the random forest in Chapters 2 and 3. The pose hypothesis pool would than be sampled according to this soft segmentation. Learning this end-to-end is also possible, since the hypotheses sampling could be modeled similarly to the hypothesis selection in DSAC. This would result in an optimization over two expectations, one over all possible sampled hypothesis pools and one over the selected hypothesis. While the prediction of a soft segmentation is vital for object pose estimation, camera localization could also benefit. The pipeline could learn which parts of a scene to avoid in hypothesis sampling, like untextured areas or unstable, moving parts like cars and pedestrians.

We trained the DSAC-based camera localization pipeline of Chapter 4 in an end-to-end fashion but not from scratch. A good **initialization** was needed or the system would quickly reach a local-minimum in end-to-end training. For initialization, we trained the system using ground truth object coordinates generated from depth images. Object coordinate ground truth can also be obtained by rendered object coordinates using a 3D model of the scene. However, in some scenarios both depth and 3D scene models are difficult or impossible to obtain. It would be desirable to initialize the system without ground truth object coordinates. In preliminary experiments, we have seen that initialization is possible with object coordinates calculated from a constant depth prior. However, even after end-to-end training, the pipeline does not reach the same accuracy as when initialized with ground truth object coordinates. At the moment its unclear whether this is solely an optimization problem or whether a different initialization is necessary.

6. CONCLUSION

Pose estimation is an important problem in computer vision with many application areas. As a research topic, it is challenging because of a wide range of difficult object types and varying image conditions. This makes it a natural test bed of machine learning techniques. While the amount of training data is usually very limited, we can utilize a vast repertoire of prior knowledge about the task, e.g. camera models and geometry. Pose estimation is therefore an interesting meeting point of machine learning and traditional computer vision. In this thesis, we introduced and approached many of the challenges of object pose estimation. Our system is versatile, robust, scalable and fast enough for practical applications. Some of our contributions, like DSAC, are applicable to a wide range of other learning problems. In the previous chapter, we discussed many exciting, open research questions in the area of object pose estimation. Specifically, we expect that many approaches and algorithms of traditional computer vision can benefit from machine learning and vise versa to yield exiting new systems for many practical problems.

A. APPENDIX

Contents

A.1. SYMBOLS AND ABBREVIATIONS

Symbol	Description
I	Image (RGB or RGB-D).
$c \in \mathcal{C}$	Object index c and set of objects $\mathcal{C}$.
i	Pixel index.
$\mathbf{p}_i$	2D Position of pixel i.
$\mathbf{x}_i^{\text{rgb}}$	RGB color at pixel i.
d_i	Depth measurement at pixel i.
$\mathbf{e}_i$	3D camera coordinate at pixel i.
$\mathbf{y}(I, i)$ or $\mathbf{y}_i$	3D object coordinate prediction of image I for pixel i. It encodes a 2D-3D correspondence between the image and an object.
$\hat{\mathbf{y}}$	Quantized object coordinate resp. proxy class.
$Y(I)$ or Y	Set of all object coordinate predictions for an image I.
$T \in \mathcal{T}$	Decision tree T of random forest $\mathcal{T}$.
l_i^j	Leaf index of tree T_j predicted for pixel i.
$\boldsymbol{\zeta}$	Random forest feature parameters.
$\mathbf{w}$, $\mathbf{v}$	Neural network weights.
J	Minimal set of pixel indices to create a pose hypothesis, i.e. $J = \{j_1, ..., j_n\}$. For RGB-D inputs $n = 4$, for RGB inputs $n = 3$.
Y_J	Minimal set of object coordinate predictions to create a pose hypothesis, i.e. $Y_J = \{\mathbf{y}_{j_1}, ..., \mathbf{y}_{j_n}\}$
$\mathbf{H}(Y_J)$	Function that creates a hypothesis from a minimal set, i.e. perspective-n-point (PnP) algorithms [GHTC03] for RGB inputs or the Kabsch algorithm [Kab76] for RGB-D inputs.
$\mathbf{h} = (\boldsymbol{\theta}, \mathbf{t})$	Pose hypothesis consisting of 3D rotation $\boldsymbol{\theta}$ and 3D translation $\mathbf{t}$, i.e. 6D vector.
C	Camera calibration matrix.
$\mathbf{R}(\mathbf{h}, Y)$	Refinement function of hypothesis $\mathbf{h}$ with access to all object coordinates Y.
$s(\mathbf{h})$	Scoring function to assess the quality of hypothesis $\mathbf{h}$
$\ell(.)$	Task loss function.
$\tilde{X}$	Optimal X w.r.t. some optimization problem.
X^*	Ground truth X.

Symbols not listed here are defined and used locally in the respective paragraph.

Abbreviation	Description
RGB	Digital image with a red, green and blue color channel.
RGB-D	Color image with an additional depth channel.
RANSAC	Random Sample Consensus
DSAC	Differentiable Sample Consensus
CNN	Convolutional Neural Network
ICP	Iterative Closest Point
PnP	Perspective-n-Point
SGD	Stochastic Gradient Descent
IoU	Intersection over Union

A.2. DATASETS

Several object instance detection datasets have been published in the past [RCT13, DBCMC12], many of which deal with 2D poses only. Lai et al. [LBRF11] published a large RGB-D dataset of 300 objects that provides ground truth poses in the form of approximate rotation angles. Unfortunately, such annotations are to coarse for the accurate pose estimation task we try to solve. A dataset with accurate 6D pose annotations was published by Hinterstoisser et al. [HLI$^+$12] and is widely used as a benchmark. We describe this dataset in Sec. A.2.1. Despite its popularity, the Hinterstoisser dataset has a few limitations which motivated us to publish two additional datasets. Firstly, the object of interest in the Hinterstoisser dataset is never substantially occluded. We therefore published a dataset of additional annotations of occluded objects in the Hinterstoisser dataset. We describe this occlusion dataset in Sec. A.2.2. Secondly, the Hinterstoisser dataset images are all recorded under static lighting conditions. In Sec. A.2.3, we describe our lighting dataset which features training and test data captured with three different lighting configurations. Finally, we describe a pose estimation dataset specific to the task of camera localization, published by Shotton et al. [SGZ$^+$13], in Sec. A.2.4. Note that all of these datasets are RGB-D datasets. Nevertheless, they are widely used for pose estimation from RGB. In this case, depth channels are ignored.

A.2.1. DATASET OF HINTERSTOISSER ET AL. [HLI$^+$12]

This dataset consists of 15 RGB-D image sequences corresponding to 15 different objects of interest. The objects feature little amount of texture and are approximately 15cm to 30cm in diameter. Each sequence consists of approximately 1,200 frames sampling the upper view hemisphere of the corresponding object including $\pm 45^\circ$ in-plane rotation. In Table A.1, we state the extent and number of test frames per object. For each frame, the rotation and translation of the object is given relative to the camera. The distance between the object of interest and the camera ranges between 65cm and 115cm. Ground truth poses were annotated using a marker board which is also visible in the images. The object of interest is usually displayed in the center of the frame surrounded by dense clutter on a work desk. The clutter consists predominately of objects from the respective other image sequences but their pose is not annotated. The specific clutter configuration changes throughout each sequence, probably to avoid severe occlusion. Therefore, the object of interest is never occluded substantially. Apart from that, the imaging conditions among all sequences are static, i.e. the camera type, lighting conditions and the global desk setup are static. The camera calibration parameters are also known with a focal length of 573px and a principal point at (325, 242).

In the original evaluation procedure, proposed by Hinterstoisser et al. in [HLI$^+$12], all natural image are test images. For each test frame, the ID of the object of interest is given, and only the pose has to be estimated. Performance is measured as the percentage of test frames per sequence where the pose has been estimated correctly, subject to some threshold. This threshold is defined via the average distance of transformed vertices of the object's 3D model, see Sec. A.3.1 for details. There are no designated training or validation images. Instead, a 3D model of each object is provided which can be used to render training or validation data synthetically, see Fig. A.1. The 3D models were reconstructed from real images using a marker board for registration, and feature coarse color information. For two objects, namely for the cup and the bowl, no 3D models are provided. Hence, we do not use them in our experiments.

Figure A.1.: **Overview of the Dataset of Hinterstoisser et al. [HLI+12].** From left to right: Two renderings of the duck's 3D model used for training, a real test frame of the duck object, and an overview of all objects of the dataset used in our experiments with the corresponding name. Note that 3D models for two out of 15 objects are not provided, namely for the bowl and the cup. Hence, they are not shown and not used in our experiments.

Table A.1.: **Statistics of the Hinterstoisser et al. Dataset.** We state the approximate extent of each object, and the number of test frames available for each object. Note that training data has to be generated using 3D models provided in the dataset.

	Approximate Size (width × height × depth)	# Test Frames
Ape	8cm × 9cm × 8cm	1,235
Bench Vise	12cm × 22cm × 22cm	1,214
Camera	14cm × 10cm × 14cm	1,200
Can	18cm × 19cm × 10cm	1,195
Cat	13cm × 12cm × 7cm	1,178
Driller	8cm × 21cm × 23cm	1,187
Duck	8cm × 9cm × 10cm	1,253
Egg Box	11cm × 7cm × 15cm	1,252
Glue	8cm × 17cm × 4cm	1,219
Hole Puncher	13cm × 10cm × 11cm	1,236
Iron	12cm × 14cm × 26cm	1,151
Lamp	12cm × 21cm × 20cm	1,226
Phone	15cm × 18cm × 9cm	1,242
Bowl	17cm × 8cm × 17cm	1,233
Cup	9cm × 10cm × 12cm	1,240

A.2.2. OCCLUSION DATASET (OUR)

The dataset of Hinterstoisser et al. [HLI+12] is free of occlusions. While the objects annotated in the dataset of [HLI+12] are embedded in dense clutter they are still fully visible in each frame. Hence, to demonstrate robustness against occlusion we created a new dataset. We annotated one sequence ("Bench Vise", ca. 1,200 frames) of the dataset of [HLI+12] with 6D poses of 8 additional objects present in the scene, i.e. we provide approximately 10,000 additional annotations for this dataset. In Table A.2, we state the number of additional annotations per object as well as the number of frames with a certain degree of occlusion. Depending on the viewing direction, these 8 objects occlude each other to a large extent making this dataset very challenging. Fig. A.2 shows one frame with all annotations marked, and a closeup of a heavily occluded object.

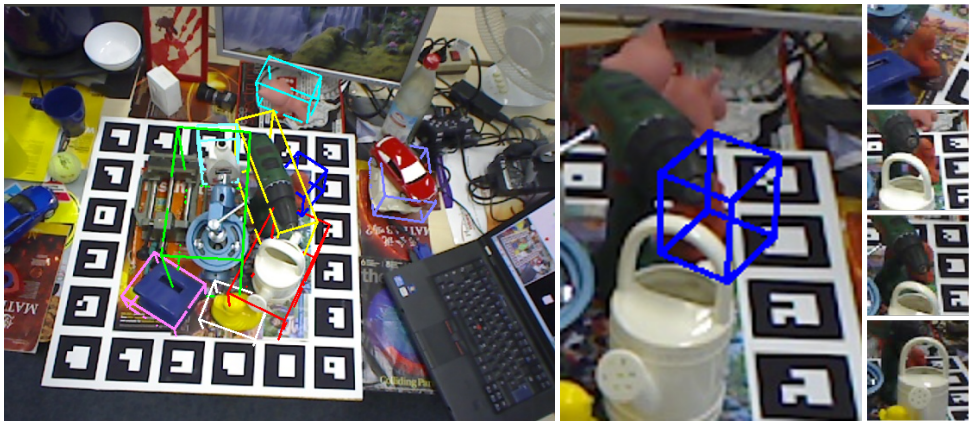

Figure A.2.: **Our Occlusion Dataset. Left:** One frame of the Bench Vise sequence of the dataset of [HLI+12] annotated with poses of 8 additional objects, each marked with a bounding box where the color encodes the object ID. The original dataset contains only the pose of one object, marked by the green bounding box. Note that some objects are occluded. **Middle:** Our annotations also include heavily occluded objects. Shown here is the Ape figure marked by a blue bounding box. **Right:** The Ape figure with different levels of occlusion, from top to bottom: 20%, 40%, 60%, 80%. See also Table A.2 for details on the calculation of occlusion values.

We annotated the sequence by initializing the pose of each object by hand, and propagating the object pose via the ground truth transformation of each frame. If the propagation produced errors or when the object was moved within the scene we re-initialized by hand. For each frame, all poses were refined using ICP. We made the annotation data publicly available, and the dataset has since been widely used, e.g. in [KBM+15, MKB+17, HLRK16, RL17]. It has also been added to the SIXD object instance pose estimation challenge [HMS+].

A.2.3. LIGHTING DATASET (OUR)

We recorded an RGB-D dataset of 20 textured and texture-less objects ranging from approximately 20cm to 55cm, see Table A.3. Fig. A.5 shows every object of our dataset. For each

Table A.2.: **Statistics of the Occlusion Dataset.** We state the total number of test frames per object as well as the number of frames where the projected object area is occluded to a certain degree. We render the object using the ground truth pose and compare the Kinect depth channel with the rendered depth at each pixel. We count the percentage of pixels where the rendered depth value is at least 1cm larger.

	# Frames			
	Total	Occlusion 0-30%	Occlusion 30-70%	Occlusion 70-100%
Ape	1,170	974	115	81
Bench Vise	1,214	1,183	31	0
Can	1,207	869	313	25
Cat	1,187	635	332	220
Driller	1,214	746	455	13
Duck	1,143	859	164	120
Egg Box	1,175	899	219	57
Glue	901	543	268	90
Hole Puncher	1,210	1,069	140	1
Sum	10,421	7,777	2,037	607

object, we obtain three sequences which differ in the lighting condition it was captured with. We record under bright artificial light, darker natural light and an artificial spot light. While the first two lighting conditions are diffuse, the last is strongly directional causing hard shadows on edges of an object. See Fig. A.4 for the different lighting conditions for five of our objects.

We recorded approximately 550 images per object (i.e. approximately 11,000 images in total), sampling the upper view hemisphere without in-plane rotation. See Table A.3 for the exact number of frames per lighting condition and object. Each frame of the dataset shows exactly one object on a marker board, which we used for registration, in front of a neutral background. The RGB images contain a substantial amount of noise due to the registration of depth and RGB channels with the Kinect SDK [WA]. We provide a texture-less 3D model of each object and the ground truth 6D pose of each frame. We use a focal length of 580px and a principal point at the image center. Fig. A.3 displays the scan and preprocessing pipeline. The objects are scanned with a commercially available Kinect camera and are segmented in each RGB-D image afterwards. This procedure is done three times with varying lighting conditions.

We propose the following evaluation procedure for this dataset. Images of two lighting conditions may be used as training data, namely *bright* and *dark* sequences. The respective *spotlight* sequence should be used for testing, i.e. a pose estimation system has to generalize from diffuse lighting to strong directional lighting. To increase the amount of training resp. test data, in-plane rotation of $\pm 45^\circ$ in 7 steps should be added to each frame. The ID of the object of interest is known for each frame and only its pose has to be estimated. Performance is measured as the percentage of test frames per sequence where the pose has been estimated correctly where we use the same pose tolerance as Hinterstoisser et al. [HLI+12], see Sec. A.3.1.

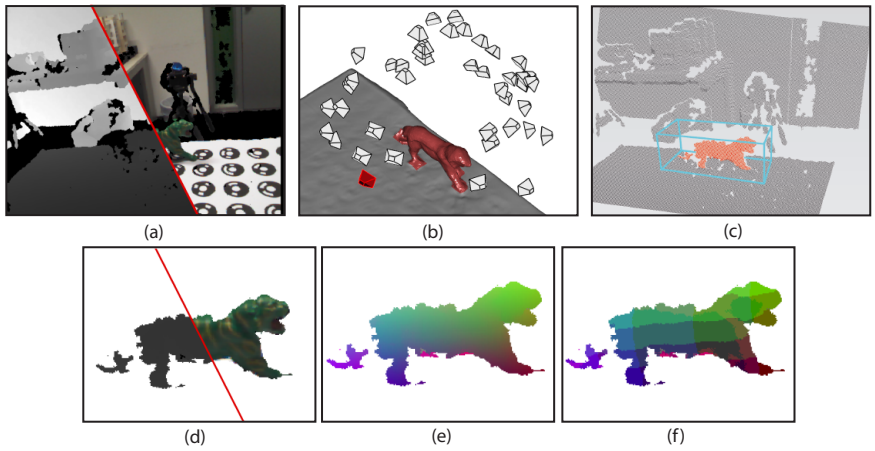

Figure A.3.: **Lighting Data Acquisition Pipeline.** For each object, we run the following acquisition procedure three times to record three different lighting conditions. We register these three different image sequences using the marker board the object is standing on. **(a)** We use the commercially available Kinect camera and KinectFusion [NIH+11] system to obtain a 3D scan of the scene together with the camera poses (b) and the RGB-D data for each captured frame. The black holes in the RGB-D images correspond to pixels were no depth information was available. **(b)** A top-side view with a subset of 50 out of 1,000 camera frusta, and the object (red) in the center. The object has been manually segmented in 3D. **(c)** A 3D bounding box (light blue) is positioned around the object, shown as the depth map from the camera with the red frustum in (b). The object mask for each RGB-D frame is then defined by all object pixels where the corresponding depth values fall inside the bounding box. **(d)** For training, we use the segmented RGB-D images. The segmentation has holes and is imperfect at boundaries, due to the inaccurate and missing depth values from Kinect. However, the test data presents similar noise characteristics. **(e)** We show the continuous 3D object coordinates calculated from the depth channel and the ground truth pose. X/Y/Z is mapped to RGB for visualization. **(f)** We show the quantized 3D object coordinates on a $5 \times 5 \times 5$ grid. We use the quantized object coordinates as proxy classes when computing the objective function for learning the tree structure in Chapter 2.

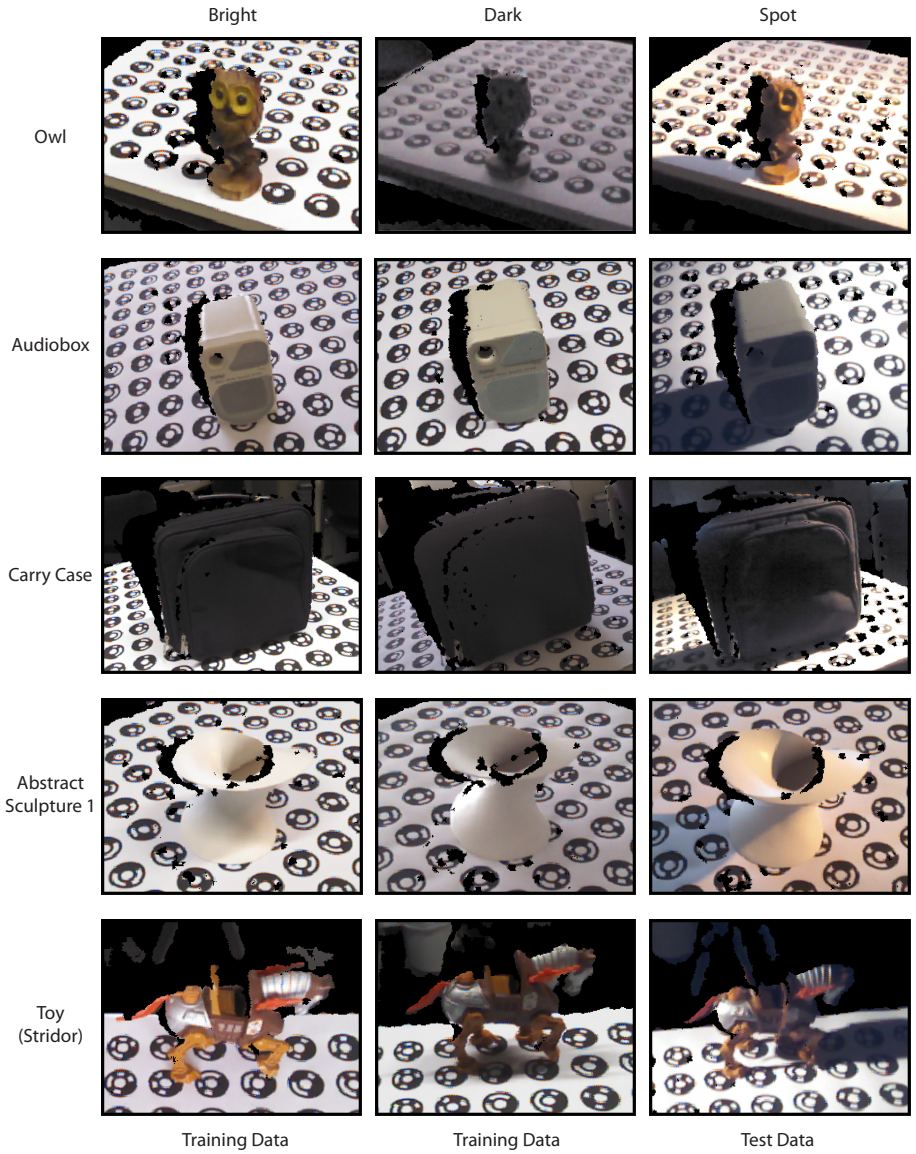

Figure A.4.: **Three Different Lighting Conditions.** For each object, we gathered training and test images under three different lighting conditions: Bright artificial light (left), darker natural light (middle), and directional spot light (right). Note that the markers on the table are used only to register scans of different lighting conditions. We generate ground truth pose annotations using Kinect Fusion [NIH+11].

Figure A.5.: **Objects of the Lighting Dataset.** Each object of our dataset segmented and labeled with its name. This is the same segmentation we use for training our decision forests.

Table A.3.: **Statistics of the Lighting Dataset.** We state the approximate extent of each object. Furthermore we list the number of frames available per lighting condition.

	Approximate Size (width x height x depth)	# Frames		
		Bright Lighting	Dark Lighting	Spotlight Lighting
Audiobox	18cm x 19cm x 20cm	179	165	199
Carry Case	40cm x 33cm x 24cm	175	167	161
Dishsoap	15cm x 18cm x 12cm	128	133	126
Helmet	35cm x 16cm x 27cm	95	135	95
Hole Puncher	27cm x 15cm x 26cm	136	163	140
Pump	19cm x 11cm x 16cm	225	237	247
Teapot	29cm x 15cm x 19cm	262	263	252
Toolbox	44cm x 24cm x 29cm	188	178	191
Toy (Battle Cat)	30cm x 11cm x 13cm	171	177	170
Toy (Panthor)	33cm x 13cm x 14cm	132	132	156
Toy (Stridor)	30cm x 19cm x 17cm	147	195	194
Stuffed Cat	22cm x 25cm x 20cm	139	139	151
Duck	14cm x 8cm x 13cm	186	164	179
Dwarf	14cm x 11cm x 16cm	181	137	170
Mouse	18cm x 17cm x 15cm	202	203	233
Owl	14cm x 17cm x 13cm	388	388	329
Elephant	30cm x 15cm x 17cm	187	184	181
Samurai	17cm x 30cm x 18cm	315	164	241
Abstract Sculpture 1	9cm x 10cm x 12cm	136	119	114
Abstract Sculpture 2	17cm x 9cm x 14cm	126	140	76

A.2.4. 7-SCENES DATASET OF SHOTTON ET AL. [SGZ+13]

The 7-Scenes camera localization dataset was published by Shotton et al. [SGZ+13]. It consists of RGB-D image sequences recorded at 7 different indoor locations measuring $3m^2$ to $12m^2$, see Table A.4 and Fig. A.6 for an overview.

For each of the seven scenes, several image sequences have been recorded by different persons moving the camera through the respective room. Kinect Fusion [NIH+11] was used to track the camera poses for each sequence, and to generate a 3D model of each scene. The dataset is split into training and test sets at the sequence level, e.g. a learning-based method has to generalize to large changes in viewpoint. Other imaging conditions, like lighting or the camera type, do not vary. The dataset features some difficulties in the form of repeating structures, like stairs, motion blur or large texture-less areas. The amount of training and test data per scene vary substantially between 1,000 and 7,000 images, see Table A.4. The camera calibration parameters are given with a focal length of 585px and a principal point in the image center.

Depth and RGB images are not registered and registration information is not given. In our experiments in Chapter 3 and 4, we registered manually using a focal length of 525px for the RGB sensor. We solved for the unknown rigid body transformation between the depth sensor and the RGB sensor by hand-clicking correspondences between RGB and depth channels of a few training images. Accuracy on this dataset is usually measured as the percentage of test frames where the camera pose error is below 5° and 5cm, see also Appendix A.3.3.

Table A.4.: **Statistics of the 7-Scenes Dataset.** We state the approximate area of each scene which we measured in the 3D models given by Shotton et al. Furthermore, we list the number of training and test frames available per scene.

	Approximate Area	# Training Frames	# Test Frames
Chess	3m × 2.5m	4,000	2,000
Fire	3.5m × 3m	2,000	2,000
Heads	2m × 1m	1,000	1,000
Office	4m × 3m	6,000	4,000
Pumpkin	2.5m × 4m	4,000	2,000
Kitchen	4.5m × 2.5m	7,000	5,000
Strairs	3.5m × 2.5m	2,000	1,000

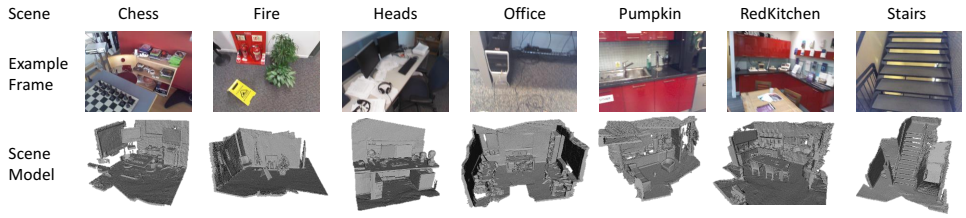

Figure A.6.: **7-Scenes Overview.** We show one example frame for each of the scenes as well as a rendering of the 3D models provided with the dataset.

A.3. EVALUATION MEASURES

Several measures of evaluating pose estimation have been proposed over the years in the literature. In object pose estimation tasks, one usually does not measure the difference between ground truth and estimated pose but rather the alignment of a 3D model of the object under the estimated pose and the ground truth pose. In Sec. A.3.1, we present one measure of this kind which is widely used for object pose estimation from RGB-D inputs. In Sec. A.3.2, we present a version of the previous measure tailored for pose estimation from RGB inputs. Finally, in Sec. A.3.3, we discuss evaluation of pose accuracy in camera localization tasks.

A.3.1. OBJECT POSE: MEASURE OF HINTERSTOISSER ET AL. [HLI+12]

Hinterstoisser et al. [HLI+12] proposed to measure accuracy as the fraction of test images where the pose of the object in question was estimated correctly. Poses are correct, if the following inequality holds:

$$\tau_p < \frac{1}{|\mathcal{V}|} \sum_{\mathbf{v} \in \mathcal{V}} ||\mathbf{h}^* \mathbf{v} - \tilde{\mathbf{h}} \mathbf{v}||, \tag{A.1}$$

where $\mathcal{V}$ is the set of all vertices of a 3D model of the object, $\mathbf{h}^*$ is the ground truth pose and $\tilde{\mathbf{h}}$ is the estimated pose. The measure computes the average distance between corresponding vertices transformed with the estimated pose and the ground truth pose. The threshold τ_p is fixed to be 10% of the object diameter. For rotationally symmetric objects (e.g. a bowl) a

slightly different metric is used:

$$\tau_p < \frac{1}{|\mathcal{V}|} \sum_{\mathbf{v}_1 \in \mathcal{V}} \min_{\mathbf{v}_2 \in \mathcal{V}} ||\mathbf{h}^* \mathbf{v}_1 - \tilde{\mathbf{h}} \mathbf{v}_2||. \tag{A.2}$$

Here, the distance is not measured between corresponding vertices but between vertices of minimal distance.

A.3.2. OBJECT POSE: 2D PROJECTION MEASURE

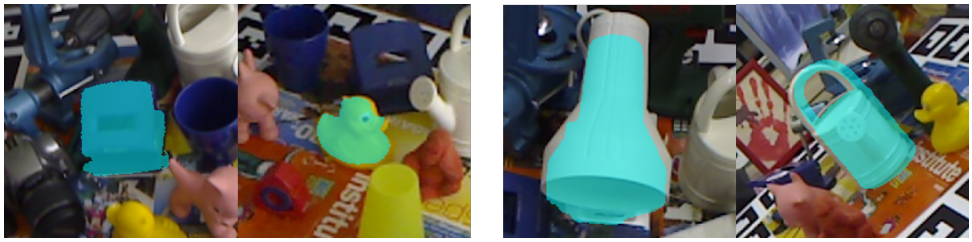

Figure A.7.: **Tolerance of Pose Estimation Measures.** We render object segmentation masks (turquoise) according to estimated poses of the object in the center of each crop. **Left:** Two examples where the segmentation seems to be well aligned visually but the measure of Hinterstoisser et al. [HLI$^+$12] (Sec. A.3.1) rejects both pose estimates because of translation error in z direction. The 2D projection measure (Sec. A.3.2) accepts both pose estimates. **Right:** Two examples where the pose seems to be less well aligned but the error is within tolerance of the measure of Hinterstoisser et al. [HLI$^+$12]. The 2D projection measure rejects both pose estimates.

The measure of Sec. A.3.1 is very sensitive to errors in object translation, e.g. for an object with a diameter of 15cm a maximum translation error of 1.5cm is within tolerance. This makes sense for pose estimation from RGB-D inputs where the location of an object can be estimated accurately in all 3 dimension. However, when poses are estimated from RGB images, the distance between the object and the camera cannot be predicted as accurately. For instance, the projected size of an object might only change the fraction of a pixel if its distance to the camera is changed by 1cm. This observation motivated us to propose a new measure for the evaluation of 6D object pose estimates.

We assess the quality of pose estimation via a tolerance on the average 2D projection error of 3D model vertices. In contrast to the measure of Hinterstoisser et al. [HLI$^+$12] of Sec. A.3.1, the 2D projection measure coincides well with visual alignment quality, see Fig. A.7. Formally, we calculate

$$\frac{1}{|\mathcal{V}|} \sum_{\mathbf{v} \in \mathcal{V}} ||C\mathbf{h}^* \mathbf{v} - C\tilde{\mathbf{h}} \mathbf{v}||_2 < 5\text{px}, \tag{A.3}$$

where $\mathcal{V}$ is the set of all object model vertices, C is the camera matrix, $\mathbf{h}^*$ is the ground truth pose and $\tilde{\mathbf{h}}$ is the estimated pose. We assume normalization of the homogeneous vector before calculating the L_2 norm. Note that we use a fixed tolerance of 5px instead of a tolerance that depends on the object diameter. We found that visual alignment is independent

of object size, and that pose estimation of large objects is usually easier. We also saw in our experiments that the version of the relaxed measure of Hinterstoisser et al. proposed for symmetric objects is overly tolerant to misalignment. Therefore, we do not use a similarly relaxed version of our 2D projection measure for symmetric objects.

A.3.3. CAMERA LOCALIZATION MEASURES

In camera localization tasks, pose accuracy is usually measured for camera orientation and camera position, separately. For orientation, the angular error is calculated between the ground truth camera rotation and the estimated camera rotation, and measured in degree. For position, the Euclidean distance is calculated between the ground truth camera translation and the estimated camera translation. Note that these errors have to be calculated in scene space. Throughout this thesis, we estimate the rigid body transformation $\mathbf{h}$ of an object relative to the camera. The camera pose is the inverse transformation, i.e. $\mathbf{h}^{-1}$, and the aforementioned errors have to be calculated with the inverse transformation, see Fig. A.8.

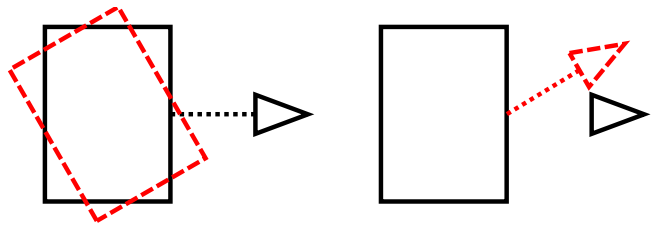

Figure A.8.: **Measuring Error in Camera Localization.** The pose of a scene (black box) has been estimated inaccurately (red box). The camera is depicted as a black triangle. If the error is measured in camera space (left), we observe a certain rotational error but no translational error, compare the centers of the black and red box. If the error is measured in scene space (right), we observe both a rotational and an addtional translational error, compare centers of the black and red triangles.

There are two common ways to measure camera localization accuracy over an image sequence. Shotton et al. [SGZ+13] propose to measure the percentage of test frames where the rotational and translational error is below 5cm and 5°. This measure is commonly used for indoor localization datasets where this high accuracy can be achieved, e.g. in [SGZ+13, VNS+15, GRKG+14, CGL+17]. For larger outdoor scenes, the threshold can be increased, e.g. in [KC17]. But, usually, accuracy for outdoor localization dataset is measured by the median rotational and translational error, respectively, e.g. in [KGC15, WHL+16, KC17].

A.4. DETAILED EXPERIMENTAL RESULTS AND PARAMETERS

In this section, we list detailed results of experiments discussed in Chapter 2 and 3. This includes measured accuracy for individual objects as well as additional qualitative results. For descriptions of the experimental setups and conclusions drawn from the results please see the corresponding sections in the respective chapters. Furthermore, we list detailed parameter setting for training random forests for object coordinate regression, and for RANSAC-based

pose optimization. Detailed results of Chapter 4 as well as the corresponding parameters are given within the chapter.

A.4.1. 6D POSE ESTIMATION USING OBJECT COORDINATES

This chapter was concerned with 6D pose estimation of a single object instance given a single RGB-D image. In particular, we investigated robustness of our method w.r.t. occlusion and lighting changes, and concluded improved accuracy of our method compared to competitors. We also conducted an ablation study w.r.t. the individual components of our scoring function. In these experiments, we measured the percentage of test frames where the pose was estimated correctly subject to the pose tolerance of Sec. A.3.1.

OBJECT POSE ESTIMATION

We list detailed results of the experiment of Sec. 2.3.1 where we estimated 6D poses on the RGB-D dataset of Hinterstoisser et al. [HLI$^+$12]. The following table lists quantitative results for individual objects, and for training the random forest with synthetic (i.e. rendered) and real training data. Furthermore, we show results when training with different background models, namely an infinite plane or uniform noise. Our approach is superior to [HLI$^+$12, RCT13] for most objects.

	Synth. Training				Real Training	
	LINEMOD [HLI$^+$12]	DTT-3D [RCT13]	Our (plane)	Our (noise)	Our (plane)	Our (noise)
Ape	**95.8%**	95.0%	**95.8%**	85.4%	**91.1%**	89.2%
Bench Vise	98.7%	98.9%	**100.0%**	98.9%	**100.0%**	99.7%
Camera	97.5%	98.2%	**99.6%**	92.1%	**98.7%**	95.5%
Can	95.4%	**96.3%**	95.9%	84.4%	**99.6%**	98.9%
Cat	99.3%	99.1%	**100.0%**	90.6%	**99.7%**	98.8%
Driller	93.6%	94.3%	99.5%	**99.7%**	99.9%	**100.0%**
Duck	**95.9%**	94.2%	**95.9%**	92.7%	**96.8%**	94.4%
Box	**99.8%**	**99.8%**	98.0%	91.1%	**100.0%**	99.2%
Glue	91.8%	96.3%	**98.9%**	87.9%	91.7%	**96.7%**
Hole Puncher	95.9%	97.5%	**99.4%**	97.9%	**99.6%**	99.0%
Iron	97.5%	98.4%	97.6%	**98.8%**	99.9%	**100.0**%
Lamp	97.7%	97.9%	**99.8%**	97.6%	**99.1%**	98.7%
Phone	93.3%	95.3%	**97.6%**	86.1%	**98.8%**	95.8%
Bowl	99.9%	99.7%	-	-	-	-
Cup	97.1%	97.5%	-	-	-	-
Average	96.6%	97.2%	**98.3%**	92.6%	**98.1%**	97.4%
Median	97.1%	97.5%	**98.9%**	92.1%	**99.6%**	98.8%
Best	99.9%	99.8%	**100.0%**	99.7%	**100.0%**	**100%**
Worst	91.8%	94.2%	**95.8%**	84.4%	**91.1%**	89.2%

Fig. A.9 shows qualitative results on the same dataset. We show estimated poses as well as predictions of the random forest for the same frame.

ROBUSTNESS W.R.T. OCCLUSION

Using our additional annotations for the dataset of Hinterstoisser et al. [HLI+12], i.e. the occlusion dataset, we measure the accuracy of our method when objects are partially occluded. The following table lists the accuracy of the system of Chapter 2 of individual objects of our occlusion dataset. We also show results using only the depth component of our scoring function to verify the benefit of including the forest predictions in hypothesis scoring.

	Full Score	Depth C. Only	LINEMOD [HLI+12]
Ape	**62.6%**	51.9%	49.8%
Bench Vise	**100.0%**	98.8%	98.7%
Can	80.2%	**98.8%**	51.2%
Cat	**50.0%**	27.7%	34.9%
Driller	**84.3%**	71.8%	59.6%
Duck	**67.6%**	57.8%	65.1%
Box	8.5%	2.4%	**39.6%**
Glue	**62.8%**	33.3%	23.3%
Hole Puncher	**89.9%**	71.5%	67.2%
Average	**67.3%**	57.1%	54.4%
Median	**67.6%**	57.8%	51.2%
Best	**100.0%**	98.8%	98.7%
Worst	8.5%	2.4%	**23.3%**

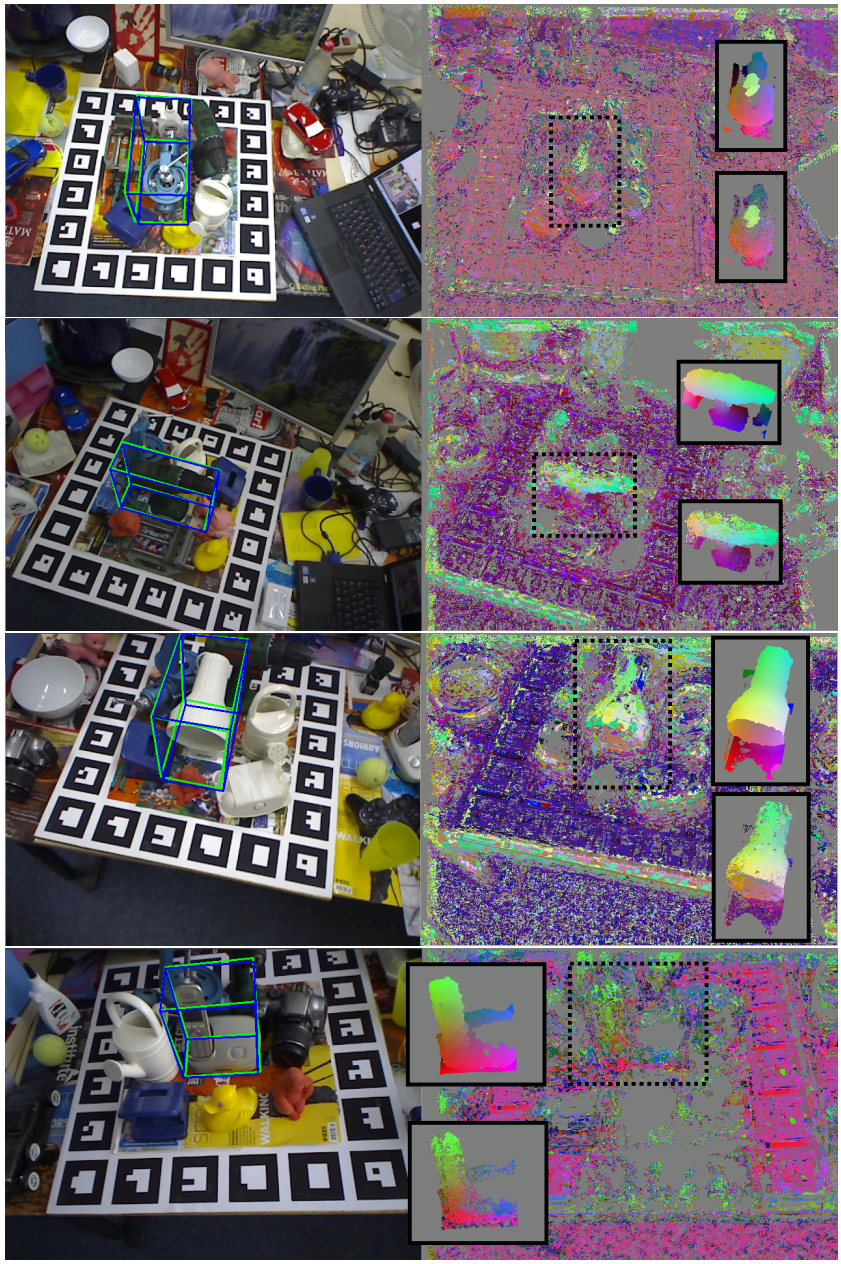

Figure A.9.: **Qualitative Results on the Dataset of Hinterstoisser et al. [HLI+12].** We show poses estimated with the system of Chapter 2 (blue bounding box) versus the ground truth pose (green bounding box). Next to each test image, we visualize the predicted object coordinates $\mathbf{y}$ from one tree of the forest. The inlay figures show the ground truth object coordinates (top) and the best object coordinates (bottom), where "best" is the best prediction of all trees with respect to ground truth (for illustration only).

ROBUSTNESS W.R.T. LIGHTING CHANGES

Using our new lighting dataset we investigated how the system of Chapter 2 can generalize to unseen lighting conditions. All objects have been trained with images from the *bright* and *dark* image sequence. We tested with different images from the *bright* lighting condition, i.e. the system has to generalize to unseen views. Furthermore we tested with images from the *spot* lighting condition, i.e. the system has to generalize to unseen views and an unseen lighting condition. Although the latter test is challenging, the drop in performance is only moderate.

Object	*bright* test condition	*spot* test condition
Audio Box	90.5%	75.4%
Carry Case	97.9%	95.9%
Dish Soap	100.0%	100.0%
Helmet	91.0%	77.6%
Hole Puncher	99.9%	98.1%
Pump	81.5%	69.3%
Teapot	99.5%	91.9%
Toolbox	99.0%	99.5%
Toy (Battle Cat)	96.9%	91.8%
Toy (Panthor)	99.7%	96.9%
Toy (Stridor)	97.8%	94.0%
Stuffed Cat	100.0%	98.3%
Duck	89.9%	81.6%
Dwarf	87.7%	67.6%
Mouse	94.6%	89.1%
Owl	97.3%	60.5%
Elephant	98.6%	94.7%
Samurai	97.7%	98.5%
Sculpture 1	92.5%	82.7%
Sculpture 2	99.9%	100.0%
Average	95.6%	88.2%
Median	97.7%	93.0%
Best	100.0%	100.0%
Worst	81.5%	60.5%

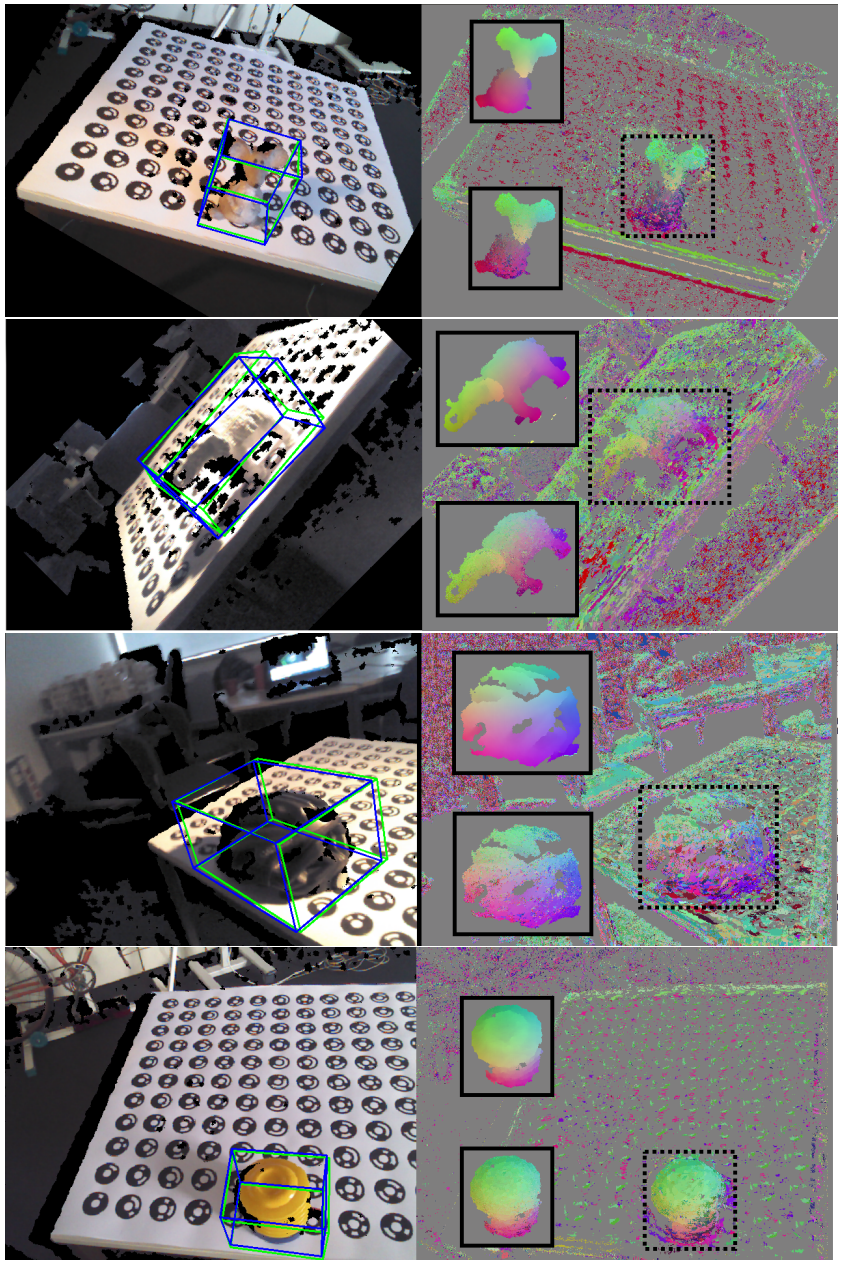

Figure A.10.: **Qualitative Results on our Lighting Dataset.** Poses estimated with our system (blue bounding box) versus the ground truth pose (green bounding box). Next to each test image are the predicted object coordinates $\mathbf{y}$ from one tree of the forest. The inlay figures show the ground truth object coordinates (top) and the best object coordinates (bottom), where "best" is the best prediction of all trees with respect to ground truth (for illustration only).

CONTRIBUTION OF SCORING COMPONENTS

We investigated the contribution of the individual components of the scoring function presented in Chapter 2 on the pose estimation accuracy. We also compare to a simpler inlier based scoring function that does not rely on a 3D model of the object. See the following table for the accuracy per object on the dataset of Hinterstoisser et al. [HLI+12]. The full scoring function gives a significant performance boost over all competing variants.

	Full Score	Depth C. Only	Obj. C. Only	Coord. C. Only	Inlier Score [SGZ+13]
Ape	**95.8%**	88.8%	75.5%	79.8%	61.5%
Bench Vise	**100.0%**	98.8%	50.6%	89.5%	37.7%
Camera	**99.6%**	96.9%	32.8%	85.3%	21.3%
Can	95.9%	**97.4%**	59.7%	70.0%	23.3%
Cat	**100.0%**	97.8%	77.6%	96.7%	47.1%
Driller	**99.5%**	99.1%	55.3%	78.4%	50.0%
Duck	**95.9%**	93.4%	49.2%	76.2%	42.9%
Box	**98.0%**	97.5%	52.6%	49.8%	27.3%
Glue	**98.9%**	95.3%	82.5%	58.7%	63.8%
Hole Puncher	**99.4%**	98.1%	25.6%	91.6%	34.5%
Iron	97.6%	**98.6%**	23.1%	80.5%	44.7%
Lamp	**99.8%**	**99.8%**	35.7%	91.6%	40.9%
Phone	**97.6%**	91.8%	15.1%	60.3%	29.1%
Average	**98.3%**	96.4%	48.9%	77.6%	40.3%
Median	**98.9%**	97.5%	50.6%	79.8%	40.9%
Best	**100.0%**	99.8%	82.5%	96.7%	63.8%
Worst	**95.8%**	88.8%	15.1%	49.8%	21.3%

PARAMETER LISTINGS

The following settings were used in all pose estimation experiments of Chapter 2.

Training Parameters	
maximum feature offset:	20 pixel meters
# feature candidates:	1,000
ratio of 'da-d' to 'da-rgb' features	0.5
# of trees $\lvert\mathcal{T}\rvert$:	3
# samples per image to learn tree structure:	1,000
# samples per image to learn leaf distributions:	5,000
minimum # samples per leaf:	50
object coordinate proxy classes $\hat{\mathbf{y}}$:	125
bandwidth of mean-shift:	25mm

Test Parameters	
score depth comp. weight λ^{depth}:	1.5
score coordinate comp. weight λ^{coord}:	1
score seg. comp. weight λ^{obj}:	1
threshold τ_d used in s_c^{depth}:	50mm
threshold τ_y used in s_c^{coord}:	$(0.2 \cdot \delta_c)^2$ with object diameter δ_c
threshold τ_p used in s_c^{coord}:	10^{-8}
# hypotheses sampled:	210
threshold used during sampling of poses:	$0.05 \cdot \delta_c$ with object diameter δ_c
inlier threshold used in refinement:	20mm
number of hypothesis to be refined:	25

A.4.2. ENHANCED 6D POSE ESTIMATION USING UNCERTAINTY INFORMATION

In this chapter, we conducted experiments regarding pose estimation of an object instance from single RGB images.

SINGLE OBJECT POSE ESTIMATION

In the following table we, list quantitative results on the dataset of Hinterstoisser et al. [HLI+12] using different pose acceptance measures. "2D Projection, Avg. Err. <5px" accepts a pose based on the average re-projection error, see Sec. A.3.2. "2D B. Box, IoU>0.5" accepts a pose if the 2D bounding boxes of the estimation and ground truth overlap at least 50%. "6D Pose, Measure of [HLI+12]" accepts a pose based on aligned of the object's 3D model, see Sec. A.3.3. "6D Pose, Err. <5cm 5°" accepts a pose if the rotational error resp. the translational error is below 5° resp. 5cm. We also list median pose errors in X/Y/Z direction resp. for rotation.

% of test frames correct using RGB inputs.

Object	2D Projection, Avg. Err. <5px	2D B. Box, IoU>0.5	6D Pose, Measure of [HLI+12]	6D Pose, Err. <5cm 5°
Ape	85.2%	98.2%	33.2%	34.4%
Bench Vise	67.9%	97.9%	64.8%	40.6%
Camera	58.7%	96.9%	38.4%	30.5%
Can	70.8%	97.9%	62.9%	48.4%
Cat	84.2%	98.0%	42.7%	34.6%
Driller	73.9%	98.6%	61.9%	54.5%
Duck	73.1%	97.4%	30.2%	22.0%
Egg Box	83.1%	98.7%	49.9%	57.1%
Glue	74.2%	96.6%	31.2%	23.6%
Hole Puncher	78.9%	95.2%	52.8%	47.3%
Iron	83.6%	99.2%	80.0%	58.7%
Lamp	64.0%	97.1%	67.0%	49.3%
Phone	60.6%	96.0%	38.1%	26.8%
Average	73.7%	97.5%	50.2%	40.6%

Median pose accuracy using **RGB** inputs.

Object	Median X Error	Median Y Error	Median Z Error	Median Rot. Error
Ape	0.2cm	0.2cm	2.1cm	6.4°
Bench Vise	0.2cm	0.2cm	2.1cm	5.7°
Camera	0.3cm	0.4cm	2.7cm	6.7°
Can	0.2cm	0.2cm	1.9cm	4.9°
Cat	0.2cm	0.2cm	2.1cm	6.4°
Driller	0.2cm	0.2cm	2.2cm	4.1°
Duck	0.3cm	0.2cm	2.7cm	9.0°
Egg Box	0.2cm	0.2cm	1.8cm	4.2°
Glue	0.3cm	0.2cm	3.1cm	8.4°
Hole Puncher	0.2cm	0.2cm	1.7cm	5.1°
Iron	0.2cm	0.2cm	1.4cm	4.2°
Lamp	0.3cm	0.2cm	1.8cm	4.7°
Phone	0.3cm	0.3cm	3.3cm	6.9°
Average	0.2cm	0.2cm	2.2cm	5.9°

The pipeline of Chapter 3 can also be adapted for RGB-D inputs. In the following, we list the corresponding quantitative results, and median pose errors.

% of test frames correct using **RGB-D** inputs.

Object	2D Projection, Avg. Err. <5px	2D B. Box, IoU>0.5	6D Pose, Measure of [HLI$^+$12]	6D Pose, Err. <5cm 5°
Ape	95.8%	99.7%	98.1%	59.0%
Bench Vise	97.3%	99.2%	99.0%	92.9%
Camera	98.7%	96.9%	99.7%	92.8%
Can	98.6%	99.8%	99.7%	89.6%
Cat	97.9%	99.8%	99.1%	80.1%
Driller	93.2%	99.4%	100.0%	93.1%
Duck	90.5%	100.0%	96.2%	52.1%
Egg Box	98.2%	99.1%	99.7%	96.8%
Glue	92.8%	100.0%	99.0%	55.1%
Hole Puncher	96.1%	99.6%	98.0%	80.3%
Iron	97.0%	99.0%	99.9%	96.9%
Lamp	92.4%	100.0%	99.5%	91.7%
Phone	95.6%	99.8%	99.6%	86.8%
Average	95.7%	99.6%	99.0%	82.1%

Median pose accuracy using **RGB-D** inputs.

Object	Median X Error	Median Y Error	Median Z Error	Median Rot. Error
Ape	0.1cm	0.1cm	0.2cm	4.4°
Bench Vise	0.1cm	0.1cm	0.2cm	2.3°
Camera	0.1cm	0.1cm	0.2cm	2.4°
Can	0.1cm	0.1cm	0.2cm	2.6°
Cat	0.1cm	0.1cm	0.1cm	2.9°
Driller	0.1cm	0.1cm	0.3cm	1.8°
Duck	0.1cm	0.1cm	0.2cm	4.8°
Egg Box	0.1cm	0.2cm	0.2cm	2.2°
Glue	0.1cm	0.1cm	0.2cm	4.6°
Hole Puncher	0.1cm	0.1cm	0.2cm	3.1°
Iron	0.1cm	0.1cm	0.2cm	2.6°
Lamp	0.2cm	0.1cm	0.2cm	2.7°
Phone	0.2cm	0.1cm	0.3cm	2.6°
Average	0.1cm	0.1cm	0.2cm	4.3°

PARAMETER LISTINGS

The following settings were used in all pose estimation experiments of Chapter 2 with RGB inputs.

Training Parameters	
maximum feature offset:	10px
# feature candidates:	1,000
# of trees $\lvert\mathcal{T}\rvert$:	3
# of auto-context levels D:	3
# samples drawn per object:	500k
# samples drawn from background:	1.5M
minimal # samples per leaf:	50
object coordinate proxy classes $\hat{\mathbf{y}}$:	125
bandwidth of mean-shift:	25mm
neighborhood used in g_C^d:	5×5
neighborhood used in $\mathbf{g}_\mathcal{Y}^d$:	3×3
sub-sampling of auto-context feature channels:	2

Test Parameters	
inlier threshold:	3px
# hypotheses sampled:	256
pixel batch size:	100,000
full refinement iterations:	100
full refinement bounds X/Y/Z/Rot:	$\pm$ 50mm/50mm/200mm/10°

For experiments of Chapter 3 using RGB-D inputs, we used a maximum feature offset of 10 pixel meters and an inlier threshold of 10mm. For camera localization experiments, we used a maximum feature offset of 20px and an inlier threshold of 10px. For camera localization we applied random in-plane rotations of $\pm$ 30° to all training images and sub-sampled auto-context feature channels by a factor of 4.

LIST OF FIGURES

BIBLIOGRAPHY

[AGT+16] R. Arandjelović, P. Gronat, A. Torii, T. Pajdla, and J. Sivic. NetVLAD: CNN architecture for weakly supervised place recognition. In *CVPR*, 2016.

[AZ13] R. Arandjelovic and A. Zisserman. All about VLAD. In *CVPR*, 2013.

[BDBC+10] S. Ben-David, J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. W. Vaughan. A theory of learning from different domains. *Mach. Learning*, 2010.

[Bis94] C. M. Bishop. Mixture density networks. Technical report, Aston University, 1994.

[BKM+14] E. Brachmann, A. Krull, F. Michel, S. Gumhold, J. Shotton, and C. Rother. Learning 6D object pose estimation using 3D object coordinates. In *ECCV*, 2014.

[BKN+17] E. Brachmann, A. Krull, S. Nowozin, J. Shotton, F. Michel, S. Gumhold, and C. Rother. DSAC-Differentiable RANSAC for camera localization. In *CVPR*, 2017.

[BLCW09] Y. Bengio, J. Louradour, R. Collobert, and J. Weston. Curriculum learning. In *ICML*, 2009.

[BMK+16] E. Brachmann, F. Michel, A. Krull, M. Y. Yang, S. Gumhold, and C. Rother. Uncertainty-driven 6D pose estimation of objects and scenes from a single RGB image. In *CVPR*, 2016.

[BRF12] L. Bo, X. Ren, and D. Fox. Unsupervised feature learning for RGB-D based object recognition. In *ISER*, June 2012.

[BT74] A. E. Beaton and J. W. Tukey. The fitting of power series, meaning polynomials, illustrated on band-spectroscopic data. *Technometrics*, 1974.

[CBK+11] D. M. Chen, G. Baatz, K. Köser, S. S. Tsai, R. Vedantham, T. Pylvänäinen, K. Roimela, X. Chen, J. Bach, M. Pollefeys, B. Girod, and R. Grzeszczuk. City-scale landmark identification on mobile devices. In *CVPR*, 2011.

[CGL+17] T. Cavallari, S. Golodetz, N. A. Lord, J. Valentin, L. Di Stefano, and P. H. Torr. On-the-fly adaptation of regression forests for online camera relocalisation. In *CVPR*, 2017.

[CM05] O. Chum and J. Matas. Matching with PROSAC - Progressive sample consensus. In *CVPR*, 2005.

[CMK03] O. Chum, J. Matas, and J. Kittler. Locally optimized RANSAC. In *DAGM*, 2003.

[CRV+15] A. Crivellaro, M. Rad, Y. Verdie, K. M. Yi, P. Fua, and V. Lepetit. A novel representation of parts for accurate 3d object detection and tracking in monocular images. In *ICCV*, 2015.

[CS13] A. Criminisi and J. Shotton. *Decision Forests for Computer Vision and Medical Image Analysis*. Springer, 2013.

[CW10] O. Chapelle and M. Wu. Gradient descent optimization of smoothed information retrieval metrics. *Information Retrieval*, 2010.

[DBCMC12] D. Damen, P. Bunnun, A. Calway, and W. Mayol-Cuevas. Real-time learning and detection of 3D texture-less objects: A scalable approach. In *BMVC*, 2012.

[DBKK16] A. Doumanoglou, V. Balntas, R. Kouskouridas, and T. Kim. Siamese regression networks with efficient mid-level feature extraction for 3d object pose estimation. *CoRR*, 2016.

[DGLVG13] M. Dantone, J. Gall, C. Leistner, and L. Van Gool. Human pose estimation using body parts dependent joint regressors. In *CVPR*, 2013.

[DKMK16] A. Doumanoglou, R. Kouskouridas, S. Malassiotis, and T. Kim. 6D object detection and next-best-view prediction in the crowd. In *CVPR*, 2016.

[DMR16] D. DeTone, T. Malisiewicz, and A. Rabinovich. Deep image homography estimation. *CoRR*, 2016.

[DT05] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005.

[DUNI10] B. Drost, M. Ulrich, N. Navab, and S. Ilic. Model globally, match locally: Efficient and robust 3D object recognition. In *CVPR*, 2010.

[EF15] D. Eigen and R. Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *ICCV*, 2015.

[EVGW+a] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2009 (VOC2009) Results. `http://www.pascal-network.org/challenges/VOC/voc2009/workshop/index.html`.

[EVGW+b] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. `http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html`.

[FB81] M. A. Fischler and R. C. Bolles. Random Sample Consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 1981.

[FGMR10] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based models. *TPAMI*, 2010.

[FJS09] V. Ferrari, F. Jurie, and C. Schmid. From images to shape models for object detection. In *IJCV*, 2009.

[FTVG06] V. Ferrari, T. Tuytelaars, and L. Van Gool. Object detection by contour segment networks. In *ECCV*, 2006.

[GHTC03] X.-S. Gao, X.-R. Hou, J. Tang, and H.-F. Cheng. Complete solution classification for the perspective-three-point problem. *TPAMI*, 2003.

[Gir15] R. Girshick. Fast R-CNN. In *ICCV*, 2015.

[GL06] I. Gordon and D. Lowe. What and where: 3D object recognition with accurate pose. In *Toward Category-Level Object Recognition*. 2006.

[GPAM+14] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *NIPS*, 2014.

[GRKG+14] A. Guzman-Rivera, P. Kohli, B. Glocker, J. Shotton, T. Sharp, A. Fitzgibbon, and S. Izadi. Multi-output learning for camera relocalization. In *CVPR*, 2014.

[GSK+11] R. Girshick, J. Shotton, P. Kohli, A. Criminisi, and A. Fitzgibbon. Efficient regression of general-activity human poses from depth images. In *ICCV*, 2011.

[GSS93] N. J. Gordon, D. J. Salmond, and A. F. M. Smith. Novel approach to nonlinear/non-gaussian bayesian state estimation. In *Radar and Signal Processing*, 1993.

[GUA+16] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky. Domain-adversarial training of neural networks. *JMLR*, 2016.

[GWR+16] A. Graves, G. Wayne, M. Reynolds, T. Harley, I. Danihelka, A. Grabska-Barwińska, S. G. Colmenarejo, E. Grefenstette, T. Ramalho, J. Agapiou, et al. Hybrid computing using a neural network with dynamic external memory. *Nature*, 2016.

[GYR+11] J. Gall, A. Yao, N. Razavi, L. Van Gool, and V. Lempitsky. Hough Forests for object detection, tracking, and action recognition. *TPAMI*, 33(11), 2011.

[HAT11] R. Hartley, K. Aftab, and J. Trumpf. L1 rotation averaging using the Weiszfeld algorithm. In *CVPR*, 2011.

[HCI+11] S. Hinterstoisser, S. Holzer, C. Cagniart, S. Ilic, K. Konolige, N. Navab, and V. Lepetit. Multimodal templates for real-time detection of texture-less objects in heavily cluttered scenes. In *ICCV*, 2011.

[HCI+12] S. Hinterstoisser, C. Cagniart, S. Ilic, P. Sturm, N. Navab, P. Fua, and V. Lepetit. Gradient response maps for real-time detection of texture-less objects. *TPAMI*, 2012.

[HHO+17] T. Hodaň, 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 *WACV*, 2017.

[HKR93] D. Huttenlocher, G. Klanderman, and W. Rucklidge. Comparing images using the Hausdorff distance. *TPAMI*, 1993.

[HLI+12] S. Hinterstoisser, V. Lepetit, S. Ilic, S. Holzer, G. Bradski, K. Konolige, and N. Navab. Model based training, detection and pose estimation of texture-less 3D objects in heavily cluttered scenes. In *ACCV*, 2012.

[HLRK16] S. Hinterstoisser, V. Lepetit, N. Rajkumar, and K. Konolige. Going further with point pair features. In *ECCV*, 2016.

[HMO16] T. Hodaň, J. Matas, and Š. Obdržálek. On evaluation of 6D object pose estimation. In *ECCV Workshops*, 2016.

[HMS+] T. Hodaň, F. Michel, C. Sahin, T.-K. Kim, J. Matas, and C. Rother. SIXD challenge 2017. `http://cmp.felk.cvut.cz/sixd/challenge\_2017/`.

[HRBLM07] G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical report, University of Massachusetts, Amherst, 2007.

[HRW07] D. Hoiem, C. Rother, and J. Winn. 3D LayoutCRF for multi-view object class recognition and segmentation. In *CVPR*, 2007.

[HSK12] S. Holzer, J. Shotton, and P. Kohli. Learning to efficiently detect repeatable interest points in depth data. In *ECCV*, 2012.

[Hub64] P. J. Huber. Robust estimation of a location parameter. *The Annals of Mathematical Statistics*, 1964.

[HZ04] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2004.

[HZL+15] T. Hodaň, X. Zabulis, M. Lourakis, Š. Obdržálek, and J. Matas. Detection and fine 3D pose estimation of texture-less objects in RGB-D images. In *IROS*, 2015.

[HZRS15a] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CoRR*, 2015.

[HZRS15b] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. In *ICCV*, 2015.

[IKH+11] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison, and A. Fitzgibbon. KinectFusion: real-time 3D reconstruction and interaction using a moving depth camera. In *Proc. UIST*, 2011.

[IZZE17] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. In *CVPR*, 2017.

[Joh] S. G. Johnson. The NLopt nonlinear-optimization package.

[JPF15] M. Jaberi, M. Pensky, and H. Foroosh. SWIFT: Sparse withdrawal of inliers in a first trial. In *CVPR*, 2015.

[Kab76] W. Kabsch. A solution for the best rotation to relate two sets of vectors. *Acta Crystallographica Section A: Crystal Physics, Diffraction, Theoretical and General Crystallography*, 1976.

[KB14] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, 2014.

[KBC+12] P. Kontschieder, S. R. Bulo, A. Criminisi, P. Kohli, M. Pelillo, and H. Bischof. Context-sensitive decision forests for object detection. In *NIPS*, 2012.

[KBM+15] A. Krull, E. Brachmann, F. Michel, M. Y. Yang, S. Gumhold, and C. Rother. Learning analysis-by-synthesis for 6D pose estimation in RGB-D images. In *ICCV*, 2015.

[KC17] A. Kendall and R. Cipolla. Geometric loss functions for camera pose regression with deep learning. In *CVPR*, 2017.

[KGC15] A. Kendall, M. Grimes, and R. Cipolla. PoseNet: A convolutional network for real-time 6-DoF camera relocalization. In *ICCV*, 2015.

[KHKH16] Y. Konishi, Y. Hanzawa, M. Kawade, and M. Hashimoto. Fast 6D pose estimation from a monocular image using hierarchical pose trees. In *ECCV*, 2016.

[KJC16] A. Kanazawa, D. W. Jacobs, and M. Chandraker. WarpNet: Weakly supervised matching for single-view reconstruction. *CoRR*, 2016.

[KKSC13] P. Kontschieder, P. Kohli, J. Shotton, and A. Criminisi. GeoF: Geodesic forests for learning coupled predictors. In *CVPR*, 2013.

[KMB+14] A. Krull, F. Michel, E. Brachmann, S. Gumhold, S. Ihrke, and C. Rother. 6-DoF model based tracking via object coordinate regression. In *ACCV*, 2014.

[KMT+16] W. Kehl, F. Milletari, F. Tombari, S. Ilic, and N. Navab. Deep learning of local RGB-D patches for 3D object detection and 6D pose estimation. In *ECCV*, 2016.

[KSH12] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In *NIPS*, 2012.

[KTN+16] W. Kehl, F. Tombari, N. Navab, S. Ilic, and V. Lepetit. Hashmod: A hashing method for scalable 3D object detection. In *BMVC*, 2016.

[LB14] M. M. Loper and M. J. Black. OpenDR: An approximate differentiable renderer. In *ECCV*, 2014.

[LBRF11] K. Lai, L. Bo, X. Ren, and D. Fox. A large-scale hierarchical multi-view RGB-D object dataset. In *ICRA*, 2011.

[LF06] V. Lepetit and P. Fua. Keypoint recognition using randomized trees. *TPAMI*, 2006.

[LLC] A. R. LLC. Amazon robotics challenge. `https://www.amazonrobotics.com/\#/roboticschallenge`.

[LMNF09] V. Lepetit, F. Moreno-Noguer, and P. Fua. EPnP: An accurate O(n) solution to the PnP problem. *IJCV*, 2009.

[Low01] D. G. Lowe. Local feature view clustering for 3D object recognition. In *CVPR*, 2001.

[Low04] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 2004.

[LSD15] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015.

[LSZ15] Y. Li, K. Swersky, and R. Zemel. Generative moment matching networks. In *ICML*, 2015.

[MCS10] M. Martinez, A. Collet, and S. S. Srinivasa. MOPED: A scalable and low latency object recognition and pose estimation system. In *ICRA*, 2010.

[MF14] L. Magri and A. Fusiello. T-Linkage: A continuous relaxation of J-Linkage for multi-model fitting. In *CVPR*, 2014.

[MKB+15] F. Michel, A. Krull, E. Brachmann, M. Y. Yang, S. Gumhold, and C. Rother. Pose estimation of kinematic chain instances via object coordinate regression. In *BMVC*, 2015.

[MKB+17] F. Michel, A. Kirillov, E. Brachmann, A. Krull, S. Gumhold, B. Savchynskyy, and C. Rother. Global hypothesis generation for 6D object pose estimation. In *CVPR*, 2017.

[MMDM+16] J. Mund, F. Michel, F. Dieke-Meier, H. Fricke, L. Meyer, and C. Rother. Introducing LiDAR point cloud-based object classification for safer apron operations. In *ESAVS*, 2016.

[MMT15] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós. ORB-SLAM: A versatile and accurate monocular SLAM system. *CoRR*, 2015.

[MSW+11] A. Montillo, J. Shotton, J. Winn, J. E. Iglesias, D. Metaxas, and A. Criminisi. Entangled decision forests and their application for semantic segmentation of CT images. In *IPMI*, 2011.

[NIH+11] R. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. Davison, P. Kohli, J. Shotton, S. Hodges, and A. Fitzgibbon. KinectFusion: Real-time dense surface mapping and tracking. In *Proc. ISMAR*, 2011.

[Nis03] D. Nistér. Preemptive RANSAC for live structure and motion estimation. In *ICCV*, 2003.

[NM65] J. A. Nelder and R. Mead. A simplex method for function minimization. In *Computer Journal*, 1965.

[NS06] D. Nistér and H. Stewénius. Scalable recognition with a vocabulary tree. In *CVPR*, 2006.

[NYB+17] A. Nguyen, J. Yosinski, Y. Bengio, A. Dosovitskiy, and J. Clune. Plug & play generative networks: Conditional iterative generation of images in latent space. In *CVPR*, 2017.

[OCLF10] M. Ozuysal, M. Calonder, V. Lepetit, and P. Fua. Fast keypoint recognition using random ferns. *TPAMI*, 2010.

[Ope17] OpenStreetMap contributors. OpenStreetMap Cartography, CC BY-SA. `https://www.openstreetmap.org`, 2017.

[PCI+07] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *CVPR*, 2007.

[RCT13] R. Rios-Cabrera and T. Tuytelaars. Discriminatively trained templates for 3D object detection: A real time scalable approach. In *ICCV*, 2013.

[RDGF15] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. *CoRR*, 2015.

[RDS+15] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *IJCV*, 2015.

[RFP08] R. Raguram, J.-M. Frahm, and M. Pollefeys. A comparative analysis of RANSAC techniques leading to adaptive real-time random sample consensus. In *ECCV*, 2008.

[RHW88] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Cognitive modeling*, 1988.

[RL17] 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. *CoRR*, 2017.

[RMH+14] V. Ramakrishna, D. Munoz, M. Hebert, J. A. Bagnell, and Y. Sheikh. Pose Machines: Articulated pose estimation via inference machines. In *ECCV*, 2014.

[RPD10] E. Rosten, R. Porter, and T. Drummond. FASTER and better: A machine learning approach to corner detection. *TPAMI*, 32, 2010.

[RRKB11] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. ORB: An efficient alternative to SIFT or SURF. In *ICCV*, 2011.

[RWHS16] J. Revaud, P. Weinzaepfel, Z. Harchaoui, and C. Schmid. DeepMatching: Hierarchical deformable dense matching. *IJCV*, 2016.

[SBXS10] M. Sun, G. R. Bradski, B.-X. Xu, and S. Savarese. Depth-encoded Hough voting for joint object detection and shape recovery. In *ECCV*, 2010.

[SGF+13] J. Shotton, R. B. Girshick, A. W. Fitzgibbon, T. Sharp, M. Cook, M. Finocchio, R. Moore, P. Kohli, A. Criminisi, A. Kipman, and A. Blake. Efficient human pose estimation from single depth images. *TPAMI*, 2013.

[SGZ+13] J. Shotton, B. Glocker, C. Zach, S. Izadi, A. Criminisi, and A. Fitzgibbon. Scene coordinate regression forests for camera relocalization in RGB-D images. In *CVPR*, 2013.

[SHSP17] J. L. Schönberger, H. Hardmeier, T. Sattler, and M. Pollefeys. Comparative evaluation of hand-crafted and learned local features. In *CVPR*, 2017.

[SHWA15] J. Schulman, N. Heess, T. Weber, and P. Abbeel. Gradient estimation using stochastic computation graphs. In *NIPS*, 2015.

[SJC08] J. Shotton, M. Johnson, and R. Cipolla. Semantic texton forests for image categorization and segmentation. In *CVPR*, 2008.

[SKS12] M. Sun, P. Kohli, and J. Shotton. Conditional regression forests for human pose estimation. In *CVPR*, 2012.

[SLK16] T. Sattler, B. Leibe, and L. Kobbelt. Efficient & effective prioritized matching for large-scale image-based localization. *TPAMI*, 2016.

[SPT+17] A. Shrivastava, T. Pfister, O. Tuzel, J. Susskind, W. Wang, and R. Webb. Learning from simulated and unsupervised images through adversarial training. In *CVPR*, 2017.

[SS16] S. Shalev-Shwartz and A. Shashua. On the sample complexity of end-to-end training vs. semantic abstraction training. *CoRR*, 2016.

[SSSI12] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural networks*, 2012.

[STS+17] T. Sattler, A. Torii, J. Sivic, M. Pollefeys, H. Taira, M. Okutomi, and T. Pajdla. Are Large-Scale 3D Models Really Necessary for Accurate Visual Localization? In *CVPR*, 2017.

[SZ14] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, 2014.

[TB10] Z. Tu and X. Bai. Auto-context and its application to high-level vision tasks and 3D brain image segmentation. *TPAMI*, 2010.

[TS14] A. Toshev and C. Szegedy. DeepPose: Human pose estimation via deep neural networks. In *CVPR*, 2014.

[TSLP14] J. Tompson, M. Stein, Y. Lecun, and K. Perlin. Real-time continuous pose recovery of human hands using convolutional networks. In *Proc. ACM SIGGRAPH*, 2014.

[TSSF12] J. Taylor, J. Shotton, T. Sharp, and A. Fitzgibbon. The Vitruvian Manifold: Inferring dense correspondences for one-shot human pose estimation. In *CVPR*, 2012.

[TTKK14] A. Tejani, D. Tang, R. Kouskouridas, and T.-K. Kim. Latent-class Hough forests for 3D object detection and pose estimation. In *ECCV*, 2014.

[TZ00] P. H. S. Torr and A. Zisserman. MLESAC: A new robust estimator with application to estimating image geometry. *CVIU*, 2000.

[TZTV16] J. Thewlis, S. Zheng, P. Torr, and A. Vedaldi. Fully-trainable deep matching. In *BMVC*, 2016.

[VJ01] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *CVPR*, 2001.

[VNS+15] J. Valentin, M. Nießner, J. Shotton, A. Fitzgibbon, S. Izadi, and P. H. S. Torr. Exploiting uncertainty in regression forests for accurate camera relocalization. In *CVPR*, 2015.

[VZ00] Y. Vardi and C.-H. Zhang. The multivariate L1-median and associated data depth. In *Proceedings of the National Academy of Sciences*, 2000.

[WA] M. C. R. WA. Kinect for Xbox 360.

[WCL+08] C. Wu, B. Clipp, X. Li, J. Frahm, and M. Pollefeys. 3D model matching with viewpoint-invariant patches (VIP). In *CVPR*, 2008.

[WHB09] S. Winder, G. Hua, and M. Brown. Picking the best DAISY. In *CVPR*, 2009.

[WHL+16] F. Walch, C. Hazirbas, L. Leal-Taixé, T. Sattler, S. Hilsenbeck, and D. Cremers. Image-based localization with spatial LSTMs. *CoRR*, 2016.

[WL15] P. Wohlhart and V. Lepetit. Learning descriptors for object recognition and 3D pose estimation. In *CVPR*, 2015.

[YM11] G. Yu and J.-M. Morel. ASIFT: An Algorithm for Fully Affine Invariant Comparison. *IPOL*, 2011.

[YTLF16] K. M. Yi, E. Trulls, V. Lepetit, and P. Fua. LIFT: Learned invariant feature transform. In *ECCV*, 2016.

[ZF14] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *ECCV*, 2014.

[ZKM05] M. Zuliani, C. S. Kenney, and B. S. Manjunath. The MultiRANSAC algorithm and its application to detect planar homographies. In *ICIP*, 2005.

[ZPSP15] C. Zach, A. Penate-Sanchez, and M.-T. Pham. A dynamic programming approach for fast and robust object pose recognition from range images. In *CVPR*, 2015.