TUM School of Computation, Information and Technology
Technische Universität München

6 DoF Pose Estimation of Known and Novel Objects With Dense Correspondences

Ivan Shugurov

Vollständiger Abdruck der von der TUM School of Computation, Information and Technology der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitz:
Prof. Dr. Alin Albu-Schaeffer

Prüfer der Dissertation:
1. Priv.-Doz. Dr. Slobodan Ilic
2. Prof. Dr. Vincent Lepetit

Die Dissertation wurde am 28.09.2022 bei der Technischen Universität München eingereicht und durch die TUM School of Computation, Information and Technology am 25.02.2023 angenommen.

Abstract

Estimating 6 degrees of freedom (6 DoF) pose of the objects from visual data is a long-standing problem in computer vision. The pose is conditioned on a known 3D CAD model of the object of interest and describes its relative position in the camera coordinate system and its orientation. Fast and precise pose estimation is essential for practical application in various domains, such as augmented and virtual reality, robotics, computer-aided medical procedures and autonomous driving, as it provides a more high-level semantic and geometric understanding of the 3D world than the standard object detection or semantic segmentation, allowing for interaction with the 3D world.

Due to its importance, the problem of 6 DoF pose estimation has been addressed numerous times in the past. However, the results have no saturated yet, and the problem still remains challenging. Examples of such challenges are a lack of real labeled data, domain gap between synthetic and real data, handling of symmetric objects, dealing with occlusions and lack of generalization to novel objects. While in the past many methods relied on depth information to infer or refine object pose, modern methods almost completely shifted to operating solely on monocular RGB images. In this way, such methods can be directly used on mobile devices, that are typically equipped with only RGB cameras. Additionally, we pay special attention to the problem of detection and pose estimation of novel objects without training on them, which was typically neglected in the past.

The main contribution of the thesis is development and analysis of 6 DoF pose estimation methods that rely on dense 2D-3D correspondences including their generalazbility to novel objects. The core idea of such methods is to train a neural network to predict correspondences between pixels of the input image and the known 3D CAD model of the object of interest. This allows for 6 DoF pose estimation with the well-known Perspective-n-Point (PnP) and RANSAC algorithms. The thesis consists of 4 main parts. The first part introduces a novel method for 6 DoF pose estimation with dense correspondences supplemented with a rendering-based object pose refiner. This efficient light-weight method allows for very precise pose estimation in real time even on mobile devices. It sets the baseline for all other methods presented in the thesis. In the second part, we explore how to use dense correspondences for object pose refinement from several frames with known relative camera transformations. The method utilizes the advances in differentiable rendering and allows for effective handling of occlusions and erroneous pose hypothesis. The explicit loss function, which is minimized using non-linear optimization, provides an advantage over the refiner from the first part. The third part focuses on in-depth analysis of correspondence estimation with deep learning. The study empirically compares the choice of input data (monocular RGB versus depth maps) and choice of training data (real versus synthetic). The thesis concludes with discussion of a one-shot method for detection and pose estimation of novel object without prior training on them. To the best of our knowledge, it is the first deep-learning-based method operating on monocular RGB images that fully generalizes to novel objects. In spite of not having been trained on target objects, it still achieves good pose quality.

Zusammenfassung

Das Abschätzen der 6 Freiheitsgrade (6 DoF) Pose der Objekte aus visuellen Daten ist ein seit langem bestehendes Problem in der Computer Vision. Die Pose basiert auf einem bekannten 3D-CAD-Modell des interessierenden Objekts und beschreibt seine relative Position im Kamerakoordinatensystem und seine Orientierung. Eine schnelle und präzise Posenschätzung ist für die praktische Anwendung in verschiedenen Bereichen wie Augmented und Virtual Reality, Robotik, computergestützten medizinischen Verfahren und autonomem Fahren unerlässlich, da sie ein besseres semantisches und geometrisches Verständnis der 3D-Welt bietet als die Standard-Objekterkennung oder semantische Segmentierung, die eine Interaktion mit der 3D-Welt ermöglichen.

Aufgrund seiner Bedeutung wurde das Problem der 6-DoF-Pose-Schätzung in der Vergangenheit mehrfach angesprochen. Die Ergebnisse sind jedoch noch nicht gesättigt, und das Problem bleibt immer noch eine Herausforderung. Beispiele für solche Herausforderungen sind ein Mangel an echten gekennzeichneten Daten, eine Domänenlücke zwischen synthetischen und echten Daten, der Umgang mit symmetrischen Objekten, der Umgang mit Okklusionen und die fehlende Verallgemeinerung auf neuartige Objekte. Während sich in der Vergangenheit viele Methoden auf Tiefeninformationen stützten, um die Objekthaltung abzuleiten oder zu verfeinern, haben sich moderne Methoden fast vollständig darauf verlagert, nur mit monokularen RGB-Bildern zu arbeiten. Auf diese Weise können solche Verfahren direkt auf Mobilgeräten verwendet werden, die typischerweise nur mit RGB-Kameras ausgestattet sind. Darüber hinaus widmen wir dem Problem der Erkennung und Posenschätzung neuartiger Objekte ohne Training besondere Aufmerksamkeit, was in der Vergangenheit typischerweise vernachlässigt wurde.

Der Hauptbeitrag der Dissertation ist die Entwicklung und Analyse von 6-DoF-Pose-Schätzmethoden, die auf dichten 2D-3D-Korrespondenzen beruhen, einschließlich ihrer Verallgemeinerbarkeit auf neuartige Objekte. Die Kernidee solcher Verfahren besteht darin, ein neuronales Netzwerk zu trainieren, Übereinstimmungen zwischen Pixeln des Eingangsbildes und dem bekannten 3D-CAD-Modell des interessierenden Objekts vorherzusagen. Dies ermöglicht eine 6-DoF-Posenschätzung mit den bekannten Perspective-n-Point (PnP)- und RANSAC-Algorithmen. Die Diplomarbeit besteht aus 4 Hauptteilen. Der erste Teil stellt eine neuartige Methode zur 6-DoF-Posenschätzung mit dichten Korrespondenzen vor, ergänzt durch einen Rendering-basierten Objektpose-Refiner. Diese effiziente und leichtgewichtige Methode ermöglicht eine sehr genaue Posenschätzung in Echtzeit auch auf mobilen Geräten. Es bildet die Grundlage für alle anderen Methoden, die in der Arbeit vorgestellt werden. Im zweiten Teil untersuchen wir, wie dichte Korrespondenzen zur Verfeinerung der Objekthaltung aus mehreren Frames mit bekannten relativen Kameratransformationen verwendet werden können. Das Verfahren nutzt die Fortschritte beim differenzierbaren Rendering und ermöglicht eine effektive Behandlung von Okklusionen und fehlerhaften Posenhypothesen. Die explizite Verlustfunktion, die durch nichtlineare Optimierung minimiert wird, bietet einen Vorteil gegenüber dem Refiner aus dem ersten Teil. Der dritte Teil konzentriert sich auf die eingehende Analyse der Korrespondenzschätzung mit Deep Learning. Die Studie vergleicht empirisch die Wahl der Eingabedaten (monokulares RGB versus Tiefenkarten) und die Wahl der Trainingsdaten (real versus synthetisch). Die Arbeit schließt mit der Diskussion einer One-Shot-Methode zur Erkennung und Posenschätzung neuartiger Objekte ohne vorheriges Training. Nach unserem besten Wissen ist es die erste Deep-Learning-basierte Methode, die mit monokularen RGB-Bildern arbeitet und sich vollständig auf neuartige Objekte verallgemeinern lässt. Obwohl es nicht auf Zielobjekte trainiert wurde, erreicht es immer noch eine gute Posenqualität.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Motivation and Problem Definition

Artificial Intelligence researchers from all over the world have been trying to teach computers how to perceive and behave like humans for decades. The field of computer vision plays a crucial role in these attempts due to a sheer amount of information about the world and its state humans perceive through vision. The initial attempts of computer vision were mostly limited to non-semantic understanding of the images. Examples of such tasks include image filtering [17, 18, 19, 20], corner detection [21, 22, 23, 24], contour and edge detection [25, 26, 27, 28], stereo matching [29, 30, 31, 32, 33, 34], image stitching [35, 36, 37, 38], structure from motion and SLAM [39, 40, 41, 42, 43], 3D reconstruction [44, 45] and many more. Even though these tasks are crucial for the understanding of the 3D world around us, they lack the the knowledge of semantics. While humans can instantly recognize all objects that they see, asses their size, asses their location and, when possible and necessary, grab them, it is an extremely challenging tasks for computers. The rise of machine learning and, later, deep learning gave us hope to find solutions to many of the aforementioned problems including semantics. The increase in computational powers of computers, hand-held and wearable devices allowed for deployment of computer vision systems based on machine learning for everyday applications. Thanks to advances in deep learning, we are now able to extract a lot of semantic information even from monocular RGB images. Examples of the tasks, revolutionized by deep learning, are image classification [46, 47, 48, 49, 50, 51, 52], object detection [53, 54, 55, 56, 57, 13, 58], semantic segmentation [59, 60, 61, 62, 63], instance segmentation [64] and panoptic segmentation [65].

The aforementioned subtasks of computer vision have been tackled in a wide range of works before. However, they are focused either on geometric or semantic scene image understanding, which is often insufficient for real-life applications requiring interaction with the 3D world. We know what objects we see but we still have no idea how big they are, how far they are, what their relative pose to the camera. Thus, for example, a robot cannot grasp the objects that it sees, an autonomous car cannot avoid an obstacle, an AR device cannot overlay any useful information atop of the real world, a surgical robot is severely restricted in how it can assist the surgeon. This motivates the need for a better joint geometry- and semantics-aware understanding of the world from a single RGB image. The solution is fortunately available and is called 6 Degrees of Freedom (6 DoF) object pose estimation.

The task of 6 DoF pose estimation consists of estimating a position and orientation of the object relative to the camera. Normally, a known 3D model of the object is assumed to be known. Then, object pose estimation is formally posed as estimating rotation $\mathbf{R} \in SO(3)$ and translation $\mathbf{t} \in \mathbb{R}^3$ which transform the object from the object coordinate system to the coordinate system of the camera to form the input image. The simplified process of applying a transformation to the object and of the image formation is provided in Figure 1.1. Estimating objects' poses brings much more semantic and geometric understanding of the scene as compared to the tasks described above. Now a computer is not only aware of what it sees but also knows where exactly the objects are in the 3D space relative to the camera. This is crucial input information of various tasks, such as robotic grasping, augmented reality, autonomous driving and medical robotics. Additionally, knowing the 3D location of the object implicitly gives us its 2D location and 2D segmentation mask of the object.

Thanks to its importance, 6 DoF object pose estimation belongs to the classical computer vision problems. Initially, it was solved by detecting hand-crafted keypoints typical for the object, encoding them with hand-crafted features and matching them to a database of pre-computed keypoints of the object. This allowed for establishing 2D-3D correspondences and estimating the pose using the PnP algorithm [1]. This approach, as was later shown in Linemod [66], did not scale well to objects with uniform textures as it is impossible to extract reliable keypoitns from them. Linemod [66] proposed to use multimodal templates for pose estimation. Another line of research explored pose estimation by matching simple descriptors of local geometry, such as Point Pair Features [15], of the scene to those of the object. This allowed for precise pose estimation but required to have access to depth information and heavily relied on pose hypothesis refinement with the ICP algorithm [67], resulting in slow runtimes.

The advent of deep learning has also transformed the field of pose estimation. Now, it is possible to extract features no matter how pronounced the textures of the objects are. Additionally, it is possible to train a neural network to detect any kind of predefined keypoints, for example corners of the projected 3D bounding box as in BB8 [68] and YOLO6D [69]. Starting with the simpler task of rotation estimation via manifold matching [70, 71], the filed has evolved to end-to-end deep learning solutions [10, 72] that beat the classical approaches, such as PPF, using only monocular RGB images as input. These faster and more precise methods allowed for a wide adoption of pose estimation methods.

1.2 Dense Correspondences

Pose estimation with dense correspondences is one of the leading paradigms in 6 DoF object pose estimation, which was explored in numerous works [73, 74, 70, 75, 76]. First, let us define what is meant by 2D-3D correspondences. Figure 1.1 provides a schematic overview of the image formation with a pinhole camera model [4]. More details on the pinhole camera model and projective geometry are provided in Section 2.2.3 and a seminal book on multiple-view geometry by Hartley et.al. [4]. Given a point $X \in \mathbb{R}^3$ on the object surface in the object coordinate system, it is transformed to the camera coordinate system as $\mathbf{R}X + \mathbf{t}$. Using the projection operator π of the pinhole camera model, the 3D point $\mathbf{R}X + \mathbf{t}$ is projected onto the image plane as a 2D point with

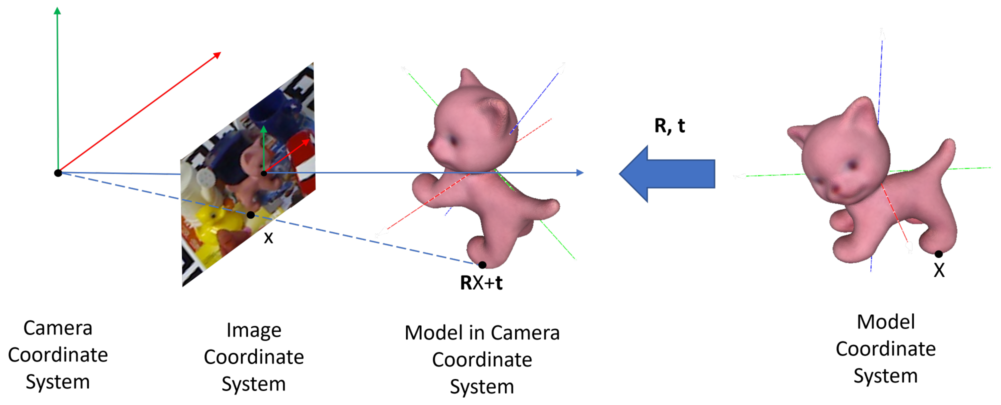


Figure 1.1: Illustration of a rigid body transformation $[\mathbf{R}|\mathbf{t}]$ and the pinhole camera model. 2D point x corresponds to the location of the 3D point X on the object surface in the object coordinate system. Such 2D-3D correspondences can be used to estimate $[\mathbf{R}|\mathbf{t}]$ using the PnP algorithm [1].

coordinates $x \in \mathbb{R}^2$. Projective transformation loses information about distance from the point. In other words, given a point x, it is impossible to directly recover $\mathbf{R}X + \mathbf{t}$. However, 2D coordinates of x is sufficient to compute a ray starting in the origin of the camera and passing through $\mathbf{R}X + \mathbf{t}$ in a closed form. Given a sufficient number of correspondences between $x_1, ...x_n \in \mathbb{R}^2$ and their 3D correspondences $X_1, ...X_n \in \mathbb{R}^3$ on the object in the object coordinate system, it is possible to estimate the transformation $\mathbf{R}, \mathbf{t}$ from the object coordinate system to the camera coordinate system using a well-known Perspective-n-Point (PnP) algorithm [1].

While having been used for pose estimation with hand-crafted descriptors in the 90s, correspondence-based pose estimation experienced a renaissance. Deep learning allowed for flexible extraction of features from RGB and prediction of arbitrary defined keypoints, rather than only the hand-crafted corners such as SIFT [77]. It even allowed for feature extraction and correspondence estimation from low-textured objects, such as models from the Linemod dataset [5], and texturelss industrial objects, as in the TLESS dataset [12]. The previous works [69, 68] proposed to predict 2D locations of a sparse set of pre-defined keypoints, such as corners of the 3d bounding box around the object. Dense correspondences is the natural extension of these ideas. The key concept is to establish 2D-3D correspondences for *all* object pixels in the image instead of only a small fraction of them. Hence, the name ”dense correspondences”. Having more correspondences contributes to increased robustness and accuracy of pose estimation since imprecise correspondences can be filtered out using the RANSAC algorithm [78].

Advantages of pose estimation with dense correspondences can be summarized as follows.

Interpretable geometric constraints. One of the main advantages of keypoint-based methods is the naturally added geometric interprentibility of the results. The underlying neural model is not responsible for directly predicting the object pose but rather responsible for an arguably simpler task of correspondence estimation, which is done directly in the image plain. The PnP algorithm, which is used to estimate the pose from correspondences, minimizes the reprojection loss between the predicted and projected locations of the correspondences. Given correspondences of reasonable quality, this ensures prediction of plausible poses.

Superior performance over other methods. Dense-correspondence-based methods tend to outperform other methods, including the sparse-correspondence-based methods. This is directly related to the number of correspondences. The larger the set of predicted correspondences are, the smaller the influence of imprecision of each of the correspondence. Low-quality correspondences, that do not fit the projection model described above, are filtered out using the RANSAC algorithm.

Additional robustness to occlusions. Dense-correspondence-based methods are less sensitive to occlusions than their sparse counterpart. When a part of the object is visible, it is still possible to predict accurate correspondences for visible pixels, while the invisible pixels can be completely ignored.

Ease of data augmentation during training. Given an image of the object and its known 2D-3D correspondences, various 2D data augmentations, such as in-plane rotations, scaling, in-painting in different backgrounds, adding extra occlusions, can be easily performed directly in 2D together with straightforward recalculation of the ground truth correspondences.

In spite of all the cons listed above, dense-correspondence-based methods are not free of disadvantages. Disadvantages of pose estimation with dense correspondences can be summarized as follows.

Need for rendering during data preparation. Ground truth poses alone are not enough for training the dense correspondence detector. Each RGB image of the object must be supplemented with a per-pixel segmentation mask and per-pixel information about corresponding 3D points. It is typically achieved by rendering the object with the texture replaced by vertex coordinates. This process introduces an additional, potentially time and computationally consuming, step during data preparation. It is, however, of a less concern if synthetic data is used, since scene simulation and photo-realistic rendering takes considerably more time than rendering of correspondences. During rendering, a particular attention must be paid to the consistency of rendered coordinates for **symmetric objects**.

No end-to-end differentiability. During training of a deep-learning-based model, the loss function is defined over the correspondences rather than over the pose itself because PnP and RANSAC algorithms are typically non-differentiable. During testing, PnP and RANSAC add an extra step to the inference pipeline making it **slower**. It is particularly true for larger number of correspondences or for noisier correspondences, which require more iterations of RANSAC to converge to good pose estimates.

Last but not least, the underlying dense correspondence models are based on architectures for semantic segmentation. This leads to a **larger number of trainable parameters.**

1.3 Pose Estimation of Novel Objects

Before learning-based feature extraction, generalization to novel objects was not an issue. Methods, based on hand-crafted keypoints, generalized naturally to novel objects. Hand-crafted keypoint detectors, such as SIFT [77] or ORB [24], are by construction local and unaware of the concept of the object. Even until today, hand-crafted keypoint detectors still prevail over the learning-based keypoints in the field of SLAM [79]. For each new object, a typical keypoint-based method required pre-computation of object's keypoint and their descriptors from several sample images, which is a deterministic process and is very different from training deep learning models. Alternatively, PPF-based methods rely on extremely simple descriptors based on distances and angles between two points and their normals. In spite of their simplicity, the descriptor is very generic, rotationally-aware and allows to generate a pose hypothesis from a single point pair. However, to achieve reasonable pose accuracy, PPF methods require a larger number of pose hypotheses and ICP refinement.

In spite of the progress of deep-learning-driven pose estimation, the existing solutions are far from perfect. Striving for high pose quality metrics, researchers typically train one deep neural network for each object. This results in a huge overhead both during the data preparation and training as well as during testing, when separate networks have to be executed for each detected objects. This is infeasible for many real-life applications such as warehouse robotics and autonomous driving, where the robot/vehicle is constantly surrounded to a huge number of various objects. On the one hand, it was addressed by category-level pose estimation pioneered by the NOCS paper [74]. The goal is to train a single network capable of pose estimation of all objects belonging to one category. For example, one network for cups, one network for chairs and so on. The effectiveness of such methods, however, directly correlates to the degree of similarity between objects within the category. The other research direction is dedicated to designing methods which detect novel objects conditioned on the 3D object model. While the classical methods, such as PPF, easily scaled to new objects without training on them, this turned out to be a surprisingly difficult task for deep learning. In the past, Pitteri et al. tried to solve it by training a detection model to detect corners of the objects [80] or to predict per-pixel shape descriptors [81]. This required, however, train and test set to contain similar looking objects. MultiPath AAE [82] proved that rotation estimation via manifold learning can generalize well to novel objects. However, the method still relied on a standard object detector trained on tha target objects. We explicitly address this issue in Chapter 7 by introducing a novel deep learning architecture capable of end-to-end detection and pose estimation of novel unseen objects. Our proposed method consists of three stages: one-shot semantic segmentation, initial viewpoint estimation via template matching and, finally, dense correspondence estimation between the matched template and the image of the object.

1.4 Challenges

6 DoF pose estimation is intrinsically a challenging and ill-defined tasks. Here, we briefly summarize the main open challenges. More detailed discussion of the topic is provided in Section 3.1.

Lack of real training data and its lack of quality. One of the most fundamental limitations for learning-based pose estimation methods stems from the difficulty of annotating real training data with pose labels. While for the other classical deep learning tasks, such as classification or semantic segmentation, it is possible to manually label images, it is simply impossible for 6 DoF pose labels. Labeling images with object poses is an intricate and tedious process, as was shown in [6, 12]. The quality of estimated ground truth labels is still far from perfect, which is clearly visible in datasets such as Linemod [5] and YCB-V [16]. To tackle this problem, we experiment with training all methods presented in the thesis on synthetic data and test their performance on real data.

Projective ambiguity. Pose estimation directly from monocular RGB images is arguably the most important issue because of the ubiquitous possible applications. Judging by the methods present in the BOP challenge [7], most of the modern deep-learning-based methods focus exclusively on this scenario for initial pose estimation. Depth or multi-view are used only for post-refienement. However, monocular pose estimation is innately ill-defined, because projective transformation and discrete nature of the images make it hard to accurately estimate distance to the object along the z axis of the camera. To deal with this problem, we first introduce a multi-view object pose refiner in Chapter 5. We then experiment with correspondence and pose prediction directly from depth maps, which are free of projective ambiguity, in Chapter 6.

Occlusions. Occlusions pose a significant challenge for pose estimation methods because they not only make detecting such object harder but also complicates correspondence estimation because only a part of the object is visible. This is implicitly handled in all methods proposed in this thesis by predicting dense correspondences.

Low textures. Even though deep learning allows for feature extraction from a picture of any surface, objects without pronounced textures still offer a difficult challenge for detection and correspondence and pose estimation. Examples of such objects are not only industrial details, such as object in TLESS [12] and Homebrewed [6] datasets, but also uniformly colored object from other datasets, such as Linemod [5]. This is partially addressed in Chapter 6 by the correspondence estimation method operating directly on depth maps instead of RGB.

Symmetries. Symmetric object make the pose estimation task less defined as several, or infinitely many poses, may look indistinguishably in the image. Symmetric object typically require additional care during ground truth preparation and in loss function definition. In all methods presented in this thesis, the symmetries are handled by rendering consistent and non-ambiguous 2D-3D correspondence maps before training.

Missing depth information. Depth information is typically used for pose refinement with the ICP algorithm [67]. Effectiveness of such refinement is typically hindered by noisy or missing depth information caused by occlusions and by object materials.

This is especially noticeable on the BOP Challenge leaderboard[7] of the TLESS [12] dataset, where multiple methods report worse results after the ICP refinement. Superior performance of RGB method from Chapter 6 allows it to be used without any additional refinement for various practical tasks. Moreover, the multi-view refiner from Chapter 5 is free of this issue.

Need for real-time performance. For pose estimation to be useful, it has to be real-time capable and very precise at the same time. This is, unfortunately, extremely challenging to achieve. While there are direct pose regression methods that are extremely fast, they typically lack the precision. Correspondence-based methods offer great pose quality but tend to be much slower due to overhead caused by PnP and RANSAC methods. The problem is exacerbated by the presence of many objects in the scene, each of which is typically processed independently. On the publicly available leaderboard of BOP Challenge [7], no method shows real-time performance on datasets such as Homebrewed [6], TLESS [12] and YCB-V [16]. The method introduced in Chapter 4 is real-time capable for single object pose estimation scenario and can even be run in real time on iPad.

Separate training for each object. A typical deep-learning-based model for pose estimation is trained only on one model at a time. This means, that for each object data preparation (either labeling of real data or rendering synthetic data) and model training have to be repeated from scratch. Moreover, current state-of-the-art models lack any capabilities to **generalize to novel objects**. This problem is explicitly tackled in one Chapter 7 of the thesis.

1.5 Contributions

In this thesis, we focus on developing novel state-of-the-arts methods for 6 DoF pose estimation using dense correspondences. Four main publications, presented in here, cover various aspects of the problem. The main contributions can be summarized as follows:

- **Precise and fast 6 DoF pose estimation method based on dense correspondences**. Object pose estimation from pure RGB images, as opposed to classical methods operating on depth or RGBD, have gained in popularity due to the availability on high-quality RGB cameras on almost all every-day devices. The progress in monocular pose estimation was made possible by the progress in deep learning, which allowed for reliable feature extraction even from low-textured objects. Building on these advances, we propose a method, nicknamed DPOD, that works by predicting multi-class object masks and dense 2D-3D correspondences between image pixels and corresponding 3D models. Dense correspondences computed by our method allow for more robust and accurate 6 DoF pose estimation. We demonstrated that for both, real and synthetic training data, our detector outperforms state-of-the-art methods of the time. While being precise, the presented approach is still real-time capable even on mobile devices. The proposed pose refinement approach also performs very well and allows for achieving a pose accuracy that surpasses all other related deep-learning-based pose refinement approaches.

- **Multi-view object pose refinement method based on differentiable rendering**. Pose estimation from monocular RGB images is innately imprecise due to projective ambiguity, which does not allow for precise distance estimation, and occlusions, which prevent the network from establishing accurate 2D-3D correspondences. We propose a method that uses an initial pose estimate and aligns it to 2D-3D correspondences estimated in several frames, with known relative camera transformations. The process is fully differentiable and is implemented using a differentiable renderer. The method is shown to work well even in case of sever occlusions, erroneous initial pose estimates and small distance between cameras. We additionally demonstrate that this method can be used for auto-labeling real data. A DPOD-variant trained on images with automatically labeled poses reaches similar performance to the DPOD trained on the same images with ground truth labels.

- **In-depth analysis of performance of dense-correspondences-based 6 DoF pose estimation**. We propose a three-stage method for object detection and pose estimation, nicknamed DPODv2,that relies on dense correspondences. We combine a 2D object detector with a dense correspondence estimation network and a multi-view pose refinement method to estimate a full 6 DoF pose. We propose a unified framework for dense correspondence estimation, whose architecture is agnostic to the input modality type available (RGB or Depth). Ensuring that the architecture is fixed, allows us to run careful ablation studies on the effects of particular design choices: using RGB or depth as input, and using real, synthetic or a mix of training data. We measure and report how these choices affect the quality of correspondences and the overall quality of pose estimates. We perform an extensive evaluation to analyze and validate the results on several publicly available challenging datasets. DPODv2 achieves excellent results on all of them while still remaining fast and scalable independent of the used data modality and the type of training data.

- **First end-to-end one-shot method for detection of novel objects without prior training on them**. One of the major bottlenecks for industrial adoption of deep-learning-based pose estimation methods is their lack of generalization abilities and time needed for data preparation and network training. A typical pose estimation neural network is trained only on one object instance at a time. It means that a new neural network has to be trained for each object, which is not which is not scalable or even feasible. To remedy this issue, we present a novel one-shot method for object detection and 6 DoF pose estimation, that does not require training on target objects. The only assumption is that textured CAD models of the target objects are available. The network takes an input object and a 3D model of the object of interest and automatically detects the object in 3D and then estimates its pose. Our proposed method consists of three stages: one-shot semantic segmentation, initial viewpoint estimation via template matching and, finally, one-shot dense correspondence estimation between the matched template and the image of the object. Then, the pose is estimated using either PnP or Kabsch algorithms. To the best of our knowledge, this is the first end-to-end one-shot method for monocular RGB methods. We evaluate the method on five different datasets and report very competitive performance in comparison to the state-of-the-art methods trained on synthetic data, even though our method is not trained on the object models used for testing.

1.6 Outline

This section summarizes the overall structure of the thesis. All methods proposed in the thesis have undergone thorough peer-reviews by experts in the field and published at the top-tier journals and conferences.

Chapter 2 - Theory and Fundamentals In this chapter, we briefly discuss the generic theoretical background, this thesis is built upon on. First, the core principles of artificial neural networks and deep learning as well as their training strategies are discussed. Then, we discuss the formal mathematical definition of 6 DoF object pose and how this pose can be practically parameterized.

Chapter 3 - Pose Estimation This chapter introduces core concepts of pose estimation in general and pose estimation with dense correspondences in particular. We first discuss the typical challenges of pose estimation. Then, we provide a discussion of related work that formed state of the art before this thesis. Then, geometric methods for pose estimation with correspondences are discussed. At the end, we introduce the main publicly available datasets, used for benchmarking pose estimation methods, and the most commonly pose quality metrics.

Chapter 4 - Pose Estimation with Dense Correspondences This chapter introduces the core idea of establishing dense 2D-3D correspondences with deep learning. The method proposed in this chapter is fast and yet establishes precise correspondences, which leads to very precise pose estimates. The experimental shows that dense correspondences are also highly useful for pose estimation of symmetric and occluded objects. The chapter additionally introduces a deep-learning-based pose refinement procedure, that operates exclusively on monocular RGB images and allows us to improve poses even further. Both networks can be trained on either real or synthetic data.

The findings of this chapter were published in the following peer-reviewed paper:

- S. Zakharov*, **I. Shugurov***, S. Ilic. *Dpod: 6d pose object detector and refiner.* IEEE International Conference on Computer Vision (ICCV), 2019 (* equal contribution)

The method was additionally registered as the following US patent:

- **I. Shugurov**, A. Hutter, S. Zakharov, S. Ilic. *Dense 6-dof pose object detector.* U.S. Patent Application No. 17/427,231. 2022

Chapter 5 - Multi-View Pose Refinement with Dense Correspondences In this chapter, a novel method for object pose refinement from multiple RGB frames with known relative camera transformations. An explicit loss function is used to find a pose that is consistent with correspondences in all frames. The loss is computed and aggregated between views using a differentiable renderer and, thus, can be optimized using the standard gradient-based techniques. Additionally, it is demonstrated how the method can be used for auto-labeling of unlabled real data.

The findings of this chapter were published in the following peer-reviewed paper:

- **I. Shugurov***, I. Pavlov*, S. Zakharov, S. Ilic. *Multi-view object pose refinement with differentiable renderer.* IEEE Robotics and Automation Letters (RA-L), 2021 (* equal contribution)

Chapter 6 - Analysis of Pose Estimation with Dense Correspondences The method proposed in this chapter extends the ideas of correspondence-based 6 DoF pose estimation and provides its in-depth analysis. The study empirically compares the choice of input data (monocular RGB versus depth maps) and choice of training data (real versus synthetic) and evaluates how these choices affect the quality of estimated correspondences and poses.

The findings of this chapter were published in the following peer-reviewed journal paper:

- **I. Shugurov**, S. Zakharov, S. Ilic. *Dpodv2: Dense correspondence-based 6 dof pose estimation.* IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), 2021

Chapter 7 - One-Shot Pose Estimation Without Re-Training In this chapter, a novel method for one-shot object detection and pose estimation is proposed. The method takes an input image and a descriptor of a known 3D object and automatically estimates its 2D location and 6 DoF pose. Taking two separate object as input allows the method to automatically generalize to novel objects without training on them. The method works by first segmenting the target object in the input image. Then, the initial viewpoint estimate is generated by template matching. Finally, dense correspondences are established between the matched template, with known object pose, and the input image with the unknown object pose. This automatically gives us dense 2D-3D correspondences between the input image and the object model. This allows for a significant reduction of time needed for data preparation and training of the model as it has to be done only once. In spite of not having been trained on the target objects, the method still regresses poses, whose quality is comparable to the standard methods trained on synthetic data.

The findings of this chapter were published in the following peer-reviewed paper:

- **I. Shugurov**, F. Li, B. Busam, S. Ilic. *OSOP: A Multi-Stage One Shot Object Pose Estimation Framework.* IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2022

Chapter 8 - Conclusion & Outlook. This chapter concludes the thesis, summarizes the results, proposed methods, and provides a brief discussion of possible future work.

2 Theory and Fundamentals

2.1 Neural Networks

Artificial neural networks have been around since the since the middle of the 20th century, when seminal works on perceptron were published by Mcculloch et all. [83] and Rosenblatt et all [84]. The perceptron model depicted in Figure 2.1a was inspired by a biological neuron depicted in Figure 2.1b. Now, in the era of deep learning, perceptron is usually referred to as a neuron. It forms a basic building block of neural networks. In this chapter, we provide a brief overview of artificial neural networks and their core components.

2.1.1 Fully-Connected Neural Networks

The most basic class of artificial neural networks is fully-connected networks. Each neuron of a fully-connected neural network, depicted in Figure 2.1b, implements the following function:

$$x_{i+1}^{j} = \phi\left(x_i^T \mathbf{w_i}^j + b_i^j\right) \tag{2.1}$$

where $x_i \in \mathbb{R}^{D_i}$ stand for an input vector to the neuron, $\mathbf{w}_i^j \in \mathbb{R}^{D_i}$ for the weights of the neuron and $b_i^j \in \mathbb{R}$ for a so-called bias and ϕ denotes a non-linear activation function. Indices i and j stand for layer and neuron indices respectively.

To form a fully-fledged fully-connected neural network, several neurons are stacked together to for a layer, as shown in Figure 2.2. Therefore, it is more convenient to think of the fully-connected neural network in terms of layers rather than particular neurons:

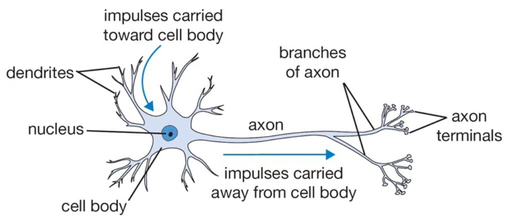
$$x_{i+1} = \phi\left(\mathbf{W}_i \cdot x_i + \mathbf{b}_i\right) \tag{2.2}$$

with $x_i \in \mathbb{R}^{D_i}$, $x_{i+1} \in \mathbb{R}^{D_{i+1}}$ are input and output feature vectors respectively, $\mathbf{W}_i \in \mathbb{R}^{D_{i+1} \times D_i}$ is matrix of weights, consisting of individual neuron weights stacked as rows, and $\mathbf{b}_i \in \mathbb{R}^{D_{i+1}}$ is a vector of biases.

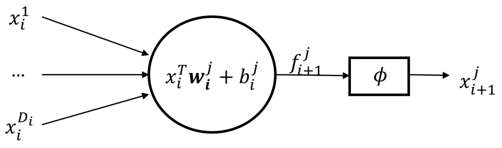
For the clarity of notation, let us use the following shorthand notation:

$$f_{i+1} \coloneqq \mathbf{W}_i \cdot x_i + \mathbf{b}_i \tag{2.3}$$

In general case, the neural network of arbitrary depth can be trained using gradient-based optimizers such as a gradien descent. This requires computation of gradients of the loss function with respect to the nerwork parameters. This is done using the so-called backpropagation [85]. The idea stems from the basic calculus rule of chain derivatives.



(a) Biological neuron



(b) Artificial Neuron

Figure 2.1: Comparison of biological and artificial neurons. Figure from [2].

Given the output of the neural network as x_{n+1}, the derivatives of a differentiable loss function $\mathcal{L}$ to the weights of the last layer $\mathbf{W}_n$ can simply be computed as following:

$$\begin{aligned}\frac{\partial \mathcal{L}\left(x_{n+1}, y_{gt}\right)}{\partial \mathbf{W}_n} &= \frac{\partial \mathcal{L}\left(x_{n+1}, y_{gt}\right)}{\partial x_{n+1}} \cdot \frac{\partial x_{n+1}}{\partial \mathbf{W}_n}. \\ &= \frac{\partial \mathcal{L}\left(x_{n+1}, y_{gt}\right)}{\partial x_{n+1}} \cdot \frac{\partial x_{n+1}}{\partial f_{n+1}} \cdot \frac{\partial f_{n+1}}{\partial \mathbf{W}_n} \\ &= \frac{\partial \mathcal{L}\left(x_{n+1}, y_{gt}\right)}{\partial x_{n+1}} \cdot \frac{\partial x_{n+1}}{\partial f_{n+1}} \cdot \frac{\partial \mathbf{W}_n \cdot x_n + \mathbf{b}_n}{\partial \mathbf{W}_n} \\ &= \frac{\partial \mathcal{L}\left(x_{n+1}, y_{gt}\right)}{\partial x_{n+1}} \cdot \frac{\partial x_{n+1}}{\partial f_{n+1}} \cdot x_n^T\end{aligned} \tag{2.4}$$

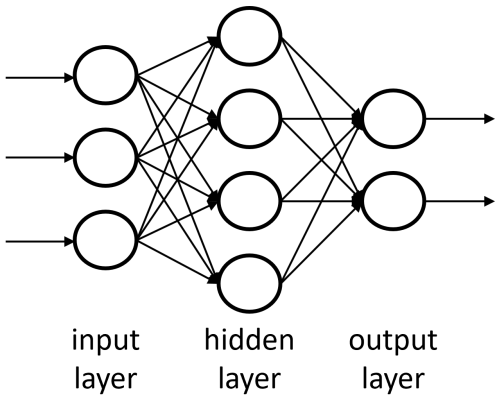
Gradients of the loss function with respect to the weights of the penultimate layer $\mathbf{W}_{n-1}$ can be computer in s similar fashion:

$$\begin{aligned}\frac{\partial \mathcal{L}\left(x_{n+1}, y_{gt}\right)}{\partial \mathbf{W}_{n-1}} &= \frac{\partial \mathcal{L}\left(x_{n+1}, y_{gt}\right)}{\partial x_{n+1}} \cdot \frac{\partial x_{n+1}}{\partial x_n} \cdot \frac{\partial x_n}{\partial \mathbf{W}_{n-1}} \\ &= \frac{\partial \mathcal{L}\left(x_{n+1}, y_{gt}\right)}{\partial x_{n+1}} \cdot \frac{\partial x_{n+1}}{\partial f_{n+1}} \cdot \frac{\partial f_{n+1}}{\partial x_n} \cdot \frac{\partial x_n}{\partial f_n} \cdot \frac{\partial f_n}{\partial \mathbf{W}_{n-1}} \\ &= \frac{\partial \mathcal{L}\left(x_{n+1}, y_{gt}\right)}{\partial x_{n+1}} \cdot \frac{\partial x_{n+1}}{\partial f_{n+1}} \cdot \frac{\partial f_{n+1}}{\partial x_n} \cdot \frac{\partial x_n}{\partial f_n} \cdot x_{n-1}^T\end{aligned} \tag{2.5}$$

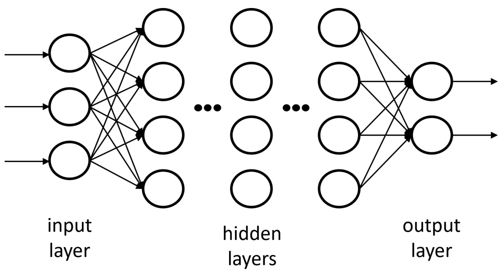
It is clear from the Equation 2.4, that $\frac{\partial \mathcal{L}(x_{n+1}, y_{gt})}{\partial x_{n+1}} \cdot \frac{\partial x_{n+1}}{\partial f_{n+1}}$ was already computed at the previous step. It means that only $\frac{\partial f_{n+1}}{\partial x_n} \cdot \frac{\partial x_n}{\partial f_n} \cdot x_{n-1}^T$ has to be computed from scratch.

A general equation for a derivative with respect to parameters of an arbitrary layer goes as follow:

$$\begin{aligned}\frac{\partial \mathcal{L}\left(x_{n+1}, y_{gt}\right)}{\partial \mathbf{W}_i} &= \frac{\partial \mathcal{L}\left(x_{n+1}, y_{gt}\right)}{\partial x_{i+1}} \cdot \frac{\partial x_{i+1}}{\partial \mathbf{W}_i} \\ &= \frac{\partial \mathcal{L}\left(x_{n+1}, y_{gt}\right)}{\partial x_{i+1}} \cdot \frac{\partial x_{i+1}}{\partial f_{i+1}} \cdot \frac{\partial f_{i+1}}{\partial \mathbf{W}_i} \\ &= \frac{\partial \mathcal{L}\left(x_{n+1}, y_{gt}\right)}{\partial x_{n+1}} \cdot \ldots \cdot \frac{\partial x_{i+2}}{\partial x_{i+1}} \cdot \frac{\partial x_{i+1}}{\partial f_{i+1}} \cdot \frac{\partial f_{i+1}}{\partial \mathbf{W}_i}\end{aligned} \tag{2.6}$$



(a) Shallow Fully-Connected Network



(b) Deep Fully-Connected Network

Figure 2.2: Fully connected neural networks.

For this general case, again $\frac{\partial \mathcal{L}(x_{n+1}, y_{gt})}{\partial x_{n+1}} \cdot ... \cdot \frac{\partial x_{i+2}}{\partial x_{i+1}} \cdot \frac{\partial x_{i+1}}{\partial f_{i+1}}$ was already computed at the previous step, and only the remaining $\frac{\partial f_{i+1}}{\partial \mathbf{W}_i}$ has to be evaluated. It means that the gradients are computer layer-wise from the last layer to the first. At step, only gradients of the next layer are re-used without recomputing them.

2.1.2 Convolutional Neural Networks

When it comes to the tasks of computer vision, the field is dominated by convolutional neural networks rather than by fully connected networks. First popularized by the LeNet paper [86] and then proven to work exceptionally well by the AlexNet paper [46], convolutions networks revolutionized the design of neural architectures for computer vision. The standard fully-connected networks are a bad fit for image processing due to the large dimensionality of the input data. For example, given an image of the standard resolution 480×640, each neuron in the first hidden layer will have to have 307200 weights excluding bias, which quickly becomes intractable. Instead, each neuron in a convolutional neural network is defined as a kernel $\mathbf{k} \in \mathbb{R}^{H_k \times W_k \times D_k}$ with H_k, W_k and D_k being height, width and the dimensionality of the kernel respectively and a bias $b \in \mathbb{R}$. Then, this kernel is applied to the input image using the standard convolutional operator in a sliding window manner as depicted in Figure 2.3. Given an input $x_i \in \mathbb{R}^{H_i \times W_i \times D_i}$ to the i-th layer, j-th neuron of the layer performs a convolution and a bias addition followed by a non-linear activation function ϕ:

$$x_{i+1}^j = \phi\left(x_i * \mathbf{k}_i^j + b_i^j\right) \tag{2.7}$$

This operation outputs a feature map $x_{i+1}^j \in \mathbb{R}^{H_{i+1} \times W_{i+1}}$ with width and height depending on the size of the kernel and on a definition of boundary conditions. Each convolutional layer is defined by D_{i+1} kernels $\mathbf{k}^1, ...\mathbf{k}^{D_{i+1}}$. As a result, the output of each layer defined as;

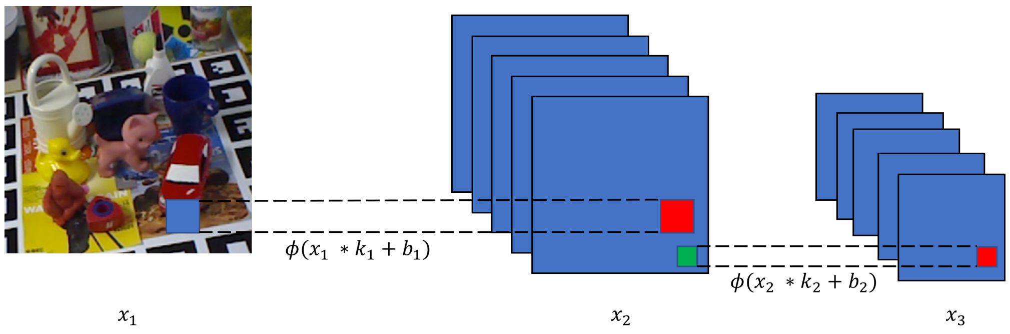


Figure 2.3: Schematic depiction of a convolutional neural network.

$$x_{i+1} = \left[\phi\left(\mathbf{k}_i^1 * x_i + b^1\right)|...|\phi\left(\mathbf{k}_i^{D_{i+1}} * x_i + b_i^{D_{i+1}}\right)\right] \tag{2.8}$$

Gradients of the loss function with respect to the kernel parameters are computed analogously to the case of fully-connected networks.

2.1.3 Essential Components of Neural Networks

Activation Functions. Non-linear activation functions are a core building block of any neural networks. While linear (or convolutional) layers are directly responsible for feature extraction, they cannot be stacked directly. A combination of linear functions is itself forms a linear function, which means that the expressive capabilities of the network do not increase of the number of layers. Thus, non-linear activation functions must be added atop of each linear layer. The two most popular activation functions are sigmoid and ReLU:

$$sigmoid\,(x) = \frac{1}{1+e^{-x}} \qquad\qquad ReLU\,(x) = max\,(0, x)$$

Normalization. Normalization layers have been a norm in deep learning since the very beginning. The original AlexNet paper relied on Local Response Normalization [87, 88]. The key moment of the development of normalization layers was the publication that introduced Batch Normalization [89]. For a layer with d-dimensional input $x = (x^1, ..., x^d)$, Batch Normalization implements the following transformation:

$$\hat{x}^k = \gamma^k \frac{x^k - \mathrm{E}\left[x^k\right]}{\sqrt{\mathrm{Var}\left[x^k\right]}} + \beta^k \tag{2.9}$$

During training, mean and variance are computed along the batch. Whereas during testing, the statistics over the entire dataset is used. $\gamma^k \in \mathbb{R}$ and $\beta^k \in \mathbb{R}$ stand for scaling and shifting parameters. Essentially, Batch Normalization layer first normalizes the output of the linear layer, but then re-scales and shifts it by learnable parameters in order not to decrease the expressive power of the network. Batch normalization is empirically

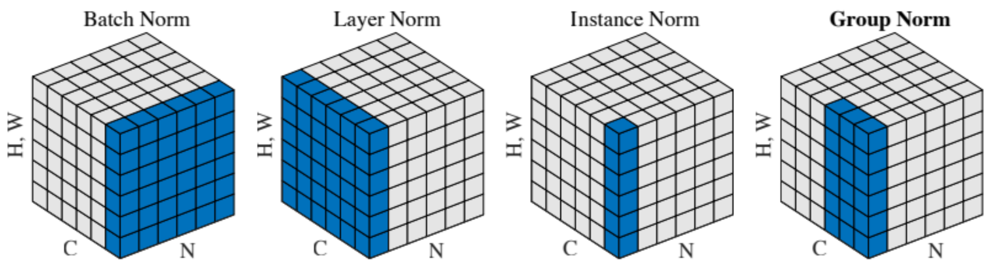


Figure 2.4: Comparison of normalization methods. Figure from [3].

proven to help faster and more reliable convergence by decreasing internal covariance shift. Another explanation of why Batch Normalization helps was shown in [90]. The claim is that Batch Normalization makes the the optimization landscape significantly smoother, allowing for more stable behavior of the gradients and faster training. Later, Instance Normalization [91] was proposed to normalize activations separately for each instance in a batch. In Group Normalization [3] normalizes groups of channels separately. An overview and comparison of various normalization techniques are provided in Figure 2.4.

Regularization. Regularization allows us to enforce priors on model parameters. This is especially useful for training very deep neural networks, when the number of trainable parameters can exceed the number of training samples. The most popular regularizers are called Weight Decays. The basic idea is to penalize the weights by either L1 or L2 loss. A particular loss fucntion comes from the assumed distribution of the weights. Another important technique is called Dropout [92], which works by randomly setting to zero a proportion of activations during training. This techniques help us fight against overfitting.

Pooling layers. An essential component of convolutional neural networks are so-called pooling layers. In contract to convolutional layers, the purpose of pooling layers is to reduce he spatial dimensionality of the feature tensor. The most commonly used pooling layers are max pooling and average pooling, which decrease the dimensionality by applying max or average operator over a local neighborhood in a sliding window fashion.

Optimization. As we discussed above, given a forward run of a network, a differentiable loss function can be minimized using gradient-based methods. The workhorse of deep learning is Stochastic Gradient Descent, which optimizes the weights of a neural network by taking a step in the negative gradient direction. Given a current estimate of network parameters as Θ_t and a differentiable loss $\mathcal{L}$, the update step is computed as follows:

$$\Theta_{t+1} \coloneqq \Theta_t - \alpha \mathrm{E}\left[\frac{\partial \mathcal{L}}{\partial \Theta_t}\right] \tag{2.10}$$

$\alpha \in \mathbb{R}$ is called an update step and is used to scale the gradient to avoid instabilities and oscillations. This parameter has to be sent by hand. In stochastic gradient descent, the expectation of the gradient (E $\left[\frac{\partial \mathcal{L}}{\partial \Theta_t}\right]$) is estimated only over samples in one batch. It is impossible to prove the convergence to the optimal solution due to non-convexity of the neural networks. However, it practice this procedure works well. Over the years, a numerous variants of gradient descent were developed with the aim of improving convergence and the speed of convergence. The most notable examples are gradient descent with momentum, Nesterov Accelerated Gradien, Adagrad [93], Adadelta [94], Adam [95].

2.2 Geometric Prerequisites

In this section, we briefly introduce geometric prerequisites needed for understanding of the thesis, namely mathematical definition of rigid body transformation, rotation parameterization and pinhole camera model. More details can be found in the standard computer vision textbooks [4, 96, 97]

2.2.1 Rigid Body Transformations

Rigid body transformation in 3D Euclidean space is understood as a mapping that preserves pair-wise point distances and normal vectors. Practically, it means rotation and translation of the object without deforming it. Following the formalism of [4], a rotation in 3D space can be mathematically represented as a matrix:

$$\mathbf{R} \in SO(3) \tag{2.11}$$

with $SO(3)$ being a Special Orthogonal Group. It is defined by the following properties:

$$SO(3) := \left\{\mathbf{R} \in \mathbb{R}^{3\times 3} \mid det(\mathbf{R}) = 1, \mathbf{R}^\intercal\mathbf{R} = \mathbf{R}\mathbf{R}^\intercal = \mathbf{I}\right\} \tag{2.12}$$

Being an algebraic group, this definition of $SO(3)$ ensures that for any number of rotations $\mathbf{R}_1, ..., \mathbf{R}_N \in SO(3)$, their product is still a valid rotation matrix $\mathbf{R}_1\mathbf{R}_2...\mathbf{R}_N \in SO(3)$. Moreover, the inverse rotation of any given rotation $\mathbf{R} \in SO(3)$ is simply its transpose $\mathbf{R}^T$.

The translation component of the transformation is simply defined as $\mathbf{t} \in \mathbb{R}^3$. Rotation and translation together make up a member of the Special Euclidean Group ($SE(3)$), which is formally defined as

$$SE(3) := \left\{\mathbf{T} \in \mathbb{R}^{4\times 4} \mid \mathbf{T} = \begin{Bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}_{1\times 3} & 1 \end{Bmatrix}, \mathbf{R} \in SO(3), \mathbf{t} \in \mathbb{R}^3\right\} \tag{2.13}$$

For any transformation $T = \begin{Bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0_{1\times 3}} & 1 \end{Bmatrix} \in SE(3)$, it has a unique closed-form inverse transformation defined by $\mathbf{T}^{-1} = \begin{Bmatrix} \mathbf{R}^\intercal & -\mathbf{R}^\intercal\mathbf{t} \\ \mathbf{0_{1\times 3}} & 1 \end{Bmatrix} \in SE(3)$

2.2.2 Rotation Parameterization

While $SO(3)$ group provides a clear mathematical definition of rotation, it is not always practical to represent rotations with 3×3 matrices because in this form it is hard to satisfy the constraints of the group. Overtime, several rotation parameterization have been proposed which are more suitable for non-linear optimization or for prediction with deep neural networks.

Euler Angles is one of the simplest rotation parametrization based on the group property of $SO(3)$ that a product of several rotation matrices is a valid rotation matrix. For each axis, an angle of rotation around it is first converted to a matrix and then applied to the point in 3D in turns.

A rotation around x-axis is defined as

$$\mathbf{R_x}(\psi) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\psi & -\sin\psi \\ 1 & \sin\psi & \cos\psi \end{pmatrix} \tag{2.14}$$

A rotation around y-axis is defined as

$$\mathbf{R_y}(\theta) = \begin{pmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{pmatrix} \tag{2.15}$$

A rotation around z-axis is defined as

$$\mathbf{R_z}(\phi) = \begin{pmatrix} \cos\phi & -\sin\phi & 0 \\ \sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{2.16}$$

The angles ψ, θ and ϕ form the Euler Angles. If we follow a convention that first a rotation the x-axis, then the y-axis, and finally the z-axis are applied, the total transformation will be defined as:

$$\begin{aligned} \mathbf{R} &= \mathbf{R_z}(\phi)\mathbf{R_y}(\theta)\mathbf{R_x}(\psi) \\ &= \begin{pmatrix} \cos\theta\,\cos\phi & \sin\psi\,\sin\theta\,\cos\phi - \cos\psi & \cos\psi\,\sin\theta\,\cos\phi + \sin\psi\,\sin\phi \\ \cos\theta\,\sin\phi & \sin\psi\,\sin\theta\,\sin\phi + \cos\psi\,\cos\phi & \cos\psi\,\sin\theta\,\sin\phi - \sin\psi\,\cos\phi \\ -\sin\theta & \sin\psi\,\cos\theta & \cos\psi\,\cos\theta \end{pmatrix} \end{aligned} \tag{2.17}$$

However, the order of applying the rotations might be chosen differently. One of the disadvantages of Euler Angles is the fact that they are non-unique. It means that various angles ψ, θ, ϕ can result in identical rotation matrix R. Another limitation is known as Gimbal lock, which happens when two rotation axes coincide after 90 degree rotation around a third axis. This basically removes one of the axes of rotation.

Axis-Angle representation uses Euler's theorem to represent a 3D rotation as a rotation θ around a unit vector $\mathbf{k} \in \mathbb{R}^3$ passing through the origin of the coordinate system.

Given an arbitrary vector $\mathbf{v} \in \mathbb{R}^3$, Axis-Angle representation uses the Rodrigues' rotation formula to rotate v:

$$\mathbf{v_{rot}} = \mathbf{v} \cos\theta + (\mathbf{k} \times \mathbf{v}) \sin\theta + \mathbf{k} (\mathbf{k} \cdot \mathbf{v}) (1 - \cos\theta) \tag{2.18}$$

where $\times$ and $\cdot$ stand for a cross and a dot products respectively. This representation is not susceptible to Gimbal lock.

Quaternions is a extension of complex numbers that allows for rotation of vectors. A quaternion is defined by a scalar $w \in \mathbb{R}$ and a vector $\mathbf{v} = (v_x, v_y, v_z) \in \mathbb{R}^3$ and is written as $\mathbf{q} = (w, \mathbf{v})$. Conceptually, quaternions are similar to Axis-Angle in a sense that they also represent a 3D rotation as a rotation around an axis. Given a unit vector $\mathbf{k} \in \mathbb{R}^3$ representing an axis of rotation and a rotation angle θ, the corresponding unit quaternion is defined

$$\mathbf{q} = (\cos\frac{\theta}{2}, \sin\frac{\theta}{2}\mathbf{k}) \tag{2.19}$$

Given a quternion $\mathbf{q} = (w, \mathbf{v})$, it can be converted to a rotation matrix as:

$$\mathbf{R}(q) = \begin{pmatrix} 1 - 2v_y^2 - 2v_z^2 & 2v_xv_y - 2v_zv_w & 2v_xv_z + 2v_yv_w \\ 2v_xv_y + 2v_zv_w & 1 - 2v_x^2 - 2v_z^2 & 2v_yv_z - 2v_xv_w \\ 2v_xv_z - 2v_yv_w & 2v_yv_z + 2v_xv_w & 1 - 2v_x^2 - 2v_y^2 \end{pmatrix} \tag{2.20}$$

A useful property of quaternions, in contrast to Euler angles and Axis-Angles, is the fact that rotations can be combined directly in the space in quaternions without converting them to rotation matrices. Given two rotations $\mathbf{R}_1, \mathbf{R}_2 \in SO(3)$ represented with their corresponding quaternions $\mathbf{q_1} = (w_1, \mathbf{v_1})$ and $\mathbf{q_2} = (w_2, \mathbf{v_2})$, rotation corresponding to $\mathbf{R}_1\mathbf{R}_2$ can be directly computed in quaternion space as:

$$\mathbf{q_1}\mathbf{q_2} = (w_1w_2 - \mathbf{v_1} \cdot \mathbf{v_2}, w_1v_2 + w_2\mathbf{v_1} + \mathbf{v_1} \times \mathbf{v_2}) \tag{2.21}$$

The downside of the quaternions is the non-minimal represenation of rotation (4 values for 3 degrees of freedom) and the fact that for any quaternion $\mathbf{q}$, quaternion $-\mathbf{q}$ represents the same rotation.

Continuous 6D rotation parameterizaion [98] presented a novel way to represent rotations with non-linear optimization in mind. This representation is shown to be more suitable for optimization due to its continuity, which was proven both mathematically and empirically. The core idea is to use two non-zero 3D vectors $\mathbf{r_1}, \mathbf{r_2} \in \mathbb{R}^3$, that are converted normalized and make orthogonal using Gram-Schmidt orthogonalization [99]. These 2 vectors form two columns of the rotation matrix. The third column is computed as a cross product of them:

$$\mathbf{R} = [\mathbf{R}_{\cdot,1} \mid \mathbf{R}_{\cdot,2} \mid \mathbf{R}_{\cdot,3}] = [\phi(\mathbf{r_1}) \mid \phi(\mathbf{R}_{\cdot,1} \times \mathbf{r_2}) \mid \mathbf{R}_{\cdot,3} \times \mathbf{R}_{\cdot,1}] \tag{2.22}$$

where ϕ stands for a normalization operator.

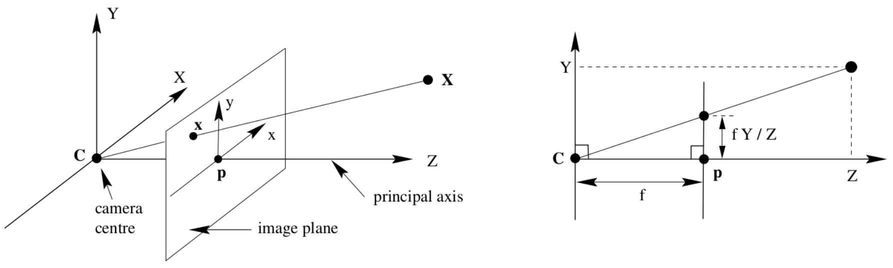


Figure 2.5: Pinhole camera model. Figure from [4].

2.2.3 Pinhole Camera Model

In this thesis, we used the standard pinhole camera model [4], visualized in Figure 2.5. It models the simplified process of 2D image formation from the 3D world, where the camera is modeled by a infinitesimal point and assumes no usage of lenses to focus light. This model is, however, well-researched, convenient and precises enough for the problems studied in this thesis.

In the basic pinhole camera model, the camera is parameterized with the matrix of intrinsics parameters:

$$K = \begin{Bmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{Bmatrix} \tag{2.23}$$

These parameters define how a 3D point is projected onto a 2D image plane. $f \in \mathbb{R}$ stands for so-called focal length, that shows the distance from the camera center (C in the Figure 2.5) to the image plane. $c_x, c_y \in \mathbb{R}$ denote a principal point of the image. In other words, it shows where the ray from the camera orthogonal to the image place intersects the plane. The exact projection transformation of any arbitrary 3D point $\mathbf{X} = (X, Y, Z)^T$ on a 2D point $\mathbf{x} = (x, y)^T$ using a projection operator $\pi_K : \mathbb{R}^3 \to \mathbb{R}^2$:

$$\mathbf{x} = \pi_K(\mathbf{X}) = \begin{pmatrix} f\frac{X}{Z} + c_x \\ f\frac{Y}{Z} + c_y \end{pmatrix} \tag{2.24}$$

This means, that transformation π_K is not invertible, as the distance information z is lost by projection. However, if the depth information (Z) is available, than 3D location of a 2D pixel $\mathbf{x}$ can be recovered:

$$\pi_K^{-1}(\mathbf{x}) = Z\dot{K}^{-1} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \mathbf{X} \tag{2.25}$$

3 Pose Estimation

In this chapter, we discuss the problem of 6 DoF pose estimation. We start off with the discussion of the common challenges of pose estimation. Then, we proceed to an overview of the related work. Then, we discuss the core methods and approaches needed for pose estimation with correspondences. The chapter concludes with a discussion of commonly used datasets and metrics for evaluation and benchmarking object pose estimation methods.

3.1 Main Challenges

While a significant progress has been made in monocular pose estimation from RGB, there are still issues to work on. The most critical are the following: 1) innate ambiguity of distance estimation from RGB images; 2) train set bias; 3) lack of or difficulty to obtain labeled real train data, 4) domain gap between real and synthetic data, 5) symmetric objects, 6) missing depth and 7) occlusions.

Projective ambiguity. The most fundamental problem comes from the perspective projection and the finitely discrete nature of RGB images. Given only a single RGB image, it is hard to estimate precise distance to the object because even relatively large translational errors might not be visible in the image. This is illustrated for various degrees of translational erroneous in Figure 3.1. To illustrate the issue, the ground truth pose of the object $\mathbf{T_{gt}} = \begin{Bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0_{1\times 3}} & 1 \end{Bmatrix}$ was perturbed with the following transformation $T_{pert} = \begin{Bmatrix} \mathbf{R} & \frac{\mathbf{t}}{\|\mathbf{t}\|_2} * (\|\mathbf{t}\|_2 + \epsilon) \\ \mathbf{0_{1\times 3}} & 1 \end{Bmatrix}$. Basically, it means that the camera ray is still the same, but the camera is shifted along it. Green line shows the outline of the object rendered on the image in its ground truth pose. The red line shows the outline of the object rendered some distance away from the ground truth along the viewing direction. In the top row, ϵ is set to 1cm, to 2cm in the middle row and to 3cm in the bottom row. It is clear, that with $\epsilon = 1cm$ both outlines look identical. With $\epsilon = 2cm$, the difference becomes more apparent on some objects but still stays indistinguishably small. Only when $\epsilon = 3$, the difference between the green and read outlines becomes clearly visible. It visually proves, that even with perfect correspondences, there will always be certain 3D error in pose estimates if only monocular RGb images are used.

Train set bias. Then comes the issue of how to rigorously evaluate the pose estimation performance and especially its generalizability. Many of the RGB detectors report pose estimation accuracy on datasets such as Linemod [5] or Occlusion [8], which do not have clear separation of train, validation and test splits. It leads to the situation when

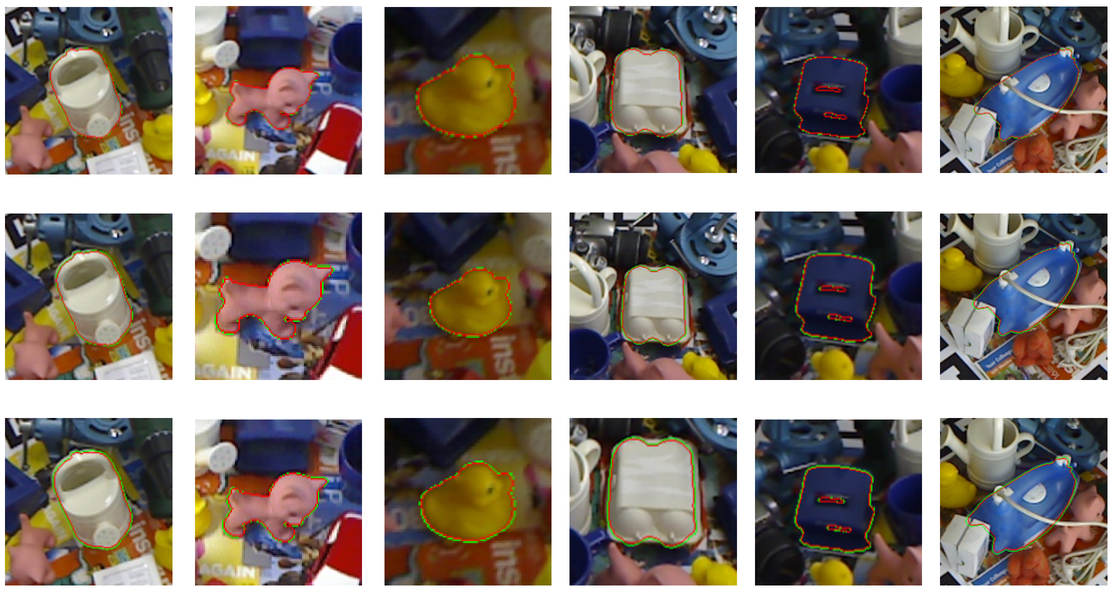


Figure 3.1: Examples of pose ambiguity in RGB images. Green line shows the outline of the object rendered on the image in its ground truth pose. The red line shows the outline of the object rendered some distance away from the ground truth along the viewing direction.

real train images are sampled exactly from the same images sequence as the test images, for example as in [100, 101, 102, 103]. Depending on the sampling procedure, this can lead to a significant bias to poses and occlusions seen in the train set and does not introduce any domain gap between the train and the test splits. As a result, the comparison between different methods and their generalization capabilities can be severely compromised. It significantly improves the reported numbers, because this sampling strategy does not introduce any domain gap between the train and the test images and because poses in the train set closely resemble the poses in the test set. Therefore, it is impossibly to judge how the methods generalize to new, unseen environments. As was shown in [6], deep learning methods trained on real data tend to perform much worse when the domain changes. This issue was to a large extend illiminated by the introduction of the BOP Challenge [7] and its unified evaluation procedure.

Lack of real train data. While training on real labeled data is still predominant in the computer vision community, its use is not practical for the task of 6 DoF pose estimation. Real train data usually have to be manually annotated to produce labels needed for a particular task. While it is relatively simple for classification and more difficult for 2D detection and segmentation, manual annotation of 6 DoF pose is simply impossible. To obtain 6 DoF pose labels, one needs to resort to complex and time-consuming multi-stage pipelines, such as in [6]. Nevertheless, real labels might still be imprecise as highlighted in Figure 3.2. This essentially limits quantity and diversity of prepared train images. A possible solution is to avoid training on real data and use synthetically generated images. However, such approaches typically must rely on

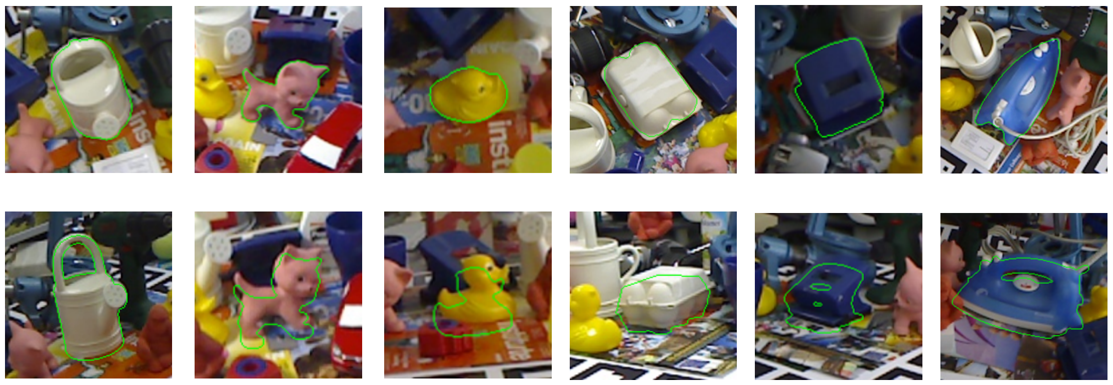


Figure 3.2: Examples of precise and imprecise ground truth pose labels. Green line shows the outline of the object rendered on the image in its ground truth pose. The top row illustrates perfect alignment between the object in the image and the rendered object. The bottom row, on the other hand, demonstrates imprecise alignment which indicates imprecise ground truth poses. All poses are real ground truth annotations from Linemod [5]

transfer learning [104, 100] or smart data pre-processing and prior geometric assumptions instead [105]. Although it was shown in [6] that detectors trained on synthetic data perform better on new unseen data, they still show worse performance than the detectors trained on real data from the target domain. Poses obtained by detectors, which utilize only synthetic data for training, still need further pose refinement to achieve precise estimates.

Domain gap. While training neural networks on syntetic data might sound appealing, it has the downside of the discrepancy between real and synthetic data domains leading to the worse quality of estimated poses. Even though there are numerous GAN-based approaches for domain adaptation, most of such methods still require access to labeled images from the target domain, which brings back all of the aforementioned problems. The problem was partially remedied by introduction of higher quality synthetic train data, as for example in the BOP challenge [7]. The problem of domain gap is illustrated in Figure 3.3.

Symmetric objects. Symmetric object are fundamentally problematic no matter what kind of input data (RGB or depth) is used. Figure 3.4 illustrates several types of symmetric objects. Symmetric objects are problematic because pose is not well-defined for them. Typically, infinitely many rotation matrices result in the non-distinguishable scene configurations. Moreover, when training a deep neural network, special care must be taken of ground truth poses to ensure that they are consistent and non-conflicting. It practically means that either the network get the same rotation matrix R as ground truth pose every time, when one of the ambiguous poses is presented to the network during training, or that the loss function explicitly handles it. Otherwise, the training time will increase significantly and training will potentially result in worse pose predictions.

Imprecise or missing depth information. In general, depth information obtained with the commodity sensors, such as Primesense or Kinect, tend to be less dense and

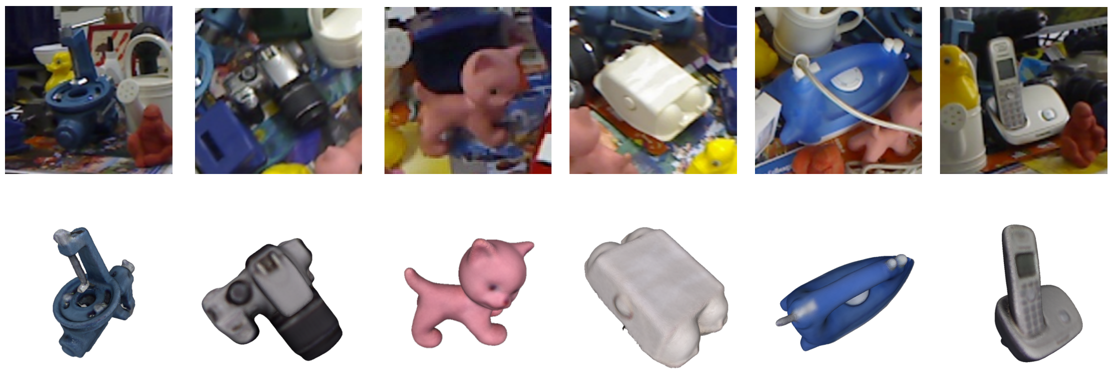


Figure 3.3: Examples of the domain gap between real and synthetic data. The top row illustrates patches with objects taken from the real test images. The bottom row illustrates rendered CAD models of the respective objects.

precise than the standard RGB cameras. Additionally, they usually have limited resolution. For the task of pose estimation, however, the most problematic aspect is missing depth. Depth is missed if the object is placed not in the effective range of the sensor, i.e. too close or too far, if the angle between the surface and the viewing direction is almost 0 or for objects of certain materials.

Occlusions. Occlusions pose a significant challenge for all stages of pose estimation pipelines. First, it is harder to detect the objects due to its smaller size in the image and potentially hidden key parts of the object. The same problems pose problems for precise correspondence estimation. Several examples of scenes with extreme occlusions are provided in Figure 3.5.

3.2 Related Work

As 6 DoF pose estimation is a crucial task needed for a wide range of downstream applications, it has been extensively studied in the past. In this section, we refer the existing deep learning-based approaches for pose estimation. Additionally, state of the art of object pose refinement is discussed.

3.2.1 Template and Retrieval-Based Methods

The core idea of template-based methods is to represent the object with a set of images, referred to as templates, capturing the object's appearance from various viewpoints. A set of such templates is extracted to approximately cover all possible object's views. At the test time, a representation of the input image is compared to the entire set using a predefined matching score. The best matches are pruned to get the final result.

One of the leading pre-deep learning template-based methods for 6 DoF pose estimation is LineMOD [66, 106]. It relies on a combination of the RGB gradients and surface

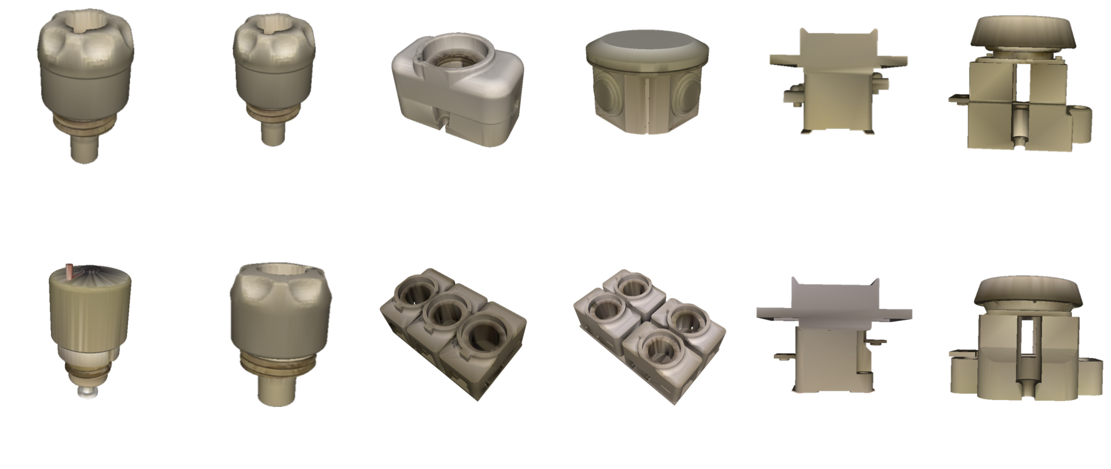

(a) Symmetric around an axis **(b)** Discrete symmetries **(c)** Almost symmetric objects

Figure 3.4: Examples of various symmetry types.

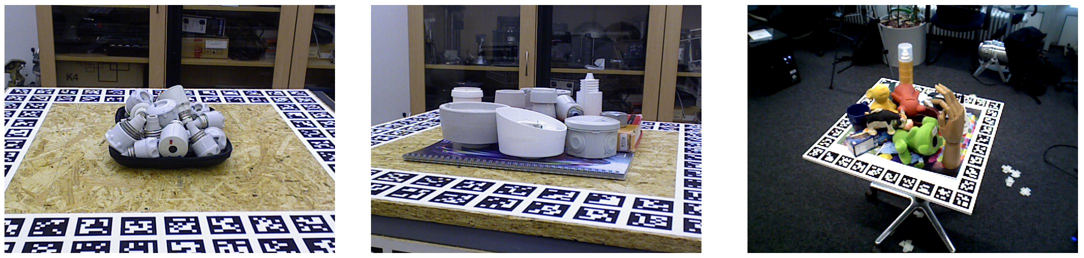


Figure 3.5: Examples of scenes with severe occlusions.

normals computed from depth images. The method can use a set of templates covering a large number of angle and scale combinations and still estimate the pose in real time because of its highly-efficient implementation. A method was further extended in [5], which allowed or usage of synthetically rendered templates. The further follow-up works aimed to improve the scalability issue related to the used number of temples allowing for significant speed-ups [107, 108, 109, 110].

With the advent of deep learning, the focus of the researchers shifted to CNNs, which replaced the traditional hand-crafted descriptors with learnable descriptors. For example, [111] mapped image data directly to the similarity-preserving descriptor space. The authors utilized a Siamese network to learn the mapping. A cost function was defined such that distance between descriptors of similar objects was minimized and maximized otherwise. In [71], it was proposed to use a triplet loss to learn how to map image data directly to the similarity-preserving descriptor space. The matching is then done by nearest neighbor search on the descriptor space to retrieve the closest neighbors. The follow-up works [70], [112] extend the method by improving the descriptor space by introducing a modified loss function, and a multi-task extension combining manifold learning with direct pose regression leveraging the advantages of both. Another line of works [113, 114] strive to improve the domain generalization capabilities of the method making it possible to train it from synthetic data while performing on par and better than real methods by using the inverse domain adaptation technique.

Building on the success of the above methods, it was proposed in AAE [115, 116] to implement a full 6 DoF pose estimation pipeline based on learned descriptors by using already computed SSD [56] detections as input. The main idea is to split object detection and pose estimation into two separate steps. First, a standard 2D detector is trained on real data. The second step, an auto encoder was trained in a weakly supervised manner. As input the auto encoder gets a patch with an object in arbitrary pose. Then, the autoencoder was trained to reconstruct the given object in its canonical pose. This way, a latent representation of the shape was learnt without any explicit constraints. In test time, a patch was mapped to its latent code using the decoder part of the autoencoder. The latent code was matched against a database of pre-computed codes. Rotation which corresponds to the closest latent code is selected as the predicted rotation. Translational component of the transformation is estimated based on the size of the bounding box. Sundermeyer et al. [82] demonstrate that rotation estimation by template matching can generalize to new objects that were not seen during training, even if the feature extractor was trained on objects from a different dataset.

Alternatively, the SSD6D detector [104] proposed an end-to end deep learning solution based on the template matching idea. The authors extended the standard out-of-the-box SSD [56] detector to jointly detect the objects and estimate their poses. SSD6D relies on a discrete viewpoint classification rather than direct regression of rotations. To make it possible, all poses are divided into a large number of discrete ones, and each of them is further divided into a smaller number of discrete in-plane rotations. As a result, this significantly reduces the solution space allowing for simpler training of the network. Translation is estimated based on the ratio of the predicted bounding box diagonal to the diagonal of the object rendered from some pre-defined distance. However, rotation discretization and approximation of translation lead only to rough pose estimates, which inevitably need further refinement.

3.2.2 Correspondence-Based Methods

A popular alternative to template-based methods are methods utilizing 2D-3D correspondences between the input image and the 3D object model. Their power lies in a reduced solution space and explicitly introduced geometric constraints. Predicting a larger number of correspondence allows for outlier removal with the standard sampling techniques such as RANSAC [4]. In contrast to template-based methods, correspondence-based methods do not suffer from the discretization problem. It is also arguably easier for a neural network to localize points in the 2D image plane rather than to work with the actual 3D space. This results in more precise pose estimates and the increased robustness to occlusions and clutter. While the idea of established 2D-3D correspondences between the image and the object of interest is not new, it has also benefited from deep learning. Pre-deep-learning, researchers proposed a variety of hand-crafted keypoint detectors and keypoint descriptors, such as SIFT [77, 21], SURF [22], BRIEF [23], ORB [24], which could be robustly and repeatedly extracted from images. Deep learning techniques, however, allowed to more flexible keypoint detection even for low-textured objects.

BB8 [68] was arguably the first to utilize correspondences predicted with deep learning to estimate object poses. The holistic method consists of three stages. In the first two stages, a coarse-to-fine segmentation is performed, the result of which is then fed to the third network trained to output projections of the object's bounding box points. Knowing 2D-3D correspondences, a 6 DoF pose is estimated using the PnP method. The main drawback of this method is its multi-stage nature, resulting in slow run times, and imprecise poses.

Building atop of advances in 2D object detection [57] and the BB8 ideas, YOLO6D [69] proposed a single-shot deep learning architecture for efficient and precise object detection and pose estimation. The key idea is to modify a standard 2D object detection netwrok to also regress locations of the reprojected bounding box corners. Keypoints are predicted separately for each candidate anchor box with respect to its 2D location and size. Once correspondences are estimated, the pose can be estimated using a EPnP solver [1]. This architecture ensures rapid keypoint estimation. Additionally, the small number of used keypoints and their quality ensures that there is no need for RANSAC. Simple EPnP predicts poses reliably even given imperfect keypoints.

PVNet [103] takes a different approach and designs a network which for every pixel in the image regresses an offset to the predefined keypoints located on the object itself. The ideas on PVNet were further developed in several works. PVNet3D [117] takes both RGB and and depth images as input. After independent feature extraction from each data modality, the features a merged and Hough voting is performed to estimate locations of the pre-defined keypoints in 3D space. Then, a least-square algorithm is used to estimate the pose. Alternatively, HybridPose [102] extends the initial PVNet approach by also predicting edges and axes of symmetries. They are used together with the predicted keypoints to estimate the pose with the modified PnP algorithm.

Another alternative is to use dense correspondences, when 2D-3D correspondences are established for all object pixels visible in the image. The assumption is that a larger number of correspondences will mitigate the problem of their inaccuracies and will result in more precise poses. While establishing few correspondences is likely to result in an inaccurate pose, dense correspondences can still lead to a good solution. Moreover, it allows for a significantly better treatment of occlusions.

iPose [118] operates in 3 stages: segmentation, 3D coordinate regression and pose estimation. In contrast, Pix2Pose [101] unify the first two stages into the end-to-end network. CDPN [73] disentangles 6 DoF pose estimation by using 3D correspondences for rotation estimation and regressing translation directly from the image. EPOS [119] proposed a different dense corresponding parameterization. Instead of a simple prediction of 3D locations of each 2D point, the network predicts two values. First, it classifies a pixel according to which object segment it belongs to. Then, a precise coordinate within the segment is regressed. This parameterization allows for easier handling of object symmetries. Combined with a robust and efficient variant of the PnP-RANSAC algorithm, the method achieves excellent results on the TLESS dataset.

3.2.3 Direct Pose Regression

It is also possible to directly predict a mapping from an input image to the object pose with deep neural networks. One of the first works to ever do that was the PoseNet [120], where a CNN is utilized to regress the position and orientation of a camera given an RGB image. Subsequent works improved the localization results by combining a CNN with LSTM units [121] and studied the effect of different loss functions on the final performance [122].

PoseCNN [16] aims to disentangle the pose estimation into several sub-tasks, i.e. regressing object masks, estimating the translation of the object centroids, and regressing quaternions for rotation. More specifically, object masks are estimated using a standard encoder-decoder network for semantic segmentation. Objects centers are estimated by pixel-wise hough voting in the direction of the center. Additionally, a distance to the object is predicted for each pixel. Then, a quaternion was estimated in the allocentric coordinate system for each detected object.

DenseFusion [123] is trained to directly predict object pose from RGBD images. Pose is predicted separately from each pixel. For each pixel, an additional pose confidence, which is trained in an unsupervised manner, is predicted. The pose with the highest confidence is chosen as the final prediction. The method performs better than the aforementioned direct methods due to its usage of depth data.

CosyPose [10] re-visited the task of direct pose regression when better synthetic train data and better understanding of pose estimation with deep learning was reached by the community. CosyPose relied on 3-stage approach to estimate initial object poses. First, a standard 2D detector detects object candidates from an RGB image. Then, an additional networks predicts rotation and translation separately for each patch. These initial coarse predictions are then refined with a DeepIM-like method [9]. Additionally, CosyPose introduced a multi-view pose refinement which utilized independent predictions from several frames.

Despite their seeming simplicity, the direct pose regression methods report significantly lower accuracy when compared to template- and correspondence-based solutions. Therefore, post-processing pose refinement, e.g. ICP, is usually performed to overcome the issue.

3.2.4 Generazability of Pose Estimation Method to Novel Objects

The aforementioned deep learning methods are generally restricted to handling only one object instance at a time. In other words, it means that a network is trained separately for each object. When a need for pose estimation of a new object arises, training data has to be generated and the network has to be re-trained from scratch. This is a crucial shortcoming that needs to be addressed to make pose estimation ready for industrial and commercial usage. There have been several attempts to deal with this problem and to allow the methods to generalize to novel objects. They can roughly be divided into two categories: category-level methods and one-shot methods.

NOCS [74] proposed a novel concept of category-level pose estimation, as opposed to instance-level pose estimation. The core idea is to train a single network for an entire category of objects. For instance, a network is trained to estimate the pose of all cups rather than of a particular cup instance. The NOCS method consists of predicting dense per-pixel 2D-3D correspondences between a detected object and a category-level Normalized Normalized Object Coordinate Space. Predicted 2D-3D correspondences are projected into 3D using the available deth maps. The final shape and pose are estimated with the Umeyama algorithm [124]. The also also released the first dataset for category-level object pose estimation and an evaluation pipeline. This line of methods was further developed in a number of works, fr example in [125, 126, 127], which allowed for more precise pose and shape estimation, including their estimation from RGB images. In spite of the progress of such methods, the performance is still highly dependent on inter-class similarity between objects used for training and testing. Additionally, a large number of category instances are required during training.

The next class of methods is referred to as one-shot methods in this thesis. In contrast to the previous methods and the standard vision models, that accept only a single RGD or RGBD image as input to the method, one-shot methods jointly take an image and a descriptor of the object of interest. With two of them at hand, they aim at localizing the object in the given RGB/RGBD image even if the object was not used for training or does not belong to object categories used for training. This allows for more flexible object detection and 6 DoF pose estimation of objects whose 3D CAD models are known.

The classical and, arguably, the only commercially available [128] application-ready one-shot approach for object detection and pose estimation is Point Pair Features (PPF) [15]. The method works by representing local geometry of the input scene and of the object model with simplistic descriptors made up of oriented point pairs. Scene and model descriptors are then exhaustively matched to each other. The final pose hypotheses, though, require ICP or image-based refinement to achieve reasonable quality of poses. Over the years, the idea was further developed to improve the speed and robustness [129, 130, 131] or to benefit from advances in deep learning [132, 133, 134, 135]. PPF-based methods led the BOP challenge [7] until very recently, when they were outperformed by deep learning methods operating purely on RGB. The disadvantages of the PPF methods is their dependence on depth information with reliable per-point normals and a slow run time, which limits their potential applications.

Another line of research attempts to capitalize on similarities between objects in training and testing sequences. CorNet [80] used only corners to approximate object geometry. Corners were detected in the image using a modified Faster-RCNN [136] and later exhaustively matched to the corners of the CAD models. In [81], a network was trained to predict dense per-pixel local shape descriptors for each object pixel. These descriptors are matched with the object model represeted with the same desriptors. In both methods, the final pose is computer with PnP and extensive inlier search with RANSAC. These methods needed a high degree of similarity between training and testing objects to work well. They were trained and evaluated on non-overlapping objects from the TLESS dataset [12] containing highly-similar industrial objects.

Sundermeyer et al. [82] extended the idea of the original Augmented Autoencoders [116] by adding a multi-path decoder on top of a shared encoder. This allowed for training on multiple of objects at the same time. The shared encoder projects RGB images of different objects onto the unified feature space, whereas object-specific decoders are used to reconstruct the objects and define the loss function. The decoders are disregarded in test time. It was empirically proven that the feature vectors, predicted with the encoder, generalize well to new unseen objects. The method, however, trains a 2D object detector on the target object classes. Similarly, Nguyen et. al. [137] that rotation estimation using template matching is robust to occlusions and generalizes to new unseen objects.

LatentFusion [138] proposes a fully differentiable end-to-end latent reconstruction and rendering pipeline. In train time, a 2D U-Net and a 3D U-Net networks [61] are trained to predict a latent object representation, that can be then rendered given an object pose. During testing, pre-trained U-Net networks are used to predict a latent object representation of a novel object from several RGB images with known relative transformations, The pose is then estimated by aligning rendered and observed depth in a render-and-compare fashion. The method, however, only deals with rotation and translation estimation of already detected objects, which is a very strong assumption. Additionally, a course initialization of the initial pose is required.

FS6D [139] proposed to explicitly find 2D-2D correspondences between a new incoming RGBD frame and a set of images (called support images) with the object of interest. Given the predicted 3D-3D correspondences, a pose of the object in the new frame can be computed as a relative transformation from one of the support views. The method requires the objects to be already detected.

In the work of [140], features computed from the input image are stacked with the object descriptor and passed to a network that directly predicts 3D rotation. Object descriptors are computed either from a 3D CAD model of the object using a PointNet [141] or by fusing features computed from multi-view RGB images similarly to [142]. However, the method assumes that the object is already perfectly localized in 2D and estimates only the rotational component of the pose rather than the full 6 DoF pose.

3.2.5 Pose Refinement

3.2.5.1 RGB Methods

In RGB images, the object pose has been traditionally refined using edge alignment [143, 143]. The idea is to align the edges of the object rendered in the pose hypothesis with the edges observed in the image. This approach is very sensitive to the image quality, clutter, occlusions and to the way in which the edge correspondences are computed. Kaskman et.al. [6] presented a more robust version of edge-based alignment that benefited from calibrated multi-view images.

With the advance of deep learning, researchers have attempted to utilize neural networks for the purpose of refinements [144, 9]. The idea here is to use an external pose estimation algorithm to obtain the initial pose approximation. Then, the object is rendered in the predicted pose. A rendered image and a given input RGB image are then put into another convolutional neural network, which directly regresses a pose offset. The process is repeated until convergence. In [144], the refiner is trained to minimize align 2D contours of the observed object and the rendered object, whereas DeepIM [9] is trained to directly maximize the ADD score. In BB8 [68], on the other hand, the refiner was trained to update predicted locations of keypoints. All of them use pose rectification, similar to [125] to take into account the fact that the pose offset is regressed from image patches, not full images, which introduces additional ambiguity.

Disadvantages of the aforementioned deep learning-based refiners are their dependence on the correct pose error priors used during training and the need to retrain them for each new object in order to obtain high-quality results.

3.2.5.2 Depth Methods

Object pose refinement has been studied thoroughly in the past. Iterative Closest Point [67] (ICP) is one of the most classical and used approaches. In a nutshell, the ICP algorithm takes a given pose hypothesis and aligns it to the point cloud. ICP refines a given pose iteratively by establishing one-to-one correspondences between the point cloud and object's vertices based on their spatial proximity and then minimizing distances between them. There are various variants of ICP and ICP-like algorithms, but their common fundamental limitation is that they require reliable registered depth information, which is not always readily available. Effectiveness of the ICP refinement is also dependent on the quality of depth maps. It restricts their applicability in many scenarios, for example, on images from mobile phones or from edge devices. The correspondences are established based on their spatial proximity. In the next step, object's pose is updated to minimize the distances between correspondences. Particular ICP variants differ in how correspondence distance is defined, how correspondences are established and how the distance is minimized [145, 146, 147].

DenseFusion [123] replaces a standard depth-based ICP with a deep learning analogue. At each iteration, we reuse the image feature embedding from the main network and perform dense fusion with the geometric features computed for the new transformed point cloud. The pose residual estimator uses as input a global feature from the set of fused pixel features. As with the standard ICP, depth information is required.

3.2.5.3 Multi-View Methods

This topic has been studied in the past in several works, most notably in [148, 149, 150]. The papers display two main trends. First, they all rely on independent pose hypothesis prediction from each monocular RGB image. Secondly, relative camera transformations between frames are assumed to be known beforehand. Known camera transformations are used to fuse pose predictions from several views in the global coordinate system. Then, either the pose hypothesis that aligns the best with the other hypotheses is chosen or poses are refined to align better in the 3D space.

In [148], pose hypotheses are simply clustered and then per-class average in the quaternion space is chosen as the final refined pose. Additionally, it uses active learning and viewpoint entropy to dynamically choose the most informative next view. In [149], pose predictions from several views are used to fit a Normal distribution in the 6D space. Then, the peak of the distribution is chosen. [149] also uses depth images as an additional input. In [150], different hypotheses are not merged as in [148, 149]. Instead, they proposed to choose the best hypothesis from the set of all of them. A hypothesis, which aligns the best with all the other hypotheses in terms of the ADD measure [5], is considered to be the most probable and picked as the final pose.

In contrast to the aforementioned methods, the approaches of in [105, 10] do no assume known relative camera transformations. In [105], a set of uncalibrated RGB images is used to create a scale-ambiguous 3D reconstruction, which was then approximated by only strong line segments present in the reconstruction. Then, objects are detected in the 3D reconstruction that correspond to joint detection in all the frames. Despite its good performance, the method had the downside that it comprised a number of complicated and time-consuming steps and the need to use a large number (72) of frames to obtain reliable reconstructions. CosyPose [10] also relies on independent pose hypotheses from each frame. However, in contrast to [148, 149, 150], they are then used in a RANSAC scheme to match pose hypotheses from several frames and produce a unified object-level scene reconstruction and approximate relative camera poses. They are jointly optimized by minimizing the multi-view reprojection error.

3.3 Pose Estimation From Correspondences

This section presents a quick overview of the tools and techniques that are necessary for pose estimation of rigid objects using 2D-3D or 3D-3D correspondences. First, we discuss how the pose is computed given a set of known correspondences. Then, an outlier removal technique is discussed.

3.3.1 Perspective-n-Point (PnP)

The essential tool for pose estimation with correspondences from monocular RGB images is the Perspective-n-Point (PnP) algorithm [4]. Visualization of the PnP problem statement and the notation used is provided in Figure 3.6.

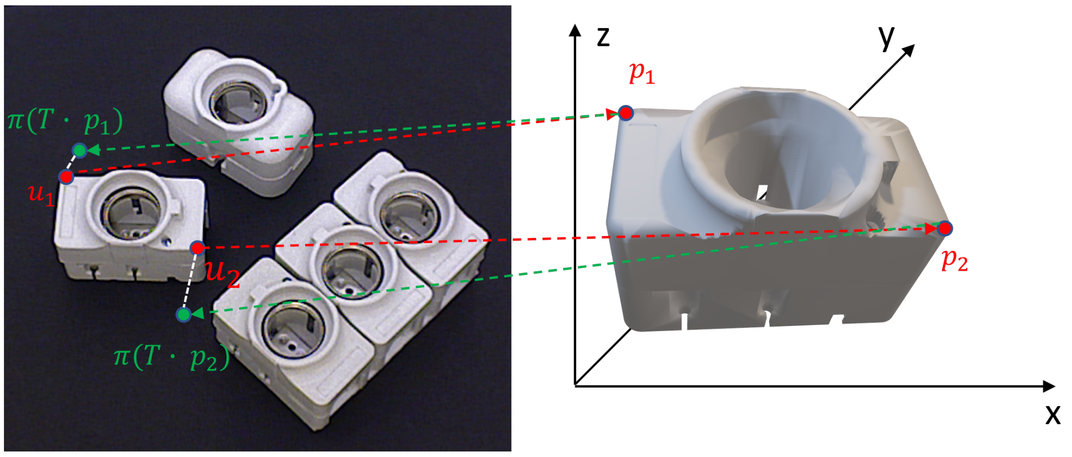
The goal of PnP is to estimate a pose of the object in the given camera frame $\mathbf{T} \in SE(3)$ given a camera with intrinsic parameters $\mathbf{K}$ and a set of 2D-3D correspondences $\mathcal{C} := \left\{\left(u_i, p_i | u_i \in \mathbb{R}^2, p_i \in \mathbb{R}^3\right)\right\}_i$. In general form, the PnP problem is typically formulated as follows:

$$\mathbf{T}^* = \underset{\mathbf{T} \in SE(3)}{\operatorname{argmin}} \sum_{i \in \mathcal{C}} \operatorname{dist}\left(\pi_{\mathbf{K}}\left(\mathbf{T} \cdot p_i\right), u_i\right) \tag{3.1}$$

where $\operatorname{dist}(\cdot)$ is an arbitrary distance metric on the $\mathbb{R}^2$ space.

In theory, 3 points are enough to estimate the full 6D pose of the object, giving rise to the P3P algorithm [151]. However, the pose quality of such methods is poor and, therefore, methods tend to use more correspondences [152, 153, 154, 155, 1]. While

Figure 3.6: Illustration of the PnP algorithm. 2D points (u_i) from an RGB images are first matched to the corresponding 3D points (p_i) on the model. Then, the reprojection loss is minimized to compute the object pose.



many formulations of the PnP problem, including P3P and EPnP [1], have a closed form solution, their initial predictions are still typically refined with additional non-linear optimization using the gauss-Newton method. Differentiable and deep-learning-based PnP algorithms were also proposed, for example in [156, 157, 158, 159, 160].

3.3.2 Kabsch Algorithm

Kabsch algorithm [124, 4] is one of the most fundamental algorithms in 3D computer vision. Visualization of the problem statement and used notation is given in Figure 3.7. Given two point clouds:

$$P \coloneqq \left\{p_i \in \mathbb{R}^3\right\}_{i=1}^N \qquad Q \coloneqq \left\{q_i \in \mathbb{R}^3\right\}_{i=1}^N \tag{3.2}$$

such that $q_i = \mathbf{T} \cdot p_i$ holds in the ideal scenario for all $i \in [1, ..., N]$ with some $\mathbf{T} \in SE(3)$. In reality, we correspondences will be noisy and, therefore, zero-centered Gaussian noise is assumed in the model: $q_i = \mathbf{T} \cdot p_i + e$. The goal is to estimate $\mathbf{T}$.

There is a closed-form solution [124, 4] to the problem which uses the Singular Value Decomposition. First, both point clouds are zero-centered with $\bar{p} = \frac{1}{N}\sum_{i=1}^N p_i$ and $\bar{q} = \frac{1}{N}\sum_{i=1}^N q_i$

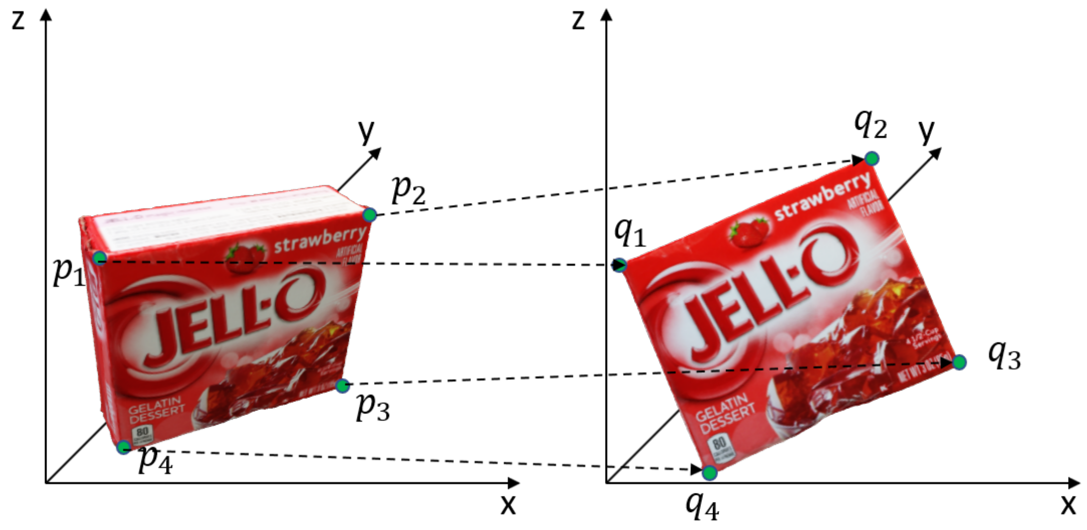
$$\bar{P} \coloneqq \{\bar{p}_i | \bar{p}_i = p_i - \bar{p}\}_{i=1}^N \qquad \bar{Q} \coloneqq \{\bar{q}_i | \bar{q}_i = q_i - \bar{q}\}_{i=1}^N \tag{3.3}$$

.

With that notation in hand, the following least square optimization problem is solved:

$$\underset{R \in SO(3)}{\operatorname{argmin}} \sum_{i=1}^N \|\bar{q}_i - \mathbf{T} \cdot \bar{p}_i\|_2^2 = \underset{\mathbf{R} \in SO(3)}{\operatorname{argmin}} \sum_{i=1}^N \|\bar{q}_i^\intercal \bar{q}_i + \bar{p}_i^\intercal \bar{p}_i - 2\bar{q}_i^\intercal \mathbf{R} \bar{p}_i\|_2^2 \tag{3.4}$$

Figure 3.7: Illustration of the Kabsch algorithm. The goal is to estimate an $SE(3)$ transformation from the initial point cloud (points p_i) to the target point cloud (points q_i) given known 3D-3D correspondences.



This minimization problem is solved in closed-form by defining the matrix

$$\mathbf{H} \coloneqq \sum_{i=1}^{N} \bar{p}_i \bar{q}_i^\intercal \tag{3.5}$$

Then, given an SVD decomposition of $\mathbf{H}$ as $\mathbf{H} = U\Sigma V^\intercal$, the optimal rotations is computed as following:

$$\mathbf{R} = UV^\intercal \tag{3.6}$$

and the translational component $\mathbf{t}$ is computed as

$$\mathbf{t} = \bar{q} - \mathbf{R} \cdot \bar{p} \tag{3.7}$$

3.3.3 RANdom SAmple Consensus (RANSAC)

In general, the PnP (or Kabsch) algorithm cannot be directly applied to 2D-3D (or 3D-3D) correspondences, as any incorrect correspondences present in them will undermine the quality of pose estimation. The RANSAC [78] algorithm is a general approach for robust model fitting, which is not limited to 6 DoF pose estimation and can be applied to a vast number of other problems that are problematic due to the presence of outliers. The overall idea is that a subset $\mathcal{C}_{in}$ of correspondences is sampled from the full set $\mathcal{C}$. Smaller cardinality of $\mathcal{C}_{in}$ increases the probability of the clean subset without any outliers. Then, the pose is estimated using only subsampled correspondence pairs.

Algorithm 1 The PnP + RANSAC algorithm, adapted from [4].

```
Input: $\mathcal{C}$ — a set of 2D-3D correspondences with outliers
Output: $\mathcal{C}_{in}$ — a set of 2D-3D correspondences which are considered as inliers, transformation matrix $\mathbf{T} \in SE(3)$.
1: Randomly select a sample of $s$ data points from $\mathcal{C}$ and use them to estimate initial rotation $R \in SO(3)$ and translation $\mathbf{t} \in \mathbb{R}^3$ with PnP, which make up a rigid body transformation hypothesis $\mathbf{T} \in SE(3)$.
2: Determine the set of 2D-3D correspondences $\mathcal{C}_{in}$ that form inliers. To be considered an inlier, the reprojection error if the correspondence should be less than a threshold $\epsilon$. Thus, $\mathcal{C}_{in} := \{(u_i, p_i) \in \mathcal{C} | \text{dist}(u_i, \pi(\mathbf{T} \cdot p_i)) < \epsilon\}$ such that it consists of all pairs with reprojection error smaller then threshold $\epsilon$.
3: If the cardinality of $\mathcal{C}_{in}$ (the number of inliers) is greater than threshold $\eta$, re-estimate rotation $\mathbf{R}$ and translation $\mathbf{t}$ using all the points in $\mathcal{C}_{in}$ and terminate.
4: If the size of $\mathcal{C}_{in}$ is less than $\eta$, select a new subset and repeat the procedure.
5: After $N$ trials the largest consensus set $\mathcal{C}_{in}$ is selected, and rotation $\mathbf{R}$ and translation $\mathbf{t}$ are re-estimated using all the points in the subset $\mathcal{C}_{in}$.
```

Having estimated a pose $\mathbf{T} \in SE(3)$, the quality of this pose estimate is evaluated. In case of 2D-3D correspondences and the PnP algorithm, it is done by projecting all the $3D$ points in $\mathcal{C}$ onto the image plane and measuring reprojection error. In case of 3D-3D correspondences, the correspondence-wise error is computed directly in the 3D space. *Inliers* are the points for which reprojection error is less than a threshold ϵ, otherwise a point is called an *outlier*. The procedure is repeated multiple times until a pose with the given number η of inliers is found or the maximal number of iterations N is reached. The final pose is estimated using all the inliers of the pose with the largest number of inliers. A more formal pseudocode of the RANSAC algorithm for pose estimation with the PnP algorithm is provided in Algorithm 1. RANSAC+Kabsch operates similarly with the only difference of how the error is computed.

It is of crucial importance to choose parameters N and η. Similarly to other sampling techniques, larger N might be beneficial as a bigger amount of poses is checked out. However, increasing N inevitably slows down the process. As a result, it is necessary to find such N which ensures the desired trade-off between speed and precision. Reprojection error threshold is also application-specific and depends to a large extent on the camera used and distances of objects from the camera. The threshold is specified in pixels. Therefore for different camera matrices, an offset of one pixel will correspond to different actual differences in the 3D space. The threshold ϵ for the Kabsch+RANSAC algorithm is more universal and generic, as it is computed directly in 3D and, thus, independent of camera intrinsic parameters.

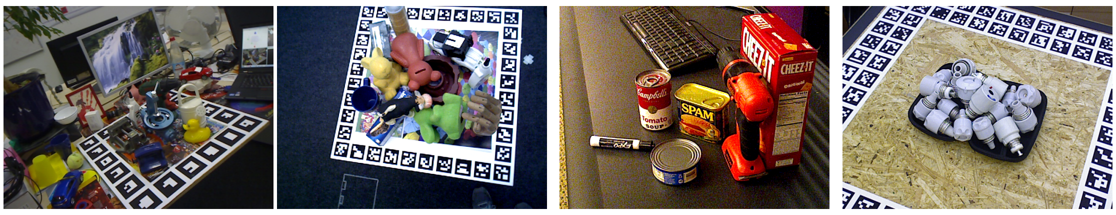
Being one of the pillars of computer vision, the RANSAC algorithm was re-worked and extended multiple times, for example by using local optimization in LO-RANSAC [161], preventing degenerate solutions in DEGENSAC+[162], using Grap-Cuts for local optimization [163], soft threshold in MAGSAC [164, 165] and USAC [166]. There were also attempts to make the RANSAC pipeline differentiable in order to directly incorporate it into deep learning pipelines [167, 168, 169]

3.4 Evaluation

Correct and fair comparison of various 6 DoF pose estimation methods is a long-standing problem of the field. Various datasets and pose correctness measures were proposed over the years. Because of that, it is often impossible to fairly asses relative performance of competing methods. A huge leap forward was done with the introduction of BOP challenge [7], that unified both metrics and evaluation protocols for several most commonly used datasets. Another important contribution of the BOP challenge was the introduction of a common synthetic dataset for training of the deep learning models, which allowed for fair comparison of methods data-wise. In this section, we briefly discuss the datasets and pose quality metrics used to evaluate methods proposed in this thesis.

3.4.1 Datasets

A large number of datasets for evaluation and benchmarking of 6 DoF pose esimations have been proposed in the past [15, 8, 6, 12, 170, 171, 172, 16]. These datasets vary in the number of objects, scenes and images. In this thesis, we mostly experimented on Linemod[5] (LM), Linemod-Occlusion (LMO) [8], HomebrewedDB (HBD) [6], YCB-V [16] and TLESS [12]. Sample images from the used datasets are provided in Figure 3.8. We will shortly discuss all of them.



(a) Linemod/Occlusion **(b)** Homebrewed **(c)** YCB-V **(d)** TLESS

Figure 3.8: Sample images from publicly available datasets used for evaluation of the methods proposed in this thesis.

Linemod dataset [5], visualized in Figure 3.8a, is arguably the most classical and used dataset for 6 DoF pose estimation. It consists of 15 objects, of which only 13 are usually used for experiments. Objects are mostly low-textured, which presented a challenge for classical computer vision methods that replied on keypoints. Each object is supplemented with approximately 1200 images where its pose is labeled. Each image has pose labels for only one object. The dataset does not have a well-defined train/test/validation split. Customary, as was done for the first time by BB8 [68], 15% of all the images are used for training and validation while the remaining images are used for evaluation. In case synthetic data is used for training, evaluation is performed on all images. Object of interest are never occluded in the images.

Linemod-Occlusion dataset [8] is the extension of the original Linemod dataset, where all 8 objects of interest were labeled in one of the image sequences of Linemod. This dataset presents a significantly more serious challenge than the standard Linemod due to the presence of occlusions. When training is performed on real data, typically 15% of the labeled images of the Linemod-Occlusion dataset are used together with all the images of the objects of interest from the Linemod dataset, which are not included in Linemod-Occlusion. The BOP challenge includes the dataset as one of its core datasets. For BOP challenge evaluation, the networks are trained on the provided synthetic PBR images and evaluated on a small subset of the entire dataset.

HomebrewedDB [6], visualized in Figure 3.8b, is a modern 6 DoF pose estimation dataset. The dataset was created with emphasises training on synthetic data. It comes with no real training data but with a clear separation of validation of train and test sets, following the best examples of machine learning datasets [173, 174]. The complexity of the scenes varies from relatively simple simple ones with only few objects per scene to heavily-cluttered and occluded. The entire dataset consists of 33 objects, including both toys and industrial parts, and 13 different scenes. Each test scene contains 1000 images with several annotated objects, Each scene comes in two versions: one filmed with PrimeSense Carmine and one filmed with Microsoft Kinect 2. Each validation scene consists of 340 images. In practice, however, only 3 scenes are used for benchmarking of pose estimation methods as a part of the BOP Challenge.

YCB-Video dataset [16], visualized in Figure 3.8c, focus on pose estimation of household object in video sequences. The dataset provides a very large (92) number of scenes and 21 objects. In practice, only 12 scenes are included in the BOP Challenge. The dataset provides a clear separation between the real training images and real testing image. The problem of the dataset is low quality of ground truth poses and low quality of images.

TLESS dataset [12], visualized in Figure 3.8d, is arguably the most challenging dataset used in this thesis. The key feature of the dataset is its industrial objects which are all symmetric, textureless and expose high inter-class similarity. Scenes typically contain a large number of objects, which causes clutter and occlusions. The dataset consists of 30 objects and 20 scenes, all of which are used by the BOP challenge. Each scene is captured by three different cameras: PrimeSense Carmine and Kinect 2 RGB-D cameras and Canon RGB camera. A separate training real set images come with the dataset, consisting of isolated objects on black backgrounds. Object models are provided as precise 3D CAD models without any texture information and as low-quality 3D reconstructions.

3.4.2 Pose Quality Metrics

This subsection discusses commonly used pose quality metrics, namely Average Distance (ADD), Visible Surface Discrepancy (VSD), Maximum Symmetry-Aware Surface Distance (MSSD), Maximum Symmetry-Aware Projection Distance (MSPD). We will use the following notation. $\hat{\mathbf{P}} \in SE(3)$ and $\bar{\mathbf{P}} \in SE\,(3)$ stand for the estimated pose and for the ground truth pose respectively. The model, defined as a set of vertices, is denoted as $M := \left\{v \in \mathbb{R}^3\right\}$

One of the most commonly used metrics, usually referred to as **ADD**, was first formulated in [5]. The metric measures the average distance between model's vertices in a correct pose and model's vertices in a predicted pose. The pose is considered correct if the distance is smaller than some pre-defined threshold which is typically set to 10% of model's diameter. In the case of non-symmetric objects, ADD is formally defined as in Equation 3.8.

$$e_{ADD} = \underset{v \in M}{\operatorname{avg}} \left\| \hat{\mathbf{P}} \cdot v - \bar{\mathbf{P}} \cdot v \right\|_2 \tag{3.8}$$

This formulation is unsuitable for symmetric objects since for them there exist transformations that transform the object, but visually the object stays the same. Nevertheless, the ADD metric can be extended so that it is applicable to symmetric objects as well, as it is demonstrated in Equation 3.9. Instead of tracking each particular model's vertex, it suggests looking for the closest vertices. Given sets M_{pred} and M_{gt} of model's vertices transformed with the predicted and ground truth transformations respectively, for each vertex v in M_{pred} a distance to the closest vertex in M_{gt} is measured. It can be efficiently implemented with KD-trees [175].

$$e_{ADD} = \underset{v_1 \in M}{\operatorname{avg}} \min_{v_2 \in M} \left\| \hat{\mathbf{P}} \cdot v_1 - \bar{\mathbf{P}} \cdot v_2 \right\|_2 \tag{3.9}$$

Visual Surface Discrepancy (VSD)[7, 176] was a next step in developing quality metrics for 6 DoF pose estimation with handling symmetric objects in mind. The core idea that instead of measuring symmetric ADD in the 3D space, we should instead compare depth maps of the object in the ground truth pose and of the same object in the predicted pose. If poses are close, then pixel-wise difference between depth maps is small. If enough pixels have small enough error, then the pose of the object is considered to be correctly estimated. This definition of the error function is innately capable of handling symmetric objects, since their symmetry transformations do not change the rendered depth maps.

$$e_{VSD}\left(\hat{V}, \bar{V}, \tau\right) = \underset{p \in \hat{V} \cup \bar{V}}{\operatorname{avg}} \begin{cases} 0 & \text{if } p \in \hat{V} \cap \bar{V} \wedge |\hat{D}(p) - \bar{D}(p)| < \tau \\ 1 & \text{otherwise} \end{cases} \tag{3.10}$$

In this equation, $\hat{D}$ and $\bar{D}$ are distance maps obtained by rendering the object model M in the estimated pose $\hat{\mathbf{P}}$ and the ground-truth pose $\bar{\mathbf{P}}$ respectively. 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 image I.

The downsides of this error function is slow computation times, which are caused by rendering the object. It has two thresholds: τ and a threshold on e_{VSD} itself which need to be adjusted manually using the expert knowledge. The error fully relies on comparing shapes, which means that even if symmetries can be resolved using color information, VSD is completely oblivious to that. Lastly, VSD is arguably more difficult for interpretation and explanation than ADD.

Maximum Symmetry-Aware Surface Distance (MSSD)[170] is defined as follows:

$$e_{MSSD}\left(\hat{P}, \bar{P}, S_M, M\right) = \min_{\mathbf{S}\in S_M} \max_{x\in M} \left\|\hat{\mathbf{P}} \cdot x - \bar{\mathbf{P}} \cdot \mathbf{S} \cdot x\right\|_2 \tag{3.11}$$

where S_M denotes for a set of symmetry transformations. The definition of MSSD closely resembles the definition of symmetric ADD. The metric is motivated by the fact that the maximum distance 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 symmetric ADD which tend to be dominated by higher-frequency surface parts, the maximum distance is less dependent on the sampling of mesh vertices. Alternatively, it can be views as an example of the Hausdorff distance.

Maximum Symmetry-Aware Projection Distance (MSPD) is an extension of the standard reprojection error first used for evaluation of pose estimation in [11]. The error is defined as follows.

$$e_{MSPD} = \min_{\mathbf{S}\in S_M} \max_{v\in M} \left\|\pi\left(\hat{\mathbf{P}} \cdot v) - \pi(\bar{\mathbf{P}} \cdot \mathbf{S} \cdot v\right)\right\|_2 \tag{3.12}$$

Operator π stands for the projection operator as defined in the pinhole camera model. The error essentially measures how far away in the image plane the locations of where a vertex is projected with the predicted pose $\hat{\mathbf{P}}$ and of the one projected with the ground truth pose $\bar{\mathbf{P}}$.

4 Pose Estimation with Dense Correspondences

In this chapter, we present a novel deep learning method for 3D object detection and 6D pose estimation from RGB images. Our method, named DPOD (Dense Pose Object Detector), estimates dense multi-class 2D-3D correspondence maps between an input image and available 3D models. Given the correspondences, a 6DoF pose is computed via PnP and RANSAC. An additional RGB pose refinement of the initial pose estimates is performed using a custom deep learning-based refinement scheme. Our results and comparison to a vast number of related works demonstrate that a large number of correspondences is beneficial for obtaining high-quality 6D poses both before and after refinement. Unlike other methods that mainly use real data for training and do not train on synthetic renderings, we perform evaluation on both synthetic and real training data demonstrating superior results before and after refinement when compared to all recent detectors. While being precise, the presented approach is still real-time capable.

4.1 Introduction

Object detection has always been an important problem in computer vision and a large body of research has been dedicated to it in the past. This problem, like many other vision problems, witnessed a complete renaissance with the advent of deep learning. Detectors like R-CNN [177], and its follow-ups Fast-RCNN [54], Faster-RCNN [55], Mask-RCNN [64], then YOLO [57] and SSD [56] marked this research field with excellent performance. All these works localize objects of interest in images in terms of tight bounding boxes around them. However, in many applications, e.g., augmented reality, robotics, machine vision, etc., this is not enough and a full 6D pose is necessary. While this problem is easier to solve in depth images, in RGB images it is still quite challenging due to perspective ambiguities and significant appearance changes of the object when seen from different viewpoints.

Recent deep learning-based approaches, such as SSD6D [104], YOLO6D [69], AAE [115], PoseCNN [16] and PVNet [103], are the current top performers for this task in RGB images. Even though they all perform evaluation on LineMOD and OCCLUSION datasets, each of them focuses on different aspects of the 6D pose estimation pipeline. The majority is trained on real data [69, 16, 103, 118] while only SSD6D [104] and AAE [115] are trained on synthetic renderings. Some are presented without refinement, like YOLO6D [69] and PoseCNN [16], while the others perform refinement. The most recent refiners are based on deep learning, e.g., DeepIM [9] that acts on poses from the PoseCNN detector and the refiner of Manhardt et al. [144] that uses SSD6D poses.

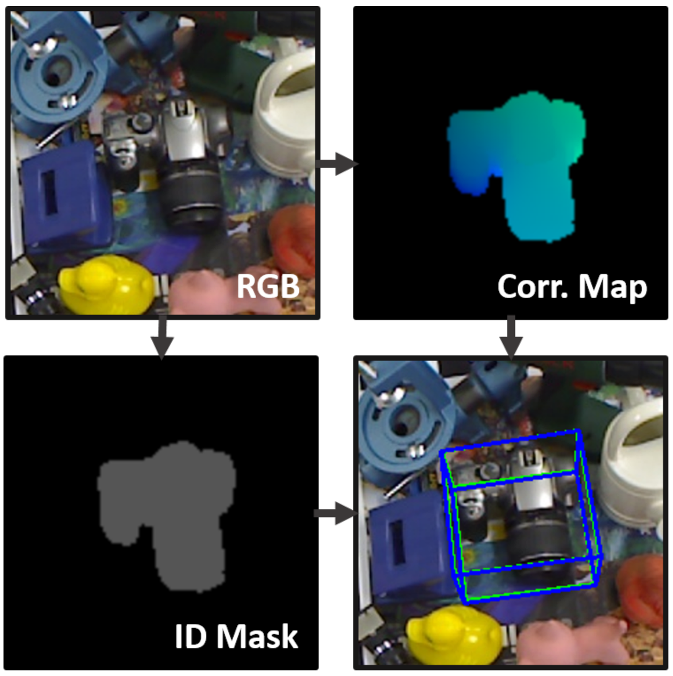


Figure 4.1: Example output of the DPOD method: Given a single RGB image, we regress its ID mask and its 2D-3D correspondences. PnP+RANSAC is then applied to estimate the final pose. The green bounding box shows the ground truth pose, while the blue one corresponds to the estimated pose. The almost perfect overlap of the bounding boxes indicates that estimations are very accurate.

Inspired by the methods of Gueler et al. [76] and Taylor et al. [75], which estimate dense correspondences between the human body model and the humans in the image, we propose a novel 3D object detector and pose estimator that also estimates dense 2D-3D correspondences. Unlike DensePose for humans, which requires a sophisticated annotation tool and enormous annotation efforts, our method is annotation-free and only requires creation of arbitrary UV texture maps of the objects, that we do automatically—mainly by spherical projections. The two key elements of our approach are: the pixel-wise prediction of the multi-class object ID masks and classification of correspondence maps that directly provide a relation between image pixels and 3D model vertices. In this way, we end up with a large number of pixel-wise correspondences, which allow for a much better pose estimation than, for example, 9 regressed virtual points of the object's bounding box as in YOLO6D.

In addition to this, we introduce a deep learning-based pose refinement network that takes initial poses estimated with our DPOD detector and enhances them. The proposed refinement approach builds on the successes of [9, 144], but is shown to be faster, simpler to train, able to be trained both on synthetic and real data, and it outperforms the former solutions in terms of pose quality. We demonstrate that even our poses, which are already of high quality, can be further improved with our refiner.

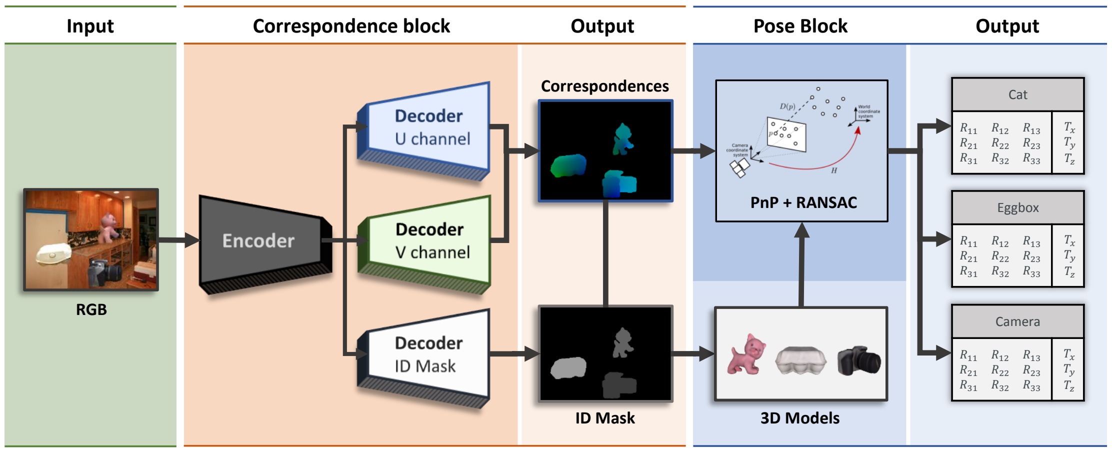


Figure 4.2: Pipeline description: Given an input RGB image, the correspondence block, featuring an encoder-decoder neural network, regresses the object ID mask and the correspondence map. The latter one provides us with explicit 2D-3D correspondences, whereas the ID mask estimates which correspondences should be taken for each detected object. The respective 6D poses are then efficiently computed by the pose block based on PnP+RANSAC.

We experimented by training our detector with only synthetic and only real images. In both cases, our unified method, named DPOD, composed of the dense pose detector and the refiner outperforms other related works. Dense correspondences not only allow for standard PnP and RANSAC to estimate accurate poses without refinement, but also pave the way for a successful pose refinement. For the models trained on real data, one iteration of refinement is enough to outperform all other reported results, even SSD6D with the depth-based ICP refinement.

In the remainder of the chapter, we first review related approaches, then introduce our approach, explaining data preparation, training, architectures and pose refinement. Finally, we present an exhaustive experimental validation and comparison with recent works, where we demonstrate the superiority of our approach.

4.2 Methodology

In this section we first discuss the training data preparation steps, followed by the neural network architecture and loss functions used, as well as the pose estimation step from dense correspondences. Finally, we describe our deep learning model-based pose refiner.

4.2.1 Data Preparation

Most recent RGB-based detectors can be divided in two groups based on the type of data they use for training: synthetic-based and real-based. The first group of methods, e.g., SSD6D [104] and AAE [115], makes use of textured 3D models, usually provided with the public 6D pose detection datasets. The objects are rendered from different viewpoints, producing a synthetic training set. The methods of the second group on the other hand, e.g., BB8 [68], YOLO6D [69], PVNet [103], use the training split of the real dataset. They utilize ground truth poses provided with the dataset and compute object masks to crop the objects from real images producing a training set.

Both types of data generation have their pros and cons. When real images sufficiently covering the object are available, it is more advantageous to use them for training. The reason is that their close resemblance to the actual objects allows for faster convergence and better results. However, training on real images biases the detector to light conditions, poses, scales and occlusions present in the training set, which might lead to problems with generalization in new environments. When, however, no pose annotations are available, which can often be the case since acquiring pose annotations is an expensive process, we are left with 3D models of the objects. With synthetic renderings, one can produce a virtually infinite number of images from different viewpoints. Despite being advantageous in terms of the pose coverage, one has to deal with the domain gap problem severely hindering the performance if no additional data augmentation is applied. Potentially, one can benefit from the advantages of both data types by mixing real and synthetic data in the training set. Therefore, approaches which can be trained on both types of data are desirable. Since our pipeline is not data-specific, we show how to generate the training data for both scenarios.

Synthetic Training Data Generation. Given 3D models of the objects of interest, the first step is to render them from different poses sufficiently covering the object. The poses are sampled from the half-sphere above the object. Additionally, in-plane rotations of the camera around its viewing direction from -30 to 30 degrees are added. Then, for each of the camera poses, an object is rendered on a black background and both RGB and depth channels are stored.

Having the renderings at hand, we use a generated depth map as a mask to define a tight bounding box for each generated rendering. Cropping the image with this bounding box position, we store RGB patches, masks separating them from the background, and the camera poses. At this point, we have everything ready for the online augmentation stage, which is described in the later subsection. This step of data preparation is identical for the detector and for the refinement pipelines.

Real Training Data Generation. In this case, an available dataset with pose annotations is divided into non-overlapping train and test subsets. Here, we follow the protocol defined by BB8 [68] and YOLO6D [69] and use 15% of data for training and the rest 85% for evaluation. Poses are selected such that the relative orientation between them is larger than a certain threshold. This approach guarantees that selected poses cover the object from all sides. For training the detector, objects are cut out from the original image using the provided mask and then stored as patches for the online augmentation stage. Additional in-plane rotations are added to artificially simulate new poses. For training the refinement, objects are left as they are.

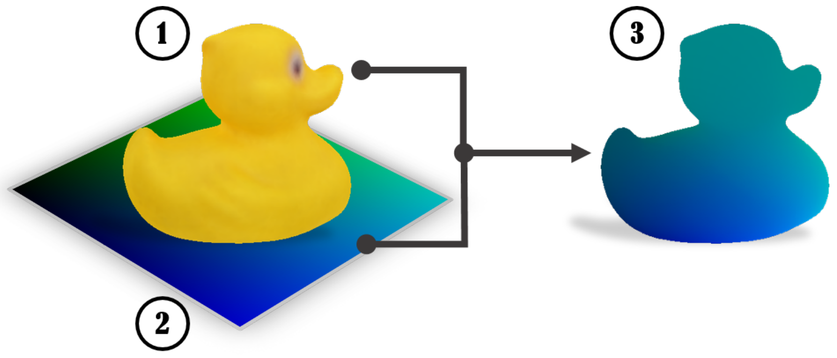


Figure 4.3: Correspondence model: Given a 3D model of interest (1), we apply a 2 channel correspondence texture (2) to it. The resulting correspondence model (3) is then used to generate GT maps and estimate poses.

4.2.1.1 Correspondence Mapping

To be able to learn dense 2D-3D correspondences, each model of the dataset is textured with a correspondence map (see Figure 4.3). A correspondence map is a 2-channel image with values ranging from 0 to 255. Objects are textured using either simple spherical or cylindrical projections. Once textured, we get a bijective mapping between the model's vertices and pixels on the correspondence map. This provides us with easy-to-read 2D-3D correspondences since given the pixel color, we can instantaneously estimate its position on the model surface by selecting the vertex with the same color value. For convenience, we call the copies of the original models textured with the correspondence map *correspondence models*. Given the predicted correspondence map, we estimate the object pose with respect to the camera using the pose estimation block, which is described later. Similar to the synthetic or real data generation steps, we render correspondence models under the same poses as for training data and store correspondence patches for each RGB patch.

4.2.1.2 Online Data Generation and Augmentation

Detection and Pose Estimation. The final stage of data preparation is the online data generation pipeline, which is responsible for providing full-sized RGB images ready for training. Generated patches (real or synthetic) are rendered on top of images from MS COCO dataset [174] producing training images containing multiple objects. It is an important step, which ensures that the detector generalizes to different backgrounds and prevents it from overfitting to backgrounds seen during training. Moreover, it forces the network to learn the model's features needed for pose estimation rather than to learn contextual features which might not be present in images when the scene changes. This step is performed no matter whether the training is being done with synthetic or real patches. We additionally augment the RGB image by random changes in brightness, saturation, and contrast, and by adding Gaussian noise. Moreover, object ID masks and correspondence patches are also rendered on top of the black background in order to generate ground truth correspondence maps. An object ID mask is constructed by assigning a class ID number to each pixel that belongs to the object.

Pose Refinement. In the case of pose refinement, pairs of images containing the object in the current (searched) pose and in the predicted pose are provided to the network. The final stage of data preparation differs considerably depending on the type of data used. In case of synthetic data, images are generated by in-painting objects on random backgrounds in a current pose. A crucial part of the augmentation is to add random light sources for every image. If real images are used for training, no in-painting is performed. In any case, produced images are further augmented as discussed above. Then a random pose is sampled around the current pose simulating the predicted pose from the detector, which will be used as an original guess of the poses to be refined. It is crucial to choose the proper prior distribution from which distorted poses are sampled.

4.3 Dense Object Detection Pipeline

Our inference pipeline is divided into two blocks: the correspondence block and the pose block (see Figure 4.2). In this section, we provide their detailed description.

Correspondence Block. The correspondence block consists of an encoder-decoder convolutional neural network with three decoder heads which regress the ID mask and dense 2D-3D correspondence map from an RGB image of size 320×240×3. The encoder part is based on a 12-layer ResNet-like [51] architecture featuring residual layers that allow for faster convergence. The decoders upsample the feature up to its original size using a stack of bilinear interpolations followed by convolutional layers. However, in principle the proposed method is agnostic to a particular choice of encoder-decoder architecture. Any other backbone architectures can be used without any need to change the conceptual principles of the method. For the ID mask head the output is a $H \times W \times O$ tensor, where H and W are the height and width of the original input image and O is the number of objects in the dataset plus one additional class for background. Similar to the ID mask head, the two correspondence heads regress tensors with the following dimensions $H \times W \times C$, where C stands for the number of unique colors of the correspondence map, i.e., 256. Each channel of the output tensors stores the probability values for the class corresponding to the channel number. Once tensors are regressed, we store them as single channel images where each pixel stores the class with the maximal estimated probability, forming the ID mask, U and V channels of the correspondence image.

Formulating color regression problem as discrete color class classification problem proved to be useful for much faster convergence and for the superior quality of 2D-3D matches. Initial experiments on direct coordinate regression showed very poor results in terms of correspondence quality. The main reason for the problem was the infinite continuous solution space, i.e., $[-1; 1]^3$, where 3 is the number of dimensions and $[-1, 1]$ is the normalized coordinate range of a 3D model. Classification of the discretized 2D correspondences allowed for a huge boost of the output quality by dramatically decreasing the output space (now 256^2, where 256 is the size of a single UV map dimension). Moreover, this parametrization also ensures that 3D points of the predicted correspondences always lie on the object surface.

The network parameters are optimized subject to the composite loss function:

$$\mathcal{L} = \alpha\mathcal{L}_m + \beta\mathcal{L}_u + \gamma\mathcal{L}_v, \tag{4.1}$$

where $\mathcal{L}_m$ is the mask loss, and $\mathcal{L}_u$ and $\mathcal{L}_v$ are the losses responsible for the quality of the U and V channels of the correspondence image. α, β, and γ are weight factors set to 1 in our case. Both $\mathcal{L}_u$ and $\mathcal{L}_v$ losses are defined as multi-class cross-entropy functions, whereas $\mathcal{L}_m$ uses the weighted version of it.

Pose Block. The pose block is responsible for the pose prediction. Given the estimated ID mask, we can observe which objects were detected in the image and their 2D locations, whereas the correspondence map maps each 2D point to a coordinate on an actual 3D model. The 6D pose is then estimated using the Perspective-n-Point (PnP) [178] pose estimation method that estimates the camera pose given correspondences and intrinsic parameters of the camera. Since we get a large set of correspondences for each model, RANSAC is used in conjunction with PnP to make camera pose prediction more robust to possible outliers. For the results presented in the evaluation section, for each pose we run 150 RANSAC iterations with the reprojection error threshold set to 1.

4.4 Deep model-based pose refinement

The proposed pose refiner is a natural extension of refiners presented in [144, 9] and relies on the strengths of both approaches. Similar to [144, 104, 179] we exploit an idea of using a network already pre-trained on ImageNet as a backbone architecture. Analogous to the detector, we used a ResNet-based architecture. Similar to [9], our loss function for pose estimation is the ADD measure with a more robust L_1 norm:

$$m = \underset{\mathbf{x}\in\mathcal{M}_s}{\operatorname{avg}} \left\| (\mathbf{R}\mathbf{x} + \mathbf{t}) - (\hat{\mathbf{R}}\mathbf{x} + \hat{\mathbf{t}}) \right\|_1, \tag{4.2}$$

representing the vertex to vertex distance between the object in a ground truth pose and predicted pose. $\mathbf{R}$, $\mathbf{t}$ denote the ground truth pose rotation and translation, whereas $\hat{\mathbf{R}}$ and $\hat{\mathbf{t}}$ denote the predicted transformation; $\mathcal{M}_s$ is a set of points sampled from the CAD model. Points are resampled at every iteration. The number of sampled points was limited to ten thousand in order to ensure the efficiency of training iterations and reasonable memory consumption.

In Figure 4.4 we show a schematic representation of the refiner. In order to be able to benefit from the network weights pretrained on ImageNet, the network has two parallel input branches, each composed of the first five ResNet layers. These layers are initialized from the pre-trained network. One branch receives an input image patch (E_{11}), while

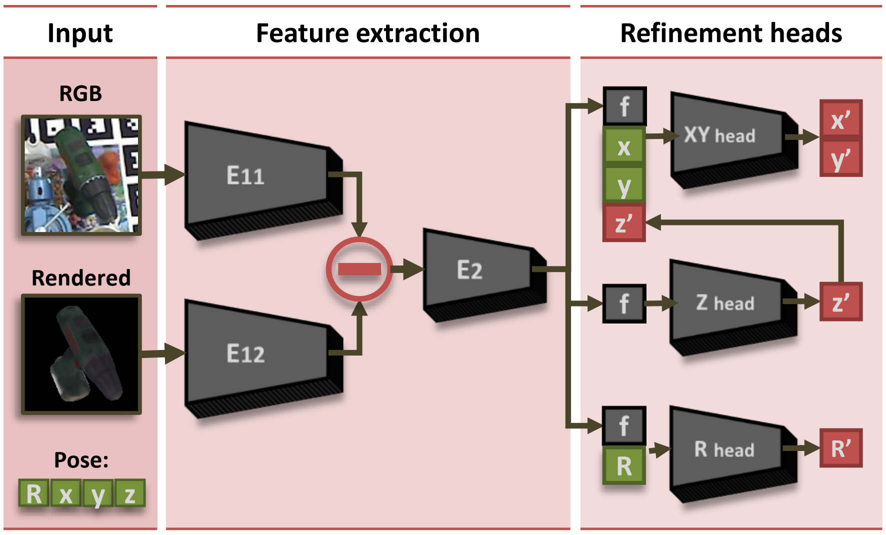


Figure 4.4: Refinement architecture: The network predicts a refined pose given an initial pose proposal. Crops of the real image and the rendering are fed into two parallel branches. The difference of the computed feature tensors is used to estimate the refined pose.

the other (E_{12}) one extracts features from the rendering of the object in the predicted pose. Then features $\mathbf{f}_r$ and $\mathbf{f}_s$ from these two networks are subtracted and fed into the next ResNet block (E_2) producing the feature vector $\mathbf{f}$. If the refinement is trained on synthetic data, it is essential to keep the first five layers unchanged and use them as the feature extractor as was shown in [56, 179, 144]. Freezing the branch that extracts features from object renderings is unnecessary as it always operates on synthetic data. The network ends with three separate output heads: one for regressing the rotation, one for regressing the translation in X and Y directions, and one for regressing the translation in Z direction. We opted for three separate heads as the scale of their outputs is different. Each head is implemented as two fully connected layers.

Rotation is always represented in the object coordinate system, which ensures that identically looking objects have the same rotation and that the network does not have to learn a more complicated transformation which arises if the world coordinate system is used. The first layer of the rotation regression head takes the feature vector $\mathbf{f}$ produced by ResNet and adds four values, which are the quaternion representing an initial rotation. The second layer takes the output of the previous one, stacks with the initial quaternion and outputs the final rotation.

The head responsible for the regression of X and Y translations operates in the coordinate system of the image rather than in the full 3D space, which significantly restricts the space of possible solutions. Similar to the rotation head, the XY regression head takes the initial 2D location of the object as input and refines it. Additionally, it takes the refined prediction of Z translation.

Weights of the fully connected layers are initialized in such a way that for the 0th iteration the network just outputs the input pose, and then during training learns how to refine those values. That significantly increases stability and speed of the training procedure as the network produces meaningful results from the very start. When a patch is cropped from the image, a padding of 5 pixels is used. This padding corresponds to approximately 10mm error in X or Y direction if the camera is 1m away from the object, which is a rather large error and very uncommon for our detector. Crops are resized to be 255×255 pixels. Increasing image size did not result in increased pose quality but slowed down the inference. As a result, of those changes we were able to train the network on one GPU with a significantly larger batch size, which made additional loss functions unnecessary.

4.5 Training Details

Our pipeline is implemented using the Pytorch deep learning framework. All the experiments were conducted on an Intel Core i7-6900K CPU 3.20GHz with NVIDIA TITAN X (Pascal) GPU. To train our method, we used the ADAM solver with a constant learning rate of 3×10^{-4} and weight decay of 3×10^{-5}.

When training on synthetic data, the problem of domain adaptation becomes one of the main challenges. Training the network without any prior parameter initialization makes it impossible to generalize to the real data. The easy solution to this problem was proposed in several works, including [179, 144], where they freeze the first layers of the network trained on a large dataset of real images, e.g., ImageNet [173] or MS COCO [174], for the object classification task. The common observation that the authors conclude is that these layers, learning low-level features, very quickly overfit to the perfect object renderings. We follow this setup, and freeze the first five layers of our encoder initialized with the weights of the same network pretrained on ImageNet. Last but not least, we found it crucial for the performance of the detector to use various light sources during the rendering of synthetic views to account for changing light conditions and shadows in the real data.

4.6 Implementation Details

The refinement network utilizes the same backbone architecture. It is a standard ResNet-like (ResNet18 in PyTorch) model with a reduced number of layers and pooling operations in comparison to the original ResNet first presented in [51]. Upsampling is implemented as bilinear interpolation rather than deconvolution in order to decrease the number of parameters and the required amount of computations. Each upsampling is followed by the concatenatination of the output feature map with the feature map from the previous level, and one convolutional layer. When the detector is trained on synthetic data, the first five layers are frozen in order to prevent overfitting to peculiarities of the rendered data. The architecture of the refinement network follows the same architectural idea, except for the absence of upsampling and presence of fully-connected layers at the

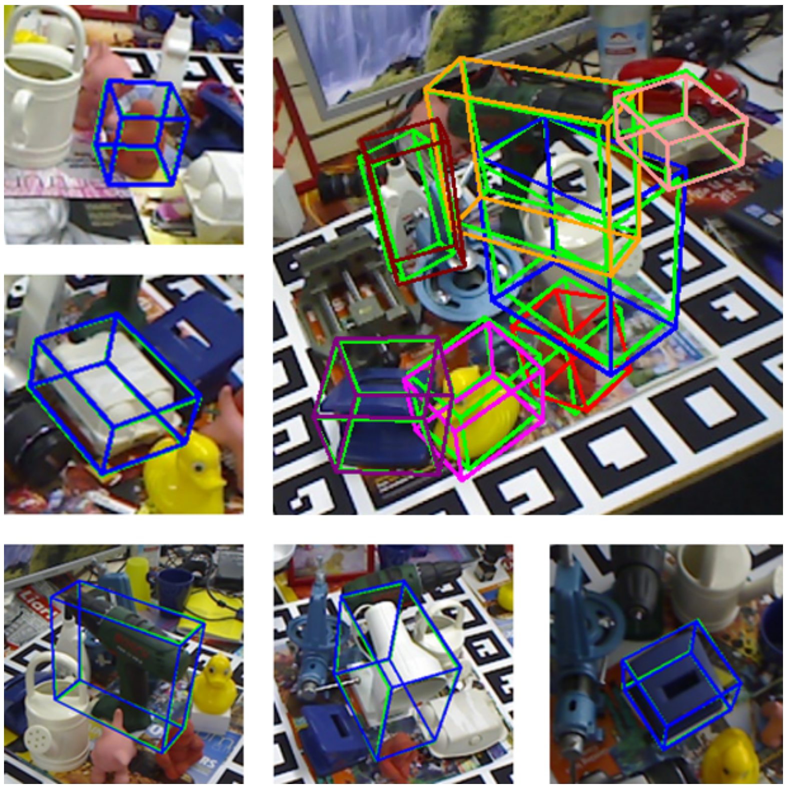


Figure 4.5: Qualitative results: Poses predicted with the proposed approach on (a) the LineMOD dataset and (b) the OCCLUSION dataset. Green bounding boxes correspond to ground truth poses, bounding boxes of other colors to predicted poses. For both datasets predicted poses are very close to correct poses.

end. Again, the first five layers are used in siamese-like fashion for extracting features from image crops and renderings.

4.7 Evaluation

In this section we evaluate our algorithm in terms of its pose and detection performance, as well as its runtime, and compare it with the state of the art RGB detector solutions.

4.7.1 Datasets

All experiments were conducted on LineMOD [5] and OCCLUSION [8] datasets, as they are the standard datasets for evaluation of object detection and pose estimation methods. The LineMOD dataset consists of 13 sequences, each containing ground truth poses for a single object of interest in a cluttered environment. CAD models for all the objects are provided as well. The OCCLUSION dataset is an extension of LineMOD, suitable for testing how well detectors can deal with occlusions. Although it comprises only one sequence, all visible objects from the LineMOD dataset are supplied with their poses.

4.7.2 Evaluation Metrics

We evaluate the quality of 6DoF pose estimation following the procedure suggested at SSD6D [104] also used in other papers. Analogously to other related papers [69, 104, 103, 16], we measure the accuracy of pose estimation using the *ADD score* [5]. ADD is defined as an average Euclidean distance between model vertices transformed with the predicted and the ground truth pose. More formally it is defined as follows:

$$m = \operatorname*{avg}_{\mathbf{x} \in \mathcal{M}} \left\| (\mathbf{R}\mathbf{x} + \mathbf{t}) - (\hat{\mathbf{R}}\mathbf{x} + \hat{\mathbf{t}}) \right\|_2 , \tag{4.3}$$

where $\mathcal{M}$ is a set of vertices of a particular model, $\mathbf{R}$ and $\mathbf{t}$ are the rotation and translation of a ground truth transformation whereas $\hat{\mathbf{R}}$ and $\hat{\mathbf{t}}$ correspond to those of an estimated transformation. The ADD metric can be extended in order to handle symmetric objects as in [5]:

$$m = \operatorname*{avg}_{\mathbf{x}_2 \in \mathcal{M}} \min_{\mathbf{x}_1 \in \mathcal{M}} \left\| (\mathbf{R}\mathbf{x}_1 + \mathbf{t}) - (\hat{\mathbf{R}}\mathbf{x}_2 + \hat{\mathbf{t}}) \right\|_2 \tag{4.4}$$

Instead of measuring distance from a predicted location of each particular model's vertex to its ground truth location, it suggests to take the closest vertex of the model transformed with the ground truth transformation.

Conventionally, a pose is considered correct if ADD is smaller than the 10% of the model's diameter. The accuracy of pose estimation is reported as the percentage of correctly estimated poses.

4.7.3 Single Object Pose Estimation

Results of the pose estimation experiments on the LineMOD dataset are reported in Table 4.1. We separately compared our method trained either on real data or on synthetic data. The table provides the comparison of deep learning-based refinement pipelines as well. The left-hand side of the table reports the accuracy of pose estimation as percentages of poses which are correct according to the ADD measure for the training done on synthetic data. If no refinement is used, our approach outperforms all other approaches by a significant margin on the majority of the objects. Moreover, the average percentage of correctly estimated poses (50%) is significantly higher than 28.65% of the second best approach. The accuracy gap is more prominent on small objects such as the ape and duck. The availability of a large number of 2D-3D correspondences ensures that the performance of our method is 5 times better than SSD6D's and almost 2 times better than AAE's. If deep learning-based refinement is used, we significantly outperform [144] with 66.43% of correct poses against 34.1%.

If trained on real data, our method is the second best after [103]. The right-hand side of Table 4.1 compares the proposed approach to the previous deep learning-based ones.

Table 4.1: Pose estimation performance: Comparison of our approach to the other RGB detectors on the LineMOD dataset. The table reports the percentages of correctly estimated poses w.r.t. the ADD score. Among the methods trained on synthetic data, our method shows the best results significantly surpassing the former state-of-the-art. The variant of our method trained on real data again demonstrates outstanding performance outperforming most of the competitors. Moreover, our new refinement pipeline improves the estimated poses even further and shows the best overall results.

Train data	**Synthetic**			**+ Refinement**		**Real**				**+ Refinement**	
Object	**SSD6D** [104]	**AAE** [115]	**Ours**	**SSD6D** [144]	**Ours**	**YOLO6D** [69]	**PoseCNN** [16]	**PVNet** [103]	**Ours**	**DeepIM** [9]	**Ours**
Ape	2.6	3.96	**37.22**	-	55.23	21.62	-	43.62	**53.28**	77.0	**87.73**
Benchvise	15.1	20.92	**66.76**	-	72.69	81.80	-	**99.90**	95.34	97.5	**98.45**
Cam	6.1	**30.47**	24.22	-	34.76	36.57	-	86.86	**90.36**	93.5	**96.07**
Can	27.3	35.87	**52.57**	-	83.59	68.80	-	**95.47**	94.10	96.5	**99.71**
Cat	9.3	17.90	**32.36**	-	65.10	41.82	-	**79.34**	60.38	82.1	**94.71**
Driller	12.0	23.99	**66.60**	-	73.32	63.51	-	96.43	**97.72**	95.0	**98.80**
Duck	1.3	4.86	**26.12**	-	50.04	27.23	-	52.58	**66.01**	77.7	**86.29**
Eggbox	2.8	**81.01**	73.35	-	89.05	69.58	-	99.15	**99.72**	97.1	**99.91**
Glue	3.4	45.49	**74.96**	-	84.37	80.02	-	**95.66**	93.83	**99.4**	96.82
Holepuncher	3.1	17.60	**24.50**	-	35.35	42.63	-	**81.92**	65.83	52.8	**86.87**
Iron	14.6	32.03	**85.02**	-	98.78	74.97	-	98.88	**99.80**	98.3	**100**
Lamp	11.4	**60.47**	57.26	-	74.27	71.11	-	**99.33**	88.11	**97.5**	96.84
Phone	9.7	**33.79**	29.08	-	46.98	47.74	-	**92.41**	74.24	87.7	**94.69**
Mean	9.1	28.65	**50**	34.1	**66.43**	55.95	62.7	**86.27**	82.98	88.6	**95.15**

If no refinement is used, the proposed approach outperforms PoseCNN and YOLO6D by a significant margin, while performing on par with PVNet on most of the objects. On average, we are better than PoseCNN by 31%, YOLO6D by 23.57%. Again, our approach uses RGB data exclusively and does not rely on depth data. Figure 4.5 provides a visual comparison of ground truth poses versus predicted poses. Poses are visualized as projections of 3D bounding boxes of models in given poses on top of a test image. In comparison to deep learning-based refinement of [9], we perform on average better by 6.55% reaching 95.15% of correct poses. When DeepIM was applied to the poses predicted by the proposed approach, ADD improved to 91.8% which is better than the original 88.6% reported in their paper, but still worse than the result of our refiner.

In conclusion, the proposed detector achieves state-of-the-art results surpassing other detectors by a large margin on synthetic data and performs either much better or comparable to the other detectors on real data. The proposed refinement clearly outperforms all the competitors both on real and synthetic data. Pose quality varies from object to object, but in general poses are significantly better for larger objects since there are more 2D-3D correspondences available. On the other hand, simplicity of the proposed approach also makes it quick. On average our detector performs at 33 FPS. The runtime can be adjusted by changing the number of RANSAC iterations, as it is the bottleneck of the pipeline. One iteration of the refinement takes 5ms, excluding the rendering time, which heavily depends on the renderer used. Two refinement iterations suffice for synthetic data, one iteration—for real data.

Table 4.2: Detection performance for multiple objects: Comparison of the state-of-the-art mean average precision (mAP) scores on the OCCLUSION dataset.

Method	**SSD6D** [104]	**YOLO6D** [69]	**Brachmann** [11]	**Ours**
mAP	0.38	0.48	0.51	0.48

4.7.4 Multiple Object Pose Estimation

Performance evaluation of the proposed detector in cases when the number of objects to detect increases and when severe occlusions are present was conducted on the OCCLUSION dataset [8]. Accuracy of object detection on the OCCLUSION dataset is conventionally reported in terms of mean average precision (mAP). The confidence score is computed based on the RANSAC inlier proportion as confidence, rendering the final score of 0.48, which is comparable the best result on this dataset (see Table 4.2). Table 4.3 demonstrates ADD scores for various detectors on the OCCLUSION dataset. Before the refinement, the proposed detector shows very competitive results in comparison to other detectors. After the refinement, the proposed approach performs substantially better and achieves the best results.

Table 4.3: Pose estimation for multiple objects: Comparison of our approach on real data to the other RGB detectors on the OCCLUSSION dataset. The table reports percentages of correctly estimated poses w.r.t. the ADD score.

Method	**YOLO6D** [69]	**PoseCNN** [16]	**SSD6D + Ref** [104]	**HMap** [180]	**PVNet** [103]	**Ours**	**Ours +Ref**
Mean	6.42	24.9	27.5	30.4	40.77	32.79	**47.25**

4.8 Additional Experiments

4.9 RANSAC Iterations

The number of RANSAC iterations crucially influences the quality of predicted poses. We ended up using 150 iterations as it yielded the best trade off between quality and runtime. The larger amount of iterations generally did not improve the results significantly, but resulted in longer execution times (see Table 4.4). Additionally, the ADD scores after one iteration of the proposed refinement are provided. They show that even 25 iterations of RANSAC are enough to beat the state-of-the-art results if the refinement is used. More iterations of RANSAC do not result in the considerable increase of pose quality.

4.9.1 Runtime analysis

In Table 4.6 we provide the runtimes of the proposed approach for all models of the LineMOD dataset. The total runtime consists of the time needed for PnP and approxi-

Table 4.4: RANSAC iterations test: The effect of the number of RANSAC iterations on the overall ADD score.

RANSAC #	**5**	**25**	**50**	**100**	**150**	**200**	**250**	**350**	**500**
ADD w/o ref	59.15	76.95	80.15	82.12	82.98	83.44	83.79	84.33	84.66
ADD w/ ref	80.45	92.59	93.88	94.79	95.15	95.31	95.39	95.38	95.39
RANSAC ms	2	6	10	17	23	28	33	42	54

mately 13 ms for all the auxiliary tasks: the network's forward pass, post-processing of predicted segmentation, and computation of 2D-3D correspondences. Table 4.5 provides comparison of the runtime of our detector with all the main competitors. All the experiments were conducted on an Intel Core i7-6900K CPU 3.20GHz with NVIDIA TITAN X (Pascal) GPU.

Table 4.5: Runtime comparison: Time-efficiency of our approach with respect to the other state-of-the-art approaches.

Method	**Frames per second**	**Refinement**
AAE [115]	4	200 ms/object
SSD6D [104]	10	24 ms/object
PVNet [103]	25	-
Ours	33	5 ms/object
YOLO6D [69]	50	-

4.9.2 Refinement

DeepIM [9] presents an iterative refinement routine that takes an initial pose estimate from any external detector and iteratively improves it. An additional per-model evaluation is provided (see Table 4.7) to have a fair comparison of DeepIM with our pose refinement. It compares the following ADD scores: 1) ADD reported in the original DeepIM paper [9], which used PoseCNN [16] to predict initial poses, 2) ADD if DeepIM is applied to poses predicted by our detector, 3) ADD if poses predicted by the proposed detector are refined with the proposed refinement. It is important to mention that two iterations of DeepIM were made, as was suggested in the paper. The proposed refinement was run only for one iteration. The table clearly shows that better initial pose hypotheses allow for better results after refinement. It is also clear that our refinement clearly outperforms DeepIM on most of the objects, while performing only insignificantly worse on others.

Table 4.6: Runtime analysis: Runtime of the proposed approach for all models of the LineMOD dataset.

Model	PnP + RANSAC (ms)	Total (ms)	FPS
Ape	7	20	50
Benchvise	40	51	20
Cam	35	49	20
Can	30	44	23
Cat	20	33	30
Driller	26	40	25
Duck	4	16	63
Eggbox	9	23	43
Glue	5	17	59
Holepuncher	20	31	32
Iron	34	48	21
Lamp	40	54	19
Phone	31	45	22
Average	23	36	33

Table 4.7: Comparison of deep learning-based refinement methods: Our refinement approach shows the overall best ADD score with respect to the latest state-of-the art method DeepIM [9].

Method/Object	Ape	Bench.	Cam	Can	Cat	Dril.	Duck	Eggb.	Gl.	Hol.	Iron	Lamp	Ph.	Avg.
PoseCNN [16] + DeepIM [9]	77.0	97.5	93.5	96.5	82.1	95.0	77.7	97.1	99.4	52.8	98.3	97.5	87.7	88.6
Ours + DeepIM [9]	78.70	98.43	**97.75**	97.57	85.16	91.55	80.24	99.68	**99.48**	75.66	99.74	**98.20**	91.38	91.81
Ours + Our ref.	**87.73**	**98.45**	96.07	**99.71**	**94.71**	**98.8**	**86.29**	**99.91**	96.82	**86.87**	**100**	96.84	**94.69**	**95.15**

4.9.3 Correspondence Quality

In this section, we demonstrate the quality of the output correspondences. Namely, each classified correspondence point is mapped to 3D and compared to the ground truth 3D point. The ground truth 3D points are obtained in exactly the same way as predicted points, i.e., by matching a UV map rendered in the ground truth pose to model's vertices.

The results per object are shown in Table 4.8. The table reports the quality of correspondences separately for real and synthetic data. For each model, mean absolute error, median absolute error, and standard deviation of absolute errors are reported in millimeters. Relatively large mean error is explained by outliers, some of which can be quite significant. Therefore, median is a better measure due to its robustness to outliers. The table shows that the median error is consistent across all the models. Additionally, it demonstrates that the median error for the detector trained on real data is noticeably lower than for the detector trained on synthetic data. This explains the superior performance of training on real data.

Figure 4.6 provides a visual comparison of predicted and ground truth UV maps and heat maps, which demonstrate where imprecisions take place. One can see that most

Table 4.8: Quantative correspondence quality: Correspondence quality for real and synthetic data estimated in terms of mean and median absolute errors, and standard deviation.

	Real Data			**Synt Data**		
Model	**Mean**	**Median**	**Std**	**Mean**	**Median**	**Std**
Ape	10.05	4.58	14.60	11.46	5.74	15.26
Benc.	10.36	4.70	19.29	15.92	6.99	25.71
Cam	6.57	4.58	10.11	13.31	7.23	20.23
Can	8.19	4.03	13.46	11.97	5.10	18.72
Cat	8.60	4.77	12.22	9.87	5.42	13.99
Driller	8.52	4.78	17.78	18.14	6.80	36.06
Duck	5.93	3.98	8.72	7.63	4.99	10.41
Eggbox	6.00	4.26	10.23	42.39	9.40	48.07
Glue	7.82	4.26	13.73	17.12	8.11	23.19
Holep.	8.25	4.87	13.30	11.28	6.81	16.04
Iron	7.18	4.51	12.31	11.06	6.89	17.34
Lamp	11.64	4.31	24.80	18.60	8.58	30.85
Phone	6.09	2.84	12.94	9.52	4.38	18.31

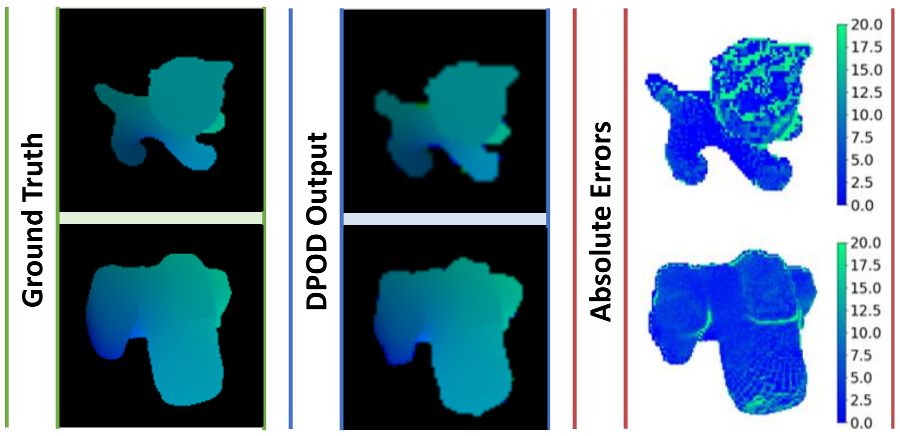


Figure 4.6: Qualitative correspondence quality: Comparison of ground truth (left), predicted (center) UV maps and heat maps (right) of absolute errors.

imprecisions are concentrated on the outer boundaries of the object and, for objects with more complex geometry, on the edges of their structural elements, i.e. in places where rapid correspondence value changes occur.

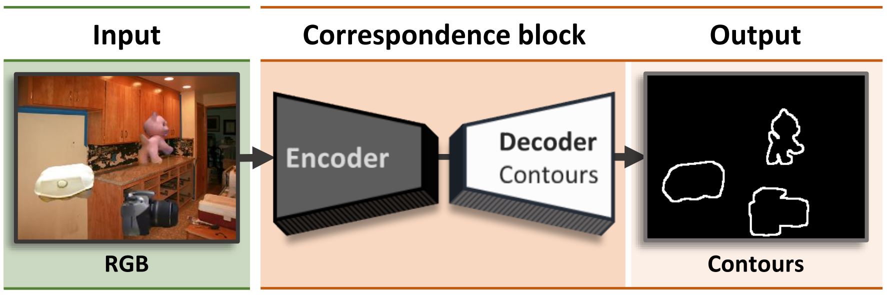


Figure 4.7: Contour regression: Additional contour regression head for multiple instance detection.

4.9.4 Multiple Instance Detection

Our detector also works when multiple instances of the same object are presented, because we parse through the regions of the output mask when the network forward pass is complete. The only limitation comes into play when several objects of the same class overlap and form a single region. In this case, only one pose will be estimated instead of two.

4.9.5 UVW Mapping

While being computationally efficient, the UV mapping has a number of drawbacks. In the majority of cases, a simple spherical projection is sufficient to achieve a satisfactory quality of the mapping. However, certain cases can require a different treatment to minimize the stretching effect where one color can cover several model vertices due to discretization. This is especially a big problem for more complicated geometries, which in some cases might require a selection of another projection type or even a manual UV mapping for reaching the best results.

A straightforward solution to this is the UVW mapping based on normalized 3D coordinates of the model. Instead of 2-channel UV maps, we then have 3-channel UVW maps that are again discretized to the range [0, 255]. The only algorithmic adjustment that has to be done is an additional W-channel classification head. While decreasing the memory efficiency and increasing a computational complexity of the network (due to a higher-dimensional solution space, i.e., 256^3 instead of 256^2 in case of UV mapping), it has an advantage of providing better quality correspondences (especially for objects with complex geometries) and of being fully automatic (see Figure 4.8 for visual comparison).

Our additional experimental ablations have shown an almost identical performance on the LineMOD and OCCLUSION datasets, but slightly higher execution times and memory requirements. Nevertheless, despite the increased complexity, we believe that this extension would prove itself useful in many real-world applications.

To overcome this, an additional contour regression head can be added to the correspondence block (see Figure 4.7). Once regressed, the output contours are simply multiplied

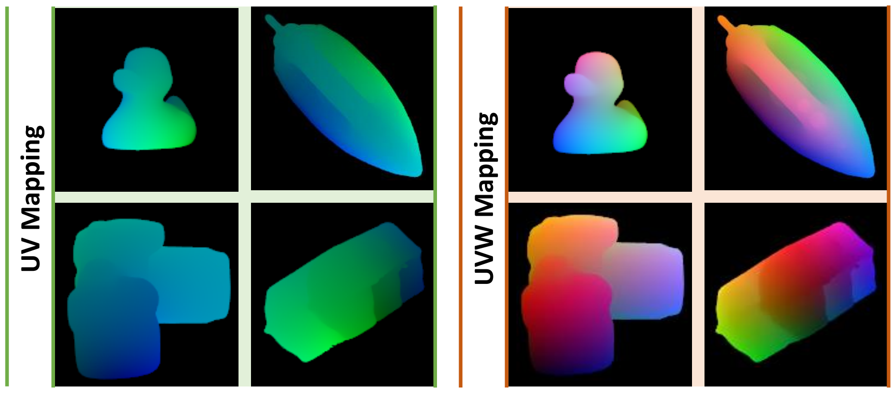


Figure 4.8: UVW mapping: Visual comparison between UV and UVW mappings.

with the ID mask forming different regions, which, as a result, allows to distinguish between different regions of the same class.

4.10 Conclusion

In this chapter, we proposed the Dense Pose Object Detector (DPOD) method that regresses multi-class object masks and dense 2D-3D correspondences between image pixels and corresponding 3D models. Unlike the best performing methods that regress projections of the object's bounding boxes [68, 69] or formulate pose estimation as a discrete pose classification problem [104], dense correspondences computed by our method allow for more robust and accurate 6D pose estimation. We demonstrated that for both, real and synthetic training data, our detector outperforms other related works, such as [69, 16], by a large margin and performs similarly to [103]. The proposed pose refinement approach also performs very well and allows for achieving a pose accuracy that surpasses all other related deep learning-based pose refinement approaches, while having a simpler and more lightweight backbone architecture.

5 Multi-View Pose Refinement with Dense Correspondences

This chapter introduces a novel multi-view 6 DoF object pose refinement approach focusing on improving methods trained on synthetic data. It is based on the DPOD detector, which produces dense 2D-3D correspondences between the model vertices and the image pixels in each frame. We have opted for the use of multiple frames with known relative camera transformations, as it allows introduction of geometrical constraints via an interpretable ICP-like loss function. The loss function is implemented with a differentiable renderer and is optimized iteratively. We also demonstrate that a full detection and refinement pipeline, which is trained solely on synthetic data, can be used for auto-labeling real data. We perform quantitative evaluation on LineMOD, Occlusion, Homebrewed and YCB-V datasets and report excellent performance in comparison to the state-of-the-art methods trained on the synthetic and real data. We demonstrate empirically that our approach requires only a few frames and is robust to close camera locations and noise in extrinsic camera calibration, making its practical usage easier and more ubiquitous.

5.1 Introduction

Object detection and 6D pose estimation in RGB images are among the most fundamental problems in computer vision, with applications encompassing autonomous driving, augmented reality and robotics. A large body of work has already been presented, but recent advances in deep learning have opened up new horizons for RGB-based algorithms, which have now started to dominate the field. However, precise 6 DoF pose estimation, required, in autonomous robotic grasping systems, for example, still remains a challenging problem. This is mainly due to the perspective ambiguity, lightning changes, clutter and occlusions. Current industrial implementations [15] are not based on deep learning and rely on depth data for increased pose accuracy. Moreover, these approaches use 3D models directly, while deep learning methods still struggle with the domain gap when trained on synthetic data rendered from 3D CAD models. Recently, CosyPose [10], a deep learning method that operates on RGB images, assumed the lead over traditional methods in the BOP challenge[7], especially when multiple images are used for pose refinement.

The task of pose refinement has also been addressed by deep learning approaches. Methods like [100, 144, 9] trained CNN networks on pairs of images, the aim being to learn to predict the pose offset between the predicted object pose and the observed detected object. There, 3D models were rendered in the predicted pose and real images

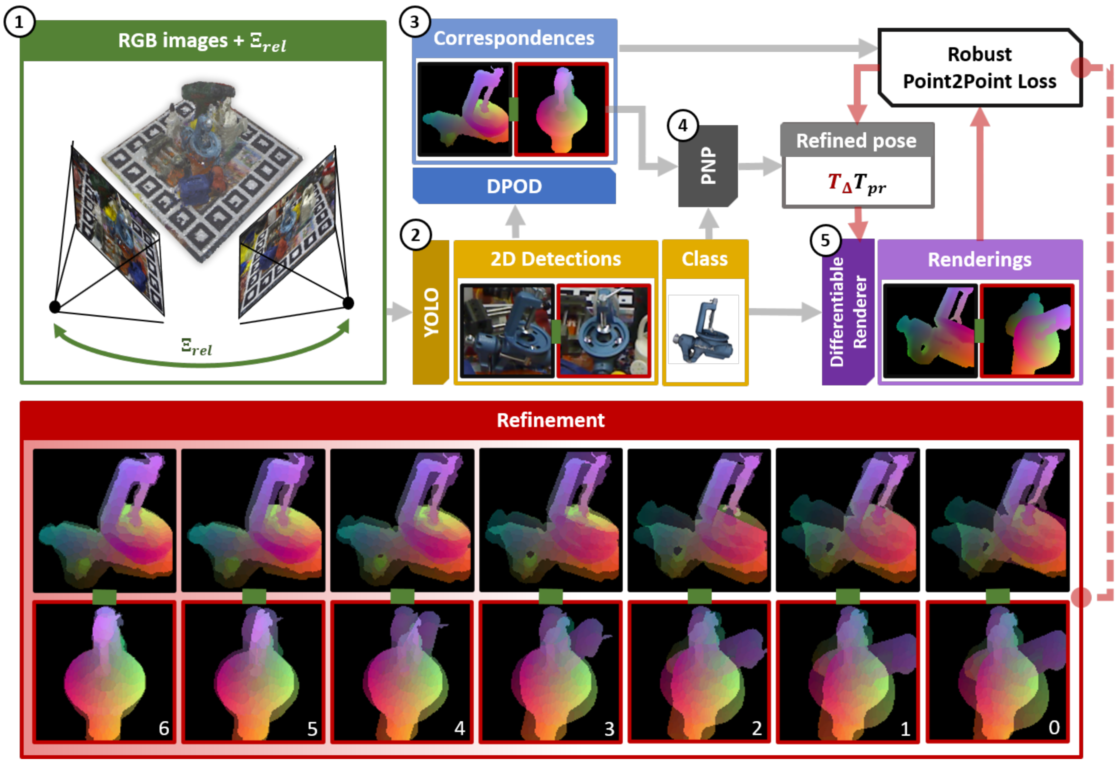


Figure 5.1: Multi-view inference and pose optimization. 1) Inputs to the algorithm are an unordered set of images and corresponding relative camera transformations. 2) YOLO is applied to each image separately to detect the object of interest in each of them. 3) Dense correspondences are predicted with the DPOD network. 4) Rough object pose in the reference frame is estimated using PnP and RANSAC using the predicted 2D-3D dense correspondences. 5) The final pose is iteratively refined using the multi-view optimization based on differentiable rendering.

were given as patches with the detected object. The main disadvantage of these refiners is that the are dependent on the correct pose error priors used during training coupled with the necessity to re-train them for each new object in order to obtain high-quality results.

We address the aforementioned problems in this paper by introducing a multi-view refinement procedure. We first train the DPOD detector [100] on synthetic data to overcome the dataset bias and the lack of real training data. We then exploit geometric multi-view constraints during refinement to cope with perspective ambiguities and occlusions in monocular RGB images that cause their sub-par performance. Compared to existing deep learning refiners [100, 144, 9], the proposed approach has the following advantages: 1) it does not require training of the refiner itself; 2) it is not object-specific; 3) it can be applied to arbitrary many images without the need for any modification; 4) it has explicit geometric constraints and, thus, an explicit objective function which is optimized during the refinement.

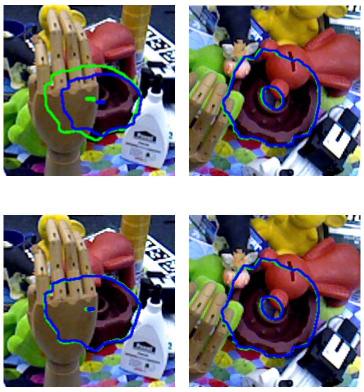
The proposed pose refinement procedure is based on differentiable renderering. It uses multiple views to add relative camera poses as constraints to the pose optimization procedure. It is done by comparing predicted and rendered dense correspondences in each frame and then transmitting the error back though the differentiable renderer to update the pose. The proposed loss function allows varying numbers of frames to be used without any changes to the optimization procedure. The loss formulation also does not impose any restrictions on where in the 3D world the cameras are placed and whether or not views from different cameras overlap as long as the object is visible in the images. We assume the availability of relative camera poses. In practice, they can be easily obtained by a number of various methods, such as placing the object on the markerboard and either using an actual multi-camera system or using a single camera but moving the markerboard. In the scenario of robotic grasping, a camera can be mounted on the robotic arm to enable observation of the object from several viewpoints.

We evaluate this approach on LineMOD [5], Occlusion [8], Homebrewed [6] and YCB-V [16] datasets and report performance that is superior to any related method trained on synthetic data and similar to or better than methods which use real training data and post-refinement. Our experiments show that our approach can robustly perform multi-view pose refinement even when relative poses are imprecise. Our results also demonstrate that the proposed refinement remains effective even in degenerate cases, when cameras are in close proximity to each other.

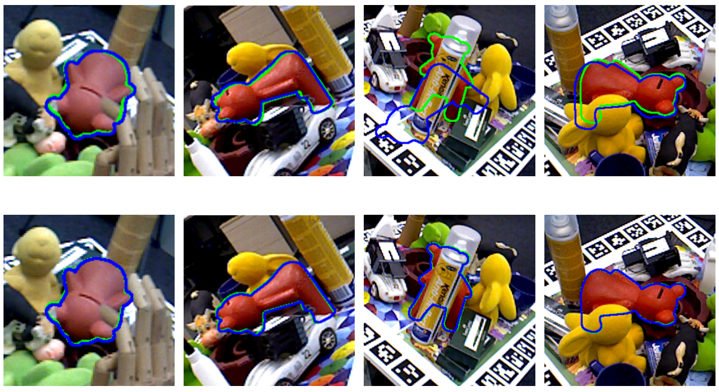
Further, we show that our framework can be used for the task of auto-labeling real data as in [181], thus removing the need for manual pose labeling. The networks are first trained on synthetic data and then used to label the real images. This procedure enables a considerable reduction in the time and effort needed to annotate the data. Our multi-view refinement pipeline enables us to automatically produce high quality pose annotations for these real images and re-train the detector on them.

5.2 Proposed Method

The complete inference pipeline is shown in Figure 5.1. The proposed method is divided into the following steps. Step 1: takes a set of images together with their known intrinsic parameters and relative transformations between cameras; Step 2: The objects of interest in each image are detected separately; Step 3: The per-object per-pixel 2D-3D correspondences are predicted independently for each detected object; Step 4: The object 6 DoF pose is estimated with EPnP [1] and RANSAC using the predicted 2D-3D correspondences; Step 5: The initial rough pose is iteratively refined to find a pose which better aligns with predicted dense correspondences in all the frames. This is done by defining a loss function over the predicted correspondences and the ideal 2D-3D correspondences which correspond to the object in the given pose. These ideal correspondences are produced with a differentiable renderer so that the entire multi-view alignment procedure is differentiable. We will now describe each step in more detail. However, the main contribution of the paper comprises the multi-view refinement in Step 5. In all experiments, we used the Soft Rasterizer renderer [182].



(a) Refinement with two views.



(b) Refinement with four views.

Figure 5.2: Example refinement results on the Homewbrewed dataset [6]. The top row shows initial per-frame poses produced by PnP before refinement, while the bottom row shows them after refinement. The outline of the object is visualized in green for the ground truth pose, and in blue for the estimated pose. This illustrates that the proposed refiner is capable of selecting a reference frame with a good initial pose and refining it even in the presence of occlusions and imprecise correspondences or when some of the initial pose hypotheses are completely incorrect, as in (b).

5.2.1 Object Detection and Pose Estimation

As the focus of the paper is on refinement and not on the whole pipeline, we have opted for one of the already available dense correspondence-based detectors. We slightly extended the DPOD [100] and trained it on synthetic training data (later referred to as PBR data) provided by the BOP challenge [7]. We separated the original DPOD architecture into two parts: 1) the YOLOv3 [13] detector trained to output tight object bounding boxes with corresponding semantic labels, and 2) the DPOD-like architecture, which predicts object masks and dense correspondences from these detections. The proposed two-stage approach simplifies and accelerates the training procedure of each component, slightly improves the quality of correspondences and leads to better performance on more challenging Homebrewed [6] and YCB-V[16] datasets. In contrast to DPOD [100], which relied on two-dimensional UV maps, we use the three-dimensional Normalized Object Coordinates Space (NOCS) [74]. This parameterization permits trivial conversion between the object coordinate system and the NOCS coordinate system. Additionally, we switched from the ResNet18 backbone to the ResNet34 backbone. The first block of layers is frozen to avoid overfitting when training on synthetic data. Data augmentation and transfer learning allowed the reliable training of the networks. YOLO was trained for 100 epochs, the augmented DPOD for 240. The last checkpoint was used in all experiments.

Let us formally define a model as a set of its vertices: $\mathcal{M} := \{v \in \mathbb{R}^3\}$. Operators that compute minimum and maximum coordinates along the vertex dimension i of all

$v \in \mathcal{M}$ are defined as:

$$min_i(\mathcal{M}) := \min_{v \in \mathcal{M}} v_i, \qquad max_i(\mathcal{M}) := \max_{v \in \mathcal{M}} v_i \tag{5.1}$$

Then, for any point p, the NOCS projection operator is defined w.r.t. the model as

$$\pi_{\mathcal{M}}(p) := \left\{ \frac{p_d - min_d(\mathcal{M})}{max_d(\mathcal{M}) - min_d(\mathcal{M})} \right\}_{d \in \{x,y,z\}} \tag{5.2}$$

and its inverse as $\pi_{\mathcal{M}}^{-1}$. The 6 DoF pose is defined as the standard rigid body transformation, where $\mathbf{T} \in \mathcal{SE}(3)$.

5.2.2 Pose refinement with differentiable renderer

Given a ground truth pose $\mathbf{T}_{gt}$ and a predicted pose $\mathbf{T}_{pr}$, the aim of pose refinement is to find a pose update $\mathbf{T}_\Delta$ that satisfies $\mathbf{T}_\Delta \cdot \mathbf{T}_{pr} = \mathbf{T}_{gt}$. It is, however, impossible to estimate a perfect $\mathbf{T}_\Delta$, because the ground truth pose $\mathbf{T}_{gt}$ is not available. For this reason, proxy loss functions have to be used, which results in a sub-optimal pose update $\mathbf{T}_\Delta$. We propose refining $\mathbf{T}_{pr}$ by optimizing the discrepancy between the predicted noisy NOCS maps from several frames and the perfect NOCS map rendered in the estimated pose with a differentiable renderer.

Assuming that N frames are used for multi-view refinement, we define the corresponding set of predicted segmentations $\tilde{\mathcal{S}} := \{\tilde{\mathbf{S}}_1, \ldots, \tilde{\mathbf{S}}_N\}$, each of which is a binary segmentation mask $\tilde{\mathbf{S}} \in \{0,1\}^{W \times H}$. A set of predicted NOCS correspondences is defined as $\tilde{\mathcal{C}} := \{\tilde{\mathbf{C}}_1, \ldots, \tilde{\mathbf{C}}_N\}$, where each $\tilde{\mathbf{C}} \in [0,1]^{W \times H \times 3}$. These are noisy estimates obtained with the modified DPOD. They are computed once and remain unchanged during refinement. The differentiable renderer is used for a differentiable definition of the following functions: binary foreground/background rendering $S : \mathcal{M} \times \mathcal{SE}(3) \to \{0,1\}^{W \times H}$ and NOCS rendering $C : \mathcal{M} \times \mathcal{SE}(3) \to [0,1]^{W \times H \times 3}$. We will omit their dependence on the CAD model $\mathcal{M}$ to keep the notation concise. For each set of images used for multi-view refinement, one of the images is taken as the reference frame, and then for each f-th image, its relative pose $\Xi_{ref \to f} \in \mathcal{SE}(3)$ is recomputed w.r.t. to the reference frame. Initial pose hypotheses $\mathcal{T} := \{\mathbf{T}_1, ..., \mathbf{T}_N\}$ are estimated separately for each frame with PnP+RANSAC. The pose of the object in the coordinate system of the reference frame is later denoted by $\mathbf{T}_{pr}$.

We use pre-computed NOCS maps in each frame and NOCS maps of the model in the estimated pose in the reference frame transformed to the coordinate system of the f-th frame by transformation $\Xi_{ref \to f} \cdot \mathbf{T}_{pr}$ to define a per-pixel loss function over predicted and rendered correspondences. The per-pixel NOCS discrepancy relates directly to the 3D structure of the object and the error in 3D. The per-pixel loss for pixel p in the f-th frame is defined as follows:

$$\mathcal{L}_{f,p} := \rho\left(\pi^{-1}\left(\tilde{\mathbf{C}}_{f,p}\right), \pi^{-1}\left(C\left(\Xi_{ref \to f} \cdot \mathbf{T}_\Delta \cdot \mathbf{T}_{pr}\right)_p\right)\right) \tag{5.3}$$

In the above equation, ρ stands for the arbitrary distance function in 3D. Essentially, for each frame, the object pose $\mathbf{T}_\Delta \cdot \mathbf{T}_{pr}$ is first transformed to the frame's coordinate system using $\Xi_{ref \to f}$ and then rendered. For each pixel, the predicted and rendered NOCS correspondences are projected into the 3D coordinate system of the model. Then, the discrepancy between them is penalized. The overall loss function is fully differentiable and dependent only on $\mathbf{T}_\Delta$, since the rendering of NOCS maps is performed using the differentiable renderer.

On the level of the full set of images used for refinement, the objective of the refinement is defined as:

$$\mathbf{T}_\Delta^* = \underset{\mathbf{T}_\Delta \in \mathcal{SE}(3)}{\operatorname{argmin}} \sum_{f=1}^{N} \sum_{p \in \mathcal{I}} \left[\tilde{\mathbf{S}}_{f,p} \cdot S \left(\Xi_{ref \to f} \cdot \mathbf{T}_\Delta \cdot \mathbf{T}_{pr} \right)_p \right] \cdot \mathcal{L}_{f,p} \tag{5.4}$$

Here, $\left[\tilde{\mathbf{S}}_{f,p} \cdot S \left(\Xi_{ref \to f} \cdot \mathbf{T}_\Delta \cdot \mathbf{T}_{pr} \right)_p \right]$ serves as an indicator function regarding whether or not the pixel is foreground in both the rendered and the predicted NOCS maps.

The minimization problem, however, cannot be solved optimally due to its non-convexity. Therefore, the loss function is minimized iteratively by gradient descent over the pose update $\mathbf{T}_\Delta$. This can be done using any gradient-based method. We normally observe convergence within 50 optimization steps. A trivial degenerate solution to the optimization problem exists, which sets the loss to zero, namely non-overlapping rendered and predicted $\tilde{\mathcal{S}}$ segmentation maps. But in reality, this is not a problem, because PnP+RANSAC provides reliable initial pose estimates. Moreover, degenerate pose hypotheses are explicitly handled in the choice of the reference frame.

There are numerous ways of implementing the distance function $\rho \colon \mathbb{R}^3 \times \mathbb{R}^3 \to \mathbb{R}^+$ and parameterizing $\mathcal{SO}(3)$ rotations. We use the continuous rotation parameterization from [98], which enables faster and more stable convergence during the optimization procedure than quaternions and Euler angles. As the predicted correspondences and matched correspondences might contain a potentially large number of outliers, a robust ρ function must be used to mitigate this. We experimented with several options and ended up with a particular case of the general robust function introduced in [183]. This function is defined as follows:

$$\rho(e, c) := 1 - \exp\left(-\frac{1}{2} \left(\frac{e}{c} \right)^2 \right) \tag{5.5}$$

c is a hyper-parameter that stands for the scale of the loss function. In our experiments, we adjusted it dynamically according to the median absolute residuals: $c := 2 \cdot MEDIAN\left(|\mathbf{e}|\right)$.

The choice of the reference frame can affect the effectiveness of pose refinement. The goal is to automatically choose a pose which has a high overlap with predicted segmentations when transformed and rendered in other views. Additionally, it should have the smallest possible loss $\mathcal{L}_f$. For each image batch, the reference frame is chosen as follows:

$$\underset{ref \in [1,..,N]}{\operatorname{argmin}} \frac{1}{K} \sum_{f=1}^{N} \frac{\sum_{p \in \mathcal{I}} \left[\tilde{\mathbf{S}}_{f,p} \cdot S \left(\Xi_{ref \to f} \cdot \mathbf{T}_{ref} \right)_p \right] \cdot \mathcal{L}_{f,p}}{IOU\left(\tilde{\mathbf{S}}_f, S \left(\Xi_{ref \to f} \cdot \mathbf{T}_{ref} \right) \right)} \tag{5.6}$$

where K stands for the number of frames with non-zero loss. Additionally, argmin ignores zero values, as they correspond to degenerate poses that are not re-projected correctly onto other frames.

The proposed refinement procedure has a certain similarity to the point-to-point ICP as well as to PnP. In terms of the ICP, point-to-point correspondences are established by rendering NOCS maps and using the predicted NOCS maps with the same spatial location in 2D, as opposed to the nearest neighbor search in the standard 3D ICP. The backpropagation through the renderer corresponds directly to the distance minimization step of the ICP. However, projective transformation is essential. This is because if only RGB information is used, and therefore predicted NOCS maps in all frames are already in the model's coordinate system and do not impose any additional spatial 3D constraints. On the other hand, rendering in order to establish correspondences and minimize the discrepancy directly relates to PnP. The proposed refiner can be seen as a multi-view PnP, where direct pixel-wise error, rather than the reprojection error, is minimized.

5.2.3 Autolabeling

We explore the usefulness of the proposed refiner for the self-annotation of weakly labeled real data, similar to the work of [181]. We assume access to weakly labeled real images with no pose labels. Again, relative transformations between cameras are needed, which is a weaker assumption than having precises per-object pose annotations. For the sake of simplicity, we use the weak 2D labels to filter out correct detections, although this is not necessary, as the detections can be filtered out with the epipolar constraints.

The autolabeling pipeline operates as follows. First, the YOLO detector and the DPOD network are trained in the usual way on synthetically generated data. They are then applied to a partition of the real data and detections are filtered out. The poses are computed using PnP+RANSAC and fed into our multi-view refinement pipeline. Finally, we retrieve the newly estimated labels and use them to build a dataset with real images. The DPOD part is trained as usual, but the predicted NOCS maps are used instead of the ground truth NOCS. The procedure allows us to achieve the performance at the level of DPOD trained on fully annotated real data. No filtering of pose predictions is performed, as ground truth poses are not available in the given scenario. This results in a few images with bad pose annotations being used for training, but they have no significant negative effect on the final performance. The augmented DPOD trained in the standard way for 240 epochs, and the last checkpoint is then used for evaluation.

5.3 Experiments

In this section, we evaluate our multi-view refinement pipeline on LineMOD [5], Occlusion [8], Homebrewed [6] and YCB-V [16] datasets to assess its properties. We then evaluate the proposed autolabeling pipeline. Lastly, we test the robustness of the proposed refiner to the imprecision of relative camera transformations and the choice of frames for refinement. We follow the standard evaluation procedure of [104, 144, 100] and report the pose accuracy only for objects correctly detected in 2D on Linemod and

Table 5.1: Percentages of correctly estimated poses w.r.t. the ADD on the LineMOD [5] dataset. DR1, DR2 and DR4 stand for the proposed refinement with 1, 2 and 4 views respectively. Single-asterisked (*) methods use real training data. Double-asterisked (**) methods report only refinement time instead of the time of the whole pipeline.

	RGB Single-View						**RGB Multi-View**						**RGBD**			
							Closest views		**Random views**		**Farthest views**					
Method	**SSD6D[104]**	**OURS**	**BB8*[68]**	**DPOD[100]**	**OURS**	**PoseCNN*[16]**	**OURS**		**OURS**		**OURS**		**AAE[115]**	**SSD6D[104]**	**DenFus*[123]**	**DenFus.*[123]**
Refinement	**DL[144]**	**-**	**DL[68]**	**DL[100]**	**DR1**	**DeepIM[9]**	**DR2**	**DR4**	**DR2**	**DR4**	**DR2**	**DR4**	**ICP**	**ICP**	**-**	**DL[123]**
Ape	-	28.04	40.40	55.23	38.351	76.95	55.28	85.38	97.89	**100**	97.7	**100**	24.35	-	80	92
Bvs.	-	71.65	91.80	72.69	85.354	97.48	94.95	99.61	99.81	**100**	**100**	99.61	89.13	-	84	93
Cam	-	58.62	55.70	34.76	76.399	93.53	87.57	98.18	99.7	100	99.7	**100**	82.1	-	77	94
Can	-	62.8	64.10	83.59	87.316	96.46	89.57	96.06	95.47	96.06	95.87	**96.85**	70.82	-	87	93
Cat	-	40.12	62.60	65.1	55.988	82.14	78.4	97.2	99.4	**100**	99.4	**100**	72.18	-	89	97
Driller	-	61.46	74.40	73.32	79.801	94.95	86.5	96.02	97.13	97.79	**98.23**	97.79	44.87	-	78	87
Duck	-	26.17	44.30	50.04	38.18	77.65	53.57	81.2	94.56	**98.87**	97.56	**98.87**	54.63	-	76	92
Eggbox	-	87.65	57.80	89.05	93.003	97.09	95.53	98.83	97.27	99.22	98.83	99.61	96.62	-	**100**	**100**
Glue	-	67.48	41.20	84.37	67.988	99.42	89.15	97.67	95.94	98.06	93.22	98.06	94.18	-	99	**100**
Holep.	-	19.09	67.20	35.35	24.568	52.81	35.38	52.69	79.27	87.69	84.81	87.31	51.25	-	79	**92**
Iron	-	83.85	84.70	98.78	94.28	98.26	98.36	**100**	**100**	**100**	**100**	**100**	77.86	-	92	97
Lamp	-	43.52	76.50	74.27	65.664	97.5	80.38	91.05	94.37	97.28	95.92	**96.5**	86.31	-	92	95
Phone	-	45.77	54.00	46.98	74.472	87.72	87.32	97.18	99.6	**100**	99.4	99.6	86.24	-	88	93
Mean	34.1	53.55	62.70	66.42	67.79	88.61	79.38	91.62	96.19	97.19	96.97	**98.02**	71.58	90.9	86	94
Time (ms)	-	56	330	36	398	+ 83**	233	159	232	162	227	163	224	100	-	-

Occlusion. Pose quality is computed in accordance with the ADD metric [5]. On Homebrewed and YCB-V, on the other hand, we submit the predicted poses to the BOP challenge [7] for evaluation and report the Average Recall (AR) metric returned by it.

Single object pose estimation. Here, we compare the quality of poses predicted with our method to different approaches on the LineMOD [5] dataset. The results are summarized in Table 5.1. The table compares our refinement method to various top-performing deep learning methods which reported an ADD score with different pose refinement approaches. It is not our aim to make direct comparisons with monocular methods, but rather to compare the quality of poses after various refinement methods.

Our augmented DPOD, labeled as OURS, achieves 53.55% of the average ADD pose accuracy without any refinement, slightly outperforming the original DPOD, which showed 50% pose accuracy. Next, we evaluate the performance of our differentiable rendering refiner in the monocular scenario (OURS DR1) and compare it to previous state-of-the-art methods based on deep learning (DL): [68, 100, 9, 144]. In this case, the pose accuracy increases from 53.55% to 67.69%, even though only one frame is used and no additional multi-view constraints. The refiner outperforms all the competitors

Table 5.2: Percentages of correctly estimated poses w.r.t. the ADD on the Occlusion [8] dataset. DR1, DR2 and DR4 stand for the proposed refinement with 1, 2 and 4 views respectively.

Train data	**Synthetic**									**Real - GT labels**			
				OURS									
Method	**OURS**	**SSD6D**[104]	**OURS**	**closest views**		**random views**		**farthest views**		**Pix2Pose**[101]	**PVNet**[103]	**DPOD**[100]	**HybPose**[102]
Refinement	**-**	**DL** [144]	**DR1**	**DR2**	**DR4**	**DR2**	**DR4**	**DR2**	**DR4**	**-**	**-**	**-**	**-**
Ape	26.1	-	15.9	21.59	31.45	57.85	**79.81**	61.74	**79.81**	22.0	15.8	18.8	53.3
Can	37.8	-	45.4	63.61	81.67	71.57	**88.05**	72.37	87.65	44.7	63.3	60.8	86.5
Cat	6.3	-	8.3	20.45	34.54	26.82	41.82	23.64	48.18	22.7	16.7	15.4	**73.4**
Driller	43.7	-	52.4	80.3	97.79	91.61	98.67	93.80	**99.11**	44.7	65.7	62.0	92.8
Duck	23.4	-	22.4	54.3	76.95	67.44	**87.01**	72.27	85.94	15.0	25.2	37.8	62.8
Eggbox	22.7	-	21.0	34.19	44.32	41.54	43.08	39.07	48.45	25.2	50.2	70.9	**95.3**
Glue	43.0	-	50.2	67.3	80.92	66.15	75.57	66.92	74.81	32.4	49.6	65.4	**92.5**
Holep.	12.9	-	15.4	45.62	74.08	60.27	**84.51**	63.30	83.50	49.5	39.7	46.9	76.7
Mean	27.0	27.5	28.9	48.42	65.22	60.41	74.82	61.64	75.93	32.0	40.8	47.3	**79.2**

Table 5.3: Results on the Homebrewed dataset [6] reported according to the Average Recall (AR) metric of the BOP challenge [7] on the BOP challenge subset of test images.

Method	Train data	Refinement	AR	Time (s)
OURS	PBR	4 furthest views	0.841	0.578
OURS	PBR	4 random views	0.835	0.584
OURS	PBR	4 closest views	0.83	0.600
OURS	PBR	2 random views	0.818	0.926
OURS	PBR	2 furthest views	0.817	0.926
OURS	PBR	2 closest views	0.787	0.91
CosyPose[10]	PBR	8 views	0.746	0.427
OURS	PBR	no	0.725	0.163
CDPNv2[73]	PBR	no	0.722	0.273
CosyPose[10]	PBR	ICP	0.712	5.326
CDPNv2	synt, PBR	ICP	0.712	0.713
CosyPose[10]	PBR	4 views	0.696	0.445
Pix2Pose[101]	PBR	ICP	0.695	3.248
CosyPose[10]	PBR	no	0.656	0.417

in that category apart from DeepIM, even though BB8 and DeepIM have the advantage of using real train data.

The addition of multiple frames introduces spatial geometric constraints that result in a significant performance boost w.r.t. the ADD score when two (OURS DR2) or four (OURS DR4) frames are used. Even though, the ADD score depends on the choice of the frames, i.e. the closest frames corresponding to weaker constraints, even the two-view refiner can outperform or perform similarly to other approaches using depth-based ICP ([115] and [104]) and DenseFusion, which uses real train data, RGBD inference and iterative DL-based refinement. The table clearly shows that even a minimum multi-view setup can bring a significant performance boost without any need for precise calibrated depth information.

Robustness to occlusions. The robustness of our method to occlusions was evaluated on Occlusion [8], Homebrewed [6] and YCB-V [16] datasets. The results are presented in Table 5.2, Table 5.3 and Table 5.4 respectively.

Unfortunately, detectors trained on synthetic data are seldom evaluated on Occlusion (Table 5.2). To the best of our knowledge, there is only an ADD score from SSD6D with the DL refinement [144]. Therefore, we instead compare to the approaches trained with real data. The advantage of these approaches is that they use real data from the target domain and also overfit to the particular occlusions present in the test images. This enables comparison with the best detectors and assessment of how closely the refiner is able to approach the performance of the algorithms trained on real data. As can be seen in Table 5.2, our refiner is able to significantly improve the performance of the non-refined baseline. If 4 views are used for refinement, the performance of our approach

Table 5.4: Results on the YCB-V dataset reported according to the Average Recall (AR) metric of the BOP challenge [7] on the BOP challenge subset of test images. CosyPose [10] results labeled with * were obtained by re-running the official implementation of the paper.

Method	Train data	Refinement	AR	Time (s)
CosyPose[10]	synt + real	ICP	0.861	2.736
CosyPose[10]	synt + real	8 views	0.853	0.285
CosyPose[10]	synt + real	4 views	0.84	0.318
CosyPose[10]	synt + real	no	0.821	0.241
Pix2Pose[101]	synt + real	ICP	0.78	2.59
EPOS[119]	synt	no	0.696	0.572
OURS	PBR	4 furthest views	0.674	0.555
OURS	PBR	4 random views	0.645	0.557
OURS	PBR	2 furthest views	0.621	848
OURS	PBR	2 random views	0.61	0.855
CosyPose*[10]	PBR	8 views	0.61	0.412
CosyPose*[10]	PBR	4 views	0.592	0.415
CosyPose[10]	PBR	no	0.574	0.342
OURS	PBR	4 closest views	0.564	0.564
OURS	PBR	2 closest views	0.541	0.853
CDPNv2[73]	synt + real	no	0.532	0.143
CDPNv2[73]	PBR	ICP	0.532	1.034
OURS	PBR	no	0.525	0.187
EPOS[119]	PBR	no	0.499	0.764
CDPNv2[73]	PBR	no	0.39	0.448

is on-par with the state-of-the-art results, even though we do not use any real data annotations.

With the Homebrewed dataset [6] (Table 5.3), all top-performing methods were trained on the synthetic PBR images.Our main aim here is to compare our refiner to Cosy-Pose [10], which also utilizes multi-view refinement. It is clear from the table that correspondence-based pose estimation methods (ours and CDPN) confidently outperform the direct pose prediction of CosyPose if no refinement is used. With the multi-view refinement, the proposed methods achieves top results even if only 2 closest views are used for refinement. Additionally, the proposed approach outperforms CDPN, CosyPose and Pix2Pose even if their predictions are refined with ICP.

Table 5.4 shows a comparison of various top-performing methods on the YCB-V [16] dataset. The dataset comes with a real train set and a set of pre-rendered synthetic images (marked as 'real' and 'synt' in the table). On the other hand, synthetic PBR images are also available. When trained on PBR images, raw non-refined CosyPose poses outperform ours. With refinement, our methods outperforms CosyPose, CDPN

Table 5.5: Percentages of correctly estimated poses w.r.t. the ADD score on the Linemod [5] dataset with noisy relative camera transformations.

	2 views						4 views					
	Trans. (% diam.)			Rot. (deg.)			Trans. (% diam.)			Rot. (deg.)		
	5	10	15	5	7.5	10	5	10	15	5	7.5	10
Ape	95	79	58	93	82	68	99	89	66	98	89	73
Bvise	97	85	63	97	91	78	100	91	70	99	93	80
Cam	97	86	65	95	78	65	100	93	56	99	86	66
Can	87	75	53	86	73	63	89	81	56	90	80	62
Cat	95	81	62	95	85	72	99	89	63	99	91	76
Driller	95	83	61	94	85	67	99	91	68	99	91	75
Duck	89	74	57	91	78	60	97	86	62	96	81	63
Eggbox	97	96	94	97	98	96	99	98	98	98	98	98
Glue	94	93	86	94	94	91	98	96	96	97	99	97
Holep.	79	65	44	73	63	51	89	76	52	85	73	62
Iron	96	79	60	97	87	75	99	87	62	99	93	78
Lamp	92	80	61	91	79	68	96	88	65	96	91	77
Phone	95	79	60	95	86	71	100	87	60	99	87	72
Mean	93	81	63	92	83	71	97	89	67	96	89	75

and EPOS trained on the same PBR data. However, training on real data still has a huge advantage on this dataset.

Runtime. To allow faster refinement time, the camera intrinsics were re-computed such that the rendered image always has the dimensions 128×128. Moreover, each model was sub-sampled to contain only 1000 faces. After performing an ablation study on the Linemod dataset, we set the number of refinement iterations to 50. As a result, if 2 views are used for refinement, the average per-image refinement time for one object is 170 milliseconds; with 4 views - 100 milliseconds. Runtime scales linearly with the number of objects in the scene. Tables 5.1,5.3, 5.4 indicate that the proposed refinement, while not being real-time-capable, displays a performance that is similar in terms of time to the multi-view matching and refinement of CosyPose and similar to or faster than ICP-based methods, depending on the dataset. All experiments were conducted on an Intel Core i9-9900K CPU 3.60GHz with NVIDIA Geforce RTX 2080 TI GPU.

Robustness to camera choices. For all datasets used for evaluation, we split each sequence into sets of 2 or 4 images. Images in the different sets do not overlap. For each set, we compute relative camera poses from individual ground truth camera poses. The pose is optimized jointly for images in the set. We experiment with three different view sampling strategies: closest views, random views and furthest views. Tables 5.1, 5.2, 5.3, 5.4show a quantitative comparison of these strategies. It is clear from the tables that even though close camera locations essentially constitute a bad setup for multi-view pose refinement, as the pose error in one frame is not necessarily visible in

Table 5.6: Percentages of correctly estimated poses w.r.t. the ADD score when automatically annotated data is used to train the monocular detector.

Train data	**Synt**	**Real - Autolabels**		**Real - GT labels**				
		DR2	**DR4**					
Method	**OURS**	**OURS**	**OURS**	**DPOD[100]**	**PVNet[103]**	**OURS**	**CDPN[73]**	**HybP.[102]**
Ape	28.04	65.29	68.84	53.28	43.62	74.78	64.38	77.6
Bvise	71.65	98.84	99.81	95.34	99.90	99.71	97.77	99.6
Cam	58.62	93.19	94.25	90.36	86.86	96.52	91.67	95.9
Can	62.80	97.44	98.62	94.1	95.47	99.02	95.87	93.6
Cat	40.12	78.54	82.04	60.38	79.34	90.52	83.83	93.5
Driller	61.46	94.81	96.47	97.72	96.43	97.9	96.23	97.2
Duck	26.17	31.05	32.46	66.01	52.58	48.41	66.76	87
Eggbox	87.65	93.48	95.24	99.72	99.15	99.9	99.72	99.6
Glue	67.48	95.26	96.91	93.83	95.66	97.68	99.61	98.7
Holep.	19.09	25.43	43.47	65.83	81.92	49.62	85.82	92.5
Iron	83.85	99.26	99.59	99.8	98.88	99.39	97.85	98.1
Lamp	43.52	90.79	93.31	88.11	99.33	92.92	97.89	96.9
Phone	45.77	88.14	90.65	74.24	92.41	93.07	90.75	98.3
Mean	53.55	80.89	83.97	82.98	86.27	87.65	89.86	94.5

other frames, the refiner still improves the poses. Random and furthest view selections tend to perform similarly on all the datasets.

Relative Pose Noise. The aim of this experiment is to demonstrate how noise in relative poses affects the overall performance of the multi-view optimization pipeline. To do this, we add noise separately to translation and rotation. For rotation transformation, we sample the perturbation angle for each axis from a normal distribution with zero mean and a standard deviation of 5, 7.5 or 10 degrees. For translation transformation, perturbations are sampled from the normal distribution with zero mean and a standard deviation computed on the basis of an object diameter: $\sigma = \frac{1}{3} \cdot (0.1 \cdot diam_{obj})$. Larger deviations render the problem innately ill-posed due to the definition of the ADD measure. If the camera is at a distance of more than 10% of the model's diameter, the pose will always be classified as incorrect according to the ADD metric. The results can be seen in Table 5.5. Even with the noisy poses, the refiner still ensures a reasonable pose quality. This table also shows that having more views is beneficial in the noisy scenario.

Autolabeling. In this experiment, we retrieve the estimated OURS+DR2 and OURS+DR4 labels and use them to replace the synthetic training set by the automatically generated real labels as discussed above. The extended training set is then used to fine-tune the synthetically trained baseline. The results can be seen in Table 5.6. It can be clearly seen that the fine-tuned network trained on autolabels significantly outperforms the synthetic DPOD and is very competitive when compared to the state-of-the-art methods trained on full ground truth labels.

5.4 Conclusions

In this paper, we propose a novel object pose refinement pipeline. We adopt the idea of using multiple frames to produce a single joint object pose estimate. The use of multiple frames enables effective use of geometric constraints. The proposed refiner is based on the external object detector, which outputs deep 2D-3D correspondences in the form of Normalized Object Coordinate space. Predicted dense correspondences from calibrated multiple frames can be brought together to obtain a better pose estimate using a differentiable renderer. The proposed approach imposes no constraints on the number of frames used and the exact positions of the cameras in the 3D world, and it remains robust even when the cameras are in close proximity to one another. We experimentally show that the refiner works excellently on Linemod, Occlusion, Homebrewed and YCB-V datasets. As an alternative use case, we demonstrate how the proposed approach can be applied to automatically annotate real train data, which has no object pose annotations. The approach even remains effective if the relative transformations in-between frames are imprecise. Moreover, even a multi-view setup with only two frames already produces excellent results. This demonstrates that the approach can be used successfully in practical applications.

6 Analysis of Pose Estimation with Dense Correspondences

This chapter introduces a three-stage 6 DoF object detection method called DPODv2 (Dense Pose Object Detector) that relies on dense correspondences. We combine a 2D object detector with a dense correspondence estimation network and a multi-view pose refinement method to estimate a full 6 DoF pose. Unlike other deep learning methods that are typically restricted to monocular RGB images, we propose a unified deep learning network allowing different imaging modalities to be used (RGB or Depth). Moreover, we propose a novel pose refinement method, that is based on differentiable rendering. The main concept is to compare predicted and rendered correspondences in multiple views to obtain a pose which is consistent with predicted correspondences in all views. Our proposed method is evaluated rigorously on different data modalities and types of training data in a controlled setup. The main conclusions is that RGB excels in correspondence estimation, while depth contributes to the pose accuracy if good 3D-3D correspondences are available. Naturally, their combination achieves the overall best performance. We perform an extensive evaluation and an ablation study to analyze and validate the results on several challenging datasets. DPODv2 achieves excellent results on all of them while still remaining fast and scalable independent of the used data modality and the type of training data.

6.1 Introduction

Object detection and 6 DoF pose estimation are not new fields in computer vision. In fact, they are among the major driving forces, being crucial for various application fields, such as augmented reality, robotics and autonomous driving. Therefore, there is a vast number of methods trying to tackle this problem.

In the pre-deep learning era, pose estimation was typically performed either completely using depth or using a combination of depth and RGB data. However, recent trends in 6 DoF pose estimation move towards disregarding depth data as a modality and using only RGB data. The reasons for that are manifold, but the most important are the omnipresent availability of RGB cameras in the modern devices and the success of deep learning methods. The progress in RGB deep learning allows the researchers to choose from a vast number of ready to use network architectures and focus their attention on extending them to predict the pose. The state of the art RGB solutions are solely based on convolutional neural networks (CNNs), demonstrating impressive results that could barely be imaginable a couple of years ago. Even though recent RGB methods, such as CosyPose [10] perform better than depth-based point pair feature approaches [15], they

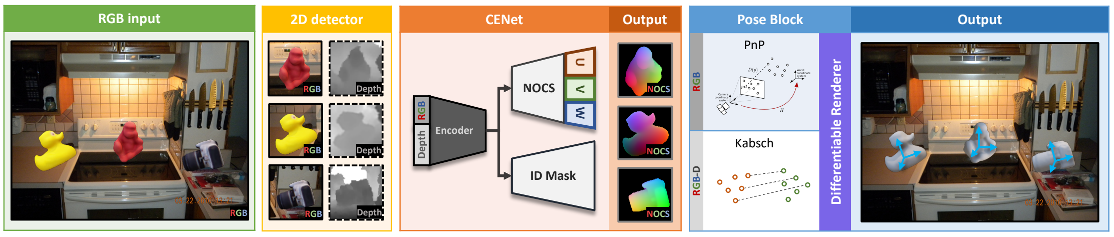


Figure 6.1: Synthetic toy example illustrating three-stage correspondence-based 6 DoF pose estimation: A full RGB image is fed to a 2D object detector for bounding box estimation. Resulting bounding boxes are then used to generate crops on the available data, which are subsequently fed into the correspondence network. If only RGB images are provided, then the pose is estimated from correspondences using 2D-3D PnP. In case registered depth data is also in place, we project the estimated correspondences into 3D and use the 3D-3D Kabsch algorithm.

still suffer from perspective ambiguities and appearance changes. This can be remedied with the additional use of depth data or an extra step with multi-view refinement.

Inspired by the works of Taylor et al. [75] and Gueler et al. [76], Brachmann et al [8] and Jafari et al. [118], we developed and compared several variations of deep dense correspondence-based 6 DoF object detectors using either RGB or depth as input. We ran a thorough analysis and ablation studies to evaluate the strengths and weaknesses of the presented modalities and compared them to the state of the art. Moreover, each correspondence estimation networks is trained on two types of data: synthetic and real. While real labeled data are most commonly used to achieve the best possible results, the acquisition of such data is often infeasible in real applications due to the high costs and significant time efforts. Synthetic data, on the other hand, is free of these drawbacks and can be easily rendered in a variety of scenes and under an unlimited number of poses. Unfortunately, the methods trained on synthetic data are subject to the domain gap problem, due to the dissimilarity between real and synthetic images. This might causes them to perform worse than the methods trained on real data. However, as was shown in HomebrewedDB [6], networks trained only synthetic data are less prone to overfitting due to a larger data variability, which allows them to perform similarly to the methods trained on real data when train and test images do not come from the same image sequence. Therefore, detectors trained on synthetic data are desirable, because the required data is easier to obtain and because such detectors tend to generalize better, thus being of higher practical significance.

Even though there is a larger number of pose estimation methods for all kinds of data modalities, they are typically designed for a specific one and have completely different architectures from methods operating on other data modalities. This makes it impossible to measure which advances come from better methods and which come from a different data modality or from better training data. In this chapter, we propose a unified deep neural network architecture capable of predicting dense correspondences either from depth maps or from RGB images. The pose estimation part enables direct comparison of the data modalities on various datasets. Additionally, in contrast to the other

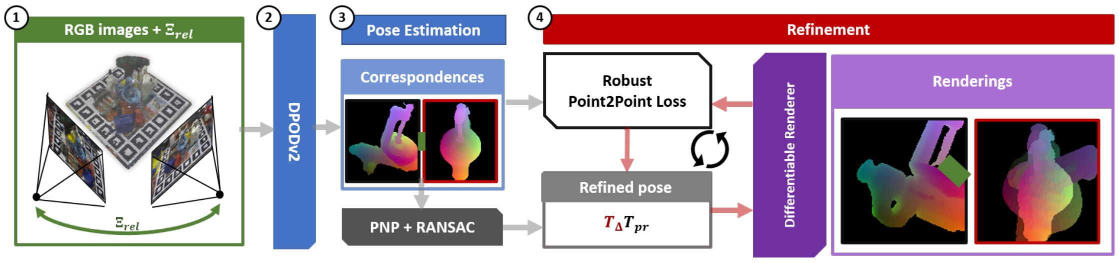


Figure 6.2: Multi-view pose optimization: The algorithm takes the output of DPODv2 from several views with known relative camera transformations as input. An initial pose hypothesis is iteratively refined until it converges to a pose, which is consistent with predicted correspondences in all frames. The proposed loss function penalizes pixel-wise distance between a predicted correspondence and a correspondence corresponding to the current pose hypothesis. The loss function is implemented using a differentiable renderer.

dense correspondence methods, we report per-pixel correspondence error to deepen the understanding of where pose imprecision comes from. From the results obtained we conclude that RGB images excel in object localization and correspondence estimation, while 3D-3D correspondences between the depth images and the model result in more accurate pose estimates. Naturally, their combination brings the best performance. We also introduce a correspondence-based refinement method based on differentable rendering. Moreover, if relative camera transformations between frames are known, the predicted poses can be additionally refined by aligning per-image dense correspondences between the object correspondences rendered in the predicted pose and the correspondences predicted by the network. The proposed loss function is implemented using a differentible renderer, which allows for its minimization with the standard gradient-based methods.

The main contributions of this work can be summarized as follows:

- A unified framework for dense correspondence estimation, whose architecture is agnostic to the input modality type available (RGB or Depth);
- A thorough analysis and ablation study of the presented methods and estimation of their strengths and weaknesses;
- An analysis of the quality of predicted correspondences;
- A correspondence-based 6 DoF pose refinement extensible to multiple views.

6.2 Methodology

In this chapter, we propose a three-stage object detection and pose estimation pipeline with a potential fourth refinement stage. It builds atop of the existing state of the art results with training on both real and synthetic data of potentially different modalities. The pipeline is visualized in Figure 6.1. The first stage is responsible for 2D object detection from RGB images, which is a standard and well-studied research area. In the second stage, a convolutional neural network, specifically designed for semantic segmentation, takes a detected patch with an object and regresses dense correspondences between input pixels and the 3D object model. Object poses is computed using the predicted correspondences in the third stage. The optional fourth stage, visualized in Figure 6.2, is responsible for pose refinement, as discussed later. Such a setup, allows for flexible mixing of data modalities and types of training data. Additionally, it simplifies the training procedure, as it breaks the task into smaller independent subtasks. In this section, we provide detailed explanations of each part. Then, we introduce and explain the multi-view refiner which operates atop of dense correspondences predicted by DPODv2.

6.2.1 2D Object Detection

The first step of the pipeline consists of an off-the-shelf 2D object detector. As shown in previous work of [104, 115, 116], it is easier to achieve good 2D detection recall if it is done by conventional object detection approaches. SSD6D, which was trained purelly on synthetic data, relied on fine-tuning a separate confidence threshold per each object class to achieve near-perfect recall. AAE[115, 116], on the other hand, trained their 2D detector completely on real data. We used YOLOv3[13] in all our experiments, although the pipeline is agnostic to a particular choice of the detector. YOLO was trained either completely on real or completely on synthetic RGB data. Another crucial reason for splitting object detection and correspondence estimation into two disjoint steps is scalability. Most of the approaches, mentioned in the related work, train a separate detector per each object to improve the results. While it is achievable on smaller datasets, such as Linemod [5], it is more challenging on more sophisticated datasets [6, 12] and close to impossible in real life.

6.2.2 Pose Parameterization With Dense Correspondences

Instead of direct pose regression, which is still challenging and ill-posed for deep learning, we advocate the use of dense per-pixel correspondences between the image and the object model. With the correspondences at hand, the pose can be estimated with a number of well-studied methods depending on the data modality of choice [4]. While in the original DPOD paper, the authors relied on well-known 2-dimensional UV maps from computer graphics, we opted for the three-dimensional Normalized Object Coordinates Space (NOCS) [74] maps. The main reason is its simplicity and the lack of need for manual adjustment of the UV maps. Each dimension of NOCS corresponds to a dimension of the object scaled uniformly to fit in $[0, 1]$ range. This parameterization allows for trivial conversion between the object coordinate system and the NOCS coordinate system, which is more suitable for regression with deep learning due to its constrained nature.

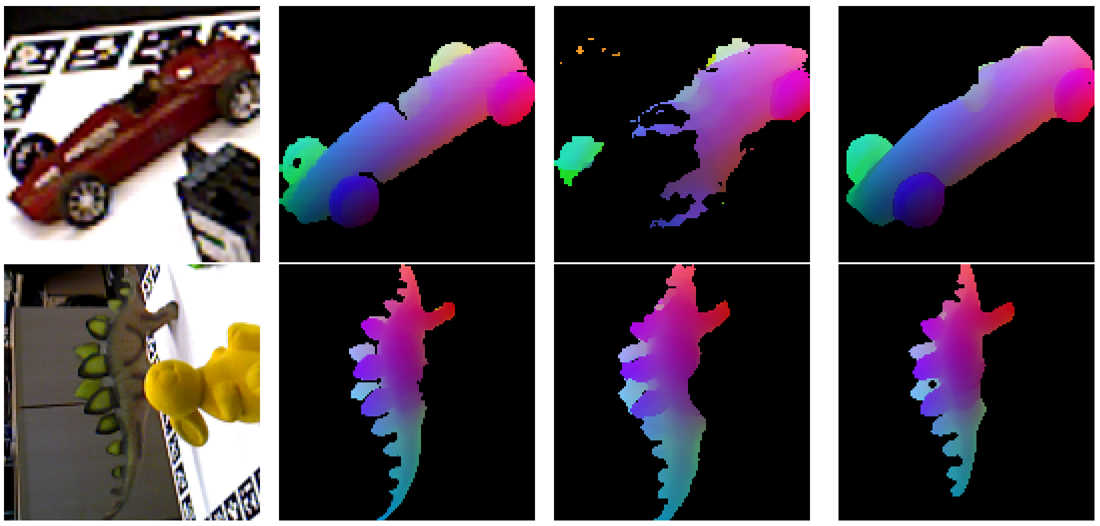


Figure 6.3: Visual comparison of predicted segmentation maps of the visible object parts and color-coded NOCS for the same patch depending on whether RGB or Depth CENet is used.

Let us formally define a model as a set of its vertices: $\mathcal{M} := \{v \in \mathbb{R}^3\}$. Operators that compute minimal and maximal coordinate along the vertex dimension i of all $v \in \mathcal{M}$ are defined as

$$min_i(\mathcal{M}) := \min_{v \in \mathcal{M}} v_i, \qquad max_i(\mathcal{M}) := \max_{v \in \mathcal{M}} v_i \tag{6.1}$$

Then, for any model's vertex v, the NOCS projection operator is defined w.r.t. the model as

$$\pi_{\mathcal{M}}(v) := \left\{ \frac{v_d - min_d(\mathcal{M})}{max_d(\mathcal{M}) - min_d(\mathcal{M})} \right\}_{d \in \{x,y,z\}} \tag{6.2}$$

The inverse of the transformation is denoted by $\pi_{\mathcal{M}}^{-1}$. We use the standard $SE(3)$ definition of 6 DoF object pose [4]. A model with the given parameterization can easily be rendered to produce pixel-wise ground truth 2D-3D correspondences. Thus, it allows for instant establishment of 2D-3D and 3D-3D correspondence if RGB or depth data is used respectively.

6.2.3 CENet: Correspondence Estimation Network

The architecture of the Correspondence Estimation Network builds on the architecture of DPOD [100]. The network is an encoder-decoder convolutional neural network with skip connections resembling UNet [61]. The network accepts the input of size $\mathcal{I} \in \mathbb{R}^{128 \times 128 \times D}$, where D stands for the number of input channels and depends on the data modality. The network has a common encoder and four separate decoder heads. One head is responsible for predicting binary per-pixel segmentation mask $\tilde{\mathcal{S}} \in \mathbb{R}^{128\times128\times2}$, while the other three output tensors, each of which $\tilde{\mathcal{C}}_d \in \mathbb{R}^{128\times128\times256}$ after softmax corresponds to pixel-wise probabilities of discretized NOCS coordinates. The encoder is based on the 34-layer ResNet [51] architecture, which proved to be both sufficiently effective and fast. The decoders upsample the computed features up to its original size using a stack of bilinear interpolations followed by convolutional layers. We used biliinear upsampling instead of upconvolutions to make inference more rapid and memory-efficient. However, in principle any other architecture for semantic segmentation could be used.

Formulating NOCS coordinate regression problem as discrete classification problem proved to be useful for much faster convergence and for the superior quality of correspondences as was also confirmed in previous work [100, 74]. Initial experiments on direct coordinate regression showed very poor results in terms of correspondence quality. The main reason for the problem was the infinite continuous solution space, i.e., $[0;1]^3$, where 3 is the number of dimensions and $[0,1]$ is the normalized coordinate range of a 3D model. Classification of the discretized 2D correspondences allowed for a large boost of the output quality by dramatically decreasing the output space, which is now 256^3 with 256 being the size of a discretized NOCS dimension.

As a result, the network is trained by optimizing two classification losses. Segmentation loss is defined as a per-pixel binary Dice loss [184] $\mathcal{L}_{DICE}$, which helps overcome the imbalanced number of foreground and background pixels and NOCS classification cross entropy loss defined only for foreground pixels (pixels occupied with non-occluded objects of interest) separately per each NOCS dimension d:

$$\mathcal{L}^d_{NOCS} = \sum_{i=1}^{128}\sum_{j=1}^{128} \mathbb{1}[\mathcal{I}_{i,j} \text{ is foreground}]\mathcal{L}_{cls}(\tilde{\mathcal{C}}_{d,i,j}, \mathcal{C}^{gt}_{d,i,j}) \tag{6.3}$$

Which results in the total per-image loss:

$$\mathcal{L} = \alpha\mathcal{L}_{DICE} + \sum_{d=1}^{3} \mathcal{L}^d_{NOCS} \tag{6.4}$$

where α is a weight factors set to 5 in all our experiments.

6.2.4 Inference with The Correspondence Estimation Network

Given YOLO predictions, we explored three different ways to first get the dense correspondences and then utilize them for pose estimation. The first approach follows the conventional DPOD paradigm of running an RGB correspondence network followed by the PnP with RANSAC. Alternatively, if depth maps are available, it is possible to project the predicted correspondences from 2D into 3D and then use the Kabsch algorithm with RANSAC to predict the pose from 3D-3D correspondences. Depth maps can also directly be used instead of RGB for inferring segmentation masks and 3D-3D correspondences. In this case, Kabsch+RANSAC is again used for pose estimation.

6.2.5 Multi-View Refinement With Differentiable Renderer

The overall pipeline of the proposed refiner is provided in Figure 6.2. The key idea of the refiner is to predict NOCS correspondences and poses separately for each frame with DPODv2 as described above. Then, if several views with known relative camera positions are available, the predicted poses can be refined to produce a pose which is plausible given correspondences in all frames and the geometric constraints imposed by the relative camera poses. This is achieved by defining a loss function which penalizes pixel-wise difference between predicted NOCS coordinates with CENet, and NOCS coordinates corresponding to the object rendered in the predicted camera poses. The loss function is implemented using the differentiable renderer and minimized iteratively by computing a pose update $\mathbf{T}_\Delta \in SE(3)$ using gradient descent at each step.

We rely on the assumption of the availability of relative camera transformations. There are multiple ways how they can be estimated. Camera transformations can be obtained by placing the object on the markerboard and either using an actual multi-camera system or using a single camera but moving the markerboard. In the scenario of robotic grasping, a camera can be mounted on the robotic arm to enable observation of the object from several viewpoints. Alternatively, SLAM methods could be used to estimate the transformations. While relative transformations are assumed to be known, the object poses still remain unknown and cannot be inferred directly from the relative camera transformations.

The differentiable renderer is used for a differentiable definition of the following functions: binary foreground/background rendering $S : \mathcal{M} \times SE(3) \to \{0,1\}^{W\times H}$ and NOCS rendering $C : \mathcal{M} \times SE(3) \to [0,1]^{W\times H\times 3}$. We will omit their dependence on the CAD model $\mathcal{M}$ to keep the notation concise. For each set of images used for multi-view refinement, one of the images is taken as the reference frame, and then for each f-th image, its relative pose $\Xi_{ref\to f} \in SE(3)$ is recomputed w.r.t. to the reference frame. Initial pose hypotheses $\mathcal{T} := \{\mathbf{T}_1, ..., \mathbf{T}_N\}$ are estimated separately for each frame with PnP+RANSAC. The pose of the object in the coordinate system of the reference frame is later denoted by $\mathbf{T}_{pr}$. $\mathbf{T}_s \in SE(3)$ is a symmetry transformation associated with each object, which must be adjusted to produce consistent NOCS maps for symmetric objects. Computation of $\mathbf{T}_s$, described in the data preparation section, is closed-form and quick but has to be performed for each object in each frame after each iteration.

For a pixel p in the f-th frame, the loss function is defined as follows:

$$\mathcal{L}_{f,p} := \rho\left(\pi^{-1}\left(\tilde{\mathbf{C}}_{f,p}\right), \pi^{-1}\left(C\left(\Xi_{ref\to f}\cdot \mathbf{T}_\Delta \cdot \mathbf{T}_{pr}\cdot \mathbf{T}_s\right)_p\right)\right) \tag{6.5}$$

where ρ is an arbitrary distance function. In a nutshell, $\tilde{\mathbf{C}}_{f,p}$ is a predicted NOCS correspondence, while $C\left(\Xi_{ref\to f}\cdot \mathbf{T}_\Delta \cdot \mathbf{T}_{pr}\cdot \mathbf{T}_s\right)_p$ projects the model in the pose $\mathbf{T}_\Delta \cdot \mathbf{T}_{pr}$ onto the corresponding frame and computes a NOCS coordinate for that pose. The inverse transformation allows us to relate the pixel-wise discrepancy to the actual error in the 3D coordinate system of the model.

On the level of the full set of images used for refinement, the objective of the refinement is defined as:

$$\mathbf{T}^*_\Delta = \underset{\mathbf{T}_\Delta \in \mathcal{SE}(3)}{\operatorname{argmin}} \sum_{f=1}^{N} \sum_{p\in\mathcal{I}} \left[\tilde{\mathbf{S}}_{f,p} \cdot S\left(\Xi_{ref\to f} \cdot \mathbf{T}_\Delta \cdot \mathbf{T}_{pr} \cdot \mathbf{T}_s\right)_p\right] \cdot \mathcal{L}_{f,p} \tag{6.6}$$

Here, $\left[\tilde{\mathbf{S}}_{f,p} \cdot S\left(\Xi_{ref\to f} \cdot \mathbf{T}_\Delta \cdot \mathbf{T}_{pr}\right)_p\right]$ serves as an indicator function which shows whether or not the pixel is foreground in both the rendered and the predicted NOCS maps.

Choice of the reference frame might have a significant effect on the effectiveness of pose refinement, especially in case of occlusions which cause bad initial poses. We aim at choosing a reference frame in such a way that its pose has the lowest error when re-projected to other frames. To deal with degenerate cases when there is no overlap or very small overlap between the projected model and the predicted foreground segmentations, the loss is scaled by their intersection over union as follows:

$$\underset{ref\in[1,..,N]}{\operatorname{argmin}} \frac{1}{K} \sum_{f=1}^{N} \frac{\sum_{p\in\mathcal{I}} \left[\tilde{\mathbf{S}}_{f,p} \cdot S\left(\Xi_{ref\to f} \cdot \mathbf{T}_{ref} \cdot \mathbf{T}_s\right)_p\right] \cdot \mathcal{L}_{f,p}}{IOU\left(\tilde{\mathbf{S}}_f, S\left(\Xi_{ref\to f} \cdot \mathbf{T}_{ref} \cdot \mathbf{T}_s\right)\right)} \tag{6.7}$$

where K stands for the number of frames with non-zero loss. Additionally, argmin ignores zero values, as they correspond to degenerate poses that are not re-projected correctly onto other frames.

6.2.6 Implementation Details

Our pipeline is implemented using the Pytorch [185] deep learning framework. All the experiments were conducted on an Intel Core i9-9900K CPU 3.60GHz with NVIDIA Geforce RTX 2080 TI GPU. YOLO was trained with the ADAM optimizer with a constant learning rate of 1×10^{-3} and weight decay of 3×10^{-5} for 100 epochs. Correspondence estimation network was trained for 240 epochs with the learning rate of 5×10^{-5} which was decreased after 40th, 120th and 200th epochs. We used EPnP [1]+RANSAC implementation from OpenCV [186] and point-to-plane ICP from Open3D [187]. For Kabsch+RANSAC we used our own minimalistic implementation.

The correspondence inference network uses 128×128 images for training and for testing, while YOLO is trained to predict tight bounding boxes around the objects. Therefore, the shortest side of each bounding box predicted by YOLO is padded to match the longest side, after which the corresponding image patch is resized preserving the aspect ratio. During training, small random imprecisions are added to each side of the ground truth bounding box to ensure that CENet is robust to noisy YOLO predictions. The ablation studies show the effectiveness of this approach, as the quality of predicted correspondences does not deteriorate much when the predicted bounding boxes are used instead of the ground truth boxes.

When training on synthetic data, the problem of domain adaptation becomes one of the main challenges. Training the network without any prior parameter initialization makes generalization to the real data difficult if possible at all. A simple solution to this problem was proposed in several works, including [179, 144], where they freeze the first layers of the network trained on a large dataset of real images, e.g., ImageNet [173] or MS COCO [174], for the object classification task. The common observation of the authors is that these layers, learning low-level features, very quickly overfit to the perfect object renderings. We found out that this setup is only partially needed for our networks. We froze first 20 layers when we trained YOLO. The weights were initialized from the YOLO trained on the MS COCO dataset [174]. In the correspondence estimation network, the weights were initialized from the Resnet pretrained on ImageNet [173] and then were all modified during training. When the correspondence network was trained on depth maps, the first layer was instead initialized randomly and then all layers were modified during training. We have found out that replacing BatchNorm [89] with InstanceNorm[91] helps overcome overfitting and allows for reliable and stable training in this case.

We used the Soft Rasterizer renderer [182] in all experiments with the multi-view refiner. The general robust function [183] was used as a distance measure ρ in the loss function to effectively deal with outliers. Continuous rotation parameterization from [98] was used to achieve faster and more stable convergence during the optimization procedure in comparison to quaternions and Euler angles.

6.3 Data Preparation

When it comes to data preparation for the tasks of object detection and pose estimation, two distinctive paradigms are observed. The most popular and conventional one follows the standard way of machine learning of collecting a large database of labeled data. Unfortunately, pose estimation requires more involved labels, which cannot be produced manually, in contrast to the standard classification or segmentation labels. Pose labeling requires sophisticated multi-step procedures, such as in [6, 12]. In spite of those challenges, real label train data remain the predominant choice of data especially because it allows to achieve excellent performance on academic benchmarks. In the deep learning on point clouds, real train data are also predominant.

On the other hand, it is possible to leverage advances in computer graphics and to rely solely on synthetically rendered train data if object CAD models are available. The advantages of this include virtually unlimited number of images and poses with no labeling effort. However, the domain gap between the real and the rendered images comes into play to hinder the detection rate and pose accuracy. In the past, the predominant way to train a deep neural network on synthetic data was to render the object on random backgrounds, as was done in DPOD [100], SSD6D [104] and in [179]. On the other hand, complete photo-realistic scene rendering is becoming more popular [188].

6.3.1 RGB Data Preparation

Real Data Preparation. Preparation of real train data is rather straightforward. Given an image, CAD models, and associated ground truth poses, it is enough to render present objects in ground truth poses. Binary segmentation masks and per-pixel NOCS coordinates are obtained by rendering the models in the ground truth poses. Following the paradigm of the previous works[69, 100] object patches corresponding to the visible object parts were extracted and inpainted in random backgrounds sampled from MSCOCO [174]. This ensures that the network learns object-specific features rather than overfits to backgrounds, leading to better generalization to new unseen scenes. For training the detector, a random number of objects were added to each background. For the correspondence estimation network, only one object is in-painted. The object patch is resized randomly to simulate different distances from camera. Additionally, a random in-plane rotation is performed. Then, we use standard color augmentations used by the backbone networks. When training the correspondence network, random occlusions also have to be added to make the network more robust. Realistically looking occlusions are approximated by sampling a random patch, corresponding to a different object, which is then in-painted randomly atop of the object of interest. Occlusions are added at random, but it is made sure of that the target object is still visible in the patch.

Synthetic Data Preparation. We relied on the PBR images provided by the BOP challenge organizers for the preparation of syntetic training data. Those images were produced by simulation using BlenderProc [188]. No additional transformations were performed on those images apart from random photometric augmentations, such as brightness augmentation. Usage of the provided PBR images also ensured more fairer comparison of our methods to the other methods on the BOP challenge datasets by making sure that the competing methods were trained on exactly the same data.

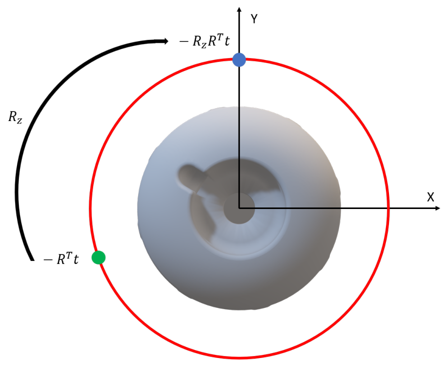
6.3.2 Depth Data Preparation

In general, training deep learning networks on depth is harder than on RGB images due to holes in the depth and a much larger range of values. To get rid of empty depth values, we fill them in using nearest neighbor interpolation. To alleviate the range problem, depth maps are parameterized similar to PointNet++[141] by replacing a depth value in each pixel with a distance ti to the mean depth in the neighborhood of size T:

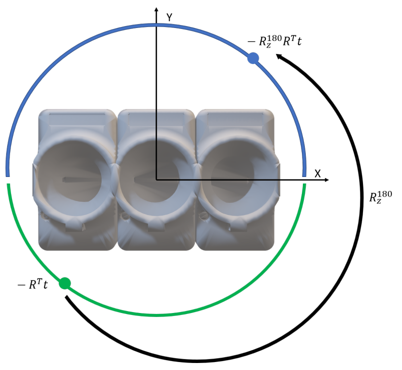
$$\mathcal{I}_{i,j} = \mathcal{I}^{raw}_{i,j} - \frac{1}{T^2} \sum_{x=-T}^{T} \sum_{y=-T}^{T} \mathcal{I}^{raw}_{i-x,j-y} \tag{6.8}$$

for $i, j \in \mathbb{R}^{128}$ and $\mathcal{I}^{raw}$ being the original depth image with already interpolated missing values. This depth parameterization ensured simpler and more stable training. We set the neighborhood radius T to 5 in our experiments.

In train time, depth maps undergo several random augmentations. First, random holes of random size are added to the depth map and then interpolated accordingly. Then, random Perlin noise [189] is added to the background, that results in smooth random backgrounds, which is especially important for synthetically generated data. Foreground depth values are augmented with random Gaussian noise. Synthetic train data was again taken from the synthetic PBR images.



(a) A continuous symmetry around the Z axis



(b) A discrete symmetry around the X axis

Figure 6.4: Pose recalculation for symmetric objects during training. Poses are disambiguated during training to produce consistent NOCS maps. In case of a continuous symmetry around Z axis (Example a), a rotation around Z is added to the initial pose to ensure that the camera is always located on the same arch in the object coordinate system. In case of discrete symmetries (Example b), all symmetric poses are mapped to the one base pose by rotation around the symmetry axis.

6.3.3 Handling Object Symmetries

While there is a large body of theoretical work on dealing with ambiguities induced by object symmetries, for instance [80, 190, 191], we followed a more conventional and hands-on approach. Similarly to the DPOD and SSD6D papers, we relied on training on consistent and unambiguous ground truth NOCS, rather than inferring multiple pose hypotheses as in [80, 191]. While DPOD and SSD6D were trained only on images of objects taken from non-ambiguous poses, we used all the images and then disambiguated the poses in order to render consistent NOCS coordinates as visualized in Figure 6.4. There are two main symmetry types in the used datasets: a continuous rotation symmetry around an axis, when every rotation around the axis results in an identically looking object, and discrete symmetry around an axis.

For objects, which are fully symmetric around an axis, NOCS maps are rendered from a consistent camera location in the object coordinate system as visualized in Figure 6.4a. Typically, the object is symmetric around its Z axis. Given an object pose $R \in SO(3), t \in \mathbb{R}^3$, which transforms from the model coordinate system into the camera coordinate system, the camera location in the model coordinate system is given by $-R^\top t$. To produce consistent NOCS maps, we rotate the object around Z axis axis in such a way, that a camera location always lies in the YZ plane of the model system, i.e. $R_z \in SO(3)$ such that the X coordinate of $-R_z R^\top t$ is 0. This makes the object always visible from the same position. R_z can easily be computed in a closed form by computing an angle between vectors $\overrightarrow{\left(\left(-R^\top t\right)_x, \left(-R^\top t\right)_y, 0\right)}$ and $\overrightarrow{(0,1,0)}^\top$. The final disambiguated object pose is defined by $RR_z^\top$ and t.

Table 6.1: Data modalities and types of train data used in the experiments.

	YOLO		CENet			
Dataset/Modality	Real RGB	Synt. RGB	Real RGB	Synt. RGB	Real depth	Synt. depth
Linemod	Yes	Yes	Yes	Yes	Yes	Yes
Occlusion	Yes	Yes	Yes	Yes	Yes	Yes
TLESS	Yes	Yes	Yes	Yes	No	Yes
HomebrewdDB	No	Yes	No	Yes	No	Yes

Handling objects with discrete symmetries in respect to the plane is rather straightforward. Assuming for the illustration purposes that rotating the object around Z axis by 180 degrees (denoted by R_z^{180}) produces an identically looking object as in Figure 6.4b. One of the symmetry sides is taken as the base region of the symmetry, denoted by blue in the figure. Then, a camera location is computed and if the camera lies on the opposite side of the object, the rotation is updated as RR_z^{180}. Basically, it ensures that the object is always visible from the same side. Even though [80] noted that this transformation might be unfavourable for neural networks due to its non-continuity, it works reliably in practice.

6.4 Experiments

6.4.1 Datasets

Over the past decade, numerous datasets for object detection and 6 DoF pose estimation have been proposed [5, 8, 12, 16, 170, 6, 171, 192, 172]. However, in this work we

Table 6.2: Pose estimation performance on Linemod on RGB images of methods trained on synthetic data: The table reports the percentages of correctly estimated poses w.r.t. the ADD score. Our approach sets the new state of the art both among the methods trained on real data and the methods trained on synthetic data. Our refiner outperforms other RGB refiners. Run times are provided as they are reported in the original papers using non-identical hardware.

Train Data			Synthetic					
Method	SSD6D [104]	AAE [116]	DPOD [100]	Ours	SSD6D [104]	DPOD [100]	Ours	Ours
Refinement	-				DL [144]	DL [100]	2 calib. views	4 calib. views
Ape	2.60	3.96	37.22	**62.14**	-	55.23	98.54	**100.00**
Bvise	15.10	20.92	66.76	**88.39**	-	72.69	**100.00**	**100.00**
Cam	6.10	30.47	24.22	**92.51**	-	34.76	99.67	**100.00**
Can	27.30	35.87	52.57	**96.66**	-	83.59	**100.00**	**100.00**
Cat	9.30	17.90	32.36	**86.17**	-	65.1	99.66	**100.00**
Driller	12.00	23.99	66.60	**90.15**	-	73.32	99.83	**100.00**
Duck	1.30	4.86	26.12	**54.86**	-	50.04	99.68	**100.00**
Eggbox	2.80	81.01	73.35	**98.64**	-	89.05	97.70	**99.04**
Glue	3.40	45.49	74.96	**95.41**	-	84.37	97.70	**98.03**
Holep.	3.10	17.60	24.50	**27.08**	-	35.35	94.01	**99.03**
Iron	14.60	32.03	85.02	**98.26**	-	98.78	**100.00**	**100.00**
Lamp	11.40	60.47	57.26	**91.04**	-	74.27	99.51	**100.00**
Phone	9.70	33.79	29.08	**74.34**	-	46.98	**100.00**	**100.00**
Mean	9.10	28.65	50.00	**81.20**	34.1	66.42	98.95	**99.70**
Time (ms)	-	24	-	32	-	-	202.00	132.00

Table 6.3: Pose estimation performance on Linemod on RGB images of methods trained on real data: The table reports the percentages of correctly estimated poses w.r.t. the ADD score. Our approach sets the new state of the art both among the methods trained on real data and the methods trained on synthetic data. Our refiner outperforms other RGB refiners. Run times are provided as they are reported in the original papers using non-identical hardware.

Train Data	**Real**										
Method	Pix2Pose [101]	DPOD [100]	PVNet [103]	CDPN [73]	HybridPose [102]	Ours	BB8 [68]	PoseCNN [16]	DPOD [100]	Ours	Ours
Refinement	-						DL [68]	DeepIM [9]	DL [100]	2 calib. views	4 calib. views
Ape	58.10	53.28	43.62	64.38	63.1	80.09	40.40	76.95	87.73	98.66	**100.00**
Bvise	91.00	95.34	**99.90**	97.77	**99.9**	99.71	91.80	97.48	98.45	**100.00**	**100.00**
Cam	60.90	90.36	86.86	91.67	90.4	**99.21**	55.70	93.53	96.07	**100.00**	**100.00**
Can	84.40	94.10	95.47	95.87	98.5	**99.60**	64.10	96.46	99.71	**100.00**	**100.00**
Cat	65.00	60.38	79.34	83.83	89.4	**95.11**	62.60	82.14	94.71	**100.00**	**100.00**
Driller	76.30	97.72	96.43	96.23	98.5	**98.91**	74.40	94.95	98.80	**100.00**	**100.00**
Duck	43.80	66.01	52.58	66.76	65.0	**79.54**	44.30	77.65	86.29	98.12	**100.00**
Eggbox	96.80	**99.72**	99.15	**99.72**	**100.0**	99.63	57.80	97.09	99.91	99.68	**100.00**
Glue	79.40	93.83	95.66	99.61	98.8	**99.81**	41.20	**99.42**	96.82	97.46	98.84
Holep.	74.80	65.83	81.92	85.82	**89.7**	72.30	67.20	52.81	86.87	99.81	**100.00**
Iron	83.40	**99.80**	98.88	97.85	**100.0**	99.49	84.70	98.26	**100.00**	**100.00**	**100.00**
Lamp	82.00	88.11	99.33	97.89	**99.5**	96.35	76.50	97.5	96.84	**100.00**	**100.00**
Phone	45.00	74.24	92.41	90.75	94.9	**96.88**	54.00	87.72	94.69	**100.00**	**100.00**
Mean	72.40	82.98	86.27	89.86	91.3	**93.59**	62.70	88.61	95.15	99.52	**99.91**
Time (ms)	100-167	36	40	30	1000	31	330	-	-	201.00	131.00

focus on the following datasets: Linemod [5], Occlusion [8] (LMO), TLESS [12] and Homebrewed [6] (HBD).

Linemod [5] is a classical dataset for object pose estimation. It is still an actively used benchmark even though it dates back to the pre-deep learning era. The datasets provides researchers with 13 3D object models. Each object comes with approximately 1200 annotated images. The images exhibit strong background clutter but almost no occlusions. In experiments with real training data, we used exactly the same training/testing split as in BB8 [68]. In the experiments with synthetic data, we used the entire datasets as test set, analogously to SSD6D [104] and DPOD [100]. We report the standard ADD recall [5] in all experiments, as it is the most widely reported metric for this dataset.

Occlusion [8] is an extension of the Linemod dataset, which consists of one Linemod sequence with 8 annotated objects. This dataset is a step forward to more complex datasets with occlusions of various degrees and a varying number of objects. We use the BOP subsequence of the dataset and report the Average Recall (AR) score of the BOP challenge. This allows us to fairly compare our method to all other methods participating in the challenge.

In spite of their popularity, Linemod and Occlusion do not have clearly separated train, validation and test splits. It leads to the situation when the images from the same sequence of frames are used for all the splits. It means that performance on the benchmarks might be significantly influenced by overfitting to the backgrounds, particular object poses, particular illumination setups and etc.

The authors of the Homebrewed [6] advocated the importance of training on synthetic data. Therefore, the datasets provides no train images, but has labeled validation set and a test set with hidden labels, as in other major machine learning datasets. Here, we also report the AR score from the BOP challenge. All ablation studies were conducted on the publicly available validation set.

The TLESS dataset [12] provides researchers with 30 objects and 20 scenes comprising of various number of objects. The dataset includes two kind of object models: precise

Table 6.4: Pose estimation performance on Linemod on depth and RGBD images: The table reports the percentages of correctly estimated poses w.r.t. the ADD score. The proposed detector shows state of the art results both if only real or only synthetic train data is used. Run times are provided as they are reported in the original papers using non-identical hardware.

Train Data						**Synthetic**			
Modality	**RGBD**	**Depth**	**Depth**	**RGBD**	**RGBD**	**RGBD**	**RGBD**	**RGB + D-Kabsch**	**RGB + D-Kabsch**
Method	AAE [116]	PPF [15]	PPF++ [131]	Ours	Ours	SSD6D [104]	Brachmann [8]	Ours	Ours
Refinement	ICP	ICP	ICP	-	ICP	ICP	iterative	ICP	-
Ape	20.55	86.50	98.50	87.54	91.91	-	-	98.38	**98.79**
Bvise	64.25	70.70	99.80	**99.92**	**99.92**	-	-	**99.92**	**99.92**
Cam	63.20	78.60	99.30	96.42	**97.84**	-	-	99.75	99.75
Can	76.09	80.20	98.70	98.16	98.66	-	-	**99.75**	99.58
Cat	72.01	85.40	99.90	99.41	99.83	-	-	**100.00**	**100.00**
Driller	41.58	87.30	93.40	97.55	97.64	-	-	**98.82**	**98.82**
Duck	32.38	46.00	98.20	90.82	94.26	-	-	98.41	**98.96**
Eggbox	98.64	97.00	98.80	98.48	98.80	-	-	**99.52**	**99.52**
Glue	96.39	57.20	75.40	99.26	99.34	-	-	**99.84**	99.75
Holep.	49.88	77.40	98.10	93.45	96.12	-	-	**97.81**	97.17
Iron	63.11	84.90	98.30	48.35	49.21	-	-	**100.00**	99.91
Lamp	91.69	93.30	96.00	50.12	50.20	-	-	99.02	**99.59**
Phone	70.96	80.70	**98.60**	96.70	97.02	-	-	98.15	98.07
Mean	64.67	78.86	96.38	88.94	90.06	90.90	98.3	99.18	**99.22**
Time (ms)	224	-	-	49	58	100	545	55	49

untextured CAD models and reconstructions of relatively low quality. There is a clear separation between train and test images, with train images not coming from the test domain. The dataset exhibits three main challenges: 1) texture-less objects, which are also similar to each other, 2) heavy occlusions and 3) various object symmetries.

As the proposed approach can be trained both on real and synthetic data and on data of different modalities, we experimented with each option given the availability of this data type. Table 6.1 summarizes which data types were used and whether synthetic or real data was used in each experiment. YOLO was trained only on RGB data as it was proven that in previous works that object detection networks can be relatively easy trained both on real and synthetic data thanks to transfer learning [179, 104]. The correspondence estimation network was trained either on RGB images or on depth maps.

Table 6.5: Pose estimation performance on Linemod on depth and RGBD images: The table reports the percentages of correctly estimated poses w.r.t. the ADD score. The proposed detector shows state of the art results both if only real or only synthetic train data is used. Run times are provided as they are reported in the original papers using non-identical hardware.

Train Data	**Real**									
Modality	**RGBD**	**RGBD**	**RGBD**	**RGBD**	**RGBD**	**RGBD**	**RGBD**	**RGBD**	**RGB + D-Kabsch**	**RGB + D-Kabsch**
Method	PointFusion [193]	DF [123]	DF [123]	Brachmann [8]	Ours	Brachmann [11]	Ours	PVN3D [117]	Ours	Ours
Refinement	-	-	DL [123]	-	-	iterative	ICP	-	ICP	-
Ape	70.40	79.50	92.30	-	97.34	-	97.62	97.30	97.72	**98.56**
Bvise	80.70	84.20	93.20	-	99.90	-	**100.00**	99.70	**100.00**	99.90
Cam	60.80	76.50	94.40	-	99.41	-	99.70	99.60	**99.90**	**99.90**
Can	61.10	86.60	93.10	-	99.41	-	99.51	99.50	**100.00**	99.80
Cat	79.10	88.80	96.50	-	99.90	-	99.90	99.80	**100.00**	**100.00**
Driller	47.30	77.70	87.00	-	97.52	-	98.02	99.30	**99.90**	**99.90**
Duck	63.00	76.30	92.30	-	94.74	-	97.09	98.20	98.31	**99.25**
Eggbox	99.90	99.90	99.80	-	99.63	-	99.72	**99.80**	99.72	99.72
Glue	99.30	99.40	100.00	-	99.71	-	99.71	**100.00**	**100.00**	**100.00**
Holep.	71.80	79.00	92.10	-	99.53	-	99.71	99.90	99.71	99.61
Iron	83.20	92.10	97.00	-	98.88	-	99.18	99.70	**100.00**	99.90
Lamp	62.30	92.30	95.30	-	99.14	-	99.04	99.80	**100.00**	99.90
Phone	78.80	88.00	92.80	-	99.43	-	99.43	99.50	**100.00**	**100.00**
Mean	73.70	86.20	94.30	98.10	98.81	99.00	99.13	99.40	99.64	**99.73**
Time (ms)	-	-	-	545	49	-	57	-	55	49

Table 6.6: Pose estimation performance comparison on the Occlusion dataset: Results are reported in terms of the Average Recall score. The results prove the effectiveness of the proposed approach on all used data modalities. Run times are provided as they are reported in the BOP challenge [7] using non-identical hardware. PPF-based methods, labeled with * in the Time column, use only CPU.

Train data	Data modality	Method	Refinement	VSD	MSSD	MSPD	AR	Time (s)
PBR	RGBD	CosyPose [10]	ICP	0.567	0.748	0.826	0.714	8.289
		Ours + Kabsch	-	0.565	0.748	0.788	0.700	0.334
		Ours + Kabsch	ICP	0.557	0.749	0.787	0.698	0.387
		PVNet [103]	ICP	0.502	0.683	0.73	0.638	-
		CDPN [73]	ICP	0.469	0.689	0.731	0.630	0.506
		Pix2Pose [101]	ICP	0.473	0.631	0.659	0.588	5.191
		Ours	ICP	0.472	0.621	0.654	0.582	0.398
		Ours	-	0.422	0.58	0.621	0.541	0.325
	RGB	Ours	4 calibrated views	0.572	0.735	0.777	0.695	2.479
		Ours	2 calibrated views	0.520	0.690	0.771	0.660	1.353
		CosyPose [10]	-	0.480	0.606	0.812	0.633	0.550
		Ours	-	0.432	0.560	0.761	0.584	0.274
		PVNet [103]	-	0.428	0.543	0.754	0.575	-
		CDPN [73]	-	0.393	0.537	0.779	0.569	0.279
		EPOS [119]	-	0.389	0.501	0.750	0.547	0.468
		Pix2Pose [101]	-	0.233	0.307	0.550	0.363	1.310
synt	RGB	EPOS [119]	-	0.29	0.38	0.659	0.443	0.487
		DPOD [100]	-	0.101	0.126	0.278	0.169	0.172
mix	RGBD	AAE [115]	ICP	0.208	0.218	0.285	0.237	1.197
	RGB	Multi-Path AAE	-	0.15	0.153	0.346	0.217	0.200
		AAE [115]	-	0.09	0.095	0.254	0.146	0.201
-	depth	Drost, PPF [15]	ICP	0.437	0.563	0.581	0.527	15.947*
		Drost, PPF [15]	ICP, 3D edges	0.409	0.525	0.542	0.492	3.389*
real	RGBD	Ours + Kabsch	ICP	0.557	0.739	0.773	0.69	0.393
		Ours + Kabsch	-	0.557	0.721	0.759	0.679	0.329
		Ours	ICP	0.478	0.62	0.66	0.586	0.453
		Ours	-	0.403	0.553	0.595	0.517	0.343
	RGB	Ours	4 calibrated views	0.598	0.751	0.793	0.714	1.326
		Ours	2 calibrated views	0.528	0.692	0.764	0.661	2.201
		Ours	-	0.427	0.55	0.728	0.568	0.278

In our experiments with the Homebrewed dataset, we completely relied on synthetically simulated train data as the dataset does not provide real train data. Real train depth data from the TLESS dataset was also not used as it is too simplistic and has no background depth to train the correspondence estimation network. We also experimented with ICP refinement to see how it compares to the proposed multi-view refiner. Exactly the same parameters of ICP were used on all datasets.

In the remaining of the section a following naming convention for out approach is used:

- *RGB* - both YOLO and CENet operate on RGB images, the final pose is computed using PnP+RANSAC;

- *RGBD* - YOLO operates on RGB images, while CENet takes a depth map as its input. The final pose is computed with Kabsch+RANSAC;

Table 6.7: Pose estimation performance comparison on the Homebrewed dataset: Results are reported in terms of the Average Recall score [7]. The results prove the effectiveness of the proposed approach on all used data modalities. Run times are provided as they are reported in the BOP challenge [7] using non-identical hardware. PPF-based methods, labeled with * in the Time column, use only CPU.

Train data	Data modality	Method	Refinement	VSD	MSSD	MSPD	AR	Time (s)
PBR	RGBD	Ours + Kabsch	ICP	0.784	0.872	0.879	0.845	0.329
		Ours + Kabsch	-	0.796	0.856	0.867	0.84	0.238
		CosyPose [10]	ICP	0.679	0.719	0.737	0.712	5.326
		CDPNv2 [73]	ICP	0.629	0.757	0.749	0.712	0.713
		Pix2Pose [101]	ICP	0.64	0.721	0.724	0.695	3.248
		Ours	ICP	0.462	0.468	0.473	0.468	0.47
		Ours	-	0.344	0.398	0.403	0.382	0.245
	RGB	Ours	4 calibrated views	0.79	0.857	0.86	0.835	0.584
		Ours	2 calibrated views	0.754	0.84	0.858	0.818	0.926
		CosyPose [10]	8 uncalibrated views	0.691	0.744	0.804	0.746	0.427
		Ours	-	0.642	0.704	0.828	0.725	0.163
		CDPNv2 [73]	-	0.614	0.708	0.845	0.722	0.273
		CosyPose [10]	4 uncalibrated views	0.646	0.685	0.756	0.696	0.445
		CosyPose [10]	-	0.613	0.634	0.721	0.656	0.417
		EPOS [119]	-	0.484	0.527	0.729	0.58	0.657
synt	RGBD	Sundermeyer-IJCV19 [115]	ICP	0.479	0.506	0.533	0.506	1.352
	RGB	Sundermeyer-IJCV19 [116]	-	0.273	0.306	0.461	0.346	0.19
		DPOD [100]	-	0.218	0.262	0.379	0.286	0.18
-	Depth	Vidal-Sensors18 [194]	ICP	0.707	0.704	0.708	0.706	2.608*
		Drost-CVPR10-3D-Edges [15]	ICP, 3D edges	0.618	0.624	0.626	0.623	127.372*
		Drost-CVPR10-3D-Only [15]	ICP	0.593	0.627	0.627	0.615	16.136*
	RGBD	Drost-CVPR10-Edges [15]	ICP,3D edges, images	0.677	0.665	0.67	0.671	144.029*

- *RGB + D-Kabsch* - YOLO and CENet use RGB as input, but then the predicted correspondences are projected into 3D, and the pose is computed with Kabsch+RANSAC.

6.4.2 Results

Linemod. We start this section by discussing the quantitative results of single object pose estimation on the Linemod dataset. Table 6.2 and Table 6.3 present the accuracy of pose estimation on monocular RGB images of methods trained on syhtetic and real data respectively. We report run times of each algorithm if the corresponding papers clearly state them. Run times are taken from the original papers, meaning that they are obtained using non-identical hardware. The table reports the percentages of correctly estimated poses w.r.t. the ADD score. The tables compare separately methods trained on synthetic data and methods trained on real data and methods with or without refinement. On synthetic data, the original DPOD reached the state of the art results beating the second best method almost by a factor of 2. The proposed approach further improved the results, surpassing DPOD by over 30%, making the ADD score comparable even to ADD scores of some of the methods trained on real data. If real training data is used, the proposed approach reaches the best results among other methods trained on real data. The proposed multi-view refiner clearly outperforms all other RGB-based refiners on the dataset even if only 2 views are used for refinement. It also proves that the refinement method handles symmetric object, such as Glue and Eggbox, well.

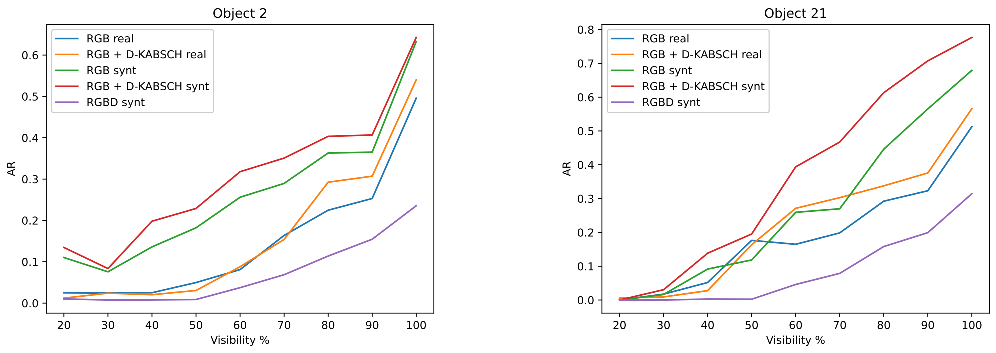


Figure 6.5: Dependence of quality of pose estimation on object's visibility. Pose quality is reported in terms of the Average Recall [7] on objects 2 and 21 from the TLESS dataset. The plots show that correspondence prediction works from RGB works considerably more reliably than from depth in case of large occlusions. It also shows that synthetic training data allows for more reliable pose estimation of occluded objects.

Table 6.4 and Table 6.5 present quantitative results on Linemod datasets in case depth data is available for methods trained on synthetic and real data respectively. Only PPF, PointFusion, DenseFusion and our method use depth as input, all other methods use it only for pose refinement. Our detector easily performs on par or outperforms other methods even if no refinement is applied. It also beats the PPF-based solutions. Another important conclusion from these tables is relative independence of pose accuracy on whether synthetic or real data or which data modality was used for training. Due to the simplicity of the dataset and to the lack of occlusions, correspondences predicted from depth maps prove to be good enough for excellent pose accuracy if the Kabsch algorithm is used. In comparison do deep learning methods which utilize real train data [123, 193], we did not explicitly rely on sensor fusion and used purely RGB or purely depth for correspondence estimation. This experiment also demonstrates the effectiveness of our proposed refiner, which easily reaches the performance scores of the best RGBD methods.

Occlusion. Moving to a more difficult scenario, where a varying number of objects is present in each image, we start evaluation with experiments on the Occlusion dataset. In this case, the ability to predict accurate segmentation masks and precise correspondences becomes of critical importance. The results for all types of detectors and training data types are summarized in Table 6.6. The table clearly shows that our method outperforms all other dense correspondence methods, e.g. CDPN, EPOS and Pix2Pose, while being only worse than CosyPose if PBR training data and no refinement were used. With the multi-view refinement our methods easily outperform all other RGBD methods and shows similar or slightly worse results than the top performing RGBD methods with ICP refinement. It also always improves the poses and the final score in contrast to ICP which worsened the AR score for some of the experiments.

Adding depth information and predicting segmentations and correspondences from depth rather than from RGB actually hurts the performance, as can be seen in the

Table 6.8: Pose estimation performance comparison on the BOP images of the TLESS dataset: Results are reported in terms of the Average Recall score [7]. The results prove the effectiveness of the proposed approach on all used data modalities. Run times are provided as they are reported in the BOP challenge [7] using non-identical hardware. PPF-based methods, labeled with * in the Time column, use only CPU.

Train data	Data modality	Method	Refinement	VSD	MSSD	MSPD	AR	Time (s)
PBR	RGBD	Ours + Kabsch	-	0.646	0.716	0.736	**0.699**	0.320
		Ours + Kabsch	ICP	0.485	0.632	0.652	0.590	0.523
		CDPNv2 [73]	ICP	0.368	0.449	0.488	0.435	2.486
		Ours	ICP	0.277	0.320	0.339	0.312	0.688
		Ours	-	0.251	0.303	0.312	0.289	0.330
	RGB	Ours	2 calibrated views	0.634	0.707	0.728	0.690	1.293
		Ours	4 calibrated views	0.645	0.710	0.712	0.689	0.886
		CosyPose [10]	-	0.571	0.589	0.761	0.64	0.493
		Ours	-	0.561	0.602	0.744	0.636	0.328
		EPOS [119]	-	0.38	0.403	0.619	0.467	1.992
		CDPNv2 [73]	-	0.303	0.338	0.579	0.407	1.849
synt	RGB	EPOS [119]	-	0.369	0.423	0.635	0.476	1.177
		DPOD [100]	-	0.048	0.055	0.139	0.081	0.206
-	Depth	Vidal-Sensors18 [194]	ICP	0.464	0.575	0.574	0.538	7.063*
		Drost-CVPR10-3D-Only [15]	ICP	0.375	0.478	0.480	0.444	9.204*
		Drost-CVPR10-3D-Edges [15]	ICP, 3D edges	0.370	0.422	0.420	0.404	62.507*
real	RGBD	Drost-CVPR10-Edges [15]	ICP, 3D edges, images	0.469	0.512	0.518	0.500	70.914*
		Ours + Kabsch	-	0.537	0.557	0.586	**0.56**	0.315
		Pix2Pose [101]	ICP	0.438	0.548	0.549	0.512	4.180
		Ours + Kabsch	ICP	0.416	0.516	0.537	0.490	0.578
	RGB	Ours	4 calibrated views	0.467	0.541	0.563	0.524	0.727
		Ours	2 calibrated views	0.461	0.530	0.578	0.523	0.973
		Ours	-	0.469	0.491	0.595	0.518	0.367
		Pix2Pose [101]	-	0.261	0.296	0.476	0.344	1.084
		Pix2Pose-Original-ICCV19 [101]	-	0.214	0.238	0.432	0.295	1.522
mix	RGB	CosyPose [10]	8 uncalibrated views	0.773	0.836	0.907	0.839	0.969
		CosyPose [10]	4 uncalibrated views	0.742	0.795	0.864	0.801	0.792
		CosyPose [10]	-	0.669	0.695	0.821	0.728	0.451
		Ours	4 calibrated views	0.679	0.742	0.740	0.72	0.909
		Ours	2 calibrated views	0.665	0.739	0.753	0.719	1.306
		Ours	-	0.579	0.621	0.764	0.655	0.306
		CDPN [73]	-	0.377	0.418	0.674	0.49	0.708
		CDPNv2 [73]	-	0.386	0.426	0.62	0.478	1.852
		Sundermeyer-IJCV19 [116]	-	0.196	0.211	0.504	0.304	0.194
	RGBD	Ours + Kabsch	-	0.665	0.738	0.756	0.720	0.7367
		CosyPose [10]	ICP	0.587	0.749	0.767	0.701	0.274
		Ours + Kabsch	ICP	0.503	0.656	0.672	0.610	0.455
		Sundermeyer-IJCV19 [116]	ICP	0.459	0.489	0.514	0.487	0.531
		CDPNv2 [73]	ICP	0.385	0.489	0.516	0.464	2.645

RGBD block of the table. It indicates inferior quality of the predicted segmentations and correspondences, that are not sufficient for reliable and precise pose estimation. Additionaly, due to imprecise segmentations, Kabsch and ICP can potentially align the object to nearby objects. On the other hand, if depth is only used for the Kabsch algorithm with the correspondences predicted by the RGB correspondence network, a larger performance boost is observed. In this configuration, the proposed method sets the state-of-the-art results on the HBD dataset even without the ICP refinement. ICP boosts the results even further. This experiment and later the experiments on TLESS show that correspondence and segmentaton prediction from pure depth maps works well only on very simple datasets such as Linemod, where the object is not occluded and clearly stands out from the scene. With the more challenging datasets, RGB-based

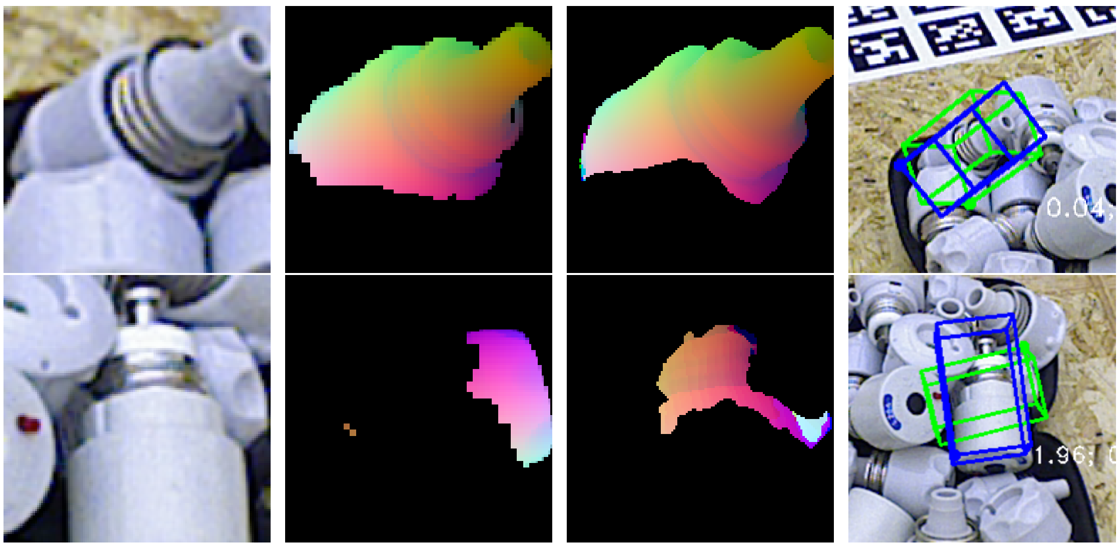


Figure 6.6: Success cases and failure cases. Top row provides an example of successfully estimated segmentation mask and NOCS correspondences even in case of occlusions and similar objects present in the image patch. The bottom row illustrates a failure case, where the CENet is confused by occlusions and similar objects belonging to other object classes, which leads to an incorrect estimated pose. The green cuboid represents the ground truth pose (up to a symmetry transformation), whereas the blue cuboid represents the estimated pose.

correspondence estimation starts to dominate due to richer RGB information available for the network.

Homebrewed. This dataset is the next step towards more complicated pose estimation benchmarks, as it consists of a larger number of more diverse objects and a varying degree of occlusions. The comparison is provided in Table 6.7. Pose accuracy is reported in terms of Average Recall as defined in [7]. As indicated before in Table 6.1, we used exclusively synthetic PBR train data data. Our method easily outperforms all other RGB methods. With the multi-view refinement, it performs on par with the best RGBD methods.

TLESS. Table 6.8 compares Average Recall of various configurations of the proposed detector to other participants of the BOP challenge 2020. Since TLESS dataset comes with a separate train set, we were able to test various combinations of YOLO and CENet. Our approach consistently outperforms all competing methods apart from CosyPose. It is interesting that usage of depth does improve the average recall as much as on other datasets. The lack of performance boost can be partially explained by the fact that approximately 16% of images are taken by a camera looking almost parallel to the table with the objects. In this case, Primesense produces mostly empty depth maps for flat surfaces and empty on edges of objects, making pose prediction from occluded objects close to impossible. This also explains why in some cases AR score is lower if ICP

Table 6.9: Pose estimation performance on Linemod on different data modalities and types of train data: The table reports the percentages of correctly estimated poses w.r.t. the ADD score. An ADD score for each data modality and train data type is provided separately on ground truth 2D detections and on YOLO detections. Symmetry-aware median L_2 correspondence error demonstrates the quality of predicted correspondences.

Dataset	Data modality	Data type	Bboxes	Refinement	ADD	Corr. Err.
Linemod	RGB	real	GT	-	93.99	2.48
				2 views	97.42	
				4 views	99.79	
			YOLO	-	93.59	2.52
				2 views	97.68	
				4 views	99.91	
		synt	GT	-	81.20	4.61
				2 views	96.47	
				4 views	99.63	
			YOLO	-	80.39	4.67
				2 views	96.33	
				4 views	99.70	
	RGBD	real	GT	-	99.85	5.31
				ICP	99.85	
			YOLO	-	98.81	5.38
				ICP	99.13	
		synt	GT	-	97.99	15.81
				ICP	98.19	
			YOLO	-	88.94	17.32
				ICP	90.06	
	RGB + D-Kabsch	real	GT	-	99.99	2.48
				ICP	99.99	
			YOLO	-	99.73	2.52
				ICP	99.64	
		synt	GT	-	99.94	4.61
				ICP	99.97	
			YOLO	-	99.22	4.67
				ICP	99.18	

refinement is applied. Again, the proposed multi-view refinement consistently improves the poses even in the presence of large occlusions and imperfect initial poses.

Another important insight from Table 6.8 and from Table 6.16 is that the correspondence estimation network trained on synthetic data actually outperforms the one trained on real data provided they are tested on exactly the same patches. There are several

Table 6.10: Pose estimation performance on the Occlusion dataset on different data modalities and type of train data: The table reports in terms of the AR score [7]. An AR score for each data modality and train data type is provided separately on ground truth 2D detections and on YOLO detections. Symmetry-aware median L_2 correspondence error demonstrates the quality of predicted correspondences.

Dataset	Data modality	Data type	Bboxes	Refinement	AR	Corr. Err.
LMO	RGB	real	GT	-	0.601	13.830
				2 views	0.690	
				4 views	0.751	
			YOLO	-	0.568	16.150
				2 views	0.661	
				4 views	0.714	
		synt	GT	-	0.649	12.275
				2 views	0.741	
				4 views	0.773	
			YOLO	-	0.584	12.987
				2 views	0.660	
				4 views	0.695	
	RGBD	real	GT	-	0.534	26.482
				ICP	0.609	
			YOLO	-	0.517	26.611
				ICP	0.586	
		synt	GT	-	0.606	21.267
				ICP	0.657	
			YOLO	-	0.582	22.356
				ICP	0.541	
	RGB + D-Kabsch	real	GT	-	0.738	13.830
				ICP	0.757	
			YOLO	-	0.679	16.150
				ICP	0.690	
		synt	GT	-	0.796	12.987
				ICP	0.793	
			YOLO	-	0.582	22.356
				ICP	0.698	

explanations to that. The real train data is very limited in the number of images and requires a lot of data augmentation to work on the real test set, while the synthetic data has considerably more images. Additionally, simulated synthetic train data naturally exhibits realistically looking occlusions, which helps generalize to the test images. It is also confirmed by the plot in Figure 6.5, where CENet trained on synthetic data consistently outperforms the one trained on real data. If ground truth patches are used, the synthetic

Table 6.11: Different view sampling strategies for the multi-view refiner on the Linemod [5] dataset.

Dataset	Data type	N views	Views sampling	ADD
Linemod	real	-	-	93.59
		2 views	Closest	86.43
			Random	99.52
			Furthest	99.71
		4 views	Closest	96.53
			Random	99.91
			Furthest	99.97
	synt	-	-	81.20
		2 views	Closest	87.45
			Random	98.95
			Furthest	99.02
		4 views	Closest	94.65
			Random	99.70
			Furthest	99.80

network achieves AR of 0.729, whereas the real network only 0.535. On the other hand, YOLO trained on real data outperforms YOLO train on synthetic data, which can be seen in much lower AR drop if the full real pipeline is used. As a result, we achieve the best result on TLESS if we combine a real YOLO and a synthetic correspondence estimation network, analogously to how it is done in AAE [115].

6.4.3 Ablation Studies

In the ablation studies, we measure what influences pose estimation the most, how imprecise YOLO detections affect the quality of correspondences and which data modality is better. Tables 6.9, 6.10, 6.15 and 6.16 report how pose estimation performance drops if YOLO detections are used instead of ground truth bounding boxes on Linemod and Occlusion, Homebrewed and TLESS datasets respectively. Those tables essentially report the upper bound of what the proposed approach can achieve, assuming perfect 2D detections, and how far from them the results actually are. They also additionally report quality of correspondences. In each object in the image, a correspondence quality is computed as median per-correspondence L2 error between a predicted correspondence and its correct ground truth location. Then, it is averaged across all objects. The metric takes into account object symmetries by choosing an equivalent symmetric pose.

RGB vs Depth vs RGB + D-Kabsch. With the advent of deep learning, much of pose estimation research has moved to monocular RGB [104, 115, 100, 102, 10]. Even if depth information was used, it was used in a separate post-processing step for pose

Table 6.12: Different view sampling strategies for the multi-view refiner on the Occlusion [11] dataset.

Dataset	Data type	N views	Views sampling	AR
LMO	real	-	-	0.568
		2 views	Closest	0.627
			Random	0.661
			Furthest	0.648
		4 views	Closest	0.693
			Random	0.714
			Furthest	0.706
	synt	-	-	0.584
		2 views	Closest	0.627
			Random	0.660
			Furthest	0.664
		4 views	Closest	0.693
			Random	0.695
			Furthest	0.687

refinement and not for inference. In this chapter, we introduced a pose estimation network capable of pose estimation from either modality, which leads to the question of which one is better. On the Linemod dataset (Table 6.9), RGB-only detectors still lag behind the depth-based counterparts especially if only synthetic data is used for their training. It is also visible from the table that correspondences predicted from RGB tend to be better than those predicted from depth even for such a simplistic dataset. Table 6.9 clearly shows the superior quality of correspondences estimated from RGB, Nevertheless, depth-based approaches still better than purely RGB approaches because of the simplicity of the dataset and the lack of occlusions.

On the other hand, the situation is different on more complicated datasets with occlusions and background clutter. On the Occlusion dataset, the RGB version of CENet works better than the depth version. Depth brings improvement only if the correspondences are computed from RGB images and then projected onto the point cloud for the Kabsch algorithm. This is also confirmed by the correspondence error analysis in Table 6.10, where correspondences computed from RGB are almost two times more precise than the once computed from depth maps. This trend is also noticeable on the more complicated Occlusion dataset (Table 6.10). It is clear from the table that the correspondence error is much larger due to occlusions. As seen from Table 6.7, on the Homebrewed dataset the proposed approach trained on RGB only significantly outperforms its counterpart trained on depth, while the RGB variant with the Kabsch algorithm shows outstanding AR. That is explained by less accurate predictions of segmentation masks and correspondences for occluded objects when only depth information is used for

Table 6.13: Different view sampling strategies for the multi-view refiner on the Homebrewed [6] dataset.

Dataset	Data type	N views	Views sampling	AR
HBD	synt	-	-	0.668
		2 views	Closest	0.693
			Random	0.808
			Furthest	0.762
		4 views	Closest	0.724
			Random	0.783
			Furthest	0.775

that. Table 6.15 clearly shows that correspondences predicted from RGB are almost five times better then their depth-based counterparts, which explains the difference in pose estimation performance. On TLESS dataset, as seen from Table 6.8 and Table 6.16, using depth again does not help at all as the quality of correspondences deteriorate. Depth CENet also produced visually worse segmentations. This phenomenon can be explained by larger degrees of occlusion and camera angles resulting in missing or very noisy depth values. Figure 6.5, additionally proves strong dependence of pose estimation quality from depth maps on the visibility of objects. It show that RGB-based CENet is much better capable of handling occlusions that the depth-based CENet.

To sum up, semantic segmentation and correspondence estimation is considerably more precise when done completely from RGB. If depth information is present, it is more beneficial to still predict them from RGB and then project into the 3D space and than use the Kabsch algorithm. Such an approach potentially also allows for alternating between PnP and Kabsch based on the quality of depth maps.

Real vs Synthetic Data. As an experiment from [6] showed, training pose estimation networks on real RGB data tends to outperformed the one on synthetic if images and poses come from exactly the same domain. That is still clearly visible on Linemod dataset in Table 6.2. Detectors trained on synthetic data are still behind their real counterparts even in spite of the progress in synthetic data generation. It is also visible from the worse quality of correspondences as seen in Table 6.9. The same holds for CENet train on synthetic depth data. On the other hand, as can be seen from TLESS Tables 6.8 and 6.16 YOLO benefits a lot from training on real data even if the train data is out of test set domain. At the same time, CENet trained on synthetic data clearly outperforms the CENet trained on real data both in terms of AR and the quality of correspondences due to better pose space coverage.

To sum up, real train data is better for training 2D detectors. For training correspondence estimation network, synthetic data is beneficial unless train and test data come from the same domain and contain objects in very similar poses.

Choice of PnP and RANSAC. In all the reported RGB results we relied on the standard implementation of EPnP [1] and of RANSAC from OpenCV, although un-

Table 6.14: Different view sampling strategies for the multi-view refiner on the TLESS [12] dataset.

Dataset	Data type	N views	Views sampling	AR
TLESS	real	-	-	0.518
		2 views	Closest	0.524
			Random	0.523
			Furthest	0.534
		4 views	Closest	0.544
			Random	0.524
			Furthest	0.540
	synt	-	-	0.636
		2 views	Closest	0.693
			Random	0.690
			Furthest	0.694
		4 views	Closest	0.712
			Random	0.689
			Furthest	0.707
	mix	-	-	0.655
		2 views	Closest	0.715
			Random	0.719
			Furthest	0.725
		4 views	Closest	0.738
			Random	0.720
			Furthest	0.742

countable number of improved versions of PnP and RANSAC have been published in the past [166, 163, 161]. Additionally, EPOS [119] reported, that a large improvement can be achieved by simple switching from OpenCV PnP+RANSAC to Graph Cut RANSAC [163] with DLS-PnP [152]. We have tried this combination on scene 3 of TLESS. It resulted in increase of AR from 21.45 to, at most, 24.02. However, we faced two complications. First, due to a large number of correspondences, RANSAC takes over a second per each object, making the entire pipeline slow and not applicable in practice. Secondly, the algorithm seemed to be very sensitive to hyper-parameters, and optimal hyper-parameters are not shared among the objects. As a result, OpenCV implementation proved to be more rapid and robust, even tough potentially resulting in sub-optimal AR scores.

Limitations of CENet. There are two main limiting factors of the CENet. The first comes from the fact that the reliable pose estimation with correspondences requires their good quality and the right number of them to deal with outliers. Therefore, large

Table 6.15: **Pose estimation performance on the Homebrewed on different data modalities and type of train data**: The table reports the percentages of correctly estimated poses w.r.t. the Average Recall score [7]. An AR score for each data modality and train data type is provided separately on ground truth 2D detections and on YOLO detections. Symmetry-aware median L_2 correspondence error demonstrates the quality of predicted correspondences.

Dataset	Data modality	Data type	Bboxes	Refinement	AR	Corr. Err.
HBD	RGB	synt	GT	-	0.723	9.52
				2 views	0.808	
				4 views	0.807	
			YOLO	-	0.668	13.78
				2 views	0.766	
				4 views	0.783	
	RGBD	synt	GT	-	0.375	52.69
				ICP	0.453	
			YOLO	-	0.362	54.10
				ICP	0.438	
	RGB + D-Kabsch	synt	GT	-	0.843	9.52
				ICP	0.853	
			YOLO	-	0.793	13.78
				ICP	0.801	

occlusions might be problematic, as indicated in Figure 6.5 and Figure 6.6. However, pose estimation of occluded object is a common problem for all methods. Another issue, illustrated in Figure 6.5 , arises when CENet receives a patch containing similarly looking objects or object of the same class.

Multi-View Refinement We have experimented with the multi-view refinement on all datasets, and on all datasets it proved to be effective, as opposed to ICP which did not always improve the poses. It can be explained by its independence from the quality of depth data and potentially better handling of objects which are occluded in only some of the frames used for refinement. We split each sequence into non-overlapping sets of 2 or 4 images used for refinement. The pose is optimized jointly and then transformed into the coordinate system of each frame. Even though the number of frames is lower than in CosyPose [10], the proposed refiner anyway produces better or similar relative improvement. This is also explained by the fact that CosyPose optimized camera and object poses jointly while we assume the availability of relative camera poses. The CosyPose refiner performs noticeably better on TLESS, but this can be explained by a much better initial poses produced by CosyPose. For each set, we compute relative camera poses from individual ground truth camera poses. In these experiments, the images were split into batches absolutely at random.

Frame selection strategy In this experiment, we tested how selection of the frames for joint refinement affects the overall performence of the refiner. Tables 6.11,6.12,6.13,6.14 summarize these experiments. We experimented with three different strategies: closest

Table 6.16: Pose estimation performance on the TLESS on different data modalities and type of train data: The table reports the percentages of correctly estimated poses w.r.t. the Average Recall score [7]. An AR score for each data modality and train data type is provided separately on ground truth 2D detections and on YOLO detections. Symmetry-aware median L_2 correspondence error demonstrates the quality of predicted correspondences.

Dataset	**Data modality**	**Data type**	**Bboxes**	**Refinement**	**AR**	**Corr. Err.**
TLESS	RGB	real	GT	-	0.535	26.946
				2 views	0.542	
				4 views	0.548	
			YOLO	-	0.518	27.516
				2 views	0.523	
				4 views	0.524	
		synt	GT	-	0.729	10.400
				2 views	0.798	
				4 views	0.795	
			YOLO	-	0.636	20.616
				2 views	0.689	
				4 views	0.690	
		mix	YOLO	-	0.655	18.373
				2 views	0.719	
				4 views	0.720	
	RGBD	synt	GT	-	0.322	34.396
				ICP	0.354	
			YOLO	-	0.289	41.087
				ICP	0.312	
	RGB + D-Kabsch	real	GT	-	0.583	26.946
				ICP	0.513	
			YOLO	-	0.560	27.516
				ICP	0.490	
		synt	GT	-	0.812	10.400
				ICP	0.678	
			YOLO	-	0.699	20.616
				ICP	0.590	
		mix	YOLO	-	0.720	18.373
				ICP	0.610	

view sampling, random sampling and furthest view sampling. Closest view sampling corresponds to the worst case scenario because it does not provide strong enough geometric constraints. In other words, an error in pose estimation from the reference frame might not be necessarily be visible in the other frames used for refinement. Occlusions will cause further problems.

As expected, the closest view sampling performs the worst among these strategies, but poses still improve in most of the cases. Random and furthest sampling tend to perform similarly to each other. This is explained that furthest views do not necessarily correspond to the best possible configuration, as the error might not be visible in them. Additionally, furthest view sampling might result in selecting views which are too far from the reference frame, introducing new occlusions.

6.5 Conclusions

In this chapter, we extend the Dense Pose Object Detector (DPOD) method, by splitting its inference into three stages with a optional fourth refinement stage. In the first stage, YOLO outputs 2D bounding boxes of the objects of interest. In the second stage, our unified CENet network predicts foreground object masks and dense 2D-3D correspondences between image pixels and corresponding 3D models either from RGB or depth input modalities with the same architecture. Object's 6 DoF is then reliably estimated with PnP+RANSAC or with Kabsch+RANSAC depending on the available data modality in the third stage. In the optional fourth stage, predicted poses are refined either with ICP or with the proposed novel multi-view refiner. The second stage network can be trained both on RGB images and on depth masks. We additionally proposed a new multi-view pose refiner based on differentiable rendering, which is used to produce a pose which is globally consistent with dense NOCS correspondences predicted in all frames. The proposed approaches have been tested on four popular datasets, each of which is challenging in its own way: Linemod, Occlusion, Homebrewed and TLESS. The detector shows excellent results on all the datasets and data modalities while still staying fast and scalable. The refiner improves the poses even further while still being reasonably fast. We provide an extensive evaluation and measured effects of the choice of data modality, and of the choice of the training data: synthetic or real. The overall results indicated that RGB is good for correspondence prediction, while depth is good for pose prediction. The more occlusions we have in the data the more benefits we get from the dense correspondences estimated from RGB. In that case, noisy depth still improves the pose with respect to the pose estimated from RGB only.

7 One-Shot Pose Estimation Without Re-Training

In this chapter, we present a novel one-shot method for object detection and 6 DoF pose estimation, that does not require training on target objects. At test time, it takes as input a target image and a textured 3D query model. The core idea is to represent a 3D model with a number of 2D templates rendered from different viewpoints. This enables CNN-based direct dense feature extraction and matching. The object is first localized in 2D, then its approximate viewpoint is estimated, followed by dense 2D-3D correspondence prediction. The final pose is computed with PnP. We evaluate the method on LineMOD, Occlusion, Homebrewed, YCB-V and TLESS datasets and report very competitive performance in comparison to the state-of-the-art methods trained on synthetic data, even though our method is not trained on the object models used for testing.

7.1 Introduction

The rapid development of high quality 6 DoF pose estimation methods is underway. According to the BOP challenge [7], which combines publicly available 6 DoF pose estimation datasets and offers standardized evaluation and comparison procedures, the field is dominated by deep learning methods [100, 10, 73, 101, 102, 119, 103, 105, 195, 115, 82, 104, 118, 8, 72, 10, 127, 196, 197]. The methods' performance is, however, limited by the availability of labeled training data. Accurate 6 DoF pose annotation of real data is a complicated and time-consuming process [6], that must be repeated manually for each new object. This severely limits the practical applicability of 6 DoF pose estimation methods. Additionally, labels can exhibit imperfection [198]. As synthetic data preparation tools improved, more methods shifted to training on synthetically rendered images. This greatly simplifies the preparation of data for new objects. These time-consuming and computationally-intensive data rendering and model training steps, on the other hand, must be repeated for each new target object of interest.

While one-shot object detection, i.e., detection of novel objects not seen during training, appears to yield promising results for conventional 2D detection, its extensions to pose estimation were very limited. There are very few related works that attempt to generalize to new objects. They primarily focus on objects from the same category [74, 125, 126], objects with very similar geometry [80, 81], rely on partially training on target objects [115], or limit the task to viewpoint estimation [140]. We extend the one-shot object detection ideas to estimate the full 6 DoF pose. Our method is trained

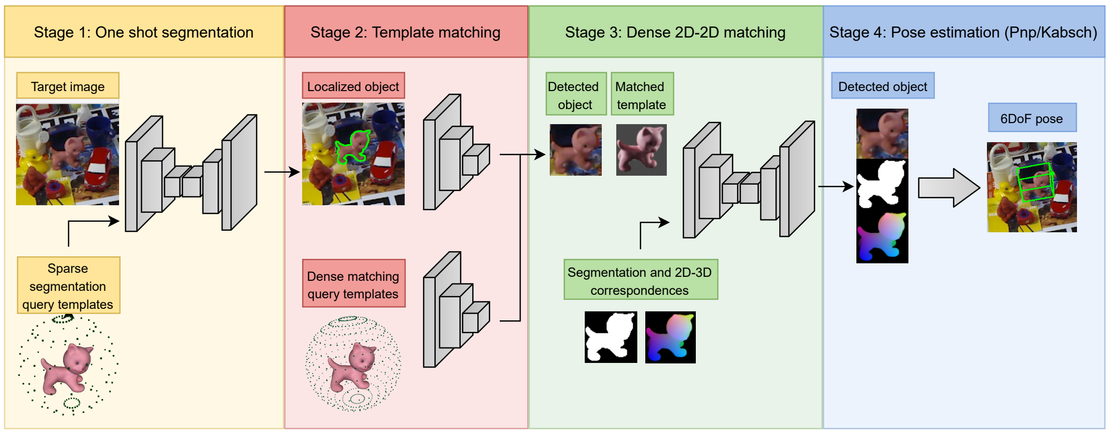

Stage 1: One shot segmentation
Target image
Sparse segmentation query templates
Stage 2: Template matching
Localized object
Dense matching query templates
Stage 3: Dense 2D-2D matching
Detected object
Matched template
Segmentation and 2D-3D correspondences
Stage 4: Pose estimation (Pnp/Kabsch)
Detected object
6DoF pose

Figure 7.1: Pipeline of the proposed detector. 1) One-shot object localization conditioned on the 3D model. 2) Initial viewpoint estimation by template matching. 3) Dense 2D-2D matching between the image patch and the matched template. 4) 6 DoF pose estimation with PnP+RANSAC or Kabsch+RANSAC. The proposed pose estimation pipeline generalizes well to new target objects not seen duing training.

only once and then automatically generalizes to new objects without training on them, obviating the need for synthetic or real data preparation and training for new objects.

The 4-stage pipeline of the proposed approach is visualized in Figure 7.1. The input to the method is a test image and a textured 3D model of the target object of interest. Inspired by OS2D [14], the method relies on dense sliding window-based feature correlation between the object, represented with a sparse set of 2D templates obtained by rendering the object from various viewpoints, and the input test image. In the first stage, one-shot object segmentation is performed. The detected object is matched to a database of object renderings in the second stage to perform initial viewpoint estimation. In the third stage, a network estimates dense 2D-2D correspondences between pixels of the input image patch and the matched template, whose pose is known. This provides us with 2D-3D correspondences between the pixels of the input image and the 3D model. This enables 6 DoF pose estimation using PnP[1] or Kabsch[199] with RANSAC [78] in the last stage.

The evaluation of our approach on five datasets (LineMOD, Occlusion, HomebrewDB, YCB-V and TLESS) proves that it fully generalizes to new objects and scenes not seen during training. Our key contributions include: 1) The first RGB-based one-shot pose estimation pipeline that truly scales to new objects without training on them. This results in a considerable time reduction required for synthetic data generation and retraining. 2) A novel architecture and a novel attention mechanism for one-shot semantic segmentation; 3) An architecture for dense 2D-2D matching that enables 2D-3D correspondence transfer from a template with known 6 DoF pose to a target image with unknown pose.

7.2 Methodology

One-shot methods for 2D object detection use a target RGB image and a query template of the object of interest as input during inference, neither of which was seen during training. The inputs in our method are a target RGB image and a 3D model of the object. The 3D model is represented by a set of 2D query templates rendered from various virtual camera viewpoints placed on a sphere around the object. Our pipeline consists of four stages as summarized in Figure 7.1 with stages object segmentation (1), template matching (2), 2D-2D matching (3), and pose estimation (4). Stages 3 and 4 can potentially be executed several times to produce multiple pose hypotheses.

In the following, we use I of size $H \times W$ to denote an image, which implicitly depends on the object model $\mathcal{M}$ and its pose $T \in SE(3)$. A feature extractor $F_{FE}^{k}(I) \in \mathbb{R}^{H^k \times W^k \times D^k}$ uses a pre-trained network to extract feature maps of depth dimension D^k from the input image. We use pre-computed feature maps from several depth levels of the network, which are indexed by $\{k \in \mathbb{N} \mid 1 \leq k \leq N\}$. $\bar{F}_{FE}^{k}(I) \in \mathbb{R}^{D^k}$ stands for a feature extractor which extends F_{FE}^{k} by spatial averaging along height and width for each depth dimension of the feature map to produce a single vector of length D^k.

7.2.1 One-Shot Segmentation

The first stage network takes an image and a descriptor of the 3D model and predicts a binary segmentation mask indicating the location of the object' visible part. The core idea is to describe a textured CAD model using a set of viewpoint-based templates generated by rendering the object in various rotations. This brings the problem closer to the standard 2D one-shot methods. The key difference is that we compute a single descriptor based on pre-rendered templates, allowing the image to be matched to all of the templates in a single shot, as opposed to the standard one-shot object detection, which treats each query template independently. Figure 7.2 depicts the overall architecture of the network.

We build on the concept of dense feature matching first, which was first proposed in [200] and later extended for the task of object detection in [14]. The basic idea is to compute per-pixel correlations between feature maps of the target image and the features from the object descriptor. Feature precomputation is visualized in Figure 7.3. Pre-computed image features $\mathbf{f}^k = F_{FE}^{k}\left(I\right) \in \mathbb{R}^{H^k \times W^k \times D^k}$ and a 4D descriptor tensor $\mathbf{o}^k = F_{FE}\left(\mathcal{M}\right) \in \mathbb{R}^{X^k \times Y^k \times Z^k \times D^k}$ for the object $\mathcal{M}$ are compared. The 4D descriptor tensor $\mathbf{o}^k$ collects all templates rendered from the virtual viewpoints on the sphere around the object, where the first two dimensions (X, Y) stand for a camera position w.r.t. the object coordinate system using polar coordinates, while the third dimension Z stands for in-plane rotations. The 4th dimension D^k refers to feature maps extracted from each template at multiple depth levels of the neural network indexed by k. Each element of the tensor is one viewpoint template represented with the corresponding feature vector. It is defined as $\mathbf{o}_{x,y,z}^{k} = \bar{F}_{FE}^{k}\left(I\left(R\left(x, y, z\right) \cdot \mathcal{M}\right)\right)$, where R is a rotation matrix representing a virtual viewpoint on the sphere. Each pixel in the feature map $\mathbf{f}^k$ is matched to the entire object descriptor $\mathbf{o}^k$, resulting in a correlation tensor $\mathbf{c}^k \in \mathbb{R}^{H^k \times W^k \times X^k \times Y^k \times Z^k}$.

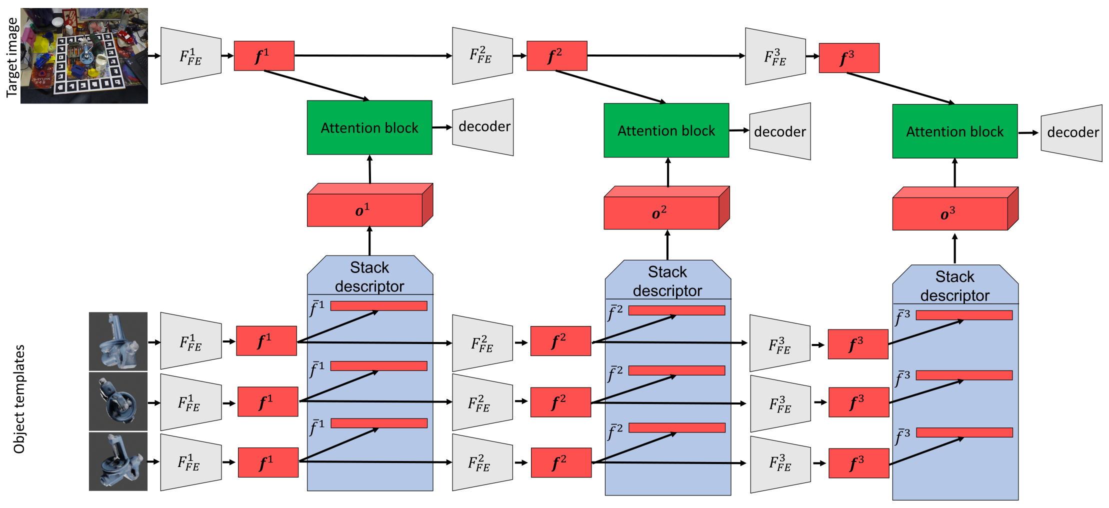


Figure 7.2: Encoder of the stage 1 network. The network takes an input image and an object model, represented by a sparse set of templates, and outputs a binary segmentation of the target image. The full detailed architecture is provided in the supplementary materials.

For a particular pixel (h, w), the correlation tensor value is defined as

$$\mathbf{c}^k_{h,w,x,y,z} = \text{corr}\left(\mathbf{f}^k_{h,w}, \mathbf{o}^k_{x,y,z}\right), \tag{7.1}$$

where corr denotes Pearson correlation. The correlation tensor is then flattened to a 3D tensor $\mathbf{c}^k \in \mathbb{R}^{H^k \times W^k \times \left(X^k Y^k Z^k\right)}$. This way, each pixel of the target image feature map gets the list of all correlations of its feature vector with all the feature vectors of the descriptor.

The flattened correlation tensor is used in two ways. First, pre-computed correlations are used directly as the input to the decoder, as in [14, 200, 201, 202]. For that, the tensor $\mathbf{c}^k$ is processed by a 1×1 convolutional layer to reduce the number of dimensions from $\left(X^k Y^k Z^k\right)$ to L^k. The correlations are also used to compute pixel-wise attention. Pixel-wise attention allows us to effectively incorporate the original image features into the feature tensor and use them for more precise segmentation.

Raw pixel-wise attention at the feature map level k is defined simply as a sum of all $\left(X^k Y^k Z^k\right)$ correlations for a given pixel as

$$A^k_{h,w} = \max\left\{0, \sum_{j=1}^{\left(X^k Y^k Z^k\right)} \mathbf{c}^k_{h,w,j}\right\}. \tag{7.2}$$

Since simple attention can be very noisy in the early layers compared to later layers, we propose to condition per-pixel attention of each particular level k on the attention from the last level k_l, which tend to be more precise but have low resolution, as

$$\hat{A}^k_{h,w} = A^k_{h,w} \bigtriangledown \left(A^{k_l}\right)_{h',w'} \tag{7.3}$$

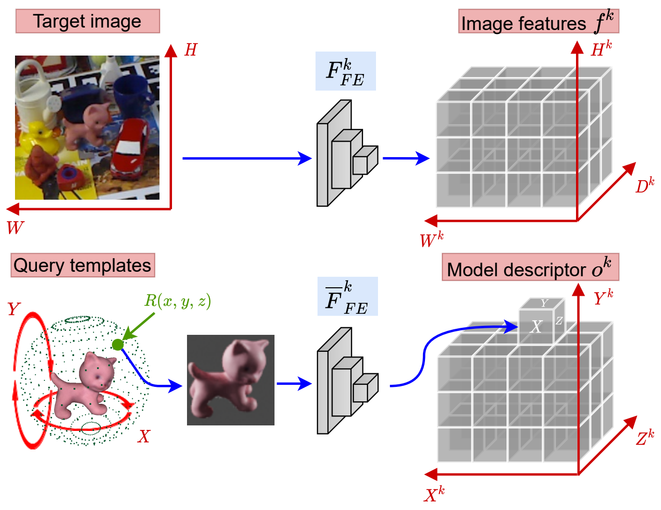


Figure 7.3: Feature computation for a target image and a 3D model. The top row illustrates how an input target RGB image is converted to a 3D feature tensor f^k. The bottom row demonstrates how object templates sampled along azimuth (axis X), elevation (Y) and in-plane rotation axis Y are transformed to a corresponding dense 4D model descriptor o^k.

$\bigtriangledown$ denotes a bilinear upsampling of the attention A^{k_l} to the size of A^k. The values are then filtered by zeroing out attention values below the average value, resulting in cleaner and more precise attention maps, as illustrated in the supplementary material:

$$\hat{A}^k_{h,w} = \begin{cases} \hat{A}^k_{h,w}, & \text{if } \hat{A}^k_{h,w} > \text{avg}_{h',w'} \hat{A}^k_{h',w'} \\ 0, & \text{otherwise} \end{cases} \tag{7.4}$$

A^{k_l} itself is thresholded but not conditioned on anything. All of the values are scaled to fall between 0 and 1. Image features are transformed using the attention maps as follows:

$$\hat{\mathbf{f}}^k_{h,w} = \hat{A}^k_{h,w} \cdot \mathbf{f}^k_{h,w} - (1 - \hat{A}^k_{h,w}) \cdot \mathbf{f}^k_{h,w}. \tag{7.5}$$

The attended features are then processed by a 1×1 convolutional layer to reduce dimensionality. Stacked $\hat{\mathbf{f}}^k$ and $\mathbf{c}^k$ are used jointly by the subsequent layers. Overall, the decoder resembles the UNet [61] approach of feature maps upsampling followed by convolutional layers until the initial image size is reached. The main distinction is that the network employs stacked $\hat{\mathbf{f}}^k$ and $\mathbf{c}^k$ at each level rather than skip connections. The network is trained to predict per-pixel probability that a pixel contains a visible part of the object. We used the Dice loss $\mathcal{L}_{Dice}$ [184] to handle imbalanced class data.

7.2.2 Template Matching

For the initial viewpoint estimation, we rely on template matching via deep manifold learning , which has been shown to scale well to a large number of objects [70, 112] and

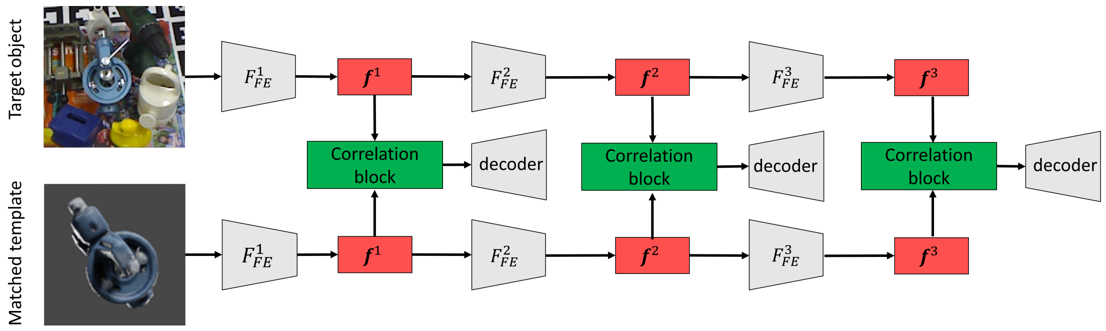


Figure 7.4: Encoder of the stage 3 network. The network takes an input image with the detected object and a matched template. Its output is a pixel-wise binary segmentation and dense 2D-2D correspondences. A detailed architecture is provided in the supplementary materials.

to generalize to new objects [82] not seen during training. We rely on the same feature extraction network F_{FE} but use only the features from the last layer. We also add one 1×1 convolutional layer to decrease the dimensions from $H \times W \times D$ to $H \times W \times D'$. Template features $\mathbf{t} \in \mathbb{R}^{H \times W \times D'}$ and image features $\mathbf{f} \in \mathbb{R}^{H \times W \times D'}$ are pre-computed from the foregrounds of the query templates and from the foreground of the detected object in the target image denoted with using the segmentation predicted in the previous step respectively. Analogously to the first stage, similarity of two patches is estimated by computing per-pixel correlations between $\mathbf{f}$ and $\mathbf{t}$ using

$$\operatorname{sim}(\mathbf{f}, \mathbf{t}) = \sum_{h,w} \operatorname{corr}(\mathbf{f}_{h,w}, \mathbf{t}_{h,w}) \,. \tag{7.6}$$

We train the network to increase similarity for patches which depict objects with very close rotations and at the same time penalize similarity for distant rotations. A modified triplet loss with dynamic margin is leveraged by optimizing

$$\mathcal{L}_{triplets} = \max\left\{0, 1 - \frac{\operatorname{sim}(\mathbf{f}^{anchor}, \mathbf{f}^{+})}{\operatorname{sim}(\mathbf{f}^{anchor}, \mathbf{f}^{-}) + m}\right\}, \tag{7.7}$$

where m is set to the angle between rotations of the object in the anchor and the puller patches. Using the terminology from [70], $\mathbf{f}^{anchor}$ is a descriptor of a randomly chosen object patch. $\mathbf{f}^{+}$ corresponds to a puller - a template in the pose very similar to the pose in the anchor, while $\mathbf{f}^{-}$ corresponds to the pusher with a dissimilar pose. A query template with the highest similarity to the detected object in the target image is chosen as a match at test time.

7.2.3 One-Shot Dense Correspondence Estimation

The goal of this stage is to establish 2D-3D correspondences between the image pixels and the object model. After the previous step, we have a patch with the detected object in unknown pose and a matched template in known pose. Establishing dense 2D-2D correspondences between the object patch and the template explicitly provides 2D-3D matches between the object pixels and the 3D object model. The correspondences can then be used to estimate the pose with PnP+RANSAC or Kabsch+RANSAC.

Similarly to the previous stages, the architecture of the 2D-2D matching follows the general idea of dense feature matching. Each pixel of the feature map $\mathbf{f}^k$, representing the input image patch of the detected object, is matched with all pixels of the template feature map $\mathbf{t}^k$ to form the correlation tensor $\mathbf{c}^k$. The network then predicts three values for each pixel: a binary foreground/background segmentation mask and a coordinate of the corresponding pixel on the template. Figure 7.4 depicts the architecture.

During training, a random object crop I_{obj} with its associated pose $T_{obj} \in SE(3)$ is sampled from a synthetic dataset. Then, a random template I_{tmp} is picked together with its pose $T_{tmp} \in SE(3)$, so that T_{obj} and T_{tmp} are relatively close. Availability of object poses allows us to compute per-pixel 2D-3D correspondence maps in both patches. Let us denote 2D-3D correspondences for the object rendered in the given pose as follows:

$$C : \mathcal{M} \times SE(3) \to [0, 1]^{W \times H \times 3} \tag{7.8}$$

Correspondences are computed as Normalized Object Coordinates (NOCS) [74]. Its inverse C^{-1} recomputes correspondences with respect to the unnormalized object coordinates, corresponding to the actual 3D object coordinates. It allows us to define a 2D correspondence pair distance in the model's 3D coordinate space:

$$d(p, p') = \left\| C^{-1} \left(I_{obj}\right)_p - C^{-1} \left(I_{tmp}\right)_{p'} \right\|_2 \tag{7.9}$$

where p and p' are pixel coordinates in the image and template patches respectively. Ground truth dense 2D-2D correspondences are established by matching pixel pairs corresponding to the closest points in the 3D coordinate system of the model. For a point $p \in \mathcal{I}_{obj}$ its corresponding template point is computed as $\operatorname*{argmin}_{p' \in \mathcal{I}_{tmp}} d\left(p, p'\right)$. We employ an outlier-aware rejection for 2D-2D correspondences with large 3D spatial discrepancy.

The segmentation loss is defined as a per-pixel Dice loss ($\mathcal{L}_{Dice}$). In addition, the network predicts a discrete 2D coordinate using a standard per-pixel cross-entropy classification loss denoted as $\mathcal{L}_{2D2D}$.

7.2.4 Pose Hypothesis Verification

We propose an optional step for generating and verifying pose hypotheses. Its goal is to reduce the imprecision caused by incorrect initial viewpoint estimation. Pose hypotheses are generated by independently estimating 2D-2D correspondences from each of the top N matched templates and estimating poses from them. We greedily remove matched templates that are too close to each other to reduce run times and ensure more diverse poses. In practice, we filtered matches using a 15-degree threshold and picked top 25 templates from them. If depth is available, hypotheses are ranked based on the quality

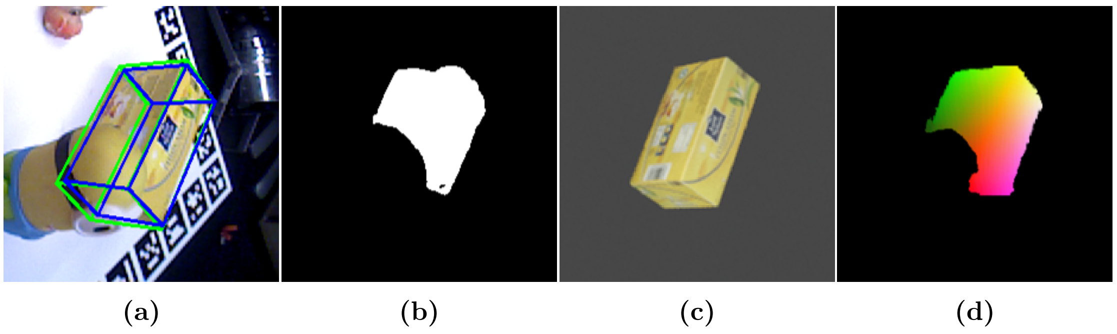


Figure 7.5: Qualitative evaluation of the proposed method on an object from the Homebrewed dataset [6]. a) an input image, cropped only for visualization purposes, with a comparison of the ground truth (green cuboid) and estimated (blue cuboid) poses; b) predicted one-shot segmentation; c) a matched template; and d) predicted correspondences as color-coded NOCS correspondences.

of fit of observed and rendered depth. In the RGB case, per-pixel correspondence error between the predicted correspondences and the rendered object is measured.

Pose hypothesis selection is done by rendering the object in the predicted pose and measuring a VSD-like pose consistency score [7]. If only RGB data is available, the score is defined over per-pixel correspondence error. If depth is available, per-pixel depth discrepancy is measured. For consistency, depth values and object sizes are expressed in millimeters. Let $\hat{S}$ be a predicted binary segmentation with $\hat{S}_p$ indicating whether a pixel p belongs to the object or not. Function $S : \mathcal{M} \times SE(3) \to [0,1]^{W\times H}$ renders a segmentation mask for a given object pose. Function $D : \mathcal{M} \times SE(3) \to \mathbb{R}^{W\times H}$ renders per-pixel depth for a given object pose, whereas $\hat{D}$ represents the observed depth map. $\mathbf{T} \in SE(3)$ is object pose.

Then, in case of RGB, pose consistency is defined as

$$\text{cons}(\tau) = \underset{p\in\hat{S}\cap S(\mathbf{T})}{\text{avg}} \begin{cases} 1, & \text{if } \left\| \hat{C}_p - C^{-1}\left(C\left(\mathbf{T}\right)_p\right)\right\|_2 < \tau \\ 0, & \text{otherwise} \end{cases} \tag{7.10}$$

In case of depth, pose consistency is defined as

$$\text{cons}(\tau) = \underset{p\in\hat{S}\cap S(\mathbf{T})}{\text{avg}} \begin{cases} 1, & \text{if } \left\| \hat{D}_p - D(\mathbf{T})_p\right\|_2 < \tau \\ 0, & \text{otherwise} \end{cases} \tag{7.11}$$

Consistency is averaged over thresholds τ from 1 to 5mm, and the pose hypothesis with the highest average consistency is selected as the final pose.

7.3 Experiments

We evaluate the proposed method on Linemod [5] (LM), Occlusion [8] (LMO), Homebrewed [6] (HBD), YCB-V [16] and TLESS [12] datasets. Each detector stage is trained

Table 7.1: 2D detection results in comparison to YOLO [13], trained on target objects, and one shot OS2D [14] detectors on the BOP split of the test data [7]. Results on the Homebrewed dataset [6] are reported on the publicly available validation split.

Dataset	Method	Precision	Recall	F1	Best Recall
LM	YOLO	0.99	0.97	0.98	0.99
	Ours	0.47	0.86	0.61	0.86
	OS2D	0.28	0.2	0.23	0.57
LMO	YOLO	0.69	0.67	0.68	0.85
	Ours	0.31	0.61	0.41	0.61
	OS2D	0.16	0.31	0.21	0.53
HBD	YOLO	0.76	0.74	0.75	0.91
	Ours	0.43	0.73	0.54	0.73
	OS2D	0.24	0.29	0.26	0.44
YCB	YOLO	0.72	0.84	0.78	0.98
	Ours	0.41	0.8	0.54	0.8
	OS2D	0.12	0.18	0.14	0.26

separately for each target dataset, so that train and test objects differ. For example, we use train the networks used for experiments on the Linemod dataset using all objects from the Homebrewed and YCB-V datasets. Linemod's and Homebrewed's common objects are skipped during training. We used the synthetic PBR images provided by the organizers of the BOP challenge [7] in all experiments. We denote methods, that used only PBR [7] synthetic images, with "PBR", methods with custom synthetic images as "synt" and methods, which used a mix of real and synthetic data, as "mix". The standard ADD score [15] with the 10% diameter threshold is reported for the Linemod dataset. The BOP Average Recall (AR) score [7] is reported for the other datasets.

7.3.1 2D Object Localization

We compare our approach's 2D detection capabilities to two baselines. First, we compare it to OS2D [14], a state of the art one-shot object detection method. For fairness, we ran OS2D with exactly the same templates as our method. Second, we compare to YOLOv3 [13], which was trained separately for each scene using the synthetic PBR image [7]. It establishes the upper bound for the object detection performance and demonstrates the recall achievable by a fully supervised 2D object detector trained on target objects. we cannot compute Mean Average Precision (mAP), because our localization network follows the semantic segmentation rather than the object detection paradigm. As an alternative, we chose a confidence threshold for each method on each dataset separately, maximizing the F1 score. We then report precision and recall that correspond to the optimal threshold as well as the highest possible recall achieved by

Table 7.2: Percentages of correctly estimated poses w.r.t. the ADD on the Linemod [5] dataset for methods trained on synthetic data. All methods apart from ours, PPF and PfS require prior training on RGB target objects.

Modality	Method	Refinement	ADD	Time (ms)
RGB	DPOD [100]	DL [100]	54.2	-
	Ours	Mult. Hyp.	43.6	1343
	DPOD [100]	-	40.5	36
	OURS	-	39.3	96
	SSD6D [104]	DL [144]	34.1	-
	AAE [115]	-	31.4	24
	PfS [140]	-	22.5	-
	SSD6D [104]	-	9.1	-
RGBD	SSD6D [104]	ICP	90.9	100
	Ours	Mult. Hyp. + ICP	81.9	749
	Ours	Mult. Hyp.	80.1	722
	PPF [15]	ICP	78.8	-
	Ours	ICP	76.8	68
	Ours	-	73.3	60
	AAE [115]	ICP	71.5	224

the detector. Table 7.1 summarizes the findings. As expected, YOLO performs better than the proposed method and OS2D, both in terms of precision and recall. Our method - not trained on the test objects - is outperformed by YOLO by 10%-20% in terms of best recall. At the same time, its recall is very close to the recall of YOLO with the confidence threshold corresponding to the best F1 score. Performance of OS2D is considerably worse, especially on more challenging datasets with more occlusion, e.g. LMO, HBD and YCB-V. Additionally, our method is ca. 800× faster with a runtime of only 25 milliseconds per object compared to 20 seconds per object for OS2D. This shows the advantages of the proposed method compared to state of the art in 2D one-shot detection.

7.3.2 6 DoF Pose Results

In this section, we assess the accuracy of poses estimated using our one-shot method. However, due to a lack of relevant work, it is not a straightforward task. Geometry-based deep learning methods [80, 81] are evaluated on a single dataset using an old pose metric that the new state of the art methods do not report. Although Multi-Path AAE [82] claims to be one-shot, it employs a 2D object detector trained on target objects. We, however, still compare to Multi-Path AAE, as it serves as a state of the art upper bound of what pose accuracy is achievable with deep learning methods capable of estimating the pose of novel objects not seen during training. when depth data is available, PPF results are reported, because it is another truly one-shot method that does not require

Table 7.3: Results on the Occlusion dataset [8] reported according to the Average Recall (AR) metric of the BOP challenge [7] on the BOP challenge subset of test images. All methods apart from ours and PPF [15] require prior training on target objects.

Method	**Train data**	**Refinement**	**AR**	**Time (s)**
CosyPose [10]	PBR	-	0.633	0.550
CDPN [73]		-	0.569	0.279
EPOS [119]		-	0.547	0.468
Pix2Pose [119]		-	0.363	1.310
Pix2Pose [119]		-	0.281	1.157
SSD6D [104]		-	0.139	-
EPOS [119]	synt	-	0.443	0.487
DPOD [100]		-	0.169	0.172
AAE [115]	mix	ICP	0.237	1.197
Multi-Path AAE [82]		-	0.217	0.200
AAE [115]		-	0.146	0.201
Drost, PPF [15]	-	ICP	0.527	15.947
Drost, PPF [15]		ICP, 3D edges	0.492	3.389
Ours + Kabsch		Mult. Hyp. + ICP	0.482	5.440
Ours + Kabsch		Mult. Hyp.	0.462	5.355
Ours + Kabsch		ICP	0.432	0.560
Ours + Kabsch		-	0.393	0.475
Ours + PnP		Mult. Hyp.	0.312	12.180
Ours + PnP		-	0.274	0.766

training on target objects. Other methods listed in the tables are explicitly trained on synthetic renderings of target objects, giving them an advantage over our method in terms of pose scores. Therefore, the results of our methods should not be directly compared to them; instead they should be used as a reference for what the standard 6 DoF pose estimation methods achieve. To summarize, we mainly compare our method to Multi-Path AAE on RGB images and to PPF on RGBD images.

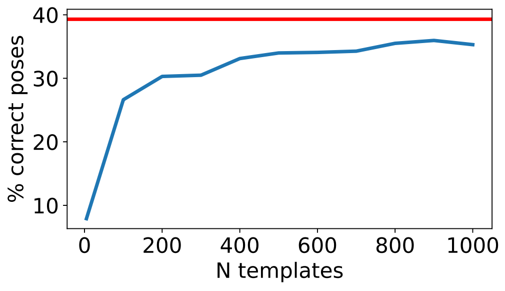
In Table 7.2, the quality of pose estimation is reported using the ADD score for the Linemod dataset. We compare our approach to other methods that use only synthetic training data because they represent what can be accomplished without access to the training data from the target domain. Our method predicts very good poses despite not having been trained on Linemod objects. It significantly outperforms SSD6D [104], and outperforms AAE [115] and SSD6D with deep learning-based refinement [144], while falling short of DPOD [100] by around 1%. Our method considerably outperform Pose from Shape [140] (PfS), which is another one-shot pose estimation method, even though PfS uses ground truth 2D detections and ground truth translations while estimating only rotation. Pose hypothesis verification raises the results by relative 10% from 39.3

Table 7.4: Results on the Homebrewed dataset [6] reported according to the Average Recall (AR) metric of the BOP challenge [7] on the BOP challenge subset of test images. All methods apart from ours and PPF [15] require prior training on target objects.

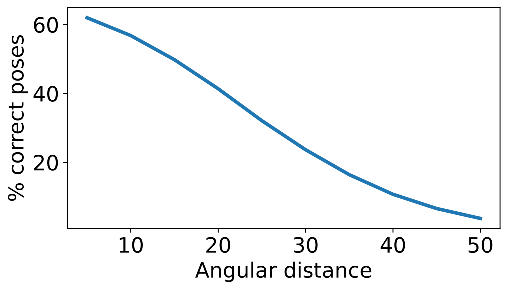
Method	Train data	Refinement	AR	Time (s)
CDPNv2 [73]	PBR	-	0.722	0.273
CosyPose [10]		ICP	0.712	5.326
CDPNv2 [73]		ICP	0.712	0.713
Pix2Pose [101]		ICP	0.695	3.248
CosyPose [10]		-	0.656	0.417
EPOS [119]		-	0.58	0.657
Pix2Pose [101]		-	0.446	0.982
AAE [115]	synt	ICP	0.506	1.352
CDPN [73]		-	0.47	0.311
AAE [115]		-	0.346	0.19
Multi-Path AAE [82]		-	0.293	0.191
DPOD [100]		-	0.286	0.18
Drost, PPF [15]	-	ICP	0.671	144.029
Ours + Kabsch		Mult. Hyp. + ICP	0.605	4.508
Drost, PPF [15]		ICP	0.603	1.659
Ours + Kabsch		ICP	0.581	0.438
Ours + Kabsch		Mult. Hyp.	0.579	4.384
Ours + Kabsch		-	0.56	0.314
Ours + PnP		Mult. Hyp.	0.492	8.183
Ours + PnP		-	0.464	0.503

to 43.6. If depth data is available, the pose can be estimated directly using 3D-3D correspondences and the Kabsch algorithm. This nearly doubles our method's ADD score, putting it above AAE with ICP refinement. Further pose hypothesis verification improves the results by around relative 10% putting it just above PPF. Segmentation of a single object takes 25 milliseconds, initial rotation approximation 10 milliseconds, 2D-2D matching 11 milliseconds. PnP takes 50 milliseconds on average, which makes the RGB-only pose estimation pipeline perform at approximately 10 FPS. If the Kabsch algorithm is used, the detector achieves 16 FPS. Additional ICP refinement on top of Kabsch slows down the detector to 14 FPS, which is still faster than other methods with ICP refinement. Pose hypothesis generation and verification in RGB takes around 1300 milliseconds and 550 milliseconds in depth images.

The results of our proposed method on the Occlusion (Table 7.3) and Homebrewed (Table 7.4) datasets show that it performs similarly to some of the methods that are trained on target objects with full supervision and trained separately for each object and scene. Even without refinement, our method outperforms Multi-Path AAE on both



(a)



(b)

Figure 7.6: Impact of the number of templates (a) and the angular distance between the ground truth and the matched template (b) on the final ADD score on Linemod dataset [5].

datasets. If depth data is available, our method falls short behind PPF only by a narrow margin while being an faster than the best performing PPF variant. On the YCB (Table 7.5) dataset, Multi-Path AAE outperforms the proposed method (Our+PnP), but it is important to note that methods trained on real or mixed data perform considerably better than methods trained solely on synthetic data on this dataset. On the other hand, our method outperforms PPF by a large margin. The results on all datasets clearly show the competitive quality of object detection and pose estimation in the proposed one-shot method and demonstrate that it generalizes well to objects, which were not seen during training. Moreover, its performance matches the performance of some of the previous state of the art methods even though they were trained on the target objects.

Table 7.6 shows evaluation of our method on the TLESS dataset [12]. We followed the Multi-Path AAE [82] evaluation pipeline and ran the 2nd and the 3rd stages of OSOP on detections from Multi-Path AAE. OSOP convincingly outperforms Multi-Path AAE and PPF on RGB and RGBD data respectively. This proves that the matching strategy does not suffer from object symmetries and heavy occlusions. We used the same networks as in the Linemod experiments.

7.3.3 Ablation Studies

We conducted three main ablation studies to determine what factors contribute to the performance of our pipeline. Table 7.7 examines the architecture choices for the localization network, Table 7.8 analyses the pose estimation, while Figure 7.6 demonstrates the robustness of the method to a smaller number of templates and to larger angular errors between the ground truth and the matched templates.

Table 7.7 shows that if we only use the feature correlation as in OS2D [14, 200], the network performs the worst achieving only 54% recall and 51% IoU. Only pixel-wise attention, on the other hand, yields comparable results. When both correlation and attention are used, network performance significantly increases which proves the effectiveness of the proposed architectural changes.

Table 7.5: **Results on the YCB-V dataset [16] reported according to the Average Recall (AR) metric of the BOP challenge [7] on the BOP challenge subset of test images.** All methods apart from ours and PPF [15] require prior training on target objects.

Method	Train data	Refinement	AR	Time (s)
CDPNv2 [73]	PBR	ICP	0.532	1.034
EPOS [119]		-	0.499	0.764
CDPNv2 [73]		-	0.39	0.448
CosyPose [10]		-	0.574	0.342
EPOS [119]	synt	-	0.696	0.572
CDPN [73]		-	0.422	0.295
DPOD [100]		-	0.222	0.341
CosyPose [10]	mix	ICP	0.861	2.736
CosyPose [10]		-	0.821	0.241
Pix2Pose [101]		ICP	0.78	2.59
CDPNv2 [73]		-	0.532	0.143
AAE [115]		ICP	0.505	1.581
AAE [115]		-	0.377	0.179
Multi-Path AAE [82]		-	0.289	0.181
Ours + Kabsch	-	Mult. Hyp. + ICP	0.572	2.606
Ours + Kabsch		ICP	0.565	0.302
Ours + Kabsch		Mult. Hyp.	0.542	2.571
Ours + Kabsch		-	0.529	0.267
Drost, PPF [15]		ICP, 3D edges	0.344	6.27
Ours + PnP		Mult. Hyp.	0.332	5.389
Drost, PPF [15]		ICP, 3D edges	0.33	1.282
Ours + PnP		-	0.296	0.41

Table 7.8 snows an analysis of the impact of various stages on pose estimation. We start by replacing the first two stages of the pipeline with ground truth and then gradually replace it with actual predictions form our networks. ADD10 score is computed w.r.t. the number of correctly detected objects unless specified otherwise in "Recall", which multiplies the ADD score by the recall. The first line sets the upper bound for the pose estimation performance by effectively replacing the first two stages with ground truth and only using predictions from the 2D-2D matching network and PnP. The second line introduces the template matching. The overall score decreases by only 2% indicating that the second stage network performs precise template matching given the GT segmentation masks and that the 2D-2D matching network is robust to larger angular differences between poses in the object patch and in the template. A large performance drop is observed in the third line, when GT segmentation is replaced with the predicted segmentation. This is further emphasized in the last row, where the ADD10 score is

Table 7.6: Results on the TLESS- dataset [12] reported according to the Average Recall (AR) metric of the BOP challenge [7] on the BOP challenge subset of test images. All methods apart from ours and PPF [15] require prior training on RGB renderings of target objects.

Method	**Train data**	**Refinement**	**AR**
CosyPose [10]		ICP	0.640
EPOS [119]		-	0.467
AAE [115]	synt	ICP	0.487
Multi-Path AAE [82]	synt	-	0.310
DPOD [100]	synt	-	0.081
Ours + Kabsch	-	-	0.532
Drost, PPF [15]	-	ICP	0.444
Ours + Kabsch	-	ICP	0.435
Drost, PPF [15]	-	ICP	0.404
Ours + PnP	-	-	0.403

corrected by detector's recall. These results indicate that the main error comes from erroneous initial viewpoint estimation. Furthermore, improving the predicted segmentation can further improve the overall performance of the one-shot pipeline by improving 2D recall and template matching. The table also shows that pose hypothesis verification helps filter out some of the erroneous poses and push the ADD score closer to the theoretical maximum of the method.

We followed the example of Multi-Path AAE and used 90K templates for all experiments. Figure 7.6a shows the ADD score on Linemod if 5 to 1K templates randomly sampled from the full set are used. The red line corresponds to the ADD score of 39.3 with all 90K templates. OSOP reaches ADD of 35.3 with only 1K templates and 38.6 with 5K templates. Figure 7.6b shows how the angular distance between the gt rotation and the matched template affects the ADD score. OSOP requires a smaller number of templates and can estimate poses even from more distant template matches because of dense correspondence estimation.

Ablation studies presented in Figures 7.7 and 7.8 illustrate how the ADD score changes depending on the minimal distance between templates in the set of pose hypotheses and

Table 7.7: Object localization and segmentation for various configurations of the proposed localization network on Linemod [5].

Configuration		**Precision**	**Recall**	**IoU**
Correlations	Attention			
✓		0.28	0.54	0.51
	✓	0.34	0.61	0.55
✓	✓	0.47	0.86	0.72

Table 7.8: ADD10 score on the Linemod [5] for different pipeline components.

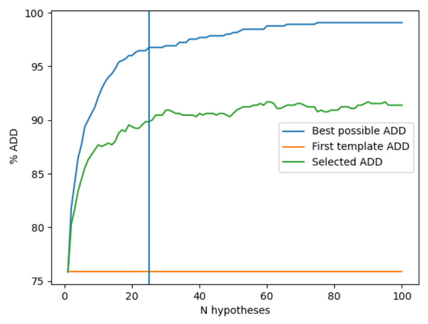
Configuration						ADD10
GT segm.	Pred. Segm.	Closest tmpl.	Matched tmpl.	Recall	Mult. Hyp.	
+		+				60.9
+			+			58.9
	+		+			45.7
	+		+	+		39.3
	+		+		+	51.0
	+		+	+	+	43.6

the number of hypotheses. The experiments were conducted on a small random subset of the data. "First template ADD" denotes the ADD reached on the subset of data using the standard OSOP pipeline without multiple hypothesis. "Best possible ADD" denotes the average ADD among all images in the subset, where for each image the best ADD among the predicted poses was picked. This sets the upper bound on what ADD the method can reach with the given number of pose hypothesis. "Selected ADD" denotes the ADD of the poses chosen with the proposed pose selection method. Experiments on Linemod and Occlusion demonstrate that the 15 degrees threshold on the distance between templates works the best, as it eliminates duplicate templates, that have a low angular distance from each other, and ensures more diverse hypothesis set. The plots also show that the ADD of chosen poses stops improving after approximately 25 pose hypotheses. Therefore, we used the threshold of 15 degrees and 25 templates in the experiments.

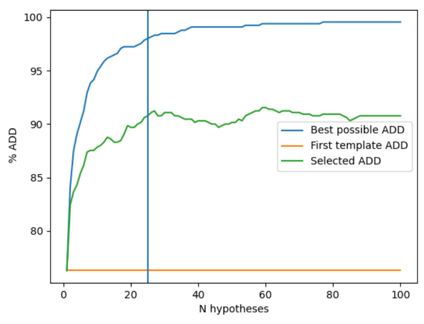
7.4 Implementation Details

The detector was implemented using Pytorch [203]. A pre-trained ResNet50 [51] served as feature extractor F_{FE} in all three stages. The feature extractor was unchanged for the segmentation and 2D-2D correspondence networks to overcome the domain gap problem. Only the last block of the ResNet was fine-tuned for the second stage in the proposed pipeline. We used the MAGSAC [164] implementation from OpenCV [186] and point-to-plane ICP from Open3D [187]. All our experiments were conducted on an Intel Core i9-9900K CPU 3.60GHz with NVIDIA Geforce RTX 2080 TI GPU. We trained the networks with the Adam optimizer [95]. The localization network and the 2D-2D matching network were trained for 50 epochs which took approximately one day on a single GPU. The second and third stage networks were trained for 10 epochs, which took approximately 2 hours. We render templates at 25 FPS with 128x128 resolution and models down-sampled to 5K faces. It takes around one hour to render 90K templates, 40 seconds to render 1K templates and 200 seconds to render 5K templates.

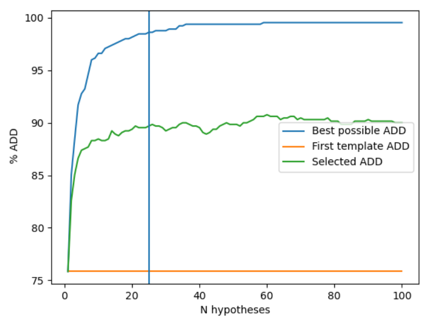
In all experiments, we used ResNet50 as the feature extractor. For the first and the third stages, we use feature maps after layers number 10, 22 and 40. The network



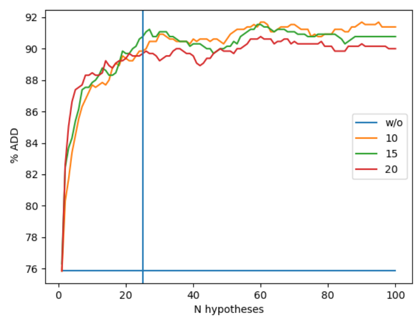
(a) 10 degrees template threshold.



(c) 15 degrees template threshold.



(b) 20 degrees template threshold.

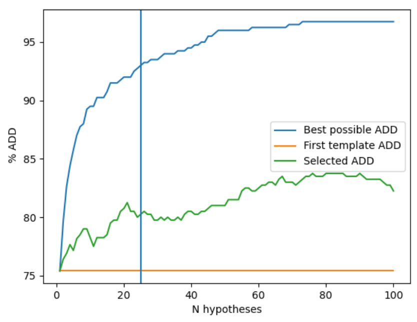


(d) Comparison of ADD with different template thresholds.

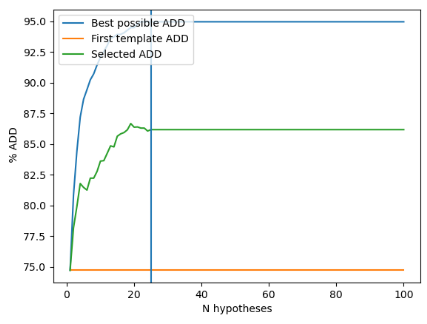
Figure 7.7: Ablation studies on pose hypothesis selection on Linemod dataset dataset [5].

is trained and tested on the full resolution images of size 480 $\times$ 640. For the image descriptor, this corresponds to $\mathbf{f}^k$ of sizes $120\times160\times256$, $60\times80\times512$ and $30\times40\times1024$. We used 2880 templates for the localization network, which corresponds to 576 camera locations with 5 in-plane rotations. The object descriptors $\mathbf{o}^k$ thus has the dimensions $16 \times 36 \times 5 \times 256$, $16 \times 36 \times 5 \times 512$ and $16 \times 36 \times 5 \times 1024$.

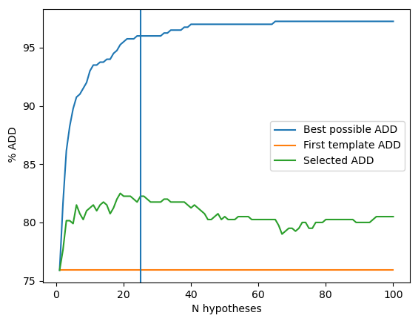
For the second stage, we used approximately 90K templates as suggested in AAE [115, 82], and map each of them to the latent space of size $8\times8\times256$. Even though the resulting descriptor is of higher dimension than in [70, 115], all descriptors for all templates still fit on a single GPU, which enables fast inference. During training, we convert the rotation matrix from the egocentric to allocentric coordinate system following [125]. This conversion ensures that the visual appearance of the object is dependent exclusively on the rotational component of the $SE(3)$ pose. The angle between rotations is computed as an arcos of quaternions representing them. Symmetric objects are ignored during training. The third stage operates on images of sizes 128 $\times$ 128.



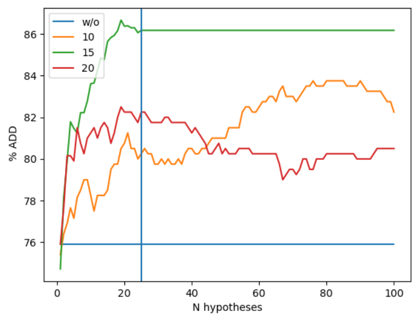
(a) 10 degrees template threshold.



(c) 15 degrees template threshold.



(b) 20 degrees template threshold.



(d) Comparison of ADD with different template thresholds.

Figure 7.8: Ablation studies on pose hypothesis selection on Occlusion dataset dataset [8]

7.5 Limitations

The method is predicated on the assumption that a pre-trained feature extractor computes distinctive features from the synthetic rendering of the object and the input image, and that the features have higher correlations for templates and images that represent the object in similar poses. This assumption is influenced by the domain gap between real and synthetic images. This problem could potentially be remedied by unsupervised domain adaptation techniques.

7.6 Conclusion

We proposed a novel object detection and 6 DoF pose estimation method that generalizes well to the objects unseen during training. To the best of our knowledge, it is the first one shot object detection and pose estimation method that does not impose any specific requirements for the objects. Our novel neural network architecture for one shot object localization performs significantly better and faster than an alternative 2D one shot detector. Our evaluation on Linemod, Occlusion, Homebrewed, YCB and TLESS datasets for pose estimation demonstrates the effectiveness of our method, which achieves similar results to methods trained on synthetic data. The proposed pipeline allows for a considerable reduction of time needed to prepare training data and to train the model, as these steps are not needed for new unseen objects. We believe it provides an interesting direction for further development of one shot 6 DoF pose detectors.

8 Conclusion & Outlook

In this final chapter, we summarize the presented methods, analyze their strengths and weaknesses, and suggest possible directions for future research.

8.1 Summary

In this thesis, we tackle the problem of 6 DoF object pose estimation. The main emphasis is devoted to development and analysis of pose estimation methods that rely on dense correspondences and to development of a method capable of detecting handling novel objects without prior training on them.

First, we introduced a deep-learning-based method for pose estimation with dense correspondences. The core idea is to establish per-pixel 2D-3D correspondences between the image and the object model. With the correspondences at hand, the final 6 DoF object pose is computed using the PnP and RANSAC methods. The method was evaluated on two challenging datasets (Linemod [5] and Occlusion [8]) and reached state-of-the-art results. An important feature of the method was the ability to be trained on either real or synthetic data. Additionally, a novel object pose refiner, that operates on monocular RGB images, was proposed. The refiner takes a tight crop around the detected object and a rendering of the object in its initial pose hypothesis. A convolutional neural network compares these two images to directly predict the pose offset. The refiner allowed us to boost the quality of pose predictions even further.

Second, we proposed a novel method for multi-view object pose refinement. The input to the method is a set of images with known relative camera transformations between them. We first estimate dense 2D-3D correspondences between the frames and the object model and estimate object pose hypothesis in each frame separately. The core principle of the refiner is to update the object pose in such a way, that the pose aligns the best with dense correspondences in all frames. The loss function is defined as a per-pixel difference of predicted 3D correspondences. The loss is implemented using a differentiable renderer, which allows for optimization with the standard gradient-based techniques. The proposed refiner achieves excellent pose accuracy on Linemod [5], Occlusion [8], Homebrewed [6] and YCB-V [16]. Additionally, we demonstrated that the proposed refiner can be used for automatic labeling of real images.

Then, we extended the ideas of dense correspondence and proposed a unified architecture capable of handling both RGB and depth input. This was used as a basis for an in-depth study of the influence of the type of input data (RGB versus Depth) and influence of the training data (real versus synthetic versus mixed) on the quality of estimated correspondences and on the quality of poses. The method was tested on four datasets

(Linemod [5], Occlusion [8], Homebrewed [6] and TLESS [12]) showing state-of-the-art results.

Finally, we tackled a challenging and relatively unexplored problem of object detection and 6 DoF pose estimation of novel object without prior training on them. Building on the ideas of one-shot object detection, we design a 4-stage pipeline to solve the task. In the first stage, a network takes an image and a descriptor of the object and outputs a binary segmentation mask. In the second stage, initial viewpoint estimation is performed using template matching. In the third stage, dense 2D-2D correspondences are established between the object in the unknown pose and the matched template in the known pose. In the final stage, the pose is computed using the standard PnP and RANSAC algorithms. The proposed method was evaluated on four different datasets (Linemod [5], Occlusion [8], Homebrewed [6], YCB-V [16] and TLESS [12]) and achieved good results on all of them in spite of not having been trained on the target objects. This method is arguably the first monocular RGB deep learning method to solve detection and pose estimation of novel objects without any constraints on the used objects, similarity between train and test objects or training parts of the pipeline on target objects.

8.2 Limitations and Future Work

In spite of the new advanced methods introduced in this thesis, the task of 6 DoF object pose estimation remains challenging. In general, the thesis focused on fully-supervised training of dense-correspondence-based models on either real or synthetic data and on generalization to novel objects. The thesis does not deal with the areas of weakly, unsupervised or few-shot learning, which have a tremendous potential for the pose estimation methods.

The dense correspondence method presented in Chapter 4 is based on semantic segmentation backbone model rather than on instance segmentation and, thus, struggles to deal with multiple object of the same type in the image. Detection via segmentation does not allow for setting a threshold on the detection confidence, as it is typically done in the object detection community. Also, one network is always trained on only one object to boost the performance of the method, which limits the scalability of the method. Runtime of the method heavily depends on the size of the object in the image, as larger objects will result in more correspondences.

The multi-view refiner method introduced in Chapter 5 offers a flexible and effective way to find globally-consistent object pose in multiples frames. One of its downsides is the assumption of known relative camera poses, which does not always hold true in practice. A joint object pose and camera pose optimization is a promising future direction, which will make the method more generic and usable in practice. Another disadvantage is slower runtimes of the method caused by differentiably rendering of the object in each refinement iteration.

The dense correspondence method presented in Chapter 6 solves some of the issues of the original method from Chapter 4 but is still not free of the disadvantages. While the first stage network, responsible for object detection, is trained to detect all objects in the scene, the correspondence estimation network is still trained separately for each object. Since each detection is processed separately, the runtime of the method scales linearly with the number of objects detected in the image.

The one-shot method introduced in Chapter 7, in spite of good results, is also not free of limitations. The semantic segmentation network of the first stage of the method performs the standard semantic segmentation rather than instance segmentation. This limits the applicability of the network if several objects of the same class are present in the image, as the network cannot distinguish between them. Then, the network depends on the templates rendered always in the same order, not allowing for usage of real labeled images or for varying the number of templates. More importantly, the overall performance of the method is based on the assumption that a backbone network pre-trained on Imagenet [173] will generalize to novel objects and will produce expressive and distinctive features. Solving the aforementioned issues might be a possible direction for future research. Another promising direction is mixing features computed from RGB and Depth for improved performance and robustness. Decreasing the number of stages of the method might results in faster runtimes.

Bibliography

[1] V. Lepetit, F. Moreno-Noguer, and P. Fua. Epnp: An accurate o (n) solution to the pnp problem. *International Journal of Computer Vision (IJCV)*, 81(2):155–166, 2009.

[2] S. University. Cs231n: Convolutional neural networks for visual recognition: `https://cs231n.github.io/neural-networks-1/`, 2022.

[3] Y. Wu and K. He. Group normalization. In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19, 2018.

[4] R. Hartley and A. Zisserman. *Multiple view geometry in computer vision.* Cambridge university press, 2003.

[5] S. Hinterstoisser, V. Lepetit, S. Ilic, S. Holzer, G. Bradski, K. Konolige, and N. Navab. Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes. In *Asian Conference on Computer Vision (ACCV)*, pages 548–562. Springer, 2012.

[6] R. Kaskman, S. Zakharov, I. Shugurov, and S. Ilic. Homebreweddb: Rgb-d dataset for 6d pose estimation of 3d objects. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, pages 0–0, 2019.

[7] T. Hodan and A. Melenovsky. Bop: Benchmark for 6d object pose estimation: `https://bop.felk.cvut.cz/home/`, 2019.

[8] E. Brachmann, A. Krull, F. Michel, S. Gumhold, J. Shotton, and C. Rother. Learning 6d object pose estimation using 3d object coordinates. In *European Conference on Computer Vision (ECCV)*, pages 536–551. Springer, 2014.

[9] Y. Li, G. Wang, X. Ji, Y. Xiang, and D. Fox. Deepim: Deep iterative matching for 6d pose estimation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 683–698, 2018.

[10] Y. Labbé, J. Carpentier, M. Aubry, and J. Sivic. Cosypose: Consistent multi-view multi-object 6d pose estimation. In *European Conference on Computer Vision (ECCV)*, pages 574–591. Springer, 2020.

[11] E. Brachmann, F. Michel, A. Krull, M. Y. Yang, S. Gumhold, et al. Uncertainty-driven 6d pose estimation of objects and scenes from a single rgb image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3364–3372, 2016.

[12] T. Hodan, P. Haluza, Š. Obdržálek, J. Matas, M. Lourakis, and X. Zabulis. T-less: An rgb-d dataset for 6d pose estimation of texture-less objects. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 880–888. IEEE, 2017.

[13] J. Redmon and A. Farhadi. Yolov3: An incremental improvement. *arXiv*, 2018.

[14] A. Osokin, D. Sumin, and V. Lomakin. Os2d: One-stage one-shot object detection by matching anchor features. In *European Conference on Computer Vision (ECCV)*, pages 635–652. Springer, 2020.

[15] B. Drost, M. Ulrich, N. Navab, and S. Ilic. Model globally, match locally: Efficient and robust 3d object recognition. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 998–1005. IEEE, 2010.

[16] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. 2018.

[17] A. P. Witkin. Scale-space filtering. In *Readings in Computer Vision*, pages 329–332. Elsevier, 1987.

[18] R. A. Haddad, A. N. Akansu, et al. A class of fast gaussian binomial filters for speech and image processing. *IEEE Transactions on Signal Processing*, 39(3):723–727, 1991.

[19] W. T. Freeman, E. H. Adelson, et al. The design and use of steerable filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 13(9):891–906, 1991.

[20] C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. In *Sixth International Conference on Computer Vision (ICCV)*, pages 839–846. IEEE, 1998.

[21] D. G. Lowe. Local feature view clustering for 3d object recognition. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages I–I. IEEE, 2001.

[22] H. Bay, T. Tuytelaars, and L. V. Gool. Surf: Speeded up robust features. In *European Conference on Computer Vision*, pages 404–417. Springer, 2006.

[23] M. Calonder, V. Lepetit, M. Ozuysal, T. Trzcinski, C. Strecha, and P. Fua. Brief: Computing a local binary descriptor very fast. *IEEE transactions on pattern analysis and machine intelligence*, 34(7):1281–1298, 2011.

[24] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. Orb: An efficient alternative to sift or surf. In *International Conference on Computer Vision (ICCV)*, pages 2564–2571. IEEE, 2011.

[25] J. Canny. A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*, (6):679–698, 1986.

[26] D. R. Martin, C. C. Fowlkes, and J. Malik. Learning to detect natural image boundaries using local brightness, color, and texture cues. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 26(5):530–549, 2004.

[27] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 33(5):898–916, 2010.

[28] J. Pont-Tuset, P. Arbelaez, J. T. Barron, F. Marques, and J. Malik. Multiscale combinatorial grouping for image segmentation and object proposal generation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 39(1):128–140, 2016.

[29] D. Marr and T. Poggio. Cooperative computation of stereo disparity: A cooperative algorithm is derived for extracting disparity information from stereo image pairs. *Science*, 194(4262):283–287, 1976.

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

[31] U. R. Dhond and J. K. Aggarwal. Structure from stereo-a review. *IEEE transactions on systems, man, and cybernetics*, 19(6):1489–1510, 1989.

[32] D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision (IJCV)*, 47(1):7–42, 2002.

[33] M. Z. Brown, D. Burschka, and G. D. Hager. Advances in computational stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 25(8):993–1008, 2003.

[34] S. M. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 519–528. IEEE, 2006.

[35] D. L. Milgram. Computer methods for creating photomosaics. *IEEE Transactions on Computers*, 100(11):1113–1119, 1975.

[36] S. Peleg. Elimination of seams from photomosaics. *Computer Graphics and Image Processing*, 16(1):90–94, 1981.

[37] S. Mann and R. W. Picard. Virtual bellows: Constructing high quality stills from video. In *Proceedings of 1st International Conference on Image Processing*, volume 1, pages 363–367. IEEE, 1994.

[38] R. Szeliski. Video mosaics for virtual environments. *IEEE computer Graphics and Applications*, 16(2):22–30, 1996.

[39] J. L. Schönberger and J.-M. Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[40] J. L. Schönberger, E. Zheng, M. Pollefeys, and J.-M. Frahm. Pixelwise view selection for unstructured multi-view stereo. In *European Conference on Computer Vision (ECCV)*, 2016.

[41] J. Engel, T. Schöps, and D. Cremers. Lsd-slam: Large-scale direct monocular slam. In *European Conference on Computer Vision (ECCV)*, pages 834–849. Springer, 2014.

[42] J. Engel, V. Koltun, and D. Cremers. Direct sparse odometry. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 40(3):611–625, 2017.

[43] C. Kerl, J. Sturm, and D. Cremers. Dense visual slam for rgb-d cameras. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2100–2106. IEEE, 2013.

[44] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *2011 10th IEEE International Symposium on Mixed and Augmented Reality*, pages 127–136. Ieee, 2011.

[45] A. Dai, M. Nießner, M. Zollhöfer, S. Izadi, and C. Theobalt. Bundlefusion: Real-time globally consistent 3d reconstruction using on-the-fly surface reintegration. *ACM Transactions on Graphics (ToG)*, 36(4):1, 2017.

[46] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[47] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[48] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9, 2015.

[49] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2818–2826, 2016.

[50] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Thirty-first AAAI conference on artificial intelligence*, 2017.

[51] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[52] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[53] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 580–587, 2014.

[54] R. Girshick. Fast r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 1440–1448, 2015.

[55] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems (NeurIPS)*, pages 91–99, 2015.

[56] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In *European Conference on Computer Vision (ECCV)*, pages 21–37. Springer, 2016.

[57] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[58] J. Redmon and A. Farhadi. Yolo9000: better, faster, stronger. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7263–7271, 2017.

[59] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3431–3440, 2015.

[60] V. Badrinarayanan, A. Kendall, and R. Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 39(12):2481–2495, 2017.

[61] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention (MICCAI)*, pages 234–241. Springer, 2015.

[62] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia. Pyramid scene parsing network. In *Proceedings of the IEEE Conference on Computer vision and Pattern Recognition*, pages 2881–2890, 2017.

[63] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(4):834–848, 2017.

[64] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 2961–2969, 2017.

[65] A. Kirillov, K. He, R. Girshick, C. Rother, and P. Dollár. Panoptic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9404–9413, 2019.

[66] 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 *International Conference on Computer Vision (ICCV)*, pages 858–865. IEEE, 2011.

[67] P. J. Besl and N. D. McKay. Method for registration of 3-d shapes. In *Sensor fusion IV: control paradigms and data structures*, volume 1611, pages 586–606. Spie, 1992.

[68] M. Rad and V. Lepetit. Bb8: A scalable, accurate, robust to partial occlusion method for predicting the 3d poses of challenging objects without using depth. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 3828–3836, 2017.

[69] B. Tekin, S. N. Sinha, and P. Fua. Real-time seamless single shot 6d object pose prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 292–301, 2018.

[70] S. Zakharov, W. Kehl, B. Planche, A. Hutter, and S. Ilic. 3d object instance recognition and pose estimation using triplet loss with dynamic margin. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 552–559. IEEE, 2017.

[71] P. Wohlhart and V. Lepetit. Learning descriptors for object recognition and 3d pose estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3109–3118, 2015.

[72] I. Shugurov, S. Zakharov, and S. Ilic. Dpodv2: Dense correspondence-based 6 dof pose estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2021.

[73] Z. Li, G. Wang, and X. Ji. Cdpn: Coordinates-based disentangled pose network for real-time rgb-based 6-dof object pose estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 7678–7687, 2019.

[74] H. Wang, S. Sridhar, J. Huang, J. Valentin, S. Song, and L. J. Guibas. Normalized object coordinate space for category-level 6d object pose and size estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2642–2651, 2019.

[75] J. Taylor, J. Shotton, T. Sharp, and A. Fitzgibbon. The vitruvian manifold: Inferring dense correspondences for one-shot human pose estimation. In *2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 103–110. IEEE, 2012.

[76] R. A. Güler, N. Neverova, and I. Kokkinos. Densepose: Dense human pose estimation in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7297–7306, 2018.

[77] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[78] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[79] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos. Orb-slam: a versatile and accurate monocular slam system. *IEEE transactions on robotics*, 31(5):1147–1163, 2015.

[80] G. Pitteri, A. Bugeau, S. Ilic, and V. Lepetit. 3d object detection and pose estimation of unseen objects in color images with local surface embeddings. In *Proceedings of the Asian Conference on Computer Vision (ACCV)*, 2020.

[81] G. Pitteri, A. Bugeau, S. Ilic, and V. Lepetit. 3d object detection and pose estimation of unseen objects in color images with local surface embeddings. In *ACCV*, 2020.

[82] M. Sundermeyer, M. Durner, E. Y. Puang, Z.-C. Marton, N. Vaskevicius, K. O. Arras, and R. Triebel. Multi-path learning for object pose estimation across domains. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13916–13925, 2020.

[83] W. S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[84] F. Rosenblatt. *The perceptron, a perceiving and recognizing automaton Project Para.* Cornell Aeronautical Laboratory, 1957.

[85] J. Heaton. Ian goodfellow, yoshua bengio, and aaron courville: Deep learning, 2018.

[86] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[87] S. Lyu and E. P. Simoncelli. Nonlinear image representation using divisive normalization. In *2008 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8. IEEE, 2008.

[88] K. Jarrett, K. Kavukcuoglu, M. Ranzato, and Y. LeCun. What is the best multi-stage architecture for object recognition? In *2009 IEEE 12th International Conference on Computer Vision (ICCV)*, pages 2146–2153. IEEE, 2009.

[89] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning (ICML)*, pages 448–456. PMLR, 2015.

[90] S. Santurkar, D. Tsipras, A. Ilyas, and A. Madry. How does batch normalization help optimization? *Advances in neural information processing systems (Neurips)*, 31, 2018.

[91] D. Ulyanov, A. Vedaldi, and V. Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016.

[92] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

[93] A. Lydia and S. Francis. Adagrad—an optimizer for stochastic gradient descent. *Int. J. Inf. Comput. Sci*, 6(5):566–568, 2019.

[94] M. D. Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

[95] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[96] Y. Ma, S. Soatto, J. Košecká, and S. Sastry. *An invitation to 3-d vision: from images to geometric models*, volume 26. Springer, 2004.

[97] R. Szeliski. *Computer vision: algorithms and applications.* Springer Science & Business Media, 2010.

[98] Y. Zhou, C. Barnes, J. Lu, J. Yang, and H. Li. On the continuity of rotation representations in neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5745–5753, 2019.

[99] L. N. Trefethen and D. Bau III. *Numerical linear algebra*, volume 50. Siam, 1997.

[100] S. Zakharov, I. Shugurov, and S. Ilic. Dpod: 6d pose object detector and refiner. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1941–1950, 2019.

[101] K. Park, T. Patten, and M. Vincze. Pix2pose: Pixel-wise coordinate regression of objects for 6d pose estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 7668–7677, 2019.

[102] C. Song, J. Song, and Q. Huang. Hybridpose: 6d object pose estimation under hybrid representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 431–440, 2020.

[103] S. Peng, Y. Liu, Q. Huang, X. Zhou, and H. Bao. Pvnet: Pixel-wise voting network for 6dof pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4561–4570, 2019.

[104] W. Kehl, F. Manhardt, F. Tombari, S. Ilic, and N. Navab. Ssd-6d: Making rgb-based 3d detection and 6d pose estimation great again. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 1521–1529, 2017.

[105] R. Kaskman, I. Shugurov, S. Zakharov, and S. Ilic. 6 dof pose estimation of textureless objects from multiple rgb frames. In *European Conference on Computer Vision (ECCV) Workshops*, pages 612–630. Springer, 2020.

[106] S. Hinterstoisser, C. Cagniart, S. Ilic, P. Sturm, N. Navab, P. Fua, and V. Lepetit. Gradient response maps for real-time detection of textureless objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 34(5), 2012.

[107] Y. Konishi, Y. Hanzawa, M. Kawade, and M. Hashimoto. Fast 6d pose estimation from a monocular image using hierarchical pose trees. In *European Conference on Computer Vision (ECCV)*. Springer, 2016.

[108] R. Rios-Cabrera and T. Tuytelaars. Discriminatively trained templates for 3d object detection: A real time scalable approach. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 2048–2055, 2013.

[109] W. Kehl, F. Tombari, N. Navab, S. Ilic, and V. Lepetit. Hashmod: A hashing method for scalable 3d object detection. In *BMVC*, volume 1, page 2, 2015.

[110] T. Hodaň, X. Zabulis, M. Lourakis, Š. Obdržálek, and J. Matas. Detection and fine 3d pose estimation of texture-less objects in rgb-d images. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4421–4428. IEEE, 2015.

[111] R. Hadsell, S. Chopra, and Y. LeCun. Dimensionality reduction by learning an invariant mapping. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2. IEEE, 2006.

[112] M. Bui, S. Zakharov, S. Albarqouni, S. Ilic, and N. Navab. When regression meets manifold learning for object recognition and pose estimation. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6140–6146. IEEE, 2018.

[113] S. Zakharov, B. Planche, Z. Wu, A. Hutter, H. Kosch, and S. Ilic. Keep it unreal: Bridging the realism gap for 2.5 d recognition with geometry priors only. In *2018 International Conference on 3D Vision (3DV)*, pages 1–11. IEEE, 2018.

[114] B. Planche, S. Zakharov, Z. Wu, A. Hutter, H. Kosch, and S. Ilic. Seeing beyond appearance-mapping real images into geometrical domains for unsupervised cad-based recognition. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2579–2586. IEEE, 2019.

[115] M. Sundermeyer, Z.-C. Marton, M. Durner, M. Brucker, and R. Triebel. Implicit 3d orientation learning for 6d object detection from rgb images. In *Proceedings of the european conference on computer vision (ECCV)*, pages 699–715, 2018.

[116] M. Sundermeyer, Z.-C. Marton, M. Durner, and R. Triebel. Augmented autoencoders: Implicit 3d orientation learning for 6d object detection. *International Journal of Computer Vision (IJCV)*, 128(3):714–729, 2020.

[117] Y. He, W. Sun, H. Huang, J. Liu, H. Fan, and J. Sun. Pvn3d: A deep point-wise 3d keypoints voting network for 6dof pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11632–11641, 2020.

[118] O. Hosseini Jafari, S. K. Mustikovela, K. Pertsch, E. Brachmann, and C. Rother. ipose: instance-aware 6d pose estimation of partly occluded objects. In *Asian Conference on Computer Vision (ACCV)*, pages 477–492. Springer, 2018.

[119] T. Hodan, D. Barath, and J. Matas. Epos: Estimating 6d pose of objects with symmetries. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11703–11712, 2020.

[120] A. Kendall, M. Grimes, and R. Cipolla. Posenet: A convolutional network for real-time 6-dof camera relocalization. In *IEEE International Conference on Computer Vision (ICCV)*, 2015.

[121] F. Walch, C. Hazirbas, L. Leal-Taixe, T. Sattler, S. Hilsenbeck, and D. Cremers. Image-based localization using lstms for structured feature correlation. In *IEEE International Conference on Computer Vision (ICCV)*, 2017.

[122] A. Kendall and R. Cipolla. Geometric loss functions for camera pose regression with deep learning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[123] C. Wang, D. Xu, Y. Zhu, R. Martín-Martín, C. Lu, L. Fei-Fei, and S. Savarese. Densefusion: 6d object pose estimation by iterative dense fusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3343–3352, 2019.

[124] S. Umeyama. Least-squares estimation of transformation parameters between two point patterns. *IEEE Transactions on Pattern Analysis & Machine Intelligence (TPAMI)*, (4):376–380, 1991.

[125] F. Manhardt, G. Wang, B. Busam, M. Nickel, S. Meier, L. Minciullo, X. Ji, and N. Navab. Cps++: Improving class-level 6d pose and shape estimationfrom monocular images with self-supervised learning. *arXiv preprint arXiv:2003.05848v3*, 2020.

[126] X. Chen, Z. Dong, J. Song, A. Geiger, and O. Hilliges. Category level object pose estimation via neural analysis-by-synthesis. In *European Conference on Computer Vision (ECCV)*, pages 139–156. Springer, 2020.

[127] F. Li, I. Shugurov, B. Busam, M. Li, S. Yang, and S. Ilic. Polarmesh: A star-convex 3d shape approximation for object pose estimation. *IEEE Robotics and Automation Letters (RA-L)*, 7(2):4416–4423, 2022.

[128] MVTec. Bop: Benchmark for 6d object pose estimation: `https://www.mvtec.com/products/halcon/`, 2019.

[129] B. Drost and S. Ilic. 3d object detection and localization using multimodal point pair features. In *2012 Second International Conference on 3D Imaging, Modeling, Processing, Visualization & Transmission*, pages 9–16. IEEE, 2012.

[130] T. Birdal and S. Ilic. Point pair features based object detection and pose estimation revisited. In *2015 International Conference on 3D Vision (3DV)*, pages 527–535. IEEE, 2015.

[131] S. Hinterstoisser, V. Lepetit, N. Rajkumar, and K. Konolige. Going further with point pair features. In *European Conference on Computer Vision (ECCV)*, pages 834–848. Springer, 2016.

[132] H. Deng, T. Birdal, and S. Ilic. Ppfnet: Global context aware local features for robust 3d point matching. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 195–205, 2018.

[133] H. Deng, T. Birdal, and S. Ilic. Ppf-foldnet: Unsupervised learning of rotation invariant 3d local descriptors. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 602–618, 2018.

[134] H. Deng, T. Birdal, and S. Ilic. 3d local features for direct pairwise registration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3244–3253, 2019.

[135] H. Yu, F. Li, M. Saleh, B. Busam, and S. Ilic. Cofinet: Reliable coarse-to-fine correspondences for robust pointcloud registration. *Advances in Neural Information Processing Systems (NeurIPS)*, 34:23872–23884, 2021.

[136] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in Neural Information Processing Systems (NeurIPS)*, 28, 2015.

[137] V. N. Nguyen, Y. Hu, Y. Xiao, M. Salzmann, and V. Lepetit. Templates for 3d object pose estimation revisited: Generalization to new objects and robustness to occlusions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6771–6780, 2022.

[138] K. Park, A. Mousavian, Y. Xiang, and D. Fox. Latentfusion: End-to-end differentiable reconstruction and rendering for unseen object pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10710–10719, 2020.

[139] Y. He, Y. Wang, H. Fan, J. Sun, and Q. Chen. Fs6d: Few-shot 6d pose estimation of novel objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6814–6824, 2022.

[140] Y. Xiao, X. Qiu, P.-A. Langlois, M. Aubry, and R. Marlet. Pose from shape: Deep pose estimation for arbitrary 3d objects. *arXiv preprint arXiv:1906.05105*, 2019.

[141] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems (NeurIPS)*, 30, 2017.

[142] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 945–953, 2015.

[143] C. Harris and C. Stennett. Rapid-a video rate object tracker. In *British Machine Vision Conference (BMVC)*, pages 1–6, 1990.

[144] F. Manhardt, W. Kehl, N. Navab, and F. Tombari. Deep model-based 6d pose refinement in rgb. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 800–815, 2018.

[145] S. Rusinkiewicz and M. Levoy. Efficient variants of the icp algorithm. In *Proceedings third international conference on 3-D digital imaging and modeling*, pages 145–152. IEEE, 2001.

[146] S. Rusinkiewicz. A symmetric objective function for icp. *ACM Transactions on Graphics (TOG)*, 38(4):1–7, 2019.

[147] F. Pomerleau, F. Colas, R. Siegwart, and S. Magnenat. Comparing icp variants on real-world data sets. *Autonomous Robots*, 34(3):133–148, 2013.

[148] J. Sock, S. Hamidreza Kasaei, L. Seabra Lopes, and T.-K. Kim. Multi-view 6d object pose estimation and camera motion planning using rgbd images. In *Proceedings*

of the IEEE International Conference on Computer Vision (ICCV) Workshops, pages 2228–2235, 2017.

[149] Ö. Erkent, D. Shukla, and J. Piater. Integration of probabilistic pose estimates from multiple views. In *European Conference on Computer Vision (ECCV)*, pages 154–170. Springer, 2016.

[150] C. Li, J. Bai, and G. D. Hager. A unified framework for multi-view multi-class object pose estimation. In *Proceedings of the european conference on computer vision (ECCV)*, pages 254–269, 2018.

[151] X.-S. Gao, X.-R. Hou, J. Tang, and H.-F. Cheng. Complete solution classification for the perspective-three-point problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(8):930–943, 2003.

[152] J. A. Hesch and S. I. Roumeliotis. A direct least-squares (dls) method for pnp. In *2011 International Conference on Computer Vision (ICCV)*, pages 383–390. IEEE, 2011.

[153] A. Penate-Sanchez, J. Andrade-Cetto, and F. Moreno-Noguer. Exhaustive linearization for robust camera pose and focal length estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 35(10):2387–2400, 2013.

[154] T. Collins and A. Bartoli. Infinitesimal plane-based pose estimation. *International Journal of Computer Vision (IJCV)*, 109(3):252–286, 2014.

[155] G. Terzakis and M. Lourakis. A consistently fast and globally optimal solution to the perspective-n-point problem. In *European Conference on Computer Vision (ECCV)*, pages 478–494. Springer, 2020.

[156] E. Brachmann and C. Rother. Learning less is more-6d camera localization via 3d surface regression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4654–4662, 2018.

[157] D. Campbell, L. Liu, and S. Gould. Solving the blind perspective-n-point problem end-to-end with robust differentiable geometric optimization. In *European Conference on Computer Vision (ECCV)*, pages 244–261. Springer, 2020.

[158] G. Wang, F. Manhardt, F. Tombari, and X. Ji. Gdr-net: Geometry-guided direct regression network for monocular 6d object pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16611–16621, 2021.

[159] B. Chen, A. Parra, J. Cao, N. Li, and T.-J. Chin. End-to-end learnable geometric vision by backpropagating pnp optimization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8100–8109, 2020.

[160] Y. Hu, P. Fua, W. Wang, and M. Salzmann. Single-stage 6d object pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2930–2939, 2020.

[161] O. Chum, J. Matas, and J. Kittler. Locally optimized ransac. In *Joint Pattern Recognition Symposium*, pages 236–243. Springer, 2003.

[162] O. Chum, T. Werner, and J. Matas. Two-view geometry estimation unaffected by a dominant plane. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 772–779. IEEE, 2005.

[163] D. Barath and J. Matas. Graph-cut ransac. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6733–6741, 2018.

[164] D. Barath, J. Matas, and J. Noskova. Magsac: marginalizing sample consensus. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10197–10205, 2019.

[165] D. Barath, J. Noskova, M. Ivashechkin, and J. Matas. Magsac++, a fast, reliable and accurate robust estimator. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1304–1312, 2020.

[166] R. Raguram, O. Chum, M. Pollefeys, J. Matas, and J.-M. Frahm. Usac: A universal framework for random sample consensus. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 35(8):2022–2038, 2012.

[167] E. Brachmann, A. Krull, S. Nowozin, J. Shotton, F. Michel, S. Gumhold, and C. Rother. Dsac-differentiable ransac for camera localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6684–6692, 2017.

[168] E. Brachmann and C. Rother. Neural-guided ransac: Learning where to sample model hypotheses. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4322–4331, 2019.

[169] E. Brachmann and C. Rother. Visual camera re-localization from rgb and rgb-d images using dsac. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2021.

[170] B. Drost, M. Ulrich, P. Bergmann, P. Hartinger, and C. Steger. Introducing mvtec itodd-a dataset for 3d object recognition in industry. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV) Workshops*, pages 2200–2208, 2017.

[171] C. Rennie, R. Shome, K. E. Bekris, and A. F. De Souza. A dataset for improved rgbd-based object detection and pose estimation for warehouse pick-and-place. *IEEE Robotics and Automation Letters (RA-L)*, 1(2):1179–1185, 2016.

[172] A. Tejani, D. Tang, R. Kouskouridas, and T.-K. Kim. Latent-class hough forests for 3d object detection and pose estimation. In *European Conference on Computer Vision (ECCV)*, pages 462–477. Springer, 2014.

[173] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 248–255. Ieee, 2009.

[174] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *European Conference on Computer Vision (ECCV)*, pages 740–755. Springer, 2014.

[175] J. L. Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.

[176] T. Hodan, J. Matas, and S. Obdrzalek. On evaluation of 6d object pose estimation. In *European Conference on Computer Vision (ECCV)*, pages 606–619. Springer, 2016.

[177] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 580–587, 2014.

[178] Z. Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 22(11):1330–1334, 2000.

[179] S. Hinterstoisser, V. Lepetit, P. Wohlhart, and K. Konolige. On pre-trained image features and synthetic images for deep learning. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, pages 0–0, 2018.

[180] M. Oberweger, M. Rad, and V. Lepetit. Making deep heatmaps robust to partial occlusions for 3d object pose estimation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 119–134, 2018.

[181] S. Zakharov, W. Kehl, A. Bhargava, and A. Gaidon. Autolabeling 3d objects with differentiable rendering of sdf shape priors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12224–12233, 2020.

[182] S. Liu, T. Li, W. Chen, and H. Li. Soft rasterizer: A differentiable renderer for image-based 3d reasoning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 7708–7717, 2019.

[183] J. T. Barron. A general and adaptive robust loss function. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4331–4339, 2019.

[184] F. Milletari, N. Navab, and S.-A. Ahmadi. V-net: Fully convolutional neural networks for volumetric medical image segmentation. In *Fourth International Conference on 3D Vision (3DV)*, pages 565–571. IEEE, 2016.

[185] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. 2017.

[186] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[187] Q.-Y. Zhou, J. Park, and V. Koltun. Open3D: A modern library for 3D data processing. *arXiv:1801.09847*, 2018.

[188] M. Denninger, M. Sundermeyer, D. Winkelbauer, D. Olefir, T. Hodan, Y. Zidan, M. Elbadrawy, M. Knauer, H. Katam, and A. Lodhi. Blenderproc: Reducing the reality gap with photorealistic rendering.

[189] K. Perlin. An image synthesizer. *ACM Siggraph Computer Graphics*, 1985.

[190] M. Bui, T. Birdal, H. Deng, S. Albarqouni, L. Guibas, S. Ilic, and N. Navab. 6d camera relocalization in ambiguous scenes via continuous multimodal inference. In *European Conference on Computer Vision (ECCV)*, pages 139–157. Springer, 2020.

[191] F. Manhardt, D. M. Arroyo, C. Rupprecht, B. Busam, T. Birdal, N. Navab, and F. Tombari. Explaining the ambiguity of object detection and 6d pose from visual data. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 6841–6850, 2019.

[192] A. Doumanoglou, R. Kouskouridas, S. Malassiotis, and T.-K. Kim. Recovering 6d object pose and predicting next-best-view in the crowd. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3583–3592, 2016.

[193] D. Xu, D. Anguelov, and A. Jain. Pointfusion: Deep sensor fusion for 3d bounding box estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 244–253, 2018.

[194] J. Vidal, C.-Y. Lin, X. Lladó, and R. Martí. A method for 6d pose estimation of free-form rigid objects using point pair features on range data. *Sensors*, 18(8):2678, 2018.

[195] I. Shugurov, I. Pavlov, S. Zakharov, and S. Ilic. Multi-view object pose refinement with differentiable renderer. *IEEE Robotics and Automation Letters (RA-L)*, 6(2):2579–2586, 2021.

[196] F. Li, I. Shugurov, B. Busam, S. Yang, and S. Ilic. Ws-ope: Weakly supervised 6-d object pose regression using relative multi-camera pose constraints. *IEEE Robotics and Automation Letters (RA-L)*, 7(2):3703–3710, 2022.

[197] F. Li, H. Yu, I. Shugurov, B. Busam, S. Yang, and S. Ilic. Nerf-pose: A first-reconstruct-then-regress approach for weakly-supervised 6d object pose estimation. *arXiv preprint arXiv:2203.04802*, 2022.

[198] B. Busam, H. J. Jung, and N. Navab. I like to move it: 6d pose estimation as an action decision process. *arXiv preprint arXiv:2009.12678*, 2020.

[199] P. J. Besl and N. D. McKay. Method for registration of 3-d shapes. In *Sensor fusion IV: control paradigms and data structures*, volume 1611, pages 586–606. Spie, 1992.

[200] I. Rocco, R. Arandjelovic, and J. Sivic. Convolutional neural network architecture for geometric matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6148–6157, 2017.

[201] I. Rocco, R. Arandjelović, and J. Sivic. Convolutional neural network architecture for geometric matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, pages 1–14, 2018.

[202] I. Rocco, R. Arandjelović, and J. Sivic. End-to-end weakly-supervised semantic alignment. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6917–6925, 2018.

[203] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems (NeurIPS)*, 32, 2019.

A Authored and Co-Authored Publications

Authored

1. S. Zakharov*, **I. Shugurov***, S. Ilic. *Dpod: 6d pose object detector and refiner.* IEEE International Conference on Computer Vision (ICCV), 2019 (* equal contribution)

2. **I. Shugurov***, I. Pavlov*, S. Zakharov, S. Ilic. *Multi-view object pose refinement with differentiable renderer.* IEEE Robotics and Automation Letters (RA-L), 2021 (* equal contribution)

3. **I. Shugurov**, S. Zakharov, S. Ilic. *Dpodv2: Dense correspondence-based 6 dof pose estimation.* IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), 2021

4. **I. Shugurov**, F. Li, B. Busam, S. Ilic. *OSOP: A Multi-Stage One Shot Object Pose Estimation Framework.* IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2022

Co-Authored

1. R. Kaskman, S. Zakharov, **I. Shugurov**, S. Ilic. *Homebreweddb: Rgb-d dataset for 6d pose estimation of 3d objects.* EEE/CVF International Conference on Computer Vision (ICCV) Workshops, 2019

2. R. Kaskman, **I. Shugurov**, S. Zakharov, S. Ilic. *6 dof pose estimation of textureless objects from multiple rgb frames.* European Conference on Computer Vision (ECCV) Workshops, 2020

3. F. Li, **I. Shugurov**, B. Busam, S. Yang, S. Ilic. *Ws-ope: Weakly supervised 6-d object pose regression using relative multi-camera pose constraints.* IEEE Robotics and Automation Letters (RA-L), 2022

4. F. Li, **I. Shugurov**, B. Busam, M. Li, S. Yang, S. Ilic. *Polarmesh: A star-convex 3d shape approximation for object pose estimation.* IEEE Robotics and Automation Letters (RA-L), 2022

5. F. Li, H. Yu, **I. Shugurov**, B. Busam, S. Yang, S. Ilic. *NeRF-Pose: First-Reconstruct-Then-Regress Approach for Weakly-supervised 6D Object Pose Estimation.* arXiv preprint, 2022

6. H. Yu, J. Hou, Z. Qin, M. Saleh, **I. Shugurov**, Kai Wang, B. Busam, S. Ilic. *CoFi-RIGA: Coarse-to-Fine Correspondences from Rotation-Invariant and Globally-Aware Descriptors for Point Cloud Registration.* Under Review, 2022
7. S. Reddy Vutukur, **I. Shugurov**, B. Busam, A. Hutter, S. Ilic. *WeLSA: Learning To Predict 6D Pose From Weakly Labeled Data Using Shape Alignment.* European Conference on Computer Vision (ECCV), 2022

Patents

1. **I. Shugurov**, A. Hutter, S. Zakharov, S. Ilic. *Dense 6-dof pose object detector.* U.S. Patent Application No. 17/427,231. 2022