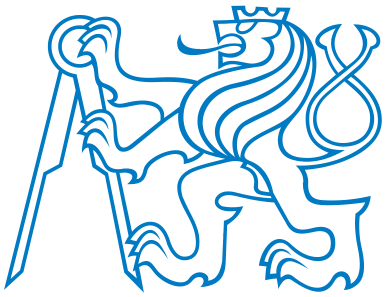

Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Cybernetics

Pose Estimation of Specific Rigid Objects

Doctoral Thesis

Ing. Tomáš Hodaň

Supervisor: Prof. Ing. Jiří Matas, Ph.D.
Ph.D. Programme: Electrical Engineering and Information Technology
Branch of Study: Artificial Intelligence and Biocybernetics

Prague, February 2021

Thesis Supervisor:

Prof. Ing. Jiří Matas, Ph.D.
Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Cybernetics
Karlovo náměstí 13
121 35 Prague 2
Czech Republic

Declaration

I hereby declare I have written this doctoral thesis independently and quoted all the sources of information used in accordance with methodological instructions on ethical principles for writing an academic thesis. Moreover, I state that this thesis has neither been submitted nor accepted for any other degree.

The results presented in this thesis were achieved during my own research in cooperation with my thesis supervisor Jiří Matas, published in [112, 101, 105, 102, 106, 104, 99, 109], with Štěpán Obdržálek, published in [112, 105, 102], with Manolis Lourakis, published in [112, 102], with Xenophon Zabulis, published in [112, 102, 106], with Dima Damen and Walterio Mayol-Cuevas, published in [101], with Pavel Haluza, published in [102], with Carsten Rother, Frank Michel, and Bertram Drost, published in [106, 104, 109], with Eric Brachmann, published in [106, 109], with Tae-Kyun Kim and Caner Sahin, published in [106, 104], with Wadim Kehl, Anders Glent Buch, Dirk Kraft, Joel Vidal, Stephan Ihrke, Fabian Manhardt, and Federico Tombari, published in [106], with Rigas Kouskouridas, Kostas Bekris, Thibault Groueix, Krzysztof Walas, Vincent Lepetit, Ales Leonardis, and Carsten Steger, published in [104], with Vibhav Vineet, Ran Gal, Emanuel Shalev, Jon Hanzelka, Treb Connell, Pedro Urbina, Sudipta N. Sinha, and Brian Guenter, published in [111], with Dániel Baráth, published in [99], with Martin Sundermeyer, published in [109, 54], with Maximilian Denninger, Dominik Winkelbauer, Dmitry Olefir, Youssef Zidan, Mohamad Elbadrawy, Markus Knauer, Harinandan Katam, and Ahsan Lodhi, published in [54], and with Yann Labbé, published in [109].

In Prague, February 2021

.....
Ing. Tomáš Hodaň

Abstract

In this thesis, we address the problem of estimating the 6D pose of rigid objects from a single RGB or RGB-D input image, assuming that 3D models of the objects are available. This problem is of great importance to many application fields such as robotic manipulation, augmented reality, and autonomous driving.

First, we propose EPOS, a method for 6D object pose estimation from an RGB image. The key idea is to represent an object by compact surface fragments and predict the probability distribution of corresponding fragments at each pixel of the input image by a neural network. Each pixel is linked with a data-dependent number of fragments, which allows systematic handling of symmetries, and the 6D poses are estimated from the links by a RANSAC-based fitting method. EPOS is applicable to a broad range of objects, including challenging ones with global or partial symmetries and without any texture, and outperformed all RGB and most RGB-D and D methods on several standard datasets.

Second, we present HashMatch, an RGB-D method that slides a window over the input image and searches for a match against templates, which are pre-generated by rendering 3D object models in different orientations. The method applies a cascade of evaluation stages to each window location, which avoids exhaustive matching against all templates. The key is a voting stage based on hashing that generates a small set of candidate templates, which are then processed by more expensive verification and refinement stages.

Third, we propose ObjectSynth, an approach to synthesize photorealistic images of 3D object models for training methods based on neural networks. The 3D object models are arranged in 3D models of indoor scenes and the images are synthesized by physically-based rendering. The resulting images yield substantial improvements compared to commonly used images of objects rendered on top of random photographs.

Fourth, we introduce T-LESS, a dataset for 6D object pose estimation that includes 3D models and training and test RGB-D images of thirty electrical parts. These objects have no significant texture or discriminative color, exhibit symmetries and similarities in shape and size, and some objects are a composition of others. T-LESS is the first dataset to include objects of such properties which are common in industrial environments.

Fifth, we define BOP, a benchmark that captures the status quo in the field. The benchmark currently comprises eleven datasets in a unified format, an evaluation methodology, an online evaluation system, and public challenges held at international workshops organized annually at the ICCV and ECCV conferences.

Abstrakt

Tématem této disertační práce je odhad 3D pozice a 3D orientace rigidních objektů z jediného RGB nebo RGB-D snímku, kdy 3D modely objektů jsou předem známé. Řešení této úlohy počítačového vidění má široké uplatnění v mnoha aplikacích, jako je například robotická manipulace, rozšířená realita nebo autonomní řízení vozidel.

Prvním přínosem práce je metoda EPOS pro odhad pozice a orientace objektů z RGB snímku. Hlavní myšlenkou je reprezentovat povrch objektů množinou kompaktních fragmentů a pro každý fragment a každý bod na snímku odhadnout pomocí neuronové sítě pravděpodobnost, s jakou daný bod leží na projekci daného fragmentu. Každý bod na snímku je na základě těchto odhadů provázán s potenciálně mnoha fragmenty, což umožňuje zachytit případné symetrie objektu. Pozice a orientace objektů jsou odhadnuty z navázaných korespondencí robustní metodou založenou na algoritmu RANSAC. Metoda EPOS je použitelná pro celou řadu objektů, včetně symetrických objektů a objektů bez textury, a překonala všechny metody pro odhad z RGB snímku a většinu metod pro odhad z RGB-D a D snímku na několika standardních datasetech.

Druhým přínosem je metoda HashMatch, která prochází vstupní RGB-D snímek posuvným oknem a pro každou pozici okna hledá odpovídající šablonu z množiny získané syntézou různých pohledů na 3D modely objektů. Každá pozice okna je vyhodnocena kaskádou kroků, díky které se metoda vyhýbá porovnávání se všemi šablonami. Klíčovým krokem je rychlá identifikace malého počtu potenciálně odpovídajících šablon pomocí hlasování, kde každý hlas je vypočítán z hloubkové informace z několika bodů v okně.

Třetím přínosem je přístup ObjectSynth pro syntézu fotorealistických snímků pro trénování metod využívajících neuronové sítě. 3D modely objektů jsou rozmístěny v 3D modelech místností a snímky jsou získány fyzikálně založeným renderováním. Metody trénované na těchto snímcích dosahují výrazného zlepšení v porovnání s běžně používanými snímky zobrazujícími 3D modely objektů na náhodných fotografiích.

Čtvrtým přínosem je dataset T-LESS, který obsahuje 3D modely a trénovací a testovací snímky třiceti elektronických součástek. Tyto součástky nemají výraznou texturu nebo barvu, v mnoha případech jsou symetrické, vzájemně si podobné tvarem či velikostí, a některé součástky jsou složeninami z ostatních. T-LESS je prvním datasetem obsahujícím objekty těchto vlastností, které jsou časté v průmyslovém prostředí.

Pátým přínosem je srovnávací systém BOP, který zachycuje status quo v odhadu pozice a orientace objektů. BOP aktuálně obsahuje jedenáct datasetů v jednotném formátu, vyhodnocovací metodologii, webový vyhodnocovací portál, a veřejné soutěže pořádané na mezinárodních seminářích organizovaných každoročně na konferencích ICCV a ECCV.

Acknowledgements

Contents

Chapter 1

Introduction

As humans, we heavily rely on visual perception when exploring and navigating the world around us. With a flash glance, we can estimate the three-dimensional structure of the surrounding scene, recognize the scene elements, and understand the semantic context. Researchers in computer vision aim to bring such perception ability to computers, i.e., to create a "seeing machine" [200], by applying mathematical techniques to digital images.

One classical area of computer vision is object recognition, which focuses on extracting information about objects from images. An object is commonly defined as a thing with a well-defined boundary and center [3], and may be of various materials and flexibility properties. Multiple object instances may be visible in an image, with each instance belonging to a possibly different object class, which may represent an object category (dogs, cars, drinks, etc.) or a specific object (a black Ford T from 1908, a 0.33l Coca-Cola can, etc.).

Depending on the required information about object classes or instances visible in the input image, we can distinguish several object recognition problems. The basic problem is image classification [218], where the goal is to predict only labels of visible object classes without requiring any information about individual object instances. The number of visible object instances is additionally required in object counting [7], their bounding boxes in object detection [116], and pixel-accurate delineations in object segmentation [89].

The problem of object pose estimation goes further and requires estimating poses of visible object instances in the 3D space. In the case of rigid objects, the pose has six degrees of freedom, i.e., three in translation and three in rotation, and is referred to as the 6D object pose (Figure 1.1). This problem is of great importance to numerous application fields. For example, in robotic manipulation, the information about object poses allows an end effector to act upon the objects. In augmented reality, this information enables enhancing one's perception of reality by visually augmenting the objects with extra information such as hints for assembly. In autonomous driving, the poses of surrounding vehicles, pedestrians, and obstacles facilitate the planning of the next actions.

1.1 Problem Formulation

In this thesis, we consider the problem of estimating the 6D pose of specific rigid objects with available 3D mesh models. In particular, given the 3D models and a set of training images showing object instances in known 6D poses, the considered problem is to estimate

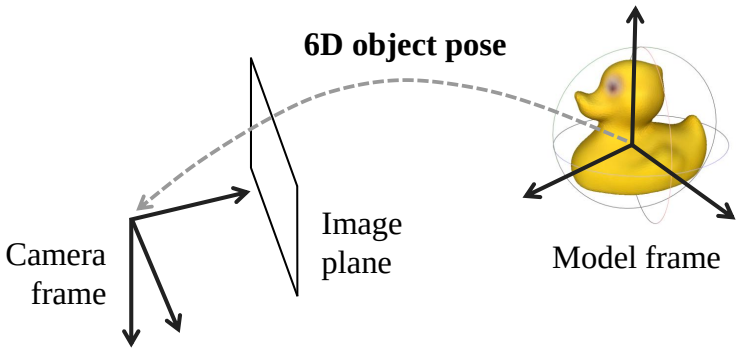

Figure 1.1. **6D pose of a rigid object.** The size and shape of a rigid object are fixed and remain unaltered when forces are applied. The pose of a rigid object is therefore fully defined by a rigid transformation with six degrees of freedom (three in translation and three in rotation) from the 3D coordinate frame of the object model to the 3D coordinate frame of the camera.

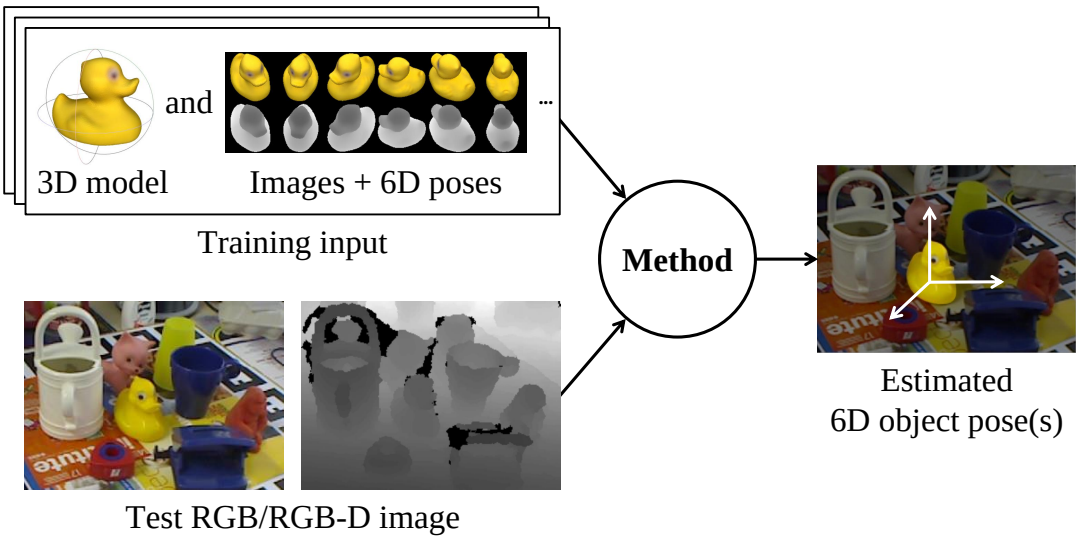

Figure 1.2. **Estimating 6D pose of specific rigid objects.** At training time, a method is given 3D object models and images annotated with 6D object poses. At test time, the goal of the method is to estimate the 6D poses of visible object instances from a single input image. The images may have RGB or RGB-D channels and the intrinsic camera parameters are known.

the 6D poses of possibly multiple instances of possibly multiple object classes from a single test image. The training and test images may have RGB or RGB-D (aligned color and depth) channels and the intrinsic camera parameters are assumed known (Figure 1.2).

Depending on whether prior information about the object instances visible in the test image is provided, we distinguish two variants of the 6D object pose estimation problem: (1) 6D object localization, where the identifiers of visible object instances are provided, and (2) 6D object detection, where no prior information is provided. The first problem is relevant for robotic assembly, where the robot needs to find a specific set of objects which are guaranteed to be present in the workspace, while the second is relevant for augmented

reality and autonomous driving, where the surrounding scene is typically unknown.

We primarily focus on the 6D object localization problem under a practical scenario where no real images of the objects are available at training time, only the 3D object models and images synthesized using the models. While capturing real images of objects under various conditions and annotating the images with 6D object poses requires a significant human effort [102], the 3D models are either available before the physical objects, which is often the case for manufactured objects, or can be reconstructed at an admissible cost.[1] Moreover, besides being useful for estimating object poses, the 3D models enable further reasoning about the objects. For example, a robot can use the estimated poses together with the model surface to compute grasps of the objects.

1.2 Limitations of Existing Methods

A popular approach to 6D object pose estimation is to extract regions of the input image that possibly contain an object, and for each region holistically predict the object class, the object pose, and a confidence score. The image regions can be extracted by instance segmentation [89], object proposal generation [113], or by exhaustive enumeration of regions over the whole image. The per-region prediction is traditionally made by matching the region against prototype image representations, called templates, which exhaustively capture different object appearances [20, 94]. The time complexity of such an approach is linear in the number of templates and the processing time quickly becomes prohibitive with an increasing number of objects and with a growing variety of object appearances. In this thesis, we propose an extension of the template matching approach which makes the time complexity largely unaffected by the number of templates.

Recently, efficient per-region prediction has been achieved with neural networks that learn object appearances implicitly through the network parameters [127, 146, 237]. The time complexity of these methods is constant in the number of learned objects, although learning new objects requires re-training the network and the accuracy tends to degrade with the number of learned objects [126]. As shown by the failure cases presented in [127, 237], estimating pose of partially occluded objects remains challenging for the recent holistic methods, even when occlusions are simulated at training time [237].

Another common approach to 6D object pose estimation, which tends to be more robust to occlusions, is to establish 2D-3D correspondences between pixels of the input image and locations on the 3D object model, and robustly estimate the pose by the PnP-RANSAC algorithm [71, 145]. Early methods establish the correspondences by matching local image features, such as SIFT [156], while recent methods mostly rely on neural networks and predict either the corresponding 3D location at each pixel [17, 188, 277] or the 2D projections of pre-selected 3D keypoints [205, 182, 192].

[1] 3D reconstruction approaches for opaque, matte, and mildly specular objects are established [178] and promising approaches for transparent and highly specular ones are emerging [204, 269, 81].

Establishing 2D-3D correspondences is challenging for objects with global or partial symmetries [169] in shape and texture. The visible part of such objects, which is determined by self-occlusions and occlusions by other objects, may have multiple fits to the object model. Consequently, the potentially corresponding 2D and 3D locations form a many-to-many relationship, i.e., a 2D image location may correspond to multiple 3D locations on the object model surface and vice versa. The existing correspondence-based methods assume a one-to-one relationship and therefore cannot reliably handle objects with symmetries. Additionally, methods relying on local image features have a poor performance on texture-less objects as the feature detectors often fail to provide a sufficient number of reliable locations and the descriptors are no longer discriminative enough [248, 112]. In this thesis, we propose a method that can handle the many-to-many relationship of potentially corresponding locations and is applicable to a broad range of objects, including objects with symmetries and objects without any texture.

One of the critical factors to the success of neural networks is the availability of a large number of training images [82]. Researchers often resort to training on synthetic images as obtaining real training images annotated with 6D object poses is expensive. The images are typically synthesized by rendering the 3D object models and pasting the renderings on top of random photographs [234, 205, 127, 97]. However, the evident domain gap between these "render & paste" training images and real test images limits the potential of neural networks. In this thesis, we propose an approach to generate highly photorealistic training images and show significant improvements over the "render & paste" images.

Although many methods for 6D object pose estimation have been published recently, it has been largely unclear which methods perform well and in which scenarios. This has been the case because new methods have usually been compared with only a few competitors on a small subset of publicly available datasets. Moreover, evaluating the 6D object pose estimates is not straightforward. Due to object symmetries and occlusions, there may be multiple 6D poses consistent with the image and no standard evaluation methodology that can reliably handle such ambiguities has emerged. In this thesis, we define a benchmark that includes a new evaluation methodology and multiple datasets covering different practical scenarios. We also present a new industry-relevant dataset.

1.3 Contributions

In this thesis, we make the following contributions:

EPOS. We present a novel method for 6D object pose estimation from an RGB image applicable to a broad range of objects, including objects with global or partial symmetries and objects without any texture. The key idea is to represent an object by a set of compact surface fragments, predict the probability distribution of corresponding fragments at each pixel of the input image by a neural network, and link each pixel with possibly multiple

surface fragments. The 6D poses are estimated from the predicted many-to-many 2D-3D correspondences by a RANSAC-based robust fitting procedure. In the BOP Challenge 2019 [100, 106], the method outperformed all RGB and most RGB-D and D methods on the T-LESS [102], YCB-V [271], and LM-O [17] datasets.

HashMatch. Another proposed method extends the traditional template matching paradigm with a cascade of evaluation stages that is applied to each location on an RGB-D input image. The cascade avoids exhaustive matching of every location against all templates and makes the time complexity of the method sub-linear in the number of templates. The key stage of the cascade is a voting procedure based on hashing that generates a small set of candidate templates, which are then processed by more expensive verification and refinement stages. The method placed fourth out of the 15 participants in the BOP Challenge 2017 [106] and was successfully deployed in a robotic assembly project.

ObjectSynth. We propose a procedural approach to synthesize photorealistic images of 3D object models, which can be used to effectively train neural networks for object pose estimation from real images. The key ingredients are: (1) 3D models of objects are arranged in 3D models of indoor scenes with realistic materials and lighting, (2) plausible geometric configurations of objects and cameras are generated by physics simulation, and (3) a high level of photorealism is achieved by physically-based rendering. The effectiveness of the synthesized training images is demonstrated on object detection experiments and confirmed in the BOP Challenge 2020 [109], where strong object pose estimation results are achieved with images synthesized by a refined version of the proposed approach.

T-LESS. By creating the T-LESS dataset, we introduce industry-relevant objects to the academic community. The included objects have no significant texture or discriminative color, exhibit symmetries and similarities in shape and size, and some objects are a composition of others. T-LESS provides two types of 3D models for each object (a manually created CAD model and a semi-automatically reconstructed one), and training and test images captured with three synchronized sensors (a structured-light and a time-of-flight RGB-D sensor and a high-resolution RGB camera) and annotated with 6D object poses.

BOP. To capture the status quo in 6D object pose estimation, we created the BOP benchmark that currently comprises of (1) eleven datasets in a unified format covering different practical scenarios, (2) an evaluation methodology with three new pose-error functions, which address limitations of the previously used functions, (3) an online evaluation system at `bop.felk.cvut.cz`, which is open for continuous submission of new results and reports the up-to-date state of the art, and (4) public challenges held at the Workshops on Recovering 6D Object Pose [110], which we organize at the ICCV and ECCV conferences.

1.4 Structure of the Thesis

The rest of the thesis is organized as follows. Chapter 2 reviews existing methods, datasets, and evaluation methodologies for 6D object pose estimation. Chapter 3 proposes Hash-Match, a method for efficient template matching. Chapter 4 presents EPOS, a method that can systematically handle objects with symmetries. Chapter 5 presents ObjectSynth, an approach for the synthesis of photorealistic training images. Chapter 6 describes T-LESS, the first dataset with industry-relevant objects. Chapter 7 introduces BOP, a widely-accepted benchmark for 6D object pose estimation. Finally, Chapter 8 concludes the thesis and suggests topics for future work.

1.5 Publications

The thesis builds on the following publications (chronologically ordered):

[112] **Tomáš Hodaň**, Xenophon Zabulis, Manolis Lourakis, Štěpán Obdržálek, Jiří Matas. *Detection and Fine 3D Pose Estimation of Texture-less Objects in RGB-D Images.* International Conference on Intelligent Robots and Systems (IROS), 2015.

[105] **Tomáš Hodaň**, Jiří Matas, Štěpán Obdržálek. *On Evaluation of 6D Object Pose Estimation.* European Conference on Computer Vision Workshops (ECCVW), 2016.

[102] **Tomáš Hodaň**, Pavel Haluza, Štěpán Obdržálek, Jiří Matas, Manolis Lourakis, Xenophon Zabulis. *T-LESS: An RGB-D Dataset for 6D Pose Estimation of Texture-less Objects.* Winter Conference on Applications of Computer Vision (WACV), 2017.

[106] **Tomáš Hodaň**, Frank Michel, Eric Brachmann, Wadim Kehl, Anders Glent Buch, Dirk Kraft, Bertram Drost, Joel Vidal, Stephan Ihrke, Xenophon Zabulis, Caner Sahin, Fabian Manhardt, Federico Tombari, Tae-Kyun Kim, Jiří Matas, Carsten Rother. *BOP: Benchmark for 6D Object Pose Estimation.* European Conference on Computer Vision (ECCV), 2018.

[104] **Tomáš Hodaň**, Rigas Kouskouridas, Tae-Kyun Kim, Federico Tombari, Kostas Bekris, Bertram Drost, Thibault Groueix, Krzysztof Walas, Vincent Lepetit, Ales Leonardis, Carsten Steger, Frank Michel, Caner Sahin, Carsten Rother, Jiří Matas. *A Summary of the 4th International Workshop on Recovering 6D Object Pose.* European Conference on Computer Vision Workshops (ECCVW), 2018.

[111] **Tomáš Hodaň**, Vibhav Vineet, Ran Gal, Emanuel Shalev, Jon Hanzelka, Treb Connell, Pedro Urbina, Sudipta N. Sinha, Brian Guenter. *Photorealistic Image Synthesis for Object Instance Detection.* International Conference on Image Processing (ICIP), 2019.

[99] **Tomáš Hodaň**, Dániel Baráth, Jiří Matas. *EPOS: Estimating 6D Pose of Objects with Symmetries*. Conference on Computer Vision and Pattern Recognition (CVPR), 2020.

[54] Maximilian Denninger, Martin Sundermeyer, Dominik Winkelbauer, Dmitry Olefir, **Tomáš Hodaň**, Youssef Zidan, Mohamad Elbadrawy, Markus Knauer, Harinandan Katam, Ahsan Lodhi. *BlenderProc: Reducing the Reality Gap with Photorealistic Rendering*. Robotics: Science and Systems (RSS) Workshops, 2020.

[109] **Tomáš Hodaň**, Martin Sundermeyer, Bertram Drost, Yann Labbé, Eric Brachmann, Frank Michel, Carsten Rother, Jiří Matas. *BOP Challenge 2020 on 6D Object Localization*. European Conference on Computer Vision Workshops (ECCVW), 2020.

Other publications not explicitly discussed in the thesis:

[101] **Tomáš Hodaň**, Dima Damen, Walterio Mayol-Cuevas, Jiří Matas. *Efficient Textureless Object Detection for Augmented Reality Guidance*. International Symposium on Mixed and Augmented Reality Workshops (ISMARW), 2015.

[190] Yash Patel, **Tomáš Hodaň**, Jiří Matas. *Learning Surrogates via Deep Embedding*. European Conference on Computer Vision (ECCV), 2020.

Chapter 2

Related Work

Research in object pose estimation has closely followed trends common in the whole field of computer vision. Early methods, dating back to the work of Roberts from 1963 [213], had to deal with a limited computational power and typically aim to fit 3D object models to the intensity edges extracted from the input image, which provide an efficient and relatively stable representation (Section 2.1.1). Following the success of SIFT-like local features from the turn of the century [156], later methods estimate the object pose from correspondences that are established between the input image and the object model by matching the local features (Section 2.1.2). With the introduction of Microsoft Kinect in 2010, the attention of the research field was steered towards estimating the object pose from RGB-D or D-only images, yielding methods based on 3D local features (Section 2.1.3), particularly successful point pair features (Section 2.1.4), and methods based on RGB-D template matching (Section 2.1.5).

Significant improvements in object pose estimation have been recently achieved by machine learning techniques. The contributions of most of the "classical" methods mentioned above are related to extracting and matching representations suitable for the problem at hand. In contrast, recent methods typically count on representations learned by general function approximators, such as random forests or deep neural networks, and focus on more abstract, conceptual aspects of the problem such as formulating the training loss function, handling object symmetries, or bridging the domain gap between synthetic training and real test images (Section 2.2).

2.1 Classical Methods

The classical, non-learning-based methods for object pose estimation are split in this section according to the image representations they rely on, as the chosen image representation often determines the type of objects and scenes for which the method is suitable. Sections 2.1.1–2.1.4 describe methods based on matching local or semi-local features extracted from an image or a point cloud, and Section 2.1.5 describes methods based on matching holistic templates of the objects against regions of the input image.

2.1.1 Edge-Based Features

In his Ph.D. thesis, Roberts [213] assumes that objects can be constructed from transformations of known simple 3D wire-frame models. His approach is to detect intensity edges in the input image, match junctions of the intensity edges with junctions of the model edges, and solve for the pose by minimizing the re-projection error (Figure 2.1). A similar approach was proposed by Lowe [155], who assumes the 3D object model is known and establishes correspondences between groups of edges that are likely to be invariant over a wide range of viewpoints (examples include instances of collinearity, proximity, or parallelism). The scalability of the matching stage to multiple objects is addressed by Lamdan and Wolfson [141] with geometric hashing, and by Beis and Lowe [12] with an approximate nearest-neighbour search. Damen et al. [49] propose a scalable approach which traces constellations of short edge segments using pre-defined paths, matches the constellations against training images of the objects using a pre-calculated hash table, and verifies the hypotheses by the distance transform. A similar approach based on grouping of neighboring edge segments is proposed by Tombari et al. [248], who also demonstrates the superiority of edge-based features for detection of texture-less objects over the texture-based local features described in Section 2.1.2.

Edge-based methods have also been proposed for the related task of object category recognition. For example, Carmichael and Hebert [29] apply weak classifiers to assign individual edge pixels to the target object or to clutter, based on the configuration of edges in the vicinity. Chia et al. [35] extract shape primitives composed of edge segments and ellipses from the input image, match the primitives against a pre-calculated codebook, and cast votes for the object center. Similarly, Opelt et al. [185] apply boosting to classify contour fragments which then vote for the object center. Danielsson et al. [50] learn consistent constellations of edge segments over object categories from training images. The most consistent pair of edge segments is selected as the aligning pair and exhaustively

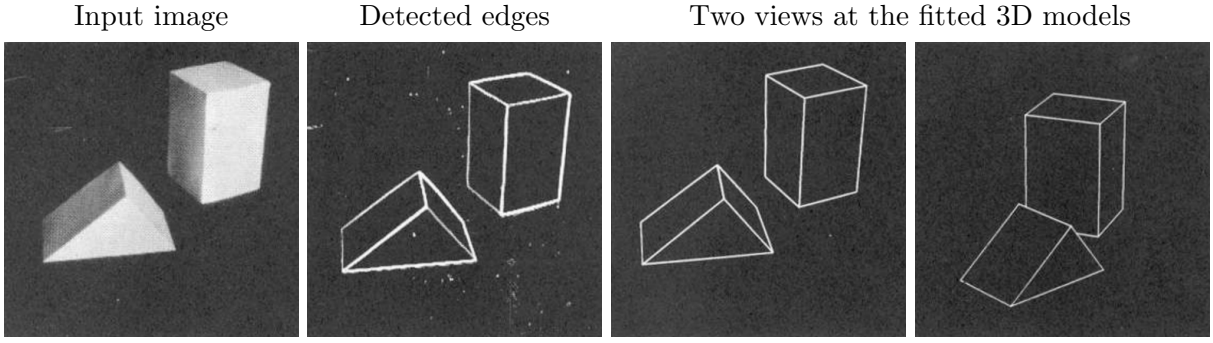

Figure 2.1. **Roberts [213] fits simple 3D models to intensity edges.** The models are represented by a set of line segments in 3D, junctions of the segments are matched with junctions of intensity edges detected in the input image, and the object pose is estimated from the established correspondences.

matched against all pairs in the input image. An extension to multiple edge segments forming a fully connected clique was proposed by Leordeanu et al. [144].

Many of the edge-based methods above rely on edge maps computed with traditional edge detectors such as Canny [28]. In [101], we show that the performance of the method by Damen et al. [49] can be improved if the the Canny detector is replaced with the detector by Dollár and Zitnick [58], which produces cleaner edge maps and favor the object outline while eschewing noise and higher-frequency patterns. The improved method works well on texture-less objects in clean scenes but struggles from spurious edges of shadows and background clutter – this limitation is typical for edge-based methods.

2.1.2 2D Local Features

Methods from this category represent an object by a set of discriminative 2D local features, which are extracted from training (color or grayscale) images of the object and need to be registered in a common 3D coordinate system to enable estimating the 6D object pose. The local features are typically invariant to changes in illumination and to similarity or affine image transformations. Examples of such features include SIFT [156], MSER [161], and SURF [11]. At test time, the training features are matched against features extracted from the test image, and the 6D object poses are estimated from the established 2D-3D correspondences typically by the PnP-RANSAC algorithm [71, 145].

Lowe [156] matches the SIFT features between a set of training images and the test image without recovering any 3D information but demonstrates the robustness of the approach against occlusion and clutter in the case of objects with a distinct and non-repeatable shape or texture. A survey of subsequent approaches that aim to recognize the objects but not to estimate their poses in the 3D space is provided by Matas and Obdržálek [162]. Ponce et al. [199] represent an object by a set of small affine-covariant image patches and a description of their 3D spatial relationship. This representation is created automatically from a small set of unordered training images and enables estimating the 6D object pose (Figure 2.2). Muja and Lowe [173] created the widely used FLANN library for an approximate nearest neighbor search with automatic algorithm configuration, which provides a significant improvement in speed of the feature matching stage. Collet et al. [42] achieve robust performance with a novel algorithm that iteratively combines feature clustering with robust pose estimation – the feature clustering quickly partitions the scene and produces object hypotheses that are used to refine the feature clusters, and the two steps iterate until convergence.

Methods based on 2D local features excel on objects with a distinct and non-repeatable texture but have difficulties with texture-less objects (Figure 2.3), where the feature detectors often fail to provide a sufficient number of reliable locations and the descriptors are not discriminative enough to provide reliable correspondences [248]. Moreover, methods based on 2D local features tend to have a poor performance on objects with symmetries, where the features are in a many-to-many relationship, i.e., a feature in the input image

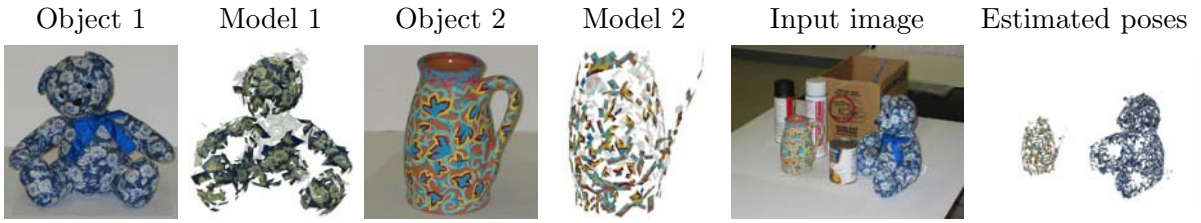

Figure 2.2. **Ponce et al. [199] represent an object by small patches registered in 3D.** This representation is created automatically from a small set of unordered training images. Registration in 3D enables estimating the 6D object pose from correspondences between the model patches and patches detected in the input image.

Figure 2.3. **Textured vs. texture-less objects.** Objects without a significant texture (right) are common in household, office, and industrial environments and are challenging for methods relying on 2D local features. This is because the feature detectors often fail to provide a sufficient number of reliable locations and the descriptors are not discriminative enough [248]. The shown images originate from [156, 125, 271, 181, 102, 248, 114, 25].

potentially corresponds to multiple features on the object model, and vice versa (Section 2.2.6). This degrades the performance of the methods which usually assume a one-to-one relationship. In Chapter 4, we present a correspondence-based method that learns the many-to-many relationship and can handle both symmetric and texture-less objects.

2.1.3 3D Local Features

A number of techniques to detect and describe repeatable and distinctive local features on 3D surfaces have been proposed after the introduction of consumer-grade depth sensors such as Microsoft Kinect, Primesense Carmine, or Intel RealSense. Examples of the 3D local features include SHOT [219], KPQ [165], ISS [280], and MeshDoG [256], and their comprehensive evaluation is provided in [249, 85]. Methods based on these features often assume that the 3D object models are available, typically in the form of mesh models.

Features extracted from the 3D models are organized into a database, matched against features extracted from the input point cloud (calculated from the depth image channel and known camera parameters), and the poses are estimated from the established correspondences, e.g., by a RANSAC-based estimation technique [186]. The pose hypotheses can be further refined individually by a surface registration method, such as Iterative Closest Point (ICP) [13, 217], or globally by optimizing all hypotheses together [186, 2]. Guo et al. [84] provide a detailed survey of methods based on 3D local features.

Similarly to methods based on 2D local features (Section 2.1.2), methods based on 3D local features tend to have a poor performance on objects with symmetries. This was also shown by the relatively low accuracy scores of the methods by Buch et al. [21, 22] on the symmetric objects from the T-LESS dataset (Section 7.3).

2.1.4 Point Pair Features

In 2010, Drost et al. [63] introduced a depth-based method that even in 2020 still belongs to the top-performing methods. Based on the idea of surflet pairs [259], the method matches pairs of oriented points (given by the 3D coordinates and the surface normals) between the point cloud of the test scene and the object model, and groups the matches via a voting scheme. At training time, point pairs from the model are exhaustively sampled and stored in a hash table. At test time, reference points are sampled in the scene and a low-dimensional voting space is defined for each reference point by restricting to those poses that align the reference point with a model point (in both the location and the orientation). Point pairs are then created from the reference point and other points sampled in the scene, similar point pairs are retrieved from the hash table, and each retrieved point pair casts a vote for an object pose. The votes are accumulated to produce a set of pose hypotheses, which are finally refined by a coarse-to-fine ICP algorithm and re-scored by the relative amount of the visible model surface (Figure 2.4).

Multiple improvements of the method by Drost et al. [63] have been proposed. Kim and Medioni [133] include visible context information, differentiating visible points, points on the surface, and invisible points. Choi and Christensen [36] augment the point pair

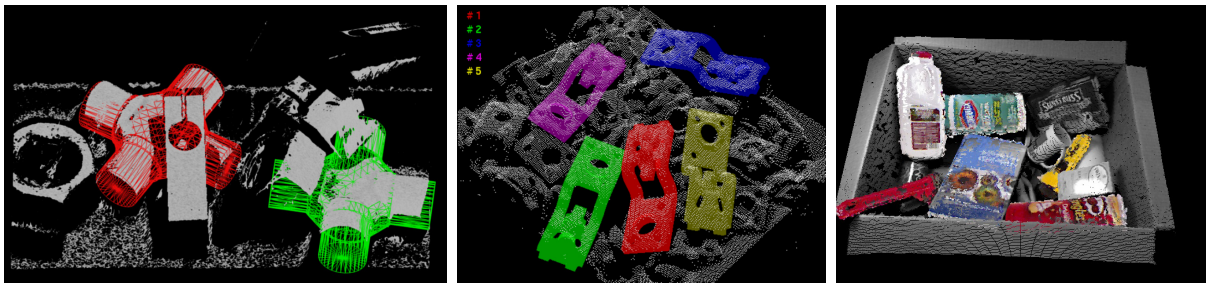

Figure 2.4. **3D models aligned to point clouds using point pair features [63, 37, 36].** The point clouds are calculated from the depth image channel and known camera parameters.

feature with color information. Drost and Ilic [61] propose a multimodal extension including edge information extracted from RGB image channels. Birdal and Ilic [14] reduce the search space by a coarse-to-fine segmentation, favor points with higher expected visibility, and filter pose hypotheses by an occlusion-aware ranking. Hinterstoisser et al. [96] introduce novel sampling and voting schemes that significantly reduce the influence of clutter and sensor noise. Inspired by [96], Vidal et al. [257] propose further improvements, including a novel view-dependent verification process, and achieved the top scores in the first two editions of the BOP Challenge in 2017 and 2019 (Sections 7.3 and 7.4). In the 2020 edition of the challenge, the point pair features were represented by a hybrid method by König and Drost [135], which placed second overall and first among fast methods with the average processing time below 1 s per image. This method starts by segmenting object instances by a deep neural network and then uses the point pair features to estimate the object poses from subsets of the point cloud defined by the instance masks.

Kiforenko et al. [132] present a comprehensive evaluation of the point pair features, including a comparison with 3D local features such as SHOT [219]. The comparison shows that the point pair features perform better in general but are inferior on point clouds with a higher degree of noise and with a lower resolution.

2.1.5 Template Matching

These methods compare prototype image representations, called templates, against regions of the input image that are of the same size as the templates and are usually extracted by an exhaustive sliding window technique. The templates show an object under different conditions, such as different viewpoints and illumination, and are obtained by capturing the object in a controlled environment (Figures 2.5 and 6.3) or by rendering the 3D model of the object [94]. In both cases, the templates are automatically annotated with 6D object poses. When a template is detected, the associated 6D pose can be used as a starting point for an edge-based or a depth-based refinement procedure [25, 94, 112].

Murase and Nayar [176] avoid exhaustive matching of an image region against all templates by compressing the templates into a low-dimensional eigenspace where each object is represented by a manifold. The identity of the object in an image region is determined by projecting the region to the eigenspace and finding the corresponding manifold, and the object pose is determined by the position on the manifold. Cai et al. [25] avoid exhaustive template matching by an efficient edge-based voting procedure that for each image region rapidly retrieves a small set of candidate templates, which are then verified by the oriented chamfer matching [224]. The voting procedure is based on hashing of distances and orientations of intensity edges that are the closest to fixed reference points, which are located on a regular grid attached to the sliding window. Multiple hash tables are used to increase the robustness against occlusion, with the hash key for each table calculated by discretizing measurements from a different subset of reference points. In Chapter 3, we present an RGB-D method that avoids exhaustive template matching by applying a

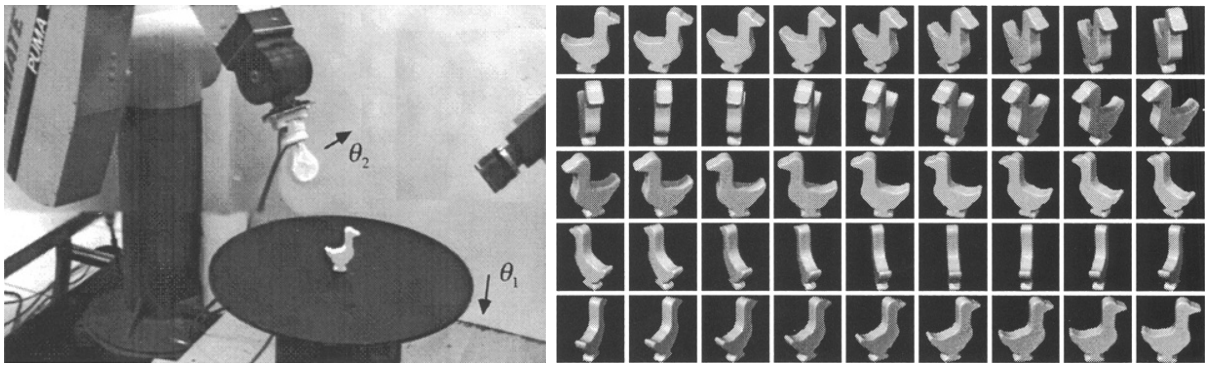

Figure 2.5. **An acquisition setup (left) for capturing template images (right) [176].** Capturing the templates in such a controlled environment allows to automatically annotate the templates with 6D object poses. Alternatively, if 3D models of the objects are available, the templates can be obtained by rendering the models, as in [94].

cascade of evaluation stages to each image region. The key stage is a voting procedure inspired by [25] and based on hashing of depth measurements and orientations of surface normals. A similar RGB-D method was concurrently proposed by Kehl et al. [129] who hash orientations of image gradients and surface normals. Hinterstoisser et al. [95, 93] represent each template by a map of color gradient orientations and, if the depth image channel is available, by a map of surface normal orientations. The orientation angles are quantized and represented as bit vectors, allowing for fast matching via binary operations. Robustness to misalignments is achieved by comparing the binarized representation with pixels in a local neighborhood. Hinterstoisser et al. [94] later made the method more efficient by matching the orientation of color gradients at only a subset of locations with large gradient magnitude, and the orientation of surface normals at only a subset of locations with locally stable normals. Rios-Cabrera et al. [212] extend the template representation of [93] by learning weights of template parts based on their discriminative power.

In general, the template matching methods tend to be sensitive to a cluttered background and partial object occlusions [176, 143]. Moreover, since a large number of templates is required to exhaustively capture possible object appearances, the application of these methods is typically limited to industrial setups with a controlled variation of the image formation conditions [112].

2.2 Learning-Based Methods

With the rise of machine learning techniques in computer vision, most 6D object pose estimation methods started to rely on image features learned by deep neural networks. Such features have been shown versatile and suitable for any type of objects, including textured and texture-less objects [99]. The methods reviewed in this section are therefore split according to the conceptual approach rather than the used features.

2.2.1 Learning to Vote for 6D Object Pose

Tejani et al. [241] adapt the template representation of [94] into a scale-invariant patch representation and train a random forest with the split function at each node defined by a random patch and a random threshold on the similarity with that patch. During the inference phase, each patch of the input image is processed by the forest, going to the left subtree of a node if the similarity with the split-function patch is below the threshold, and to the right subtree otherwise. Each tree in the forest maps a patch to a leaf that stores a set of 6D pose votes, and the pose estimates are obtained by aggregating the votes from all patches. Doumanoglou et al. [60] improve the method by replacing the representation of [94] with features calculated with an auto-encoder neural network. Kehl et al. [128] follow a similar strategy but instead of using a random forest they calculate auto-encoder features for the image patches and find the nearest neighbors in a codebook, where each entry is associated with 6D pose votes (Figure 2.6).

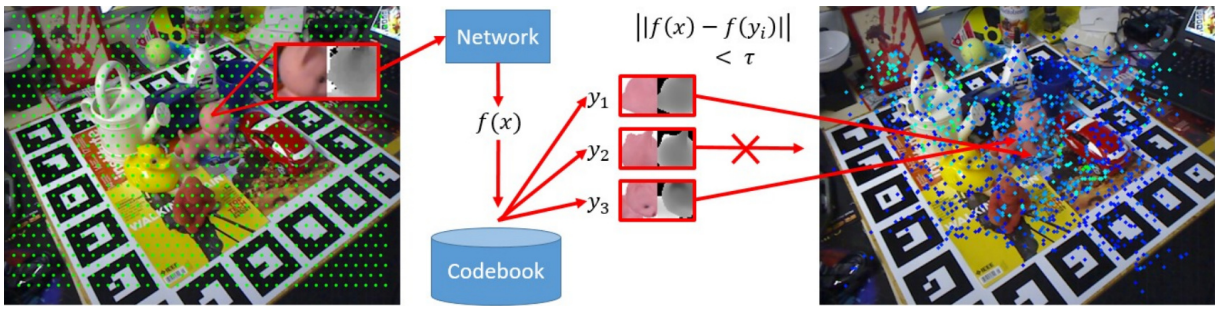

Figure 2.6. **Patch-wise 6D voting by Kehl et al. [128].** Auto-encoder features are calculated for scale-invariant patches extracted from the input RGB-D image, and the nearest features retrieved from a codebook cast votes for the 6D object pose.

2.2.2 Predicting 3D Object Coordinates

Inspired by the work of Taylor et al. [240] on human pose estimation, Brachmann et al. [17] apply a random forest to the RGB-D input image to densely predict the object identity and the 3D object coordinates, i.e. the 3D location in the coordinate frame of the object model (Figure 2.7). Split functions at the tree nodes are defined by differences in RGB and depth. A pool of pose hypotheses is generated by a RANSAC-based procedure, with each hypothesis calculated from a triplet of predicted 3D-3D correspondences by the Kabsch algorithm. The top hypotheses are iteratively refined to maximize the alignment of the predicted correspondences as well as the alignment of the observed depth with the object model, and the best hypothesis is chosen as the final pose estimate.

Brachmann et al. [18] later extended the method in three ways. Firstly, instead of a single random forest, a stack of random forests is trained using the auto-context framework [255]. Secondly, when the identities of objects visible in the input image are un-

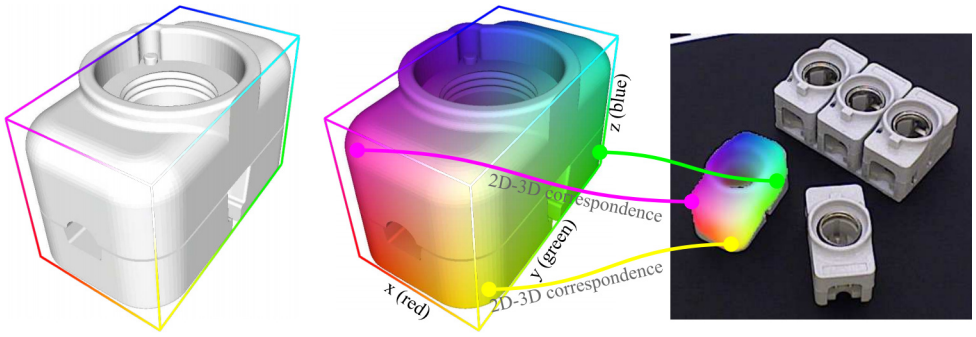

Figure 2.7. **Densely predicting 3D object coordinates [17, 188].** The 3D points on the model surface (left) are mapped to RGB (middle) for visualization. The 3D object coordinates are predicted for densely sampled pixels of the input image (right) to establish 2D-3D correspondences (or 3D-3D if the depth image channel is available). The 6D object poses are then estimated from the correspondences by a RANSAC-based procedure.

known, the RANSAC-based procedure generates not only the object pose but also the object identity. Thirdly, to represent uncertainty, the random forest predicts at each pixel a distribution over 3D object coordinates. The distribution is defined by a Gaussian mixture model, which is suitable for objects with no or a few discrete symmetries, but not for objects with continuous symmetries. A pixel from the silhouette of objects with continuous symmetries corresponds to a circle on the 3D object model, which is problematic to represent with several Gaussian distributions. The method is further extended by Michel et al. [167] who substitute the RANSAC-based procedure with a conditional random field to identify geometrically consistent clusters of predicted object coordinates.

Multiple subsequent methods focus primarily on RGB images, predict the 3D object coordinates with a neural network instead of the random forest, and estimate the object poses from 2D-3D correspondences by the PnP-RANSAC algorithm [71, 145]. Jafari et al. [121] and Nigam et al. [179] segment object instances to eliminate surrounding clutter and occluders, and predict the 3D object coordinates only for pixels in the instance masks. Zakharov et al. [277] predict the UV texture coordinates instead of the 3D object coordinates. Park et al. [188] detect the object instances with 2D bounding boxes and for each detection predict the object silhouette, the 3D object coordinates, and the expected per-pixel error of the coordinates. Additionally, they apply adversarial training to improve prediction in the occluded parts and propose a novel loss function that can handle objects with global symmetries by guiding predictions to the closest symmetric pose. Li et al. [149] estimate the 3D rotation from predicted 2D-3D correspondences and the 3D translation by directly regressing a scale-invariant 3D translation vector. Instead of the 3D object coordinates, Pitteri et al. [194] predict an embedding of local 3D geometry and match it to 3D locations on the object model that have a similar embedding vector.

2.2.3 Predicting 2D Projections of 3D Keypoints

Another approach to establish the 2D-3D correspondences is to predict the 2D projections of a fixed set of 3D keypoints, which are pre-selected for each object model (Figure 2.8). Rad and Lepetit [205] define the 3D keypoints by the corners of the 3D bounding box of the object model, for each keypoint predict a probability distribution over all pixels of a 2D object detection, and link each 3D keypoint with the maximum of the predicted distribution. Pavlakos et al. [191] define the 3D keypoints manually on the model surface. Oberweger [182] increase the robustness of the approach to partial occlusions by predicting the probability distribution from multiple small patches independently and accumulating the predictions before establishing the correspondences. Fu and Zhou [74] present a similar idea. Tremblay et al. [254] predict over the whole image nine probability distributions, one for each corner of the 3D bounding box and one for its centroid, and eight vector fields pointing from the projections of the eight corners to the projection of the corresponding centroid. The vector fields enable recovering poses of multiple instances of the same object without the 2D object detection stage. Tekin et al. [242] achieve a speed of 50 frames per second with a method inspired by the YOLO object detector [207]. They split the input image into regular cells, and for each cell predict the probability of each object's presence and regress the 2D projection coordinates of the 3D keypoints. Hu et al. [115] follow a similar strategy but use smaller cells and aggregate the predictions over the cells. Peng et al. [192] densely regress 2D unit vectors pointing to the 2D projections of the 3D keypoints and find the 2D projections via a RANSAC-based procedure. Pitteri et al. [195] focus on objects with prominent corners, detect generic 3D corners by predicting 2D projections of virtual keypoints around the corners, and estimate object poses by fitting 3D object models to the detected 3D corners. The advantage of this approach is that no re-training is necessary when adding new objects.

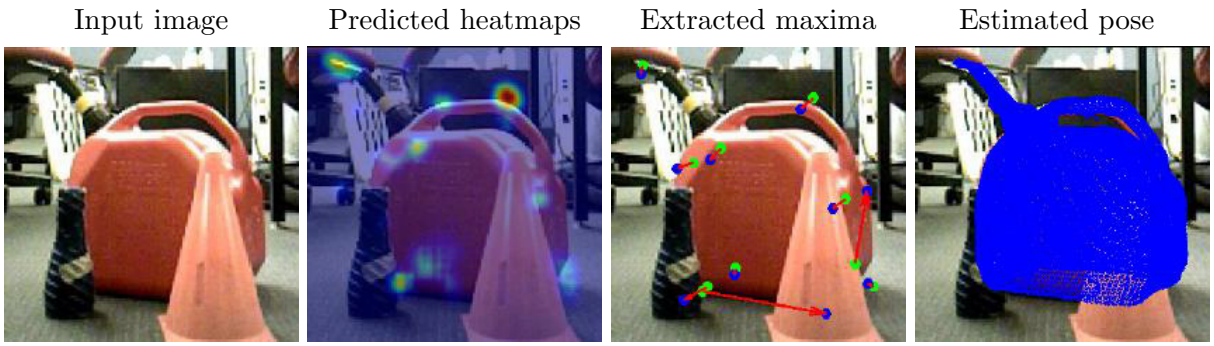

Figure 2.8. **Predicting 2D projections of 3D keypoints.** The 2D projections are usually obtained either by extracting maxima of predicted keypoint-specific heatmaps, as done in [205, 191, 182, 254] and shown in this figure, or by directly regressing the 2D coordinates, as done in [242, 115, 192]. In the third image, the green dots show the extracted maxima and the blue dots show the ground-truth projections of the 3D keypoints associated with the 3D object model.

2.2.4 Classifying Into Discrete 3D Viewpoints

This group includes methods that apply a 2D object detector such as SSD [153] or Faster R-CNN [208] to localize object instances with amodal 2D bounding boxes (covering the whole instances including the occluded parts), and classify each box into a set of discrete 3D viewpoints. The 6D object poses are calculated from the predicted 3D viewpoints and from the scale and location of the 2D bounding boxes. Because the predicted 3D viewpoints are discrete and estimating accurate amodal 2D bounding boxes is challenging [127], a post-refinement step is usually necessary to achieve a competitive performance.

Kehl, Manhardt et al. [127] extend the SSD object detector [153] to predict for each anchor box also a probability distribution over a set of discrete 3D viewpoints. Sundermeyer et al. [237], who received the best paper award at ECCV 2018, propose to calculate a global descriptor of a localized object instance by the so-called Augmented Autoencoder network. The network consists of an encoder, which maps the image region to a latent descriptor space, and a decoder, which maps the descriptor to a denoised reconstruction of the image region. The network is trained on renderings of the 3D object models that are heavily augmented by randomizing the light positions and reflectance properties, inserting random background images, varying image contrast and brightness, adding Gaussian blur and color distortions, and by creating random occlusions. At inference time, the nearest neighbor of the descriptor is retrieved from a pre-calculated codebook, where each entry is associated with a 3D orientation (Figure 2.9). Instead of training a specific encoder and decoder per object, Sundermeyer et al. [235] propose to share a single encoder among multiple objects, which dramatically improves the scalability of the method.

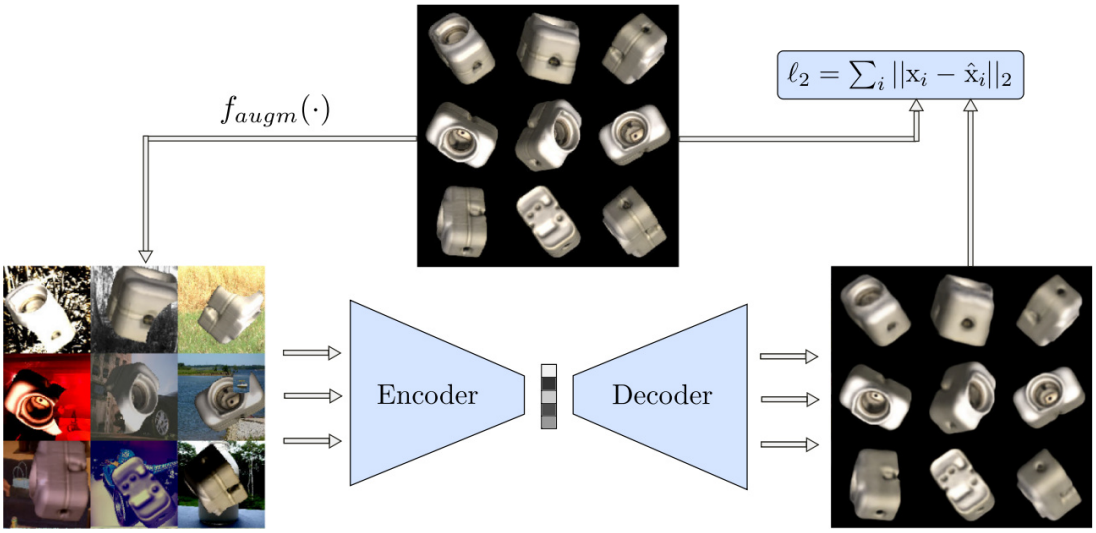

Figure 2.9. **Augmented Autoencoder by Sundermeyer et al. [237].** The network is trained to reconstruct clean object views from their augmented versions. The encoder is used to calculate descriptors of 2D object detections that are matched against a codebook.

2.2.5 Regressing 3D Orientation and 3D Location

Methods that aim to directly regress the 3D orientation and/or 3D location of the objects have also emerged. Xiang et al. [271] first densely predict object labels, unit vectors pointing to the 2D projection of the corresponding object center, and the distance of the object center from the camera. The 2D projections of object centers are found by Hough voting, and masks of object instances are estimated by clustering pixels that vote for the same object center. The 3D location of the object center is calculated from the predicted 2D projection of the center and the predicted distances from the camera, which are averaged over the instance mask. The 3D orientation is then predicted by cropping a region determined by the instance mask and regressing to a quaternion representation. Manhardt et al. [158] regress multiple quaternions for each 2D object detection to estimate the distribution of possible poses induced by object symmetries. Do et al. [57] predict the 3D orientation by regressing a Lie algebra representation. Li et al. [146] and Mahendran et al. [157] predict the 3D orientation via a mixed classification-regression scheme – a 2D object detection is classified into 3D orientation bins and refined by regressing a continuous delta w.r.t. the center of a bin. This scheme allows capturing distributions that are non-unimodal due to object symmetries and does not suffer from quantization errors. Li et al. [146] use this scheme to predict also the 3D object location. Bui et al. [24] directly regress the 3D orientation and 3D location of the object from X-ray images and show that the accuracy can be improved if the network is trained by jointly minimizing a pose error, as in [130], and a re-projection error. Wang et al. [262] aggregate deep per-pixel features over an object instance mask to obtain a global instance-level feature. The 6D object pose and its confidence are then predicted at each pixel of the instance mask from

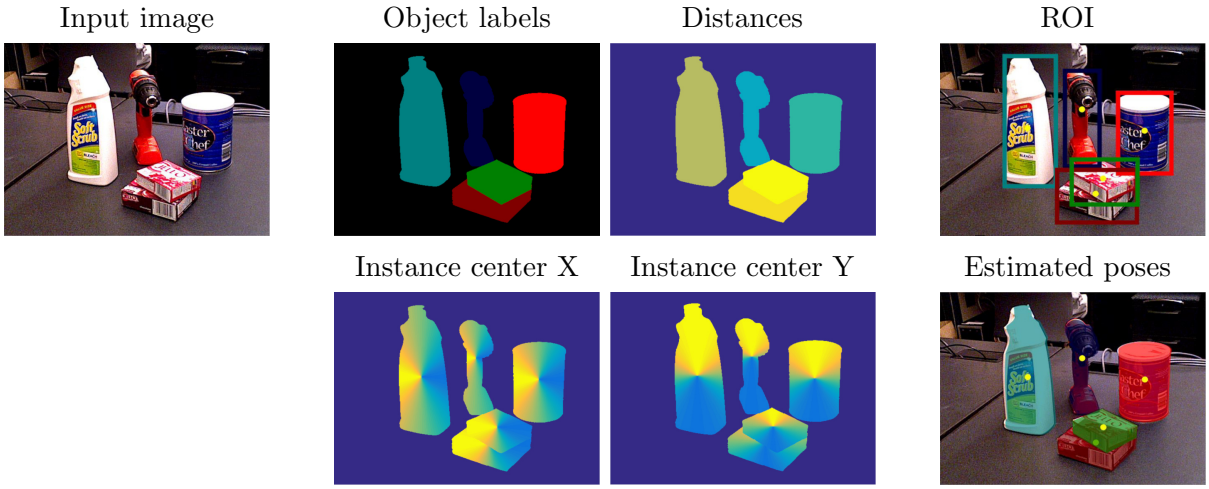

Figure 2.10. **Regressing 3D orientation and 3D location by Xiang et al. [271].** Object instances are first found by densely predicting object labels, unit vectors to the 2D projection of the object center, and the distance from the camera. The 3D orientation of an instance is then predicted by cropping the corresponding region and regressing to a quaternion representation.

a concatenation of the per-pixel features and the global feature, and the pose with the highest confidence is selected as the final estimate. Labbé et al. [139] propose a method that detects the objects by Mask R-CNN [89] and to each detection applies a neural network for coarse pose estimation followed by a neural network for iterative refinement. Both networks have the same structure inspired by DeepIM [147], but the first network is trained to predict the relative transformation w.r.t. the canonical pose and the second network is trained to predict small deltas w.r.t. the current estimate. Instead of regressing the discontinuous 4D quaternion representation, as done in [147], Labbé et al. regress a continuous 6D representation from [281] which is more suitable for training.

2.2.6 Handling the Pose Ambiguity

The object pose may be ambiguous, i.e., there may be (infinitely) many poses that are consistent with the image. The ambiguity is caused by the existence of multiple fits of the visible part of the object surface to the object model. The visible part is determined by self-occlusion and occlusion by other objects and the multiple fits are induced by global or partial object symmetries [105, 169]. See Figure 2.11 for examples of ambiguous poses.

When the object pose is ambiguous, a 2D image location potentially corresponds to multiple 3D locations on the object model, and vice versa. Such a many-to-many relationship degrades the performance of the correspondence-based methods (Sections 2.2.2 and 2.2.3) which assume a one-to-one relationship. The correspondence-based methods can be split into a classification-based and a regression-based group. The classification-based methods predict up to one corresponding 3D location for each 2D location by classifying into 3D bins [17, 179], or predict up to one 2D location for each 3D keypoint, where the 2D location is typically given by the maximum response in a predicted heatmap [191, 182, 74] (Figure 2.8). Both approaches yield a set of correspondences which carries only a limited support for each of the possible object poses. On the other hand, the regression-based methods [242, 277, 192] need to compromise among the potentially corresponding locations and tend to return the average, which is often not a valid solution. For example, the average of all points on a sphere is the center of the sphere, which is not a valid surface location. The problem is illustrated in Figure 2.12. Besides, the pose ambiguity also causes problems to methods aiming to regress the pose directly (Section 2.2.5), as these methods usually assume a unimodal pose distribution.

The problem of pose ambiguity Rad and Lepetit [205] assume that the global object symmetries are known and propose a pose normalization applicable to the case when the projection of the axis of symmetry is close to vertical. Pitteri et al. [196] introduce a pose normalization that is not limited to this special case. Kehl et al. [127] train a classifier for only a subset of viewpoints defined by global object symmetries. Corona et al. [45] show that predicting the order of rotational symmetry can improve the accuracy of pose estimation. Xiang et al. [271] optimize a loss function that is invariant to global object symmetries. Park et al. [188] and Labbé et al. [139] guide pose regression by calculating

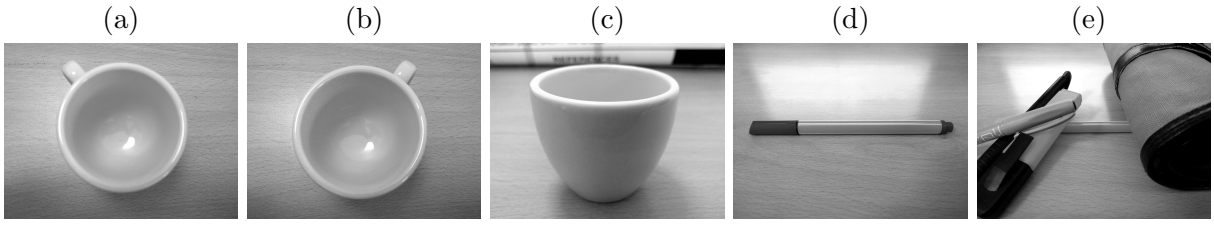

Figure 2.11. **Object pose ambiguity.** Different poses of a cup (a, b) cannot be distinguished if the handle is not visible due to self-occlusion (c). The pose of a pen (d) is ambiguous if its discriminative ends are occluded by other objects (e).

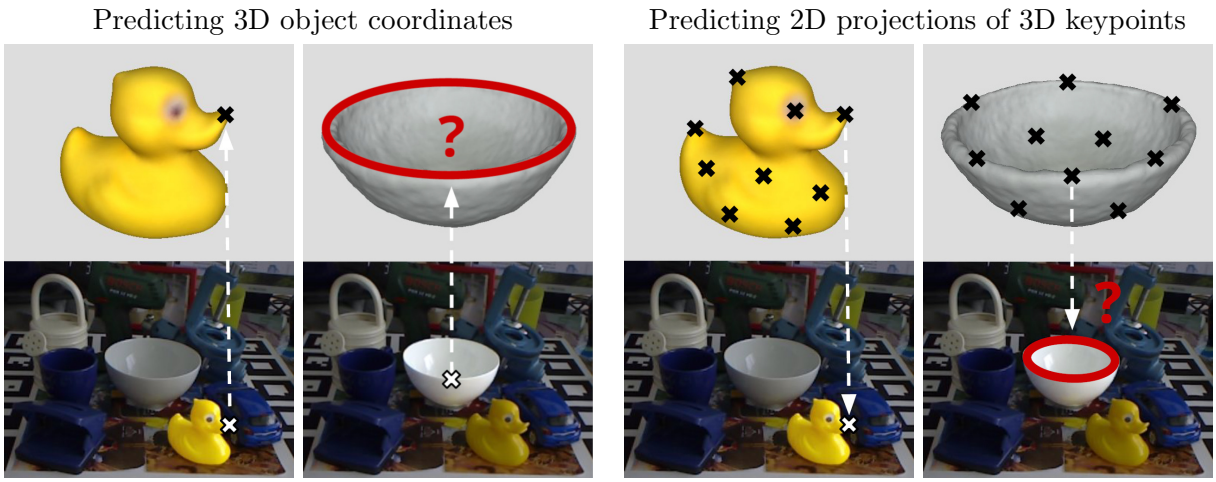

Figure 2.12. **Ambiguity of 2D-3D correspondences.** A 2D image location corresponds to a single 3D location on the object model in the case of distinct object parts, but to one of multiple indistinguishable 3D locations in the case of global or partial symmetries (left two columns). Existing correspondence-based methods (i) compromise among the possible 3D locations and tend to return the average, which is often not a valid solution, or (ii) consider only the most confident 3D location, which is also problematic as this yields a limited support for each of the possible object poses. A similar problem arises when predicting 2D projections of 3D keypoints.

the loss w.r.t. to the closest symmetric pose. However, all of these approaches cover only pose ambiguities due to global object symmetries, not due to partial object symmetries.

As the EPOS method, which we present in Chapter 4, the methods by Li et al. [146], Manhardt et al. [158], Sundermeyer et al. [237], and Ammirato et al. [6] can handle pose ambiguities due to both global and partial object symmetries without requiring any prior information about the symmetries. Li et al. [146] deals with the possibly non-unimodal pose distribution by a classification and regression scheme applied to the rotation and translation space. Manhardt et al. [158] predicts multiple poses for each object instance to estimate the distribution of possible poses induced by symmetries. Sundermeyer et al. [237] detects object instances in the input image, for each detected region calculates a descriptor, and find its nearest neighbor among pre-calculated descriptors obtained by rendering the object model from different viewpoints and under different occlusions. This

approach handles object symmetries implicitly and allows to enumerate all poses that are consistent with the image by considering all nearby descriptors [53]. Ammirato et al. [6] learn to predict object orientation via an adversarial training framework that consists of a generator and a discriminator. The generator predicts the object orientation from the input image and the discriminator evaluates the visual similarity of the input image and the object model rendered in the predicted orientation. Since the visual similarity is used as a training signal for the generator, the generator is encouraged to predict any orientation that is consistent with the input image. Nevertheless, all of these methods rely on localizing the object instances with amodal 2D bounding boxes, which is challenging when the instances are partially occluded [127]. EPOS does not rely on such localization.

2.2.7 Bridging the Domain Gap

Deep neural networks have become the standard tool for tackling many computer vision problems including object detection and object pose estimation. Besides the progress in optimization and architecture design of neural networks and the development of graphics cards that accelerate the training process, another critical factor to the success of neural networks is the availability of a large number of training images [82]. However, capturing and annotating real training images for problems such as object pose estimation requires a significant human effort or a specialized acquisition setup [102].

An alternative to acquiring real training images is synthesizing images by computer graphics. This approach scales well as it requires only a minimal human effort, which may include 3D modeling. Su et al. [234] synthesize images of 3D object models for viewpoint estimation, Hinterstoisser et al. [97] for object instance detection (Figure 2.13), and Dosovitskiy et al. [59] for optical flow estimation. They all use a fixed OpenGL pipeline and paste the rendered pixels over randomly selected real photographs. Rad et al. [205], Tekin

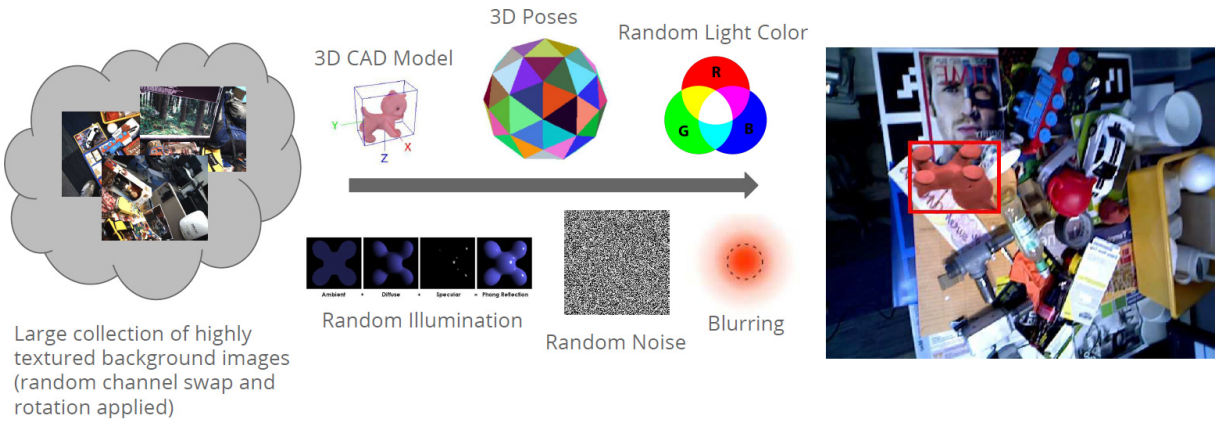

Figure 2.13. **Image synthesis pipeline by Hinterstoisser [97].** The 3D object models are rendered by OpenGL on top of randomly selected photographs, with randomized parameters of the Phong illumination model, random light color, and random image noise and blur.

et al. [242] and Dwibedi et al. [65] paste segments of objects from real images on other real images for object detection and pose estimation. Dvornik et al. [64] demonstrate the importance of selecting suitable background images, and Hinterstoisser et al. [98] achieve improvements when both the objects and the background are synthetic. While these approaches are easy to implement, the resulting images are not realistic. Objects often have inconsistent shading with respect to the background scene, interreflections and shadows are missing, and the object pose and context are not natural.

Another line of work explore rendering of complete scenes and generating corresponding ground-truth maps. Richter et al. [211, 210] leverage existing commercial game engines to acquire training data for several tasks. Synthia [215] and Virtual KITTI [75] were generated using virtual cities modeled from scratch. Handa et al. [88] and Zhang et al. [278] model 3D scenes for semantic scene understanding and McCormac et al. [163] synthesize RGB-D video frames from simulated cameras moving within static scenes.

Despite training neural networks on massive datasets of diverse synthetic images, a large drop in performance was observed when models trained only on synthetic images were tested on real images [211, 234, 216]. The domain gap between the synthetic and real images can be reduced by domain adaptation techniques, which aim to learn domain invariant representations or to transfer trained models from one domain to another [48]. Besides, the issue can be mitigated by domain randomization techniques, which randomize rendering parameters and were shown beneficial for training object detection and pose estimation models [245, 253, 237, 276] (Figure 2.14).

A different line of work, presumably complementary to the domain adaptation and randomization techniques, has recently tried to reduce the domain gap by synthesizing training images with a higher degree of visual realism. The use of physically-based rendering has been considered with this motivation and shown promising results [148, 278].

Physically-based rendering (PBR) techniques, e.g., Arnold [80], accurately simulate the flow of light energy in the scene by ray tracing. This approach naturally accounts for complex illumination effects such as scattering, refraction and reflection, including diffuse

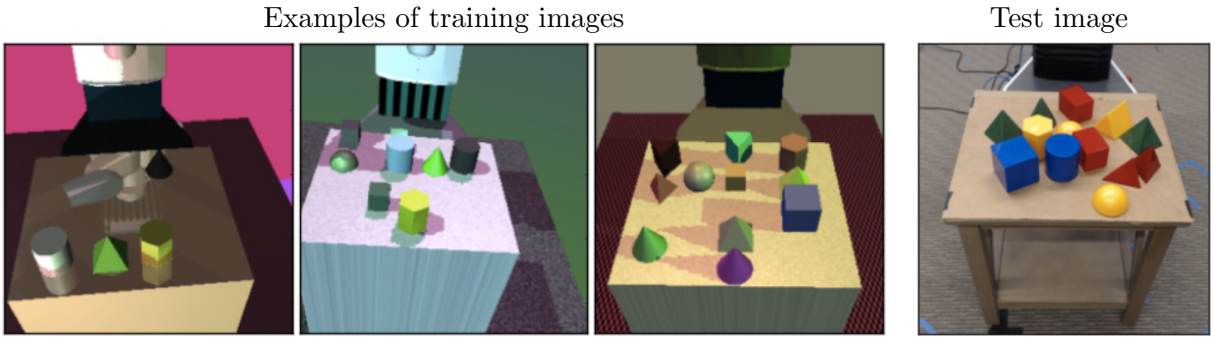

Figure 2.14. **Domain randomization by Tobin et al. [245].** An object detector is trained on hundreds of thousands of low-fidelity rendered images with randomized camera/object positions, textures, and lighting conditions. At test time, the detector is applied to real images.

and specular interreflection between the objects and the scene and between the objects themselves. The rendered images look realistic and are often difficult to differentiate from real photographs [193]. Rendering techniques based on rasterization, e.g., OpenGL [225], can approximate the complex effects in an ad hoc way through custom shaders, but the approximations cause physically incorrect artifacts that are difficult to eliminate [160]. Physically-based rendering has been noticeably slower than rasterization, however, the recently introduced Nvidia RTX ray tracing GPU promises a substantial reduction of the rendering time. Moreover, the rendering time can be reduced by tracing fewer rays per pixel and applying a denoiser to effectively eliminate noise in the rendered image [118].

Li and Snavely [148] use PBR images to train models for intrinsic image decomposition and Zhang et al. [278] for semantic segmentation, normal estimation and boundary detection. Wood et al. [268] use PBR images of eyes for training gaze estimation models. Attias et al. [172] render photorealistic images of 3D car models placed within 3D scene models and show the benefit over naive rendering methods, whereas Tremblay et al. [254] render objects in physically plausible poses in diverse scenes, but do not use physically-based rendering. Other ways to generate photorealistic images [267, 226] and methods to synthesize realistic depth images [197] have also been proposed.

In Chapter 5, we present an approach to synthesize PBR images of 3D object models arranged in physically plausible poses inside 3D scene models, and show that the images can be used to effectively train networks for object detection or object pose estimation.

2.2.8 Using the Depth Image Channel

While most existing methods for 6D object pose estimation that are based on neural networks apply the networks only to the RGB channels, promising methods using the depth channel as an additional input start to emerge. Wang et al. [262] segment objects in RGB channels, in each mask calculate per-pixel color features by an auto-encoder network and per-pixel depth features by PointNet [202], fuse the two types of features, and regress object poses as described in Section 2.2.5. He et al. [91] fuse color and depth features (the latter are calculated by PointNet++ [203]) and regress 3D coordinates of model keypoints. Similarly, Qi and Chen [201] use fused features to vote for 3D object centers and predict 3D bounding boxes. Hagelskjær and Buch [87] process a color point cloud with PointNet to find candidate object locations and to establish 3D-3D correspondences at each such location. Chen et al. [32] train PointNet to predict the mask and translation of an object instance from the depth channel. Li et al. [146] process the RGB and depth channels with two convolutional branches and fuse them before predicting the object pose via a mixed classification-regression scheme (Section 2.2.5). Sock et al. [230] process the depth channel with a convolutional neural network that is trained jointly for 2D detection, 6D pose estimation, and hypotheses verification. Félix and Neves [214, 206] and König and Drost [135] propose hybrid methods which segment object instances using a neural network and estimate the object pose from each mask using the point pair features.

2.3 Refinement Methods

The accuracy of pose estimates produced by any of the methods reviewed above can be often improved by methods specialized on estimating small refinement transformations. The refinement is crucial especially for template-matching methods (Sections 2.1.5) and methods predicting the pose by classification into discrete 3D viewpoints (Sections 2.2.4).

The Iterative Closest Point (ICP) algorithm [13] is the golden standard for pose refinement from point clouds. ICP aims to find an aligning transformation between two sets of points by alternating between establishing putative correspondences between the closest points and estimating a transformation that aligns them. When refining object poses, the source point cloud is defined by vertices of the 3D object model in the estimated pose or calculated from rendered depth image of the model. The target point cloud is calculated from the depth channel of the input image. ICP converges to the correct transformation when the source point cloud is initially sufficiently close to the target point cloud. However, when the two point sets are relatively far apart or have a small overlap, the strategy of matching closest points generates large numbers of incorrect correspondences. The situation is aggravated by the inevitable presence of noise and outliers. Consequently, ICP can easily get stuck in local minima and its performance largely depends on the quality of initialization. To counter this, numerous enhancements to the basic ICP have been proposed that aim to improve the speed of convergence or increase the robustness to local minima, outlying points, and noise [217]. These enhancements often require considerable trial and error for tuning their parameters. ICP can be applied also to the intensity edges to align the projection of the object model to the image content [279, 49].

An alternative to ICP is Particle Swarm Optimization (PSO) [198, 184, 120], which stochastically evolves a population of candidate pose estimates over multiple iterations

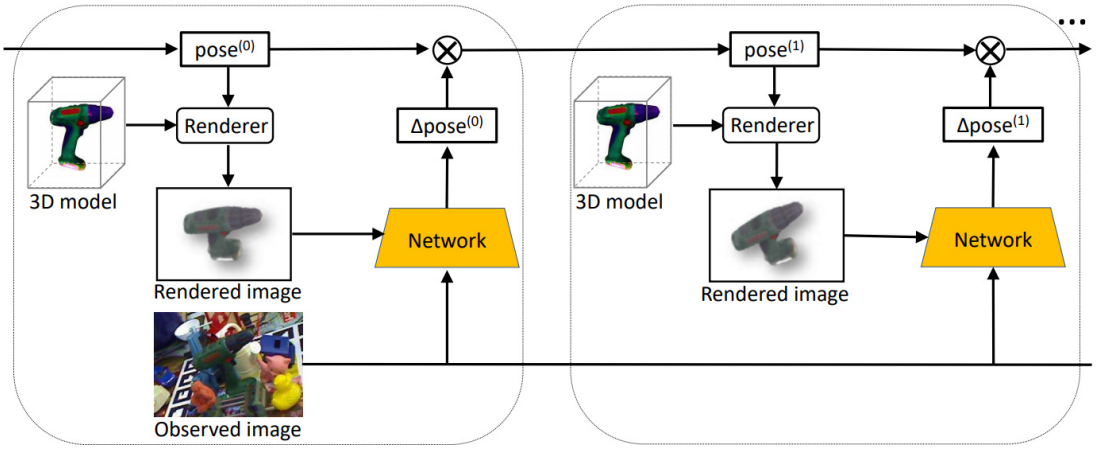

Figure 2.15. **Deep iterative refinement by Li et al. [147].** A neural network is trained to predict a relative rigid transformation from a cropped region of the input RGB image and a rendering of the 3D object model in the current pose estimate.

and was demonstrated to be less prone to local minima than ICP [275]. PSO is applied for post-refinement in the template-matching method proposed in Chapter 3.

Learning-based refinement methods have emerged as well. Oberweger et al. [183] use a neural nework to iteratively refine a hand pose estimate. The network predicts a pose update from the input image and a rendered image of the hand in the current pose. The update is applied to the current pose and the process is repeated. A conceptually similar procedure was used to refine object pose estimates by Rad and Lepetit [205], Li et al. [147] (Figure 2.15), Manhardt and Kehl et al. [159], and Wang et al. [262]. Bauer et al. [10] additionally integrate physics simulation to achieve physically plausible pose estimates.

2.4 Datasets

This section reviews datasets for estimating the 6D pose of specific rigid objects, grouped by the type of included images, and also mentions datasets for similar problems. If not stated otherwise, the datasets include ground truth in the form of 6D object poses.

2.4.1 RGB-D Datasets

Most datasets used for research in object pose estimation are captured with consumer-grade RGB-D sensors which became widely available with the launch of Microsoft Kinect in 2010. The first generation of Kinect is based on the structured-light principle – a light speckle pattern is projected onto the scene using a near-infrared laser emitter and the light reflected back to a standard off-the-shelf infrared camera is analyzed to estimate the depth of the scene surfaces [131, 136]. Although the exact details are not publicly available as the technology is patented, it is known that the sensor relies upon established computer vision techniques such as depth from focus and depth from stereo. In 2014, the second generation of Kinect was released. This completely new sensor is based on the time-of-flight principle in which the depth of a scene is measured by the absolute time needed by a light wave to travel into the scene and, after reflection, back to the sensor.

For the evaluation of methods based on the 3D local features, Aldoma et al. [2] created a dataset with 3D mesh models without color of 35 household objects. The objects are often symmetric and mutually similar in shape and size. The dataset includes 50 test RGB-D images of table-top scenes with multiple objects in single instances, with no clutter and various levels of occlusion. Several small datasets that were used for evaluation of the SHOT descriptor are provided by Salti et al. [219]. These datasets include synthetic data as well as data acquired with a spacetime-stereo method and an RGB-D sensor.

The Challenge and Willow datasets [273], which were collected for the 2011 ICRA Solutions in Perception Challenge, share a set of 35 textured household objects. Training data for each object is given in the form of 37 RGB-D training images that show the object from different viewpoints, plus a color point cloud obtained by merging the train-

ing images. The Challenge and Willow datasets respectively contain 176 and 353 test RGB-D images of several objects in single instances placed on top of a turntable. The Willow dataset also features distractor objects and object occlusion. Similar is the TUW dataset [1] that presents 17 textured and texture-less objects shown in 224 test RGB-D images. Instead of a turntable setup, images were obtained by moving a robot around a static cluttered environment with some objects appearing in multiple instances. The BigBIRD dataset [228] includes images of 125 mostly textured objects that were captured in isolation on a turntable with multiple calibrated RGB-D and DSLR sensors, with fixed lighting and object-sensor distance, and with no occlusions or clutter. For each object, the dataset provides 600 RGB-D point clouds, 600 high-resolution RGB images, and a color 3D mesh model reconstructed from the point clouds. Georgakis et al. [79] provide 6735 test RGB-D images from kitchen scenes including a subset of the BigBIRD objects, with the ground truth in the form of 2D bounding boxes and 3D point labeling. Lai et al. [140] created an extensive dataset with 250K segmented RGB-D images of 300 household objects captured on a turntable from three elevations. The dataset also contains 22 test video sequences with a few hundred RGB-D frames in each showing the objects in household scenes. The ground truth is provided only in the form of approximate rotation angles for training images and in the form of 3D point labeling for test images. Schlette et al. [222] synthesized RGB-D images from simulated object manipulation scenarios involving 4 texture-less objects from the Cranfield assembly benchmark [43].

The RGB-D datasets by Hinterstoisser et al. [94], Brachmann et al. [17], Tejani et al. [241], Doumanoglou et al. [60], Rennie et al. [209], Drost et al. [62], Xiang et al. [271], and Kaskman et al. [126] are included in the BOP benchmark and reviewed in Section 7.2. The T-LESS dataset, which features industry-relevant texture-less objects with symmetries and mutual similarities and is also included in BOP, is presented in Chapter 6.

2.4.2 Depth-Only and RGB-Only Datasets

The depth-only dataset of Mian et al. [166] includes 3D mesh models of 5 objects and 50 test depth images from an industrial range scanner which show the modeled objects occluding each other. A similar dataset is provided by Taati et al. [239]. The Desk3D dataset by Bonde et al. [16] includes 3D mesh models of 6 objects captured in over 850 test depth images with occlusion, clutter and similarly looking distractor objects. The images were taken with an RGB-D sensor but only the depth images are publicly available.

The IKEA dataset by Lim et al. [150] includes RGB images of objects that are aligned with their 3D models. Crivellaro et al. [46] supply 3D CAD models and annotated RGB sequences with 3 highly occluded and texture-less objects. Muñoz et al. [174] provide RGB sequences of 6 texture-less objects captured in isolation against a clean background and without occlusion. Further to the above, there exist RGB datasets such as [49, 248, 212, 114] for which the ground truth is provided only in the form of 2D bounding boxes.

2.4.3 Datasets for Similar Problems

The RGB-D dataset of Michel et al. [168] is focused on articulated objects with the goal to estimate the 6D pose of each object part, subject to the constraints introduced by their joints. There are also datasets for categorical pose estimation. For example, the 3DNet [266] and the UoB-HOOC [261] contain generic 3D models and RGB-D images annotated with 6D object poses. The UBC VRS [164], the RMRC (a subset of NYU Depth v2 [227] with annotations derived from [83]), the B3DO [122], and the SUN RGB-D [231] provide no 3D models and ground truth only in the form of bounding boxes. The PASCAL3D+ [270] and the ObjectNet3D [272] provide generic 3D models and RGB images annotated with ground-truth 6D poses. A comprehensive overview of RGB-D datasets for various computer vision tasks is provided by Firman [70].

2.5 Evaluation Methodologies

This section reviews common approaches to evaluate 6D object pose estimation, including the problem formulation and definition of functions measuring error of the pose estimates.

2.5.1 6D Object Pose Estimation Problems

Methods for 6D object pose estimation usually report their predictions on the basis of two sources of information. Firstly, at training time, a method is given 3D object models and/or training images showing the objects in known poses. Secondly, at test time, the method is provided with an input image and possibly a list of object instances visible in the image. If the list is provided, the problem is referred to as 6D object localization and the evaluated method reports pose estimates of the listed instances. Otherwise, the problem is referred to as 6D object detection and the method reports pose estimates of instances which the method detects. In both cases, the method also reports confidences of the estimates. The intrinsic camera parameters are assumed known.

In the 6D object localization problem, the performance of a method is measured by the recall rate, i.e., the fraction of annotated object instances for which a correct pose was estimated (commonly used criteria of pose correctness are described in Section 2.5.2). In the 6D object detection problem, the performance is typically measured by scores based on the precision and recall, such as the F1-score [241].

The aspect which is evaluated on the 6D object detection but not on the 6D object localization problem is the capability of the method to calibrate the predicted confidence scores across all object models, i.e., whether the same score represents the same level of confidence no matter what object model the pose is estimated for. This calibration is important for achieving a good performance w.r.t. to both the precision and recall. The 6D object localization problem still requires the method to sort the pose estimates, although only within the set of pose estimates associated with the same object model – the

method needs to output the top n pose estimates for a given object model which are then evaluated against n ground-truth poses associated with that model.

In the literature on 6D object pose estimation, methods are mostly evaluated on the 6D object localization problem, mainly because the accuracy scores on this simpler problem are still far from being saturated (the current state of the art is reported in Section 7.4).

2.5.2 Measuring Pose Error

The object pose is defined by a rigid transformation from the 3D coordinate system of the model to the 3D coordinate system of the camera and represented by a 3×4 matrix $\mathbf{P} = [\mathbf{R} \,|\, \mathbf{t}]$, where $\mathbf{R}$ is a 3×3 rotation matrix and $\mathbf{t}$ is a 3×1 translation vector.

Evaluating object pose estimates is not straightforward. This is because each object instance visible in an image is annotated with a single ground-truth pose in all commonly used datasets, but the object pose may be ambiguous and there may be (infinitely) many poses consistent with the image (Section 2.2.6). The indistinguishable poses should be treated as equivalent, but explicitly enumerating all of them is difficult as one would need to identify the visible object parts in each image and find their fits to the object model.

In several papers, e.g., [63, 36, 18], the error of an estimate $\hat{\mathbf{P}} = [\hat{\mathbf{R}} \,|\, \hat{\mathbf{t}}]$ w.r.t. the ground-truth $\bar{\mathbf{P}} = [\bar{\mathbf{R}} \,|\, \bar{\mathbf{t}}]$ is measured by the translational and rotational errors (the latter is defined by the angle from the axis-angle representation of the relative rotation [170]):

$$e_{\text{TE}}\big(\hat{\mathbf{t}}, \bar{\mathbf{t}}\big) = \big\|\bar{\mathbf{t}} - \hat{\mathbf{t}}\big\|_2, \;\; e_{\text{RE}}\big(\hat{\mathbf{R}}, \bar{\mathbf{R}}\big) = \arccos\Big(\big(\text{Tr}\big(\hat{\mathbf{R}}\bar{\mathbf{R}}^{-1}\big) - 1\big) \,/\, 2\Big) \tag{2.1}$$

The estimated pose $\hat{\mathbf{P}}$ is considered correct if both e_{TE} and e_{RE} are below a respective threshold. Choi and Christensen [36] set the thresholds to 1 cm and 10°, Brachmann et al. [18] to 5 cm and 5°, and Drost et al. [63] to $0.1d$ cm and 12°, where d is the object diameter, i.e., the largest distance between any pair of points on the model surface.

The rotational error could be extended to take into account the axes of symmetry, which can be identified as in Section 7.1.5. The downside of measuring the error directly in rotation is that even a small deviation in rotation can yield a large misalignment of the object surface in the case of elongated objects or objects with a complex surface. The rotational error is therefore less relevant for applications such as robotic grasping or augmented reality, where the surface alignment is the main indicator of the pose quality.

The most widely used pose-error function has been the Average Distance (AD) by Hinterstoisser et al. [94], defined as the average distance from vertices V_M of the object model M in the ground-truth pose $\bar{\mathbf{P}}$ to the vertices in the estimated pose $\hat{\mathbf{P}}$. If all views at the object are distinguishable, i.e., the object looks different from every viewpoint, the distance is measured between the corresponding vertices (in homogeneous coordinates):

$$e_{\text{ADD}}\big(\hat{\mathbf{P}}, \bar{\mathbf{P}}, V_M\big) = \frac{1}{|V_M|} \sum_{\mathbf{x} \in V_M} \big\|\bar{\mathbf{P}}\mathbf{x} - \hat{\mathbf{P}}\mathbf{x}\big\|_2 \tag{2.2}$$

Otherwise, for an object with indistinguishable views (i.e., the object looks the same from two or more viewpoints), the distance is measured to the closest vertex, which may not necessarily be the corresponding vertex:

$$e_{\text{ADI}}\big(\hat{\mathbf{P}}, \bar{\mathbf{P}}, V_M\big) = \frac{1}{|V_M|} \sum_{\mathbf{x_1} \in V_M} \min_{\mathbf{x}_2 \in V_M} \left\| \bar{\mathbf{P}}\mathbf{x}_1 - \hat{\mathbf{P}}\mathbf{x}_2 \right\|_2 \tag{2.3}$$

The estimated pose $\hat{\mathbf{P}}$ is usually considered correct if $e \leq 0.1d$, where e is e_{ADD} or e_{ADI}, and d is the object diameter defined as above.

The downside of ADD and ADI is their high dependence on the geometry of the object model and on the sampling density of its surface – the average distance tends to be dominated by higher-frequency surface parts. Besides, the existence of indistinguishable object views does not imply that the whole object is symmetric, as parts that break the symmetry may be occluded. ADI penalizes misalignment of such invisible parts, which may not be desirable for applications such as robotic manipulation with suction cups where only alignment of the visible part is relevant. For example, if (c) in Figure 2.11 was the input image showing a cup, and (a) and (b) were the top views at the cup in the ground-truth and the estimated pose, ADI would yield a non-zero error because the invisible handle has a different position in the two poses. Another example of invisible parts whose misalignment would be penalized are inner parts of the object models, which are often included in CAD models used in industry. Moreover, the ADI variant may yield unintuitively low errors for poses that are clearly distinguishable due to a many-to-one vertex matching that may be established by the search for the closest vertex, and due to the possibility of matching the outer and the inner model parts – see Section 7.1.2 for examples.

In Section 7.1, we propose a new methodology which is used in BOP and includes three pose-error functions that address the downsides of the functions reviewed above.

Chapter 3

HashMatch

Hashing for Efficient Template Matching

The method presented in this chapter, referred to as HashMatch, aims at accurate 6D pose estimation of texture-less objects with available 3D models from a single RGB-D image. The method slides a window of a fixed size over the input image in multiple scales and searches for a match against a set of object templates. The templates are pre-generated by rendering isolated 3D object models in different orientations and annotated with 6D object poses. Instead of matching each window location to all templates, HashMatch applies an efficient cascade of evaluation stages to each window location, which makes the computational complexity of the method sub-linear in the number of templates.

The evaluation cascade starts with fast filtering that rapidly rejects most window locations by a simple objectness check based on the number of discontinuities in the depth image channel. For each remaining window location, a set of candidate templates is retrieved by an efficient voting procedure based on hashing of depth measurements and surface normals, which are sampled on a regular grid. The candidate templates are then verified by a sequence of tests evaluating the consistency of color, depth, image gradients, and surface normals. Finally, the approximate 6D object pose associated with each detected template is used as a starting point for a stochastic, population-based optimization procedure that refines the pose by fitting the 3D object model to the depth image channel.

The pipeline of HashMatch is illustrated in Figures 3.1 and 3.2 together with the typical numbers of detection candidates advancing through individual stages. The key is the voting stage based on hashing, which substantially reduces the number of candidates, usually by three orders of magnitude, before the more expensive template verification stage. The object detection part of the method is detailed in Section 3.1 and the pose refinement part in Section 3.2. An experimental evaluation is provided in Section 3.3, demonstrating the proposed method to achieve accuracy comparable to the state of the art, while improving the computational complexity w.r.t. the number of templates.

The HashMatch method was published in [112], used to report baseline results on T-LESS (Chapter 6), placed fourth out of the 15 participants in the BOP Challenge 2017 (Section 7.3), and was successfully deployed in a robotic assembly project (Section 3.3.2).

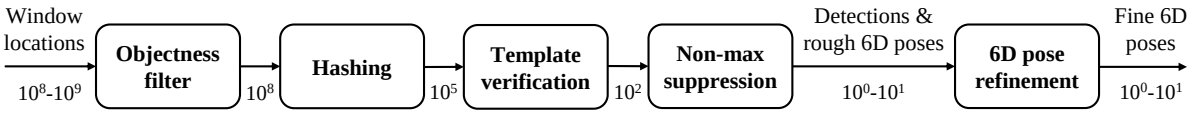

Figure 3.1. **HashMatch pipeline.** The method applies an cascade-style evaluation to each sliding window location. Note the typical numbers of detection candidates, each defined by a template identifier and a window location, advancing through individual stages (for a 5 px sliding step, 108×108 px templates, a VGA input image, and an image pyramid with four larger and four smaller scales with a scaling factor of 1.2). The hashing stage efficiently reduces the number of candidates usually by three orders of magnitude before the more expensive verification stage.

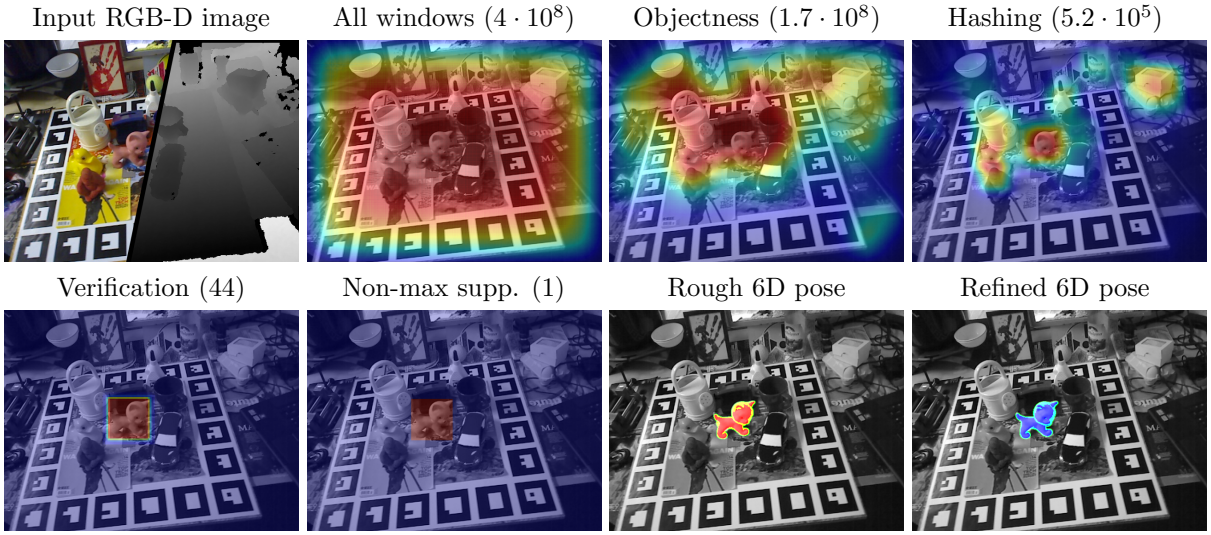

Figure 3.2. **6D localization of a cat in a sample RGB-D image from [94].** The numbers of detection candidates advancing through the pipeline are in brackets. The heatmaps show the density – red indicates a high and blue indicates a low number of candidates overlapping a pixel.

3.1 Object Detection by Template Matching

Detection of objects in the input RGB-D image is based on a sliding window approach operating on an image pyramid constructed from the input image. Let W denote a set of all tested window locations. Each window location is defined by a triplet $w = (u_w, v_w, s_w)$, where (u_w, v_w) are the 2D image coordinates of the center of the window and s_w is the image scale. The window is a square with its size (in pixels) fixed across all image scales. The number of tested window locations $|W|$ is a function of the image resolution, sliding step, size of the sliding window, scale range (e.g., two or four octaves), and scale-space discretization. The objects are represented with a set T of RGB-D templates which are of the same fixed size as the sliding window. For each object, the set T typically contains several thousands of templates showing the object at the same distance but in different 3D orientations (Figure 3.3). The distance is defined between the camera center and the

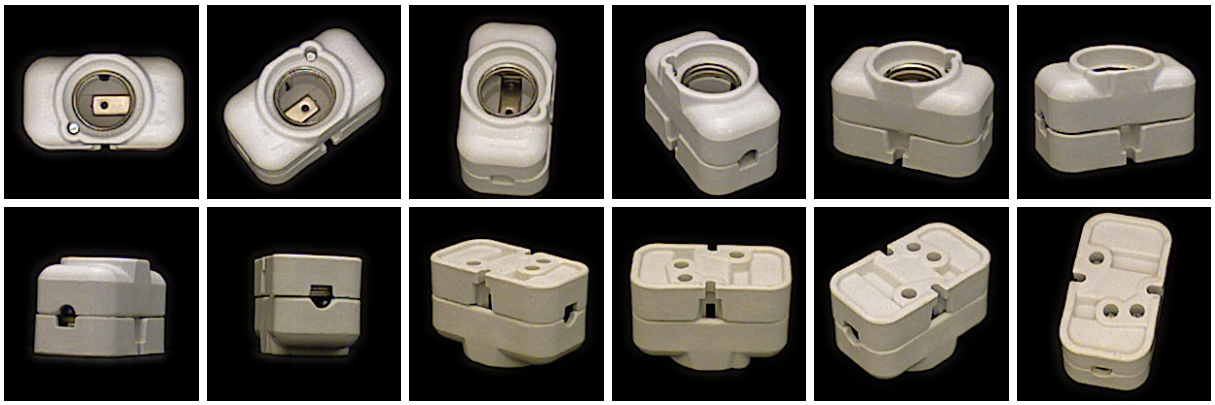

Figure 3.3. **RGB channels of sample templates from [102].** Each object is represented by multiple (usually several thousands) templates showing the object under different 3D orientations (i.e., under different azimuth, elevation and in-plane rotation angles).

object centroid[1], and chosen so that the object silhouette from all viewing angles fits the template. A template $t \in T$ is associated with the object identifier o_t and the object pose given by a rotation matrix $\mathbf{R}_t$ and a translation vector $\mathbf{t}_t$. Templates can be obtained by rendering the 3D models or by capturing the objects using a specialized setup [102].

In general, each window location $w \in W$ needs to be compared against all templates, with the asymptotic computational complexity being $O(|W||T|)$. This is expensive even for a moderate number of objects. We therefore propose an evaluation cascade where the locations W are quickly reduced (Section 3.1.1) and the candidate templates T are substantially pruned (Section 3.1.2) before the template matching itself (Section 3.1.3).

3.1.1 Objectness Filter

Each window location w is first assessed with a simple *objectness* measure, i.e., the likelihood that the location contains an object [4, 34]. This is a binary classification problem of labeling w as a background or a potential foreground, with the foreground defined by T.

The proposed objectness measure is based on the number of depth-discontinuity edgels inside the window and is computed via an integral image for efficiency. Depth-discontinuity edgels arise at pixels where the response of the Sobel operator computed over the depth channel is larger than a threshold τ, which is in the same units as the depth channel. A window location is classified as a potential foreground if the number of depth edgels is larger than a threshold θ. In the experiments presented in Section 3.3, the threshold τ is set to 30% of the diameter[2] of the smallest object in the database, and θ is set to 30% of the minimum number of depth edgels present in any template. This setting is tolerant to partial occlusions but still strong enough to prune most of the

[1]The object centroid is defined by the center of the 3D bounding box of the object model and is aligned with the origin of the 3D coordinate system of the object model.

[2]The object diameter is defined as the largest distance between any pair of points on the model surface.

window locations – depending on the scene clutter, roughly 60% to 90% window locations are pruned in images of the considered robot workspace depicted in Figure 3.8. Only window locations classified as a potential foreground are processed further.

3.1.2 Hypothesis Generation by Hashing-Based Voting

In this stage, a small subset of potentially matching templates is quickly identified for each window location w that passed the objectness filter. The set of candidate templates is found using multiple hash tables H, which is an operation with a constant computational complexity in the number of stored templates $|T|$. Each hash table $h \in H$ is indexed by a hash key obtained by discretizing a feature vector F_h, which is extracted from a window location w or a template t. Multiple hash tables are constructed to increase the robustness, each table with a different definition of the associated feature vector F_h. At training time, the hash tables are filled with identifiers of templates, with possibly multiple but typically a low number of identifiers stored at each hash key. At test time, the hash tables are used to vote for the potentially matching templates. A template t can receive up to $|H|$ votes, in which case all of the feature vectors from t are discretized into the same hash keys as the feature vectors from w. Up to n templates with the highest number of votes, which received at least m votes, are passed onward to the next stage of the cascade ($n = 100$ and $m = 3$ in the experiments presented in Section 3.3).

Feature Extraction and Quantization. A feature vector F_h associated with a hash table h is defined on a regular grid of $r \times r$ reference points placed over the templates at training time and over the sliding window at test time. The vector consists of $k-1$ depth differences and k 3D normal vectors measured at k selected reference points (Figure 3.4): $F_h = (d_2 - d_1, d_3 - d_1, \ldots, d_k - d_1, \mathbf{n}_1, \ldots, \mathbf{n}_k)$. The depth differences $d_i - d_1$ are quantized into b_d bins, with the quantization boundaries learned from all templates to achieve equal frequency binning, i.e., each bin contains approximately the same number of examples. The normal vectors are quantized as in [93] into b_n bins according to their orientation. In the presented experiments, we set $r = 12$, $b_d = 5$, $b_n = 8$, and select $k = 3$ reference points as described below. This setting yields a hash table with $5^2 8^3 = 12800$ indices.

Selection of Reference Points. To increase the robustness against occlusion and noise, multiple hash tables are constructed and the feature vector F_h is measured at different reference points for each table h. The k reference points associated with a table can be selected randomly. Alternatively, the selection can be optimized to fill the tables uniformly (for stable detection time) and to cover maximally independent measurements (for robustness). Since the optimal selection is NP-complete, a hybrid strategy is employed – a set of q, where $q \gg |H|$, k-tuples is generated and the subset with the largest joint entropy of the quantized measurements is retained ($q = 5000$ and $|H| = 100$ in the experiments).

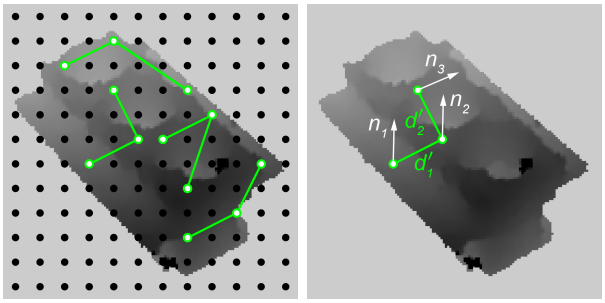

Figure 3.4. **Feature extraction.** A grid of reference points is placed over the templates at training time and the sliding window at test time. A feature vector F_h is defined by depth differences $\{d'_1, d'_2\}$ and normal vectors $\{\mathbf{n}_1, \mathbf{n}_2, \mathbf{n}_3\}$ measured at three reference points associated with the hash table h. The feature vector is discretized and used to index the hash table h – to insert a template at training time and retrieve potentially matching templates at test time. The feature vector is valid only when the depth value is available at all three reference points.

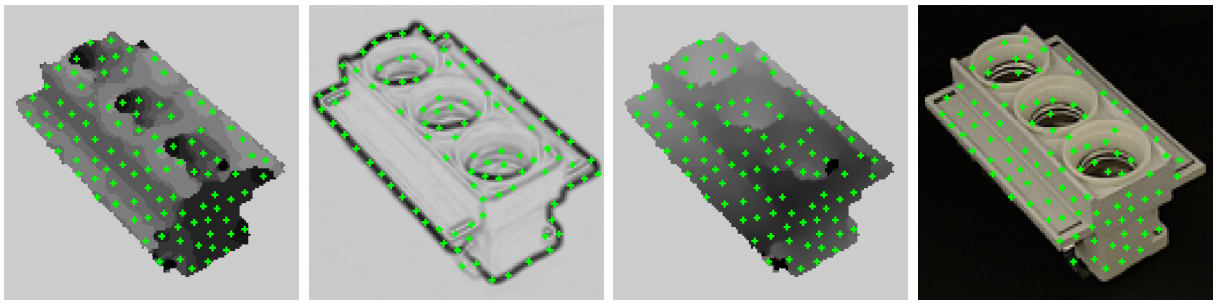

Figure 3.5. **Multimodal hypothesis verification.** The consistency between a sliding window location and a template is evaluated in different modalities at pixels that are selected independently for each template. Left to right: surface normals, image gradients, depth, color.

The hashing-based voting procedure is inspired by Cai et al. [25], who assume a grayscale input image and hash distances and orientations of intensity edges instead of depth and surface normals, and similar to the concurrent work of Kehl et al. [129], who assume an RGB-D image and hash orientations of image gradients and surface normals.

3.1.3 Multimodal Hypothesis Verification

Thanks to the preceding hypothesis generation stage, only up to n templates are considered for each window location w that passed the initial objectness filter, which makes the complexity of the verification stage constant in the number of stored templates $|T|$. The verification proceeds in a sequence of tests evaluating the object size (T_1), surface normals (T_2), image gradients (T_3), surface distance (T_4), and surface color (T_5). The tests are ordered by increasing computational cost. If a test fails, the next ones are skipped.

Test T_1 verifies that the scale of the detected object o_t (i.e., the level of the image pyramid at which the object is detected) is in accordance with the depth image channel.

The surface of the object o_t as seen in the template t detected at the window location w is expected to be approximately at distance $e = s_w\|\mathbf{t}_t\|_2 - 0.5d_t$, where s_w is the image scale associated with the window w, $\|\mathbf{t}_t\|_2$ is the Euclidean distance between the camera center and the object centroid, and d_t is the object diameter. The expected distance e is compared with distances calculated at five reference points – the center of the sliding window and the centers of its four quadrants.[3] The template t passes test T_1 if the distance at one or more of the five reference points is between $e/\sqrt{s}$ and $e\sqrt{s}$, where s is the scale factor between two consecutive levels of the image pyramid.

Tests T_2 and T_3 verify the orientation of the surface normals and of the intensity gradients at p selected pixels ($p = 100$ in our experiments; Figure 3.5). Similarly to [94], the pixels are selected independently for each template at training time. Pixels for test T_2 are selected at locations with locally stable orientation of surface normals. Pixels for test T_3 are selected at locations with large gradient magnitude (i.e., typically on the contour in the case of texture-less objects). The orientation of surface normals and intensity gradients is quantized and compared between the template and the window, which can be done efficiently by bitwise operations using the response maps described in [95].

The distance test T_4 and the color test T_5 reuse pixels selected for the surface normal test T_2. In test T_4, a difference δ between the distance measured in the template and the distance measured in the sliding window is calculated at each tested pixel.[3] A tested pixel is matched if $|\delta - \mathrm{med}_\delta| < \alpha d_t$, where med_δ is the median value of differences δ from all tested pixels, d_t is the object diameter, and α is a coefficient set to 0.05 in our experiments. Finally, pixel colors in test T_5 are compared in the HSV space as in [94].

A template passes tests T_2 to T_5 if at least v pixels in each test have a matching value within a small neighborhood of $g \times g$ pixels around the corresponding location in the sliding window. The matching value is searched in the neighborhood to compensate for the sliding window step and for the discretization of orientations during training. In our experiments, v was set to $0.6p$ to leave a tolerance for partial occlusions, and the extent of the neighborhood g was set to 5 px.

A template t that passes all the tests at a location w is assigned a final score computed as $c_{t,w} = \sum_{i\in\{2\ldots5\}} c_{t,w,i}$, where $c_{t,w,i}$ is the fraction of matching feature points in test T_i.

3.1.4 Non-Maxima Suppression

The verified templates are collected from all different locations and scales. Since different views of an object are often alike, and since multiple objects may be similar, there may be multiple overlapping detections. Unique detections are identified by repeatedly retaining the candidate with the highest score r, and removing all detections that have a large overlap with it. The score is defined as $r = c(a/s)$, where c is the verification score defined above, s is the detection scale, and a is the area of the object silhouette in

[3] The distance from the camera center to a 3D point $\mathbf{x_u}$ that projects to the pixel $\mathbf{u}$ can be calculated from the intrinsics and the value at $\mathbf{u}$ in the depth channel, which represents the Z coordinate of $\mathbf{x_u}$.

the template. Weighting the score by the silhouette area favors detections which explain more of the scene (e.g., when a cup is seen from a side, with the handle visible, a template depicting the handle is preferred over other templates which may have the same score c but do not show the handle). The retained detections are passed to the pose refinement stage, together with the approximate 6D poses associated with the templates.

3.2 Refinement by Particle Swarm Optimization

For a template t detected at a window location w, the pose refinement stage receives as input a 3D model of the object o_t, the object pose $\mathbf{P}_t = [\mathbf{R}_t \,|\, \mathbf{t}_t]$, the depth image channel, and the camera intrinsic parameters.

The pose $\mathbf{P}_t$ associated with the template t is valid when the center of t is aligned with the principal point of the image in the original scale. To reflect the 2D offset (u_w, v_w) and the scale s_w of the window location w, at which t was detected, $\mathbf{P}_t$ is transformed to: $\mathbf{P}_{t,w} = [\mathbf{R}_a\mathbf{R}_t \,|\, \mathbf{R}_a\mathbf{t}_t/s_w]$, where the rotation matrix $\mathbf{R}_a$ aligns the optical axis to a vector that originates at the optical center and passes through the pixel (u_w, v_w).

The pose $\mathbf{P}_{t,w}$ is refined by the Particle Swarm Optimization (PSO) [198], which stochastically evolves a population of candidate poses over multiple iterations. A candidate pose is evaluated by rendering the depth image of the 3D object model in that pose and comparing the rendering with the input depth image. As shown in [275], PSO is less prone to local minima compared to the commonly used Iterative Closest Point (ICP) algorithm [13]. A detailed description of the pose refinement procedure is in [275, 112].

3.3 Experimental Evaluation

This section compares the performance of HashMatch with other methods for 6D object localization (Section 3.3.1) and describes a robotic assembly project in which the method was deployed (Section 3.3.2). Results of HashMatch on more datasets are in Section 7.3.

3.3.1 6D Object Localization

Evaluation Methodology. Given a single RGB-D test image and an object identifier, the task is to estimate the 6D pose of the object. The estimate with the highest confidence $c_{t,w}$ is selected and evaluated in each image. The estimates are evaluated as in [94] using the Average Distance (AD) pose-error function defined in Section 2.5 – the ADI variant is used to evaluate pose estimates of objects #3, #7, #10, and #11, and the ADD variant for the other objects. A pose estimate is considered correct if the AD error is below 10% of the object diameter. The accuracy of a method is measured by the recall rate, i.e., the fraction of annotated object instances for which a correct pose is estimated.

Dataset. HashMatch is evaluated on the Linemod (LM) dataset [94], which includes 3D models of 15 texture-less objects and, for each object, a test set consisting of approximately 1200 RGB-D images in VGA resolution. Each image shows one annotated object instance under mild occlusion, heavy 2D and 3D clutter, and large viewpoint variation.

Compared Methods. The proposed method is compared with the Linemod [95] and Linemod++ [93] methods by Hinterstoisser et al., and with the methods by Drost et al. [63] and Kehl et al. [129]. Linemod follows an exhaustive template matching approach. Linemod++ extends it by two verification steps – a color check and a depth check by a rough but fast ICP. A finer ICP is then applied to the best of the remaining pose hypotheses. The essential difference of HashMatch is the addition of the objectness filter and the hypothesis generation stage which avoids the exhaustive search. The Hashmod method by Kehl et al. [129] is similar to HashMatch (they hash orientations of image gradients and surface normals, instead of depth measurements and orientations of surface normals). The method of Drost et al. [63] takes a different approach based on matching pairs of oriented 3D points between the point cloud of the test scene and the 3D object model, and aggregating the matches via a voting scheme.

Templates and Parameters. The set of templates T is obtained by rendering the 3D object models from a uniformly sampled upper half of a hemisphere centered at the object centroid (with a step of 10° in both azimuth and elevation). To achieve invariance to rotation around the optical axis, an in-plane rotation is additionally applied to each template (from -40° to 40° with a step of 10°). In total, an object is represented by 2916 templates of size 108×108 px. Each test image is scanned at 9 scales (4 larger and 4 smaller scales with a scaling factor of 1.2) with a scanning step of 5 px.

Results. HashMatch achieves the average recall rate of 95.4%, i.e., the recall rate averaged over the 15 objects (Table 3.1). HashMatch outperforms Linemod [95] by 12.4%, Drost et al. [63] by 16.1%, and is slightly inferior to Linemod++ [93] (-1.2%) and Hashmod (-1.1%). As for HashMatch, the computational complexity of Hashmod is sub-linear in the number of templates, while the complexity of the other compared methods is linear.

The sub-linear complexity of HashMatch in the number of loaded templates is demonstrated in Figure 3.7 (left). The average processing time per image is 0.75 s when only 2916 templates of a single object are loaded and 2.08 s when 43740 templates of 15 objects are loaded. The latter is noticeably lower than the expected processing time of exhaustive template matching, i.e., $0.75 \cdot 15 = 11.25$ s. The recall rate drops by only 1% when the templates of all 15 objects are loaded. The benefit of the refinement stage can be seen in Figure 3.7 (right) that compares the average AD errors of the initial and refined estimates.

The reported processing time is achieved with our parallelized C++ implementation on a modern desktop PC equipped with a 16 core CPU and an NVIDIA GTX 780 GPU (the GPU was employed only for the refinement stage). As reported in [94], the method of Drost et al. takes on average 6.3 s and Linemod++ 0.12 s. The latter was achieved

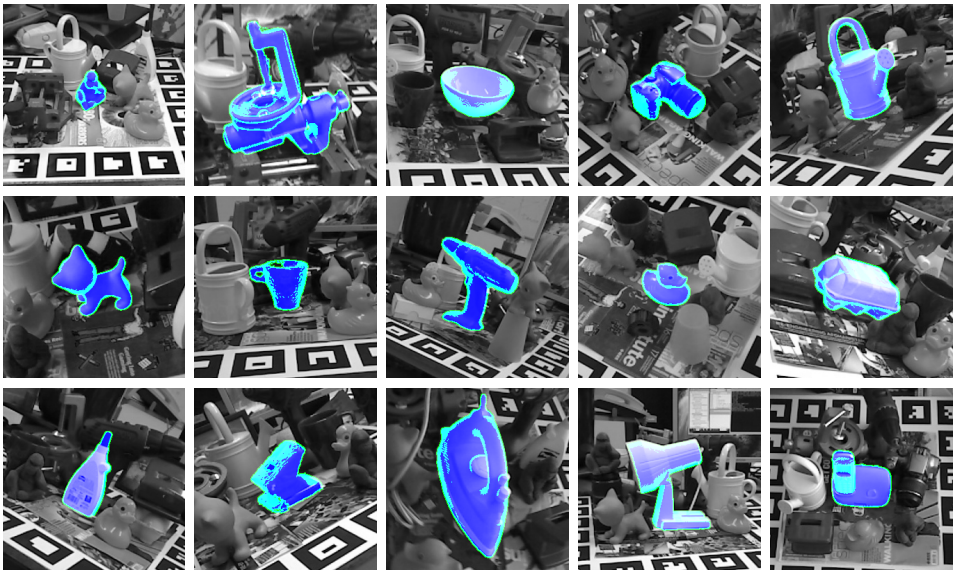

Figure 3.6. **HashMatch results on the LM dataset [94].** Shown are sample test images which were darkened, cropped and overlaid with 3D object models in the estimated 6D poses.

Method	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	Avg.
HashMatch	93.9	**99.8**	98.8	95.5	**95.9**	98.2	**99.5**	94.1	94.3	**100**	**98.0**	88.0	97.0	88.8	89.4	95.4
Hashmod	**96.1**	92.8	99.3	**97.8**	92.8	98.9	96.2	**98.2**	94.1	99.9	96.8	95.7	96.5	98.4	**93.3**	96.5
LM++	95.8	98.7	**99.9**	97.5	95.4	**99.3**	97.1	93.6	**95.9**	99.8	91.8	**95.9**	**97.5**	**97.7**	**93.3**	**96.6**
LM	69.4	94.0	99.5	79.5	79.5	88.2	80.7	81.3	75.9	99.1	64.3	78.4	88.8	89.8	77.8	83.0
Drost	86.5	70.7	95.7	78.6	80.2	85.4	68.4	87.3	46.0	97.0	57.2	77.4	84.9	93.3	80.7	79.3

Table 3.1. **Recall rates [%] on the LM dataset [94].** A pose estimate is considered correct if the AD error is below 10% of the object diameter. The object ID's are as in [106].

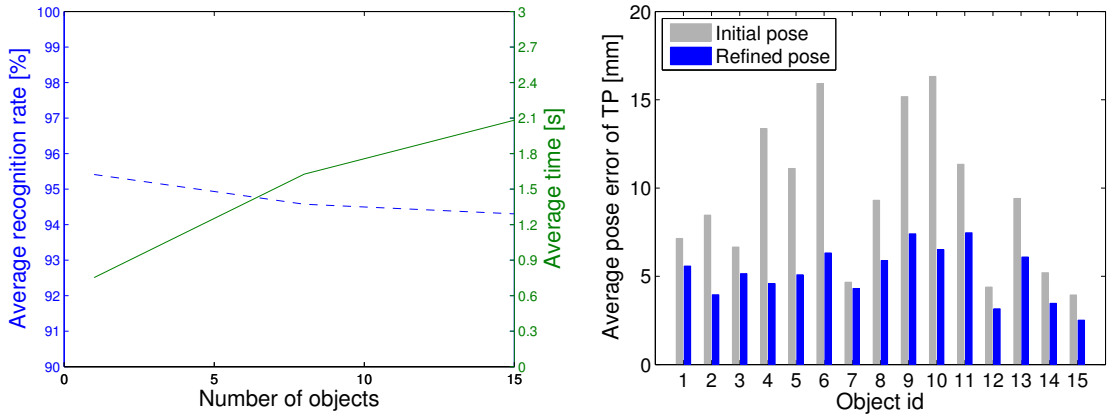

Figure 3.7. **Left: Sub-linear complexity in the # of templates.** The average recall rate (dashed line) and the average processing time are plotted w.r.t. the number of loaded templates (2916 templates per object). The processing time is 0.75 s when templates of a single object are loaded and 2.08 s when templates of 15 objects are loaded, which is noticeably lower than the expected processing time of an exhaustive template matching approach, i.e., $0.75 \cdot 15 = 11.25$ s. **Right: Average AD errors of initial/refined pose estimates.** Only pose estimates whose AD error is below 10% of the object diameter after the refinement are considered.

with a highly optimized implementation using SSE parallelization and a limited scale space (for each object, only a limited set of scales was considered). With a similar level of optimization, HashMatch is expected to run even faster thanks to its sub-liner complexity.

HashMatch placed fourth out of the 15 participants of the BOP Challenge 2017 (Section 7.3). The leaderboard, comparing the participating methods on seven datasets, are presented in Section 7.3.3. Besides the results of the complete HashMatch method, the leaderboard shows also results of HashMatch without the pose refinement stage – the average recall dropped by 11.8% without the refinement (from 67.2% to 55.4%). Visualizations of sample HashMatch results on the LM dataset can be found in Figure 3.6.

Compared to modern methods based on neural networks, such as EPOS presented in Chapter 4, the advantage of HashMatch is the speed of learning new objects. While retraining a network may take several hours or days, HashMatch only requires to recalculate the hash tables, which typically takes less than a minute. However, as shown in Section 7.3.3, EPOS significantly outperforms HashMatch in the recall rate on most datasets (+38% on TUD-L, +27% on LM-O, +15% on T-LESS, -2% on IC-BIN). The fact that EPOS uses only RGB channels makes these improvements even more notable.

3.3.2 Robotic Application

HashMatch was successfully deployed in the DARWIN FP7 EU project [51] aiming to develop a dexterous assembler robot working with embodied intelligence. Figure 3.8 shows the estimated poses in the considered environment. When a robotic gripper was present in the scene, its posture was accurately provided by motor encoders and used to mask out the corresponding pixels in the image, preventing them from contaminating the pose estimation. As shown in Figure 3.7 (right), HashMatch achieves a sub-centimeter average accuracy of the estimated poses, which meets the requirements imposed by the compliant grippers of the industrial robotic arms used in the project (Stäubli RX130 and RX90).

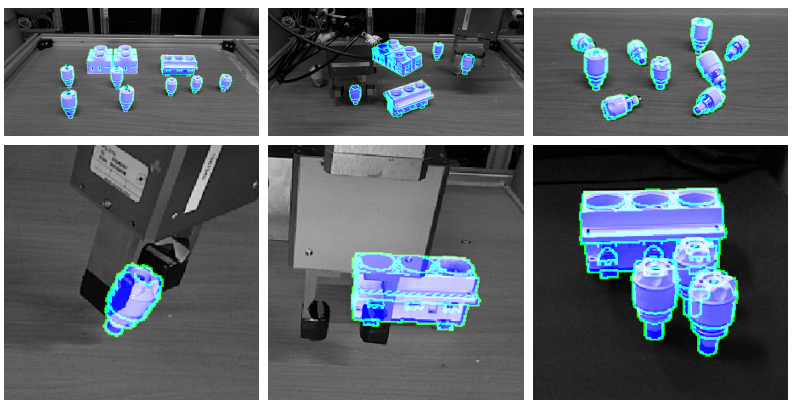

Figure 3.8. **HashMatch supporting robotic manipulation in the DARWIN project.** Industrial robotic arms with compliant grippers were manipulating texture-less electrical fuses and fuse boxes of different types, which were later included in the T-LESS dataset (Chapter 6).

Chapter 4

EPOS

Estimating 6D Pose of Objects with Symmetries

This chapter presents EPOS, a method for estimating the 6D pose of specific rigid objects with available 3D models from a single RGB input image. The method is based on establishing 2D-3D correspondences between densely sampled pixels of the input image and 3D locations on the object model. The key idea is to represent an object by a controllable number of compact surface fragments, which allows handling object symmetries by predicting multiple potential 2D-3D correspondences per pixel (Figure 4.1), and ensures a consistent number and uniform coverage of candidate 3D locations on objects of any type. Thanks to this representation, the method is applicable to a broad range of objects – besides those with a distinct and non-repeatable shape or texture (a shoe, a box of corn flakes, etc. [156, 42]), the method can handle texture-less objects and objects with global or partial symmetries (a bowl, a cup, etc. [102, 62, 94]).

Potential correspondences between densely sampled pixels and the fragments are predicted by an encoder-decoder network. At each pixel, the network predicts (i) the probability of each object's presence, (ii) the probability of each fragment given the object's presence, and (iii) a precise 3D location on each fragment (Figure 4.2). By modeling the probability of fragments conditionally, the uncertainty due to global and partial object symmetries (which are assumed unknown) is decoupled from the uncertainty of the object's presence. The decoupled uncertainty due to symmetries is then used to guide the selection of possibly multiple potential correspondences at each pixel. The resulting set of potential correspondences collected at all pixels forms a many-to-many relationship, i.e., a pixel may correspond to multiple 3D locations on the model surface and vice versa.

Poses of possibly multiple instances of every object model are estimated from the potential correspondences by an efficient and robust variant of the PnP-RANSAC algorithm integrated in the Progressive-X scheme [9]. The efficiency is achieved by the PROSAC sampler [39] that prioritizes potential correspondences with a high predicted probability, while the robustness is achieved by a novel function for measuring the quality of a pose hypothesis. This function considers only the most accurate potential correspondence at each pixel as only up to one may be compatible with the hypothesis; the other potential correspondences provide alternative explanations and do not influence the quality.

Recent correspondence-based methods predict only a single 3D location per pixel

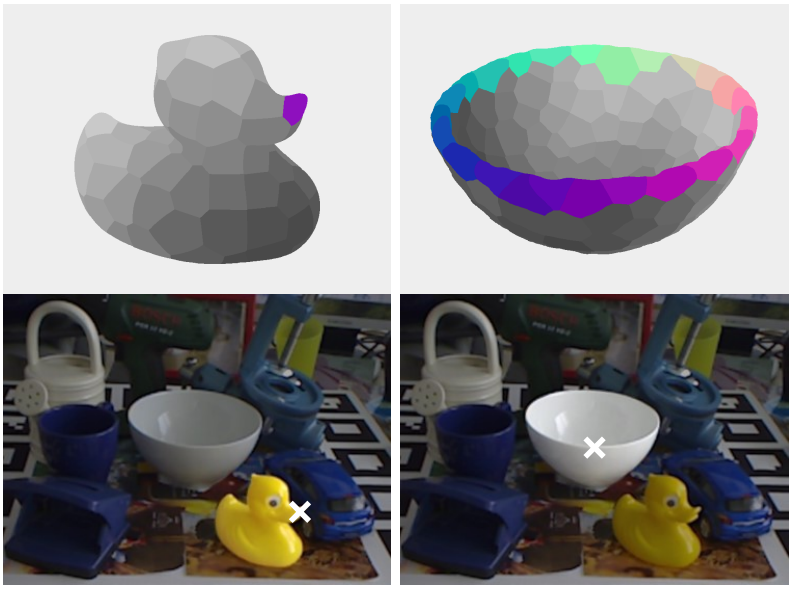

Figure 4.1. **Ambiguity of 2D-3D correspondences.** A 2D image location corresponds to a *single* 3D location on the object model in the case of distinct object parts (left), but to one of *multiple* indistinguishable 3D locations in the case of global or partial object symmetries (right). Representing an object by surface fragments allows predicting *possibly multiple* potential correspondences per pixel. The potentially corresponding fragments are colored by mapping (x, y, z) coordinates of their centers in the model space to (R, G, B) channels.

and therefore cannot reliably handle object symmetries (Section 2.2.2). On the other hand, classical correspondence-based methods based on matching local features (Sections 2.1.2 and 2.1.3) can establish multiple correspondences per pixel, but perform poorly on texture-less objects because the feature detectors often fail to provide a sufficient number of reliable locations and the descriptors are no longer discriminative enough [248, 112].

Other part-based object representations have appeared in the literature, mainly with the motivation to increase the robustness of methods against occlusion, but have not been utilized to handle object symmetries. For example, Crivellaro et al. [47] represent an object by a set of manually selected distriminative parts and estimate the 6D pose of each part by predicting the 2D projections of pre-selected 3D keypoints. Brachmann et al. [17] and Nigam et al. [179] split the 3D bounding box of the object model into uniform bins and predict up to one corresponding bin per pixel. However, they represent each bin with its center which yields correspondences with limited precision. For human pose estimation, Güler et al. [5] segment the 3D surface of the human body into semantically-defined parts. At each pixel, they predict a single label of the corresponding part and the UV texture coordinates defining the precise 3D location on the part. In contrast, to effectively capture the partial object symmetries, we represent an object by a set of compact surface fragments of similar size and predict possibly multiple labels of the corresponding fragments per pixel. Besides, we regress the precise location in local 3D coordinates of the fragment instead of the UV coordinates. Using the UV coordinates

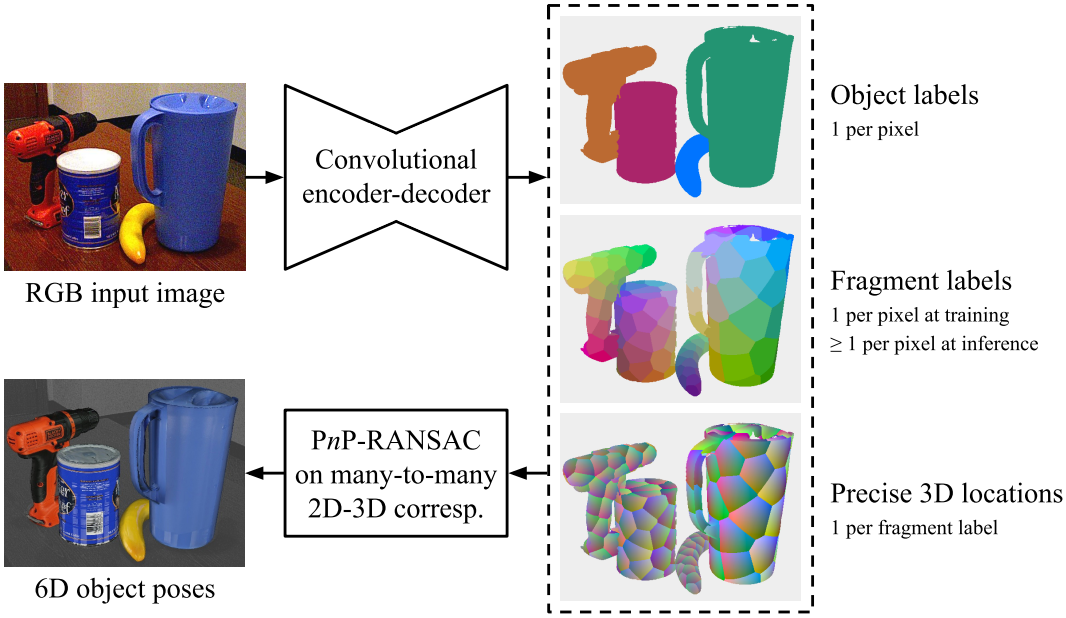

Figure 4.2. **EPOS pipeline.** At training time, the encoder-decoder network is provided for each pixel an annotation in the form of a single object label (representing an object model), a single fragment label, and a single 3D location in the local coordinate frame of the fragment. At test time, the network predicts at each pixel (i) a probability distribution over object models, (ii) a probability distribution over all fragments of the most probable object model, which is used to select a set of potentially corresponding fragments, and (iii) a precise 3D location in the local coordinate frame of each of the selected fragments. Each pixel is then linked with the precise 3D locations on the potentially corresponding fragments, and 6D poses of possibly multiple instances of each object model are estimated by an efficient and robust variant of the PnP-RANSAC algorithm. In the visualizations, fragments are colored by mapping (x, y, z) coordinates of their centers in the model space to (R, G, B), and the precise 3D locations are colored by mapping (x, y, z) coordinates in the local fragment space to (R, G, B).

requires a well-defined topology of the mesh model, which may need manual intervention, and is problematic for objects with a complicated surface such as a coil or an engine [62].

Compared with the participants of the BOP Challenge 2019 [100, 106], EPOS outperformed all RGB methods and most RGB-D and D methods on the T-LESS [102] and LM-O [17] datasets, which include texture-less and symmetric objects captured in cluttered scenes under various levels of occlusion. On the YCB-V [271] dataset, which includes both textured and texture-less objects, the method was superior to all competitors, with a significant 27% absolute improvement over the second-best RGB method. These results are achieved without any post-refinement of the estimated poses, such as [159, 147, 205]. As shown later in Section 7.3.3, EPOS significantly outperforms also the HashMatch method proposed in Chapter 3, even though HashMatch relies on RGB-D image channels.

EPOS was published in [99], is described in detail in Sections 4.1–4.3 and evaluated in Section 4.4. The project website with a video presentation, a real-world demonstration, the source code, and pre-trained models is available at: `cmp.felk.cvut.cz/epos`.

4.1 Surface Fragments

A mesh model $M_o = (V_o, T_o)$, defined by a set of 3D vertices V_o and a set of triangular faces T_o, is assumed available for each object, where objects are referred to by labels $o \in O = \{1, \ldots, m\}$. The set of all 3D points on the model surface, denoted as S_o, is split into n fragments referred to by labels $f \in F = \{1, \ldots, n\}$. A surface fragment f of an object o is defined as $S_{o,f} = \{\mathbf{x} \in S_o \mid d(\mathbf{x}, \mathbf{g}_{o,f}) < d(\mathbf{x}, \mathbf{g}_{o,i}), \forall i \in F, i \neq f\}$, where $d(\cdot)$ is the Euclidean distance of two 3D points and $\{\mathbf{g}_{o,f}\}_{f=1}^{n}$ is a set of fragment centers. The centers are found by the farthest point sampling algorithm which iteratively selects the vertex from V_o that is farthest from the already selected vertices. The algorithm starts with the centroid of the model which is then removed from the final set of centers.

Note that the number of fragments was fixed across all objects for the experiments presented in Section 4.4. A study of object-specific numbers of fragments, which may improve performance and may depend on factors such as the physical object size, shape, or the range of distances of the object from the camera, is left for future work.

4.2 Predicting Potential 2D-3D Correspondences

This section describes the prediction of potential 2D-3D correspondences between densely sampled pixels of the input image and precise 3D locations on the object models. Each object is represented by a set of surface fragments which allow predicting multiple potential correspondences per pixel.

4.2.1 Decoupling Uncertainty Due to Symmetries

The probability that a fragment f of an object o is visible at a pixel with image coordinates $\mathbf{u} = (u, v) \in \{1, \ldots, w\} \times \{1, \ldots, h\}$, where (w, h) is the image size, is modeled as:

$$\Pr(Y{=}f, X{=}o \,|\, \mathbf{u}) = \Pr(Y{=}f \,|\, X{=}o, \mathbf{u}) \Pr(X{=}o \,|\, \mathbf{u})$$

where X and Y are random variables representing the object and the fragment respectively. The probability $\Pr(Y{=}f, X{=}o \,|\, \mathbf{u})$ may be low because (1) the object o is not present at the pixel $\mathbf{u}$ or the object identity is ambiguous, or (2) the identity of the fragment visible at $\mathbf{u}$ is ambiguous, which may be caused by global or partial symmetries of the object. To disentangle these two ambiguities, we predict $a_o(\mathbf{u}) = \Pr(X{=}o \,|\, \mathbf{u})$ and $b_{o,f}(\mathbf{u}) = \Pr(Y{=}f \,|\, X{=}o, \mathbf{u})$ separately, instead of directly predicting $\Pr(Y{=}f, X{=}o \,|\, \mathbf{u})$. Expressing the probability distribution $b_{o,f}$ separately enables selecting a data-dependent number of potentially corresponding fragments per pixel (described in Section 4.2.6).

4.2.2 Regressing Precise 3D Locations

A surface fragment f of an object o is associated with a regressor $\mathbf{r}_{o,f} : \mathbb{R}^2 \rightarrow \mathbb{R}^3$, which at a pixel $\mathbf{u}$ predicts the corresponding 3D location: $\mathbf{r}_{o,f}(\mathbf{u}) = (\mathbf{x} - \mathbf{g}_{o,f})/h_{o,f}$. The regressor produces *3D fragment coordinates*, which are defined in a 3D coordinate frame with the origin at the fragment center $\mathbf{g}_{o,f}$. The 3D point $\mathbf{x}$ is expressed in the 3D coordinate frame of the model, and the scalar $h_{o,f}$ normalizes the regression range and is defined as the length of the longest side of the 3D bounding box of the fragment.

4.2.3 Dense Prediction

A single deep neural network with an encoder-decoder structure, e.g., DeepLabv3+ [31], is adopted to densely predict $a_i(\mathbf{u})$, $b_{o,f}(\mathbf{u})$ and $\mathbf{r}_{o,f}(\mathbf{u})$, $\forall o \in O$, $\forall f \in F$, $\forall i \in O \cup \{0\}$, where 0 is reserved for the background class. For m objects, each represented by n surface fragments, the network has $4mn+m+1$ output channels ($m+1$ for probabilities of the objects and the background, mn for probabilities of the surface fragments, and $3mn$ for the 3D fragment coordinates).

4.2.4 Network Training

The network is trained by minimizing the following loss averaged over all pixels $\mathbf{u}$:

$$\begin{aligned} L(\mathbf{u}) = {} & E\big(\bar{\mathbf{a}}(\mathbf{u}), \mathbf{a}(\mathbf{u})\big) \\ & + \lambda_1 \sum\nolimits_{o \in O} \bar{a}_o(\mathbf{u}) E\big(\bar{\mathbf{b}}_o(\mathbf{u}), \mathbf{b}_o(\mathbf{u})\big) \\ & + \lambda_2 \sum\nolimits_{o \in O} \bar{a}_o(\mathbf{u}) \sum\nolimits_{f \in F} \bar{b}_{o,f}(\mathbf{u}) H\big(\bar{\mathbf{r}}_{o,f}(\mathbf{u}), \mathbf{r}_{o,f}(\mathbf{u})\big) \end{aligned} \tag{4.1}$$

where E is the softmax cross-entropy loss, H is the Huber loss [117], $\mathbf{a}(\mathbf{u}) = [a_0(\mathbf{u}), \ldots, a_m(\mathbf{u})]$, and $\mathbf{b}_o(\mathbf{u}) = [b_{o,1}(\mathbf{u}), \ldots, b_{o,n}(\mathbf{u})]$. The ground-truth one-hot vectors $\bar{\mathbf{a}}(\mathbf{u})$ and $\bar{\mathbf{b}}_o(\mathbf{u})$ indicate which object (or the background) and which fragment is visible at the pixel $\mathbf{u}$. Elements of these ground-truth vectors are denoted as $\bar{a}_o(\mathbf{u})$ and $\bar{b}_{o,f}(\mathbf{u})$. The vector $\bar{\mathbf{b}}_o(\mathbf{u})$ is defined only if the object o is present at $\mathbf{u}$. The ground-truth 3D fragment coordinates are denoted as $\bar{\mathbf{r}}_{o,f}(\mathbf{u})$. Weights λ_1 and λ_2 are used to balance the loss terms.

The network is trained on images annotated with ground-truth 6D object poses. Vectors $\bar{\mathbf{a}}(\mathbf{u})$, $\bar{\mathbf{b}}_o(\mathbf{u})$, and $\bar{\mathbf{r}}_{o,f}(\mathbf{u})$ are obtained by rendering the 3D object models in the ground-truth poses with a custom OpenGL shader. Pixels outside the visibility masks of the objects, which are calculated as in [105], are considered to be the background. The training images are augmented by randomly adjusting brightness, contrast, hue, and saturation, and by applying random Gaussian noise and blur, similarly to [97] (Figure 4.3).

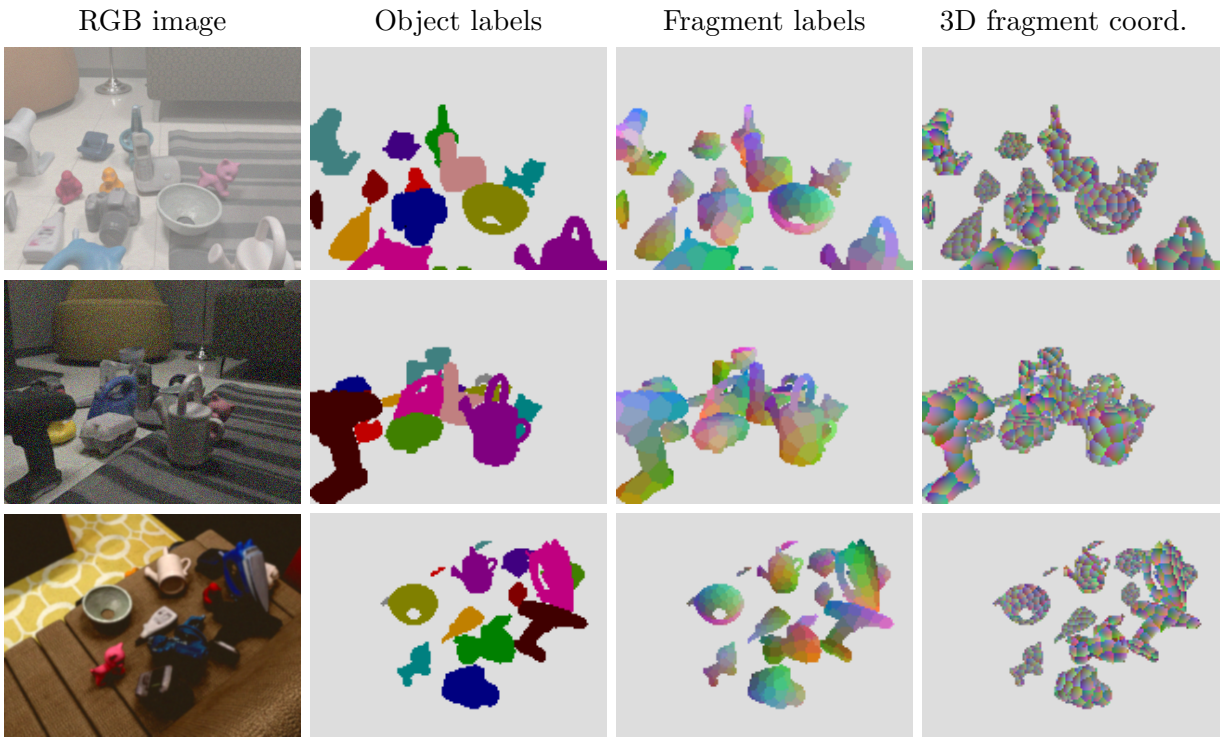

Figure 4.3. **Training data.** The left column shows sample synthetic images from [111] augmented by randomly adjusting brightness, contrast, hue, and saturation, and by applying random Gaussian noise and blur, similarly to [97]. The right three columns show the ground-truth generated by a custom OpenGL shader (labels are converted to one-hot vectors for training).

4.2.5 Learning in the Presence of Object Symmetries

In the case of global or partial object symmetries, a pixel may have multiple potentially corresponding 3D locations that are indistinguishable in the given image (Figure 4.1). One could try to find the indistinguishable 3D locations and train the network to predict a high probability for each of the surface fragments on which these locations lie. Finding 3D locations that are indistinguishable due to global object symmetries is relatively easy as the global symmetries can be detected from the shape and texture of the object models (Section 7.1.5). On the other hand, finding 3D locations that are indistinguishable due to partial object symmetries is difficult as one would need to identify the visible object parts in each training image and find their fits to the object models.

Instead of explicitly listing all indistinguishable 3D locations, we provide the network with only a single corresponding 3D location per pixel during training (see Equation 4.1) and let the network learn all of the potential correspondences implicitly. This is achieved by minimizing the softmax cross-entropy loss $E\left(\bar{\mathbf{b}}_o(\mathbf{u}), \mathbf{b}_o(\mathbf{u})\right)$, which corresponds exactly to minimizing the Kullback-Leibler divergence of distributions $\bar{\mathbf{b}}_o(\mathbf{u})$ and $\mathbf{b}_o(\mathbf{u})$ [82], and by ensuring that object poses in the training set are sampled uniformly w.r.t. the three rotation angles and the distance from the camera. Such sampling of object poses is easy

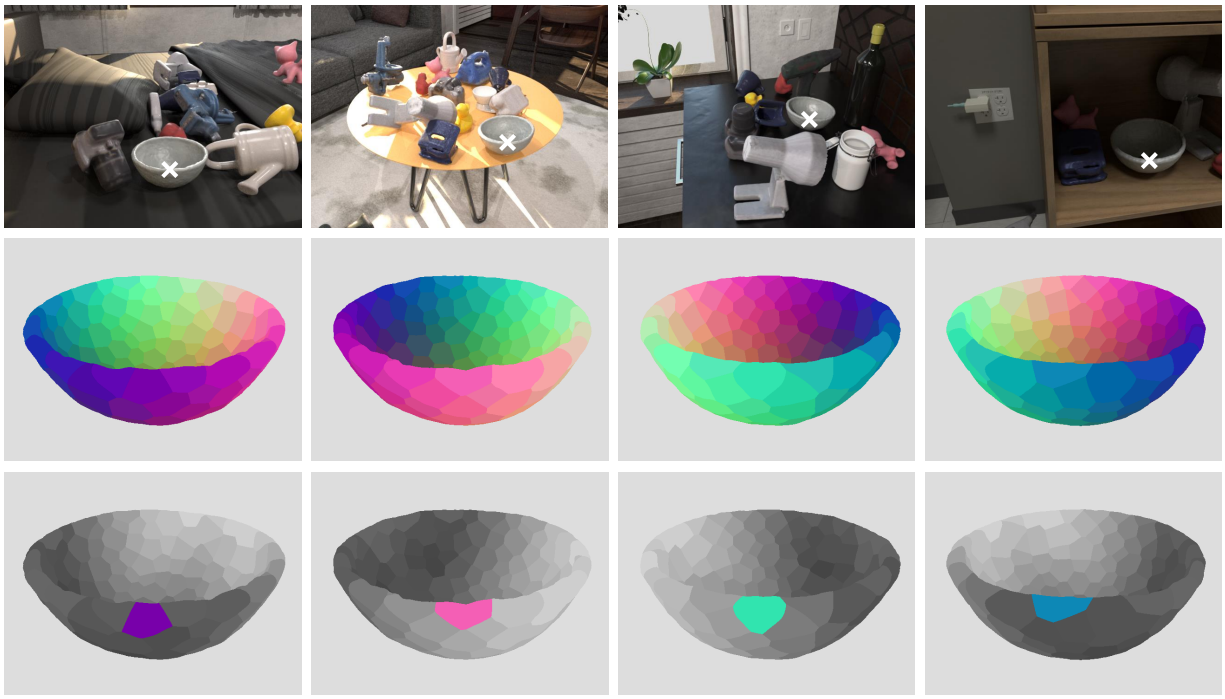

Figure 4.4. **Learning in the presence of object symmetries.** The first row shows sample synthetic training images, the second row visualizes the surface fragments of the bowl oriented around the vertical axis as in the images, and the third row highlights the ground-truth fragments assigned to the pixels marked in the first row. Instead of explicitly identifying all indistinguishable (i.e., potentially corresponding) fragments at each pixel of the training images (which may be not trivial if the object pose is ambiguous due to occlusions), each pixel is assigned a single ground-truth fragment. By providing the network with multiple training images showing the objects in uniformly distributed poses, the network is expected to learn a uniform distribution over the indistinguishable fragments. The color-coding of fragments is explained in Figure 4.1.

to achieve with synthetic training images and ensures that each of the indistinguishable fragments is indicated by the ground-truth one-hot distribution $\bar{\mathbf{b}}_o(\mathbf{u})$ at a comparable number of pixels $\mathbf{u}$. The network is then expected to learn at these pixels a high value of the probability $b_{o,f}(\mathbf{u})$ for all of the indistinguishable fragments. For example, the probability distribution on a sphere is expected to be uniform, i.e., $b_{o,f}(\mathbf{u}) = 1/n, \forall f \in F$. On a bowl, the probability is expected to be constant around the axis of symmetry (Figure 4.4).

4.2.6 Establishing Potential 2D-3D Correspondences

A pixel $\mathbf{u}$ is linked with a 3D location $\mathbf{x}_{o,f}(\mathbf{u}) = h_{o,f}\mathbf{r}_{o,f}(\mathbf{u}) + \mathbf{g}_{o,f}$ on all fragments for which $a_o(\mathbf{u}) > \tau_a$ and $b_{o,f}(\mathbf{u}) / \max_{i=1}^{n}(b_{o,i}(\mathbf{u})) > \tau_b$. The threshold τ_b is relative to the maximum to collect locations from all indistinguishable fragments that are expected to have a similarly high probability $b_{o,f}(\mathbf{u})$. The set of potential 2D-3D correspondences established for object o, which may cover multiple instances of this object, is denoted as $C_o = \{(\mathbf{u}, \mathbf{x}_{o,f}(\mathbf{u}), s_{o,f}(\mathbf{u}))\}$, where $s_{o,f}(\mathbf{u}) = a_o(\mathbf{u})b_{o,f}(\mathbf{u})$ is a confidence.

4.3 Robust and Efficient 6D Pose Fitting

The next step of the pipeline is to estimate the 6D poses of possibly multiple instances of each object o from the set of potential 2D-3D correspondences C_o. A 6D pose is defined by a rigid transformation from the 3D model frame to the 3D camera frame, and the correspondences define links between the 3D model frame and the 2D image frame.

If only a single instance of an object o was visible in the image, and if only up to one correspondence was established per pixel, the pose could be effectively estimated by solving the Perspective-n-Point (PnP) problem [145]. Popular algorithms to solve this problem include P3P [134] for $n = 3$ correspondences, EPnP [145] for $n \geq 4$, and DLS-PnP [92] for $n \geq 3$. For robust estimation on real-world data, these algorithms are often combined with the RANSAC fitting scheme [71]. In this scheme, the PnP problem is solved repeatedly on a randomly sampled subset of correspondences and the final output is defined by the pose hypothesis with the highest quality, which is typically defined by the number of inlier correspondences [71] or their likelihood [250]. A correspondence $(\mathbf{u}, \mathbf{x})$ is considered an inlier w.r.t. a pose estimate $\hat{\mathbf{P}} = [\hat{\mathbf{R}} \,|\, \hat{\mathbf{t}}]$, if the re-projection error:

$$e\big(\mathbf{u}, \mathbf{x}, \hat{\mathbf{R}}, \hat{\mathbf{t}}\big) = \big\|\mathbf{u} - \text{proj}\big(\hat{\mathbf{R}}\mathbf{x} + \hat{\mathbf{t}}\big)\big\|_2 \tag{4.2}$$

is below a threshold τ_r. Function proj$(\cdot)$ is the 2D projection with values in pixels. If the re-projection error is larger or equal to τ_r, the correspondence is considered an outlier.

In our case, multiple instances of an object o may be visible in the image, their number may be unknown (depending on the 6D object pose estimation problem; Section 1.1) and there may be multiple potential correspondences established at each pixel. Specifically, with respect to a single pose hypothesis, the set C_o of potential correspondences may include three types of outliers. First, it may include outliers due to erroneous prediction of the 3D locations. Second, for each 2D/3D location there may be up to one correspondence which is compatible with the pose hypothesis; the other correspondences act as outliers. Third, correspondences originating from different instances of the object o are also incompatible with the pose hypothesis. The set C_o may be therefore contaminated with a high proportion of outliers and a robust and efficient estimator able to estimate poses of multiple object instances is needed.

4.3.1 Single-Instance Fitting

The 6D pose of a single instance of an object o is estimated from the set of correspondences C_o by GC-RANSAC [8], a variant of RANSAC which locally optimizes every new so-far-the-best hypothesis by alternating between two steps: (i) selecting the inliers by a graph-cut optimization, and (ii) refining the hypothesis from the currently selected inliers. The inlier selection approach is motivated by the fact that both inliers and outliers are typically spatially coherent, which is the case also for 2D-3D correspondences – a corre-

spondence near an inlier or outlier (in 2D and 3D) is more likely to be an inlier or outlier respectively. A neighborhood graph of correspondences for the graph-cut optimization is constructed by describing each correspondence with a 5D vector consisting of the 2D and 3D coordinates (in pixels and centimeters) and linking two 5D descriptors if their Euclidean distance is below a threshold τ_d.[1] The unary and pair-wise energy terms to optimize are defined using a kernel function of the re-projection error truncated at τ_r.

A pose hypothesis is calculated by the P3P solver [134] from a triplet of correspondences sampled by PROSAC [39], which first focuses on correspondences with high confidence $s_{o,f}$ (defined in Section 4.2.6) and progressively blends to a uniform sampling. Triplets which form 2D triangles with the area below a threshold τ_t or have collinear 3D locations are rejected, and pose hypotheses behind the camera or with the determinant of the rotation matrix equal to -1 (i.e., an improper rotation matrix [86]) are discarded. The hypothesis that passes these early-rejection tests is refined from all inliers by the EPnP solver [145] followed by the Levenberg-Marquardt (L-M) optimization [171]. EPnP and L-M are used also for refinement in the local optimization which is applied when a new so-far-the-best hypothesis is found.

The quality of a pose hypothesis $\hat{\mathbf{P}} = [\hat{\mathbf{R}} \,|\, \hat{\mathbf{t}}]$ is defined as:

$$q\big(\hat{\mathbf{R}}, \hat{\mathbf{t}}, U_o, C_{o,\mathbf{u}}, \tau_r\big) = \frac{1}{|U_o|} \sum_{\mathbf{u} \in U_o} \max_{(\mathbf{u},\mathbf{x},\cdot) \in C_{o,\mathbf{u}}} \max\left(0,\, 1 - \frac{e\big(\mathbf{u}, \mathbf{x}, \hat{\mathbf{R}}, \hat{\mathbf{t}}\big)^2}{\tau_r^2}\right) \tag{4.3}$$

where U_o is a set of pixels at which the correspondences C_o are established, $C_{o,\mathbf{u}} \subset C_o$ are correspondences established at the pixel $\mathbf{u}$, $e(\cdot)$ is the re-projection (Equation 4.2), and τ_r is the inlier-outlier threshold. The use of the truncated quadratic function is inspired by MSAC [251, 142]. At each pixel, the quality q considers only the most accurate potential correspondence as only up to one may be compatible with the hypothesis; the others provide alternative explanations and do not influence the quality.

GC-RANSAC runs for up to τ_i iterations until the quality q of a pose hypothesis reaches a threshold τ_q. The hypothesis with the highest q defines the final outcome.

4.3.2 Multi-Instance Fitting

The 6D poses of multiple instances of an object o are estimated from correspondences C_o by the Progressive-X fitting scheme [9]. In this scheme, pose hypotheses are proposed one by one and maintained in a set H. The hypotheses are proposed by the single-instance fitting method (described in Section 4.3.1) with the quality function modified to encourage

[1] Optimal units of the 2D and 3D coordinates forming the 5D space may depend on factors such as the object size, its distance from the camera or intrinsic camera parameters. Representing the 2D and 3D coordinates in pixels and centimeters respectively worked well on all datasets used in our experiments.

exploring correspondences that have not been explained by any hypothesis from H:

$$q\big(\hat{\mathbf{R}}, \hat{\mathbf{t}}, H, U_o, C_{o,\mathbf{u}}, \tau_r\big) = \frac{1}{|U_o|} \sum_{\mathbf{u} \in U_o} \max_{(\mathbf{u},\mathbf{x},\cdot) \in C_{o,\mathbf{u}}} \max\left(0, \min\left(1 - \frac{e\big(\mathbf{u}, \mathbf{x}, \hat{\mathbf{R}}, \hat{\mathbf{t}}\big)^2}{\tau_r^2}, \frac{e'\big(\mathbf{u}, \mathbf{x}, H\big)^2}{\tau_r^2}\right)\right) \tag{4.4}$$

where the function e' is defined by the minimum re-projection error over the set H:

$$e'\big(\mathbf{u}, \mathbf{x}, H\big) = \min_{[\mathbf{R}\,|\,\mathbf{t}] \in H} e\big(\mathbf{u}, \mathbf{x}, \mathbf{R}, \mathbf{t}\big) \tag{4.5}$$

A proposed pose hypothesis is added to H if the Jaccard similarity [246], calculated between the set of inliers of the proposed hypothesis and the set of inliers of all hypotheses from H, is below a threshold τ_j. After adding a hypothesis, the set H is consolidated by the PEARL convex optimization [119], which minimizes an energy balancing geometric errors and regularity of inlier clusters by alternating between two steps: (i) selecting inliers for each hypothesis from H by the alpha-expansion optimization, and (ii) refining all hypotheses from the selected inliers. A hypothesis is removed from H when no inliers are assigned to it in the first step. PEARL utilizes the spatial coherence of correspondences using the same neighborhood graph as GC-RANSAC and runs until convergence.

Progressive-X terminates if the probability of finding another instance [9] falls below a threshold τ_p or if the specified number of object poses is estimated (the latter condition applies only in the case of the 6D object localization problem).

4.4 Experimental Evaluation

This section compares the performance of EPOS with other 6D object pose estimation methods and presents ablation experiments demonstrating the benefit of representing an object by surface fragments, of regressing the precise 3D locations on the fragments, and of fitting the pose by a modern robust estimator.

4.4.1 Experimental Setup

Evaluation Methodology. We follow the evaluation methodology of the BOP Challenge 2019 described in Section 7.1. In short, a method is evaluated on the 6D object localization problem, and the error of an estimated pose $\hat{\mathbf{P}}$ w.r.t. the ground-truth pose $\bar{\mathbf{P}}$ is calculated by three pose-error functions: Visible Surface Discrepancy (VSD), which treats indistinguishable poses as equivalent by considering only the visible object part; Maximum Symmetry-Aware Surface Distance (MSSD), which considers a set of pre-identified global object symmetries and measures the surface deviation in 3D; and Maximum Symmetry-Aware Projection Distance (MSPD), which considers the object symmetries and measures

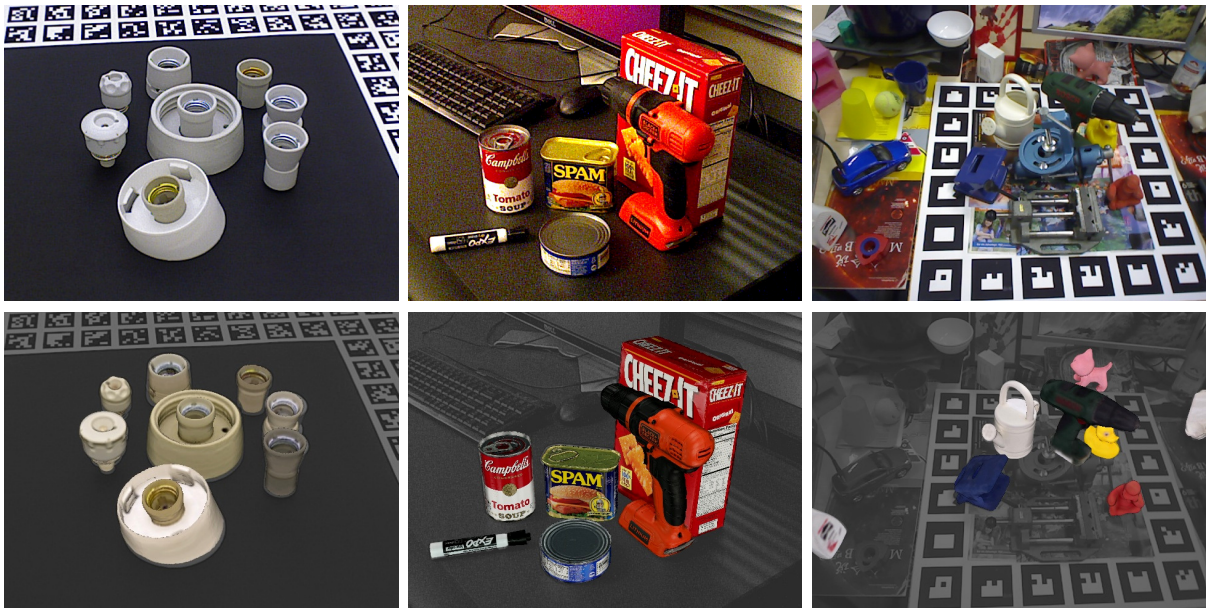

Figure 4.5. **Example EPOS results** on T-LESS (left), YCB-V (middle) and LM-O (right). At the top are renderings of the 3D object models in poses estimated from the RGB images at the bottom. All eight LM-O objects, including two truncated ones, are detected in the bottom example. More results, including a real-world demo, can be found on the project website.

the perceivable deviation. MSPD is suitable for the evaluation of RGB methods, for which estimating the Z translational component (along the optical axis) is more challenging. An estimated pose is considered correct w.r.t. a pose-error function e, if $e < \theta_e$, where $e \in \{\text{VSD}, \text{MSSD}, \text{MSPD}\}$ and θ_e is the threshold of correctness. The fraction of annotated object instances, for which a correct pose is estimated, is referred to as Recall. The Average Recall w.r.t. a function e, denoted as AR_e, is defined as the average of the Recall rates calculated for multiple settings of the threshold θ_e and also for multiple settings of the misalignment tolerance τ in the case of VSD. The accuracy of a method is measured by the Average Recall: $\text{AR} = (\text{AR}_{\text{VSD}} + \text{AR}_{\text{MSSD}} + \text{AR}_{\text{MSPD}}) / 3$. As EPOS uses only the RGB image channels, besides AR we report AR_{MSPD}.

Datasets. The experiments are conducted on datasets T-LESS [102], YCB-V [271], and LM-O [17]. The datasets include 3D object models and RGB-D images of VGA resolution with ground-truth 6D object poses (EPOS uses only the RGB channels). LM-O contains eight, mostly texture-less objects and 200 test images of the objects in a clutered scene under various levels of occlusion. YCB-V includes 21 objects, which are both textured and texture-less, and 900 test images showing the objects with occasional occlusion and limited clutter. T-LESS contains 30 objects with no significant texture and with symmetries and mutual similarities. It includes 1000 test images from 20 scenes with varying complexity, including challenging scenes with multiple instances of several objects and with a high amount of clutter and occlusion. See Section 7.2 for more details.

Training Images. The network is trained on several types of images. For T-LESS, we use all 30K physically-based rendered (PBR) images from SyntheT-LESS [196], 50K images of objects rendered by OpenGL on random photographs from NYU Depth V2 [227], which we generated similarly to [97], and all 38K real images from [102] showing objects on black background, where we replaced the background with random photographs. For YCB-V, we use the provided 113K real and 80K synthetic images. For LM-O, we use 67K PBR images from [111] (scenes 1 and 2) and 50K images of objects rendered by OpenGL on random photographs. No real images of the objects are used for training on LM-O.

Optimization. We use the DeepLabv3+ encoder-decoder network [31] with Xception-65 [38] as the backbone. The network is pre-trained on the Microsoft COCO dataset [151] and fine-tuned on the training images detailed above for 2M iterations. The training images are augmented as described in Section 4.2.4, the batch size is set to 1, the initial learning rate to 0.0001, parameters of batch normalization are not fine-tuned and other hyper-parameters are set as in [31]. To overcome the domain gap between the synthetic training and real test images, we apply the simple technique from [97] and freeze the "early flow" part of Xception-65. For LM-O, we additionally freeze the "middle flow" since there are no real training images in this dataset.

Method Parameters. The rates of atrous spatial pyramid pooling in DeepLabv3+ are set to 12, 24, and 36, and the output stride to 8 px. The spatial resolution of the output channels is doubled by bilinear interpolation, i.e., locations $\mathbf{u}$ for which the predictions are made are at the centers of 4×4 px regions of the input image. A single network is trained for all objects in a dataset, an object is represented by $n = 64$ surface fragments (unless stated otherwise), and the other parameters are set as follows: $\lambda_1 = 1$, $\lambda_2 = 100$, $\tau_a = 0.1$, $\tau_b = 0.5$, $\tau_d = 20$, $\tau_r = 4$ px, $\tau_i = 400$, $\tau_q = 0.5$, $\tau_t = 100$ px, $\tau_j = 0.8$, $\tau_p = 0.5$.

4.4.2 Main Results

Accuracy. Table 4.1 compares the performance of EPOS with the participants of the BOP Challenge 2019 [100, 106]. EPOS outperforms all RGB methods on all three datasets by a large margin in both AR and AR_{MSPD} scores. On the YCB-V dataset, it achieves 27% absolute improvement in both scores over the second-best RGB method and also outperforms all RGB-D and D methods. On the T-LESS and LM-O datasets, which include symmetric and texture-less objects, EPOS achieves the overall best AR_{MSPD} score. As the BOP rules require the method parameters to be fixed across datasets, Table 4.1 reports scores achieved with objects from all datasets represented by 64 fragments. As reported in Table 4.2, increasing the number of fragments from 64 to 256 yields in some cases additional improvements but around double image processing time. Note that we do not perform any post-refinement of the estimated poses, such as [159, 147, 277, 205].

Results of EPOS on the latest BOP Challenge 2020 [109] can be found in Table 7.3. EPOS was trained only on the synthetic training images provided for the 2020 edition

6D object pose estimation method	Image	T-LESS [102]		YCB-V [271]		LM-O [17]		Time
		AR	AR_*	AR	AR_*	AR	AR_*	
EPOS	RGB	**47.6**	**63.5**	**69.6**	**78.3**	**44.3**	**65.9**	0.75
Zhigang-CDPN-ICCV19 [149]	RGB	12.4	17.0	42.2	51.2	37.4	55.8	0.67
Sundermeyer-IJCV19 [237]	RGB	30.4	50.4	37.7	41.0	14.6	25.4	0.19
Pix2Pose-BOP-ICCV19 [188]	RGB	27.5	40.3	29.0	40.7	7.7	16.5	0.81
DPOD-ICCV19 (synthetic) [277]	RGB	8.1	13.9	22.2	25.6	16.9	27.8	0.24
Pix2Pose-BOP_w/ICP-ICCV19 [188]	RGB-D	–	–	**67.5**	**63.0**	–	–	–
Drost-CVPR10-Edges [63]	RGB-D	**50.0**	**51.8**	37.5	27.5	**51.5**	**56.9**	144.10
Félix&Neves-ICRA17-IET19 [214, 206]	RGB-D	21.2	21.3	51.0	38.4	39.4	43.0	52.97
Sundermeyer-IJCV19+ICP [237]	RGB-D	48.7	51.4	50.5	47.5	23.7	28.5	1.10
Vidal-Sensors18 [257]	D	**53.8**	**57.4**	**45.0**	**34.7**	**58.2**	**64.7**	4.93
Drost-CVPR10-3D-Only [63]	D	44.4	48.0	34.4	26.3	52.7	58.1	10.47
Drost-CVPR10-3D-Only-Faster [63]	D	40.5	43.6	33.0	24.4	49.2	54.2	2.20

Table 4.1. **BOP Challenge 2019** [100, 106] results on the T-LESS, YCB-V and LM-O datasets. Objects are represented by 64 surface fragments in EPOS. Top scores among methods using the same image channels are **bold**, the best overall scores are **blue**. AR_* represents AR_{MSPD}. The time (in seconds) is the average image processing time averaged over the three datasets.

of the challenge and all parameters were set as described above. Overall, EPOS placed 16th out of 26 methods. However, the better performing methods applied an RGB-based or depth-based post-refinement of the estimated poses, used also real training images, or trained one neural network per object instead of one network per dataset. As other methods, EPOS would likely benefit from these design choices as well. For example, the ICP post-refinement [217] improved the accuracy of the Pix2Pose method [188] by absolute 24.9% and teleported this method from the 22nd to the 4th place. The importance of a post-refinement stage was demonstrated also by other methods – top nine methods applied ICP or an RGB-based refiner, similar to DeepIM [147]. Moreover, adding real training images improved the accuracy on datasets T-LESS, TUD-L, and YCB-V[2] on average by 15.8% for the CosyPose method [139] and by 13.2% for the CDPN method [149] (compare rows #3 and #5 and rows #10 and #14 in Table 7.3). The top-performing CosyPose method identified strong augmentation of training images as one of the key components which could be used in EPOS as well. Experiments with a post-refinement stage, real training images and the strong augmentation are left for future work.

Speed. EPOS takes 0.75 s per image on average on datasets T-LESS, YCB-V, and LM-O (Table 4.1).[3] As the other RGB methods, which are all based on convolutional neural networks, EPOS is noticeably faster than the RGB-D and D methods, which are slower typically due to an ICP post-refinement stage. The RGB methods of [237, 277] are 3–4 times faster but significantly less accurate than EPOS. On the seven core datasets from

[2]The other datasets included in the challenge do not provide real training images.
[3]With a 6-core Intel i7-8700K CPU, 64GB RAM, and Nvidia P100 GPU.

the BOP Challenge 2020, EPOS takes 1.87 s per image on average (Table 7.3). This processing time is higher mainly because of the ITODD and IC-BIN datasets which include images with multiple object instances (the average and the maximum number of instances per image are 4.2 and 81 for ITODD and 12.3 and 19 for IC-BIN). Depending on the application requirements, the trade-off between the accuracy and speed of EPOS can be controlled by, e.g., the number of fragments, the network size, the image resolution, the density of pixels at which the correspondences are predicted, or the maximum number of GC-RANSAC iterations. Besides, our implementation could be further optimized.

4.4.3 Ablation Experiments

Surface Fragments. The accuracy scores of EPOS for different numbers of surface fragments are shown in the upper half of Table 4.2. With a single fragment, the method performs direct regression of the so-called 3D object coordinates [17], similarly to [121, 188, 149]. The accuracy increases with the number of fragments and reaches the peak at 64 or 256 fragments. On all three datasets, the peaks of both AR and AR_{MSPD} scores are 18–33% higher than the scores achieved with the direct regression of the 3D object coordinates. This significant improvement demonstrates the effectiveness of fragments on various types of objects, including symmetric, texture-less, and textured objects.

On T-LESS, the accuracy drops when the number of fragments is increased from 64 to 256. We suspect this is because the fragments become too small (T-LESS includes smaller objects) and training becomes challenging due to fewer examples per fragment.

The average number of potential correspondences increases with the number of fragments, i.e., each pixel is linked with more fragments (columns Corr. in Table 4.2). At the same time, the average number of fitting iterations tends to decrease (columns Iter.), i.e., a pose hypothesis with the quality q above threshold τ_q is found earlier. This shows that the fitting method can benefit from knowing multiple potential correspondences per pixel. However, although the average number of iterations decreases, the average image processing time tends to increase (with a higher number of fragments) due to a higher computational cost of the network inference and of each fitting iteration. Setting the number of fragments to 64 provides a practical trade-off between the speed and accuracy.

Regression of 3D Fragment Coordinates. The upper half of Table 4.2 shows scores achieved with regressing the precise 3D locations, while the lower half shows scores achieved with the same network models but using the fragment centers (Section 4.1) instead of the regressed locations. Without the regression, the scores increase with the number of fragments as the deviation of the fragment centers from the true corresponding 3D locations decreases. However, the accuracy is often noticeably lower than with the regression. With a single fragment and without the regression, all pixels are linked to the same fragment center and all samples of three correspondences are immediately rejected because they fail the non-collinearity test, hence the low processing time.

n	T-LESS [102]					YCB-V [271]					LM-O [17]				
	AR	AR_*	Corr.	Iter.	Time	AR	AR_*	Corr.	Iter.	Time	AR	AR_*	Corr.	Iter.	Time
With regression of 3D fragment coordinates															
1	17.2	30.7	911	347	0.97	41.7	52.6	1079	183	0.56	26.8	47.5	237	111	0.53
4	39.5	57.1	1196	273	0.95	54.4	66.1	1129	110	0.52	33.5	56.0	267	58	0.51
16	45.4	62.7	1301	246	0.96	63.2	72.7	1174	71	0.51	39.3	61.3	275	54	0.50
64	**47.6**	**63.5**	1612	236	1.18	69.6	78.3	1266	56	0.57	44.3	**65.9**	330	53	0.49
256	45.6	59.7	3382	230	2.99	**71.4**	**79.8**	1497	56	0.94	**46.0**	65.4	457	70	0.60
Without regression of 3D fragment coordinates															
1	0.0	0.0	911	400	0.23	0.0	0.0	1079	400	0.17	0.0	0.0	237	400	0.24
4	3.2	8.8	1196	399	0.89	3.0	7.4	1129	400	0.53	5.2	15.2	267	390	0.50
16	13.9	37.5	1301	396	1.02	16.1	36.4	1174	400	0.61	17.1	47.7	275	359	0.55
64	29.4	55.0	1612	380	1.35	41.5	66.6	1266	383	0.73	31.0	62.3	330	171	0.55
256	43.0	58.2	3382	299	2.95	64.5	77.7	1497	206	0.88	43.2	64.9	457	72	0.58

Table 4.2. **Number of fragments and regression.** The table reports the Average Recall scores for different numbers of surface fragments (n) with and without regression of the 3D fragment coordinates (the fragment centers are used in the case of no regression). AR_* represents AR_{MSPD}. The table also reports the average number of correspondences established per object model in an image, the average number of GC-RANSAC iterations to fit a single pose (both are rounded to integers), and the average image processing time (in seconds).

RANSAC variant	Non-minimal solver	T-LESS [102]		YCB-V [271]		LM-O [17]		Time
		AR	AR_*	AR	AR_*	AR	AR_*	
OpenCV RANSAC	EPnP [145]	35.5	47.9	67.2	76.6	41.2	63.5	0.16
MSAC [250]	EPnP [145] + L-M [171]	44.3	61.0	63.8	73.7	39.7	61.7	0.49
GC-RANSAC [8]	DLS-PnP [92]	44.3	59.5	67.5	76.1	35.6	53.9	0.53
GC-RANSAC [8]	EPnP [145]	46.9	62.6	69.2	77.9	42.6	63.6	0.39
GC-RANSAC [8]	EPnP [145] + L-M [171]	**47.6**	**63.5**	**69.6**	**78.3**	**44.3**	**65.9**	0.52

Table 4.3. **RANSAC variants and non-minimal solvers.** The P3P solver [134] is used to estimate the pose from a minimal sample of 2D-3D correspondences. The non-minimal solvers are applied when estimating the pose from a larger-than-minimal sample. The reported time (s) is the average time to fit poses of all object instances in an image averaged over the datasets.

Even though the regressed locations are not guaranteed to lie on the model surface, their average distance from the surface was measured to be less than 1 mm (with 64 and 256 fragments), which is negligible compared to the object sizes. No improvement was observed when the regressed locations were replaced by the closest points on the surface.

Robust Pose Fitting. Table 4.3 evaluates several methods for robust pose estimation from 2D-3D correspondences: RANSAC [71] from OpenCV (function `solvePnPRansac`), MSAC [252], and GC-RANSAC [8]. The methods were evaluated within the Progressive-X scheme (Section 4.3.2), with the P3P solver [134] to estimate the pose from a minimal sample, i.e., three correspondences, and with several solvers to estimate the pose from a

non-minimal sample. In OpenCV RANSAC and MSAC, the non-minimal solver refines the pose from all inliers. In GC-RANSAC, it is additionally used in the graph-cut-based local optimization which is applied when a new so-far-the-best pose is found. We tested OpenCV RANSAC with all available non-minimal solvers and achieved the best scores with EPnP [145]. The top-performing method on all datasets is GC-RANSAC with EPnP followed by the Levenberg-Marquardt optimization [171] as the non-minimal solver. Note the improvement in accuracy, especially on T-LESS, over OpenCV RANSAC which is used by many recent correspondence-based methods, e.g., [188, 277, 205, 242].

Chapter 5

ObjectSynth

Synthesis of Photorealistic Training Images

This chapter presents an approach to synthesize highly photorealistic images of 3D object models, which are experimentally shown effective for training deep neural networks for detecting the objects and estimating their pose from real test images. The proposed approach has three key ingredients. First, 3D models of objects are arranged in 3D models of complete indoor scenes with realistic materials and lighting. Second, plausible geometric configurations of objects and cameras in the scenes are generated by physics simulation. And, third, a high degree of visual realism is achieved by physically-based rendering (PBR). Example images synthesized by the proposed approach are in Figure 5.1.

The experiments show that the Faster R-CNN object detector [208] trained on the synthesized images achieves a 24% absolute improvement in mAP@.75IoU on real test images from the RU-APC dataset [209], and 11% on real test images from the LM-O dataset [17, 94]. The improvements are relative to a baseline where the training images are synthesized by rendering object models on top of random photographs – similar images are commonly used for training methods for tasks such as object instance detection [65], object instance segmentation [97], and 6D object pose estimation [127, 205].

Modeling of objects and scenes is described in Section 5.1, arranging the object models in the scene models and generating camera poses in Section 5.2, and rendering in Section 5.3. The effectiveness of the photorealistic training images is evaluated in Section 5.4. A refined version of the proposed approach, which was used to synthesize training images for participants of the BOP Challenge 2020 (Chapter 7), is described in Section 5.5.

Tomáš Hodaň was working on ObjectSynth during his internship at Microsoft Research in Redmond in 2018. The work was published in [111]. The project website with a dataset of 400K photorealistic images is available at: `thodan.github.io/objectsynth`, and a dataset of 350K images synthesized for the BOP Challenge 2020 at: `bop.felk.cvut.cz`.

5.1 Scene and Object Modeling

Achieving a high degree of visual realism requires 3D models that are accurate in terms of geometry, texture, and material. We worked with 3D models of 15 objects from the Linemod (LM) dataset [94] and 14 objects from the Rutgers APC (RU-APC) dataset [209]

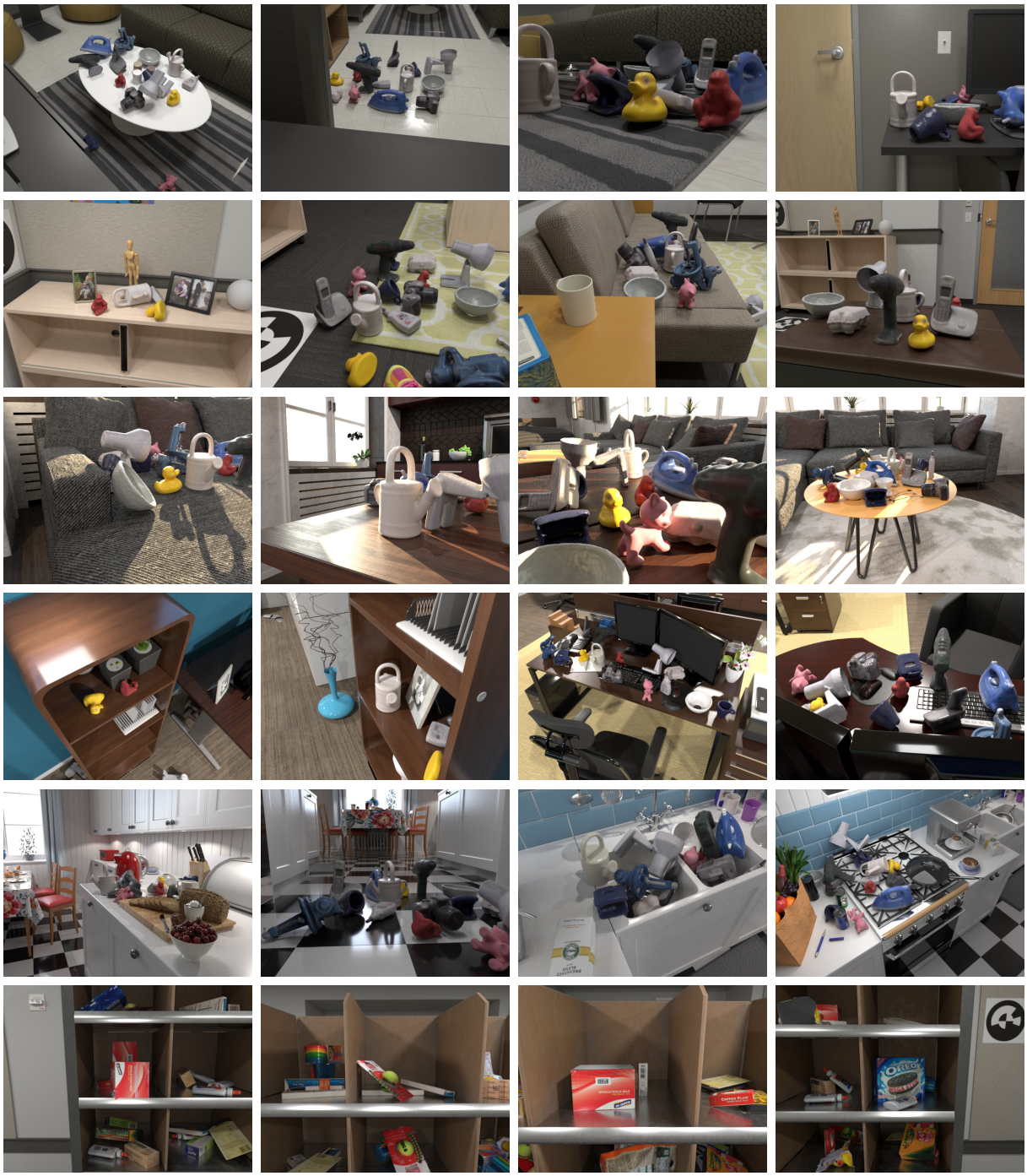

Figure 5.1. **Example images synthesized by ObjectSynth.** The top five rows show images of the LM objects [94], the bottom row shows images of the RU-APC objects [209]. The images are annotated with 2D bounding boxes, masks and 6D poses of the objects.

(Figure 5.2, top). We started with the refined versions of the object models from BOP [106], provided as color meshes with surface normals, and manually assigned PBR materials to the models which match the appearance of the objects in real images. The material prop-

erties were controlled by the roughness, specular, and metallic parameters.[1] Since objects from both datasets are mostly texture-less and of a homogeneous material, the same material properties were assigned to the whole model surface. A Lambertian material was assigned to the RU-APC models as the objects are mostly made of cardboard.

The 3D object models were arranged in 3D models of six furnished scenes (Figure 5.2, bottom). Scenes 1–5 represent the office and household environments and include fine details and typical objects, e.g., the kitchen (Scene 5) contains dishes in a sink or a bowl of cherries. Scene 6 contains a shelf from the Amazon Picking Challenge 2015 [274].

The scene models were created using standard 3D tools, primarily Autodesk Maya, by artists at Microsoft Research. Scenes 1 and 2 are reconstructions of real-world environments obtained using LiDAR and photogrammetry 3D scans which served as a guide for artists. Materials were recreated using photographic reference, PBR material scanning [175], and color swatch samples [180]. Scenes 3–5 were purchased online [69], their geometry and materials were refined, and clutter and chaos was added to mimic a real environment. A 3D geometry model of the shelf in Scene 6 was provided in the Amazon Picking Challenge 2015 [274]. Reference imagery of the shelf was used to create textures and materials that match its appearance. Exterior light coming to the scene through windows was modeled with Arnold Physical Sky [80] which can accurately depict atmospheric effects and time-of-day variation. Interior lights were modeled with standard light sources such as area and point lights.

5.2 Scene and Object Composition

Multiple stages to arrange the objects on were manually selected in each scene. A stage is defined by a polygon and is typically located on tables, chairs and other places with a distinct structure, texture or illumination. Placing objects on such locations maximizes the diversity of the rendered images. One stage per shelf bin was created in Scene 6.

An arrangement of a set of objects was generated in two steps: (1) poses of the 3D object models were initialized above a randomly selected stage, and (2) a physically plausible arrangement was reached by simulating the objects falling on the stage under gravity and undergoing mutual collisions. The poses were initialized by FLARE [76], a rule-based system for generation of object layouts for augmented reality applications, and physics was simulated by NVIDIA PhysX. The initial height of the objects above the stage was randomly sampled from 5 to 50 cm. For the LM objects, one instance per object model was added to the stage and initialized with the canonical orientation, i.e., the cup was up-

[1] The roughness parameter controls the range of angles in which the reflected light is scattered, which influences the sharpness of the reflection. The specular parameter controls the facing (along normal) reflectivity. The metallic parameter controls the blending factor of the metallic and non-metallic material models. In the case of a metallic material, the base color controls the color of the specular reflection and most light is reflected as specular reflections. Non-metallic materials have specular reflections that are the same color as the incoming light and barely reflects when looking at the surface face-on [66].

Figure 5.2. **3D models of objects and scenes.** Objects from the LM dataset [94] (1st row) were rendered in Scenes 1–5 and objects from RU-APC [209] (2nd row) in Scenes 3 and 6.

right, the cat was standing on her legs, etc. For the RU-APC objects, up to five instances per object model were added to the stage and initialized with random orientations.

Multiple cameras were positioned around each object arrangement. Instead of fitting all objects within the camera frustum, the camera was pointed at a randomly selected object. This allowed for better control of the scale of the rendered objects. The azimuth angles, elevation angles, and distances of the cameras were uniformly sampled from ranges determined by the ground-truth 6D object poses from the test images. A mask of the focused object and a mask of its visible part were rendered before rendering the RGB image. The RGB image was rendered only if at least 30% of the object was visible.

Quality	AA	D rays	S rays	D depth	S depth	Max depth
Low	1	1	1	1	1	2
Medium	9	4	4	3	2	3
High	25	9	4	3	3	4
BlenderProc4BOP	50	1	1	3	0	3

Table 5.1. **Ray-tracing parameters [80] for different quality settings.** AA is the number of rays sampled per image pixel. D/S rays is the number of indirect diffuse/specular rays sampled when a ray from the camera hits a diffuse/specular surface. After the first reflection, only a single indirect diffuse/specular ray is sampled. D/S depth is the maximum number of diffuse/specular reflections, and Max depth is the maximum number of reflections in total. Besides indirect rays that reflect from other scene elements, the color of a surface is determined by direct rays from light sources. BlenderProc4BOP is discussed later in Section 5.5.

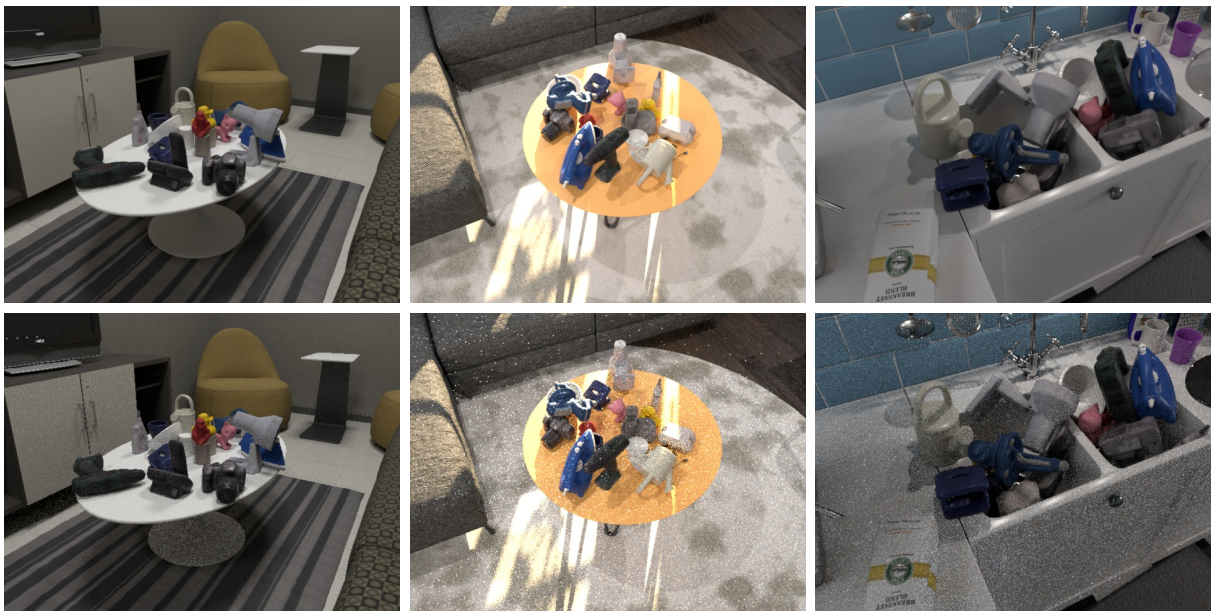

Figure 5.3. **High vs. low rendering quality.** The same images rendered at the high (top) and the low (bottom) quality settings of the Arnold physically-based renderer [80]. Notice the noise due to insufficient illumination sampling in the low-quality images.

5.3 Physically-Based Rendering

Besides accurate 3D models, achieving a high degree of visual realism requires accurate simulation of lighting, including soft shadows, reflections, refractions and indirect light bounces. For the experiments in Section 5.4, the images were synthesized by Arnold [80], a physically-based renderer simulating the flow of light energy in the scene by ray tracing.

Three images were rendered per camera in a low, medium and high quality settings. The mapping of the settings to Arnold parameters is in Table 5.1. Increasing the number of rays sampled per pixel (AA) reduces aliasing artifacts due to insufficient sampling of

geometry and noise due to insufficient sampling of illumination. Increasing the number of indirect diffuse rays sampled when a ray from the camera hits a diffuse surface (D rays) and and from a specular surface (S rays) reduces illumination noise. Increasing the maximum number of diffuse and specular reflections (D/S depth) improves the accuracy of the illumination by gathering more of the light energy bounced around in the scene.

The images were rendered on 16-core Intel Xeon 2.3GHz processors with 112GB RAM. GPU's were not used. The average rendering time per image in VGA resolution was 15 s in low, 120 s in medium, and 720 s in high quality. A CPU cluster with 400 nodes allowed us to render 2.3M images in low, 288K in medium, and 48K in high quality within a day.

A set of 200K images in low and 200K images in high quality is available on the project website. LM objects are rendered in Scenes 1–5 and RU-APC objects in Scenes 3 and 6. Each instance is annotated with the 2D bounding box, segmentation mask and 6D pose.

5.4 Experiments

The experiments presented in this section evaluate the effectiveness of PBR images for training the Faster R-CNN object detector [208]. Specifically, the experiments focus on three aspects: (1) the importance of PBR images over the commonly used images of objects rendered on top of random photographs, (2) the importance of the high PBR quality, and (3) the importance of accurately modeling the scene context.

5.4.1 Experimental Setup

Datasets. The experiments were conducted on the reduced versions [106] of datasets Linemod-Occluded (LM-O) [17, 94] and Rutgers APC (RU-APC) [209]. The datasets include 3D object models and real RGB-D test images of VGA resolution (only RGB channels were used). The ground-truth 2D bounding boxes for the evaluation of 2D object detection were calculated from the provided ground-truth 6D object poses. LM-O contains 200 images with ground-truth annotations for 8 LM objects captured with various levels of occlusion. RU-APC contains 14 object models and 1380 images which show the objects in a cluttered warehouse shelf. Example test images are in Figure 5.5.

"Render & Paste" Training Images (R&P). The baseline images were generated by a "render & paste" pipeline similar to [97] – the 3D object models were rendered by OpenGL on top of random real photographs pulled from NYU Depth Dataset V2 [227]. For a more direct comparison, the R&P images were generated by imitating the PBR images – the 3D object models were rendered in the same poses as in the PBR images. Generating one R&P image took 3 s on average. Examples are in Figure 5.4.

Object Instance Detection. The experiments were conducted with two network architectures of Faster R-CNN: ResNet-101 [90] and Inception-ResNet-v2 [238]. The networks

Figure 5.4. **Types of training images.** The baseline "render & paste" training images (top) were generated by rendering the 3D object models in the same poses as in the PBR images (bottom) and pasting the renderings on top of random photographs.

were pre-trained on the Microsoft COCO dataset [151] and fine-tuned on synthetic images for 100K iterations. The learning rate was set to 0.001 and multiplied by 0.96 every 1K iterations. To virtually increase diversity of the training set, the images were augmented by randomly adjusting brightness, contrast, hue, and saturation, and by applying random Gaussian noise and blur, similarly to [97]. The implementation of Faster R-CNN from Tensorflow Object Detection API [116] was used. The detection accuracy was measured by mAP@.75IoU, i.e., the mean average precision with a strict IoU threshold of 0.75 [151]. For each test image, detections of object models annotated in the image were considered.

5.4.2 Importance of PBR Images for Training

On RU-APC, Faster R-CNN with Inception-ResNet-v2 trained on the high-quality PBR images achieves a significant 24% absolute improvement of mAP@.75IoU over the same method trained on R&P images (Table 5.3, PBR-h vs. R&P). It is noteworthy that PBR images yield almost 35% or higher absolute improvement on five objects, and overall achieve a better performance on 12 out of 14 objects. This is achieved when the objects are rendered in Scene 6 that accurately imitates the scene visible in test images. When the objects are rendered in Scene 3 (PBR-ho vs. R&P), the accuracy score still increases by 11% (the importance of scene context is discussed later in Section 5.4.4). Improvements, although not so dramatic, can also be observed with the ResNet-101 network. On LM-O, PBR images yield almost 11% absolute improvement, with a large gain on 7 out of 8 objects (Table 5.2, PBR-h vs. R&P).

	1	5	6	8	9	10	11	12	mAP
Inception-ResNet-v2									
PBR-h	**60.3**	**44.5**	**56.7**	**53.4**	**81.8**	**48.6**	9.6	**92.3**	**55.9**
PBR-l	57.3	35.8	53.3	52.6	77.8	23.8	3.1	94.5	49.8
R&P	30.7	45.4	42.5	32.4	77.1	33.4	**19.6**	76.7	44.7
ResNet-101									
PBR-h	**46.3**	**40.3**	**48.5**	**58.0**	**76.4**	**39.5**	4.7	**85.5**	**49.9**
PBR-l	44.1	26.6	41.6	53.7	73.7	24.5	1.1	91.6	44.6
R&P	35.5	45.3	37.1	44.6	75.0	33.6	**12.7**	76.8	45.1

Table 5.2. **Object detection scores on LM-O:** Per-object average precision (AP@.75IoU) and mean average precision (mAP@.75IoU) of Faster R-CNN trained on high/low PBR images of objects in Scenes 1–5 (PBR-h, PBR-l), and baseline R&P images. Object ID's as in [106].

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	mAP
Inception-ResNet-v2															
PBR-h	57.1	93.3	**88.0**	61.2	80.4	**62.5**	**99.0**	98.1	**73.2**	**44.8**	65.4	**70.9**	86.8	**26.4**	71.9
PBR-l	**57.6**	**96.3**	84.6	**62.2**	**81.3**	60.5	98.7	**98.6**	73.1	**44.8**	**79.3**	67.2	**90.5**	25.6	**72.9**
PBR-ho	46.2	61.9	56.0	55.8	54.4	69.8	89.0	89.3	81.6	21.5	72.7	58.3	43.3	21.7	58.7
R&P	33.5	47.5	71.5	32.7	42.4	13.5	44.9	73.0	57.4	44.5	47.6	35.8	87.6	40.6	48.0
ResNet-101															
PBR-h	30.6	**93.7**	**91.6**	**68.2**	72.1	56.7	93.4	**93.6**	**75.2**	**42.6**	**84.5**	60.9	73.2	21.4	**68.4**
PBR-l	26.8	87.6	87.3	64.0	**79.8**	27.8	**95.2**	90.4	66.2	37.5	83.1	**61.4**	**79.3**	**25.3**	65.1
PBR-ho	**35.2**	64.4	58.4	52.9	46.7	53.0	71.5	73.8	69.3	32.2	66.2	51.8	28.3	19.3	51.6
R&P	29.1	38.5	82.0	59.2	52.4	**59.1**	79.5	75.0	36.4	36.8	75.1	50.6	48.5	14.8	52.7

Table 5.3. **Object detection scores on RU-APC:** Per-object average precision (AP@.75IoU) and mean average precision (mAP@.75IoU) of Faster R-CNN trained on high/low-quality PBR images of in-context objects in Scene 6 (PBR-h, PBR-l), high-quality PBR images of out-of-context objects in Scene 3 (PBR-ho), and baseline images (R&P). Object ID's as in BOP [106].

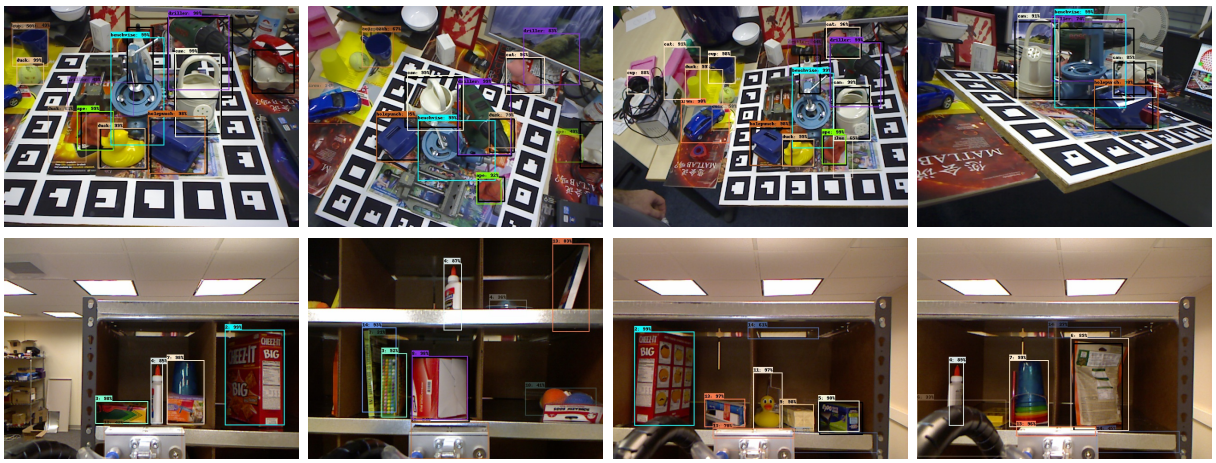

Figure 5.5. **Object detections.** The Faster R-CNN object detector was trained on high-quality PBR images and evaluated on real test images from LM-O (top) and RU-APC (bottom).

5.4.3 Importance of PBR Quality

On LM-O, Faster R-CNN with Inception-ResNet-v2 trained on the high-quality PBR images achieves an improvement of almost 6% over the low-quality PBR images (Table 5.2, PBR-h vs. PBR-l). A similar improvement is not observed on RU-APC when training on PBR images rendered in Scene 6. The illumination in Scene 6 is simpler, there is no incoming outdoor light and the materials are mainly Lambertian. There are therefore no complex reflections and the low-quality PBR images from this scene are cleaner than, e.g., when rendered in Scenes 3–5. This suggests that the low quality is sufficient for scenes with simpler illumination and materials.

5.4.4 Importance of Scene Context

To analyze the importance of accurately modeling the scene context, the RU-APC objects were rendered in two setups: (1) "in context" in Scene 6, and (2) "out of context" in Scene 3 (Figure 5.6). Following the taxonomy of contextual information from [56], the in-context setup faithfully imitates the gist, geometric, semantic, and illumination aspects of the context of the test scene. The out-of-context setup exhibits discrepancies in all of these aspects. Training images of in-context objects yield an absolute improvement of 13% with Inception-ResNet-v2 and 16% with ResNet-101 over the images of out-of-context objects (Table 5.3, PBR-h vs. PBR-ho). This demonstrates the benefit of accurately modeling the context of the test scene.

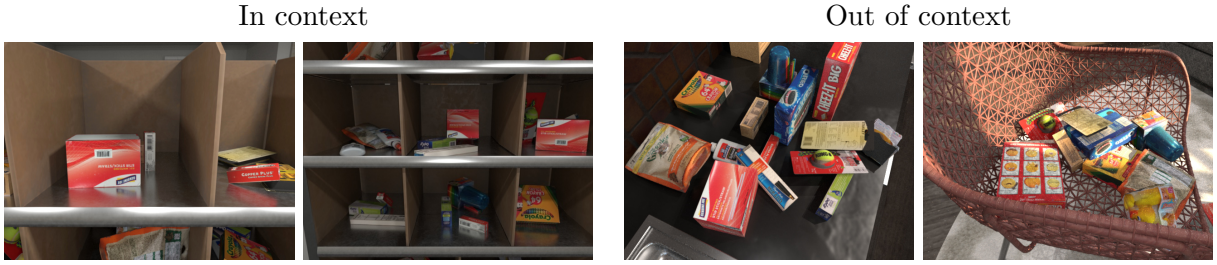

Figure 5.6. **Images of in-context and out-of-context RU-APC objects.** The in-context images are rendered in Scene 6 which accurately models the scene in test images.

5.5 Training Images for BOP Challenge 2020

In the BOP Challenge 2020 (Section 7.4), the participants were provided with 350K PBR training images generated and automatically annotated by BlenderProc4BOP [55, 54], an open-source and light-weight physically-based renderer which implements a refined version of the synthesis approach proposed above.

To improve efficiency, BlenderProc4BOP renders objects not in 3D models of indoor scenes but inside an empty cube, with objects from other datasets serving as distractors.

Figure 5.7. **Synthetic training images.** Methods for 6D object pose estimation have been commonly trained on “render & paste” images synthesized by OpenGL rendering of 3D object models randomly positioned on top of random backgrounds. Instead, participants of the BOP Challenge 2020 were provided 350K PBR training images rendered by ray tracing and showing the 3D object models in physically plausible poses inside a cube with a random PBR material.

To achieve a rich spectrum of generated images, a random PBR material from the CC0 Textures library [52] is assigned to the walls of the cube, and light with a random intensity and color is emitted from the room ceiling and from a randomly positioned point source. As shown in Table 5.3, the ray-tracing parameters are set economically, with 50 rays sampled per pixel but only a single indirect ray sampled when a ray from the camera hits a surface. Noise in the rendered images is reduced by the Intel Open Image Denoiser [118]. This setup keeps the computational cost low – the full generation of one RGB-D image in VGA resolution takes **only 1–3 seconds** on a standard desktop computer with a modern GPU. A set of 50K images can be therefore rendered on 5 GPU’s overnight.

Instead of trying to accurately model the object materials, the specular, metallic, and roughness properties are randomized. Realistic object poses are achieved by dropping the 3D object models on the ground plane of the cube using the PyBullet physics engine integrated in Blender [44]. This allows creating dense but shallow piles of objects that introduce various levels of occlusion. Since test images from the LM dataset show the objects always standing upright, the objects from LM are not dropped but instead densely placed on the ground plane in upright poses using automated collision checks. Each object

arrangement is rendered from 25 random camera poses. The azimuth angles, elevation angles, and distances of the cameras are uniformly sampled from ranges determined by the ground-truth 6D object poses from the test images, and the in-plane rotation angles are generated randomly. Examples of the generated images are in Figure 5.7.

The data generated by BlenderProc4BOP (RGB and depth images, ground-truth 6D object poses, camera intrinsics) is saved in the BOP format, allowing to interface with utilities from the BOP toolkit [108]. The source code and configuration files to reproduce or modify the generation process are publicly available [15]. Users are encouraged to build on top of the open-source code and release their extensions.

In the BOP Challenge 2020, methods achieved noticeably higher accuracy scores when trained on the PBR training images generated by BlenderProc4BOP than when trained on the "render & paste" images. Although adding real training images yielded even higher scores, competitive results were achieved with PBR images only – out of the 26 evaluated methods, the fifth was trained only on the PBR images. See Section 7.4 for details.

Chapter 6

T-LESS

An RGB-D Dataset with Texture-less Objects

This chapter introduces T-LESS, a publicly available dataset for 6D object pose estimation that includes 3D models and training and test RGB-D images of thirty commodity electrical parts. Recognizing and estimating the 6D pose of these objects is challenging because the objects have no significant texture or discriminative color, exhibit symmetries and similarities in shape and size, and some objects are a composition of others (Figures 6.1 and 6.2). Objects exhibiting similar properties are common in industrial environments and T-LESS is the first dataset to include such objects.

The training and test images were captured with a triplet of sensors, i.e., a structured light RGB-D sensor (Primesense Carmine 1.09), a time-of-flight RGB-D sensor (Microsoft Kinect v2), and an RGB sensor (Canon IXUS 950 IS). The sensors were time-synchronized and had similar perspectives. The images were captured with an automated acquisition setup which systematically sampled images from a view sphere, resulting in ~39K training and ~10K test images from each sensor. The training images show objects in isolation on a black background, while the test images originate from twenty table-top scenes with arbitrarily arranged objects. Complexity of the test scenes varies from those with several isolated objects and a clean background to very challenging ones with multiple instances of several objects and with a high amount of occlusion and clutter. Additionally, the dataset contains two types of 3D mesh models for each object – one manually created in CAD software and one semi-automatically reconstructed from the training RGB-D images. All instances of the modeled objects in the training and test images are annotated with accurate ground-truth 6D poses – see the visualizations in Figure 6.1 for a qualitative and Section 6.7.1 for a quantitative evaluation of the ground-truth poses.

The dataset is intended for evaluating various flavors of the 6D object pose estimation problem and other related problems such as 2D object detection [248, 101] and segmentation [247, 79]. Since images from three types of sensors are available, the dataset allows to study the importance of different input modalities for the aforementioned problems. Another possibility is to use the training images for evaluating 3D object reconstruction methods [232] where the provided CAD models can serve as the ground truth.

The objectives in designing T-LESS were to provide a dataset of substantial but manageable size, with a rigorous and complete ground-truth annotation that is accurate to

Figure 6.1. **Challenges in T-LESS.** Recognizing and estimating the 6D pose of T-LESS objects is challenging because the objects are texture-less, without a discriminative color, exhibit symmetries and similarities in shape and size, and are often seen under heavy occlusion and clutter. Cropped test images (top) are shown overlaid with colored 3D object models in the ground-truth 6D poses (bottom). Instances of the same object have the same color.

the level of the sensor resolution, and with graded complexity that makes the dataset reasonably future-proof, i.e., solvable but not solved by the current state-of-the-art methods. The difficulty of T-LESS is demonstrated in Section 6.7.2 on the performance of the HashMatch method, which achieves a relatively low accuracy on T-LESS compared to the accuracy on the well-established LM dataset (Section 3.3.1). T-LESS was selected as one of the core datasets in the BOP benchmark (Chapter 7) and, as of 2020, it is still one of the more difficult datasets in the benchmark.

The applied approach to capture the training and test images, annotate the images with ground-truth 6D object poses, and reconstruct the 3D object models is described in Sections 6.1–6.6. The accuracy of the ground-truth poses and the difficulty of the dataset is assessed in Section 6.7. The T-LESS dataset was published in [102] and is available on the project website: `cmp.felk.cvut.cz/t-less`.

6.1 Acquisition Setup

The training and test images were captured with the aid of a setup shown in Figure 6.3. The objects were placed on a turntable and the sensors were attached to a jig with adjustable tilt. Poses of the sensors were estimated using a marker field that was affixed to the turntable. The marker field was extended vertically to the turntable sides to facilitate estimating the poses at lower elevations. When capturing training images, the objects

Figure 6.2. **T-LESS objects and scenes.** The dataset includes 3D models and training images of 30 objects (top) and test images of 20 scenes (bottom) – shown overlaid with colored 3D object models in the ground truth poses. The images were captured from a systematically sampled view sphere around an object/scene and are annotated with accurate ground-truth 6D poses of all instances of the modeled objects. Some objects are a composition of others. For example, objects 7 and 8 are built up from object 6, object 9 is made of three copies of object 10, and the center part of objects 17 and 18 is nearly identical to object 13.

were placed in the middle of the turntable and in the front of a black screen that ensured a uniform background from all elevations. To introduce a non-uniform background in test images, a sheet of plywood with markers at its edges was placed on the turntable. In some scenes, the objects were placed on other objects such as books to invalidate a ground plane assumption that might be made by some methods. The depth of object surfaces in the training and test images is in the range of 0.53–0.92 m, which is within the sensing ranges of the RGB-D sensors: 0.35–1.4 m for Carmine and 0.5–4.5 m for Kinect.

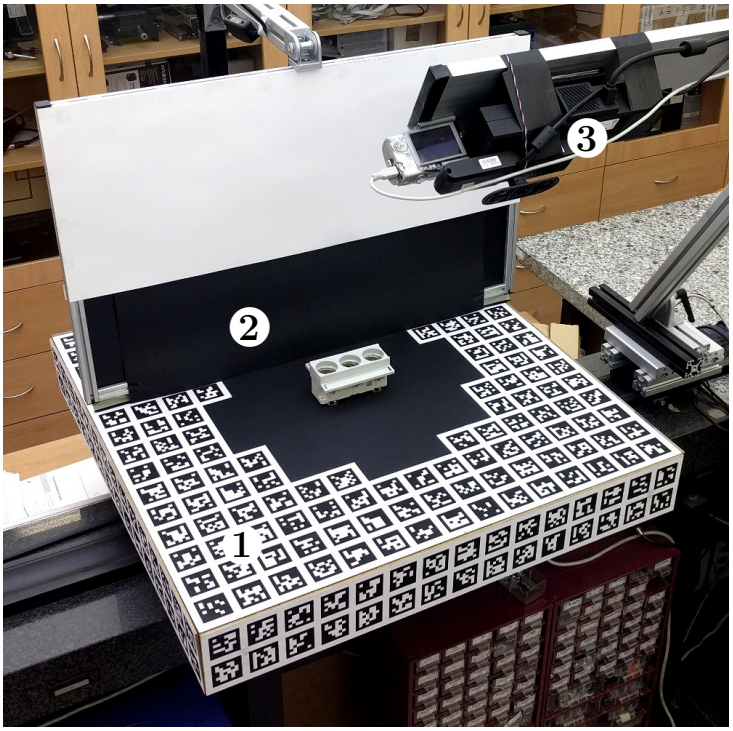

Figure 6.3. **Acquisition setup:** (1) a turntable with a marker field, (2) a screen ensuring a black background in training images (removed when capturing test images), (3) a triplet of sensors attached to a jig with adjustable tilt.

6.2 Calibration of Sensors

The intrinsic and distortion parameters of the sensors were estimated with the standard checkerboard-based procedure from OpenCV [19]. High calibration accuracy is affirmed by low values of the root-mean-square re-projection error calculated at the checkerboard corners: 0.51 px for Carmine, 0.35 px for Kinect, and 0.43 px for Canon. For the RGB-D sensors, the calibration was performed with the RGB images. The depth images were aligned to the RGB images by the factory depth-to-color registration from the official SDK's (OpenNI 2.2 and Kinect for Windows SDK 2.0). The color and aligned depth images, which are included in the dataset, are already processed to remove radial distortion.

All sensors were synchronized and extrinsically calibrated with respect to the turntable to enable registration of any pair of images. Since the images were taken while the turntable was moving, precise synchronization was essential to ensure that the images from the three sensors are taken from similar viewpoints. Poses of the sensors were estimated using fiducial BCH-code markers from ARToolKitPlus [258]. Specifically, the 2D detections of particular markers in an image combined with the knowledge of their physical location on the turntable provided a set of 2D-3D correspondences. The sensor pose was then estimated from the correspondences by the PnP-RANSAC algorithm [71, 145] and refined by a non-linear minimization of the cumulative re-projection error [154].

The root-mean-square re-projection error, which was calculated at the marker corners in all test images, is 1.27 px for Carmine, 1.37 px for Kinect, and 1.50 px for Canon. This measure combines errors in the intrinsic calibration, marker field detection and pose estimation, and is therefore larger than the aforementioned error of the intrinsic calibration.

6.3 Training and Test Images

The dataset includes training images of every object in isolation captured from a full view sphere with the acquisition setup described in Section 6.1. The training images were captured from elevation angles $-85°$ to $85°$ with a $10°$ step and from the complete range of azimuth angles with a $5°$ step. Views from the upper and lower hemispheres were taken separately, turning the object upside down in between. In total, the dataset includes $18 \cdot 72 = 1296$ training images per object from each sensor. Exceptions are objects 19 and 20 which were captured only from the upper hemisphere (648 images from elevation $85°$ to $5°$). These objects are horizontally symmetric and the views from the upper hemisphere are therefore sufficient. The test scenes were captured from elevation angles $15°$ to $75°$ with a $10°$ step and from the complete range of azimuth angles with a $5°$ step. In total, the dataset includes $7 \cdot 72 = 504$ test images per test scene from each sensor.

The captured images were cropped to remove regions at the image borders, which mostly showed the fiducial markers and were irrelevant for 6D object pose estimation. To hide markers that were still visible in the training images after the cropping, and therefore to ensure a black background everywhere around the objects, the training images were gradually darkened from the object mask towards the image borders.

The provided training RGB-D images from Carmine and Kinect are of resolution 400×400 px, the training RGB images from Canon of 1900×1900 px, the test RGB-D images from Carmine and Kinect of 720×540 px, and the test RGB images from Canon of 2560×1920 px. Example images from the three sensors are in Figure 6.4.

6.4 Depth Correction

Similarly to [72, 233], the depth measurements from the RGB-D sensors were observed to exhibit a systematic error. To eliminate the error, the depth measurements were collected at image locations of the marker corners together with their expected values calculated from the known marker coordinates. The measurements were collected from the range of $0.53 - 0.92$ m in which the objects appear in the training and test images. The following correction models were then found by least-squares fitting: $d_c = 1.0247 \cdot d - 5.19$ for Carmine, and $d_c = 1.0266 \cdot d - 26.88$ for Kinect, where d and d_c are the measured and corrected depth values in millimeters. The correction reduced the mean absolute difference from the expected depth from 12.4 to 2.8 mm for Carmine and from 7.0 to 3.6 mm for Kinect. The correction is already applied to all depth images included in the dataset.

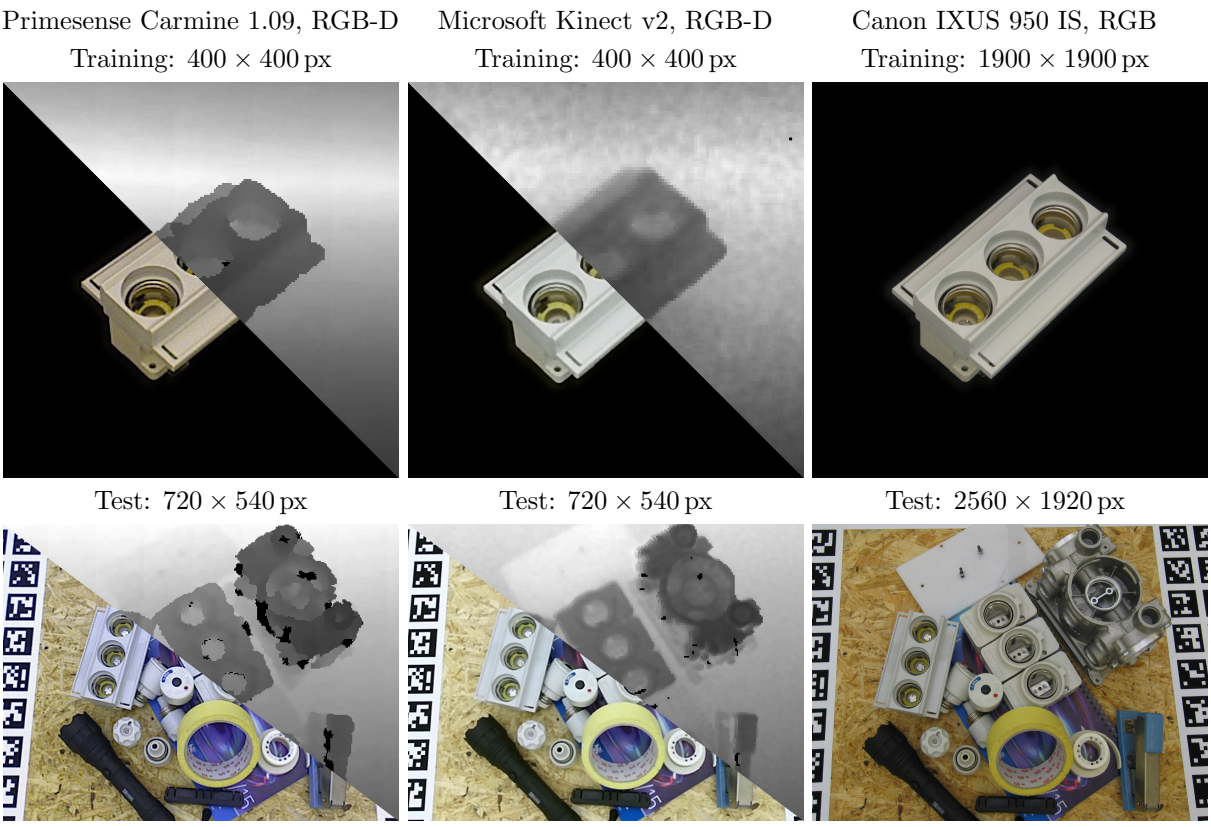

Figure 6.4. **Training and test images captured by three sensors.** For the RGB-D images, the left halves show the RGB channels and the right halves show the depth channel.

6.5 3D Object Models

A manually-created CAD model and a semi-automatically reconstructed model are available for each object (Figure 6.5). Models of both types are provided in the form of 3D meshes with surface normals. The surface normal at a mesh vertex was calculated by MeshLab [40] as the angle-weighted sum of face normals incident to the vertex [243]. The surface color, stored at the mesh vertices, is available only in the reconstructed models.

The reconstructed models were obtained using the volumetric 3D mapping method by Steinbrücker et al. [232]. The input to the method was the RGB-D training images from Carmine and the camera poses estimated from the fiducial markers (Section 6.2). For each object, two partial models were reconstructed first, one from the upper and one from the lower view hemisphere. The partial models were aligned by the Iterative Closest Point (ICP) algorithm applied to the model vertices, followed by manual refinement to ensure correct registration of surface details that are visible only in color. The camera poses were then brought into a common reference frame by the found alignment transformation and the full object model was reconstructed using images from all cameras. The models contained some minor artifacts, e.g., small spikes, which were removed manually. Some of the objects have small shiny metal parts, such as the plug poles, which are not

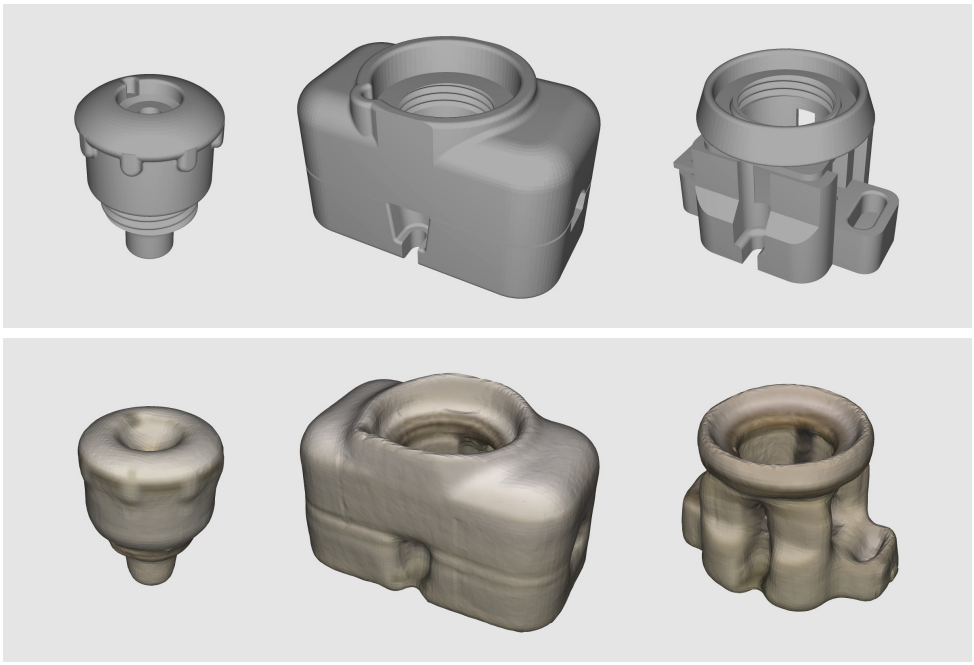

Figure 6.5. **Examples of 3D object models.** Top: Manually created CAD models. Bottom: Semi-automatically reconstructed models which include also surface color. Surface normals at model vertices are included in both model types.

reconstructed because their depth was not reliably captured by the depth sensor.

The reconstructed models were aligned to the CAD models by the ICP algorithm followed by manual refinement. Models of both types are therefore defined in the same coordinate system and share the same ground-truth 6D poses. The origin of the model coordinate system coincides with the center of the 3D bounding box of the CAD model.

The geometrical similarity of the two model types was assessed by calculating the average distance from vertices of the reconstructed models to the closest points on the surface of the CAD models. The average distance over all object models is 1.01 mm, which is very low compared to the size of the objects that ranges from 58.13 mm for object 13 to 217.16 mm for object 8. The Metro software [41] was used to measure the distance.

6.6 Ground-Truth 6D Object Poses

To collect the ground-truth 6D object poses for images of a test scene, a dense 3D model of the scene was first reconstructed from all images of the scene by the method of Steinbrücker et al. [232]. The CAD object models were then manually aligned to the scene model, the object models were rendered into several high-resolution images from the Canon camera using known camera-to-turntable transformations, misalignments were identified in the renderings, and the poses were manually refined accordingly. The refinement was repeated until a satisfactory alignment of the renderings with the images of the scene was achieved.

6.7 Design Validation and Experiments

This section evaluates the accuracy of the ground-truth 6D object poses and assesses the difficulty of the 6D object localization task on the T-LESS dataset.

6.7.1 Accuracy of the Ground-Truth 6D Object Poses

The depth images, corrected as described in Section 6.4, were compared with depth images obtained by graphically rendering the 3D object models in the ground-truth poses. Table 6.1 shows the statistics of the difference δ between depth values in the two images, which was calculated at pixels with a valid depth value in both images and aggregated over all training and test images. Differences exceeding 5 cm were considered to be outliers and pruned before calculating the statistics. The outliers represent ~2.5 % of the differences and are typically caused by erroneous depth measurements or by occlusion induced by distractor objects in the case of test images.

In the case of Carmine, the captured depth images align well with the rendered depth images as indicated by the mean difference μ_δ being close to zero. In the case of Kinect, the RGB and depth images were observed to be slightly misregistered, which is the cause of the positive bias in μ_δ. The average absolute difference $\mu_{|\delta|}$ is less than 5 mm for Carmine and 9 mm for Kinect, which is near the accuracy of the sensors [131] and is relatively small compared to the size of the T-LESS objects. The error statistics are slightly favorable for the reconstructed object models which were created from the captured depth images and exhibit similar characteristics and artifacts as the images. For example, the plug poles are not visible in the depth images and are therefore missing in the reconstructed models, but are present in the CAD models.

6.7.2 6D Object Localization

The difficulty of 6D object localization on the T-LESS dataset is assessed with Hash-Match, the template-matching method described in Chapter 3. The method is evaluated on all test RGB-D images from Carmine, the templates are generated from the RGB-D training images from Carmine, and the CAD models are used in the pose refinement stage. The accuracy of the method is measured by the recall rate, i.e., the fraction of annotated object instances for which a correct pose is estimated. A pose estimate is considered correct if the ADI error defined in Section 2.5 is below 10% of the object diameter.

Figure 6.6 presents the recall rates per object (top) and per scene (middle). The lowest recall rates are achieved on objects that are similar to other objects. For example, object 1 is often confused with object 2, as are objects 20, 21 and 22. Likewise, test scenes containing similar objects are more difficult, with the hardest one being scene 20 that contains multiple instances of several similar objects. Besides the mutual similarity

Sensor, model type	μ_δ	σ_δ	$\mu_{\lvert\delta\rvert}$	$\text{med}_{\lvert\delta\rvert}$
Carmine, CAD	-0.60	8.12	4.53	2.57
Carmine, reconstructed	-0.79	7.72	4.28	2.46
Kinect, CAD	4.46	11.76	8.76	5.67
Kinect, reconstructed	4.08	11.36	8.40	5.45

Table 6.1. **Quality of the ground-truth 6D object poses.** Statistics of differences between the depth of object models in the ground-truth poses and the captured depth (in millimeters) – μ_δ and σ_δ are the mean and the standard deviation of the differences, $\mu_{\lvert\delta\rvert}$ and $\text{med}_{\lvert\delta\rvert}$ are the mean and the median of the absolute differences.

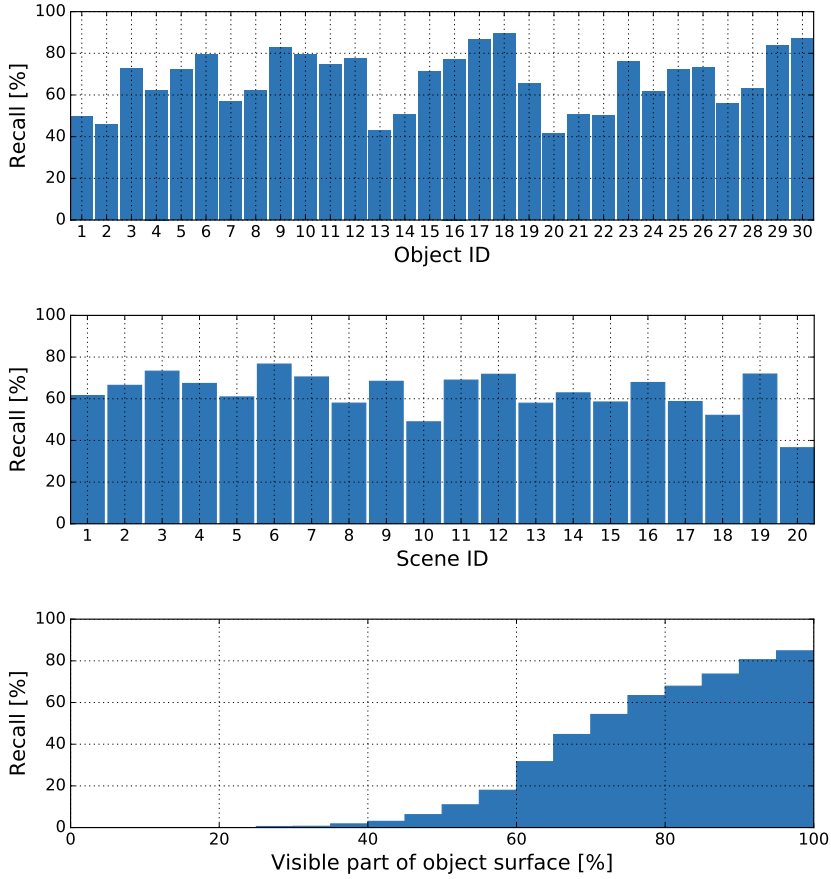

Figure 6.6. **HashMatch results on T-LESS.** Shown are the recall rates per object (top), per scene (middle), and w.r.t. the visible fraction of the object surface (bottom).

of objects, occlusion is another challenge presented in T-LESS – as shown at the bottom of Figure 6.6, the recall rate decreases proportionally with the visible fraction of objects.

The recall rate of HashMatch averaged over all T-LESS objects is 67.2%, which is noticeably lower than the recall rate of 95.4% which this method achieves on the well-established and rather saturated LM dataset [94]. A margin for improvement, indicating new challenges brought by the T-LESS dataset, is evident also from the results of the recent BOP Challenge 2020 described in Section 7.4.

Chapter 7

BOP

The Benchmark for 6D Object Pose Estimation

The progress of research in computer vision has been strongly influenced by challenges and benchmarks, which enable to evaluate and compare methods and better understand their limitations. The Middlebury benchmark [221, 220] for depth from stereo and optical flow estimation was one of the first that gained large attention. The PASCAL VOC challenge [68], based on a photo collection from the internet, was the first to standardize the evaluation of object detection and image classification. It was followed by the ImageNet challenge [218] which has pushed image classification methods to new levels of accuracy. The key was a large-scale dataset that enabled training of deep neural networks, which then quickly became a game-changer for many other tasks [137]. With increasing maturity of computer vision methods, recent benchmarks moved to real-world scenarios. A great example is the KITTI benchmark [77] focusing on problems related to autonomous driving. It showed that methods ranking high on established benchmarks, such as the Middlebury, may perform below average when moved outside the laboratory conditions.

Many methods for 6D object pose estimation have been published recently, e.g., [241, 138, 112, 18, 265, 127, 205, 167], but it was unclear which methods perform well and in which scenarios. The most commonly used LM dataset by Hinterstoisser et al. [94] was not intended as a general benchmark and has several limitations: the lighting conditions are constant and the objects are easy to distinguish, unoccluded, and located around the image center. Some of these limitations were later addressed – Brachmann et al. [17] created the LM-O dataset by adding ground-truth annotation for occluded objects in the LM dataset, the T-LESS dataset described in Chapter 6 introduced industry-relevant objects with symmetries and similarities, and Drost et al. [62] created the ITODD dataset with objects from reflective materials. However, the datasets were provided in different formats and there was no standard evaluation methodology. New methods were usually compared with only a few competitors on a small subset of datasets.

The BOP benchmark was created both to capture the status quo in the field and to systematically measure the progress in the future. Unlike the PASCAL VOC and ImageNet challenges, the task of 6D object pose estimation requires a specific set of calibrated modalities that cannot be easily acquired from the internet. In contrast to KITTY, it was not necessary to record large amounts of new data. We covered many practical scenar-

Figure 7.1. **3D object models available in the BOP datasets.** The models are provided as 3D meshes with surface normal vectors and surface color. The color is saved per mesh vertex or as an UV texture. The models were created manually in CAD software or by KinectFusion-like systems for 3D surface reconstruction [178]. T-LESS includes both CAD and reconstructed models with surface color available only for the reconstructed models shown above. ITODD includes only CAD models with no color. The figure shows models rendered at the same distance.

ios by combining existing datasets. Additionally, we created two datasets with varying lighting conditions, which is an aspect that was not covered by the existing datasets.

As of 2020, the BOP benchmark comprises of: (1) eleven datasets in a unified format which cover various types of objects and practical scenarios – see Figures 7.1 and 7.2, (2) an evaluation methodology with three new pose-error functions which address limitations of the previously used functions, (3) an online evaluation system which is open for continuous submission of new results and reports the current state of the art, and (4) public challenges held at the International Workshops on Recovering 6D Object Pose [110]

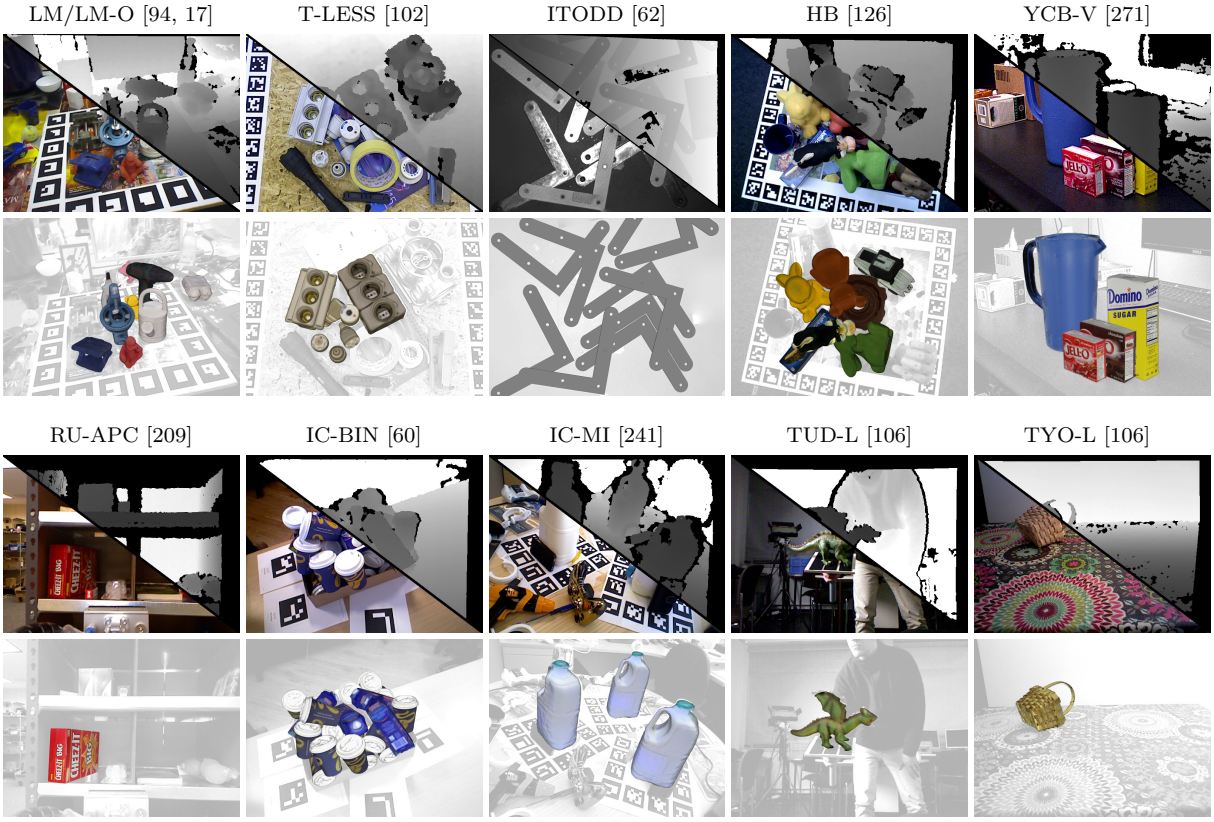

Figure 7.2. **Example test RGB-D images from the BOP datasets**. The upper rows show split views of the RGB and D image channels. The lower rows show the images overlaid with 3D object models in the ground-truth 6D poses.

organized annually at the ICCV and ECCV conferences. Tomáš Hodaň has been leading the organization of the BOP benchmark together with Frank Michel (up to 2018) and later together with Martin Sundermeyer.

The evaluation methodology is defined in Section 7.1, the datasets are introduced in Section 7.2, and the public challenges are described and their results analyzed in Sections 7.3 and 7.4. The initial version of the BOP benchmark together with the results of the first public challenge from 2017 were published in [106] and the results of the following two public challenges from 2019 and 2020 in [109]. The datasets, evaluation system, and up-to-date leaderbords are available on the project website: `bop.felk.cvut.cz`. Evaluation scripts are available in the BOP Toolkit [108].

7.1 Evaluation Methodology

The evaluation methodology detailed in this section defines the 6D object localization task on which the methods are evaluated (Section 7.1.1), functions to measure the error of 6D pose estimates which can (fully or partially) deal with pose ambiguities (Sections 7.1.2–7.1.4), and the accuracy score used to rank the methods (Section 7.1.6).

7.1.1 6D Object Localization

Methods are evaluated on the task of 6D localization of a **v**arying number of **i**nstances of a **v**arying number of **o**bjects from a single RGB-D image. This variant of the 6D object localization task is referred to as ViVo and defined as:

Training input: For each object, with a label $o \in \{1, \ldots, k\}$, a method is given a 3D mesh model M_o (typically with a color texture) and synthetic or real RGB-D images showing instances of the object in known 6D poses. Any of the image channels may be used.

Test input: The method is provided with an image I and a list $L = [(o_1, n_1), \ldots, (o_m, n_m)]$, where n_i is the number of instances of the object o_i present in I.

Test output: The method produces a list $E = [E_1, \ldots, E_m]$, where E_i is a list of n_i pose estimates for instances of the object o_i. An estimate is given by a 3×3 rotation matrix $\mathbf{R}$, a 3×1 translation vector $\mathbf{t}$, and a confidence s. The matrix $\mathbf{P} = [\mathbf{R} \,|\, \mathbf{t}]$ defines a transformation from the 3D model coordinates to the 3D camera coordinates.

Note that in the first challenge from 2017, methods were evaluated on a simpler variant of the 6D object localization task with the goal to estimate the 6D pose of a **s**ingle **i**nstance of a **s**ingle **o**bject. This variant is referred to as SiSo and detailed in Section 7.3.2.

In BOP, methods have been evaluated on variants of the 6D object localization task for two reasons. First, the accuracy scores on this simpler task are still far from being saturated (Section 7.4). Second, the 6D object detection task requires computationally expensive evaluation as many more hypotheses need to be evaluated to calculate the precision/recall curve, which is typically used for evaluating detection. Calculating the 6D pose errors is more expensive than, for example, calculating the intersection over union of 2D bounding boxes used to evaluate 2D object detection.

7.1.2 VSD: Visible Surface Discrepancy

To calculate the VSD error of an estimated pose $\hat{\mathbf{P}}$ w.r.t. the ground-truth pose $\bar{\mathbf{P}}$ in an image I, an object model M is first rendered in the two poses. The result of the rendering is two distance maps $\hat{D}$ and $\bar{D}$.[1] As described below, the distance maps are compared with the distance map D_I of the test image I to obtain the visibility masks $\hat{V}$ and $\bar{V}$, i.e., the sets of pixels where the model M is visible in the image I. Given a misalignment tolerance τ, the error is calculated as follows ($[\cdot]$ is the Iverson bracket; see also Figure 7.3):

$$e_{\text{VSD}}\big(\hat{D}, \bar{D}, \hat{V}, \bar{V}, \tau\big) = \frac{1}{|\hat{V} \cup \bar{V}|} \sum_{\mathbf{u} \in \hat{V} \cup \bar{V}} \Big[\mathbf{u} \notin \hat{V} \cap \bar{V} \;\vee\; \big|\hat{D}(\mathbf{u}) - \bar{D}(\mathbf{u})\big| \geq \tau\Big] \tag{7.1}$$

[1] A distance map stores at a pixel $\mathbf{u}$ the distance from the camera center to a 3D point $\mathbf{x}_\mathbf{u}$ that projects to $\mathbf{u}$. The distance map can be readily computed from the depth map which stores at $\mathbf{u}$ the Z coordinate of $\mathbf{x}_\mathbf{u}$ and which is a typical output of Kinect-like sensors.

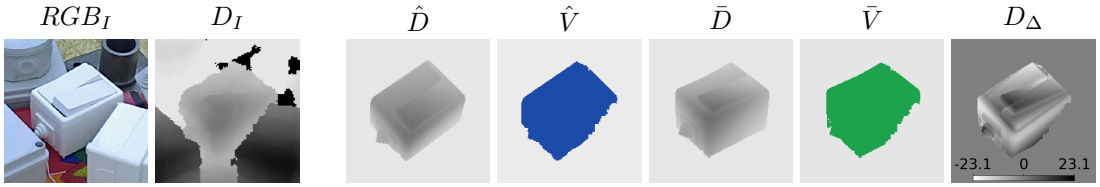

Figure 7.3. **Quantities used in the calculation of VSD.** Left: The color channels RGB_I (only for illustration) and the distance map D_I of a test image I. Right: The distance maps $\hat{D}$ and $\bar{D}$ are obtained by rendering the object model M in the estimated pose $\hat{\mathbf{P}}$ and the ground-truth pose $\bar{\mathbf{P}}$ respectively. The visibility masks $\hat{V}$ and $\bar{V}$ of the model surface that is visible in I are obtained by comparing $\hat{D}$ and $\bar{D}$ with D_I. The distance differences $D_\Delta(\mathbf{u}) = \hat{D}(\mathbf{u}) - \bar{D}(\mathbf{u})$, $\forall \mathbf{u} \in \hat{V} \cap \bar{V}$, are used for the pixel-wise evaluation of the surface alignment.

Visibility Masks. Given the rendered distance image $\bar{D}$, with $\bar{D}(\mathbf{u}) = 0$ for pixels outside the object mask, and given the distance image D_I of the test scene, with $D_I(\mathbf{u}) = 0$ for pixels with missing measurements, the visibility mask $\bar{V}$ is defined as a set of pixels where the surface of the model M in the ground-truth pose $\bar{\mathbf{P}}$ is at most by a tolerance δ behind the scene surface, or where the distance of the scene surface is unknown:

$$\bar{V} = \left\{ \mathbf{u} \,|\, \bar{D}(\mathbf{u}) > 0 \;\wedge\; \left(\bar{D}(\mathbf{u}) - D_I(\mathbf{u}) \leq \delta \;\vee\; D_I(\mathbf{u}) = 0 \right) \right\} \tag{7.2}$$

An object is considered visible at pixels with unknown distance to enable evaluating poses of glossy objects, e.g., objects from the ITODD dataset [62], whose surface is not always captured in the distance image. Note that an object was considered invisible at such pixels in the initial version of the evaluation methodology [106, 105].

For the visibility mask $\hat{V}$ of the model M in the estimated pose $\hat{\mathbf{P}}$, the definition (7.2) is extended by including all pixel from the visibility mask $\bar{V}$ regardless the distance of the scene surface at these pixels. This is to ensure that the surface of the object captured in D_I does not occlude the object model in the estimated pose. The mask $\hat{V}$ is defined as:

$$\hat{V} = \left\{ \mathbf{u} \,|\, \hat{D}(\mathbf{u}) > 0 \;\wedge\; \left(\hat{D}(\mathbf{u}) - D_I(\mathbf{u}) \leq \delta \;\vee\; D_I(\mathbf{u}) = 0 \;\vee\; \mathbf{u} \in \bar{V} \right) \right\} \tag{7.3}$$

Properties of VSD. The VSD error is calculated only over the visible part of the model and thus indistinguishable poses are treated as equivalent. This is a desirable property not provided by the common pose-error functions, including ADD and ADI [94, 105].

Note that VSD evaluates the alignment of the object shape but not of its color. This is because most of the models currently included in BOP have baked shadows and reflections in their textures, which makes it difficult to robustly evaluate the color alignment.

Comparison of VSD and ADI. The most widely used pose-error functions have been ADD and ADI [94, 105] calculated as the average distance from vertices of the model M in the ground-truth pose $\bar{\mathbf{P}}$ to vertices of M in the estimated pose $\hat{\mathbf{P}}$. The distance is measured to the corresponding vertices if all views of the object are distinguishable (ADD) and to the closest vertices otherwise (ADI). The estimated pose $\hat{\mathbf{P}}$ is considered

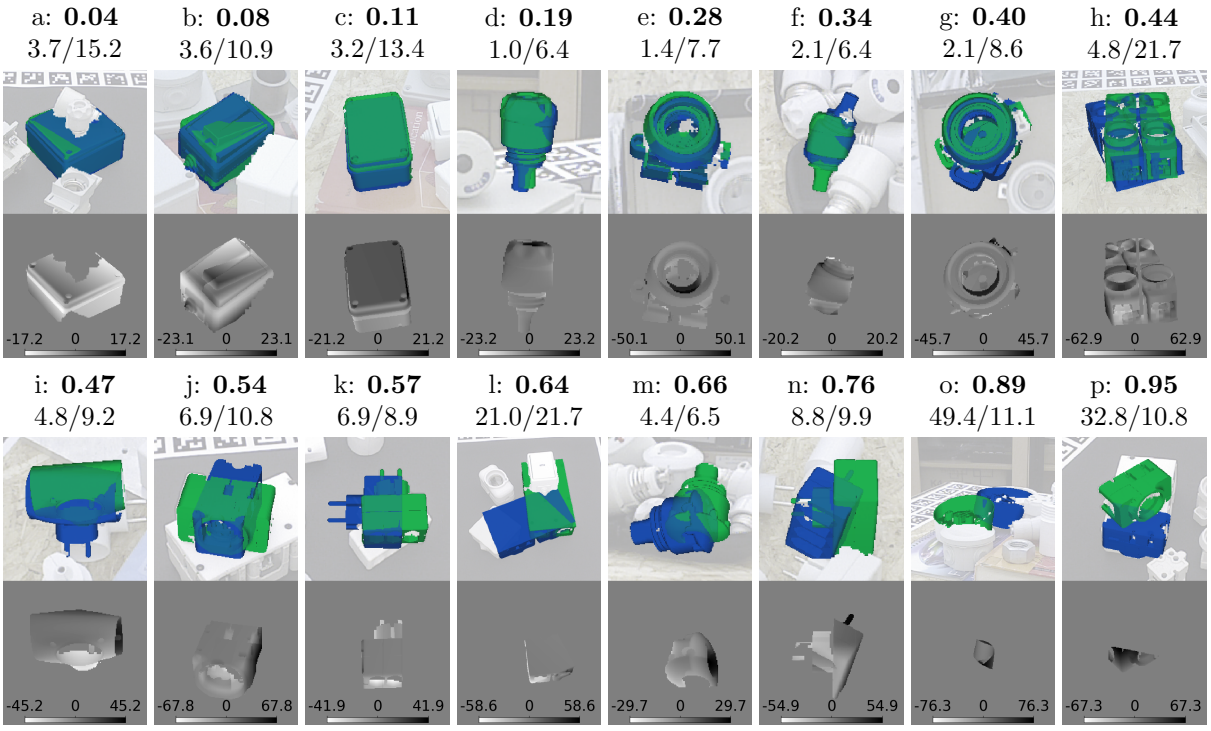

Figure 7.4. **Comparison of VSD and ADI.** The VSD errors (bold, $\tau = 20\,\text{mm}$, $\delta = 15\,\text{mm}$) are compared with the ADI errors (in mm, the threshold of correctness θ_{AD} is after the slash) on example pose estimates sorted by increasing VSD. Top: Cropped and brightened test images overlaid with renderings of the model in the estimated pose $\hat{\mathbf{P}}$ in blue, and in the ground-truth pose $\bar{\mathbf{P}}$ in green. Only the part of the model surface that falls into the respective visibility mask is shown. Bottom: Distance differences D_Δ. Case (b) is considered in Figure 7.3.

correct if $e \leq \theta_{\text{AD}} = 0.1d$, where e is ADD or ADI, and d is the object diameter.

As discussed in Section 2.5.2, ADI may yield unintuitively low errors for poses that are clearly distinguishable due to a many-to-one vertex matching that may be established by the search for the closest vertex, and due to the possibility to match the outer and the inner model parts. This is shown in Figure 7.4, which compares VSD and ADI on example pose estimates of objects that have indistinguishable views and their models include the inner parts. Overall, (f)-(n) yield low ADI errors and satisfy the correctness criterion of Hinterstoisser et al. These estimates, together with estimates (o) and (p), would be marked as incorrect by requiring the VSD error to be at most 0.3, which is the criterion of correctness used in the initial version of the benchmark [106]. In the current version, the accuracy scores are calculated over multiple thresholds (see Section 7.1.6).

7.1.3 MSSD: Maximum Symmetry-Aware Surface Distance

The MSSD error is defined as the maximum distance between a vertex $\mathbf{x} \in V_M$ of the object model M in the ground-truth pose $\bar{\mathbf{P}} = [\bar{\mathbf{R}} \,|\, \bar{\mathbf{t}}]$ and the same vertex $\mathbf{x}$ in the estimated pose $\hat{\mathbf{P}} = [\hat{\mathbf{R}} \,|\, \hat{\mathbf{t}}]$, while considering global symmetry transformations S_M of the

model M (the symmetry transformations are identified as described in Section 7.1.5):

$$e_{\text{MSSD}}\big(\hat{\mathbf{R}},\hat{\mathbf{t}},\bar{\mathbf{R}},\bar{\mathbf{t}},S_M,V_M\big)=\min_{[\mathbf{R}\,|\,\mathbf{t}]\in S_M}\max_{\mathbf{x}\in V_M}\big\|\big(\hat{\mathbf{R}}\mathbf{x}+\hat{\mathbf{t}}\big)-\big(\bar{\mathbf{R}}\big(\mathbf{R}\mathbf{x}+\mathbf{t}\big)+\bar{\mathbf{t}}\big)\big\|_2 \tag{7.4}$$

Properties of MSSD. The maximum distance between model vertices is relevant for robotic manipulation, where the maximum surface deviation strongly indicates the chance of a successful grasp. Moreover, compared to the average distance used in the pose-error functions ADD and ADI [105, 94], which tends to be dominated by higher-frequency surface parts such as the thread of a fuse, the maximum distance is less dependent on the geometry of the object model and the sampling density of its surface.

As the MSSD error is calculated over the entire model surface, misalignment of the invisible surface part is penalized. This may not be desirable for applications such as robotic manipulation with suction cups where only the alignment of the visible part is relevant (note that VSD evaluates only the visible surface part). Pose ambiguities due to global object symmetries are handled properly in MSSD by explicitly considering all the symmetry transformations.

7.1.4 MSPD: Maximum Symmetry-Aware Projection Distance

The MSPD error is defined as the maximum distance between the 2D projection of a vertex $\mathbf{x}\in V_M$ of the object model M in the ground-truth pose $\bar{\mathbf{P}}=[\bar{\mathbf{R}}\,|\,\bar{\mathbf{t}}]$ and the 2D projection of the same vertex $\mathbf{x}$ in the estimated pose $\hat{\mathbf{P}}=[\hat{\mathbf{R}}\,|\,\hat{\mathbf{t}}]$, while considering the global symmetry transformations S_M ($\text{proj}(\cdot)$ is the 2D projection):

$$e_{\text{MSSD}}\big(\hat{\mathbf{R}},\hat{\mathbf{t}},\bar{\mathbf{R}},\bar{\mathbf{t}},S_M,V_M\big)=\min_{[\mathbf{R}\,|\,\mathbf{t}]\in S_M}\max_{\mathbf{x}\in V_M}\big\|\text{proj}\big(\hat{\mathbf{R}}\mathbf{x}+\hat{\mathbf{t}}\big)-\text{proj}\big(\bar{\mathbf{R}}\big(\mathbf{R}\mathbf{x}+\mathbf{t}\big)+\bar{\mathbf{t}}\big)\big\|_2 \tag{7.5}$$

Properties of MSPD. Since MSPD does not evaluate the alignment along the optical (Z) axis and measures only the perceivable discrepancy, it is relevant for augmented reality applications and suitable for evaluating RGB-only methods, for which estimating the alignment along the optical axis is more challenging. Compared to the "2D Projection" pose-error function proposed in [17], MSPD explicitly considers the global object symmetries and replaces the average by the maximum distance to increase the robustness against geometry and sampling of the object model. As MSSD, MSPD penalizes misalignment of the invisible surface part.

7.1.5 Identifying Global Object Symmetries

The set of global symmetry transformations of an object model M, which is used in MSSD and MSPD, is identified in two steps. Firstly, a set of candidate symmetry transformations is defined as $S'_M=\{\mathbf{S}\in SE(3)\mid h(V_M,\mathbf{S}V_M)<\varepsilon\}$, where $h(\cdot,\cdot)$ is the Hausdorff distance, V_M are 3D vertices of the model M in the canonical pose, and $\mathbf{S}V_M$ are the

3D vertices transformed by a rigid transformation $\mathbf{S} = [\mathbf{R} \,|\, \mathbf{t}]$, which consists of a 3×3 rotation matrix $\mathbf{R}$ and a 3×1 translation vector $\mathbf{t}$. The allowed deviation is bounded by $\varepsilon = \max(15\,\text{mm}, 0.1d)$, where d is the diameter of M and the truncation at 15 mm avoids breaking symmetries by too small details. Note that even though all symmetry transformations of objects with a finite size are 3D rotations around an axis of symmetry, we have no guarantees that the axes of symmetry of the model M intersect the origin of the model coordinate system. Therefore, we need to consider elements of the special Euclidean group $SE(3)$ as candidates for the symmetry transformations. In practice, the set S'_M is found by dense sampling of the rotational and translational spaces and covers both discrete and continuous (discretized) global symmetries of the model M.

In the second step, the final set of symmetry transformations S_M is defined as a subset of S'_M that consists of those transformations that cannot be resolved by the color texture of M. This was decided subjectively by the BOP organizers – as mentioned in Section 7.1.2, the object models included in BOP have baked shadows and reflections in textures, which makes it difficult to evaluate the color alignment programmatically.

7.1.6 Accuracy Score

An estimated pose is considered correct with respect to a pose-error function e, if $e < \theta_e$, where $e \in \{e_{\text{VSD}}, e_{\text{MSSD}}, e_{\text{MSPD}}\}$ and θ_e is a threshold of correctness. All three pose-error functions are used as each evaluates different qualitites of the pose estimates and is relevant for a different target application.

The fraction of annotated object instances for which a correct pose is estimated is referred to as recall. The average recall with respect to a function e, denoted as AR_e, is defined as the average of recall rates calculated for multiple settings of the threshold θ_e, and also for multiple settings of the misalignment tolerance τ in the case of e_{VSD}. In particular, AR_{VSD} is the average of recall rates calculated for the misalignment tolerance τ ranging from 5% to 50% of the object diameter with a step of 5%, and for the threshold θ_{VSD} ranging from 0.05 to 0.5 with a step of 0.05. AR_{MSSD} is the average of recall rates calculated for θ_{MSSD} ranging from 5% to 50% of the object diameter with a step of 5%. Finally, AR_{MSPD} is the average of recall rates calculated for θ_{MSPD} ranging from $5r$ to $50r$ with a step of $5r$, where $r = w/640$ and w is the image width in pixels.

The accuracy of a method on a dataset D is measured by $\text{AR}_D = (\text{AR}_{\text{VSD}} + \text{AR}_{\text{MSSD}} + \text{AR}_{\text{MSPD}})/3$. The overall accuracy on the core datasets is then measured by AR_{Core} defined as the average of the per-dataset AR_D scores. In this way, each dataset is treated as a separate sub-challenge which avoids AR_{Core} being dominated by larger datasets.

7.2 Datasets

The BOP benchmark currently includes eleven datasets in a unified format. Each dataset includes 3D object models and training and test RGB-D images annotated with ground-truth 6D object poses. The 3D object models were created manually or using KinectFusion-like systems for 3D surface reconstruction [178]. The training images are real or synthetic. All test images are real. The 3D object models are shown in Figure 7.1, sample test images in Figure 7.2, and parameters of the datasets are provided in Table 7.1.

Seven of the datasets were selected as core datasets (marked with a star in Table 7.1). In the recent public challenges from 2019 and 2020, a method had to be evaluated on all core datasets to be considered for the main challenge awards [109]. The core datasets include photorealistic training images generated by the approach described in Section 5.5. Datasets T-LESS, TUD-L, and YCB-V include real training images and most datasets include also training images obtained by OpenGL rendering of the 3D object models on a black background. The test images were captured in scenes with varied complexity, often with clutter and occlusion. Besides the training and test images, the HB and ITODD datasets include also validation images – in this case, the ground-truth poses are publicly available for the validation images but not for the test images. A description of the individual datasets is provided in the following paragraphs.

LM/LM-O [94, 17]. Linemod (LM) has been the most widely used dataset for 6D object pose estimation. It contains 15 texture-less objects with discriminative color, shape, and size. Each object is associated with a set of test images showing one annotated object instance under significant clutter but only mild occlusion. Linemod-Occluded (LM-O) provides ground-truth annotation for all other instances of the modeled objects in one of the test sets, which introduces challenging test cases with various levels of occlusion.

T-LESS [102]. This dataset features 30 industry-relevant objects with no significant texture or discriminative color, with symmetries and mutual similarities in shape and size, and with some objects being a composition of other objects. T-LESS includes images from three different sensors and two types of 3D object models. Only RGB-D images from the Primesense sensor are included in BOP. See Chapter 6 for details.

ITODD [62]. The MVTec Industrial 3D Object Detection Dataset (ITODD) is focused on objects and settings typical for industrial bin picking and object inspection. The dataset contains 28 mostly texture-less objects with different characteristics in terms of material properties (glossy vs. Lambertian), symmetry (no vs. discrete or continuous symmetries), shape (flat, long, compact, etc.), and size (diameters from 24 to 270 mm). The objects are captured in various arrangements with two industrial 3D sensors of two qualities (providing aligned grayscale and depth images) and three high-resolution grayscale sensors. Only images from the higher-quality 3D sensor are included in BOP.

			Train. im.		Val im.	Test im.		Test inst.	
Dataset	Core	Objects	Real	PBR	Real	All	Used	All	Used
LM [94]		15	–	50000	–	18273	3000	18273	3000
LM-O [17]	∗	8	–	50000	–	1214	200	9038	1445
T-LESS [102]	∗	30	37584	50000	–	10080	1000	67308	6423
ITODD [62]	∗	28	–	50000	54	721	721	3041	3041
HB [126]	∗	33	–	50000	4420	13000	300	67542	1630
YCB-V [271]	∗	21	113198	50000	–	20738	900	98547	4123
RU-APC [209]		14	–	–	–	5964	1380	5964	1380
IC-BIN [60]	∗	2	–	50000	–	177	150	2176	1786
IC-MI [241]		6	–	–	–	2067	300	5318	800
TUD-L [106]	∗	3	38288	50000	–	23914	600	23914	600
TYO-L [106]		21	–	–	–	1670	1670	1670	1670

Table 7.1. **Parameters of the BOP datasets.** Most datasets include also training images obtained by OpenGL rendering of the 3D object models on a black background (not shown in the table). Extra PBR training images can be rendered by BlenderProc4BOP [55, 54]. If a dataset includes both validation and test images, the ground-truth annotations are public only for the validation images. All test images are real. Column "Test inst./All" shows the number of annotated object instances for which at least 10% of the projected surface area is visible in the test images. Columns "Used" show the number of test images and object instances used in the BOP Challenge 2019 and 2020.

HB [126]. The HomebrewedDB dataset (HB) contains high-quality reconstructed 3D models of 33 objects (17 toys, 8 industry-relevant, and 8 household objects) captured in 13 test scenes with two RGB-D sensors. The complexity of the scenes ranges from simple scenes with several isolated objects on a plain background to challenging scenes with heavily occluded objects and extensive clutter. Only RGB-D images from the Primesense sensor captured in three scenes including 16 objects are currently included in BOP.

YCB-V [271]. The original YCB-Video dataset provides 92 RGB-D video sequences (80 for training, 12 for test) with over 133K frames showing 21 household, both textured and texture-less, objects from the YCB dataset [27]. The objects are arranged in scenes with mild clutter and various levels of occlusion. For the BOP benchmark, we have manually selected 75 images with higher-quality ground-truth poses from each test video sequence.

RU-APC [209]. The Rutgers APC dataset includes 14 textured products from the Amazon Picking Challenge 2015 [67], each associated with test images showing the product in a cluttered warehouse shelf. The camera was equipped with LED strips to ensure constant lighting. Ten objects from the original dataset which are non-rigid or poorly captured by the depth sensor were omitted.

IC-MI/IC-BIN [241, 60]. The dataset of Tejani et al. (IC-MI) provides 3D models of two texture-less and four textured household objects. The test images show multiple object instances under clutter and mild occlusion. The dataset of Doumanoglou et al.

(IC-BIN, "scenario 2" from the original dataset) includes test images which simulate a bin-picking scenario and show two objects from IC-MI in multiple instances under heavy occlusion. We have removed test images with low-quality ground-truth poses from both datasets and refined the ground-truth poses for the remaining images in IC-BIN.

TUD-L/TYO-L [106]. These two datasets were introduced together with the initial version of the BOP benchmark [106] and include household objects captured under different settings of ambient and directional light. The TU Dresden Light dataset (TUD-L) contains training and test image sequences that show three moving objects under eight lighting conditions. The object poses were annotated by manually aligning the 3D object model with the first frame of the sequence and propagating the pose frame-by-frame by ICP. The Toyota Light dataset (TYO-L) contains 21 objects, each captured in multiple poses on a table with four different table cloths and five different lighting conditions. The ground-truth object poses were obtained by estimating rough poses from manually established 2D-3D correspondences and refining the poses by ICP. The images in both datasets are labeled by categorized lighting conditions.

7.3 BOP Challenge 2017

The BOP Challenge 2017 [107] (originally named SIXD Challenge 2017) was the first step in shaping the BOP benchmark. It was organized together with the 3rd International Workshop on Recovering 6D Object Pose [260] at the ICCV 2017 conference. Participants of the challenge were submitting the results of their methods by e-mail from May 11, 2017 to March 14, 2018. The experimental setup is described in Section 7.3.2, and the results are analyzed in Section 7.3.3 and published in [106].

7.3.1 Evaluated Methods

The methods evaluated in the BOP Challenge 2017 cover all major research directions of the 6D object pose estimation field which were active at that time. A general description of the methods is in Chapter 2, and a description of the evaluated versions, together with the setting of their key parameters, in the following paragraphs. If not stated otherwise, the image-based methods were trained on the images obtained by OpenGL rendering of the 3D object models on a black background.

Template Matching Methods. Hodaň-15 [112] is the HashMatch template-matching method described in Chapter 3 which applies an efficient cascade-style evaluation to each sliding window location. The templates were generated by applying the full circle of in-plane rotations with 10° step to a portion of the synthetic training images, resulting in 11–23K templates per object. Other parameters were set as described in [112]. Results of the method without the last refinement step are reported under the name Hodaň-15-nr.

Methods Based on Point-Pair Features. Drost-10 [63], i.e., the method which introduced the point pair features, was evaluated using function `find_surface_model` from HALCON 13.0.2 [177]. The sampling distance for the model and the scene was set to 3% of the object diameter, 10% of points were used as the reference points, and the surface normals were computed using the `mls` method. Points farther than 2 m were discarded.

Drost-10-edge extends Drost-10 by detecting depth edges and favoring poses in which the model contours are aligned with the edges. A multi-modal refinement minimizes the surface distances and the distances from the reprojected model contours to the detected edges. The evaluation was performed using the same software and parameters as Drost-10 but with the parameter `train_3d_edges` being activated.

In the evaluated version, Vidal-18 [257] sorts the 500 most-voted pose candidates by a surface fitting score and refines the 200 best candidates by a projective ICP. For the final 10 candidates, they evaluate the consistency of the object surface and the object silhouette with the scene. The sampling distance for the model and the scene was set to 5% of the object diameter, and 20% of the scene points were used as the reference points.

Learning-Based Methods. Brachmann-14 [17] and Brachmann-16 [18] use a random forest to predict, at each pixel of the input image, the object identity and the 3D object coordinates. Brachmann-16 [18] presents three improvements of Brachmann-14 – see Section 2.2.2. The first two improvements were disabled for the evaluation since we deal with RGB-D input, and it is known which objects are visible in the image. The parameters of Brachmann-14 were set as: maximum feature offset: 20 px, features per tree node: 1000, training patches per object: 1.5M, number of trees: 3, size of the hypothesis pool: 210, refined hypotheses: 25. The parameters of Brachmann-16 were set as: maximum feature offset: 10 px, features per tree node: 100, number of trees: 3, number of sampled hypotheses: 256, pixels drawn in each RANSAC iteration: 10K, inlier threshold: 1 cm. Real training images from TUD-L and T-LESS were used for both methods.

In the Tejani-14 method [241], each patch of the input image is processed by a random forest and casts 6D votes which are then aggregated to produce the final pose estimates. The evaluated version omits the iterative refinement step and does not perform ICP. The features and forest parameters were set as in [241]: number of trees: 10, maximum depth of a tree: 25, number of features in both the color gradient and the surface normal channel: 20, patch size: 1/2 the image, rendered images used to train each forest: 360.

Kehl-16 [128] is another voting-based method which calculates auto-encoder features for the image patches and retrieves up to k nearest neighbors (whose distance is below a threshold t) from a codebook, where each entry is associated with 6D pose votes. The 6D hypotheses space is filtered to remove spurious votes, modes are identified by mean-shift and refined by ICP. The final hypotheses are verified in color, depth and surface normals to suppress false positives. The main parameters of the method were set as follows: patch size: 32×32 px, patch sampling step: 6 px, k-nearest neighbors: 3, threshold t: 2, number of extracted modes from the pose space: 8. Real training images were used for T-LESS.

Methods Based on 3D Local Features. Buch-16 [21] is a RANSAC-based method which iteratively samples three feature correspondences between the object model and the scene. The correspondences are obtained by matching 3D local shape descriptors and used to generate 6D pose candidates, whose quality is measured by the consensus set size. The final pose is refined by ICP. The method achieved the state-of-the-art results on earlier object recognition datasets captured by LIDAR, but suffers from a cubic complexity in the number of correspondences. The number of RANSAC iterations was set to 10000, allowing only for a limited search in cluttered scenes. The method was evaluated with several descriptors: 153d SI [123], 352d SHOT [219], 30d ECSAD [124], and 1536d PPFH [23]. None of the descriptors utilize color.

The Buch-17 method [22] is based on the observation that a correspondence between two oriented points on the object surface is constrained to cast votes in a 1-DoF rotational subgroup of the special Euclidean group $SE(3)$ of object poses (this observation is used also in methods based on point pair features). The time complexity of the method is thus linear in the number of correspondences. Kernel density estimation is used to efficiently combine the votes and estimate the 6D pose. As Buch-16, the method relies on 3D local shape descriptors and refines the final pose estimate by ICP. The parameters were set as in [22]: 60-angle tessellation was used for casting rotational votes, and the translation/rotation bandwidth was set to 10 mm/22.5°.

7.3.2 Experimental Setup

SiSo Variant of 6D Object Localization. The methods were evaluated on the task of 6D localization of a **s**ingle **i**nstance of a **s**ingle **o**bject. A test target was defined by a pair (I, o), where I is a test image and o is an identifier of an object model visible in the image. If multiple instances of the object model o are visible in the image, then the pose of an arbitrary instance may have been reported. If multiple object models are shown in the image, and annotated with their ground truth poses, then each object model defined a different test target. For example, if a test image shows three object models, each in two instances, then three test targets were defined and the pose of one of the two object instances had to be estimated for each test target.

The SiSo task reflects the industry-relevant bin-picking scenario where a robot needs to grasp a single arbitrary instance of the required object, e.g., a component such as a bolt or a nut, and perform some operation with it. The difficulty of "multiple instances, find one that you pick" is close to "find the instance in most favorable pose" (least occlusion, unambiguous view). Most methods were expected, but not required, to report the most favorable pose, treating the rest as clutter. The simpler SiSo variant of the 6D object localization task was chosen because it allowed evaluating all relevant methods out of the box. Since then, the state of the art has advanced and we have moved to the more challenging ViVo variant in the 2019 and 2020 editions of the challenge (Section 7.4).

Pose Error and Accuracy Score. Compared to the evaluation methodology presented in Section 7.1 and used in the 2019 and 2020 editions of the challenge, the error of a 6D object pose estimate was in the 2017 edition measured with only the VSD pose-error function, and the correctness of the estimate was decided with a single threshold $\theta_{\text{VSD}} = 0.3$. The tolerance δ used in the estimation of the visibility masks in VSD was set to 15 mm. The accuracy of a method on a dataset was measured by the recall rate calculated as the fraction of test targets for which a correct pose was estimated. The overall accuracy was measured by the average of the per-dataset recall rates.

Datasets. The methods were evaluated on datasets LM, LM-O, IC-MI, IC-BIN, T-LESS, RU-APC, and TUD-L. Only subsets of test images were used to remove redundancies and speed up the evaluation. The test targets were defined by object instances for which at least 10% of the projected surface area is visible in the test images.

Training and Test Rules. A method had to use fixed hyper-parameters across all objects and datasets. For training, a method may have used the provided 3D models and training images, and rendered extra training images using the models. However, not a single pixel of test images may have been used, nor the individual ground-truth poses or masks provided for the test images. Ranges of the azimuth and elevation camera angles, and a range of the camera-object distances calculated from the ground-truth poses in test images is the only information about the test set that may have been used for training.

7.3.3 Results

The accuracy scores of the evaluated methods for the misalignment tolerance $\tau = 20$ mm and the threshold of correctness $\theta_{\text{VSD}} = 0.3$ are shown in Table 7.2. Ranking of the methods according to the accuracy score is mostly stable across the datasets. Methods based on point-pair features perform best. Vidal-18 is the top-performing method with the average recall of 74.6%, followed by Drost-10-edge, Drost-10, and the template-matching method Hodaň-15, all with the average recall above 67%. Brachmann-16 is the best learning-based method, with 55.4%, and Buch-17-ppfh is the best method based on 3D local features, with 54.0%. Scores of Buch-16-si and Buch-16-shot are inferior to the other variants of this method and not presented. Figure 7.5 shows the average of the per-dataset scores for different values of τ and θ. If the misalignment tolerance τ is increased from 20 mm to 80 mm, the scores increase only slightly for most methods. Similarly, the scores increase only slowly for $\theta > 0.3$. This suggests that poses estimated by most methods are either of a high quality or totally off, i.e., it is a hit or miss.

Table 7.2 also shows accuracy scores of EPOS (Chapter 4) on datasets LM-O, IC-BIN, T-LESS and TUD-L.[2] EPOS outperforms most participants of the BOP Challenge 2017

[2]The object poses were originally estimated by EPOS for the BOP Challenge 2020 and here evaluated by the methodology of the BOP Challenge 2017. Results for some datasets are missing as not all datasets from 2017 were included among the core datasets in 2020 on which EPOS was trained.

#	Method	Avg.	LM	LM-O	IC-MI	IC-BIN	T-LESS	RU-APC	TUD-L	Time
1	Vidal-18 [257]	74.60	87.83	59.31	95.33	96.50	66.51	36.52	80.17	4.7
2	Drost-10-edge [63]	71.73	79.13	54.95	94.00	92.00	67.50	27.17	87.33	21.5
3	Drost-10 [63]	68.06	82.00	55.36	94.33	87.00	56.81	22.25	78.67	2.3
4	Hodaň-15 [112]	67.23	87.10	51.42	95.33	90.50	63.18	37.61	45.50	13.5
5	Brachmann-16 [18]	55.44	75.33	52.04	73.33	56.50	17.84	24.35	88.67	4.4
6	Hodaň-15-nr [112]	55.40	69.83	34.39	84.67	76.00	62.70	32.39	27.83	12.3
7	Buch-17-ppfh [22]	54.02	56.60	36.96	95.00	75.00	25.10	20.80	68.67	14.2
8	Kehl-16 [128]	36.97	58.20	33.91	65.00	44.00	24.60	25.58	7.50	1.8
9	Buch-17-si [22]	36.81	33.33	20.35	67.33	59.00	13.34	23.12	41.17	15.9
10	Brachmann-14 [17]	34.61	67.60	41.52	78.67	24.00	0.25	30.22	0.00	1.4
11	Buch-17-ecsad [22]	22.90	13.27	9.62	40.67	59.00	7.16	6.59	24.00	5.9
12	Buch-17-shot [22]	15.64	5.97	1.45	43.00	38.50	3.83	0.07	16.67	6.7
13	Tejani-14 [241]	9.23	12.10	4.50	36.33	10.00	0.13	1.52	0.00	1.4
14	Buch-16-ppfh [21]	7.20	8.13	2.28	20.00	2.50	7.81	8.99	0.67	47.1
15	Buch-16-ecsad [21]	2.38	3.70	0.97	3.67	4.00	1.24	2.90	0.17	39.1
	EPOS (Chapter 4)	–	–	78.28	–	88.50	77.78	–	83.67	–

Table 7.2. **Results of the BOP Challenge 2017.** Recall rates (in %) for the misalignment tolerance $\tau = 20$ mm and the threshold of correctness $\theta_{\text{VSD}} = 0.3$. The recall rate is the percentage of test targets for which a correct object pose was estimated. The methods are sorted by the average of the per-dataset recall rates. The right-most column shows the average processing time per test target (in seconds). The last row shows results of the EPOS method (Chapter 4).

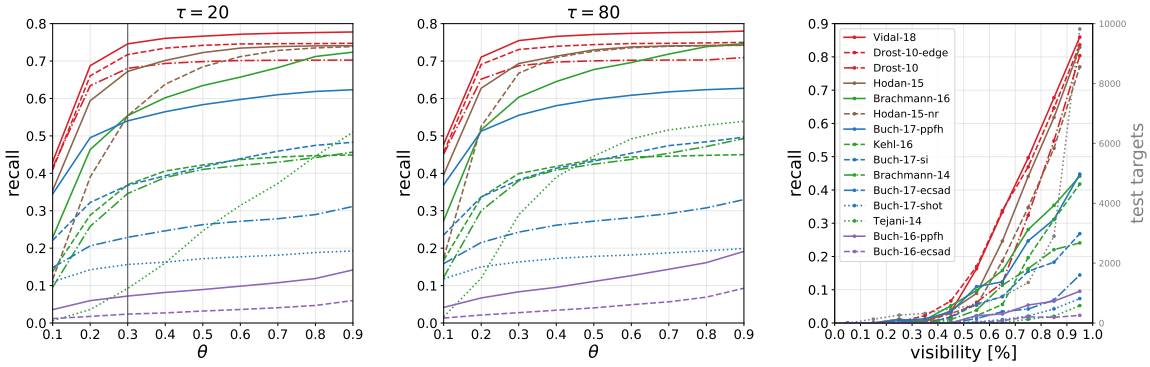

Figure 7.5. **Recall with respect to τ, θ_{VSD}, and the visible surface fraction.** Left/middle: Average of the per-dataset recall rates for the misalignment tolerance τ fixed to 20 mm and 80 mm, and for different values of the correctness threshold θ_{VSD}. The curves do not change much for $\tau > 80$ mm. Right: The recall rates w.r.t. the visible fraction of the target object. If more instances of the object are present in a test image, the largest visible fraction is considered.

on these datasets. Compared to the highest scores achieved on the datasets, the score of EPOS is 5% lower on TUD-L and 8% lower on IC-BIN, but 10% higher on T-LESS and 19% higher on LM-O. These are strong results considering that EPOS relies only on RGB image channels while the other methods also use the depth image channel.

The average running times per test target are reported in Table 7.2. However, the methods were evaluated on different computers and the presented running times are therefore not directly comparable. Moreover, the methods were optimized primarily for accu-

racy, not for speed. For example, we evaluated Drost-10 with several parameter settings and observed that the running time can be lowered by a factor of ~5 to 0.5 s with only a relatively small drop of the average recall from 68.1% to 65.8%. However, in Table 7.2 we present the result with the highest recall. Brachmann-14 could be sped up by subsampling the 3D object models and Hodaň-15 by using less object templates.

Occlusion is a big challenge for the methods, as shown by the accuracy scores dropping swiftly already at low levels of occlusion (Figure 7.5, right). The big gap between LM and LM-O scores provide further evidence. All methods perform on LM by at least 30% better than on LM-O, which includes the same but partially occluded objects. Inspection of estimated poses on T-LESS test images confirms the weak performance for occluded objects. Scores on TUD-L show that varying lighting conditions present a serious challenge for methods that rely on synthetic RGB training images, which were generated with fixed lighting. Methods relying only on depth information (e.g., Vidal-18, Drost-10) are noticeably more robust under such conditions. Note that Brachmann-16 achieved a high score on TUD-L despite relying on RGB images because it used real training images, which were captured under the same range of lighting conditions as the test images. Methods based on 3D local features and learning-based methods have very low scores on T-LESS, which is likely caused by the object symmetries and similarities. All methods perform poorly on RU-APC, which is likely because of a higher level of noise in the depth images.

7.4 BOP Challenge 2019 and 2020

The BOP Challenge 2019 was organized together with the 5th International Workshop on Recovering 6D Object Pose [103] at ICCV 2019, and the BOP Challenge 2020 together with the 6th International Workshop on Recovering 6D Object Pose [110] at ECCV 2020. Participants were submitting their results to the online evaluation system at `bop.felk.cvut.cz`, from July 30 to October 21 in 2019, and from June 5 to August 19 in 2020. The results are analyzed in Section 7.4.3 and published in [109].

In the BOP Challenge 2019, methods using the depth image channel, which were mostly based on the point pair features (PPF's) [63], clearly outperformed methods relying only on the RGB channels, all of which were based on deep neural networks (DNN's). The PPF-based methods match pairs of oriented 3D points between the point cloud of the test scene and the 3D object model, and aggregate the matches via a voting scheme. As each pair is described by only the distance and relative orientation of the two points, PPF-based methods can be effectively trained directly on the 3D object models, without the need to synthesize any training images. In contrast, DNN-based methods require a large number of annotated training images, which have been typically obtained by OpenGL rendering of the 3D object models with random backgrounds [127, 205, 97, 65]. However, as shown in Chapter 5, the evident domain gap between these "render & paste" training images and real test images presumably limits the potential of the DNN-based methods.

To reduce the gap between the synthetic and real domains and thus to bring fresh air to the DNN-based methods, we created BlenderProc4BOP [55, 54, 15], an open-source and light-weight physically-based renderer (PBR). Furthermore, to reduce the entry barrier of the challenge and to standardize the training set, the participants were provided with 350K pre-rendered PBR images described in Section 5.5.

In 2020, the DNN-based methods finally caught up with the PPF-based methods – five methods outperformed Vidal-Sensors18 [257], the PPF-based winner from 2017 and 2019. Three of the top five methods, including the top-performing one, are single-view variants of CosyPose, a DNN-based method by Labbé et al. [139]. Strong data augmentation, similar to [237], was identified as one of the key ingredients of this method. The second is a hybrid DNN+PPF method by König and Drost [135], and the fourth is Pix2Pose, a DNN-based method by Park et al. [188]. The first two methods used RGB-D image channels, while the third method achieved strong results with RGB channels only.

Methods achieved noticeably higher accuracy scores when trained on PBR images than when trained on "render & paste" images. Although adding real training images yielded even higher scores, competitive results were achieved with PBR images only – out of the 26 evaluated methods, the fifth was trained only on the PBR images. Interestingly, the increased photorealism from the PBR images led to clear improvements of also the CosyPose method, despite the strong data augmentation applied to the training images.

7.4.1 Evaluated Methods

In total, 26 methods were evaluated on all seven core datasets. Results of 11 methods were received in 2019 and results of 15 methods in 2020 (column Year in Table 7.3). Compared to the "classical" methods evaluated in the 2017 edition of the challenge, which mostly do not involve any machine learning techniques and are prevailed by methods based on the point pair features, the majority of methods evaluated in the 2019 and 2020 editions reflect the current trend in computer vision and utilize deep neural networks. The following paragraphs provide a brief overview of the evaluated methods. Details are provided in Chapter 2 and in the documentation of submissions on the project website.

PPF-Based Methods. The method by Drost et al. [63], which introduced the point pair features, is evaluated in four versions, including the original version with two parameter settings and two extensions utilizing intensity and depth edges for voting and refinement. Evaluated is also the method by Vidal et al. [257].

DNN+PPF-Based Methods. The hybrid method by Félix and Neves [214, 206] segments object instances by Mask R-CNN [89], selects the corresponding part of the 3D point cloud of the test scene for each instance mask, estimates the object pose using the point pair features [63], and refines the pose with ICP. The method of König and Drost [135] follows the same pipeline, with the instance masks predicted by RetinaMask [73] or Mask R-CNN [89], whichever performs better on the validation set.

DNN-Based Methods. The EPOS method described in Chapter 4, Pix2Pose by Park et al. [188], DPOD by Zakharov et al. [277], and the method by Liu et al. [152] predict 2D-3D correspondences with deep neural networks and solve for the object pose using variants of the PnP-RANSAC algorithm. The CDPN method by Li et al. [149] estimates the 3D rotation from 2D-3D correspondences and the 3D translation by directly regressing a scale-invariant 3D translation vector. The CosyPose method by Labbé et al. [139] first predicts 2D bounding boxes of the objects using Mask R-CNN [89], and then applies to each box a DNN model for coarse pose estimation followed by a DNN model for iterative refinement. The AAE method by Sundermeyer et al. [237] localizes the objects with 2D bounding boxes and calculates a descriptor of each detected region by the so-called Augmented Autoencoder, which consists of an encoder mapping the image region to a latent descriptor space and a decoder mapping the descriptor to a denoised reconstruction of the input image region. The later method by Sundermeyer et al. [235] shares a single encoder among multiple objects, which dramatically improves the scalability of the method. The PointVoteNet2 method by Hagelskjær and Buch [87] estimates the object poses from a point cloud using a PointNet backbone [202].

7.4.2 Experimental Setup

Both 2019 and 2020 editions of the challenge followed the evaluation methodology described in Section 7.1. A method had to be evaluated at least on the seven core datasets (LM-O, T-LESS, TUD-L, IC-BIN, ITODD, HB, YCB-V) to be considered for the main challenge awards [109]. Only subsets of test images were used to remove redundancies and speed up the evaluation, and only object instances for which at least 10% of the projected surface area is visible in the test images were to be localized. The participants had to follow the same rules for training and test as in the 2017 edition.

7.4.3 Results

In 2020, methods based on deep neural networks finally caught up with methods based on point pair features – four DNN-based and one DNN+PPF-based method from 2020 outperform the method of Vidal et al. [257], the PPF-based winner of the challenges from 2017 and 2019 (columns PPF and DNN in Table 7.3). The neural networks are in all but one case applied only to the RGB channels, while the depth channel is often used for an ICP refinement (columns Train, Test, Refine). Only the PointVoteNet2 method [87] applies a neural network to the RGB and depth channels. It is noteworthy that the overall third DNN-based method does not use the depth channel at all.

Three of the top five methods, including the top-performing one, are single-view variants of the CosyPose method. The top variant, with the AR_{Core} score of 69.8%, additionally applies a depth-based ICP refinement which improves the score by 6.1% (method #1 vs. #3 in Table 7.3). One of the key ingredients of CosyPose is a strong data augmentation

#	Method	Avg.	LM-O	T-LESS	TUD-L	IC-BIN	ITODD	HB	YCB-V	Time
1	CosyPose-ECCV20-Synt+Real-ICP [139]	69.8	71.4	70.1	93.9	64.7	31.3	71.2	86.1	13.74
2	König-Hybrid-DL-PointPairs [135]	63.9	63.1	65.5	92.0	43.0	48.3	65.1	70.1	0.63
3	CosyPose-ECCV20-Synt+Real [139]	63.7	63.3	72.8	82.3	58.3	21.6	65.6	82.1	0.45
4	Pix2Pose-BOP20_w/ICP-ICCV19 [188]	59.1	58.8	51.2	82.0	39.0	35.1	69.5	78.0	4.84
5	CosyPose-ECCV20-PBR [139]	57.0	63.3	64.0	68.5	58.3	21.6	65.6	57.4	0.47
6	Vidal-Sensors18 [257]	56.9	58.2	53.8	87.6	39.3	43.5	70.6	45.0	3.22
7	CDPNv2_BOP20-RGB-ICP [149]	56.8	63.0	46.4	91.3	45.0	18.6	71.2	61.9	1.46
8	Drost-CVPR10-Edges [63]	55.0	51.5	50.0	85.1	36.8	57.0	67.1	37.5	87.57
9	CDPNv2_BOP20-PBR-ICP [149]	53.4	63.0	43.5	79.1	45.0	18.6	71.2	53.2	1.49
10	CDPNv2_BOP20-RGB [149]	52.9	62.4	47.8	77.2	47.3	10.2	72.2	53.2	0.94
11	Drost-CVPR10-3D-Edges [63]	50.0	46.9	40.4	85.2	37.3	46.2	62.3	31.6	80.06
12	Drost-CVPR10-3D-Only [63]	48.7	52.7	44.4	77.5	38.8	31.6	61.5	34.4	7.70
13	CDPN_BOP19-RGB [149]	47.9	56.9	49.0	76.9	32.7	6.7	67.2	45.7	0.48
14	CDPNv2_BOP20-PBR [149]	47.2	62.4	40.7	58.8	47.3	10.2	72.2	39.0	0.98
15	leaping from 2D to 6D [152]	47.1	52.5	40.3	75.1	34.2	7.7	65.8	54.3	0.42
16	EPOS-BOP20-PBR [99]	45.7	54.7	46.7	55.8	36.3	18.6	58.0	49.9	1.87
17	Drost-CVPR10-3D-Only-Faster [63]	45.4	49.2	40.5	69.6	37.7	27.4	60.3	33.0	1.38
18	Félix&Neves-ICRA17-IET19 [214, 206]	41.2	39.4	21.2	85.1	32.3	6.9	52.9	51.0	55.78
19	Sundermeyer-IJCV19+ICP [237]	39.8	23.7	48.7	61.4	28.1	15.8	50.6	50.5	0.86
20	Zhigang-CDPN-ICCV19 [149]	35.3	37.4	12.4	75.7	25.7	7.0	47.0	42.2	0.51
21	PointVoteNet2 [87]	35.1	65.3	0.4	67.3	26.4	0.1	55.6	30.8	-
22	Pix2Pose-BOP20-ICCV19 [188]	34.2	36.3	34.4	42.0	22.6	13.4	44.6	45.7	1.22
23	Sundermeyer-IJCV19 [237]	27.0	14.6	30.4	40.1	21.7	10.1	34.6	37.7	0.19
24	SingleMultiPathEncoder-CVPR20 [235]	24.1	21.7	31.0	33.4	17.5	6.7	29.3	28.9	0.19
25	Pix2Pose-BOP19-ICCV19 [188]	20.5	7.7	27.5	34.9	21.5	3.2	20.0	29.0	0.79
26	DPOD (synthetic) [277]	16.1	16.9	8.1	24.2	13.0	0.0	28.6	22.2	0.23

#	Method	Year	PPF	DNN	Train	...type	Test	Refine
1	CosyPose-ECCV20-Synt+Real-ICP [139]	2020	-	3/set	rgb	pbr+real	rgb-d	rgb+icp
2	König-Hybrid-DL-PointPairs [135]	2020	yes	1/set	rgb	syn+real	rgb-d	icp
3	CosyPose-ECCV20-Synt+Real [139]	2020	-	3/set	rgb	pbr+real	rgb	rgb
4	Pix2Pose-BOP20_w/ICP-ICCV19 [188]	2020	-	1/obj	rgb	pbr+real	rgb-d	icp
5	CosyPose-ECCV20-PBR [139]	2020	-	3/set	rgb	pbr	rgb	rgb
6	Vidal-Sensors18 [257]	2019	yes	-	-	-	d	icp
7	CDPNv2_BOP20-RGB-ICP [149]	2020	-	1/obj	rgb	pbr+real	rgb-d	icp
8	Drost-CVPR10-Edges [63]	2019	yes	-	-	-	rgb-d	icp
9	CDPNv2_BOP20-PBR-ICP [149]	2020	-	1/obj	rgb	pbr	rgb-d	icp
10	CDPNv2_BOP20-RGB [149]	2020	-	1/obj	rgb	pbr+real	rgb	-
11	Drost-CVPR10-3D-Edges [63]	2019	yes	-	-	-	d	icp
12	Drost-CVPR10-3D-Only [63]	2019	yes	-	-	-	d	icp
13	CDPN_BOP19-RGB [149]	2020	-	1/obj	rgb	pbr+real	rgb	-
14	CDPNv2_BOP20-PBR [149]	2020	-	1/obj	rgb	pbr	rgb	-
15	leaping from 2D to 6D [152]	2020	-	1/obj	rgb	pbr+real	rgb	-
16	EPOS-BOP20-PBR [99]	2020	-	1/set	rgb	pbr	rgb	-
17	Drost-CVPR10-3D-Only-Faster [63]	2019	yes	-	-	-	d	icp
18	Félix&Neves-ICRA17-IET19 [214, 206]	2019	yes	1/set	rgb-d	syn+real	rgb-d	icp
19	Sundermeyer-IJCV19+ICP [237]	2019	-	1/obj	rgb	syn+real	rgb-d	icp
20	Zhigang-CDPN-ICCV19 [149]	2019	-	1/obj	rgb	syn+real	rgb	-
21	PointVoteNet2 [87]	2020	-	1/obj	rgb-d	pbr	rgb-d	icp
22	Pix2Pose-BOP20-ICCV19 [188]	2020	-	1/obj	rgb	pbr+real	rgb	-
23	Sundermeyer-IJCV19 [237]	2019	-	1/obj	rgb	syn+real	rgb	-
24	SingleMultiPathEncoder-CVPR20 [235]	2020	-	1/all	rgb	syn+real	rgb	-
25	Pix2Pose-BOP19-ICCV19 [188]	2019	-	1/obj	rgb	syn+real	rgb	-
26	DPOD (synthetic) [277]	2019	-	1/scene	rgb	syn	rgb	-

Table 7.3. **Results of the BOP Challenge 2019 and 2020.** The methods are ranked by the AR_{Core} score (the third column of the upper table) which is the average of the per-dataset AR_D scores (the following seven columns). The scores (in %) are defined in Section 7.1.6. The last column of the upper table shows the average image processing time averaged over the datasets (in seconds). The lower table shows properties discussed in Sections 7.4.3 and 7.4.4.

technique similar to [236]. As reported in [139], using the augmentation for training the pose estimation models improved the accuracy on T-LESS from 37.0% to 63.8%. Access to a GPU cluster was also crucial as training of one network took ~10 hours on 32 GPU's.

The second is a hybrid method by König and Drost with AR_{Core} of 63.9%. This method is noticeably faster than the top-performing CosyPose variant, mainly thanks to a highly optimized implementation of the ICP algorithm from HALCON [177].

Another method that outperformed Vidal et al. is Pix2Pose with AR_{Core} of 59.1%. The ICP refinement is crucial for this method as it improves the score by an absolute 24.9% and teleports the method from the 22nd to the 4th place. The importance of refinement was also demonstrated by other methods – top nine methods applied ICP or an RGB-based refiner, similar to DeepIM [147] (column Refine in Table 7.3).

Training a special DNN model per object has been a common practice in the field, also followed by most participants of the 2020 edition of the challenge. However, the CosyPose and König-Hybrid methods showed that a single DNN model can be effectively shared among multiple objects (column DNN in Table 7.3). CosyPose trains three models per dataset – one for detection, one for coarse pose estimation, and one for iterative pose refinement, whereas König-Hybrid trains only one model for instance segmentation.

Overall, the EPOS method trained on the PBR images placed 16th. The better performing methods applied an RGB-based or depth-based post-refinement of the estimated poses, used real training images, trained one neural network per object instead of one network per dataset, or applied strong data augmentation. As discussed in Chapter 4, EPOS would likely benefit from these design choices as well.

7.4.4 The Effectiveness of Photorealistic Training Images

In 2020, most DNN-based methods were trained either only on the photorealistic (PBR) training images, or also on real training images which are available in datasets T-LESS, TUD-L, and YCB-V (column Train type in Table 7.3)[3]. Although adding real training images yields higher scores (compare scores of methods #3 and #5 or #10 and #14 on T-LESS, TUD-L, and YCB-V in Table 7.3), competitive results can be achieved with PBR images only, as demonstrated by the overall fifth PBR-only variant of the CosyPose method. This is an important result considering that PBR-only training does not require any human effort for capturing and annotating real training images.

The PBR training images yield a noticeable improvement over the "render & paste" synthetic images obtained by OpenGL rendering of the 3D object models on real photographs. For example, the CDPN method with the same hyper-parameter settings im-

[3]Method #2 used also synthetic training images obtained by cropping the objects from real validation images in the case of HB and ITODD and from OpenGL-rendered images in the case of other datasets, and pasting the cropped objects on images from Microsoft COCO [151]. Method #24 used PBR and real images for training Mask R-CNN [89] and OpenGL images for training a single Multi-path encoder. Two of the CosyPose variants (#1 and #3) also added the "render & paste" synthetic images provided in the original YCB-V dataset, but these images were later found to have no effect on the accuracy score.

Detection	Pose estim.	T-LESS	TUD-L	YCB-V
PBR+Real	PBR+Real	72.8	82.3	82.1
PBR	PBR	64.0	68.5	57.4
PBR	Render & paste v1	16.1	60.4	44.9
PBR	Render & paste v2	60.0	58.9	58.5
Render & paste v1	Render & paste v1	6.1	49.5	26.5
Render & paste v2	Render & paste v2	45.3	42.4	25.7

Table 7.4. **The effect of different training images**. Shown are the AR_{Core} scores achieved by the CosyPose method [139] when different types of images were used for training its object detection (i.e. Mask R-CNN [89]) and pose estimation stage. The "render & paste v1" images were obtained by OpenGL rendering of the 3D object models on random real photographs. The "render & paste v2" images were obtained similarly, but the CAD models of T-LESS objects were assigned a random surface texture instead of a random gray value, the background of most images was assigned a synthetic texture, and 1M instead of 50K images were generated. Interestingly, the increased photorealism brought by the PBR images yields noticeable improvements despite the strong data augmentation applied by CosyPose to the training images.

proved by absolute 20.2% on HB, by 19.5% on LM-O, and by 7% on IC-BIN when trained on 50K PBR images per dataset vs. 10K "render & paste" images per object (compare methods #13 and #20 in Table 7.3). As shown in Table 7.4, the CosyPose method improved by a significant 57.9% (from 6.1% to 64.0%) on T-LESS, by 19.0% on TUD-L, and by 30.9% on YCB-V when trained on 50K PBR images per dataset vs. 50K "render & paste v1" images per dataset. The "render & paste v1" images used for training CosyPose were obtained by imitating the PBR images, i.e., the 3D object models were rendered in the same poses as in the PBR images and pasted on real backgrounds.

As an additional experiment, we trained the CosyPose method on another variant of the "render & paste" images, generated as in [139] and referred to as "render & paste v2". The main differences compared to the "render & paste v1" variant described in the previous paragraph are: (a) the CAD models of T-LESS objects were assigned a random surface texture instead of a random gray value, (b) the background was assigned a real photograph in 30% images and a synthetic texture in 70% images, and (c) 1M instead of 50K images were generated. As shown in Table 7.4, "render & paste v2" images yield a noticeable improvement of 39.2% over "render & paste v1" on T-LESS, but no improvement on TUD-L (−7.1%) and YCB-V (−0.8%). This may suggest that randomizing the surface texture of the texture-less CAD models of T-LESS objects improves the generalization of the network by forcing the network to focus more on shape than on lower-level patterns, as found in [78]. When generating the PBR images, which yield the highest accuracy on T-LESS, the CAD models were assigned a random gray value, as in "render & paste v1", but the effect of randomizing the surface texture may have been achieved by randomizing the PBR material (Section 5.5) – further investigation is needed to clearly answer these questions. The importance of both the objects and the background being synthetic, as suggested in [98], was not confirmed in this experiment – "render & paste v1" images with

only real backgrounds achieved higher scores than "render & paste v2" images on TUD-L and YCB-V. However, the first ten convolutional layers of Mask R-CNN ("conv1" and "conv2_x" of ResNet-50 [90]) used for object detection in the CosyPose method were pre-trained on Microsoft COCO [151] but not fine-tuned, whereas all layers were fine-tuned in [98]. The benefit of having 1M vs. 50K images is indecisive since 50K PBR images were sufficient to achieve high scores.

Both types of "render & paste" images are far inferior compared to the PBR images, which yield an average improvement of 35.9% over "render & paste v1" and 25.5% over "render & paste v2" images (Table 7.4). Interestingly, the increased photorealism brought by the PBR images is important despite the strong data augmentation which the CosyPose method applies to the training images. Since the poses of objects in PBR and "render & paste v1" images are identical, the ray-tracing rendering technique, PBR materials and objects realistically embedded in synthetic environments seem to be the decisive factors for successful "sim2real" transfer [54].

We also observed that the PBR images are more important for training DNN models for object detection/segmentation (e.g., Mask R-CNN [89]) than for training DNN models for pose estimation from the detected regions (Table 7.4). In the case of CosyPose, if the detection model is trained on PBR images and the later two models for pose estimation are trained on the "render & paste v2" instead of the PBR images, the accuracy drops moderately (64.0% to 60.0% on T-LESS, 68.5% to 58.9% on TUD-L) or does not change much (57.4% vs. 58.5% on YCB-V). However, if also the detection model is trained on the "render & paste v1" or "render & paste v2" images, the accuracy drops severely (the low accuracy achieved with "render & paste v1" on T-LESS was discussed earlier).

Chapter 8

Conclusion

We addressed the problem of estimating the 6D pose of specific rigid objects from a single RGB or RGB-D input image. We primarily focused on the problem of 6D object localization, where the identifiers of visible object instances are provided together with the input image, and assumed that the 3D object models are available.

Starting on the shoulders of giants who had worked on this problem for decades, we made contributions that address limitations of the existing methods, enable effective training of learning-based methods without capturing any real images, introduce an industry-relevant dataset with new challenges, and set a standard for benchmarking.

First, we proposed EPOS, a method based on a deep neural network that predicts 2D-3D correspondences between densely sampled pixels of the input RGB image and 3D locations on the object model, and solves for the object poses using an efficient variant of the PnP-RANSAC algorithm. The key idea is to represent an object by a controllable number of compact surface fragments, which allows handling object symmetries by predicting multiple potential 2D-3D correspondences at each pixel and ensures a consistent number and uniform coverage of candidate 3D locations on objects of any type. In the BOP Challenge 2019, EPOS outperformed all RGB and most RGB-D and D methods on the T-LESS and LM-O datasets. On the YCB-V dataset, it was superior to all competitors, with a large margin over the second-best RGB method.

Second, we presented HashMatch, an RGB-D template matching method that applies a cascade of evaluation stages to each sliding window location, which avoids exhaustive matching against all templates. The key stage of the cascade is a voting procedure based on hashing, which substantially reduces the number of candidate templates (usually by three orders of magnitude) before a more expensive template verification stage, and makes the time complexity of the method largely unaffected by the number of templates. The method was fourth out of the 15 participants in the BOP Challenge 2017 and was successfully deployed in DARWIN, a robotic assembly project.

Third, we presented ObjectSynth, an approach to synthesize photorealistic training images of 3D object models. The 3D models of objects are arranged in 3D models of complete indoor scenes with realistic materials and lighting, plausible geometric configurations of objects and cameras are generated by physics simulation, and a high degree of visual realism is achieved by physically-based rendering (PBR). The generated images were shown to yield significant improvements in 2D object detection and 6D object pose

estimation, compared to the commonly used training images synthesized by rendering object models on top of random photographs. A refined version of the proposed synthesis approach was implemented in BlenderProc4BOP, a new open-source and light-weight physically-based renderer and procedural data generator. BlenderProc4BOP was used to synthesize photorealistic training images for the participants of the BOP Challenge 2020. Although adding real training images yielded even higher scores, competitive results were achieved with the photorealistic training images only – out of the 26 evaluated methods, the fifth method was trained only on the photorealistic images.

Fourth, we created T-LESS, a publicly available dataset for training and testing methods for 6D object pose estimation. The dataset includes 3D models and training and test RGB-D images of thirty commodity electrical parts. Recognition and pose estimation of these objects is challenging because the objects have no significant texture or discriminative color, exhibit symmetries and similarities in shape and size, and some objects are a composition of others. Objects exhibiting similar properties are common in industrial environments and T-LESS is the first dataset to include such objects.

Fifth, we lead the BOP benchmark with the goal to capture the status quo in the field and systematically measure the progress in the future. The benchmark currently comprises of eleven datasets in a unified format which cover various practical scenarios, an evaluation methodology with three new pose-error functions which address limitations of the previously used functions, an online evaluation system which is open for continuous submission of new results and reports the current state of the art, and public challenges held at the International Workshops on Recovering 6D Object Pose which we organize annually at the ICCV and ECCV conferences. The recent BOP Challenge 2020 showed that methods based on neural networks finally caught up with methods based on point pair features, which were dominating previous editions of the challenge. Although the top-performing methods rely on RGB-D image channels, strong results were achieved with RGB channels only, also thanks to the provided photorealistic training images.

As research never ends, other interesting problems in the field of object pose estimation remain open, with methods tackling some of them starting to emerge. Examples include pose estimation of piece-wise rigid objects [168], flexible objects [5], or object categories [264, 244, 30, 33], and learning to recognize objects in a model-free [26, 187, 189], few-shot [187, 189], weakly-supervised [223], or a self-supervised manner [263, 229].

Bibliography

[1] Aitor Aldoma, Thomas Fäulhammer, and Markus Vincze. "Automation of "ground truth" annotation for multi-view RGB-D object instance recognition datasets". In: *IROS* (2014). repo.acin.tuwien.ac.at/tmp/permanent/dataset_index.php.

[2] Aitor Aldoma, Federico Tombari, Luigi Di Stefano, and Markus Vincze. "A global hypotheses verification method for 3D object recognition". In: *ECCV* (2012). users.acin.tuwien.ac.at/aaldoma/datasets/ECCV.zip.

[3] Bogdan Alexe, Thomas Deselaers, and Vittorio Ferrari. "What is an object?" In: *CVPR* (2010).

[4] Bogdan Alexe, Thomas Deselaers, and Vittorio Ferrari. "Measuring the Objectness of Image Windows". In: *TPAMI* (2012).

[5] Rıza Alp Güler, Natalia Neverova, and Iasonas Kokkinos. "DensePose: Dense human pose estimation in the wild". In: *CVPR* (2018).

[6] Phil Ammirato, Jonathan Tremblay, Ming-Yu Liu, Alexander Berg, and Dieter Fox. "SymGAN: Orientation estimation without annotation for symmetric objects". In: *WACV* (2020).

[7] Carlos Arteta, Victor Lempitsky, and Andrew Zisserman. "Counting in the wild". In: *ECCV* (2016).

[8] Dániel Baráth and Jiří Matas. "Graph-Cut RANSAC". In: *CVPR* (2018).

[9] Dániel Baráth and Jiří Matas. "Progressive-X: Efficient, Anytime, Multi-Model Fitting Algorithm". In: *ICCV* (2019).

[10] Dominik Bauer, Timothy Patten, and Markus Vincze. "VeREFINE: Integrating Object Pose Verification With Physics-Guided Iterative Refinement". In: *RAL* (2020).

[11] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. "SURF: Speeded up robust features". In: *ECCV* (2006).

[12] Jeffrey S. Beis and David G. Lowe. "Indexing without invariants in 3d object recognition". In: *TPAMI* (1999).

[13] Paul J Besl and Neil D McKay. "A Method for Registration of 3-D Shapes". In: *TPAMI* (1992).

[14] Tolga Birdal and Slobodan Ilic. "Point pair features based object detection and pose estimation revisited". In: *3DV* (2015).

[15] *BlenderProc4BOP*. 2020. URL: https://github.com/DLR-RM/BlenderProc/blob/master/README_BlenderProc4BOP.md.

[16] Ujwal Bonde, Vijay Badrinarayanan, and Roberto Cipolla. "Robust Instance Recognition in Presence of Occlusion and Clutter". In: *ECCV* (2014). `sites.google.com/site/ujwalbonde/publications/downloads`.

[17] Eric Brachmann, Alexander Krull, Frank Michel, Stefan Gumhold, Jamie Shotton, and Carsten Rother. "Learning 6D object pose estimation using 3D object coordinates". In: *ECCV* (2014).

[18] Eric Brachmann, Frank Michel, Alexander Krull, Michael Ying Yang, Stefan Gumhold, and Carsten Rother. "Uncertainty-Driven 6D Pose Estimation of Objects and Scenes from a Single RGB Image". In: *CVPR* (2016).

[19] Gary Bradski and Adrian Kaehler. *Learning OpenCV: Computer vision with the OpenCV library*. O'Reilly Media, Inc., 2008.

[20] Roberto Brunelli. "Template matching techniques in computer vision: Theory and practice". In: *John Wiley & Sons* (2009).

[21] Anders G Buch, Henrik G Petersen, and Norbert Krüger. "Local shape feature fusion for improved matching, pose estimation and 3D object recognition". In: *SpringerPlus* (2016).

[22] Anders Glent Buch, Lilita Kiforenko, and Dirk Kraft. "Rotational Subgroup Voting and Pose Clustering for Robust 3D Object Recognition". In: *ICCV* (2017).

[23] Anders Glent Buch and Dirk Kraft. "Local Point Pair Feature Histogram for Accurate 3D Matching". In: *BMVC* (2018).

[24] Mai Bui, Shadi Albarqouni, Michael Schrapp, Nassir Navab, and Slobodan Ilic. "X-Ray PoseNet: 6 DoF pose estimation for mobile X-Ray devices". In: *WACV* (2017).

[25] Hongping Cai, Tomáš Werner, and Jiří Matas. "Fast detection of multiple textureless 3-D objects". In: *ICVS* (2013).

[26] Ming Cai and Ian Reid. "Reconstruct Locally, Localize Globally: A Model Free Method for Object Pose Estimation". In: *CVPR* (2020).

[27] Berk Calli, Arjun Singh, Aaron Walsman, Siddhartha Srinivasa, Pieter Abbeel, and Aaron M Dollar. "The YCB object and model set: Towards common benchmarks for manipulation research". In: *ICAR* (2015).

[28] John Canny. "A computational approach to edge detection". In: *TPAMI* (1986).

[29] Owen Carmichael and Martial Hebert. "Object recognition by a cascade of edge probes". In: *BMVC* (2002).

[30] Dengsheng Chen, Jun Li, Zheng Wang, and Kai Xu. "Learning canonical shape space for category-level 6D object pose and size estimation". In: *CVPR* (2020).

[31] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. "Encoder-decoder with atrous separable convolution for semantic image segmentation". In: *ECCV* (2018).

[32] Wei Chen, Xi Jia, Hyung Jin Chang, Jinming Duan, and Ales Leonardis. "G2L-Net: Global to Local Network for Real-time 6D Pose Estimation with Embedding Vector Features". In: *CVPR* (2020).

[33] Xu Chen, Zijian Dong, Jie Song, Andreas Geiger, and Otmar Hilliges. "Category Level Object Pose Estimation via Neural Analysis-by-Synthesis". In: *ECCV* (2020).

[34] Ming-Ming Cheng, Ziming Zhang, Wen-Yan Lin, and Philip Torr. "BING: Binarized normed gradients for objectness estimation at 300fps". In: *CVPR* (2014).

[35] Alex Yong-Sang Chia, Susanto Rahardja, Deepu Rajan, and Maylor Karhang Leung. "Object recognition by discriminative combinations of line segments and ellipses". In: *CVPR* (2010).

[36] C. Choi and H.I. Christensen. "3D Pose Estimation of Daily Objects Using an RGB-D Camera". In: *IROS* (2012).

[37] Changhyun Choi, Yuichi Taguchi, Oncel Tuzel, Ming-Yu Liu, and Srikumar Ramalingam. "Voting-based pose estimation for robotic assembly using a 3D sensor". In: (2012).

[38] François Chollet. "Xception: Deep learning with depthwise separable convolutions". In: *CVPR* (2017).

[39] Ondrej Chum and Jiří Matas. "Matching with PROSAC – Progressive sample consensus". In: *CVPR)* (2005).

[40] Paolo Cignoni, Marco Callieri, Massimiliano Corsini, Matteo Dellepiane, Fabio Ganovelli, and Guido Ranzuglia. "MeshLab: an Open-Source Mesh Processing Tool". In: *Eurographics Italian Chapter Conf.* (2008).

[41] Paolo Cignoni, Claudio Rocchini, and Roberto Scopigno. "Metro: measuring error on simplified surfaces". In: *Computer Graphics Forum* (1998).

[42] Alvaro Collet, Manuel Martinez, and Siddhartha S Srinivasa. "The MOPED framework: Object recognition and pose estimation for manipulation". In: *IJRR* (2011).

[43] K Collins, AJ Palmer, and K Rathmill. "The development of a European benchmark for the comparison of assembly robot programming systems". In: *Robot technology and applications* (1985).

[44] Blender Online Community. *Blender – a 3D modelling and rendering package*. Blender Foundation, 2018. URL: `http://www.blender.org`.

[45] Enric Corona, Kaustav Kundu, and Sanja Fidler. "Pose estimation for objects with rotational symmetry". In: *IROS* (2018).

[46] Alberto Crivellaro, Mahdi Rad, Yannick Verdie, Kwang Moo Yi, Pascal Fua, and Vincent Lepetit. "A Novel Representation of Parts for Accurate 3D Object Detection and Tracking in Monocular Images". In: *ICCV* (2015). `cvlab.epfl.ch/data/3d_object_tracking`.

[47] Alberto Crivellaro, Mahdi Rad, Yannick Verdie, Kwang Moo Yi, Pascal Fua, and Vincent Lepetit. "Robust 3D object tracking from monocular images using stable parts". In: *TPAMI* (2017).

[48] Gabriela Csurka. "A comprehensive survey on domain adaptation for visual applications". In: *Domain adaptation in computer vision applications* (2017).

[49] 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).

[50] Oscar Danielsson, Stefan Carlsson, and Josephine Sullivan. "Automatic learning and extraction of multi-local features". In: *ICCV* (2009).

[51] *DARWIN: Dextrous Assembler Robot Working with embodied Intelligence*. `https://cordis.europa.eu/project/id/270138`. 2015.

[52] Lennart Demes. *CC0 Textures*. `https://cc0textures.com/`. 2020.

[53] Xinke Deng, Arsalan Mousavian, Yu Xiang, Fei Xia, Timothy Bretl, and Dieter Fox. "PoseRBPF: A Rao-Blackwellized particle filter for 6D object pose tracking". In: *RSS* (2019).

[54] Maximilian Denninger, Martin Sundermeyer, Dominik Winkelbauer, Dmitry Olefir, Tomáš Hodaň, Youssef Zidan, Mohamad Elbadrawy, Markus Knauer, Harinandan Katam, and Ahsan Lodhi. "BlenderProc: Reducing the Reality Gap with Photorealistic Rendering". In: *RSS Workshops* (2020).

[55] Maximilian Denninger, Martin Sundermeyer, Dominik Winkelbauer, Youssef Zidan, Dmitry Olefir, Mohamad Elbadrawy, Ahsan Lodhi, and Harinandan Katam. "BlenderProc". In: *arXiv preprint arXiv:1911.01911* (2019).

[56] Santosh K Divvala, Derek Hoiem, James H Hays, Alexei A Efros, and Martial Hebert. "An empirical study of context in object detection". In: *CVPR* (2009).

[57] Thanh-Toan Do, Trung Pham, Ming Cai, and Ian Reid. "Real-time monocular object instance 6D pose estimation". In: *BMVC* (2019).

[58] Piotr Dollár and C Lawrence Zitnick. "Structured forests for fast edge detection". In: *ICCV* (2013).

[59] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick Van Der Smagt, Daniel Cremers, and Thomas Brox. "Flownet: Learning optical flow with convolutional networks". In: *ICCV* (2015).

[60] Andreas Doumanoglou, Rigas Kouskouridas, Sotiris Malassiotis, and Tae-Kyun Kim. "Recovering 6D Object Pose and Predicting Next-Best-View in the Crowd". In: *CVPR* (2016).

[61] Bertram Drost and Slobodan Ilic. "3D object detection and localization using multimodal point pair features". In: *Second International Conference on 3D Imaging, Modeling, Processing, Visualization & Transmission* (2012).

[62] Bertram Drost, Markus Ulrich, Paul Bergmann, Philipp Hartinger, and Carsten Steger. "Introducing MVTec ITODD – A dataset for 3D object recognition in industry". In: *ICCVW* (2017).

[63] Bertram Drost, Markus Ulrich, Nassir Navab, and Slobodan Ilic. "Model globally, match locally: Efficient and robust 3D object recognition". In: *CVPR* (2010).

[64] Nikita Dvornik, Julien Mairal, and Cordelia Schmid. "Modeling visual context is key to augmenting object detection datasets". In: *ECCV* (2018).

[65] Debidatta Dwibedi, Ishan Misra, and Martial Hebert. "Cut, paste and learn: Surprisingly easy synthesis for instance detection". In: *ICCV* (2017).

[66] Epic Games. *Physically Based Materials in Unreal Engine*. 2020. URL: https://docs.unrealengine.com/en-US/Engine/Rendering/Materials/PhysicallyBased.

[67] Clemens Eppner, Sebastian Höfer, Rico Jonschkowski, Roberto Martín-Martín, Arne Sieverling, Vincent Wall, and Oliver Brock. "Lessons from the Amazon picking challenge: Four aspects of building robotic systems." In: *Robotics: science and systems* (2016).

[68] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. "The PASCAL Visual Object Classes (VOC) Challenge". In: *IJCV* (2010).

[69] *Evermotion*. www.evermotion.org.

[70] Michael Firman. "RGBD datasets: Past, present and future". In: *CVPRW* (2016).

[71] M. A. Fischler and R. C. Bolles. "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography". In: *Communications of the ACM* (1981).

[72] Sergi Foix, Guillem Alenya, and Carme Torras. "Lock-in time-of-flight (ToF) cameras: a survey". In: *Sensors Journal* (2011).

[73] Cheng-Yang Fu, Mykhailo Shvets, and Alexander C Berg. "RetinaMask: Learning to predict masks improves state-of-the-art single-shot detection for free". In: *arXiv preprint arXiv:1901.03353* (2019).

[74] Mingliang Fu and Weijia Zhou. "DeepHMap++: Combined Projection Grouping and Correspondence Learning for Full DoF Pose Estimation". In: *Sensors* (2019).

[75] A Gaidon, Q Wang, Y Cabon, and E Vig. "Virtual Worlds as Proxy for Multi-Object Tracking Analysis". In: *CVPR* (2016).

[76] Ran Gal, Lior Shapira, Eyal Ofek, and Pushmeet Kohli. "FLARE: Fast layout for augmented reality applications". In: *ISMAR* (2014).

[77] Andreas Geiger, Philip Lenz, and Raquel Urtasun. "Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite". In: *CVPR* (2012).

[78] Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A Wichmann, and Wieland Brendel. "ImageNet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness". In: *arXiv preprint arXiv:1811.12231* (2018).

[79] Georgios Georgakis, Md Alimoor Reza, Arsalan Mousavian, Phi-Hung Le, and Jana Kosecka. "Multiview RGB-D Dataset for Object Instance Detection". In: *3DV* (2016). cs.gmu.edu/~robot/gmu-kitchens.html.

[80] Iliyan Georgiev, Thiago Ize, Mike Farnsworth, Ramon Montoya-Vozmediano, Alan King, Brecht Van Lommel, Angel Jimenez, Oscar Anson, Shinji Ogaki, Eric Johnston, et al. "Arnold: A brute-force production path tracer". In: *TOG* (2018).

[81] Clement Godard, Peter Hedman, Wenbin Li, and Gabriel J Brostow. "Multi-view reconstruction of highly specular surfaces in uncontrolled environments". In: *3DV* (2015).

[82] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[83] Ruiqi Guo and Derek Hoiem. "Support surface prediction in indoor scenes". In: *ICCV* (2013). `ttic.uchicago.edu/~rurtasun/rmrc/indoor.php`.

[84] Yulan Guo, Mohammed Bennamoun, Ferdous Sohel, Min Lu, and Jianwei Wan. "3D object recognition in cluttered scenes with local surface features: a survey". In: *TPAMI* (2014).

[85] Yulan Guo, Mohammed Bennamoun, Ferdous Sohel, Min Lu, Jianwei Wan, and Ngai Ming Kwok. "A comprehensive performance evaluation of 3D local feature descriptors". In: *IJCV* (2016).

[86] H. Haber. "Three-Dimensional Proper and Improper Rotation Matrices". In: *University of California, Santa Cruz Physics 116A Lecture Notes* (2011).

[87] Frederik Hagelskjær and Anders Glent Buch. "PointVoteNet: Accurate Object Detection And 6 DOF Pose Estimation In Point Clouds". In: *ICIP* (2020).

[88] Ankur Handa, Viorica Patraucean, Vijay Badrinarayanan, Simon Stent, and Roberto Cipolla. "Understanding real world indoor scenes with synthetic data". In: *CVPR* (2016).

[89] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. "Mask R-CNN". In: *ICCV* (2017).

[90] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep residual learning for image recognition". In: *CVPR* (2016).

[91] Yisheng He, Wei Sun, Haibin Huang, Jianran Liu, Haoqiang Fan, and Jian Sun. "PVN3D: A deep point-wise 3D keypoints voting network for 6DoF pose estimation". In: *CVPR* (2020).

[92] Joel A Hesch and Stergios I Roumeliotis. "A direct least-squares (DLS) method for PnP". In: *ICCV* (2011).

[93] 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". In: *TPAMI* (2012).

[94] 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).

[95] 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).

[96] Stefan Hinterstoisser, Vincent Lepetit, Naresh Rajkumar, and Kurt Konolige. "Going further with point pair features". In: *ECCV* (2016).

[97] Stefan Hinterstoisser, Vincent Lepetit, Paul Wohlhart, and Kurt Konolige. "On Pre-Trained Image Features and Synthetic Images for Deep Learning". In: *ECCVW* (2018).

[98] Stefan Hinterstoisser, Olivier Pauly, Hauke Heibel, Marek Martina, and Martin Bokeloh. "An annotation saved is an annotation earned: Using fully synthetic training for object detection". In: *ICCVW* (2019).

[99] Tomáš Hodaň, Dániel Baráth, and Jiří Matas. "EPOS: Estimating 6D Pose of Objects with Symmetries". In: *CVPR* (2020).

[100] Tomáš Hodaň, Eric Brachmann, Bertram Drost, Frank Michel, Martin Sundermeyer, Jiří Matas, and Carsten Rother. *BOP Challenge 2019*. `https://bop.felk.cvut.cz/challenges/bop-challenge-2019/`.

[101] Tomáš Hodaň, Dima Damen, Walterio Mayol-Cuevas, and Jiří Matas. "Efficient Texture-less Object Detection for Augmented Reality Guidance". In: *ISMARW* (2015).

[102] Tomáš Hodaň, Pavel Haluza, Štěpán Obdržálek, Jiří Matas, Manolis Lourakis, and Xenophon Zabulis. "T-LESS: An RGB-D Dataset for 6D Pose Estimation of Texture-less Objects". In: *WACV* (2017).

[103] Tomáš Hodaň, Rigas Kouskouridas, Tae-Kyun Kim, Jiří Matas, Carsten Rother, Vincent Lepetit, Ales Leonardis, Krzysztof Walas, Carsten Steger, Eric Brachmann, Bertram Drost, and Juil Sock. *5th International Workshop on Recovering 6D Object Pose*. `http://cmp.felk.cvut.cz/sixd/workshop_2019/`. 2019.

[104] Tomáš Hodaň, Rigas Kouskouridas, Tae-Kyun Kim, Federico Tombari, Kostas Bekris, Bertram Drost, Thibault Groueix, Krzysztof Walas, Vincent Lepetit, Ales Leonardis, Carsten Steger, Frank Michel, Caner Sahin, Carsten Rother, and Jiří Matas. "A Summary of the 4th International Workshop on Recovering 6D Object Pose". In: *ECCVW* (2018).

[105] Tomáš Hodaň, Jiří Matas, and Štěpán Obdržálek. "On Evaluation of 6D Object Pose Estimation". In: *ECCVW* (2016).

[106] Tomáš Hodaň, Frank Michel, Eric Brachmann, Wadim Kehl, Anders Glent Buch, Dirk Kraft, Bertram Drost, Joel Vidal, Stephan Ihrke, Xenophon Zabulis, Caner Sahin, Fabian Manhardt, Federico Tombari, Tae-Kyun Kim, Jiří Matas, and Carsten Rother. "BOP: Benchmark for 6D Object Pose Estimation". In: *ECCV* (2018).

[107] Tomáš Hodaň, Frank Michel, Caner Sahin, Tae-Kyun Kim, Jiří Matas, and Carsten Rother. *SIXD Challenge 2017*. `http://cmp.felk.cvut.cz/sixd/challenge_2017/`. 2017.

[108] Tomáš Hodaň and Martin Sundermeyer. *BOP Toolkit*. 2020. URL: `https://github.com/thodan/bop_toolkit`.

[109] Tomáš Hodaň, Martin Sundermeyer, Bertram Drost, Yann Labbé, Eric Brachmann, Frank Michel, Carsten Rother, and Jiří Matas. "BOP Challenge 2020 on 6D Object Localization". In: *ECCVW* (2020).

[110] Tomáš Hodaň, Martin Sundermeyer, Rigas Kouskouridas, Tae-Kyun Kim, Jiří Matas, Carsten Rother, Vincent Lepetit, Ales Leonardis, Krzysztof Walas, Carsten Steger, Eric Brachmann, Bertram Drost, and Juil Sock. *6th International Workshop on Recovering 6D Object Pose*. `http://cmp.felk.cvut.cz/sixd/workshop_2020/`. 2020.

[111] Tomáš Hodaň, Vibhav Vineet, Ran Gal, Emanuel Shalev, Jon Hanzelka, Treb Connell, Pedro Urbina, Sudipta Sinha, and Brian Guenter. "Photorealistic Image Synthesis for Object Instance Detection". In: *ICIP* (2019).

[112] Tomáš Hodaň, Xenophon Zabulis, Manolis Lourakis, Štěpán Obdržálek, and Jiří Matas. "Detection and Fine 3D Pose Estimation of Texture-less Objects in RGB-D Images". In: *IROS* (2015).

[113] Jan Hosang, Rodrigo Benenson, Piotr Dollár, and Bernt Schiele. "What makes for effective detection proposals?" In: *TPAMI* (2015).

[114] Edward Hsiao and Martial Hebert. "Occlusion reasoning for object detection under arbitrary viewpoint". In: *TPAMI* (2014). `www.cs.cmu.edu/~./hebert/occarbview.html`.

[115] Yinlin Hu, Joachim Hugonot, Pascal Fua, and Mathieu Salzmann. "Segmentation-driven 6D object pose estimation". In: *CVPR* (2019).

[116] Jonathan Huang, Vivek Rathod, Chen Sun, Menglong Zhu, Anoop Korattikara, Alireza Fathi, Ian Fischer, Zbigniew Wojna, Yang Song, Sergio Guadarrama, et al. "Speed/accuracy trade-offs for modern convolutional object detectors". In: *CVPR* (2017).

[117] Peter J Huber. "Robust estimation of a location parameter". In: *Breakthroughs in statistics*. 1992.

[118] *Intel Open Image Denoise*. 2020. URL: `https://www.openimagedenoise.org/`.

[119] Hossam Isack and Yuri Boykov. "Energy-based geometric multi-model fitting". In: *IJCV* (2012).

[120] S. Ivekovič, E. Trucco, and Y. Petillot. "Human Body Pose Estimation with Particle Swarm Optimisation". In: *Evolutionary Computation* (2008).

[121] Omid Hosseini Jafari, Siva Karthik Mustikovela, Karl Pertsch, Eric Brachmann, and Carsten Rother. "iPose: Instance-aware 6D pose estimation of partly occluded objects". In: *ACCV* (2018).

[122] Allison Janoch, Sergey Karayev, Yangqing Jia, Jonathan T Barron, Mario Fritz, Kate Saenko, and Trevor Darrell. "A category-level 3D object dataset: Putting the Kinect to work". In: *Consumer Depth Cameras for Computer Vision* (2013). `kinectdata.com`.

[123] Andrew E. Johnson and Martial Hebert. "Using spin images for efficient object recognition in cluttered 3D scenes". In: *TPAMI* (1999).

[124] Troels Bo Jørgensen, Anders Glent Buch, and Dirk Kraft. "Geometric edge description and classification in point cloud data with application to 3D object recognition". In: *VISAPP* (2015).

[125] Philipp Jund, Nichola Abdo, Andreas Eitel, and Wolfram Burgard. "The Freiburg groceries dataset". In: *arXiv preprint arXiv:1611.05799* (2016).

[126] Roman Kaskman, Sergey Zakharov, Ivan Shugurov, and Slobodan Ilic. "HomebrewedDB: RGB-D Dataset for 6D Pose Estimation of 3D Objects". In: *ICCVW* (2019).

[127] Wadim Kehl, Fabian Manhardt, Federico Tombari, Slobodan Ilic, and Nassir Navab. "SSD-6D: Making RGB-based 3D detection and 6D pose estimation great again". In: *ICCV* (2017).

[128] Wadim Kehl, Fausto Milletari, Federico Tombari, Slobodan Ilic, and Nassir Navab. "Deep learning of local RGB-D patches for 3D object detection and 6D pose estimation". In: *ECCV* (2016).

[129] Wadim Kehl, Federico Tombari, Nassir Navab, Slobodan Ilic, and Vincent Lepetit. "Hashmod: A Hashing Method for Scalable 3D Object Detection". In: *BMVC* (2015).

[130] Alex Kendall, Matthew Grimes, and Roberto Cipolla. "PoseNet: A convolutional network for real-time 6-DOF camera relocalization". In: *ICCV* (2015).

[131] Kourosh Khoshelham and Sander Oude Elberink. "Accuracy and Resolution of Kinect Depth Data for Indoor Mapping Applications". In: *Sensors* (2012).

[132] Lilita Kiforenko, Bertram Drost, Federico Tombari, Norbert Krüger, and Anders Glent Buch. "A performance evaluation of point pair features". In: *CVIU* (2018).

[133] Eunyoung Kim and Gerard Medioni. "3D object recognition in range images using visibility context". In: *IROS* (2011).

[134] Laurent Kneip, Davide Scaramuzza, and Roland Siegwart. "A novel parametrization of the perspective-three-point problem for a direct computation of absolute camera position and orientation". In: *CVPR* (2011).

[135] Rebecca Koenig and Bertram Drost. "A Hybrid Approach for 6DoF Pose Estimation". In: *ECCVW* (2020).

[136] Jeff Kramer, Nicolas Burrus, Florian Echtler, Herrera C Daniel, and Matt Parker. *Hacking the Kinect*. Vol. 268. Springer, 2012.

[137] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *NIPS* (2012).

[138] Alexander Krull, Eric Brachmann, Frank Michel, Michael Ying Yang, Stefan Gumhold, and Carsten Rother. "Learning analysis-by-synthesis for 6D pose estimation in RGB-D images". In: *ICCV* (2015).

[139] Yann Labbé, Justin Carpentier, Mathieu Aubry, and Josef Sivic. "CosyPose: Consistent multi-view multi-object 6D pose estimation". In: *ECCV* (2020).

[140] Kevin Lai, Liefeng Bo, Xiaofeng Ren, and Dieter Fox. "A large-scale hierarchical multi-view RGB-D object dataset". In: *ICRA* (2011). `rgbd-dataset.cs.washington.edu`.

[141] Yehezkel Lamdan and Haim J Wolfson. "Geometric hashing: A general and efficient model-based recognition scheme". In: *ICCV* (1988).

[142] Karel Lebeda, Jiří Matas, and Ondřej Chum. "Fixing the locally optimized ransac–full experimental evaluation". In: *BMVC*. 2012.

[143] Ales Leonardis and Horst Bischof. "Dealing with occlusions in the eigenspace approach". In: *CVPR* ().

[144] Marius Leordeanu, Martial Hebert, and Rahul Sukthankar. "Beyond local appearance: Category recognition from pairwise interactions of simple features". In: *CVPR* (2007).

[145] Vincent Lepetit, Francesc Moreno-Noguer, and Pascal Fua. "EPnP: An Accurate O(n) Solution to the PnP Problem". In: *IJCV* (2009).

[146] Chi Li, Jin Bai, and Gregory D Hager. "A unified framework for multi-view multi-class object pose estimation". In: *ECCV* (2018).

[147] Yi Li, Gu Wang, Xiangyang Ji, Yu Xiang, and Dieter Fox. "DeepIM: Deep iterative matching for 6D pose estimation". In: *ECCV* (2018).

[148] Zhengqi Li and Noah Snavely. "CGIntrinsics: Better intrinsic image decomposition through physically-based rendering". In: *ECCV* (2018).

[149] Zhigang Li, Gu Wang, and Xiangyang Ji. "CDPN: Coordinates-Based Disentangled Pose Network for Real-Time RGB-Based 6-DoF Object Pose Estimation". In: *ICCV* (2019).

[150] Joseph J Lim, Hamed Pirsiavash, and Antonio Torralba. "Parsing IKEA Objects: Fine Pose Estimation". In: *ICCV* (2013). `ikea.csail.mit.edu`.

[151] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. "Microsoft COCO: Common objects in context". In: *ECCV* (2014).

[152] Jinhui Liu, Zhikang Zou, Xiaoqing Ye, Xiao Tan, Errui Ding, Feng Xu, and Xin Yu. "Leaping from 2D Detection to Efficient 6DoF Object Pose Estimation". In: *ECCVW* (2020).

[153] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. "SSD: Single shot multibox detector". In: *ECCV* (2016).

[154] Manolis Lourakis and Xenophon Zabulis. "Model-Based Pose Estimation for Rigid Objects". In: *Computer Vision Systems* (2013).

[155] David G Lowe. "Three-dimensional object recognition from single two-dimensional images". In: *Artificial intelligence* (1987).

[156] David G Lowe et al. "Object recognition from local scale-invariant features." In: *ICCV* (1999).

[157] Siddharth Mahendran, Haider Ali, and Rene Vidal. "A mixed classification-regression framework for 3D pose estimation from 2D images". In: *BMVC* (2018).

[158] Fabian Manhardt, Diego Martin Arroyo, Christian Rupprecht, Benjamin Busam, Nassir Navab, and Federico Tombari. "Explaining the Ambiguity of Object Detection and 6D Pose from Visual Data". In: *ICCV* (2019).

[159] Fabian Manhardt, Wadim Kehl, Nassir Navab, and Federico Tombari. "Deep model-based 6D pose refinement in RGB". In: *ECCV* (2018).

[160] Steve Marschner and Peter Shirley. *Fundamentals of computer graphics*. CRC Press, 2015.

[161] Jiří Matas, Ondřej Chum, Martin Urban, and Tomáš Pajdla. "Robust wide-baseline stereo from maximally stable extremal regions". In: *Image and vision computing* (2004).

[162] Jiří Matas and Štěpán Obdržálek. "Object recognition methods based on transformation covariant features". In: *12th European Signal Processing Conference* (2004).

[163] John McCormac, Ankur Handa, Stefan Leutenegger, and Andrew J Davison. "SceneNet RGB-D: Can 5M synthetic images beat generic ImageNet pre-training on indoor segmentation?" In: *ICCV* (2017).

[164] David Meger and James J Little. "Mobile 3D object detection in clutter". In: *IROS* (2011). `www.cs.ubc.ca/labs/lci/vrs`.

[165] Ajmal Mian, Mohammed Bennamoun, and Robyn Owens. "On the repeatability and quality of keypoints for local feature-based 3D object retrieval from cluttered scenes". In: *IJCV* (2010).

[166] Ajmal S Mian, Mohammed Bennamoun, and Robyn Owens. "Three-dimensional model-based object recognition and segmentation in cluttered scenes". In: *TPAMI* (2006). `staffhome.ecm.uwa.edu.au/~00053650/recognition.html`.

[167] Frank Michel, Eric Alexander Kirillov Brachmann, Alexander Krull, Stefan Gumhold, Bogdan Savchynskyy, and Carsten Rother. "Global Hypothesis Generation for 6D Object Pose Estimation". In: *CVPR* (2017).

[168] Frank Michel, Alexander Krull, Eric Brachmann, Michael Ying Yang, Stefan Gumhold, and Carsten Rother. "Pose estimation of kinematic chain instances via object coordinate regression". In: *BMVC* (2015). `cvlab-dresden.de/iccv2015-articulation-challenge`.

[169] Niloy J Mitra, Leonidas J Guibas, and Mark Pauly. "Partial and approximate symmetry detection for 3D geometry". In: *ACM Transactions on Graphics* (2006).

[170] Adam Morawiec. *Orientations and Rotations: Computations in Crystallographic Textures*. Springer Science & Business Media, 2004.

[171] Jorge J Moré. "The Levenberg-Marquardt algorithm: Implementation and theory". In: *Numerical analysis*. 1978.

[172] Yair Movshovitz-Attias, Takeo Kanade, and Yaser Sheikh. "How useful is photo-realistic rendering for visual learning?" In: *ECCV* (2016).

[173] Marius Muja and David G Lowe. "Fast approximate nearest neighbors with automatic algorithm configuration." In: *VISAPP* (2009).

[174] Enrique Muñoz, Yoshinori Konishi, Vittorio Murino, and Alessio Del Bue. "Fast 6D pose estimation for texture-less objects from a single RGB image". In: *ICRA* (2016).

[175] *Mura*. `www.muravision.com`.

[176] Hiroshi Murase and Shree K Nayar. "Visual learning and recognition of 3-D objects from appearance". In: *IJCV* (1995).

[177] *MVTec HALCON*. 2020. URL: `https://www.mvtec.com/halcon/`.

[178] Richard Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew Davison, Pushmeet Kohi, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. "KinectFusion: Real-time dense surface mapping and tracking". In: *ISMAR* (2011).

[179] Apurv Nigam, Adrian Penate-Sanchez, and Lourdes Agapito. "Detect Globally, Label Locally: Learning Accurate 6-DOF Object Pose Estimation by Joint Segmentation and Coordinate Regression". In: *RAL* (2018).

[180] *NIX Color Sensor*. `www.nixsensor.com`.

[181] Štěpán Obdržálek and Jiří Matas. "Object recognition using local affine frames". In: *Proc. Research Reports of CMP, Czech Technical University* (2006).

[182] Markus Oberweger, Mahdi Rad, and Vincent Lepetit. "Making deep heatmaps robust to partial occlusions for 3D object pose estimation". In: *ECCV* (2018).

[183] Markus Oberweger, Paul Wohlhart, and Vincent Lepetit. "Training a feedback loop for hand pose estimation". In: *ICCV* (2015).

[184] Iason Oikonomidis, Nikolaos Kyriazis, and Antonis A. Argyros. "Efficient model-based 3D tracking of hand articulations using Kinect". In: *BMVC* (2011).

[185] Andreas Opelt, Axel Pinz, and Andrew Zisserman. "A boundary-fragment-model for object detection". In: *ECCV* (2006).

[186] Chavdar Papazov and Darius Burschka. "An efficient RANSAC for 3D object recognition in noisy and occluded scenes". In: *ACCV* (2010).

[187] Keunhong Park, Arsalan Mousavian, Yu Xiang, and Dieter Fox. "LatentFusion: End-to-End Differentiable Reconstruction and Rendering for Unseen Object Pose Estimation". In: *CVPR* (2020).

[188] Kiru Park, Timothy Patten, and Markus Vincze. "Pix2Pose: Pixel-Wise Coordinate Regression of Objects for 6D Pose Estimation". In: *ICCV* (2019).

[189] Kiru Park, Timothy Patten, and Markus Vincze. "Neural Object Learning for 6D Pose Estimation Using a Few Cluttered Images". In: *ECCV* (2020).

[190] Yash Patel, Tomáš Hodaň, and Jiří Matas. "Learning Surrogates via Deep Embedding". In: *ECCV* (2020).

[191] Georgios Pavlakos, Xiaowei Zhou, Aaron Chan, Konstantinos G Derpanis, and Kostas Daniilidis. "6-DoF object pose from semantic keypoints". In: *ICRA* (2017).

[192] Sida Peng, Yuan Liu, Qixing Huang, Xiaowei Zhou, and Hujun Bao. "PVNet: Pixel-wise Voting Network for 6DoF Pose Estimation". In: *CVPR* (2019).

[193] Matt Pharr, Wenzel Jakob, and Greg Humphreys. *Physically based rendering: From theory to implementation*. Morgan Kaufmann, 2016.

[194] Giorgia Pitteri, Aurélie Bugeau, Slobodan Ilic, and Vincent Lepetit. "3D Object Detection and Pose Estimation of Unseen Objects in Color Images with Local Surface Embeddings". In: *ACCV* (2020).

[195] Giorgia Pitteri, Slobodan Ilic, and Vincent Lepetit. "CorNet: Generic 3D Corners for 6D Pose Estimation of New Objects without Retraining". In: *ICCVW*. 2019.

[196] Giorgia Pitteri, Michaël Ramamonjisoa, Slobodan Ilic, and Vincent Lepetit. "On Object Symmetries and 6D Pose Estimation from Images". In: *3DV* (2019).

[197] Benjamin Planche, Ziyan Wu, Kai Ma, Shanhui Sun, Stefan Kluckner, Oliver Lehmann, Terrence Chen, Andreas Hutter, Sergey Zakharov, Harald Kosch, et al. "DepthSynth: Real-time realistic synthetic data generation from CAD models for 2.5D recognition". In: *3DV* (2017).

[198] Riccardo Poli, James Kennedy, and Tim Blackwell. "Particle Swarm Optimization". In: *Swarm Intelligence* (2007).

[199] Jean Ponce, Svetlana Lazebnik, Fredrick Rothganger, and Cordelia Schmid. "Toward true 3D object recognition". In: *Reconnaissance de Formes et Intelligence Artificielle* (2004).

[200] Simon JD Prince. *Computer vision: models, learning, and inference*. Cambridge University Press, 2012.

[201] Charles R Qi, Xinlei Chen, Or Litany, and Leonidas J Guibas. "ImVoteNet: Boosting 3D object detection in point clouds with image votes". In: *CVPR* (2020).

[202] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. "PointNet: Deep learning on point sets for 3D classification and segmentation". In: *CVPR* (2017).

[203] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. "Pointnet++: Deep hierarchical feature learning on point sets in a metric space". In: *Advances in neural information processing systems* (2017).

[204] Yiming Qian, Minglun Gong, and Yee Hong Yang. "3D reconstruction of transparent objects with position-normal consistency". In: *CVPR* (2016).

[205] Mahdi Rad and Vincent Lepetit. "BB8: A scalable, accurate, robust to partial occlusion method for predicting the 3D poses of challenging objects without using depth". In: *ICCV* (2017).

[206] Carolina Raposo and Joao P Barreto. "Using 2 point+normal sets for fast registration of point clouds with small overlap". In: *ICRA* (2017).

[207] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. "You only look once: Unified, real-time object detection". In: *CVPR* (2016).

[208] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. "Faster R-CNN: towards real-time object detection with region proposal networks". In: *TPAMI* (2017).

[209] Colin Rennie, Rahul Shome, Kostas E Bekris, and Alberto F De Souza. "A Dataset for Improved RGBD-Based Object Detection and Pose Estimation for Warehouse Pick-and-Place". In: *RA-L* (2016). `www.pracsyslab.org/rutgers_apc_rgbd_dataset`.

[210] Stephan R Richter, Zeeshan Hayder, and Vladlen Koltun. "Playing for benchmarks". In: *ICCV* (2017).

[211] Stephan R Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. "Playing for data: Ground truth from computer games". In: *ECCV* (2016).

[212] Reyes Rios-Cabrera and Tinne Tuytelaars. "Discriminatively trained templates for 3D object detection: A real time scalable approach". In: *ICCV* (2013).

[213] Lawrence G Roberts. "Machine perception of three-dimensional solids". PhD thesis. Massachusetts Institute of Technology, 1963.

[214] Pedro Rodrigues, Michel Antunes, Carolina Raposo, Pedro Marques, Fernando Fonseca, and Joao Barreto. "Deep segmentation leverages geometric pose estimation in computer-aided total knee arthroplasty". In: *Healthcare Technology Letters* (2019).

[215] German Ros, Laura Sellart, Joanna Materzynska, David Vazquez, and Antonio M Lopez. "The SYNTHIA dataset: A large collection of synthetic images for semantic segmentation of urban scenes". In: *CVPR* (2016).

[216] Artem Rozantsev, Mathieu Salzmann, and Pascal Fua. "Beyond sharing weights for deep domain adaptation". In: *TPAMI* (2018).

[217] Szymon Rusinkiewicz and Marc Levoy. "Efficient variants of the ICP algorithm". In: *Third International Conference on 3-D Digital Imaging and Modeling* (2001).

[218] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. "ImageNet Large Scale Visual Recognition Challenge". In: *IJCV* (2015).

[219] Samuele Salti, Federico Tombari, and Luigi Di Stefano. "SHOT: Unique signatures of histograms for surface and texture description". In: *CVIU* (). `www.vision.deis.unibo.it/research/80-shot`.

[220] Daniel Scharstein and Chris Pal. "Learning Conditional Random Fields for Stereo." In: *CVPR* (2007).

[221] Daniel Scharstein and Richard Szeliski. "A Taxonomy and Evaluation of Dense Two-Frame Stereo Correspondence Algorithms". In: *IJCV* (2002).

[222] Christian Schlette et al. "A new benchmark for pose estimation with ground truth from virtual reality". In: *Production Engineering* (2014). `www.mmi.rwth-aachen.de/exchange/data/pesi2014/benchmark.htm`.

[223] Jianzhun Shao, Yuhang Jiang, Gu Wang, Zhigang Li, and Xiangyang Ji. "PFRL: Pose-Free Reinforcement Learning for 6D Pose Estimation". In: *CVPR* (2020).

[224] Jamie Shotton, Andrew Blake, and Roberto Cipolla. "Contour-based learning for object detection". In: *ICCV* (2005).

[225] Dave Shreiner. *OpenGL programming guide: the official guide to learning OpenGL, versions 3.0 and 3.1*. Pearson Education, 2009.

[226] Ashish Shrivastava, Tomas Pfister, Oncel Tuzel, Joshua Susskind, Wenda Wang, and Russell Webb. "Learning from Simulated and Unsupervised Images through Adversarial Training." In: *CVPR* (2017).

[227] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. "Indoor segmentation and support inference from RGBD images". In: *ECCV* (2012).

[228] A. Singh, J. Sha, K. S. Narayan, T. Achim, and P. Abbeel. "BigBIRD: A large-scale 3D database of object instances". In: *ICRA* (2014). `rll.berkeley.edu/bigbird`.

[229] Juil Sock, Guillermo Garcia-Hernando, Anil Armagan, and Tae Kyun Kim. "Introducing Pose Consistency and Warp-Alignment for Self-Supervised 6D Object Pose Estimation in Color Images". In: *3DV* (2020).

[230] Juil Sock, Kwang In Kim, Caner Sahin, and Tae-Kyun Kim. "Multi-task deep networks for depth-based 6D object pose and joint registration in crowd scenarios". In: *BMVC* (2018).

[231] Shuran Song, Samuel P Lichtenberg, and Jianxiong Xiao. "Sun RGB-D: A RGB-D scene understanding benchmark suite". In: *CVPR* (2015). `rgbd.cs.princeton.edu`.

[232] Frank Steinbrücker, Jürgen Sturm, and Daniel Cremers. "Volumetric 3D mapping in real-time on a CPU". In: *ICRA* (2014). `github.com/tum-vision/fastfusion`.

[233] Jürgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. "A benchmark for the evaluation of RGB-D SLAM systems". In: *IROS* (2012).

[234] Hao Su, Charles R Qi, Yangyan Li, and Leonidas J Guibas. "Render for CNN: Viewpoint estimation in images using cnns trained with rendered 3D model views". In: *ICCV* (2015).

[235] Martin Sundermeyer, Maximilian Durner, En Yen Puang, Zoltan-Csaba Marton, Narunas Vaskevicius, Kai O Arras, and Rudolph Triebel. "Multi-path learning for object pose estimation across domains". In: *CVPR* (2020).

[236] Martin Sundermeyer, Zoltan-Csaba Marton, Maximilian Durner, Manuel Brucker, and Rudolph Triebel. "Implicit 3D orientation learning for 6D object detection from RGB images". In: *ECCV* (2018).

[237] Martin Sundermeyer, Zoltan-Csaba Marton, Maximilian Durner, and Rudolph Triebel. "Augmented Autoencoders: Implicit 3D Orientation Learning for 6D Object Detection". In: *IJCV* (2019).

[238] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. "Inception-v4, inception-resnet and the impact of residual connections on learning." In: *AAAI* (2017).

[239] Babak Taati, Michel Bondy, Piotr Jasiobedzki, and Michael Greenspan. "Variable dimensional local shape descriptors for object recognition in range data". In: *ICCV* (2007). `rcvlab.ece.queensu.ca/~qridb/lsdPage.html`.

[240] Jonathan Taylor, Jamie Shotton, Toby Sharp, and Andrew Fitzgibbon. "The Vitruvian Manifold: Inferring dense correspondences for one-shot human pose estimation". In: *CVPR* (2012).

[241] Alykhan Tejani, Danhang Tang, Rigas Kouskouridas, and Tae-Kyun Kim. "Latent-Class Hough Forests for 3D Object Detection and Pose Estimation". In: *ECCV* (2014). `www.iis.ee.ic.ac.uk/rkouskou/research/LCHF.html`.

[242] Bugra Tekin, Sudipta N Sinha, and Pascal Fua. "Real-time seamless single shot 6D object pose prediction". In: *CVPR* (2018).

[243] Grit Thürrner and Charles A Wüthrich. "Computing vertex normals from polygonal facets". In: *Journal of Graphics Tools* (1998).

[244] Meng Tian, Marcelo H Ang, and Gim Hee Lee. "Shape Prior Deformation for Categorical 6D Object Pose and Size Estimation". In: *ECCV* (2020).

[245] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. "Domain randomization for transferring deep neural networks from simulation to the real world". In: *IROS* (2017).

[246] Roberto Toldo and Andrea Fusiello. "Robust multiple structures estimation with j-linkage". In: *ECCV*. 2008.

[247] Federico Tombari, Luigi Di Stefano, and Simone Giardino. "Online learning for automatic segmentation of 3D data". In: *IROS* (2011).

[248] Federico Tombari, Alessandro Franchi, and Luigi Di Stefano. "BOLD features to detect texture-less objects". In: *ICCV* (2013).

[249] Federico Tombari, Samuele Salti, and Luigi Di Stefano. "Performance evaluation of 3D keypoint detectors". In: *IJCV* (2013).

[250] Philip Torr. "Bayesian model estimation and selection for epipolar geometry and generic manifold fitting". In: *IJCV* (2002).

[251] Philip Torr and Andrew Zisserman. "Robust computation and parametrization of multiple view relations". In: *ICCV*. 1998.

[252] Philip Torr and Andrew Zisserman. "MLESAC: A new robust estimator with application to estimating image geometry". In: *CVIU* (2000).

[253] Jonathan Tremblay, Aayush Prakash, David Acuna, Mark Brophy, Varun Jampani, Cem Anil, Thang To, Eric Cameracci, Shaad Boochoon, and Stan Birchfield. "Training deep networks with synthetic data: Bridging the reality gap by domain randomization". In: *CVPRW* (2018).

[254] Jonathan Tremblay, Thang To, Balakumar Sundaralingam, Yu Xiang, Dieter Fox, and Stan Birchfield. "Deep object pose estimation for semantic robotic grasping of household objects". In: *CoRL* (2018).

[255] Zhuowen Tu and Xiang Bai. "Auto-context and its application to high-level vision tasks and 3D brain image segmentation". In: *TPAMI* (2009).

[256] Ranjith Unnikrishnan and Martial Hebert. "Multi-scale interest regions from unorganized point clouds". In: *CVPRW* (2008).

[257] Joel Vidal, Chyi-Yeu Lin, Xavier Lladó, and Robert Martí. "A Method for 6D Pose Estimation of Free-Form Rigid Objects Using Point Pair Features on Range Data". In: *Sensors* (2018).

[258] Daniel Wagner and Dieter Schmalstieg. "ARToolKitPlus for pose tracking on mobile devices". In: *CVWW* (2007).

[259] Eric Wahl, Ulrich Hillenbrand, and Gerd Hirzinger. "Surflet-pair-relation histograms: a statistical 3D-shape representation for rapid classification". In: *3DIM* (2003).

[260] Krzysztof Walas, Tomáš Hodaň, Tae-Kyun Kim, Jiří Matas, Carsten Rother, Frank Michel, Vincent Lepetit, Ales Leonardis, Carsten Steger, Caner Sahin, and Rigas Kouskouridas. *3rd International Workshop on Recovering 6D Object Pose*. `http://cmp.felk.cvut.cz/sixd/workshop_2017/`. 2017.

[261] Krzysztof Walas and Aleš Leonardis. *UoB Highly Occluded Object Challenge*. `https://www.cs.bham.ac.uk/~walask/uob_hooc/`. 2016.

[262] Chen Wang, Danfei Xu, Yuke Zhu, Roberto Martín-Martín, Cewu Lu, Li Fei-Fei, and Silvio Savarese. "DenseFusion: 6D object pose estimation by iterative dense fusion". In: *CVPR* (2019).

[263] Gu Wang, Fabian Manhardt, Jianzhun Shao, Xiangyang Ji, Nassir Navab, and Federico Tombari. "Self6D: Self-Supervised Monocular 6D Object Pose Estimation". In: *ECCV* (2020).

[264] He Wang, Srinath Sridhar, Jingwei Huang, Julien Valentin, Shuran Song, and Leonidas J Guibas. "Normalized object coordinate space for category-level 6D object pose and size estimation". In: *CVPR* (2019).

[265] Paul Wohlhart and Vincent Lepetit. "Learning Descriptors for Object Recognition and 3D Pose Estimation". In: *CVPR* (2015).

[266] Walter Wohlkinger, Aitor Aldoma, Radu B Rusu, and Markus Vincze. "3DNet: Large-scale object class recognition from CAD models". In: *ICRA* (2012). `repo.acin.tuwien.ac.at/tmp/permanent/3d-net.org`.

[267] Erroll Wood, Tadas Baltrušaitis, Louis-Philippe Morency, Peter Robinson, and Andreas Bulling. "Learning an appearance-based gaze estimator from one million synthesised images". In: *Proceedings of the Ninth Biennial ACM Symposium on Eye Tracking Research & Applications* (2016).

[268] Erroll Wood, Tadas Baltrusaitis, Xucong Zhang, Yusuke Sugano, Peter Robinson, and Andreas Bulling. "Rendering of eyes for eye-shape registration and gaze estimation". In: *ICCV* (2015).

[269] Bojian Wu, Yang Zhou, Yiming Qian, Minglun Cong, and Hui Huang. "Full 3D reconstruction of transparent objects". In: *ACM TOG* (2018).

[270] Yu Xiang, Roozbeh Mottaghi, and Silvio Savarese. "Beyond PASCAL: A benchmark for 3D object detection in the wild". In: *Winter Conference on Applications of Computer Vision* (2014).

[271] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox. "PoseCNN: A convolutional neural network for 6D object pose estimation in cluttered scenes". In: *RSS* (2018).

[272] Yu Xiang et al. "ObjectNet3D: A Large Scale Database for 3D Object Recognition". In: *ECCV* (2016). `cvgl.stanford.edu/projects/objectnet3d`.

[273] Ziang Xie, Arjun Singh, Justin Uang, Karthik S Narayan, and Pieter Abbeel. "Multimodal blending for high-accuracy instance recognition". In: *IROS* (2013). `rll.berkeley.edu/2013_IROS_ODP`.

[274] Kuan-Ting Yu, Nima Fazeli, Nikhil Chavan-Dafle, Orion Taylor, Elliott Donlon, Guillermo Diaz Lankenau, and Alberto Rodriguez. "A Summary of Team MIT's Approach to the Amazon Picking Challenge 2015". In: *arXiv preprint arXiv:1604.03639* (2016).

[275] Xenophon Zabulis, Manolis Lourakis, and Panagiotis Koutlemanis. "3D Object Pose Refinement in Range Images". In: *ICVS* (2015).

[276] Sergey Zakharov, Wadim Kehl, and Slobodan Ilic. "DeceptionNet: Network-driven domain randomization". In: *ICCV* (2019).

[277] Sergey Zakharov, Ivan Shugurov, and Slobodan Ilic. "DPOD: 6D Pose Object Detector and Refiner". In: *ICCV* (2019).

[278] Yinda Zhang, Shuran Song, Ersin Yumer, Manolis Savva, Joon-Young Lee, Hailin Jin, and Thomas Funkhouser. "Physically-based rendering for indoor scene understanding using convolutional neural networks". In: *CVPR* (2017).

[279] Zhengyou Zhang. "Iterative point matching for registration of free-form curves and surfaces". In: *IJCV* (1994).

[280] Yu Zhong. "Intrinsic shape signatures: A shape descriptor for 3D object recognition". In: *ICCVW* (2009).

[281] Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li. "On the continuity of rotation representations in neural networks". In: *CVPR* (2019).