UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Kelvin Batista da Cunha

Domain adaptation using randomized knowledge for monocular 6DoF pose estimation

Recife

2024

Kelvin Batista da Cunha

Domain adaptation using randomized knowledge for monocular 6DoF pose estimation

Work presented to the Graduate Program in Computer Science of the Centro de Informática of the Universidade Federal de Pernambuco, as a partial requirement for obtaining a Doctor's degree in Computer Science.

Concentration Area: Media and Interaction

Supervisor: Veronica Teichrieb

Co-supervisor: Francisco Paulo Magalhães Simões

Recife

2024

.Catalogação de Publicação na Fonte. UFPE - Biblioteca Central

Cunha, Kelvin Batista da.
Domain Augmentation Using Randomized Knowledge for 6dof Pose Estimation / Kelvin Batista da Cunha. - Recife, 2024.
91f.: il.

Tese (Doutorado) - Universidade Federal de Pernambuco, Centro de Informática, Pós-Graduação em Ciências da Computação, 2024.
Orientação: Veronica Teichrieb.

1. Estimação de pose; 2. Detecção de objetos; 3. Randomização de domínio. I. Teichrieb, Veronica. II. Título.

UFPE-Biblioteca Central CDD 004

Kelvin Batista da Cunha

“Domain adaptation using randomized knowledge for monocular 6DoF pose estimation”

Tese de Doutorado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do título de Doutor em Ciência da Computação. Área de Concentração: Mídia e Interação

Aprovada em: 06/06/2024.

RESUMO

A pose 6DoF (seis graus de liberdade) de objetos rígidos é fundamental para resolver várias tarefas na visão computacional, facilitando a interação entre elementos físicos e virtuais. Recentes avanços em visão computacional, particularmente através de aprendizado profundo (DL), aumentaram significativamente a precisão das técnicas de estimação de pose. Modelos de DL são capazes de extrair detalhes intrinsecos da cena, capacitando-os a discernir e se adaptar a diversos cenários com eficiência. Além disso, as metodologias de DL demonstram versatilidade excepcional, sendo capazes de assimilar vários tipos de entrada. Ainda, é notável a capacidade deestes métodos em extrair características de objetos exclusivamente a partir de dados RGB, ajustando modelos que apresentam desempenho em tempo real em uma variedade de dispositivos. Essa capacidade não só simplifica os requisitos computacionais, mas também amplia a aplicabilidade desses modelos em configurações do mundo real. No entanto, algoritmos de DL muitas vezes requerem grandes bases de dados, adaptadas a distribuições específicas. Adquirir, anotar e manter tais bases não é apenas caro e demorado, mas também suscetível a imprecisões, falhando em encapsular completamente o domínio de aplicação. Nossos estudos iniciais analisaram o impacto das mudanças de distribuição dos dados na estimativa de pose 6DoF, revelando a dependência dos modelos aos dados de treinamento e sua susceptibilidade aos desafios do mundo real (ou seja, generalização no conjunto de teste). Variações raramente encontradas durante o treinamento, como mudanças na aparência do objeto (por exemplo, tamanho, cor, geometria), condições do ambiente (por exemplo, iluminação, velocidade de movimento, oclusão) e hardware da câmera (ou seja, quando o modelo é treinado com uma câmera, mas testado com outra), podem afetar drasticamente a precisão do modelo. Para enfrentar esse desafio, propomos uma pipeline que gera uma variedade diversificada de sequências sintéticas usando modelos CAD de objetos. Ao randomizar elementos da cena em cada quadro, mesmo que as condições pareçam incoerentes ou surrealistas, podemos treinar modelos supervisionados usando dados simulados, reduzindo assim a dependência de dados reais rotulados e permitindo adaptação a transformações contínuas. Além disso, estendemos nossa pipeline introduzindo uma nova estratégia baseada em geração sintética randomizada foto-realista para mitigar variações de domínio na estimativa de pose monocular 6DoF, enquanto são preservadas características originais da cena para reduzir a lacuna entre o domínio real e simulado. Aproveitando uma combinação de técnicas de reconstrução NeRF (Neural Radiance Fields) e randomização de domínio, nossa abordagem demonstra a viabilidade de alcançar modelos precisos de estimativa de pose com menor dependência de dados reais. Finalmente, propomos uma pipeline de estimativa de pose 6DoF sem CAD usando imagens randomizadas para rastreamento de objetos. Como contribuição adicional, propomos o C3PO, uma base de dados cross-device organizada para diferentes dispositivos de acordo com diferentes desafios da estimativa de pose. O conjunto de dados inclui mais de 100000 imagens RGB completas com anotações de pose para três objetos impressos em 3D e três câmeras diferentes, abordando questões como oclusão, mudanças de iluminação, desfoque de movimento, variação de cor e variação de escala. Usando o C3PO, podemos avaliar o desempenho do método diante de diferentes desafios para analisar o impacto dos dados randomizados. Experimentos comparativos com o estado-da-arte em conjuntos de dados publicamente disponíveis, incluindo linemod, linemod-Occlusion, C3PO e HomebrewedDB, indicam a validade de nossa abordagem. Destacamos a importância de nossas contribuições enfatizando o impacto da randomização nos desafios associados a variações de domínio, como mudanças na iluminação, desfoque de movimento e oclusão de objetos.

Palavras-chaves: Estimação de pose. Detecção de objetos. Randomização de domínio.

ABSTRACT

The 6DoF (six-degrees-of-freedom) pose of rigid objects is pivotal in solving various tasks within computer vision, facilitating seamless interaction between physical and virtual elements. Recent advancements in vision-based pose estimation, particularly through deep learning (DL), have significantly enhanced accuracy. DL models are adept at extracting intricate scene details, empowering them to discern and adapt to diverse scenarios with efficiency. Still, DL methodologies demonstrate exceptional versatility, capable of assimilating various input types. Noteworthy is their ability to distill object features exclusively from RGB data, fitting models that exhibit real-time performance across a spectrum of devices. This capability not only streamlines computational requirements but also broadens the applicability of such models in real-world settings. However, DL often requires extensive datasets tailored to specific target distributions. Acquiring, annotating, and maintaining such datasets is not only costly and time-consuming but also susceptible to inaccuracies, failing to fully encapsulate the application domain. Our initial studies analyzed the impact of distribution shifts on 6DoF pose estimation, revealing models' reliance on training data and their susceptibility to real-world challenges (i.e., generalization on test set). Variations rarely encountered during training, such as changes in object appearance (e.g., size, color, geometry), environmental conditions (e.g., illumination, motion speed, occlusion), and camera hardware (i.e., when the model is trained with one camera but tested with a different one), can drastically affect model accuracy. To address this challenge, we propose a pipeline that generates a diverse array of synthetic sequences using CAD models of objects. By randomizing scene elements in each frame, even if conditions appear incoherent or surrealistic, we can train supervised models using simulated data, thereby reducing the dependency on labeled real data and enabling adaptation to continuous transformations in the target distribution. Furthermore, we extended our pipeline by introducing a novel strategy based on a photo-realistic randomized synthetic generation to mitigate target domain variations within monocular deep 6DoF pose estimation while preserving source features to reduce the domain gap. Leveraging a combination of NeRF (Neural Radiance Fields) reconstruction and domain randomization techniques, our approach demonstrates the feasibility of achieving accurate pose estimation models with reduced reliance on real data. Finally, we propose a CAD-free 6DoF pose estimation pipeline using randomized frames for object tracking, seamlessly integrating object detection and optical flow. As an additional contribution, we propose C3PO, a cross-device dataset organized for each device according to different challenges in pose estimation. The dataset includes more than 100000 full RGB images with pose annotations for three 3D printed objects and three different cameras, addressing issues such as occlusion, illumination changes, motion blur, color variation, and scale variation. Using C3PO, we can assess the method's performance in the face of different isolated challenges to analyze the impact of randomized data in each variation. Comprehensive experiments against state-of-the-art methods on publicly available datasets, including linemod, linemod-Occlusion, C3PO, and HomebrewedDB, indicate the validity of our approach. Emphasizing the impact of randomization in addressing challenges associated with domain variations, such as changes in environmental lighting, motion blur, and object occlusion, underscores the significance of our contributions.

Keywords: Pose estimation. Object detection. Domain randomization.

LIST OF FIGURES

LIST OF TABLES

CONTENTS

1 INTRODUCTION

Estimating the six degrees-of-freedom (6DoF) pose of rigid objects in RGB images has been a longstanding challenge in computer vision. This task has broad industrial applications, including mixed reality (TJADEN et al., 2018), robotics (TREMBLAY et al., 2018; LI et al., 2022), autonomous vehicles (TREMBLAY et al., 2018; CAO et al., 2024), human-computer interaction (WU et al., 2017), medical applications, smart cities, and others (GUAN et al., 2024). Recently, pose estimation has seen significant advancements in accuracy due to machine learning (ML) methods, which leverage data to infer relevant characteristics from the observed domain without relying solely on hand-crafted features (HOQUE et al., 2021; SUNDERMEYER et al., 2023).

In pose estimation, neural networks, particularly deep learning models, can learn and encode essential information about objects in a scene, achieving impressive performance by learning attributes directly from pixels provided by input images (TEKIN; SINHA; FUA, 2018; ZAKHAROV; SHUGUROV; ILIC, 2019; PENG et al., 2019; PARK; PATTEN; VINCZE, 2019; DI et al., 2021; SU et al., 2022; XIE; BHATNAGAR; PONS-MOLL, 2023). While these RGB-based models may not always surpass their RGB-D counterparts (KEHL et al., 2016; TAN; NAVAB; TOMBARI, 2017; TIAN et al., 2020; NGUYEN et al., 2024; HONG; HUNG; CHEN, 2024; PARK; KIM; SEO, 2024) (which utilize additional sensor data) in specific scenarios, they offer the advantage of more accessible training and suitability for deployment on simpler devices. However, a large dataset is necessary to compensate for additional data, such as depth maps.

Extracting knowledge from the data source allows the model to adapt to variations within the distribution. Consequently, sufficient real data is essential for optimizing these models, providing the necessary information to extract relevant features and map task characteristics. However, obtaining a high-quality pose estimation dataset suitable for training these methods is non-trivial. Changing the target domain alters the sample distribution, impacting the features learned from the initial source domain. Annotating a large set of images encompassing all possibilities of the target domain is cumbersome, costly, and time-consuming. For example, in pose estimation, we must capture data that includes variations in scene illumination, motion, sensor noise, and background or foreground information.

Additionally, it is crucial to account for how object characteristics can vary due to viewpoint, time of day, location, or interaction with other objects (HINTERSTOISSER et al., 2012; TEJANI et al., 2018; GARON; LAURENDEAU; LALONDE, 2018; CUNHA et al., 2020a). These factors can complicate the annotation process, as well as the setup of the capturing environment, calibration of cameras, and configuration of illumination and viewpoints, all of which need a standardized procedure. Moreover, annotations are not guaranteed to be free from inaccuracies, which can introduce errors and bias during model training (CUNHA et al., 2022; HODAN et al., 2018).

Furthermore, the availability of a CAD model introduces another challenge when applying the model to new domains. Many algorithms rely on 3D CAD data to establish correspondences between 3D object data and visible pixels in the camera's image. Obtaining precise 3D data, especially for large and complex objects, or acquiring the hardware necessary to capture such data can be difficult. In real-world applications, slight variations in 3D information between different objects can lead to inaccuracies in pose predictions (CUNHA et al., 2022).

Domain adaptation emerges as an effective strategy to mitigate domain variability. Domain adaptation methods aim to improve task execution on an unlabeled domain by exploiting relevant information from a labeled source domain (PATEL et al., 2015; CSURKA, 2017; ZAKHAROV et al., 2022). Techniques such as rendering photorealistic images (DENNINGER et al., 2020; ZAKHAROV et al., 2022) or randomizing scene properties (CUNHA et al., 2022; HINTERSTOISSER et al., 2019) inject additional knowledge into training, thereby reducing reliance on real labeled images. Within the scope of domain adaptation, domain randomization (DR) is a category of algorithms designed to extend knowledge from a source domain by creating random simulated data with few or no labeled samples. The objective of DR is to generate data with the highest possible variation by altering the scene setup, including changes in objects' positions, materials, textures, illumination, and background. This approach ensures that during model deployment, new real samples are perceived as just another variation (TOBIN et al., 2017; TREMBLAY et al., 2018). Consequently, the ML model can handle real-world challenges not explicitly labeled or mapped in the training data.

While synthetic datasets make creating diverse scenes and conditions easier, a gap remains between simulated and real samples, potentially leading to variations in expected results. Randomization approaches may also introduce noise, hindering the model's ability to recognize objects in real-world scenarios (CUNHA et al., 2020b). Additionally, many methods that rely on 3D CAD information for rendering new images face challenges when CAD data is unavailable, a disparity known as the reality gap.

In a preliminary study, we analyzed the impact of data variability on 6DoF pose estimation. Specifically, we investigated how changes in camera sensors and environments affect the model's predictions. Based on our findings, we propose different training strategies that combine real and randomized synthetic images to reduce the number of real images required during training (CUNHA et al., 2020b; CUNHA et al., 2022). This approach has also enhanced the model's performance when encountering previously unseen challenges. However, it fails to map certain aspects in variations that cannot be easily mapped in the synthetic domain (e.g., the interaction of lighting and the object).

Recently, Neural Radiance Fields (NeRFs) (MILDENHALL et al., 2021; VERBIN et al., 2022) have offered a photorealistic solution for novel view rendering. These models use a few real images to encode properties in a neural network, reducing the impact of the domain gap caused by inaccurate rendering parameters. By implicitly recovering 3D scene information from a few real images with known poses, NeRFs eliminate the need for additional 3D data, such as CAD models. However, their inability to generalize to new scene properties, common in 6DoF pose estimation tasks, highlights the need to combine the generalization capacity of domain randomization strategies with NeRF rendering. This hybrid approach aims to bridge the domain gap and enhance model performance by addressing CAD-free randomization. Thus, we incorporate domain randomization into NeRF image generation and use this data to train a pose estimation model. The combination of NeRF and DR enables the generation of realistic frames and the training of models in various scenarios without relying on CAD models.

In this context, the main objective of this research is to develop a domain randomization approach enhanced by algorithms capable of learning real-world characteristics. This method aims to simplify the distribution available for training, transforming it into a more generalizable scenario and targeting coherent guided randomization. Consequently, we intend to train basic models using a randomized training strategy in a self-supervised manner, enabling continuous identification, creation, and adaptation of patterns from domain variations. Additionally, the proposed method addresses experiments in cross-device scenarios, evaluating data variability from different devices. To further reduce the domain gap caused by rendered synthetic frames, we combine our randomization approach with NeRFs. This allows us to leverage the features learned by NeRF, enhancing model representation through the randomization approach. This hybrid approach aims to bridge the domain gap and improve model performance based on CAD-free randomization. By incorporating randomization into NeRF image generation, we can train a pose estimation model, enabling the generation of realistic frames and training models in various scenarios.

1.1 OBJECTIVES

This research investigates the challenges posed by domain variation in the context of 6DoF pose estimation, which affect the effective use of RGB monocular deep learning-based models in real-world scenarios. The specific objectives of this research are outlined below.

- Investigate challenges in optimizing models for different distributions in the context of 6DoF object pose estimation, including addressing variability in the target's appearance, object occlusion, changes in environmental illumination, and motion blur.
- Explore the impact of camera variations on pose estimation models' predictions and evaluate how to train these models to handle such variations.
- Propose new learning strategies to develop solutions that can train models without the target object or with minimal data, enabling adaptation to domain variations.
- Evaluate the proposed solutions through qualitative and quantitative experiments, using appropriate metrics to validate the results in the specified scenarios.

As a contribution to this work, the following points can be highlighted:

- We introduce C3PO, a cross-device 3D printed object dataset for 6DoF pose estimation. C3PO includes images captured from various devices and is structured around multiple challenges that account for environmental settings and object appearance variations.
- A domain randomization approach adapted to the context of 6DoF pose estimation, considering the unique features of cameras and objects that define scene variations.
- A NeRF-based domain randomization (Rand-NeRF) pipeline for synthesizing images in 6DoF object pose estimation, eliminating the need for CAD models. To the best of our knowledge, this is the first method that integrates domain randomization and neural representation without CAD models, bridging the domain gap and enhancing model performance through CAD-free randomization.
- A Comprehensive evaluation of our approaches using publicly available datasets to assess model robustness and identify primary limitations. Our results demonstrate comparability to state-of-the-art benchmarks in simple scenarios and surpass performance in challenging scenarios involving changes in environmental light, motion blur, and occlusion.

1.2 WORK STRUCTURE

Chapter 2 discusses related works. Chapter 3 details the process of acquiring, processing, annotating, and validating the dataset created for evaluating our use case. In Chapter 4, a thorough analysis of the dataset distribution highlights the differences among each sequence created. Chapter 5 outlines the baseline method for 6DoF pose estimation, the training configuration, and the metrics used to evaluate pose accuracy. Additionally, preliminary experiments demonstrate the technical performance of the generated dataset. Chapter 6 and Chapter 7 describe the randomized data generator and the training strategies applied to integrate rendered randomized frames with real frames. These chapters assess how training with different data distributions impacts model accuracy. In Chapter 8, we present evidence of the impact of each variation by replicating the preliminary experiments from Chapter 5. We also discuss each challenge and strategy, describing which combination of data works best for each case evaluated. Chapter 9 outlines our improvements on the proposed method by incorporating NeRFs into the randomization pipeline. Chapter 10 demonstrates the method's performance over several experiments compared to the state-of-the-art. Finally, Chapter 11 concludes our work. A general overview of the proposed approach can be seen in Figure 1.2

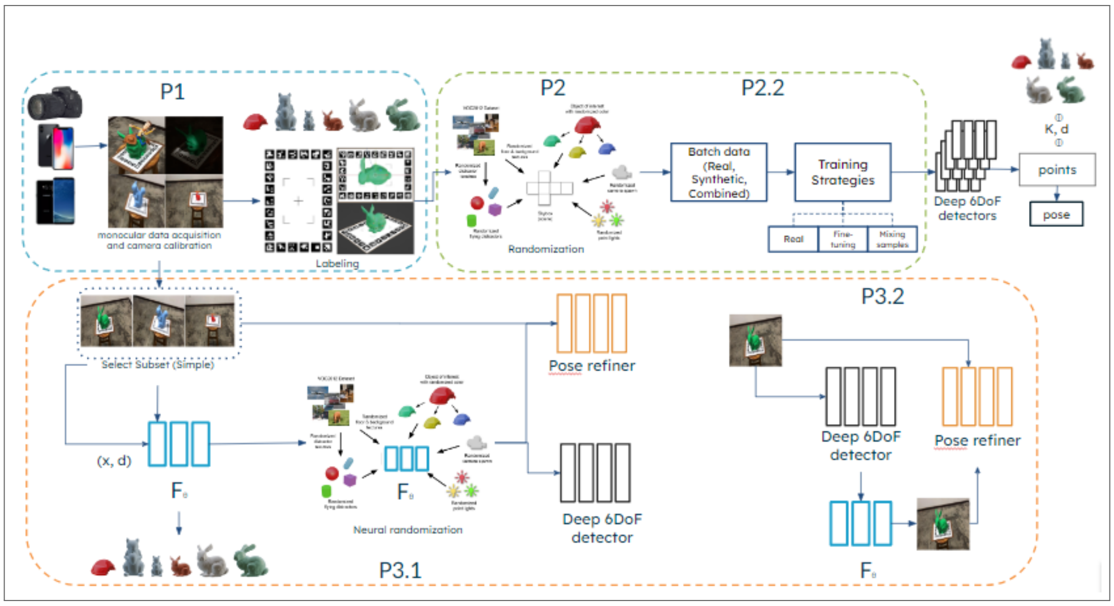

Figure 1 – Overview of the approach proposed in this paper. P1 refers to data acquisition and processing (Chapter 3 and Chapter 4), P2 and P2.2 refers to domain randomization generation and the detector training using real or synthetic data (Chapters 5, 6, and 8) and P3.3 refers to the pipeline extension using NeRF to represent the object's geometry (Chapters 9 and 10).

Source: CUNHA et al. (2024)

2 RELATED WORKS

2.1 6DOF POSE ESTIMATION

2.1.1 Classical pose estimation

Recently, many robust markerless 6DoF pose estimation approaches have been proposed. Out of these, RGB-D techniques have received special attention (TAN; NAVAB; TOMBARI, 2017; TIAN et al., 2020; NGUYEN et al., 2024; HONG; HUNG; CHEN, 2024; PARK; KIM; SEO, 2024). These works obtain a high accuracy due to the RGB input frame and the depth maps. Nevertheless, those depth maps suffer from limited view distance and struggle with illumination such as sunlight. RGB-D techniques are prone to fail in those cases, as they are left with just the RGB data. Considering how much less common and more expensive RGB-D sensors (compared to standard RGB), multiple works have also attempted to solve this issue using purely RGB (TJADEN et al., 2018; TEKIN; SINHA; FUA, 2018; DI et al., 2021; SU et al., 2022; XIE; BHATNAGAR; PONS-MOLL, 2023).

Several approaches have sought to develop methods using only RGB information (SONG; SONG; HUANG, 2020; VALENÇA et al., 2021; LIU et al., 2022; HAI et al., 2023). Most of these models are more complex and must deal with less information in learning. The lack of depth information from the sensors used in RGB techniques increases the need for intermediate steps to estimate the depth values for each pixel, sometimes deeper models are used to increase the level of representation, and these models are less computationally efficient.

Traditionally, techniques have used image processing and geometric concepts to tackle this problem. Region-based works (WANG et al., 2019; TJADEN et al., 2018) evaluate the color distribution in image regions. Edge-based works attempt to match the CAD model's edges to the RGB input's gradients (TRINH et al., 2018). Feature-based use keypoints and descriptors (HU et al., 2019; HE et al., 2020). Other techniques attempt to use traditional machine learning algorithms to interpret extracted features and optimize a hypothesis (SAHIN; KIM, 2018; DOUMANOGLOU et al., 2016). Finally, there are hybrid approaches that merge the aforementioned geometric concepts with machine learning (TAN; NAVAB; TOMBARI, 2017; TEJANI et al., 2018; TIAN et al., 2020).

Based on the main characteristics of geometric and machine learning methods, Tan et al. propose a hybrid approach (TAN; NAVAB; TOMBARI, 2017), which combines the main features of the optimization and learning processes to perform tracking in 6DOF. The model performs temporal tracking using a Random Forest with manually defined characteristics and optimization. The learning algorithm is capable of dealing with different scenarios and challenges due to its nature to generalize information from the training data, while the optimization process is less sensitive to cases of local minima, converging the model for better accuracy and causing noises like jitter. Using RGBD information, the algorithm matches information extracted from the object's contours (RGB) and the object's internal region (depth). The extracted information is passed to the Random Forest algorithm, which consists of a set of simple classifications that, when grouped, investigate the transformation of the vectors that update the object's pose. Optimization is used to minimize the energy function that records the correspondence between the projections of the points obtained for the object model and the scene, avoiding prediction errors that can occur in transformation queries. The model is capable of handling occlusion issues, polluted backgrounds, and low lighting conditions. However, the method is sensitive to sunlight, fast movement, and depends on the distance from the camera to the tracked object. Another work that is based on previously defined characteristics and combines optimization and learning methods is proposed in (TEJANI et al., 2018). The characteristics used are obtained through the adaptation of methods for pattern matching. The generated descriptors are integrated into the Random Forest algorithm, which is trained on the RGBD data distribution. Training can be performed with synthetic images of 3D models of objects. During pose inference, the process is iteratively refined, being designed to deal with high occlusion and polluted background scenarios, generating detections for multiple objects simultaneously.

Kehl et al. propose the combination of deep learning and voting to perform detection in 6DOF from RGBD images (KEHL et al., 2016). The work follows the voting paradigm based on local features, as input images are divided into patches containing color and depth information. The architecture is trained to receive patches and return the same information as output. The objective is to generate a function that produces features extracted from the intermediate layers of the architecture that are dimensionally reduced and stored in a feature dictionary. The set of stored information is associated with each known pose annotation for the training data.Inference is performed by using K-Nearest Neighbor (KNN) to each region of the image, selecting a pose with the highest number of votes associated with the characteristics found in the input image. Training this method avoids the use of data augmentation and can be carried out using synthetic data. However, detection failures may occur when used for real data due to the risk of overfitting the model to synthetic data.

2.1.2 Deep 6DoF pose estimation

In recent years, deep learning has become popular in pose estimation. Those can learn about the object of interest by being trained with images without needing pre-defined features or clues like traditional approaches do. This characteristic makes deep learning especially suitable for performing pose estimation in scenarios with little available information, such as plain RGB. Though it has been shown to work for real-time object tracking (with temporal consistency) (GARON; LALONDE, 2017; LI et al., 2018; WEN et al., 2023), most DL works attempt to perform object detection on a frame by frame basis (KEHL et al., 2017; TEKIN; SINHA; FUA, 2018; PARK; PATTEN; VINCZE, 2019; SONG; SONG; HUANG, 2020; DI et al., 2021; SU et al., 2022; LI et al., 2023).

Garon et al., in their work (Deep 6-DoF) (GARON; LALONDE, 2017) use a CNN to perform pose estimation, using multiples inputs from the RGB sensor. The algorithm can track objects using two consecutive frames from the target video to estimate the relative transformation, training the network with synthetic images. The Deep 6-DoF uses a simple architecture, so it can be very efficient, allowing its usage for real-time tracking. However, the model is sensitive to tracking loss, failing to recover when the detected object is lost, propagating the estimation error throughout the rest of the execution. The model is trained to handle cases of severe occlusion and is the first CNN-based model using the temporal information of consecutive frames. However, it does not get better accuracy scores and can only track one object per model.

Newer deep learning approaches have been using more complex, deeper pipelines to perform both pose prediction and refinement directly (LI et al., 2018; ZAKHAROV; SHUGUROV; ILIC, 2019; DI et al., 2021). As an example, the DeepIM proposed by Li et al. (LI et al., 2018) uses the result of PoseCNN (XIANG et al., 2017) for the development of a new approach to 6-DoF pose refinement. Based on the PoseCNN estimation, DeepIM performs an iterative process to refine the pose using the FlowNet architecture (DOSOVITSKIY et al., 2015). The technique iteratively adjusts the pose building a SE(3) representation to predict the relative transformation between the object observed in the image and the object rendered with the estimated pose. Thus, it can render a new image by applying the transformation computed. The process ends when the pose on the input image matches the pose estimated on the rendered image. This process makes the model accurate, leaving a very low execution time and unsuitable for real-time applications.

On the other hand, some approaches simplify the problem by attempting to detect image keypoints (HE et al., 2020) or bounding box corners (TEKIN; SINHA; FUA, 2018; SONG; SONG; HUANG, 2020) and only then perform minimization calculations in a more traditional way, such as by solving least-squares problems or similar algorithms. The simplified architectures of such approaches make training the model much easier while achieving high accuracy values. However, they still suffer from a major drawback: the need for large amounts of annotated data. Deep learning 6DoF works can detect synthetic objects when trained with synthetic data, but most still require real data to detect real objects.

2.2 6DOF DATASETS AND BENCHMARKS

Over the years, several datasets have been proposed to assist in the investigation and evaluation of methods for pose estimation (HODAN et al., 2018; YUAN; HOOGENKAMP; VELTKAMP, 2021). Most of them provide color (RGB) and depth information, captured by RGB-D cameras such as the Asus Xtion Pro Live, Primensense Carmine, and Microsoft Kinect. Also, some datasets already provide a synthetic set to train deep learning and model-based approaches, rendering images through different views of the 3D object on a black background. The synthetic data are relevant to evaluating strategies that can be optimized by the geometric information in the synthetic data, preserving the real samples to be used only in the test cases.

Each dataset evaluates a specific problem, focusing on a particular set of challenges for pose estimation. Likewise, the organization of strategies for labeling, validating information, and structuring the samples are distinct for each dataset. Most datasets can assess occlusion, illumination changes, cluttered environment, and motion blur challenges. However, no dataset provides structured data for each challenge isolated on the same problem.

LINEMOD (HINTERSTOISSER et al., 2012)[1] is a very popular dataset in the pose estimation domain. The dataset is composed of 11 non-textured objects with different geometric characteristics. The images come from the Kinect sensor, properly calibrated to gather the object's relative pose to the camera. The CAD model of each object is also available. The LINEMOD deals with light changes and motion blur, generating scenes polluted by other distracting objects in a fixed environment. In its extension, named Linemod-occluded, the dataset also addresses the problem of partial occlusion between the objects. Despite its extensive use, the linemod has an average of 1000 labeled images for each object, requiring data processing and

[1] http://campar.in.tum.de/Main/StefanHinterstoisser

partitioning techniques to train deep learning techniques. Also, the dataset provides a small set of synthetic images that can help train specific techniques.

Garon et al. (GARON; LALONDE, 2017)[2] provide examples for evaluating tracking models in cases of partial occlusion. The dataset contains four objects recorded in scenes where each object suffers a different degree of occlusion. The authors also provide the functions used to generate synthetic images for training, facilitating the creation of new virtual images with the same process used in the paper.

In the industry context, T-LESS (HODAN et al., 2017)[3], ITODD (DROST et al., 2017)[4], HomebrewedDB (KASKMAN et al., 2019)[5], and PROFACTOR3D (AKKALADEVI et al., 2016) bring different objects with aspects mapped to simulate industrial use cases. The scenes simulate an environment with well-controlled lighting, while objects are characterized by pieces of generic and similar shapes with a high level of symmetry. In contrast, Toyota light and TUD light[6] focus on the illumination aspect, providing scenes with variations of the ambient with 8 and 5 different lighting settings, respectively. TUD light also addresses the motion problem with non-stationary objects. Finally, the large scale datasets HOPE[7], Rutgers APC (RENNIE et al., 2016)[8], and YCB (XIANG et al., 2017)[9] were proposed for the robotic manipulation scenario. The datasets contain different warehouse objects in cluttered environments and changes in ambient light. Some objects also have similar characteristics (such as product boxes), are symmetrical in many axes, and are differentiated only by their color or texture.

Recently, the BOP Benchmark (HODAN et al., 2018) succeeded in unifying the metrics used and the datasets organization, facilitating the evaluation of new pose estimation approaches. Also, the BOP authors propose an online rating system[10] to evaluate new techniques and keep up-to-date the main state-of-the-art results on the datasets contained in the benchmark (HODAŇ et al., 2020). Nevertheless, no dataset with real information addresses environmental and device variations for the same object, aspects relevant to industry, and end-user applications. Still, it is costly to replicate the process used in each dataset (data capture, annotation, and synthetic data generation) to generate information for new use cases.

[2] http://vision.gel.ulaval.ca/ jflalonde/publications/projects/deepTracking/index.html
[3] http://cmp.felk.cvut.cz/t-less/
[4] https://www.mvtec.com/company/research/datasets/mvtec-itodd/
[5] http://campar.in.tum.de/personal/ilic/homebreweddb/index.html
[6] http://cmp.felk.cvut.cz/sixd/challenge_2017/
[7] https://github.com/swtyree/hope-dataset
[8] https://robotics.cs.rutgers.edu/pracsys/rutgers-apc-rgb-d-dataset/
[9] https://rse-lab.cs.washington.edu/projects/posecnn/
[10] https://bop.felk.cvut.cz/home/

2.3 DOMAIN ADAPTATION

Obtaining and labeling relevant data from a specific problem is essential for training learning models; many deep learning approaches need many examples for training. Most machine learning techniques rely on data provided in training to perform prediction of new datasets, assuming a similar distribution between the source domain used for training and a target domain related to the real test case (CSURKA, 2017; ALGHONAIM; JOHNS, 2021). As seen in the previous section, several large-scale datasets for object detection are available publicly containing several types of objects and scenarios for training (CALLI et al., 2015; EVERINGHAM et al., 2012; DENG et al., 2009). However, it is often necessary to generate specific datasets when dealing with a particular problem or domain not mapped on the data available.

Some approaches use transfer learning to adapt the knowledge obtained from a certain domain to a similar one. A small set of data (real or simulated) from the new domain can adjust the learned information of a model previously trained on a large dataset (PAN; YANG, 2009; HINTERSTOISSER et al., 2018). Transfer-learning is useful when the domain of interest is well known, obtaining data variations that map the test scenarios.

Still, domain adaptation approaches turn up as an alternative to mitigate the domain variability. It is similar to transfer learning approaches, aiming to improve task execution on an unlabeled domain by exploiting and augmenting relevant information from a labeled source domain. Some works propose methods to create a model with unsupervised training that can perform predictions for unseen variations in the same task or quickly adapt their parameters to infer solutions for different distributions (PATEL et al., 2015).

Recent works discuss the importance of knowing the real camera specifications and how they can impact simulating data to reproduce and adapt the real domain (LIU et al., 2020; OUYANG et al., 2021; ZAKHAROV et al., 2022). In this way, Mandl et al. (MANDL et al., 2021) propose an approach for coherent rendering capable of jointly simulating all major components of an arbitrary camera using a neural network. The method can learn from a generic set of images the camera's properties, simulating the lens effect, sensors noises and colors, and the ISP characteristics. By doing this, the authors demonstrate the improvements in rendering virtual scenes simulating fine details of real images without major loss.

In contrast, Volpi et al. (VOLPI; LARLUS; ROGEZ, 2021) shows the importance of addressing domain shifts over time. Data distribution can vary according to changes related to the environment (e.g., illumination, noise, occlusion) or the target (e.g., shape, color, appearance). So, the model must learn to deal with such variations without forgetting earlier information. To achieve this, the authors propose a meta-learning strategy simulating auxiliary meta-domains generated through augmented images to avoid model catastrophic forgetting when introduced to new visual domains.

Rozanstev et al. (ROZANTSEV; LEPETIT; FUA, 2015) propose an algorithm able to learn the rendering parameters to simulate the environment behavior, reducing the gap between the simulated and real domain. The learned parameters can create new training images of the object of interest in arbitrary 3D poses. The approach effectively simulates the blurring and motion effects for the object boundaries. Also, it can capture the image's noise and the features of the object's material. Likewise, Rozanstev et al. (ROZANTSEV; SALZMANN; FUA, 2018) extend the concept of adaptation by introducing a two-stream deep architecture. The networks are trained jointly in the two domains (i.e., synthetic and real), and their weights are related but distinct. The algorithm applies a regularization strategy to prevent the weights from being too far from each other. Zakharov et al. (ZAKHAROV et al., 2022) enhance this concept by utilizing photorealistic rendering generated by a neural network to simulate scene variations. Their approach involves training models to understand the physically based properties of objects. These models are then used to create a sampler that randomizes the values predicted into the rendering engine. By employing a CAD model, the system predicts material properties and their variations for more accurate rendering.

Furthermore, domain randomization (DR) is an alternative to extend knowledge from a source domain, creating simulated data when few or no labeled samples are available to extract knowledge. The objective of DR is to make synthetic data generate the most variation as possible from the source domain, considering new samples as just another simulated variation (TOBIN et al., 2017; ALGHONAIM; JOHNS, 2021; TREMBLAY et al., 2018; ZAKHAROV et al., 2022; CUNHA et al., 2022). So, the ML model can handle real-world challenges that are not labeled or mapped in the solution. Some techniques investigate the use of DR in the context of object detection and pose estimation, mixing simulated information with real data. Thus, DR diminishes the concern with replicating the real world's exact characteristics.

2.4 SYNTHETIC DATA GENERATION

Capturing and annotating real datasets are time-consuming and prone to constant error at multiple stages. This fact is especially true for 6DoF, as the ground truth is hard to capture, given the higher complexity of the information. Nevertheless, many deep learning approaches still require large amounts of real data to work properly due to the reality gap between virtual and real images. The reality gap makes it hard for the network to learn about the real object by seeing only a virtual version. Tobin et al. (TOBIN et al., 2017) explored the concept of domain randomization of synthetic frames to tackle the reality gap. Using it, they trained a neural network to localize simple geometric shapes on top of a table for robotic manipulation. This work also provided a robotics-oriented localization training dataset with high variability to the neural network that can generalize to real-world data. The dataset generator was built using the MuJoCo Physics Engine (TODOROV; EREZ; TASSA, 2012) and generates images by randomizing the positions and textures of the object of interest and surrounding objects, as well as randomizing illumination, camera position, distractor configuration, and the type and amount of noise present on the images. Although performing a task simpler than 6DoF pose estimation, this work is relevant as the system can achieve high accuracy (1.5 cm) without pre-training on real images.

Tremblay et al. (TREMBLAY et al., 2018), from NVIDIA, also used DR to present a system for training deep neural networks for 2D car detection on outdoor scenarios. They render scenes using Unreal Engine (UE4) by randomly changing the objects' number, type, texture, camera translation and rotation, illumination, background, and floor textures. Also, it includes a set of distractors, either contextual distractors (objects similar to possible real scene elements, positioned randomly but coherently) or flying distractors (geometric shapes with random texture, size, and position). Their network presented better results when using the synthetic DR dataset mixed with real images. Other alternatives tested were using only real images and using a mix of real and realistic (non-randomized) synthetic data from the VKITTI dataset (GAIDON et al., 2016). Even though this work does not perform 3D detection or 6DoF estimation, it is a step forward in this direction, as the complexity of the objects and scenes is very significant.

NVIDIA also published their Deep learning Dataset Synthesizer (NDDS) (TO et al., 2018), a UE4 plugin that can generate synthetic data in real-time by randomizing lights, objects, cameras, poses, textures, and distractors. The plugin can export images, segmentation, depth maps, object pose annotations, bounding boxes, keypoints, and custom stencils. The Falling Things dataset (FAT) (TREMBLAY; TO; BIRCHFIELD, 2018) uses NDDS and focuses on the context of 6DoF object detection. Each photorealistic image consists of a stereo frame pair with corresponding depth images, 3D poses, bounding boxes, and the segmentation image of each scene's objects. The objects were extracted from the YCB dataset (CALLI et al., 2015) and used in 3 virtual environments within UE4 in random positions that change after the objects fall into the scenes. However, it does not perform DR.

The Synthetic Image Dataset for 3D Object Pose Recognition with Distractors (SIDOD) dataset (JALAL et al., 2019) is similar to the FAT dataset but with the presence of flying distractors (see (TREMBLAY et al., 2018)) and domain randomization. Lee et al. (LEE et al., 2019) extended NDDS to handle robotic joints and export the joint information to the network training. Still, all these NDDS datasets are synthetic, without similar real frames to compare against, so they are best suited for fully synthetic approaches.

Hinterstoisser et al. (HINTERSTOISSER et al., 2019) proposed a strategy that generates cluttered synthetic frames and trains a network to perform 2D detection of complex objects. Though the scenes are cluttered, the technique guarantees that the objects are presented equally to the neural network. The model trained with only synthetic data outperformed the model trained with only real data. The authors also investigated individual aspects of the image generation pipeline, which revealed that blur and illumination color are the scene aspects that influence results the most.

In the 6DoF context, Tremblay et al. (TREMBLAY et al., 2018) proposed a DOPE deep learning approach that infers the 6DoF poses of known objects from a single RGB image without requiring refinement. The system uses a combination of photorealistic images from the FAT dataset (TREMBLAY; TO; BIRCHFIELD, 2018) and non-photorealistic domain randomized frames with varying types of flying distractors (cones, pyramids, spheres, cylinders, partial toroids, arrows), randomized illumination, occlusion, and distractor and background textures. Their DOPE approach had comparable performance to a state-of-the-art network trained using a mix of real and synthetic data and was the first deep 6DoF work to achieve such precision levels with only synthetic data. Our work aims to deepen our understanding of how these data distributions influence a network. In particular, investigate how DR alone can make networks easier to train and more robust to challenges. The motivation is that photorealistic frames (such as those from the FAT dataset) are hard to generate and can be comparable in difficulty and time consumption to recording and annotating real data, depending on the challenge. Thus, it is important to understand how the simpler way (domain randomization) alone can impact results.

2.5 NEURAL RENDERING

Recently, numerous approaches have emerged within the scope of neural network rendering (MANDL et al., 2021; ZAKHAROV et al., 2022; LI et al., 2023; YEN-CHEN et al., 2021; GE et al., 2022). These approaches utilize models to predict object or scene geometry, either from a set of real images or CAD models, enabling the generation of synthetic labeled images. Within this landscape, various strategies for image generation have been explored, including the utilization of generative networks (KIM et al., 2024; HOU et al., 2024; LIU; LUO; LIU, 2021), differentiable functions (LIN; WANG; LUCEY, 2020; ZAKHAROV et al., 2020), and neural radiance fields (LI et al., 2023; BARRON et al., 2022; MARTIN-BRUALLA et al., 2021; BIAN et al., 2023; VERBIN et al., 2022).

NeRF (Neural Radiance Fields) (MILDENHALL et al., 2021) was initially proposed for synthesizing novel viewpoints of entire scenes. The model employs a neural network to encode a 5D input vector containing the spatial location and viewing direction of a point, returning a view-dependent volume density and emitted radiance at that location. By querying these coordinates along camera rays, classical volume rendering can project the output into an image. Since the process is fully differentiable, it only requires a set of images with known poses for training. This model has shown impressive results in rendering photorealistic novel views of complex scenes and has become seminal in the field of neural rendering for view synthesis.

However, NeRF has limitations. Once the features are learned, the scene properties cannot be changed. Additionally, the model struggles to replicate certain scene aspects, such as reflections in materials. Another significant drawback is that querying the rays is a demanding process, requiring substantial time and resources for training and rendering images. Over the years, numerous works have attempted to improve these characteristics, such as optimizing inference through region sampling (YEN-CHEN et al., 2021; PERAZZO et al., 2023; SHAFIEI et al., 2021; LIU et al., 2020; DENG et al., 2022; CHEN et al., 2022), enhancing reconstruction quality (BOSS et al., 2021; SRINIVASAN et al., 2021; WIZADWONGSA et al., 2021; ZHANG et al., 2021b), or improving its representation (MARTIN-BRUALLA et al., 2021; VERBIN et al., 2022; BARRON et al., 2022; BIAN et al., 2023; ZHANG et al., 2021a).

As the representation capacity of NeRF improved, it began to be applied in various computer vision contexts (GAO et al., 2022; CAO et al., 2024; LI et al., 2022; CLEAC'H et al., 2023; XU et al., 2024). Recently, iNeRF (YEN-CHEN et al., 2021) proposed inverting the Neural Radiance Field to perform pose estimation. By leveraging NeRF's differentiable characteristics, iNeRF optimizes an unknown pose by iteratively comparing the resulting image with the target image, attempting to minimize the photometric difference between pixels. NerfPose (LI et al., 2023) extended this idea by using NeRF's implicit representation in place of CAD models in the pose estimation pipeline. This allows for learning object pose estimation from a few annotated images with known poses without relying on CAD structures. With the advancements in NeRFs, several works were being proposed on the scope of pose estimation (WEN et al., 2023; SU et al., 2021; LIN et al., 2021; BIAN et al., 2023). In this work, we aim to evaluate a novel strategy that utilizes photorealistic randomized synthetic generation to mitigate target domain variations in monocular deep 6DoF pose estimation while preserving source features to reduce the domain gap. By combining NeRF reconstruction with domain randomization techniques, we hypothesize that it is possible to achieve accurate pose estimation models with a reduced reliance on real data and without prior information from CAD models.

3 DATA ACQUISITION AND ANNOTATION

As previously mentioned, many existing datasets focus on specific problems. However, only some offer examples that address distinct challenges for different objects, making evaluating techniques in varied scenarios difficult. No public dataset currently addresses variations in environment or devices for the same object (real or synthetic), which are crucial issues for end-user applications. To address this gap, we created a new dataset, **C3PO** (cross-device dataset of 3d printed objects), specifically for 3D printed objects, structured to present different challenges in each sequence (CUNHA et al., 2022). Each scene in the dataset includes specific variations in environmental characteristics (such as lighting, occlusion, and fast motion) and object characteristics (including color, geometry, and scale). We recorded all videos using three different cameras, each with distinct properties. The videos taken with different cameras maintain consistent scene parameters (i.e., the same configuration for the environment and object), allowing for the assessment of how a model trained with one camera performs when tested with another under the same conditions. In total, the dataset comprises $108,330$ annotated frames (real and synthetic). The dataset, including images, labels, and objects, is publicly available on GitHub.[1]. The following sections describe the environment configuration and frame generation in detail.

3.1 CAMERA CALIBRATION

We used different back-facing mobile cameras, namely, an Apple iPhone X and a Samsung Galaxy S8. Additionally, we included a DSLR Canon EOS 80D with an 18mm lens. In the following sections, we will refer to these cameras as **X**, **S8**, and **EOS**, respectively. The sequences were recorded at a resolution of 1920×1080 and a frame rate of 60 FPS. Camera parameters were recovered using a script for camera calibration available through OpenCV [2]. For each camera, we obtained the focal length $(f_x,\ f_y)$ and principal points $(c_x,\ c_y)$. These parameters collectively constitute the camera's intrinsic matrix (K), as defined by the matrix in Equation 3.1. Additionally, we recover the camera's distortion coefficient vector $(\hat{d})$ defined by the radial (k_1, k_2, k_3) and tangential (p_1, p_2) lens distortions 3.2.

[1] https://github.com/o-kbc/eC3PO
[2] https://github.com/lvrma/CaLib

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{3.1}$$

$$\hat{d} = [k_1, k_2, k_3, p_1, p_2] \tag{3.2}$$

It is important to note that we used the default setup for the EOS camera, given its wide range of configurable parameters, which are not customizable on smartphones (e.g., ISO, shutter speed, aperture size). To address the general case where the model needs to handle various conditions, we maintained the automatic settings available on each device. Despite ensuring consistent viewpoints and locations, using the EOS introduces an additional challenge due to the variability in image processing during acquisition. Figure 3.1 shows examples of images taken by each camera. The images were captured from the same position and at the same time, using the default camera setup. A quick visual analysis reveals how the field of view and color representation can vary between sensors due to differences in digitalization and internal processing pipelines. These variations can significantly affect how the RGB data is represented.

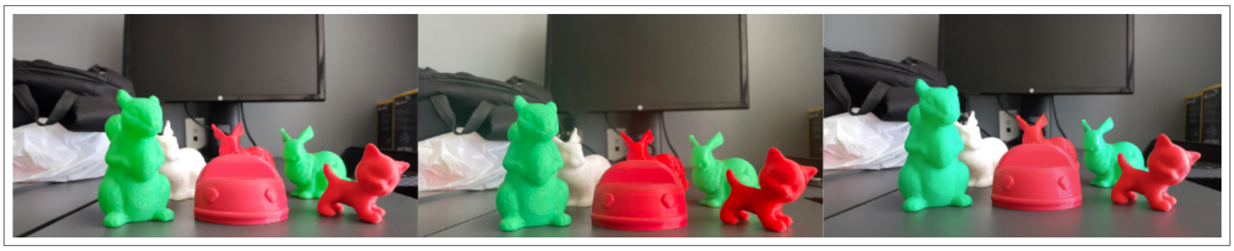

Figure 2 – Images taken for each camera used. From left to right: Galaxy s8, IPhone X, EOS 80D.

Source: CUNHA et al. (2024)

3.2 POSE-MARKERFIELD

Our real data acquisition tool, called **pose-markerfield**, was developed in C++ using OpenCV and OpenGL for frame annotation. Unlike state-of-the-art approaches, our tool annotates frames using only RGB information. This capability allows us to use a wide variety of cameras, including different mobile cameras, personal webcams, or monitoring cameras, to generate data and validate techniques across diverse real-world applications. Additionally, setting up scenes and generating annotated frames is straightforward to replicate on any standard device.

For each object, we create a field of ARuCo markers (GARRIDO-JURADO et al., 2014). At the center of each marker field, there is a designated area that defines the location of the real object during recording. The total size of this area is based on the object's dimensions obtained from its CAD model. Since we cannot directly recover the depth, the CAD model's pivot is repositioned before the annotation process so that its z-axis value aligns with the plane of the marker field. This pivot adjustment simplifies the annotation process, requiring only the x and y axes to be adjusted to find the initial pose, which may vary due to different camera offsets. By knowing the initial camera position (relative to the marker field) and its calibration parameters (e.g., K and $\hat{d}$), we can adjust the annotation to recover the relative camera-object rotation (r) and translation (t) vectors for each frame, defined in Equation 3.3. Figure 3.2 shows an example of the marker field used and an object placed on it for data acquisition.

$$r = [r_x, r_y, r_z]; t = [t_x, t_y, t_z] \tag{3.3}$$

An initial offset is applied to the object's position to align with the camera's principal point shift. Similarly, the object's scale is adjusted to match its real dimensions, ensuring the correct projection of the CAD model. All values for position (X, Y, Z), rotation $(roll, pitch, yaw)$, and scale (o_s) are parameterized and corrected during the annotation process. The user must validate the initial pose before beginning the annotation process. Once the initial pose is confirmed, the system iteratively processes all frames, capturing the relative movement between the camera and the marker field. Given the known real dimensions of the object, the size of the markers, and the object's position within the marker field, we can calculate the camera-object relative position based on the estimated position of the marker field.

By projecting the CAD model using the 6DoF pose, we can segment the pixels corresponding to the object, removing the background and creating a binary segmentation mask. The segmented frame is then combined with a random background from the VOC dataset (EVERINGHAM et al., 2012), given the constant marker field beneath the object. Changing the background is necessary to avoid overfitting, as we do not want the model to learn the markers as part of the object. The projected 3D model's depth map is also stored to supplement

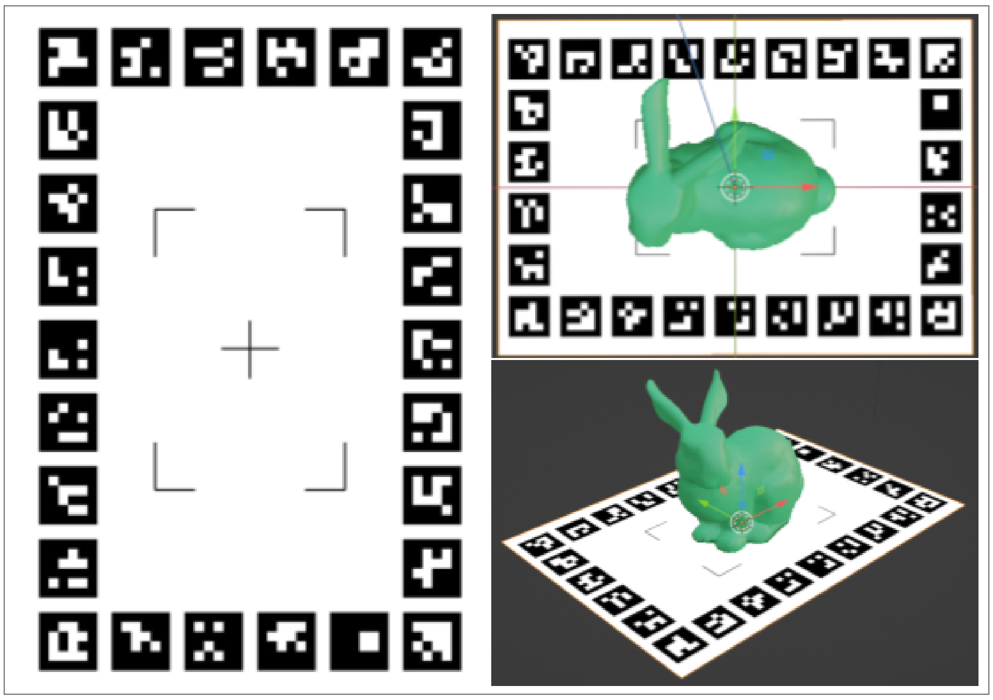

Figure 3 – Example of the markerfield created for the data acquisition. The object is placed on the central area marked on the field, and the CAD model pivot is adjusted to the same position of the markerfield origin (Z=0).

Source: CUNHA et al. (2024)

the lack of RGB-D data. Using the CAD's 3D structure, we can create a pseudo-depth map of the object. While this does not account for other objects in the scene, it is helpful for the randomization generation. Finally, using the 6DoF pose and CAD data, the eight corners of the object's 3D bounding box can be projected onto the frame according to the relation specified in equation 3.4, where $P_{2d}^T = [x, y, w]$, $P_{3d}^T = [X, Y, Z, W]$, and R is defined by the rotation matrix corresponding to the rotation vector r. This initial pose and points are stored along with the segmentation mask, depth map, and new background. Figure 3.2 illustrates the results obtained from the pose-markerfield process.

$$P_{2d} = K[R|t]P_{3d} \tag{3.4}$$

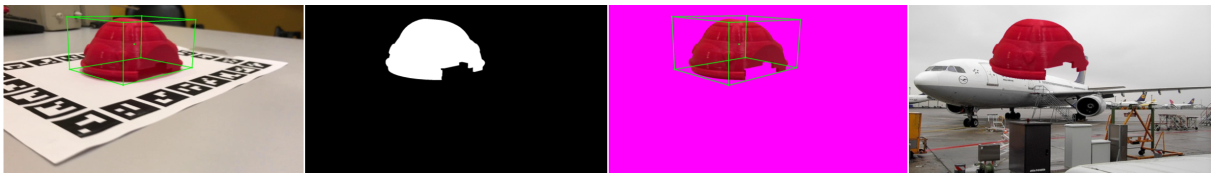

Figure 4 – Result obtained in the annotation using the pose-markerfield. From left to right: original frame with the estimated bounding box corners; segmentation mask obtained through the CAD model projection; Filtered image with no background; Result of the frame augmentation merging a random background with the filtered image.

Source: CUNHA et al. (2024)

3.3 ENVIRONMENT CONFIGURATION AND SCENES GENERATION

The cameras were placed on a stabilizer during recording to prevent unwanted jitter. Each sequence was recorded with the camera pointed toward the object at a fixed distance, capturing viewpoints in a half-sphere pattern around the object and the surface plane it stood on. The recorded sequences encompass a wide range of viewpoints, camera rotations, and challenges such as motion blur and varying illumination conditions, including a moving point light in a dark room, indirect sunlight, and diverse indoor lighting patterns.

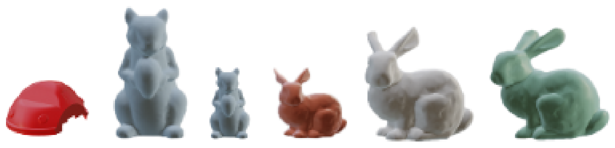

Figure 5 – Printed object used in this work. In total we have three for train and test and three objects used exclusively in test.

Source: CUNHA et al. (2024)

The objects used for recording were made from solid-color Polylactic Acid (PLA), a standard 3D printing material known for partially reflecting the scene's illumination. Three objects were selected for analysis in each **training configuration**. The **bunny** object, printed with green filament, is the largest, measuring ($18 \times 20 \times 18$ centimeters) for ($width \times height \times depth$). The **car** object, printed with red filament, has a smaller size, measuring ($7.5 \times 6 \times 10.5$) centimeters. The **squirrel** object, featuring the most complex geometry, measures ($10.5 \times 18 \times 16.5$) centimeters and was printed with blue filament. Figure 3.3 illustrates each object.

Additionally, we have included three supplemental **test objects** (Figure 3.3): a small red bunny measuring ($9 \times 10 \times 8$) centimeters (similar filament color and scale as the car); a white

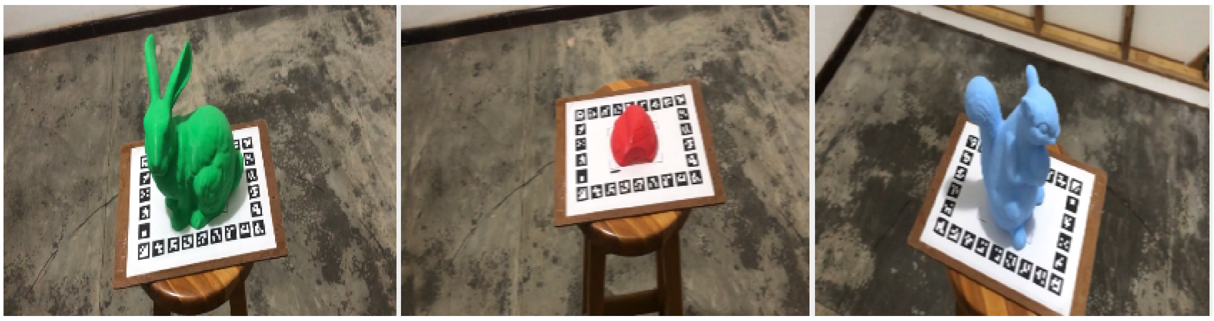

Figure 6 – Illustration of the printed objects used in the simple scenes (training) and the environment challenges occlusion, motion, and illumination (test). From left to right: Green bunny; Red car; Blue squirrel.

Source: CUNHA et al. (2024)

bunny with the same size as the large green bunny; and a small squirrel with the same color as the original squirrel but measuring $(3.5 \times 6 \times 5.5)$ centimeters. We have used these objects to create scenes to evaluate how color, geometry, and scale variation affect the models. As mentioned, the three main objects in their basic appearance (a green bunny, a red car, and a blue squirrel) are used in training, recorded in the simple sequence, while their variations are used exclusively for testing.

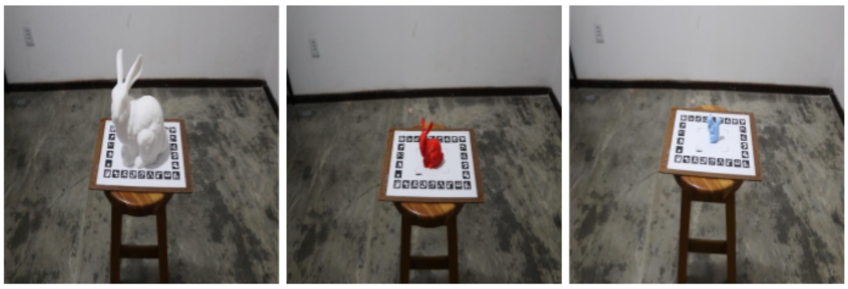

Figure 7 – Illustration of the printed objects used in test scenes. From left to right: White bunny (color); Red bunny (color-scale); Blue squirrel (scale).

Source: CUNHA et al. (2024)

The specific sequences are organized below by environment challenge and object variation. Figure 3.3 shows an example of each sequence recorded for two devices.

- **Simple**: The camera moves around the object with slow and controlled movements. Indoor illumination is bright. These sequences were split into distinct train and test sets;
- **Motion**: Same as Simple but with faster camera motion, introducing motion blur and shaking;

- **Occlusion**: Same as Simple, but with occluders around the main object, obstructing part of the view;
- **Dark**: Same as Simple but with all indoor lights turned off and only a small amount of ambient light leaking into the room, allowing for the camera to see the object barely;
- **Lantern**: Same as Dark, but the environment light varies abruptly around the object. The scene is set in a dark room with a focused, moving spotlight from a lantern moving over parts of the object at random;
- **Scale**: Same as Simple, but the object is of an unexpected scale. This challenge was done only for the blue squirrels;
- **Color**: Same as Simple, but the object of interest has an unexpected color, although its geometry remains the same as seen in training. We have only recorded this scene for the bunnies.

This research uses frames from the simple variation for training and testing, while frames from other sequences are used exclusively for testing. In our notation, "**Sequence**(**Camera**)" refers to the sequence type and its camera. For example, **Simple(S8)** refers to a Simple sequence recorded using the Galaxy S8.

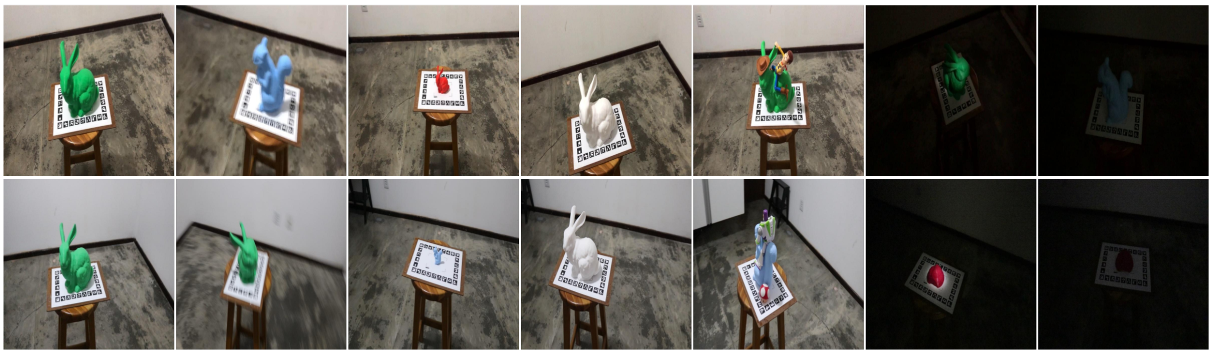
Figure 8 – Example of the real dataset generated for the test cases. The top and bottom images refer to iPhone X and EOS 80D, respectively. The challenges are ordered from left to right, such as simple, motion, scale, color, occlusion, lantern, and dark.

Source: CUNHA et al. (2024)

The table 1 shows the image distribution across scenes for each camera. As described earlier, the variations include simple (test), occlusion, motion, lantern, and dark, containing images for the three main objects (Figure 3.3). Color and scale are represented by the additional

Table 1 – Dataset distribution for all cameras and scene variations. The last column shows the total number of images for each scenario. The last row displays the number of frames for each camera.

Scene	Galaxy S8	IPhone X	EOS 80D	Total Scene
Simple	5139	5149	5177	15465
Simple (test)	2204	2208	2222	6634
Motion	533	545	507	1585
Occlusion	3744	3644	3694	11082
Color	2508	2532	2485	7525
Scale	2498	2567	2480	7545
Lantern	3781	3792	3744	11317
Dark	3794	3689	3694	11177
Total Camera	24201	24126	24003	–

objects (Figure 3.3). The size of motion is smaller than other scenes, as we keep the same procedure to acquire the viewpoints while walking fast around the object. In total, we have $24,201$, $24,126$, and $24,003$ images for the Galaxy S8, iPhone X, and EOS 80D, respectively. Overall, we have $72,330$ real labeled images.

4 ANALYSIS ON THE DATA DISTRIBUTION

We introduce the C3PO dataset to assess our use case within the context of 3D-printed objects. This scope is particularly challenging for the textureless nature of such objects, which often have fewer discernible features (CUNHA et al., 2020a; CUNHA et al., 2022). Through this dataset, we aim to evaluate the performance of deep learning models when faced with unseen challenges and scenarios not included during training while exploring potential alternatives for enhancing inference. This chapter analyzes the dataset's distribution, elucidating the variations among samples used for training and testing models for each challenge and device.

As detailed earlier, RGB-based pose estimation methods can extract valuable insights from the image's pixel data, making them versatile for various applications and device types. However, relying solely on visual cues from RGB channels could impact the model's accuracy. RGB represents an additive color space where all channels are correlated based on the intensity of light hitting the surface, resulting in a blend of chrominance and luminance details. Variations in captured images can stem from covariance changes induced by environmental factors such as low illumination, sunlight, or blur, as well as device-specific characteristics like lens effects, field of view, chromatic aberration, sensor noise, and color.

Figure 4 shows the simple scenario created for the C3PO dataset, featuring slow and controlled movements around the objects. The figure presents examples of each object (bunny, car, and squirrel) captured using three cameras (EOS 80D, Galaxy S8, and iPhone X). Upon visual inspection, it becomes apparent that variations in represented color, sharpness, and field of view cause the primary discrepancies among the images from different cameras. Notably, the EOS 80D camera contains a wider field of view than the other devices. At the same time, the iPhone X tends to produce images with brighter tones even under similar environmental conditions. As mentioned earlier, we have limited control over the processing of frames on the iPhone X and Galaxy S8. Each camera follows a specific preprocessing pipeline, which includes adjustments in color representation to balance the captured light from the environment. Also, variations in lenses and sensors may introduce additional distortions or noise that the device processes before rendering the final result. Additionally, lens positions and aperture sizes are typically fixed values in smartphone devices, with corrections applied post-capture by software. While we can physically control some parameters of the EOS 80D, we opted to keep them fixed for simplicity. Nevertheless, a visual comparison of each image in figure 4 shows noticeable differences in how objects are represented. While such differences may seem insignificant in the context of 2D detection, they can significantly impact inference in 6DoF pose estimation. This perception is because we have more free variables to estimate, requiring millimetric precision while estimating the pose. Therefore, it becomes a challenge for RGB-only models to handle these variations effectively, especially when training data is limited. Insufficient training samples can lead to overfitting, emphasizing the importance of dataset size in optimizing models.

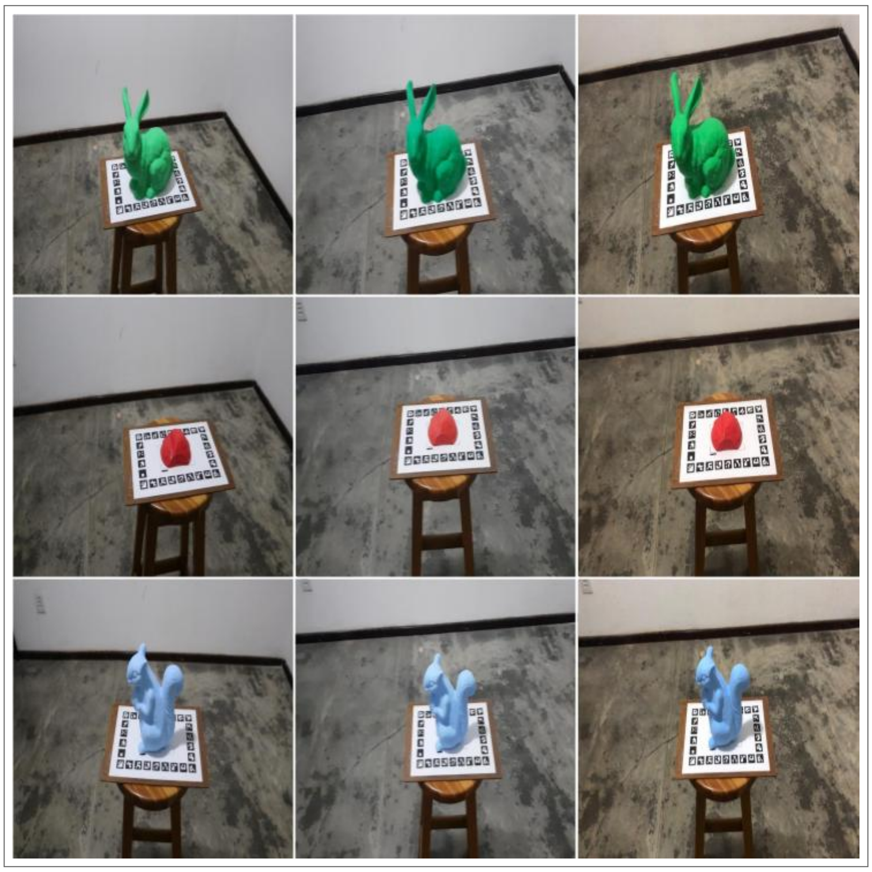

Figure 9 – Illustration of the differences between the images of each object and cameras used in this work. Objects are ordered from top to bottom as bunny, car, and squirrel. The images for each camera are ordered from left to right as EOS 80D, Galaxy S8, and iPhone X, respectively.

Source: CUNHA et al. (2024)

Figure 4 illustrates the density plot histogram over HSV space. The dataset is sampled to calculate channel density, cropping the area delimited by the object's dimension. The values are grouped, correlating each channel on the 2d histogram, relating the channels H (Dominant Wavelength) and S (color purity). H and S direction variations remain large for all devices, although they change at different ranges for the S channel. In any case, these punctual differences can be easily corrected with techniques to augment the data on training, adding robustness to such variations.

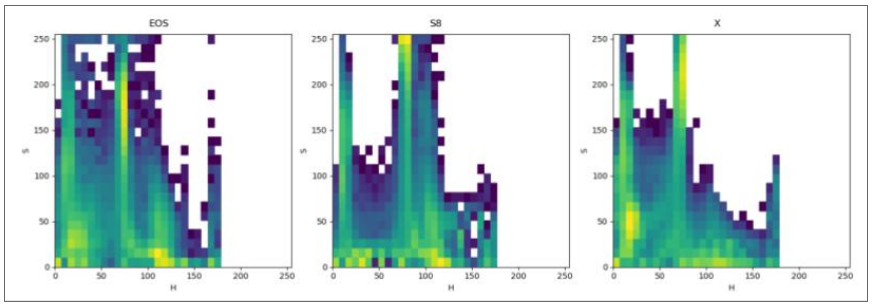

Figure 10 – 2D histogram density plot using HS channels for the EOS 80D, Galaxy S8, and iPhone X with the bunny simple scenario.

Source: CUNHA et al. (2024)

In motion scenes containing faster movements, the overall behavior remains quite similar to that of the simple scenario. As expected, the variation in speed does not significantly alter the distribution of colors quantitatively. However, as depicted in Figure 4, the variation becomes more uniform due to blurring, resulting in a compact plot (right column). As we can see, there are slight changes in how color is distributed merely by altering the movement speed. While the color distribution remains uniform, the RGB frames (left column) reveal that fast movement causes motion blur and can hide significant features that might otherwise be extracted from pixels. This observation presents a challenge, resulting in a loss of object details that could be discerned.

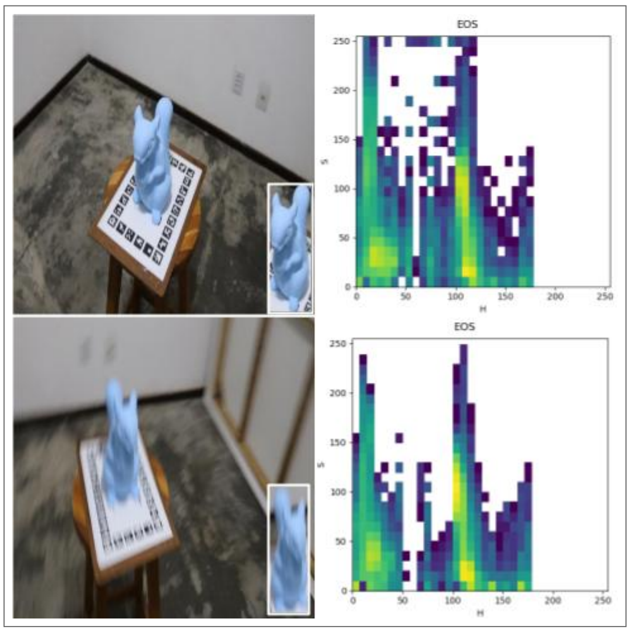

Figure 11 – Simple (top) and motion (bottom) scenario using EOS camera. The left part illustrates the 2D histogram density plot using HS channels for each scenario.

Source: CUNHA et al. (2024)

For occlusion, as depicted in Figure 4, the color variation within the object area becomes pronounced. The occluding elements make it challenging to discern a clear pattern within the target area, potentially complicating this scenario for monocular RGB-based methods. Additionally, in Figure 4, apart from the obstruction in the object's geometry, we observe a nearby object with similar material and color to the target, which could distract the detector.

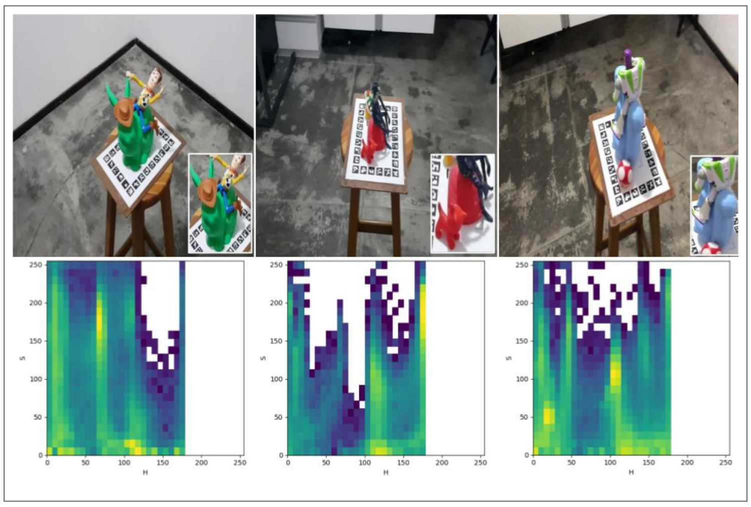

Figure 12 – Top images illustrate the occlusion scenario for each object. The bottom ones show the 2D histogram density plot using HS channels for each correspondent object in the same column.

Source: CUNHA et al. (2024)

As expected, the color scenario has a significative variation akin to the occlusion case. However, the regions of higher color densities are dispersed across different ranges. Unlike the simple scenario, this substantial deviation presents a more complex challenge for data augmentation strategies, as each distribution behaves differently. Figure 4 illustrates the color scenario for the bunny object captured with the EOS 80D camera. This complexity brings a significant challenge for RGB-based pose estimation, as color is essential when extracting features from pixels for textureless objects.

The lantern and dark scenarios are the most challenging for capturing RGB information. These scenes suffer from different types of illumination, including light color, position, and intensity variations. As depicted in Figure 4, contrary to the occlusion case, the density plot reveals minimal information about the image color. In these instances, particularly in the dark scenario, discerning any characteristic in the frame, including geometric details, proves exceedingly tricky.

Lastly, we compute the Structural Similarity Image Metric (SSIM) (WANG et al., 2004; HORE; ZIOU, 2010; MANDL et al., 2021). The $SSIM(x, y)$ is defined by equation 4.1, where for

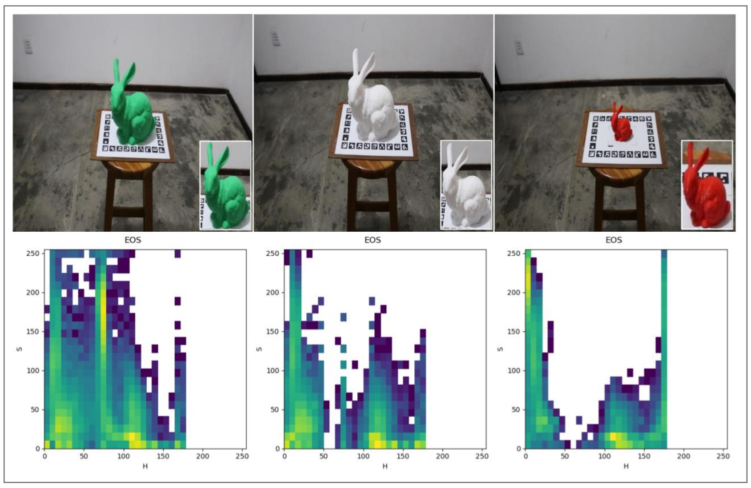

Figure 13 – Top images illustrate the color scenario for the bunny object. The bottom ones show the 2D histogram density plot using HS channels for each correspondent object in the same column.

Source: CUNHA et al. (2024)

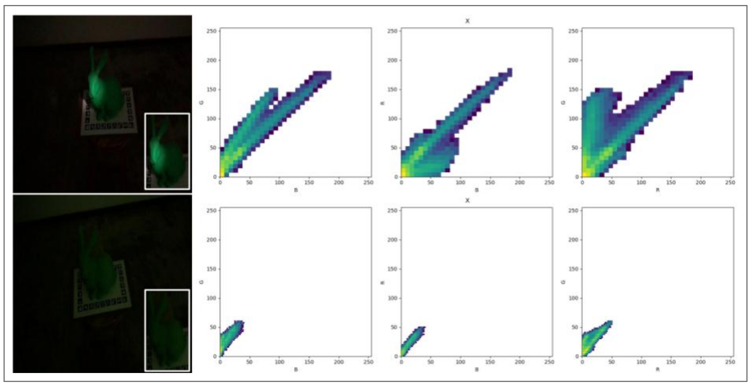

Figure 14 – The top left and bottom left images correspond to the lantern and dark scenario using the bunny and iPhone X. The top right and bottom right are their correspondent density plots for the RGB channels.

Source: CUNHA et al. (2024)

an input image x,y the values μ_x, μ_y, δ_x, δ_y, and δ_{xy} represent, respectively, the pixel sample mean of x, the pixel sample mean of y, variance of x, variance of y, and the covariance of x and y. c_1 and c_2 are two variables that stabilize the division with a weak denominator defined by the dynamic range of pixel values. SSIM is utilized for image quality assessment and can measure the similarity between two images. We utilize the simple scenario (training distribution) as a reference to acquire the metric values for each scene variation (test distribution). As mentioned in section 3.3, training exclusively applies images from the simple scenario, while the other scenes are reserved for testing. We compute both metrics within the area delimited by the object's dimensions, similar to the density plot.

$$SSIM(x, y) = \frac{(2\mu_x\mu_y + c_1)(2\delta_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\delta_x^2 + \delta_y^2 + c2)} \tag{4.1}$$

Table 2 displays the average value for each camera and the overall average for all scenes. Although the results highlight significant differences between the test scenarios and the training set, the obtained values confirm the previous analysis: the motion scenario aligns closely with the training distribution, whereas the scenarios involving occlusion, color variation, and scale exhibit more pronounced discrepancies from the training set, making them more challenging to deal. Lantern and dark cases are the most challenging, with very low similarity levels concerning the training set.

Table 2 – Results taking the simple scenario (training set) as a reference to calculate the SSIM value for each test scenario.

	Motion	Occlusion	Color	Scale	Lantern	Dark
Model	SSIM	SSIM	SSIM	SSIM	SSIM	SSIM
AVG-S8	0.4117	0.2476	0.2979	0.2845	0.2056	0.1779
AVG-EOS	0.4208	0.2504	0.3143	0.2619	0.1888	0.1667
AVG-X	0.4052	0.2378	0.2983	0.2663	0.1698	0.1464
AVG	0.4126	0.2453	0.3035	0.2709	0.1881	0.1637

5 DEEP 6DOF POSE ESTIMATION

We draw inspiration from the supervised network proposed by Tekin et al. (TEKIN; SINHA; FUA, 2018) to evaluate the impact of different training data. We selected this approach because it is robust, simple to train, and publicly available[1]. Although this model does not achieve the highest overall accuracy on benchmarks like Linemod and Linemod-Occlusion, it produces results comparable to state-of-the-art RGB-based models for 6DoF pose estimation. Additionally, its structure is easily adaptable to new architectures. Since our goal is to assess the impact of data variation and synthetic data on RGB-only approaches, this method is suitable for our purposes.

The approach employs a deep end-to-end network with a monocular RGB input to estimate the object's pose. It extends the YOLOv2 (REDMON; FARHADI, 2017) model to predict the eight bounding box corners, the object's centroid, and a confidence value for each detection. The 6DoF pose is then optimized using the PnP (LEPETIT; MORENO-NOGUER; FUA, 2009) algorithm, which compares these predicted 2D projections in the image plane with the corresponding points of the 3D bounding box from the object's CAD model.

Building on the initial architecture, we updated the model by integrating a new detection architecture (i.e., YOLOv8). We extended the output layer to regress the nine keypoints corresponding to the object center and 3D bounding box. Additionally, we enhanced the network to accept intrinsic camera parameters and distortion coefficients as input, enabling it to handle multiple camera devices and domains more effectively while optimizing the pose from 3D corners. In the original model proposed by Tekin, the architecture predicts the bounding box points along with object class probabilities and confidence scores, requiring a two-step training process to optimize each component separately. Our updated architecture, utilizing YOLOv8 as the backbone, decouples the heads for predicting class and keypoints. This improvement eliminates the need for a preliminary optimization step to locate objects in the image before predicting the keypoints. Moreover, with the YOLOv8 backbone, we leverage the Spatial Pyramid Pooling (SPP) layers to extract multi-scale feature representations, enhancing the model's ability to capture spatial hierarchies and improving pose estimation accuracy.

We aim to adjust class probabilities during each optimization step using the $L_{classes}$ function as the cross-entropy loss. The original YOLO architectures use the IoU score as the

[1] https://github.com/Microsoft/singleshotpose

confidence value. However, as Tekin et al. discussed, this score is unsuitable for estimating the 3D projections. Thus, as suggested, we apply the score defined by the function $L_{conf} = c(i)$ (Equation 5.1). The confidence function $c(i)$ for a sample i returns the confidence value for a predicted 2D point (x_{pr}, y_{pr}) based on its distance $D(i)$ from the ground-truth (x_{gt}, y_{gt}). The distance must be less than the threshold d_{th}. The L2-norm (L_{points}) is used to optimize the 3D bounding box projected into the image plane, computed according to the equation 3.4. Since the prediction branches are independent, we can train both from the beginning. Our total loss function (L_{total}) is defined by Equation 5.2.

$$c(i) = \begin{cases} e^{\alpha(1-\frac{D_T(i)}{d_{th}})} & \text{if } D_T(i) < d_{th} \\ 0 & \text{otherwise} \end{cases} \tag{5.1}$$

$$L_{total} = \lambda_{cls} L_{classes} + \lambda_{conf} L_{conf} + \lambda_{points} L_{points} \tag{5.2}$$

5.1 MODEL TRAINING

The desktop used in this work has a quad-core CPU @ 3.60 GHz, 32 GB of RAM, and a GeForce RTX 3060 GPU with 12 GB VRAM. We have trained each model for 500 epochs using an SGD optimizer and a learning rate of $\alpha = 0.001$, decreasing tenfold every 100 epoch. The training initializes the models using weights pre-trained on ImageNet, which are available on the ultralytics repository [2]

We augmented the images for each epoch by applying the object mask to a random background from the VOC dataset (EVERINGHAM et al., 2012). This augmentation prevented overfitting caused by the constant marker field below the object; an example is shown in Figure 5.1. Following the approach in Tekin et al. (TEKIN; SINHA; FUA, 2018), we used a mask for data augmentation tasks such as resizing the object, changing its position, and applying random color distortions. During training, frames were uniformly scaled in width and height by a random factor of 32, within the range of 320 to 680 pixels, to increase robustness to scale changes further. Camera parameters were adjusted accordingly.

[2] https://github.com/ultralytics/ultralytics

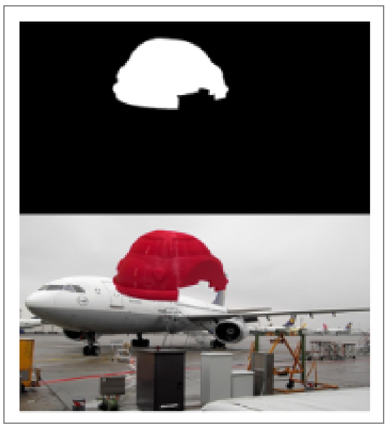

Figure 15 – Example of augmentation performed in the detector training. We remove the background with marker field information to prevent the model from learning features from the markers.

Source: CUNHA et al. (2024)

5.2 METRICS

The pose accuracy predicted was measured using 2 different metrics, described below.

5.2.1 Reprojection Accuracy (Rep.)

We consider a prediction correct if and only if the mean 2D Euclidean distance between the predicted pose and the ground truth, considering the 2D projection of all mesh vertices, is less than 5 pixels (TEKIN; SINHA; FUA, 2018). The 2D projection is defined according to equation 5.3, where K represents the 3×3 intrinsic camera matrix containing the camera calibration data, $E = [R|t]$ represents the 3×4 extrinsic parameter matrix describing the object's 3D rotation and translation, $\tilde{v} = [x, y, z, w]$ is a CAD model vertex in homogeneous coordinates, and $\pi(v) = [x/z, y/z]$ is the normalization operator which brings the object from the projective to the real space.

$$X_{2D} = \pi(K \cdot E \cdot \tilde{v}) \tag{5.3}$$

5.2.2 3D Pose Accuracy (ADD)

The accuracy is measured by the pose accuracy (ADD) or its variation for symmetric objects (ADD-S) (HINTERSTOISSER et al., 2013). The ADD is defined by equation 5.4. This expression is based on the pose error (E) and represents the average 3D Euclidean distance between all mesh vertices (V), with a length of N, when multiplied by the 6DoF ground truth pose (P_{gt}) and the predicted pose (P_{pr}).

$$E = \frac{1}{N}\sum_{i=0}^{N} ||(P_{pr} \cdot V_i) - (P_{gt} \cdot V_i)|| \tag{5.4}$$

Following (HINTERSTOISSER et al., 2013), a pose is considered correct if $E < .10d$, where d is the maximum distance between two mesh vertices, representing the CAD diameter. The ADD-S metric measures the error for symmetric objects as the average distance to the closest model point.

Figure 5.2.2 shows a visual representation of how the thresholds work to compute the errors in Rep. and ADD.

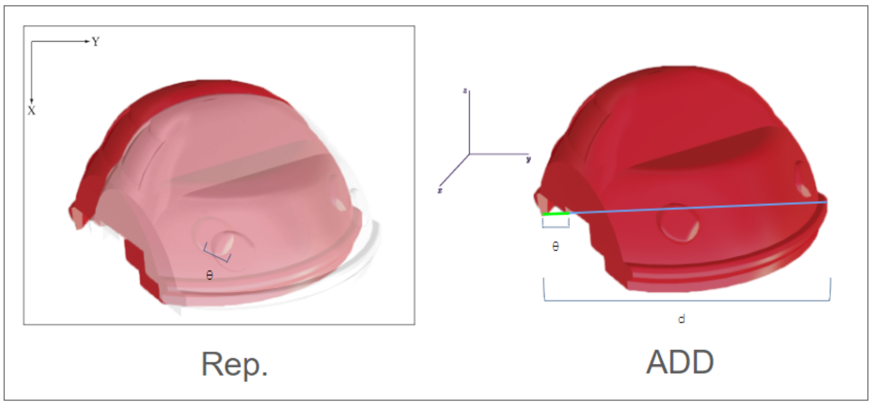

Figure 16 – Visual representation of the threshold computed in Rep. and ADD from left to right, respectively.

Source: CUNHA et al. (2024)

5.3 POSE ESTIMATION EVALUATION

This section evaluates the pose estimation performance across various challenges in the C3PO dataset. For this evaluation, models were trained using only the simple sequences for each object and camera. Thus, we can evaluate the behavior of the technique for testing situations not anticipated by the training set. We refer to models trained on the real dataset as **FullReal**, and use the notation "**FullReal(Camera)**" to specify a model trained with images from a particular camera. For example, **FullReal(EOS)** refers to the model trained with the EOS training set images.

5.3.1 Testing models on the simple sequences

Since the Simple scenes were the only ones used in training, these tests evaluated the scenario in which the model is trained and tested on the same distribution, which is the most common in machine learning techniques.

As expected, Table 3 shows that models trained on real data perform well when tested on data from the same distribution. Specifically, the models demonstrate good performance for a known camera since the test scenario closely resembles the training data. However, the accuracy decreases when a model is evaluated using an unseen camera during training. For instance, the **FullReal(X)** model in Table 3 sees its accuracy drop from 86.93% and 53.19% to approximately 48% and 21% (for Rep. 2D and ADD respectively), when going from Simple(X) to Simple(S8) or Simple(EOS). In Figure 5.3.1, we can visually observe the error by projecting the object mesh with the pose predicted by the model. The results shown in the figure were obtained by training the model on the X images and testing it on the three cameras (X, S8, and EOS).

Table 3 – Models trained and tested on the Simple scenario. Results are obtained by averaging individual scores for the three main objects: green bunny, red car, and blue squirrel.

	Simple(S8)		Simple (X)		Simple(EOS)	
Model	Rep.	ADD (10%)	Rep.	ADD (10%)	Rep.	ADD (10%)
FullReal(S8)	**84.37**	42.85	38.95	10.88	57.97	14.06
FullReal(X)	48.78	21.75	86.93	53.19	49.86	21.09
FullReal(EOS)	23.46	3.12	21.70	1.90	**79.15**	26.54

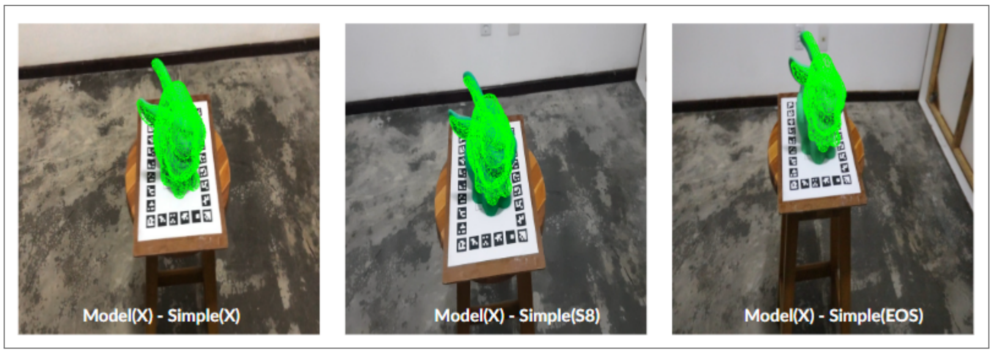

Figure 17 – Results obtained by testing a model trained on images from X and tested on all devices (X, S8, and EOS; left to right, respectively).

Source: CUNHA et al. (2024)

5.3.2 Camera variation

Table 4 illustrates the impact of varying the source of the samples used in training. The models FullReal10(S8), FullReal50(S8), and FullReal(S8) were trained with 10%, 50%, and 100% of the S8 training set, respectively. Similarly, FullReal(S8, X) and FullReal(S8, X, EOS) utilized the full datasets from their respective cameras.

Table 4 – Performance of models trained with real data from different cameras.

.

Model	Simple(S8)		Simple(X)		Simple(EOS)	
	Rep.	ADD (10%)	Rep.	ADD (10%)	Rep.	ADD (10%)
FullReal10(S8)	46.03	17.73	21.24	11.32	34.45	20.98
FullReal50(S8)	70.66	31.49	24.25	12.48	57.57	15.87
FullReal(S8)	84.37	42.85	38.93	10.88	57.97	14.06
FullReal(S8,X)	84.80	**47.34**	88.78	**39.81**	55.18	32.12
FullReal(S8,X,EOS)	**92.86**	41.85	**89.23**	30.69	**93.16**	**46.14**

The table 4 shows that the performance of the FullReal10(S8) and FullReal50(S8) models is lower than the others, as expected, due to the limited training data. Notably, the FullReal(S8) model performed worse than FullReal10(S8) and FullReal50(S8) on Simple(X) and Simple(EOS). This result indicates that models trained with more data from a single camera might overfit the specific properties of that camera, which is crucial to consider when developing such models.

Interestingly, FullReal(S8, X) and FullReal(S8, X, EOS) outperform FullReal(S8) even on Simple(S8), suggesting that training with data from multiple cameras improves the model's generalization to different camera parameters. However, labeling data from multiple cameras is costly and time-consuming, making this approach less feasible in practice.

5.3.3 Environment and object variation

From Table 5, it is evident that the model struggles to detect several variations of the environment and object characteristics. As discussed in Chapter 4, accuracy is significantly decreased for occlusion, color, scale, lantern, and dark scenarios. However, the trained models achieved similar accuracy for the motion scenes compared to the simple scenario. Curiously, there is an improvement in accuracy for the FullReal(EOS) model in the motion(S8) scenario.

In motion sequences, the faster movements introduce more blur and jitter, potentially softening specific characteristics of the S8 camera. Consequently, the model trained with images from the EOS camera performs better in this scenario. This observation extends to the FullReal(X) model, which shows a slight decrease in Rep. accuracy and an increase in ADD accuracy. Still, results for the FullReal(S8) model decrease in the Motion(S8) scene compared to the Simple(S8). While the training and test distributions are identical for Simple(S8), Motion(S8) introduces unforeseen variations that affect pose estimation accuracy.

Table 5 – Results for the specific challenges described in Section 3.3 recorded using the Galaxy S8.

	Simple(S8)		Motion(S8)		Occlusion(S8)		Color(S8)		Scale(S8)		Lantern(S8)		Dark(S8)	
Model	Rep.	ADD (10%)	Rep.	ADD (10%)	Rep.	ADD (10%)	Rep.	ADD (10%)	Rep.	ADD (10%)	Rep.	ADD (10%)	Rep.	ADD (10%)
FullReal(S8)	84.37	42.85	60.78	40.7	8.37	7.37	0.00	0.00	0.00	0.07	2.34	2.37	0.07	0.15
FullReal(X)	48.78	21.75	45.82	22.82	6.98	4.58	0.00	0.00	0.00	0.23	5.74	6.61	0.00	0.00
FullReal(EOS)	23.46	3.12	57.95	40.7	5.95	5.38	0.00	0.24	0.00	0.07	0.19	0.80	0.00	0.00

6 GENERATING SYNTHETIC DATA THROUGH DOMAIN RANDOMIZATION

As an alternative to deal with the failures caused by domain variations, we built a tool (**DR-Render**) to train new models using a domain randomization approach. The randomized data helps the estimator deal with unpredicted challenges in real cases, minimizing the influence of domain variation. DR-Render was based on the work proposed by Tremblay et al. (TREMBLAY et al., 2018), which creates deep learning models trained with a mixture of real and synthetic data to detect vehicles in images and their variations using as few real images as possible. Figure 6 illustrates our randomization approach, and the entire pipeline can be visualized in figure 6.

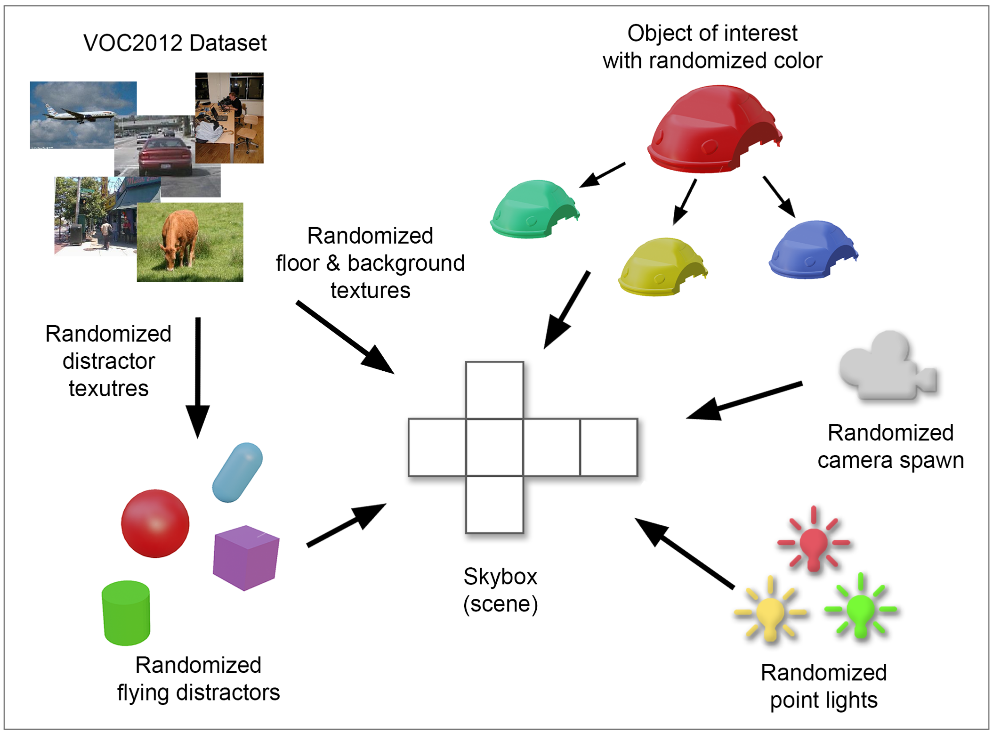

Figure 18 – Diagram illustrating the DRRender process and the objects included in the randomization during the synthetic data generation.

Source: CUNHA et al. (2020b)

Similar to the methodology proposed by Tremblay et al. (TREMBLAY et al., 2018), our synthetic scenes follow a basic structure for randomization. Each frame is rendered with variations such as altering the object's material appearance, introducing lights spawned at random locations with varying colors and intensities, and incorporating flying distractors in the form of cubes, cylinders, spheres, and capsules. Given our focus on the 3D domain, we expanded upon these variations by incorporating changes to the floor plane texture and creating a skybox, which defines the area where the object is positioned above the floor plane. The skybox and floor plane textures were changed to prevent the model from learning environmental features unrelated to the object itself. We used textures from the Visual Object Classes (VOC) dataset (EVERINGHAM et al., 2012), which offers a wide range of textures suitable for our cluttered indoor scenario. Additionally, we varied the type of lighting (point, radial, and sunlight) to simulate diverse environmental conditions and their interactions with the object, resulting in shadows at varying intensities and directions. The number of lights within the skybox limits was randomly set between 0 and 15.

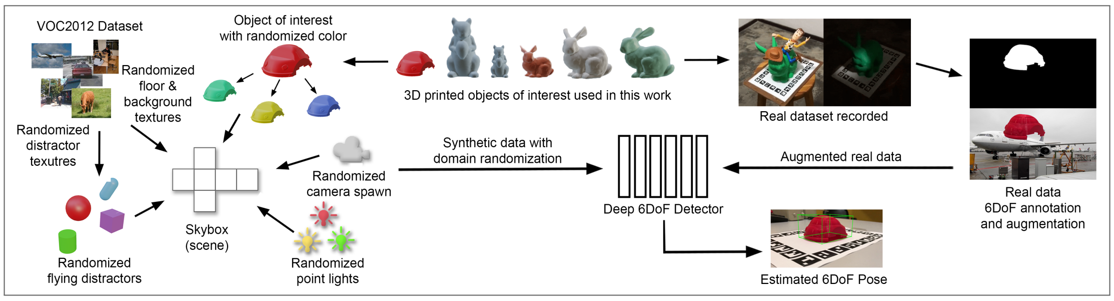
VOC2012 Dataset
Object of interest with randomized color
Randomized floor & background textures
Randomized distractor texutres
3D printed objects of interest used in this work
Real dataset recorded
Synthetic data with domain randomization
Augmented real data
Randomized camera spawn
Skybox (scene)
Deep 6DoF Detector
Real data 6DoF annotation and augmentation
Randomized flying distractors
Randomized point lights
Estimated 6DoF Pose

Figure 19 – Diagram illustrating the complete pipeline using the annotation, DR-Render, and detector for pose estimation.

Source: CUNHA et al. (2022)

Furthermore, for the 6DoF scenario, we expanded our randomization approach to simulate different parameters for the virtual cameras used to generate the data. These virtual cameras were configured with varying sensor sizes, output sizes, and intrinsic parameters (such as focal length and principal point in pixels, as defined in equation 3.1) to mimic those obtained by calibrating the physical cameras used during real data acquisition. We could generate a corresponding virtual representation for each camera by simulating its real parameters and adjusting their values as needed. Additionally, as documented in Kehl et al. (KEHL et al., 2017), randomizing the camera's position enhances robustness to multiple viewpoints during pose estimation. Consequently, the virtual camera was consistently oriented towards the origin (directed at the object of interest). Its spawn location was randomized within the skybox cube, with radial distances limited from $4m$ to $10m$, azimuth angles ranging from $0°$ to $180°$, and polar angles ranging from $0°$ to $360°$, following the spherical coordinate system. These ranges were chosen with consideration for the indoor, small object scenario addressed in this study.

Additionally, we introduced flying distractors based on simple 3D geometries to enhance robustness against occlusions. Unlike Tremblay et al. (TREMBLAY et al., 2018), whose focus was on vehicles, our scenario (involving small indoor objects) frequently encountered instances where significant occlusions occurred in frames. Consequently, we implemented an additional step in the rendering process to address this issue. To mitigate cases of total or near-total occlusion, we implemented a filter to discard frames where more than 35% of the object's 2D area was occluded. Moreover, we varied the material color of the object of interest to encourage the network to learn its structure rather than relying solely on its appearance. Given our textureless scenario, the object's color was randomized without adding texture images. Figure 6 and 6 show examples of the resulting images.

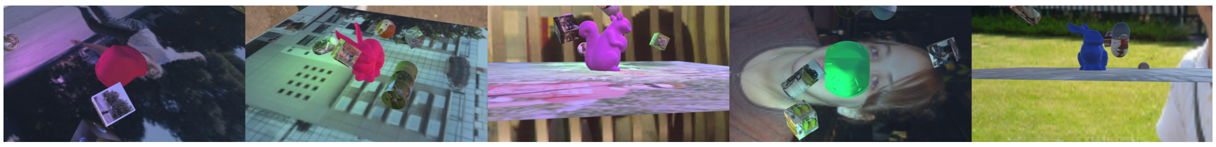

Figure 20 – Resulting frames from the DRRender generation. The process creates synthetic data by randomizing the light, positions, textures, materials, objects in the environment, and the camera parameters.

Source: CUNHA et al. (2020b)

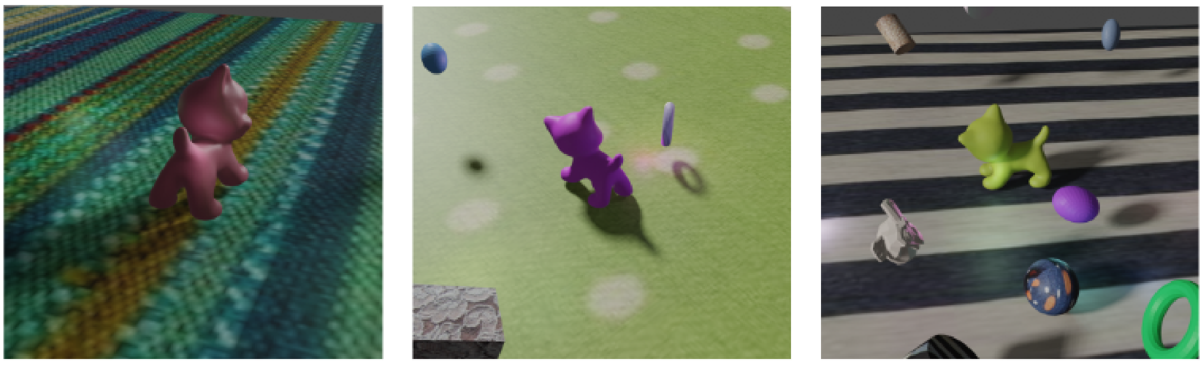

Figure 21 – Resulting frames from the DRRender generation changing rendering parameters for the same object.

Source: CUNHA et al. (2024)

7 TRAINING STRATEGIES USING DOMAIN RANDOMIZATION

To assess the influence of incorporating randomized data, we trained using various strategies combining real and synthetic datasets to refine the models. Additionally, we analyzed each combination of real and synthetic images by varying the proportion of each image domain during training. This approach aims to determine the extent to which synthetic data can be utilized while minimizing the reliance on real samples in the training process. Figure 7 provides a diagram illustrating each training variation. These strategies apply distinct portions of the datasets generated through the pose-markerfield and DRRender, enabling the identification of diverse training combinations tailored to specific test scenarios. We describe below each training strategy used.

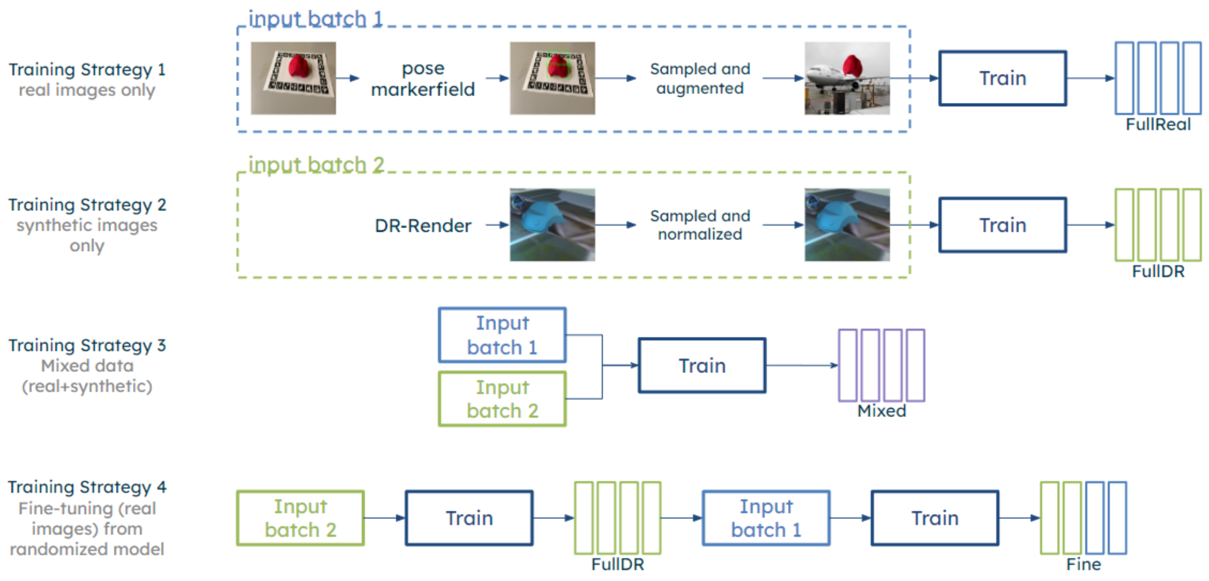

Figure 22 – Diagram illustrating the strategies for training the 6DoF detectors combining the real and synthetic datasets.

Source: CUNHA et al. (2024)

- **Full real training** (training strategy 1): A model trained exclusively with real images generated by the pose-markerfield, using images from the simple sequence as it is the unique variation exposed during training. This resulting model is referred to as **FullReal**.
- **Full DR training** (training strategy 2): A model trained using only synthetic data generated by the DRRender. The resulting model is referred to as **FullDR**.

- **Mixed training** (training strategy 3): A model optimized combining the training batches with real images and synthetic frames at random. We train models varying the portions of real data on the batches. The resulting models are named **Mixed10**, **Mixed50**, and **Mixed100**, each one corresponding to the amount of real data used (10%, 50%, and 100% of the real images available, respectively).

- **Fine-tuning training** (training strategy 4): The model training uses only the synthetic set in its entirety (FullDR); then, the model is fine-tuned with real frames in a second training step (simple sequences). The hyper-parameters to the first train and to fine-tune are the same. As with the mixed training, we have tuned models using 10%, 50%, and 100% of the real frames resulting in the **Fine10**, **Fine50**, and **Fine100**, respectively.

We executed each training strategy for every object and camera evaluated, resulting in 70 trained models. Following the notation in section 5.3, **"Strategy{ratio}(Camera)"** refers to a model trained using a particular strategy with a specific camera, using only a subset of the real samples available to train, specified by the $ratio$ value. For instance, **Fine50(S8)** denotes a model trained with DRRender synthetic data and fine-tuned with 50% of the Galaxy S8 training set. Similarly, **Mixed10(S8)** refers to a model trained using a combination of synthetic images and 10% of the real images from the Galaxy S8 training set. In all variations, the entire synthetic dataset is used when required. In total, there are $12,000$ labeled synthetic images for each geometry. Thus, the synthetic training set has $36,000$ images.

8 EVALUATION OF POSE ESTIMATION USING DOMAIN RANDOMIZATION

As in section 5.3, we evaluate the pose estimation performance given the different challenges contained in the dataset. This section replicates the same tests but compares the models trained with different strategies using synthetic data generated through the DRRender. Model training uses only simple sequences for each object and camera, and the test uses other scenarios. The training parameters and metrics are the same as in section 5.1 and section 5.2, respectively.

8.1 SIMPLE SEQUENCES

Table 6 shows that for a known camera, models trained solely on real data (FullReal) perform the best, as discussed in section 5.3.1. Introducing synthetic data or training with fewer real images (10% of the training set) slightly decreases model accuracy. However, training with synthetic images while retaining 50% or 100% of the real dataset for fine-tuning (Fine50 and Fine100) achieved comparable or better accuracy than FullReal models. In the simple scenario, synthetic images can introduce unnecessary variations, sometimes reducing accuracy.

Nevertheless, Fine and Mix models show promising results when evaluated on the same scene using a previously unseen camera during training, while the accuracy of FullReal models decreases. Although Fine(X) and Mix(X) variations do not match FullReal(X) accuracy in the Simple(X) scenario, they improve accuracy on Simple(S8) and Simple(EOS). This observation suggests that synthetic data enhances the robustness of camera changes. Models might benefit from synthetic frames simulating real cameras' internal parameters, using different colors, and ignoring distortion parameters. Notably, the accuracy variation for the EOS 80D is more significant than for the Galaxy S8 and iPhone X, likely because the EOS 80D is not a smartphone camera. Models trained with the EOS 80D face more difficulty generalizing when tested on sequences from other cameras. Therefore, in real applications, considering camera characteristics (sensor, lens distortion, color representation) and overall hardware is crucial before deciding on an approach.

The results indicate that the FullReal model is the best strategy when the training data fully maps the target distribution. The average (AVG) models in Table 6 demonstrate the significant impact of the training camera on accuracy. For instance, the model trained with sequences from the Galaxy S8 shows a drop of $20 \sim 30$ accuracy points when tested with sequences from the iPhone X and the EOS 80D. This decrease can significantly affect the end user's experience in real applications. The mentioned variation in results can be assessed in Figure 8.1, which shows the average results of each model trained on X, S8, and EOS images and tested on the entire simple sequence test set for each device. If sufficient camera-specific data is unavailable, fine-tuning and alternative approaches must be used. Synthetic data allows training with less real data (at least 50% less) while still achieving similar accuracy for the same camera or slightly better accuracy for a different camera, decreasing the annotation effort.

Table 6 – Models tested on the Simple scenario using synthetic data on training. Results are obtained by averaging individual scores for the three main objects: green bunny, red car, and blue squirrel. AVG refers to a single camera's average of all Full, Mix, and Fine variations.

	Simple(S8)		Simple (X)		Simple(EOS)	
Model	Rep.	ADD (10%)	Rep.	ADD (10%)	Rep.	ADD (10%)
FullReal(S8)	**84.37**	42.85	38.95	10.88	57.97	14.06
Mix10(S8)	35.91	15.91	14.98	10.23	29.37	13.03
Mix50(S8)	61.24	36.67	27.39	17.03	36.77	29.78
Mix100(S8)	67.74	46.73	32.98	**28.45**	44.10	30.63
Fine10(S8)	72.99	39.95	37.42	17.56	41.17	22.37
Fine50(S8)	77.19	41.38	50.65	26.54	63.60	**33.16**
Fine100(S8)	81.07	**52.91**	**51.73**	28.24	**65.46**	31.84
AVG(S8)	**68.64**	**39.48**	36.30	19.85	48.35	24.98
FullReal(X)	48.78	21.75	86.93	53.19	49.86	21.09
Mix10(X)	25.59	21.97	44.31	26.44	33.61	25.93
Mix50(X)	42.44	29.76	70.63	50.53	46.24	34.07
Mix100(X)	53.47	**43.34**	74.66	50.05	52.03	38.19
Fine10(X)	38.35	22.95	65.92	32.32	44.54	27.78
Fine50(X)	54.18	26.40	85.87	44.30	51.30	**41.57**
Fine100(X)	**55.14**	27.55	**87.52**	**56.86**	**52.16**	40.70
AVG(X)	45.42	27.67	**73.69**	**44.81**	47.11	**32.76**
FullReal(EOS)	23.46	3.12	21.70	1.90	**79.15**	26.54
Mix10(EOS)	7.09	3.89	8.58	1.94	35.12	8.825
Mix50(EOS)	17.90	7.07	22.64	8.38	64.48	32.59
Mix100(EOS)	21.59	8.74	**24.69**	9.01	70.49	**33.53**
Fine10(EOS)	22.96	11.65	17.13	9.02	63.89	15.28
Fine50(EOS)	**34.52**	**22.91**	18.45	**11.03**	72.97	24.70
Fine100(EOS)	33.89	18.30	24.53	8.03	75.69	23.96
AVG(EOS)	23.06	10.81	19.67	7.04	**66.00**	23.63

Figure 8.1 illustrates the impact of camera variation on accuracy for each object. As expected, with its complex geometry, the squirrel presented the lowest results overall. However,

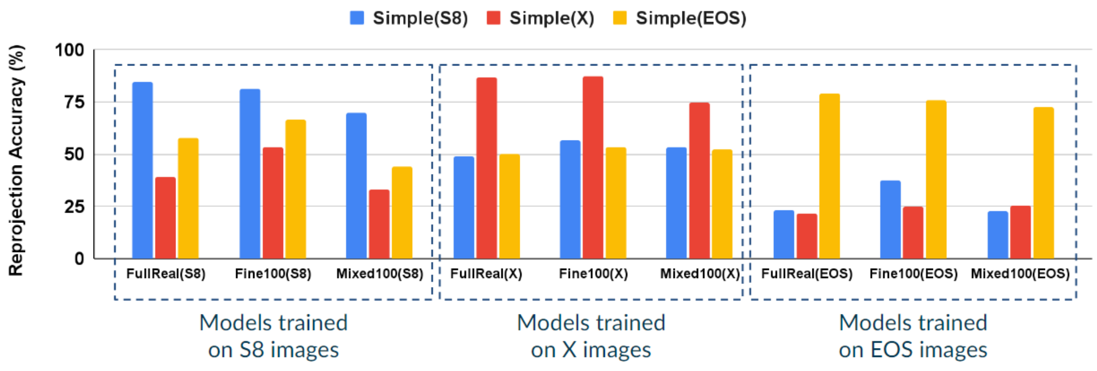

Figure 23 – Average results of all objects testing the models trained on each device in simple sequences.

Source: CUNHA et al. (2024)

the results for the squirrel are also the most balanced among the three objects.

8.1.1 Environment and object variation

Next, we discuss the challenges presented in Table 7 considering the training strategies using domain randomization. It is important to note that though we are just testing with the Galaxy S8 sequences, we are also making the same sequences publicly available for the other cameras. In this discussion, we used only one camera (S8) to isolate behaviors and variations better.

Table 7 – Results for the specific challenges described in section 3.3 recorded using the Galaxy S8.

Model	Motion(S8)		Occlusion(S8)		Color(S8)		Scale(S8)		Lantern(S8)		Dark(S8)	
	Rep.	ADD (10%)	Rep.	ADD (10%)	Rep.	ADD (10%)	Rep.	ADD (10%)	Rep.	ADD (10%)	Rep.	ADD (10%)
FullReal(S8)	60.78	40.7	8.37	7.37	0.00	0.00	0.00	0.07	2.34	2.37	0.07	0.15
Mix10(S8)	41.49	42.2	5.99	7.78	37.40	9.41	2.39	**4.78**	1.09	1.61	0.07	0.15
Mix50(S8)	47.08	**52.92**	10.58	12.57	43.04	15.08	1.51	3.18	2.53	2.66	1.19	2.63
Mix100(S8)	53.07	51.49	**18.56**	**31.73**	**47.24**	**18.58**	**3.98**	1.59	5.66	6.37	**7.65**	**9.33**
Fine10(S8)	44.56	43.46	11.97	22.15	8.05	8.84	0.79	3.01	2.66	2.34	0.00	0.00
Fine50(S8)	59.05	46.9	5.38	12.57	5.58	8.01	0.79	2.30	**8.39**	**6.94**	0.15	0.23
Fine100(S8)	**62.67**	50.66	8.98	14.37	5.45	6.53	0.87	1.59	4.71	4.76	0.31	0.69
FullReal(X)	45.82	22.82	6.98	4.58	0.00	0.00	0.00	0.23	5.74	6.61	0.00	0.00
Mix10(X)	32.51	42.2	5.79	8.38	33.22	**13.91**	0.79	0.71	2.40	4.51	0.07	0.31
Mix50(X)	39.13	51.49	10.17	14.97	38.13	10.22	1.11	**3.50**	5.73	15.01	0.23	0.55
Mix100(X)	40.15	46.92	19.16	21.55	**43.44**	6.67	**3.39**	1.19	**13.07**	**22.53**	**1.59**	**3.18**
Fine10(X)	47.24	38.58	10.77	24.55	5.80	1.60	0.79	0.79	1.77	5.04	0.00	0.00
Fine50(X)	46.14	46.69	**14.37**	**24.78**	3.64	2.24	0.79	0.81	4.76	9.28	0.00	0.00
Fine100(X)	**51.88**	**62.67**	8.98	24.55	0.00	1.08	0.15	0.79	8.23	16.14	0.07	0.15
FullReal(EOS)	57.95	40.7	5.95	5.38	0.00	0.24	0.00	0.07	0.19	0.80	0.00	0.00
Mix10(EOS)	48.34	31.81	5.99	4.19	25.02	11.66	0.79	0.31	0.00	0.16	0.00	0.00
Mix50(EOS)	61.41	60.93	9.58	12.57	**36.28**	**14.15**	0.55	0.66	0.69	0.08	0.07	0.23
Mix100(EOS)	57.48	40.94	**18.58**	**31.73**	32.75	9.89	0.23	0.71	0.56	0.08	0.31	0.47
Fine10(EOS)	64.8	50.39	11.97	22.15	6.72	8.24	0.79	0.87	0.72	0.24	0.00	0.00
Fine50(EOS)	**68.5**	**65.11**	6.58	12.57	5.07	5.63	0.79	1.03	2.58	2.82	0.07	0.23
Fine100(EOS)	63.7	62.44	8.98	14.39	5.07	4.60	0.31	1.27	1.45	1.53	0.07	0.07

In the motion sequences, the movements are faster, causing more blur and jitter. As seen

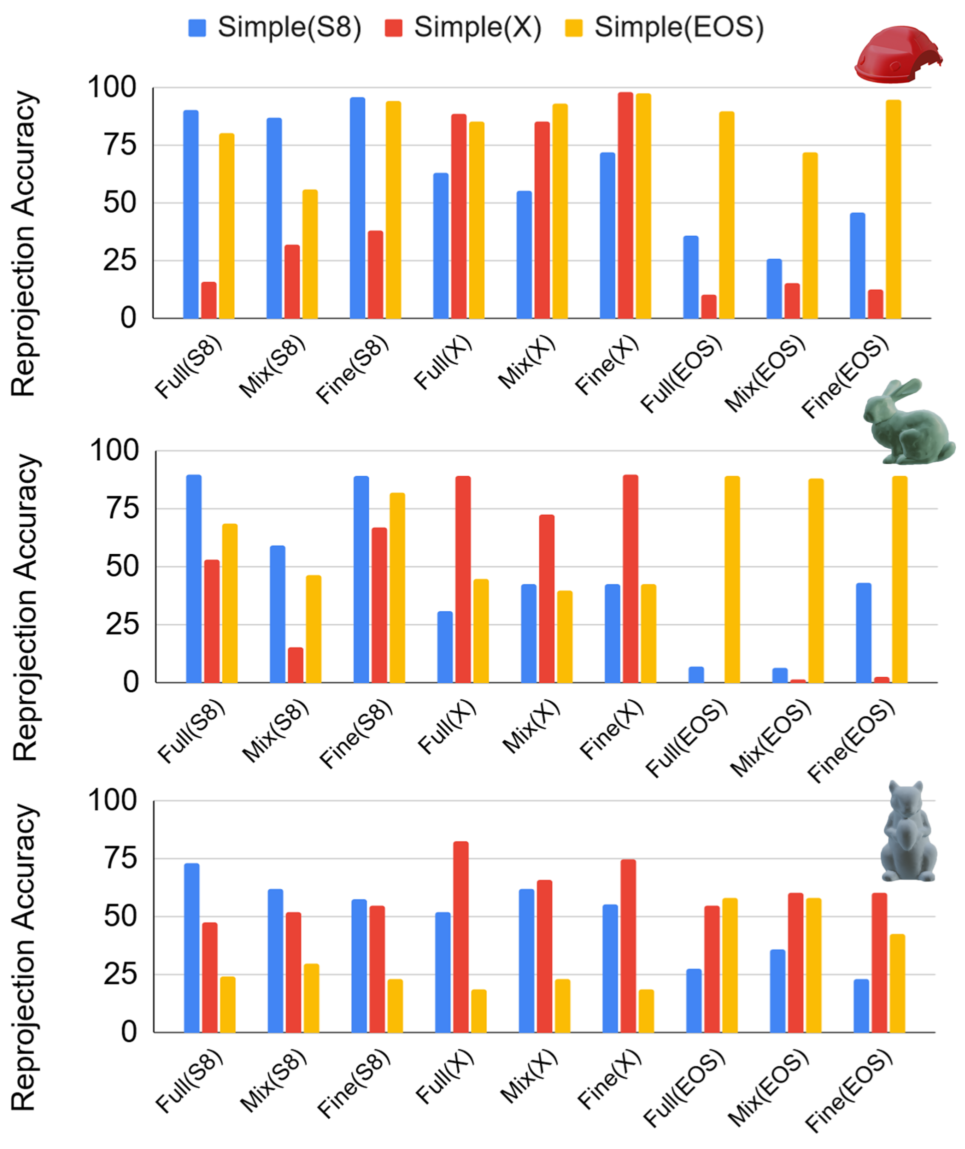

Figure 24 – Results obtained for models trained varying strategies for each type of camera. From top to bottom: car, green bunny, and squirrel.

Source: CUNHA et al. (2024)

in section 5.3.3, accuracy is lost compared to the Simple scenario. Since our synthetic scenes did not include motion blur, no model can be considered fully trained for this challenge. However, models trained with the S8 camera using the Fine and Mix variations achieved accuracy values similar to the FullReal models. Even Fine10, Fine50, Mix10, and Mix50 had accuracy values comparable to Full. Thus, we can say that training with more images that do not contain blur does not necessarily help deal with this challenge, so there is a diminishing return behavior toward the number of training examples used. As observed in section 5.3.3 for the FullReal(EOS) results, synthetic data seems to have inserted enough variation for the models trained with different cameras to enable some models to handle blurred images better.

Models using only 50% of the real dataset, like Mix50 and Fine50, nearly doubled their ADD scores compared to the FullReal models. This result supports the idea that camera parameter overfitting can be mitigated by using synthetic and real training data.

The same conclusions from section 5.3.3 remain for the occlusion test case. The results suffer a more pronounced drop in the accuracy scores. Despite this, the Fine and Mix models tend to improve overall accuracy significantly, especially Mix. Thus, using occlusions in the synthetic frames is relevant to forcing the models to learn variations in the object's geometry. Such information is unavailable in the FullReal training using simple images, resulting in FullReal models not learning to handle obstructions in the object's appearance. Even using only 10% and 50% of the real data, Mix and Fine models achieved accuracy similar to FullReal.

As seen in section 5.3.3, the FullReal models cannot detect an object of a different color, indicating that color information can quickly become highly relevant during RGB training, to the point of overfitting. This behavior happened even though we applied color distortions when augmenting the real data, albeit with the intent of adding robustness to sensor and illumination variations, not objects of a different color. However, color augmentation can be further tuned depending on the situation. Actual object color changes in synthetically rendered images are likely more effective in dealing with this variation in real frames. Nevertheless, the Fine models performed slightly better than the FullReal ones. It might be due to the constant changes in the object's material during synthetic training, causing the model to carefully ponder learning the object's color. Such information seems necessary for a model to deal with color changes. The results showed that the model could preserve these changes for the second training phase. However, the overfitting caused by real data in the FullReal models has also happened to the Fine models in their second step, obtaining less expressive results. The Mix models further confirm this idea, which has obtained the best results for the object's color variation. The Mix strategy seems to enable the model to give more weight to the object's geometry, probably due to the texture variation in the synthetic images presented during all the training. It is important to remember that such variations happen during rendering and consider aspects of the object's interaction with the ambient lighting, thus including a broader range of variations without additional adjustments.

In the case of scale, the original model is, in theory, capable of handling variations in scale, but what we see in the results of this challenge is very different. The small object used is roughly 3 times smaller in every dimension, which accounts for a roughly 27 times smaller volume, which is very significant. This difference is visible in figure 3.3, more significant than all variations inserted in the augmentation and synthetic generation. For this particular approach, the bounding box points predicted in pixels are too close for a tiny object, resulting in very high error rates. Our metrics are also harsher on a smaller object, as the diameter threshold concerning the squirrel's 3D model remains the same. The Mix models performed slightly better in this scenario than other models. The reason is similar to the case of color variation. The model is frequently subjected to a more severe scale change during all the training, using synthetic and real images. Still, the lower results obtained for the EOS can be explained by it having a wider field of view as per figure 3.3, which can make the object even smaller.

The Full models performed worse in the lantern scenario than in previous cases. Models trained with synthetic images have seen different types of illumination during training, including light color, position, and intensity variations. Therefore, these models can handle this challenge better, even if such light variations did not exist in the real dataset, representing a step towards overcoming the reality gap. The Fine strategy seems better than the Mix, as realistic light information is needed to tackle this challenge. It might also be promising to improve the data augmentation on real frames by changing the saturation and brightness. This step might increase the real-world usability of the synthetic light variation, as it simulates photorealistic illumination changes in the scene.

The dark is the most challenging case and got the worst results overall. Finding any characteristic in the frame in this scenario, including geometry details and color, is difficult. Even the lighting variations in the randomized rendering were insufficient to deal with this case. This scenario is one of the hardest unsolved challenges in the scope of monocular RGB-only methods. Only Mix100 trained for the S8 got usable results. It might be partly due to automatic processing done by the S8's sensor, which helped it work in low-light conditions, coupled with the network being trained in this same camera domain. Such frame preprocessing strategies might be the only way for monocular RGB approaches to handle challenges like this, apart from estimating depth out of RGB images.

8.1.2 Training strategies

Our tests yielded some expected conclusions: training a model exclusively with labeled real images that accurately represent the scene, illumination, and challenges is the optimal choice. Similarly, using training data from different cameras enhances performance when a model needs to work with multiple cameras. Even for a model targeted at a single camera, training with data from other cameras can be beneficial if that data covers the relevant scene challenges.

However, this strategy reveals significant flaws when the target distribution changes. As seen in Tables 6 and 7, even with all available real training data (i.e., FullReal models), accuracy decreases when the camera or environmental conditions change. Synthetic data is advantageous for applications where the entire target distribution is unknown or varies frequently. Mix and Fine models outperform FullReal models when varying camera parameters and introducing environmental challenges.

Mixing synthetic and real data during training generally improves test performance, even if just slightly. Models trained with a few real images (10% and 50%) can still achieve accuracy similar to their FullReal counterparts. Results show that it is possible to train models more efficiently using less real data without losing accuracy. Although precision gains may be insignificant, especially for more complex challenges, the results are promising given the difficulty of acquiring large amounts of accurate, real, domain-specific labeled data. The mixed training process is complex, requiring the model to learn the characteristics of different distributions simultaneously. However, this approach reduces sensitivity to overfitting environmental parameters, suggesting that synthetic data acts as a regularizer.

For Fine models, trained on synthetic data and fine-tuned with real data, the best variations used 50% of the real dataset. This shows that more real data is not always better. The two-step training process allows the model to first learn environmental challenges and object characteristics, then fine-tune with real data. Too much fine-tuning leads to overfitting and a loss of generalization. Fine-tuned models are the best approach for handling camera changes but struggle more than Mix models to retain synthetic variations needed for other challenges.

These results, especially when comparing the Fine and Mix models, explain how crucial model initialization is to the training process, depending on the generalization behavior desired. Regardless of the approach chosen, the FullReal strategy is the worst unless there is specific training data for a particular target, with little to no variation. As we know, the reality gap can impact overall performance, making it necessary to use some real data, even if the amount is reduced. Even with randomization, which creates significant variation, estimating initial parameters coherent with our distribution is still necessary. Therefore, there is a strong argument for using synthetic data for monocular RGB 6DoF detection, especially for our indoor scenario using handheld objects and cameras. Thus, we can improve these strategies by leveraging knowledge extracted from a small amount of real samples and combining it with domain randomization. Figure 8.1.2 shows examples of how each training strategy handles the variations tested. As described, FullReal models can deal with tests in the same domain, Fine models handle changes when the object's appearance is preserved, and Mixed models manage drastic visual changes in objects.

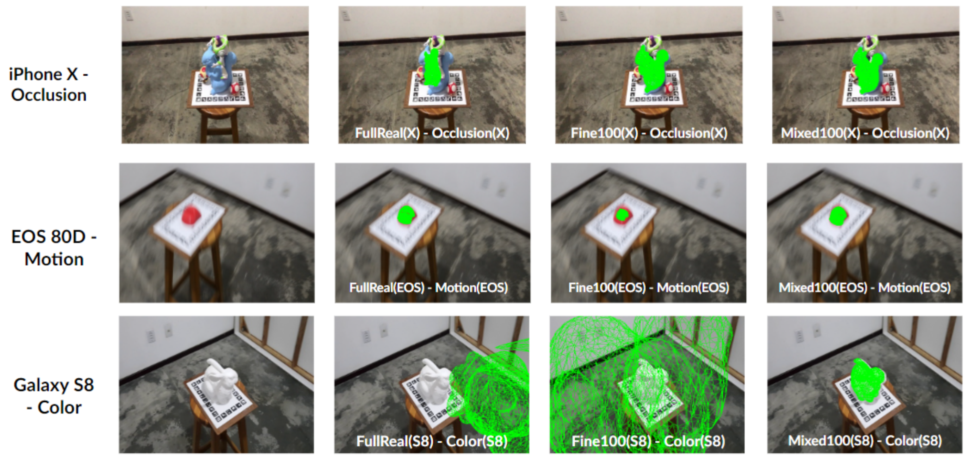
iPhone X - Occlusion
FullReal(X) - Occlusion(X)
Fine100(X) - Occlusion(X)
Mixed100(X) - Occlusion(X)
EOS 80D - Motion
FullReal(EOS) - Motion(EOS)
Fine100(EOS) - Motion(EOS)
Mixed100(EOS) - Motion(EOS)
Galaxy S8 - Color
FullReal(S8) - Color(S8)
Fine100(S8) - Color(S8)
Mixed100(S8) - Color(S8)

Figure 25 – Example of results obtained for each training strategy on the domain variations proposed in C3PO. From top to bottom, it shows images taken from X, EOS, and S8 in the Occlusion, Motion, and Color variation, respectively. From left to right, we can observe the results from FullReal, Fine, and Mixed models tested on the same camera where the model was trained.

Source: CUNHA et al. (2024)

9 RAND-NERF: ADVANCING 6DOF POSE ESTIMATION USING RANDOMIZATION AND NERF AUGMENTATION

9.0.1 Motivation

NeRF (MILDENHALL et al., 2021) was initially proposed to reconstruct a viewed scene from a set of sample images with known poses. Utilizing an MLP model, we can encode a frame that represents the scene based on each of its 3D coordinates, along with the corresponding view direction, and map this information to color and density estimations at those positions. NeRF encodes a scene as a continuous representation in the MLP (F_Θ). Given a 3D point in space (x) and its view direction (d) as input, it predicts the RGB color (c) and the volume density (δ); $F_\Theta(x, d) \mapsto (c, \delta)$. As it was designed for model-free scene reconstruction, the model exhibited shortcomings in reconstructing certain aspects of real scenes. Subsequent models have enhanced the rendering process by addressing various characteristics such as reflections, ambient illumination, render performance, and fine-detail reconstruction.

RefNeRF (VERBIN et al., 2022) expanded upon this idea by introducing a new pipeline with an architecture capable of predicting interpretable components to describe objects in the rendering process. Its architecture decouples scene properties by predicting normals ($\hat{n}$), roughness (ρ), specular tint (s), specular color (c_s), and diffuse color (c_d). The final object components are further estimated by incorporating the predicted data into the tone-mapping process. Color is defined as specified in equation 9.1.

$$c = \gamma(c_d + s \odot c_s) \tag{9.1}$$

Building upon the concept of RefNeRF, we propose using a NeRF architecture capable of predicting interpretable components that describe object properties for physical-based rendering. Predicting these components meaningfully enables us to adjust the predicted parameters, creating a domain-randomized representation of the original scene.

9.0.2 NeRF Randomization

The proposed architecture is illustrated in Figure 26. Initially, we train the NeRF model following the RefNeRF pipeline, retaining features corresponding to the physically based rendering. The training of the reconstruction model (i.e., NeRF) involves a two-step process.

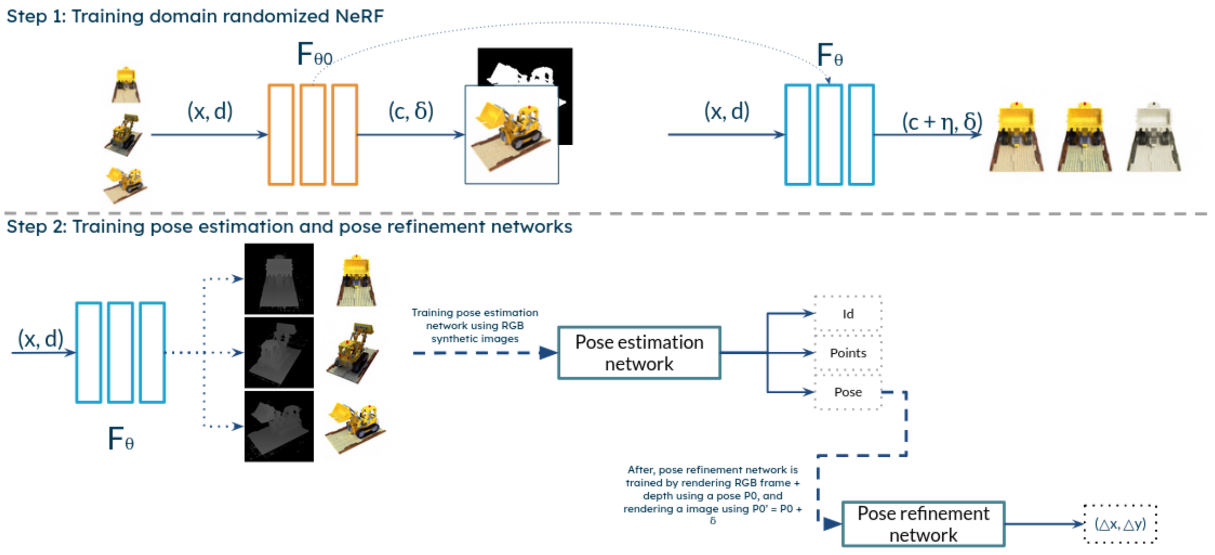
Step 1: Training domain randomized NeRF
(x, d)
Fθ0
(c, δ)
Fθ
(x, d)
(c + η, δ)
Step 2: Training pose estimation and pose refinement networks
(x, d)
Fθ
Training pose estimation network using RGB synthetic images
Pose estimation network
Id
Points
Pose
After, pose refinement network is trained by rendering RGB frame + depth using a pose P0, and rendering a image using P0' = P0 + δ
Pose refinement network
(Δx, Δy)

Figure 26 – Training structure combining domain randomization in NeRF optimization. In step 1, we conduct multi-stage training using NeRF to optimize object reconstruction while introducing noise to the predicted components to simulate feature variations for randomization during rendering. In step 2, the optimized model is used to render images to train pose regression and refinement networks. Both models are employed during the inference phase to recover object pose without relying on prior geometry, such as CAD models or additional real images.

Source: CUNHA et al. (2024)

Initially, we guide the NeRF to learn genuine object features without altering visual aspects, employing the L1-norm to compute loss between real input (I_r) and synthetic image (I_s). Subsequently, we fine-tune the model parameters, prioritizing the optimization of high-level details, utilizing the contextual loss function 9.2. VGG features (Ψ) are applied in the loss (L_c) to extract information from both the rendered and target images, trying to prioritize a cross-domain alignment between the perceived features. Figure 27 shows the results obtained after the object is learned by the NeRF model (before adjusting object borders). Although the DRRender image contains the best image quality, it cannot reproduce the noises and distortions observed in the real image. NeRF images aim to render a photorealistic representation while preserving some aspects of the real camera capture.

$$L_c = |\Psi(I_r) - \Psi(I_s)| \tag{9.2}$$

In the second step, we adjust the values of the component representing the object's color and illumination, introducing bias to the predicted diffusion and specular components (ϕ_d, ϕ_s) sampled from a normal distribution $(\mathcal{N}(0,1))$. Color is computed as equation 9.1 after updating the parameters (c_d) and (c_s) according equation 9.3. This forces the model to reconstruct fine

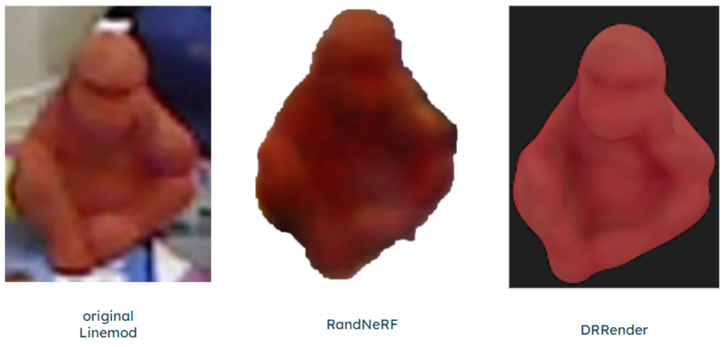

Figure 27 – Visual result of the reconstruction obtained by rendering with NeRF model and comparing with the synthetic rendered image from DRrender. From left to right, it is possible to observe the real image, NeRF image, and DRRender image, respectively.

Source: CUNHA et al. (2024)

details while accommodating variations in color representation without penalizing optimization. In the context of domain randomization, our goal is to approximate object geometry, and fidelity in mapping the object's real features is not strictly necessary. We utilize the object mask in the dataset annotations to filter information during model training. The NeRF model outputs both the RGB frame and a disparity map, which can be converted into the object depth map. The coordinates in the final representation are then normalized. This process is illustrated in Figure 28. The model is adjusted to create RandNeRF by altering the predicted information using RGB frames in the object area. Since we have the binary segmentation mask of the object in the scene, we can change the object's appearance without altering the background.

$$c'_d = c_d + \phi_d; c'_s = c_s + \phi_s \tag{9.3}$$

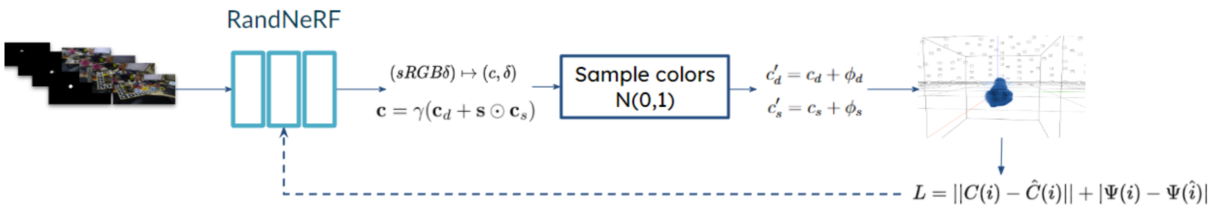
RandNeRF
$(sRGB\delta) \mapsto (c, \delta)$
$\mathbf{c} = \gamma(\mathbf{c}_d + \mathbf{s} \odot \mathbf{c}_s)$
Sample colors
N(0,1)
$c'_d = c_d + \phi_d$
$c'_s = c_s + \phi_s$
$L = ||C(i) - \hat{C}(i)|| + |\Psi(i) - \Psi(\hat{i})|$

Figure 28 – To induce randomization aspects in the NeRF representation, we adjusted the model by changing the colors of the object during optimization, creating the RandNeRF model. This was done using a binary segmentation mask to filter the relevant area representing the object's pixels.

Source: CUNHA et al. (2024)

To enhance the robustness against occlusion, we introduce distractor objects into the rendered frames. Inspired by Tremblay et al. (TREMBLAY et al., 2018) and our DR-Render (CUNHA et al., 2022), we incorporate 3D distractor objects post-image rendering. This addition of 3D object distractors maps contextual characteristics responsible for occlusion, seamlessly integrating simulated information into the same domain. Following the suggestion by Hintertoisser et al. (HINTERSTOISSER et al., 2019), incorporating 3D correspondent data proves advantageous, improving the models' capacity to learn occlusion characteristics in real scenes and outperforming the use of random 2D shapes or random images as occluders. To refine the interaction between the target object and distractor objects, we leverage the generated masks from both, applying a Gaussian filter to the intersecting regions. This approach is intended to simulate real camera artifacts that emerge when a surface transition occurs for close objects. The occlusion is added dynamically during training after the NeRF models generate randomized images. The images are generated to ensure a maximum of 35% object occlusion. Finally, we simulate motion blur, convolving simple directional filters with the final image. Figure 29 illustrates the final result after processing the rendered image.

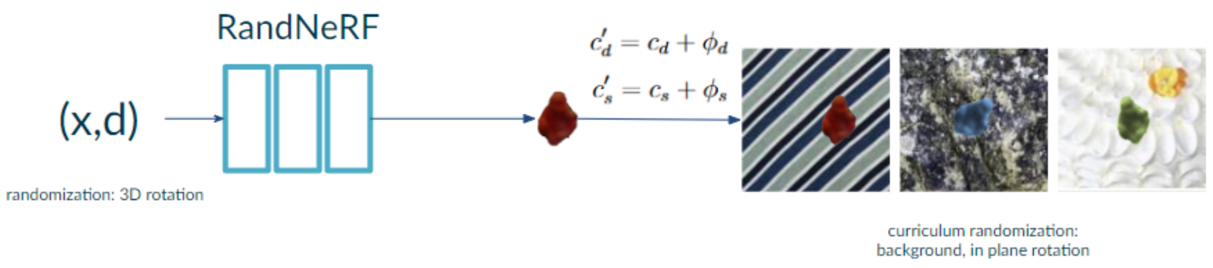
RandNeRF
(x,d)
randomization: 3D rotation
$c'_d = c_d + \phi_d$
$c'_s = c_s + \phi_s$
curriculum randomization:
background, in plane rotation

Figure 29 – RandNeRF randomization result after image processing applying in-plane rotations and random background.

Source: CUNHA et al. (2024)

9.0.3 Training models with randomization

In the second step, we utilize the rendered image to train a coarse pose estimator. Using our detector described in chapter 5, we expand a CNN to perform regression on the image points representing the projection of the object's 3D bounding box. We can train the detector using randomized synthetic frames by employing NeRF to render the image at each step, passing the point location and view direction as input (Figure 26). We train the network by optimizing the L_{total} loss defined in equation 5.2. During inference, to estimate the relative pose (i.e., extrinsic matrix), we leverage the camera intrinsic parameters and coefficients using the Perspective-n-Point (PnP) algorithm.

Following the initial pose estimation, pose refinement is based on the displacement between two consecutive images. Using the randomization process with NeRFs, we dynamically render various viewpoints. In each iteration, we input both an initial pose and a perturbed pose, representing a variation of the initial pose in the NeRF model. Subsequently, we render images from both inputs, compute the pose difference, and convert it into image coordinates (Δx, Δy). This information serves as the basis for initializing the training of the pose refinement network. Figure 30 shows the pose refinement architecture. The model contains two branches: In the first branch, we train feature extraction to capture data from two different domains (e.g., real and synthetic), creating a correlation volume that is processed to generate a feature block of correlated pairs. The second branch is responsible for extracting context information from the domain we want to predict (the real one) and is composed of frozen weights. The correlated pairs from the real and synthetic domains are concatenated with the contextual feature map. Finally, we compute the displacements from the returned activations and optimize them using the GRU layer, as done by the RAFT model (TEED; DENG, 2020). Throughout the process, we aim to minimize the displacement between the two input frames, utilizing the pseudo-depth estimated by NeRF as a supervised signal during training. We calculate the displacement between the two RGB frames during the inference step.

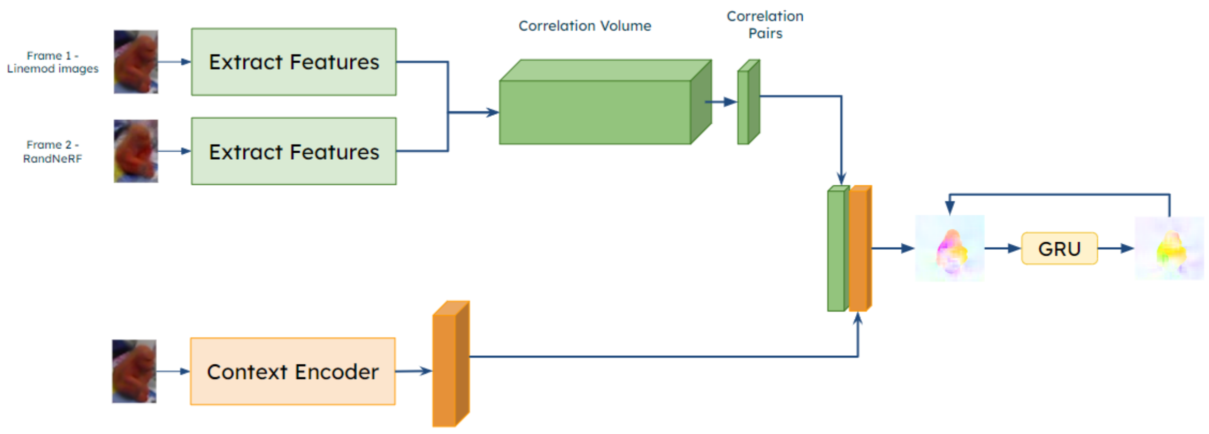

Figure 30 – Pose refinement architecture.

Source: CUNHA et al. (2024)

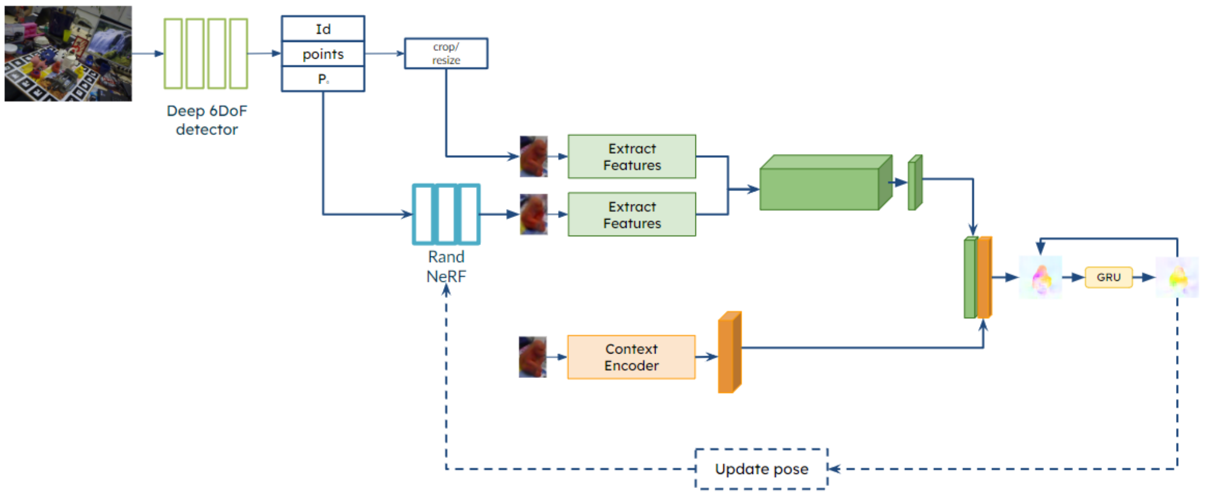

Figure 31 – Inference step: We provide the input RGB image to the pose estimation network, which predicts an initial pose. We then utilize the NeRF model to generate a new image (based on the predicted pose) and feed both images to the pose refinement network. The estimated difference between the real and synthetic image is then employed to update the predicted pose.

Source: CUNHA et al. (2024)

9.0.4 6DoF pose estimation and tracking

In the inference phase (Figure 31), we rely on the representation previously learned during NeRF training rather than depending on CAD models. Our coarse pose estimator estimates the initial pose. We utilize this prediction to generate an image using the NeRF model. Subsequently, both the area corresponding to the initial prediction and the rendered image are fed into our pose refinement network. The coarse pose generates a template image rendered by our NeRF model. We then compare the rendered result with the original image to detect any displacement between the expected pose (input image) and the estimated pose (rendered image). The pose refinement network, taking the input and the rendered image, predicts this displacement between our initial estimation. The resulting displacement (x, y in the image plane) is applied to refine our initial pose predicted by the CNN detector.

9.1 IMPLEMENTATION DETAILS

9.1.1 Randomization and training

The desktop employed in this work was a quad-core CPU running at 3.60 GHz, 32 GB of RAM, and an NVIDIA GeForce RTX 3060 GPU 12GB VRAM. NeRF training requires two days for each object in our setup. To minimize the use of real labeled frames, instead of using the entire training set, we randomly select a subset consisting of 10% of the available samples. To ensure representative viewpoints around the object, we define a distance between consecutive selections, which is set at 5% of the total number of frames. We use the RGB frame and the object mask (e.g., depth annotation) during training, along with the pose annotation available in the dataset (rotation matrix and translation vector) to determine the point location and viewing direction. Additionally, the space where we project the rays is constrained by the object's position, defining the near and far parameters for NeRF training based on the object's center and diameter. Following NeRF training, we generated a randomized dataset by manipulating the diffusion and specular components predicted by the model. For each component, we introduced noise sampled from a uniform distribution. We rendered $10,000$ images for each object, encompassing diverse viewpoints, colors, and distractor objects. The distractors consisted of basic geometric shapes randomly placed in the scene, complemented by contextual information from other learned objects.

For both the pose estimator and pose refinement networks, we conducted training for 500 epochs using an ADAM optimizer with a learning rate of $\alpha = 0.001$ and applied cosine annealing as the scheduler. A warm-up period of 20 epochs was adopted in the pose estimation phase. For each epoch, we introduced randomization by augmenting images with random backgrounds. As mentioned earlier, this augmented background served the purpose of preventing overfitting. Additionally, to enhance robustness against scale changes, we keep the uniform sampled scales in width and height by a random factor ranging from 320 to 680, with corresponding adjustments to camera parameters. During pose refinement, the CRAFT (SUI et al., 2022) served as the base network for calculating image differences. The accuracy is measured by the pose accuracy (ADD) or its variation for symmetric objects (ADD-S) (HINTERSTOISSER et al., 2013). The ADD is defined by equation 5.4. Following the previous experiments, a pose is considered correct if $E < .10d$, where d is the maximum distance between two mesh vertices, representing the CAD diameter.

9.1.2 Datasets

To validate the approach, we utilized the linemod (HINTERSTOISSER et al., 2013) and linemod-occlusion (BRACHMANN et al., 2014) datasets (Figure 9.1.2), which are widely applied in the literature for assessing methods in the 6DoF pose estimation task. The linemod dataset encompasses variations in object pose, featuring the object centrally positioned in the scene against a cluttered background. Linemod-occlusion, an extension of linemod, includes the same scenes annotated with different objects, introducing variations in object position and causing occlusion depending on the viewpoint.

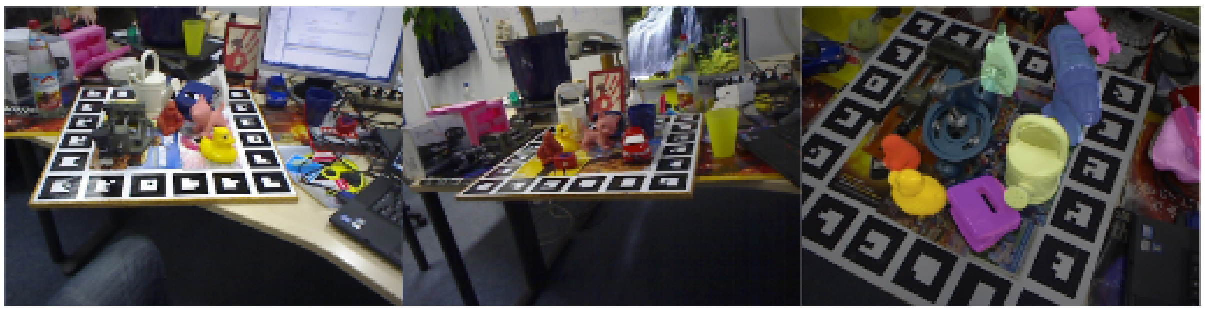

Figure 32 – Linemod and Linemod-occlusion datasets

Source: HINTERSTOISSER et al. (2013), BRACHMANN et al. (2014)

In addition, we conducted evaluations on the HomebrewedDB dataset (KASKMAN et al., 2019) (Figure 9.1.2), which comprises diverse scenes featuring the same objects with variations in ambient light. This evaluation aims to validate the robustness of the randomized model in handling generalization under different light conditions.

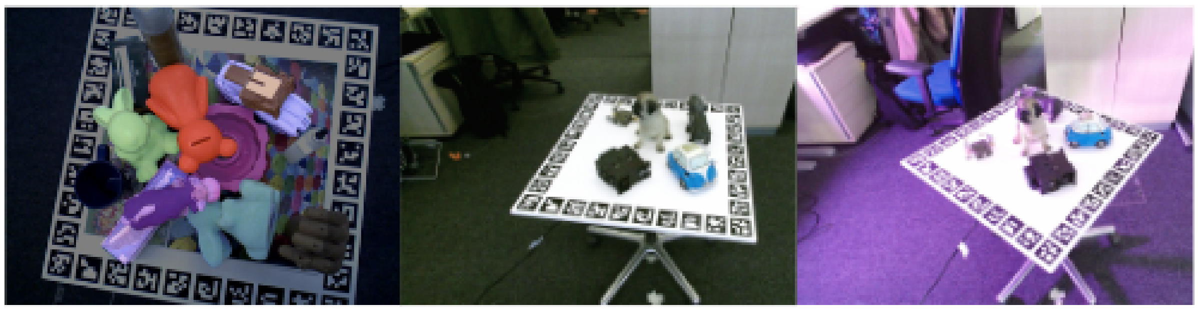

Figure 33 – HomebrewedDB dataset

Source: KASKMAN et al. (2019)

Finally, we applied the C3PO dataset (CUNHA et al., 2022) (Figure 9.1.2), consisting of small 3D-printed objects recorded in front of various challenges presented in specific scenes, such as dark ambient lighting, fast motion, and occlusion as discussed in chapter 4.

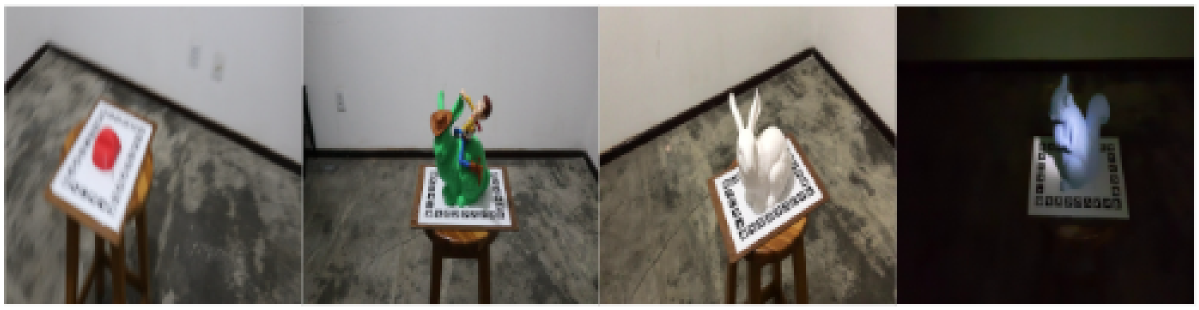

Figure 34 – C3PO dataset

Source: CUNHA et al. (2022)

10 EVALUATING RAND-NERF: RESULTS AND DISCUSSIONS

10.1 6DOF ESTIMATION ON LINEMOD

We validated the model's accuracy under straightforward scenarios in the initial evaluation by comparing our approach with state-of-the-art models on the linemod dataset; results are shown in Table 8.

Table 8 – Results obtained on the Linemod dataset using the ADD-S metric. Bold font indicates the best category result, while underlined values are the best between methods without CAD models. We compare Rand-NeRF performance against DPOP (ZAKHAROV; SHUGUROV; ILIC, 2019), PVNet (PENG et al., 2019), LieNet (DO et al., 2019), and NerfPose (LI et al., 2023).

Object	DPOD	PVNet	LieNet	Nerf-Pose	Rand-NeRF (Ours)
CAD	yes	yes	no	no	no
Ape	53.3	43.6	38.8	<u>**93.1**</u>	90.1
Benchvise	95.2	**99.9**	71.2	<u>99.6</u>	98.9
Cam	90.0	86.9	52.5	<u>**98.9**</u>	97.2
Can	94.1	95.5	86.1	<u>**99.7**</u>	97.3
Cat	60.4	79.3	66.2	<u>**98.1**</u>	87.6
Drill	97.4	96.4	82.3	<u>**98.7**</u>	94.3
Duck	66.0	52.6	32.5	<u>**94.2**</u>	89.8
Eggbox	99.6	99.2	79.4	<u>**99.9**</u>	96.6
Glue	93.8	95.7	63.7	99.3	<u>**99.6**</u>
Hole puncher	64.9	81.9	56.4	<u>**96.5**</u>	94.2
Iron	**99.8**	98.9	65.1	97.8	<u>99.4</u>
Lamp	88.1	**99.3**	89.4	<u>98.7</u>	91.5
Phone	71.4	92.4	65.0	<u>**97.3**</u>	92.4
Average	82.6	86.3	65.27	**97.83**	94.53

Following the approach outlined in NerfPose proposed by Li et al. (LI et al., 2023), we categorized the results into CAD-based and model-free methods (e.g., without using CAD). Model-free methods pose a more significant challenge as there is no prior information about the object geometry, making training and evaluation more demanding. As depicted in Table 8, our model demonstrates accuracy comparable to other models in the dataset. Notably, in the model-free category, NerfPose achieved the best overall results in pose estimation. Given linemod's limited variation in scene conditions, our randomization approach is less effective in this scenario. Furthermore, CAD-based approaches are not able to outperform model-free methods. Thus, considering that CAD information is not always available, our approach stands out by achieving results that are close to the best, even without CAD data. It is important to highlight that although the original training set contains approximately 1000 images, we use only 100 real images in the training process. Despite this significant reduction, we achieve comparable results, thereby reducing the need for a large set of real labeled images.

10.2 OCCLUSION HANDLING ON LINEMOD-OCCLUSION

Table 9 compares the mean ADD-S in Linemod (LM) and Linemod occlusion (LM-OCC). Comparing the results on LM-OCC, our model-free approach and NerfPose exhibit a less pronounced drop in accuracy. Both approaches employ specific steps to address occlusion during training. Therefore, our context-based occlusion approach has proven to be the most effective in handling these scenarios. CAD-based approaches face more challenges in occlusion, as they need to match the model data with what is visualized in the image.

Table 9 – Mean ADD-S value obtained on Linemod (LM) and Linemod-Occlusion(LM-OCC). We compare Rand-NeRF with PoseCNN (XIANG et al., 2017), PVNet (PENG et al., 2019), and NerfPose (LI et al., 2023)

	PoseCNN	PVNet	NerfPose	Rand-NeRF (Ours)
CAD	yes	yes	no	no
Mean LM	62.7	86.3	**97.8**	94.5
Mean LM-OCC	24.9	40.8	51.4	**58.2**

10.3 DOMAIN VARIATION ON HOMEBREWEDDB

In this experiment, we follow the setup outlined by Zakharov et al. (ZAKHAROV et al., 2022). All methods are trained on HomebrewedDB-scene5 and evaluated on the test set of scene5 and scene10. The primary objective is to assess the impact of illumination variation while keeping the objects in the scene consistent. In this evaluation, we categorize the approaches into CAD-based, utilizing only synthetic data generated with a CAD model; CAD-based+real, incorporating supplementary real data; and modeless, which does not rely on any information from CAD models during training. As shown in table 10, our approach performs comparably with state-of-the-art models in simple pose estimation (e.g., same scene). The highest accuracy is achieved by the Pix2PixHD method (WANG et al., 2018), which relies on both real and CAD data for training. However, as we alter the scene configuration by evaluating the model under different lighting conditions (scene 10), our approach can maintain accuracy with a slight loss and outperform all other methods. We train the model to handle illumination changes while varying the predicted color during training. This enables us to retain learned information from real images, adjusting its parameters when presenting these images to the pose estimation model.

Table 10 – Results obtained on the HB dataset using the mean ADD-S, training on scene 5 and evaluating on scene 5 and 10 (varying illumination). Bold font indicates the best category resultt.

Model	train	scene-5	scene-10
RenderNet (ZAKHAROV et al., 2022)	CAD	88.2	40.4
Pix2Pix (ISOLA et al., 2017)	CAD+real	73.5	29.1
Pix2PixHD (WANG et al., 2018)	CAD+real	**89.6**	32.3
Rand-NeRF (Our)	CAD-free	83.1	**75.1**

10.4 SCENE CHALLENGES ON C3PO DATASET

This test evaluates our approach in the C3PO dataset (CUNHA et al., 2022). This dataset was developed to address variations in scenes, objects, and devices, containing frames captured under diverse challenges such as fast motion, occlusion, object color variation, and light changes. A set of images gathered from various devices is also available. Still, since our work does not aim to address cross-device robustness at this stage, we utilized a specific set from a single device, varying scene and object characteristics.

We believe that using the rendering encoded in the model structure, rather than explicitly modeling the object, has additional benefits. NeRF can capture details of the real object that are difficult to model, simplifying the process by requiring only a few images to approximate the object structure. This approach allows us to introduce variations based on an initial estimation of the real features. Furthermore, achieving photorealism and defining variations through explicit modeling is a complex task. We compare the results with the approach of our previous model (CUNHA et al., 2022) (DR-Render), which is a CAD-based domain randomization using traditional rendering engines. Table 11 illustrates that our approach outperforms DR-Render in all settings, effectively handling simple scenarios, fast motion, and occlusion. DR-Render, being based on rendering engines, faces the reality gap problem. Randomization is applied only to synthetic frames with a higher difference from real ones. Still, it requires a considerable amount of real data for further optimization, potentially diminishing the impact of randomization on the model's weights. Our approach uses synthetic data, incorporating only a few real images mapped into the NeRF model, with randomization directly applied to the reconstructed image. Although it may not be a perfect representation, the difference is less pronounced, as we are only varying a previously learned set of parameters instead of initializing scenes with random values for light, colors, and materials. Figure 35 and Figure 36, we can see visual results projecting the pose predicted by the RandNeRF pipeline for each object and two different challenges in the C3PO dataset.

Table 11 – Mean ADD-S results obtained on the C3PO dataset under different scene conditions. Bold font indicates the best category result.

Model	Fast motion	Occlusion	Object color	Lantern
DR-Render (CUNHA et al., 2022)	50.6	31.7	18.6	06.9
Rand-NeRF	**75.1**	**64.7**	**50.2**	**32.8**

Moreover, despite the decrease in the score, our technique can handle object color variation and environments with dark illumination (e.g., lantern pointing towards the object).

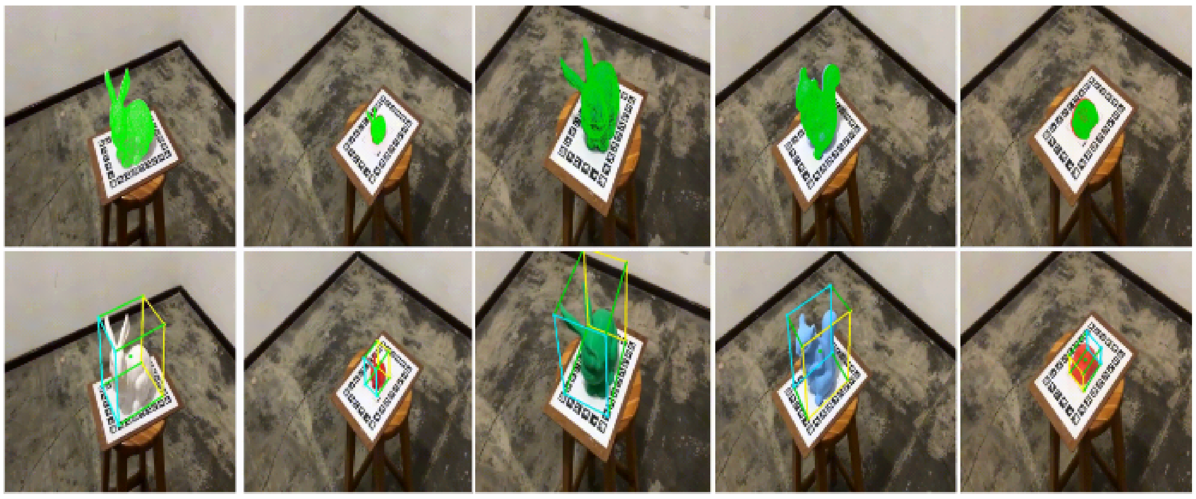

Figure 35 – RandNeRF visual result for each object in C3PO dataset.

Source: CUNHA et al. (2024)

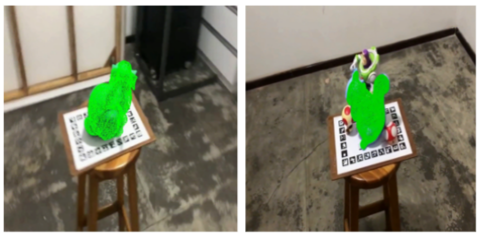

Figure 36 – RandNeRF visual result for the challenges motion and occlusion in C3PO dataset.

Source: CUNHA et al. (2024)

10.5 LIMITATIONS

As seen from section 10.1, randomization proves less effective in scenarios where variations are not pronounced. Although the method successfully detects the object, randomization unnecessarily escalates training complexity in scenarios that are not beneficial. Rendering randomized frames demands considerable computational effort, requiring the creation of random scenarios with maximal variation, even if such scenarios are not realistic. This behavior, previously observed in the DR-Render approach, remains consistent in the Rand-NeRF scenario.

Additionally, the pipeline is highly dependent on the detection approach. When the initial pose estimation is sufficiently close, we can accurately recover the pose by adjusting pixel displacements. However, with a disparate estimation, achieving the expected result becomes challenging. Similarly, while the method can match poses in moderate movement, larger movements necessitate more iterations to align the object in the scene. Also, the NeRF representation is not always accurate and may introduce slight variations in the true pose. Since randomization is not focused on representation fidelity, it becomes essential when estimating the pose and can impact the overall model's performance. Furthermore, rendering images from NeRFs at each step contributes to increased inference time.

10.6 RUNNING TIME

Using a 448x448 input image with an RTX 3060, the inference process takes only 0.06 seconds (0.02 seconds for the detection phase and 0.04 seconds for pose refinement). Rendering the image at each step takes 0.2 seconds, resulting in a total time of approximately 0.26 seconds.

11 CONCLUSIONS

This work explored various aspects of training a monocular RGB-based deep learning model for 6DoF pose estimation. By examining multiple scenarios of domain variation, we highlighted important considerations for assessing new methods within this scope. Our findings confirm that domain variations can significantly impact model predictions. While traditional augmentations can address simple variations, they often fail to generalize to more severe domain changes. Specifically, when environmental settings and object appearances are altered, a more sophisticated approach that accounts for the 3D properties of the scene is required. Additionally, we emphasized the importance of considering camera settings, as device-specific characteristics can affect generalization when only RGB data is available. Increasing the training set can mitigate this issue, but it is essential to recognize that sufficient labeled samples may not always be readily available.

Furthermore, we have proposed a practical approach, the DR-Render method, which evaluates the impact of different training strategies on the performance of a 6DoF pose estimator. This method incorporates randomization concepts extended to 3D rendering for pose estimation. While the models trained using this approach still require real data, making them unsuitable for end-user applications in a broad sense, we have also suggested strategies to mitigate this issue by reducing the number of labeled images needed. We have also assessed which training strategies are most effective for under-explored scenarios involving variations in the environment, target object, and capturing device.

Moreover, we have introduced a unique resource in the form of a labeled dataset, C3PO, which contains both real and synthetic images. This dataset is distinct in its comprehensive coverage of essential aspects of pose estimation evaluation, including scenes designed to pose specific challenges and replicated using a variety of devices. To our knowledge, no other public dataset offers this combination of cross-device samples structured by challenges derived from objects and environments.

The suggestions in this work are particularly relevant for 3D-printed, low-texture objects, for which we believe our results to be even more directly applicable. For future work, we must perform a careful ablation study to identify the role of each domain randomization aspect when rendering synthetic scenes and how other synthetic generation approaches can impact 6DoF detection. It would also be interesting to evaluate how real data augmentation approaches individually influence these points.

Additionally, we have extended our model by proposing a new pipeline based on NeRF and domain randomization. NeRF has proven to be an impressive approach for the realistic reconstruction of viewed frames. Domain randomization, on the other hand, allows us to simulate scene variations without being constrained by realistic configurations. By combining NeRF's capability to reconstruct scenes with real information and enhancing its robustness to scene variations through randomization strategies, we have achieved a powerful synergy that holds significant potential for improving 6DoF pose estimation.

We simulate random light variations by altering the estimated color and process images through augmentations and random occlusions. This approach demonstrates its utility in training 6DoF pose estimation models without the need for large datasets. Moreover, we showcase how randomization improves model generalization in the face of scene variations such as light changes, fast motion, and occlusions. Acknowledging that this work presents diverse challenges, there is ample room for improvements in the method's pipeline.

Furthermore, we can extend the approach by incorporating additional randomization steps, conducting a comprehensive analysis through an ablation study, and reducing dependence on the detector model.

11.0.1 Contributions

The direct and indirect contributions of this work are listed below.

- DA CUNHA, KELVIN B.; BRITO, CAIO ; VALENÇA, LUCAS ; FIGUEIREDO, LUCAS ; SIMÕES, FRANCISCO ; TEICHRIEB, VERONICA . The impact of domain randomization on cross-device monocular deep 6DoF detection. PATTERN RECOGNITION LETTERS, v. 159, p. 224-231, 2022.
- DA CUNHA, KELVIN B.; BRITO, CAIO ; VALENCA, LUCAS ; SIMOES, FRANCISCO ; TEICHRIEB, VERONICA . A Study on the Impact of Domain Randomization for Monocular Deep 6DoF Pose Estimation. In: 2020 33rd SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI), 2020, Recife/Porto de Galinhas. 2020 33rd SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI), 2020. p. 332.
- Da Cunha, Kelvin; Uchiyama, Hideaki; Simões, Francisco; Teichrieb, Veronica. AugNerf:

Advancing 6DoF object pose estimation via NeRF and domain randomization. (SUBMITTED)

- FELIX, HEITOR ; SIMÕES, FRANCISCO ; CUNHA, KELVIN ; TEICHRIEB, VERONICA . Image Processing Techniques to Improve Deep 6DoF Detection in RGB Images. In: XXI Symposium on Virtual and Augmented Reality, 2019. Anais Estendidos do Simpósio de Realidade Virtual e Aumentada (SVR). p. 19.
- Norberto, J. M., Cunha, K. B. D., Silva Júnior, M. R. D., Simões, F. P. M. (2023, November). Automating the audit of the Brazilian electronic ballot operation: A new dataset for 6DoF pose estimation of the voter terminal based on domain randomization. In Proceedings of the 25th Symposium on Virtual and Augmented Reality (pp. 56-65).

REFERENCES

AKKALADEVI, S.; ANKERL, M.; HEINDL, C.; PICHLER, A. Tracking multiple rigid symmetric and non-symmetric objects in real-time using depth data. In: IEEE. *2016 IEEE International Conference on Robotics and Automation (ICRA)*. [S.l.], 2016. p. 5644–5649.

ALGHONAIM, R.; JOHNS, E. Benchmarking domain randomisation for visual sim-to-real transfer. In: IEEE. *2021 IEEE International Conference on Robotics and Automation (ICRA)*. [S.l.], 2021. p. 12802–12808.

BARRON, J. T.; MILDENHALL, B.; VERBIN, D.; SRINIVASAN, P. P.; HEDMAN, P. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2022. p. 5470–5479.

BIAN, W.; WANG, Z.; LI, K.; BIAN, J.-W.; PRISACARIU, V. A. Nope-nerf: Optimising neural radiance field with no pose prior. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2023. p. 4160–4169.

BOSS, M.; BRAUN, R.; JAMPANI, V.; BARRON, J. T.; LIU, C.; LENSCH, H. Nerd: Neural reflectance decomposition from image collections. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. [S.l.: s.n.], 2021. p. 12684–12694.

BRACHMANN, E.; KRULL, A.; MICHEL, F.; GUMHOLD, S.; SHOTTON, J.; ROTHER, C. Learning 6d object pose estimation using 3d object coordinates. In: SPRINGER. *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part II 13*. [S.l.], 2014. p. 536–551.

CALLI, B.; SINGH, A.; WALSMAN, A.; SRINIVASA, S.; ABBEEL, P.; DOLLAR, A. M. The ycb object and model set: Towards common benchmarks for manipulation research. In: *2015 International Conference on Advanced Robotics (ICAR)*. [S.l.: s.n.], 2015. p. 510–517.

CAO, J.; LI, Z.; WANG, N.; MA, C. Lightning nerf: Efficient hybrid scene representation for autonomous driving. *arXiv preprint arXiv:2403.05907*, 2024.

CHEN, A.; XU, Z.; GEIGER, A.; YU, J.; SU, H. Tensorf: Tensorial radiance fields. In: SPRINGER. *European Conference on Computer Vision*. [S.l.], 2022. p. 333–350.

CLEAC'H, S. L.; YU, H.-X.; GUO, M.; HOWELL, T.; GAO, R.; WU, J.; MANCHESTER, Z.; SCHWAGER, M. Differentiable physics simulation of dynamics-augmented neural objects. *IEEE Robotics and Automation Letters*, IEEE, v. 8, n. 5, p. 2780–2787, 2023.

CSURKA, G. Domain adaptation for visual applications: A comprehensive survey. *arXiv preprint arXiv:1702.05374*, 2017.

CUNHA, K. B. da; BRITO, C.; VALENÇA, L.; SIMÕES, F.; TEICHRIEB, V. A study on the impact of domain randomization for monocular deep 6dof pose estimation. In: IEEE. *2020 33rd SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*. [S.l.], 2020. p. 332–339.

CUNHA, K. B. da; BRITO, C.; VALENÇA, L.; SIMÕES, F.; TEICHRIEB, V. A study on the impact of domain randomization for monocular deep 6dof pose estimation. In: IEEE. *2020 33rd SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*. [S.l.], 2020. p. 332–339.

CUNHA, K. B. da; BRITO, C.; VALENÇA, L.; FIGUEIREDO, L.; SIMÕES, F.; TEICHRIEB, V. The impact of domain randomization on cross-device monocular deep 6dof detection. *Pattern Recogn. Lett.*, Elsevier Science Inc., USA, v. 159, n. C, p. 224–231, jul 2022. ISSN 0167-8655.

DENG, J.; DONG, W.; SOCHER, R.; LI, L.-J.; LI, K.; FEI-FEI, L. Imagenet: A large-scale hierarchical image database. In: IEEE. *2009 IEEE conference on computer vision and pattern recognition*. [S.l.], 2009. p. 248–255.

DENG, K.; LIU, A.; ZHU, J.-Y.; RAMANAN, D. Depth-supervised nerf: Fewer views and faster training for free. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2022. p. 12882–12891.

DENNINGER, M.; SUNDERMEYER, M.; WINKELBAUER, D.; OLEFIR, D.; HODAN, T.; ZIDAN, Y.; ELBADRAWY, M.; KNAUER, M.; KATAM, H.; LODHI, A. Blenderproc: Reducing the reality gap with photorealistic rendering. In: *International Conference on Robotics: Sciene and Systems, RSS 2020*. [S.l.: s.n.], 2020.

DI, Y.; MANHARDT, F.; WANG, G.; JI, X.; NAVAB, N.; TOMBARI, F. So-pose: Exploiting self-occlusion for direct 6d pose estimation. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. [S.l.: s.n.], 2021. p. 12396–12405.

DO, T.; PHAM, T.; CAI, M.; REID, I. Real-time monocular object instance 6d pose estimation. BMVC Press, 2019.

DOSOVITSKIY, A.; FISCHER, P.; ILG, E.; HAUSSER, P.; HAZIRBAS, C.; GOLKOV, V.; SMAGT, P. V. D.; CREMERS, D.; BROX, T. Flownet: Learning optical flow with convolutional networks. In: *Proceedings of the IEEE international conference on computer vision*. [S.l.: s.n.], 2015. p. 2758–2766.

DOUMANOGLOU, A.; KOUSKOURIDAS, R.; MALASSIOTIS, S.; KIM, T.-K. Recovering 6D object pose and predicting next-best-view in the crowd. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2016. p. 3583–3592.

DROST, B.; ULRICH, M.; BERGMANN, P.; HARTINGER, P.; STEGER, C. Introducing mvtec itodd-a dataset for 3d object recognition in industry. In: *Proceedings of the IEEE International Conference on Computer Vision Workshops*. [S.l.: s.n.], 2017. p. 2200–2208.

EVERINGHAM, M.; GOOL, L. V.; WILLIAMS, C. K. I.; WINN, J.; ZISSERMAN, A. *The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results*. 2012. Http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

GAIDON, A.; WANG, Q.; CABON, Y.; VIG, E. Virtual worlds as proxy for multi-object tracking analysis. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. [S.l.: s.n.], 2016.

GAO, K.; GAO, Y.; HE, H.; LU, D.; XU, L.; LI, J. Nerf: Neural radiance field in 3d vision, a comprehensive review. *arXiv preprint arXiv:2210.00379*, 2022.

GARON, M.; LALONDE, J.-F. Deep 6-dof tracking. *IEEE transactions on visualization and computer graphics*, IEEE, v. 23, n. 11, p. 2410–2418, 2017.

GARON, M.; LAURENDEAU, D.; LALONDE, J.-F. A framework for evaluating 6-dof object trackers. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. [S.l.: s.n.], 2018. p. 582–597.

GARRIDO-JURADO, S.; MUÑOZ-SALINAS, R.; MADRID-CUEVAS, F. J.; MARÍN-JIMÉNEZ, M. J. Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognition*, Elsevier, v. 47, n. 6, p. 2280–2292, 2014.

GE, Y.; BEHL, H.; XU, J.; GUNASEKAR, S.; JOSHI, N.; SONG, Y.; WANG, X.; ITTI, L.; VINEET, V. Neural-sim: Learning to generate training data with nerf. In: SPRINGER. *European Conference on Computer Vision (ECCV)*. [S.l.], 2022. p. 477–493.

GUAN, J.; HAO, Y.; WU, Q.; LI, S.; FANG, Y. A survey of 6dof object pose estimation methods for different application scenarios. *Sensors*, MDPI, v. 24, n. 4, p. 1076, 2024.

HAI, Y.; SONG, R.; LI, J.; SALZMANN, M.; HU, Y. Rigidity-aware detection for 6d object pose estimation. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2023. p. 8927–8936.

HE, Y.; SUN, W.; HUANG, H.; LIU, J.; FAN, H.; SUN, J. 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)*. [S.l.: s.n.], 2020.

HINTERSTOISSER, S.; LEPETIT, V.; ILIC, S.; HOLZER, S.; BRADSKI, G.; KONOLIGE, K.; NAVAB, N. Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes. In: SPRINGER. *Asian conference on computer vision*. [S.l.], 2012. p. 548–562.

HINTERSTOISSER, S.; LEPETIT, V.; ILIC, S.; HOLZER, S.; BRADSKI, G.; KONOLIGE, K.; NAVAB, N. Model-based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes. In: SPRINGER. *Computer Vision–ACCV 2012: 11th Asian Conference on Computer Vision, Daejeon, Korea, November 5-9, 2012, Revised Selected Papers, Part I 11*. [S.l.], 2013. p. 548–562.

HINTERSTOISSER, S.; LEPETIT, V.; WOHLHART, P.; KONOLIGE, K. On pre-trained image features and synthetic images for deep learning. In: *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*. [S.l.: s.n.], 2018. p. 0–0.

HINTERSTOISSER, S.; PAULY, O.; HEIBEL, H.; MARTINA, M.; BOKELOH, M. An annotation saved is an annotation earned: Using fully synthetic training for object detection. In: *Proceedings of the IEEE/CVF international conference on computer vision workshops*. [S.l.: s.n.], 2019. p. 0–0.

HODAN, T.; HALUZA, P.; OBDRŽÁLEK, Š.; MATAS, J.; LOURAKIS, M.; ZABULIS, X. T-less: An rgb-d dataset for 6d pose estimation of texture-less objects. In: IEEE. *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*. [S.l.], 2017. p. 880–888.

HODAN, T.; MICHEL, F.; BRACHMANN, E.; KEHL, W.; GLENTBUCH, A.; KRAFT, D.; DROST, B.; VIDAL, J.; IHRKE, S.; ZABULIS, X. et al. Bop: Benchmark for 6d object pose estimation. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. [S.l.: s.n.], 2018. p. 19–34.

HODAŇ, T.; SUNDERMEYER, M.; DROST, B.; LABBÉ, Y.; BRACHMANN, E.; MICHEL, F.; ROTHER, C.; MATAS, J. Bop challenge 2020 on 6d object localization. In: SPRINGER. *European Conference on Computer Vision*. [S.l.], 2020. p. 577–594.

HONG, Z.-W.; HUNG, Y.-Y.; CHEN, C.-S. Rdpn6d: Residual-based dense point-wise network for 6dof object pose estimation based on rgb-d images. *arXiv preprint arXiv:2405.08483*, 2024.

HOQUE, S.; ARAFAT, M. Y.; XU, S.; MAITI, A.; WEI, Y. A comprehensive review on 3d object detection and 6d pose estimation with deep learning. *IEEE Access*, IEEE, v. 9, p. 143746–143770, 2021.

HORE, A.; ZIOU, D. Image quality metrics: Psnr vs. ssim. In: IEEE. *2010 20th international conference on pattern recognition*. [S.l.], 2010. p. 2366–2369.

HOU, F.; QIAO, B.; DONG, J.; MA, Z. S-cyclegan: A novel target signature segmentation method for gpr image interpretation. *IEEE Geoscience and Remote Sensing Letters*, IEEE, 2024.

HU, Y.; HUGONOT, J.; FUA, P.; SALZMANN, M. Segmentation-driven 6d object pose estimation. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. [S.l.: s.n.], 2019.

ISOLA, P.; ZHU, J.-Y.; ZHOU, T.; EFROS, A. A. Image-to-image translation with conditional adversarial networks. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2017. p. 1125–1134.

JALAL, M.; SPJUT, J.; BOUDAOUD, B.; BETKE, M. Sidod: A synthetic image dataset for 3d object pose recognition with distractors. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. [S.l.: s.n.], 2019. p. 0–0.

KASKMAN, R.; ZAKHAROV, S.; SHUGUROV, I.; ILIC, S. Homebreweddb: Rgb-d dataset for 6d pose estimation of 3d objects. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*. [S.l.: s.n.], 2019. p. 0–0.

KEHL, W.; MANHARDT, F.; TOMBARI, F.; ILIC, S.; NAVAB, N. SSD-6D: Making RGB-based 3D detection and 6D pose estimation great again. In: *Proceedings of the International Conference on Computer Vision (ICCV 2017), Venice, Italy*. [S.l.: s.n.], 2017. p. 22–29.

KEHL, W.; MILLETARI, F.; TOMBARI, F.; ILIC, S.; NAVAB, N. Deep learning of local RGB-D patches for 3D object detection and 6D pose estimation. In: SPRINGER. *European Conference on Computer Vision*. [S.l.], 2016. p. 205–220.

KIM, J.; CHO, H.; KIM, J.; TIRUNEH, Y. Y.; BAEK, S. Sddgr: Stable diffusion-based deep generative replay for class incremental object detection. *arXiv preprint arXiv:2402.17323*, 2024.

LEE, T. E.; TREMBLAY, J.; TO, T.; CHENG, J.; MOSIER, T.; KROEMER, O.; FOX, D.; BIRCHFIELD, S. Camera-to-robot pose estimation from a single image. *arXiv preprint arXiv:1911.09231*, 2019.

LEPETIT, V.; MORENO-NOGUER, F.; FUA, P. Ep n p: An accurate o (n) solution to the p n p problem. *International journal of computer vision*, Springer, v. 81, p. 155–166, 2009.

LI, F.; VUTUKUR, S. R.; YU, H.; SHUGUROV, I.; BUSAM, B.; YANG, S.; ILIC, S. Nerf-pose: A first-reconstruct-then-regress approach for weakly-supervised 6d object pose estimation. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. [S.l.: s.n.], 2023. p. 2123–2133.

LI, Y.; LI, S.; SITZMANN, V.; AGRAWAL, P.; TORRALBA, A. 3d neural scene representations for visuomotor control. In: PMLR. *Conference on Robot Learning*. [S.l.], 2022. p. 112–123.

LI, Y.; WANG, G.; JI, X.; XIANG, Y.; FOX, D. Deepim: Deep iterative matching for 6d pose estimation. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. [S.l.: s.n.], 2018. p. 683–698.

LIN, C.-H.; MA, W.-C.; TORRALBA, A.; LUCEY, S. Barf: Bundle-adjusting neural radiance fields. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. [S.l.: s.n.], 2021. p. 5741–5751.

LIN, C.-H.; WANG, C.; LUCEY, S. Sdf-srn: Learning signed distance 3d object reconstruction from static images. *Advances in Neural Information Processing Systems*, v. 33, p. 11453–11464, 2020.

LIU, W.; LUO, B.; LIU, J. Synthetic data augmentation using multiscale attention cyclegan for aircraft detection in remote sensing images. *IEEE Geoscience and Remote Sensing Letters*, IEEE, v. 19, p. 1–5, 2021.

LIU, Y.; WEN, Y.; PENG, S.; LIN, C.; LONG, X.; KOMURA, T.; WANG, W. Gen6d: Generalizable model-free 6-dof object pose estimation from rgb images. In: SPRINGER. *European Conference on Computer Vision*. [S.l.], 2022. p. 298–315.

LIU, Z.; LIAN, T.; FARRELL, J.; WANDELL, B. A. Neural network generalization: The impact of camera parameters. *IEEE Access*, IEEE, v. 8, p. 10443–10454, 2020.

MANDL, D.; ROTH, P. M.; LANGLOTZ, T.; EBNER, C.; MORI, S.; ZOLLMANN, S.; MOHR, P.; KALKOFEN, D. Neural cameras: Learning camera characteristics for coherent mixed reality rendering. In: IEEE. *2021 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. [S.l.], 2021. p. 508–516.

MARTIN-BRUALLA, R.; RADWAN, N.; SAJJADI, M. S.; BARRON, J. T.; DOSOVITSKIY, A.; DUCKWORTH, D. Nerf in the wild: Neural radiance fields for unconstrained photo collections. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2021. p. 7210–7219.

MILDENHALL, B.; SRINIVASAN, P. P.; TANCIK, M.; BARRON, J. T.; RAMAMOORTHI, R.; NG, R. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, ACM New York, NY, USA, v. 65, n. 1, p. 99–106, 2021.

NGUYEN, T.-H.; LE, V.-H.; DO, H.-S.; TE, T.-H.; PHAN, V.-N. Tqu-slam benchmark dataset for comparative study to build visual odometry based on extracted features from feature descriptors and deep learning. *Future Internet*, MDPI, v. 16, n. 5, p. 174, 2024.

OUYANG, H.; SHI, Z.; LEI, C.; LAW, K. L.; CHEN, Q. Neural camera simulators. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2021. p. 7700–7709.

PAN, S. J.; YANG, Q. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, IEEE, v. 22, n. 10, p. 1345–1359, 2009.

PARK, B.-S.; KIM, J.-K.; SEO, Y.-H. 3d pose estimation using joint-based calibration in distributed rgb-d camera system. *Computers & Graphics*, Elsevier, v. 120, p. 103917, 2024.

PARK, K.; PATTEN, T.; VINCZE, M. Pix2pose: Pixel-wise coordinate regression of objects for 6d pose estimation. In: *Proceedings of the IEEE International Conference on Computer Vision*. [S.l.: s.n.], 2019. p. 7668–7677.

PATEL, V. M.; GOPALAN, R.; LI, R.; CHELLAPPA, R. Visual domain adaptation: A survey of recent advances. *IEEE signal processing magazine*, IEEE, v. 32, n. 3, p. 53–69, 2015.

PENG, S.; LIU, Y.; HUANG, Q.; ZHOU, X.; BAO, H. Pvnet: Pixel-wise voting network for 6dof pose estimation. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2019. p. 4561–4570.

PERAZZO, D.; LIMA, J. P.; VELHO, L.; TEICHRIEB, V. Directvoxgo++: Grid-based fast object reconstruction using radiance fields. *Computers & Graphics*, Elsevier, v. 114, p. 96–104, 2023.

REDMON, J.; FARHADI, A. Yolo9000: better, faster, stronger. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2017. p. 7263–7271.

RENNIE, C.; SHOME, R.; BEKRIS, K. E.; SOUZA, A. F. D. A dataset for improved rgbd-based object detection and pose estimation for warehouse pick-and-place. *IEEE Robotics and Automation Letters*, IEEE, v. 1, n. 2, p. 1179–1185, 2016.

ROZANTSEV, A.; LEPETIT, V.; FUA, P. On rendering synthetic images for training an object detector. *Computer Vision and Image Understanding*, Elsevier, v. 137, p. 24–37, 2015.

ROZANTSEV, A.; SALZMANN, M.; FUA, P. Beyond sharing weights for deep domain adaptation. *IEEE transactions on pattern analysis and machine intelligence*, IEEE, v. 41, n. 4, p. 801–814, 2018.

SAHIN, C.; KIM, T.-K. Category-level 6D Object Pose Recovery in Depth Images. *arXiv preprint arXiv:1808.00255*, 2018.

SHAFIEI, M.; BI, S.; LI, Z.; LIAUDANSKAS, A.; ORTIZ-CAYON, R.; RAMAMOORTHI, R. Learning neural transmittance for efficient rendering of reflectance fields. *arXiv preprint arXiv:2110.13272*, 2021.

SONG, C.; SONG, J.; HUANG, Q. Hybridpose: 6d object pose estimation under hybrid representations. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2020. p. 431–440.

SRINIVASAN, P. P.; DENG, B.; ZHANG, X.; TANCIK, M.; MILDENHALL, B.; BARRON, J. T. Nerv: Neural reflectance and visibility fields for relighting and view synthesis. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2021. p. 7495–7504.

SU, S.-Y.; YU, F.; ZOLLHOEFER, M.; RHODIN, H. A-nerf: Surface-free human 3d pose refinement via neural rendering. *arXiv preprint arXiv:2102.06199*, v. 2, n. 3, p. 4, 2021.

SU, Y.; SALEH, M.; FETZER, T.; RAMBACH, J.; NAVAB, N.; BUSAM, B.; STRICKER, D.; TOMBARI, F. Zebrapose: Coarse to fine surface encoding for 6dof object pose estimation. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2022. p. 6738–6748.

SUI, X.; LI, S.; GENG, X.; WU, Y.; XU, X.; LIU, Y.; GOH, R.; ZHU, H. Craft: Cross-attentional flow transformer for robust optical flow. In: *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition (CVPR)*. [S.l.: s.n.], 2022. p. 17602–17611.

SUNDERMEYER, M.; HODAN, T.; LABBE, Y.; WANG, G.; BRACHMANN, E.; DROST, B.; ROTHER, C.; MATAS, J. Bop challenge 2022 on detection, segmentation and pose estimation of specific rigid objects. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. [S.l.: s.n.], 2023. p. 2785–2794.

TAN, D. J.; NAVAB, N.; TOMBARI, F. Looking beyond the simple scenarios: combining learners and optimizers in 3D temporal tracking. *IEEE Transactions on Visualization & Computer Graphics*, IEEE, n. 1, p. 1–1, 2017.

TEED, Z.; DENG, J. Raft: Recurrent all-pairs field transforms for optical flow. In: SPRINGER. *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*. [S.l.], 2020. p. 402–419.

TEJANI, A.; KOUSKOURIDAS, R.; DOUMANOGLOU, A.; TANG, D.; KIM, T.-K. Latent-Class Hough Forests for 6-DoF Object Pose Estimation. *IEEE transactions on pattern analysis and machine intelligence*, IEEE, v. 40, n. 1, p. 119–132, 2018.

TEKIN, B.; SINHA, S. N.; FUA, P. Real-time seamless single shot 6d object pose prediction. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2018. p. 292–301.

TIAN, M.; PAN, L.; JR, M. H. A.; LEE, G. H. Robust 6d object pose estimation by learning rgb-d features. *arXiv preprint arXiv:2003.00188*, 2020.

TJADEN, H.; SCHWANECKE, U.; SCHÖMER, E.; CREMERS, D. A region-based gauss-newton approach to real-time monocular multiple object tracking. *IEEE transactions on pattern analysis and machine intelligence*, IEEE, v. 41, n. 8, p. 1797–1812, 2018.

TO, T.; TREMBLAY, J.; MCKAY, D.; YAMAGUCHI, Y.; LEUNG, K.; BALANON, A.; CHENG, J.; HODGE, W.; BIRCHFIELD, S. *NDDS: NVIDIA Deep Learning Dataset Synthesizer*. 2018. <https://github.com/NVIDIA/Dataset_Synthesizer>.

TOBIN, J.; FONG, R.; RAY, A.; SCHENEIDER, J.; ZAREMBA, W.; ABBEEL, P. Domain randomization for transferring deep neural networks from simulation to the real world. In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. [S.l.: s.n.], 2017. p. 23–30.

TODOROV, E.; EREZ, T.; TASSA, Y. Mujoco: A physics engine for model-based control. In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. [S.l.: s.n.], 2012. p. 5026–5033.

TREMBLAY, J.; PRAKASH, A.; ACUNA, D.; BROPHY, M.; JAMPANI, V.; ANIL, C.; TO, T.; CAMERACCI, E.; BOOCHOON, S.; BIRCHFIELD, S. Training deep networks with synthetic data: Bridging the reality gap by domain randomization. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. [S.l.: s.n.], 2018. p. 969–977.

TREMBLAY, J.; TO, T.; BIRCHFIELD, S. Falling things: A synthetic dataset for 3d object detection and pose estimation. In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. [S.l.: s.n.], 2018.

TREMBLAY, J.; TO, T.; SUNDARALINGAM, B.; XIANG, Y.; FOX, D.; BIRCHFIELD, S. Deep object pose estimation for semantic robotic grasping of household objects. *arXiv preprint arXiv:1809.10790*, 2018.

TRINH, S.; SPINDLER, F.; MARCHAND, E.; CHAUMETTE, F. A modular framework for model-based visual tracking using edge, texture and depth features. In: IEEE. *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. [S.l.], 2018. p. 89–96.

VALENÇA, L.; SILVA, L.; CHAVES, T.; GÓMES, A.; FIGUEIREDO, L.; COSSIO, L.; TANDEL, S.; LIMA, J. P.; SIMÕES, F.; TEICHRIEB, V. Real-time monocular 6dof tracking of textureless objects using photometrically-enhanced edges. In: *VISIGRAPP (5: VISAPP)*. [S.l.: s.n.], 2021. p. 763–773.

VERBIN, D.; HEDMAN, P.; MILDENHALL, B.; ZICKLER, T.; BARRON, J. T.; SRINIVASAN, P. P. Ref-nerf: Structured view-dependent appearance for neural radiance fields. In: IEEE. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. [S.l.], 2022. p. 5481–5490.

VOLPI, R.; LARLUS, D.; ROGEZ, G. Continual adaptation of visual representations via domain randomization and meta-learning. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2021. p. 4443–4453.

WANG, H.; SRIDHAR, S.; HUANG, J.; VALENTIN, J.; SONG, S.; GUIBAS, L. J. Normalized object coordinate space for category-level 6d object pose and size estimation. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2019. p. 2642–2651.

WANG, T.-C.; LIU, M.-Y.; ZHU, J.-Y.; TAO, A.; KAUTZ, J.; CATANZARO, B. High-resolution image synthesis and semantic manipulation with conditional gans. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2018. p. 8798–8807.

WANG, Z.; BOVIK, A. C.; SHEIKH, H. R.; SIMONCELLI, E. P. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, IEEE, v. 13, n. 4, p. 600–612, 2004.

WEN, B.; TREMBLAY, J.; BLUKIS, V.; TYREE, S.; MÜLLER, T.; EVANS, A.; FOX, D.; KAUTZ, J.; BIRCHFIELD, S. Bundlesdf: Neural 6-dof tracking and 3d reconstruction of unknown objects. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2023. p. 606–617.

WIZADWONGSA, S.; PHONGTHAWEE, P.; YENPHRAPHAI, J.; SUWAJANAKORN, S. Nex: Real-time view synthesis with neural basis expansion. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2021. p. 8534–8543.

WU, P.-C.; WANG, R.; KIN, K.; TWIGG, C.; HAN, S.; YANG, M.-H.; CHIEN, S.-Y. Dodecapen: Accurate 6dof tracking of a passive stylus. In: *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*. [S.l.: s.n.], 2017. p. 365–374.

XIANG, Y.; SCHMIDT, T.; NARAYANAN, V.; FOX, D. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. *arXiv preprint arXiv:1711.00199*, 2017.

XIE, X.; BHATNAGAR, B. L.; PONS-MOLL, G. Visibility aware human-object interaction tracking from single rgb camera. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2023. p. 4757–4768.

XU, X.; ZHANG, T.; WANG, S.; LI, X.; CHEN, Y.; LI, Y.; RAJ, B.; JOHNSON-ROBERSON, M.; HUANG, X. Customizable perturbation synthesis for robust slam benchmarking. *arXiv preprint arXiv:2402.08125*, 2024.

YEN-CHEN, L.; FLORENCE, P.; BARRON, J. T.; RODRIGUEZ, A.; ISOLA, P.; LIN, T.-Y. inerf: Inverting neural radiance fields for pose estimation. In: IEEE. *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. [S.l.], 2021. p. 1323–1330.

YUAN, H.; HOOGENKAMP, T.; VELTKAMP, R. C. Robotp: A benchmark dataset for 6d object pose estimation. *Sensors*, Multidisciplinary Digital Publishing Institute, v. 21, n. 4, p. 1299, 2021.

ZAKHAROV, S.; AMBRUș, R.; GUIZILINI, V.; KEHL, W.; GAIDON, A. Photo-realistic neural domain randomization. In: SPRINGER. *European Conference on Computer Vision (ECCV)*. [S.l.], 2022. p. 310–327.

ZAKHAROV, S.; KEHL, W.; BHARGAVA, A.; GAIDON, A. Autolabeling 3d objects with differentiable rendering of sdf shape priors. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. [S.l.: s.n.], 2020. p. 12224–12233.

ZAKHAROV, S.; SHUGUROV, I.; ILIC, S. Dpod: 6d pose object detector and refiner. In: *Proceedings of the IEEE International Conference on Computer Vision*. [S.l.: s.n.], 2019. p. 1941–1950.

ZHANG, J.; YANG, G.; TULSIANI, S.; RAMANAN, D. Ners: Neural reflectance surfaces for sparse-view 3d reconstruction in the wild. *Advances in Neural Information Processing Systems*, v. 34, p. 29835–29847, 2021.

ZHANG, X.; SRINIVASAN, P. P.; DENG, B.; DEBEVEC, P.; FREEMAN, W. T.; BARRON, J. T. Nerfactor: Neural factorization of shape and reflectance under an unknown illumination. *ACM Transactions on Graphics (ToG)*, ACM New York, NY, USA, v. 40, n. 6, p. 1–18, 2021.