ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

DEPARTMENT OF COMPUTER SCIENCE
AND ENGINEERING

ARTIFICIAL INTELLIGENCE

MASTER THESIS

in

Machine Learning for Computer Vision

CONSISTENT 6D POSE ESTIMATION VIA GEOMETRY-GUIDED OBJECT POSE GRAPH CONSTRUCTION

CANDIDATE
Ludovico Granata

SUPERVISOR
Prof. Samuele Salti

CO-SUPERVISORS
Dr. Fabian Manhardt
Dr. Yan Di

Academic year 2021-2022

Session 3rd

Contents

List of Figures

List of Tables

Abstract

The process of determining the position and orientation of an object in 3D space with respect to the camera is known as 6D pose estimation. This is a fundamental problem in computer vision that has numerous real-world applications. Most of the existing approaches in the literature focus solely on the object itself, without considering its relationships with other objects within the scene. Therefore, in this thesis, we propose a novel method for refining the monocular 6D object pose at the instance level of multiple objects by leveraging geometric information about the relationships of neighboring objects. Our experimental results demonstrate that our architecture, which is trained exclusively on synthetic images, can produce satisfactory results within the synthetic domain. However, when tested on real-world images, it does not deliver the same level of performance.

Introduction

6D Pose Estimation refers to the process of determining the position and orientation of an object in 3D space with respect to the camera. It is a fundamental problem in computer vision that has many real-world applications such as robotic manipulation[4], augmented reality[16] and autonomous driving[24]. The task of 6D pose estimation is particularly challenging, since it requires not only detecting the objects, but also determining their position in the 3D space, despite cluttered environments, occlusions, and varying lighting conditions. To overcome some of these challenges, most traditional methods make use of RGB and depth data, commonly referred to as RGB-D methods. With the advent of deep learning and especially with Convolutional Neural Networks (CNNs)[17], the accuracy of monocular methods (i.e., using only one RGB image) has improved significantly and is now comparable to that of RGB-D methods.

6D Pose estimation can operate at either the instance level or the categorical level. At the instance level, the specific objects in the scene are already known and their computer-aided design (CAD) 3D models are provided as input during training and testing, while, at the categorical level, only the categories of the objects in the scene are known, making the task harder. Instance-level 6D pose estimation is more suitable for closely controlled environments where high precision is desired and objects are not expected to change while categorical-level 6D pose estimation is better suited for less constrained use cases.

In this thesis, we present a method to refine the instance-level monocular 6D pose estimation of multiple objects in the same scene. Our approach takes advantage of the relationship between pairs of objects to gain a more comprehensive understanding of the whole scene, instead of focusing on individual objects. By having a more general understanding of the scene we aim to achieve a more consistent placement of each object resulting in improved accuracy. Specifically, we aim to refine the 6D pose estimates generated by GDR-Net[23], the state-of-the-art model for this task, by using a graph neural network, where each node of the graph represents an object in the scene and each edge encodes geometric features extracted from pairs of neighboring objects.

The structure of this thesis is as follows. The first chapter offers an overview of related works that incorporate graphs and object relations in their pipeline, which provided inspiration for our work. The second chapter introduces the proposed architecture and details each component and its purpose. The third chapter focuses on the dataset we used and the metric used to evaluate the results. Lastly, the fourth chapter outlines the training environment, the loss function utilized, the training process, and the final results.

Chapter 1

Background and Related Works

The aim of this chapter is to give a comprehensive understanding of the existing techniques and approaches used to tackle the problem of 6D Pose Estimation by exploiting relations between objects in the scene. We will start with an introduction to Graph Neural Networks, as they will be used in our architecture, and then we will present three approaches that inspired our work: MonoPair[2], Holistic 3D scene Understanding[28] and Holistic Pose Graph[26].

1.1 Graph Neural Networks

Representing data as graphs is a natural approach for many real-world applications where relationships between entities need to be considered. With the rise of deep learning models in recent years, there have been many attempts to apply this method to graphs, however, the application is not straightforward. For instance, convolution neural networks (CNNs)[17] can achieve excellent performance when applied to images because they can exploit their grid-like nature, allowing them to take advantage of hierarchical patterns and extract high-level features. The fundamental mechanism of CNNs is to learn a fixed-size filter that scans every pixel in the image and combines it with the surrounding pixels. Graphs, on the other hand, can have an irregular structure, so convolution and filtering cannot be defined as with images. Therefore,

researchers have identified two main ways to apply convolution on graphs: spectral graph convolution and spatial graph convolution.

Spectral graph convolutions are defined in the spectral domain that is based on the graph Fourier transform. Here, convolution can be computed by taking the inverse Fourier transform of the multiplication between two Fourier transformed graph signals. The limitations of this type of approach include the high computational costs of the Fourier transformations, as well as the difficulties in generalizing to graphs with different topologies. To address the limitations of spectral graph convolution, researchers have proposed an alternative approach, known as spatial graph convolution. The central idea of spatial graph convolution is to directly aggregate the features of neighboring nodes in the graph to compute a new representation for each node. As a result, spatial graph convolution approaches are more computationally efficient as they don't require the computation of the Fourier transform, and also they can capture local relationships, being able to generalize to different types of graphs. One of the most influential papers on this subject is the Graph Convolutional Network (GCN)[12], which introduced a method that is still used today and inspired many subsequent works. GCN is often considered to bridge the gap between spectral-based methods and spatial-based methods, as can be explained from both perspectives.

A graph **G** is defined as $\mathbf{G} = \{\mathbf{V}, \mathbf{E}\}$, where **V** is the set of nodes and **E** is the set of edges. The adjacency matrix **A** is built by setting the entry $\mathbf{A}(i, j)$ to 1 if there is an edge between i and j, 0 otherwise. The degree matrix of **A** , **D** , is a diagonal matrix defined as

$$D(i,i) = \sum_{j=1}^{n} \mathcal{A}(i,j)$$

The output $\mathbf{H}^{l+1}$ of the convolution at layer l is defined as

$$H^{l+1} = \sigma(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)})$$

where $\tilde{\mathbf{A}}$ is the adjacency matrix summed with the identity matrix, while $\tilde{\mathbf{D}}$, the degree matrix computed on $\tilde{\mathbf{A}}$, is used to normalize the adjacency matrix. The matrix $\mathbf{W}^{(l)}$ contains the trainable weights while $\sigma(\cdot)$ is the activation function that introduces non-linearity to the network.

In the following, we introduce another model based on graph convolution, called Graph R-CNN[27] that has been used as a reference for our architecture.

1.1.1 Graph R-CNN

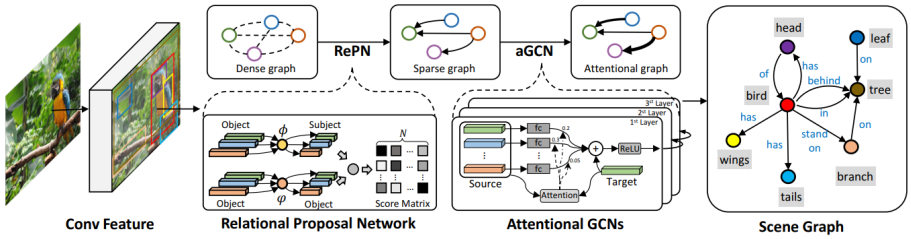


Figure 1.1: Pipeline of the Graph R-CNN architecture [27]

Graph R-CNN[27] is a scene graph generation model that is capable of effectively detecting objects and their semantic relations from an image. The pipeline, shown in Figure 1.1 consists essentially of three parts: an object proposal, a relation proposal, and an attentional GCN. The Faster R-CNN [20] framework is used to extract the object proposals from the image including the coordinates of the bounding box and the label associated with the detected object. The Relation Proposal Network is then introduced to prune the dense graph obtained by adding a relation between each object proposal. In particular, not every pair of objects will be related, so a relatedness score is learned and every relationship with a score below a certain threshold is pruned.

The final graph is constructed considering both the object proposal and the relation proposal as nodes. An edge is added between all the object nodes and also between the relation node and the object involved in the relation. Once

the final graph is built, the author proposes to use an attentional graph convolutional network (aGCN)[22]. Instead of weighting each node of a neighborhood with coefficients based on the symmetrically normalized adjacency matrix like in GCNs, the weighting coefficients are learned through an attention mechanism that consists of a 2-layer MLP over the concatenation of pairs of node features. The final attention coefficients are obtained by computing the softmax over the resulting scores

$$u_{ij} = W_2\sigma(W_1[z_i, z_j])$$

$$a_i = \texttt{softmax}(u_i)$$

The constructed graph has different types of connections:

- object $\leftrightarrow$ relationship
- relationship $\leftrightarrow$ subject
- object $\leftrightarrow$ object

each type of connection will have its own weights, so the features of an object node z_i^o can be computed as:

$$z_i^o = \sigma(W^{oo}Z^o a^{oo} + W^{rs}Z^r a^{rs} + W^{ro}Z^r a^{ro})$$

while for relationship node:

$$z_i^r = \sigma(z_i^r + W^{sr}Z^o a^{sr} + W^{or}Z^o a^{or})$$

where s = subjects, o = objects, and r = relationships.

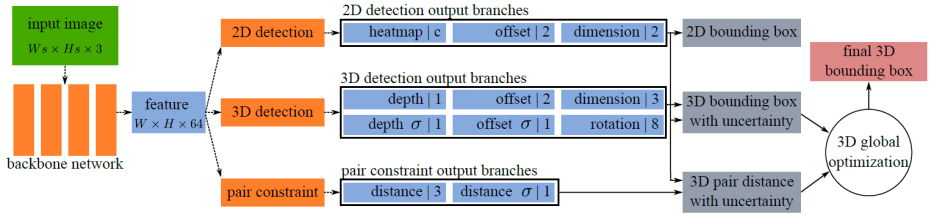


Figure 1.2: Overview of the MonoPair architecture [2]

1.2 MonoPair

MonoPair[2] is a research paper that proposes a novel method for Monocular 3D Object Detection using pairwise spatial relationships. Published in 2020 this work was the first to add relations between pairs of objects to estimate the 6D pose.

The proposed network uses a one-stage architecture, as shown in Figure 1.2. It is composed of a backbone network followed by three task-specific branches. It takes as input a monocular RGB image, then the backbone extracts a feature map that will be the input to the three following branches:

- **2D detection** $\rightarrow$ the 2D detection module, derived from CenterNet[6], has to detect the objects in the image, estimating their bounding boxes and their labels;
- **3D detection** $\rightarrow$ the 3D detection module predicts the center of the 3D bounding box, the size, and its orientation. The center is determined in two steps: first, the offset between its projection onto the 2D feature map and the center of the 2D bounding box is calculated, then the depth is determined. The dimension of the bounding box is regressed directly, while, to determine the object orientation, local orientation is used instead of global orientation in the camera coordinate system. This approach is preferred because local orientation is more meaningful when dealing with image features;
- **Pair Constraint** $\rightarrow$ the pair constraint module estimates the pairwise

geometric constraint among adjacent objects via keypoints on the feature map. For each selected pair, the keypoint is placed in the middle of their 2D bounding box centers, and the regression target for the keypoints is the 3D distance between the two objects.

The 3D bounding boxes estimated by the network, involved in at least one of the pair constraints, will be refined by a 3D global optimization step, one of the main contributions of the paper. The optimization is formulated as a nonlinear least square problem

$$\underset{(u_t,v_t,z_t)}{\arg\min}\ e^T W e$$

where e is the error vector, **W** is the weight matrix for the different errors, u and v are the projected center of the 3D bounding box on the feature map, while z is the depth. There are three error terms e^x, e^y, and e^z that are obtained by measuring the inconsistency for each axis between the estimated 3D distance and the distance obtained as the difference of the 3D bounding box of each object.

1.3 Holistic 3D Scene Understanding

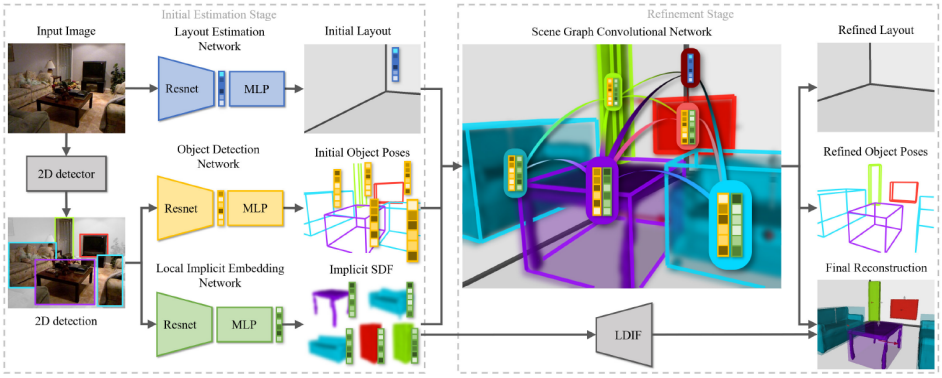


Figure 1.3: Overview of the Holistic 3D Scene Understanding architecture [28]

Holistic 3D Scene Understanding [28] is a research paper that proposes a

new pipeline for holistic 3D Scene Understanding from a single image. The architecture goes beyond the scope of this thesis as it not only estimates the 6D poses of the objects but also estimates the 3D geometry of each object. Even if the task is different we have identified some interesting ideas, such as the use of Scene Graph Convolutional Networks.

The proposed architecture consists of two stages: the initial estimation stage and the refinement stage, as shown in Figure 1.3. The initial estimation stage includes a 2D detector to identify the 2D bounding box of the objects in the scene, an Object Detection Network (ODN) to compute 3D bounding boxes, and a Local Implicit Embedding Network (LIEN) to extract local shape information from each object to infer 3D geometry. The input image is also processed by the Layout Estimation Network (LEN) to produce 3D layout bounding boxes and the relative camera pose.

In the refinement stage, the initial 3D bounding box predictions are refined by incorporating scene context information using the Scene Graph Convolutional Network. This network is based on Graph R-CNN [27], it creates a graph by treating objects in the scene as nodes and connecting each object to others, allowing information to flow between them. In a second step relation nodes are added between pairs of neighboring object's nodes. Each node has a set of features: object nodes will include the object id, the initial 3D bounding box estimation from the ODN network, and the object geometry embedding from the LIEN network; the relation nodes will have as feature the 2D bounding box coordinate for the two objects involved in the relation. The graph is designed with different types of nodes, so different message passing weights are defined for each of the source-destination types for the graph convolution. The final refined results are obtained by passing each node's representation to a Multi Layers Perceptron (MLP).

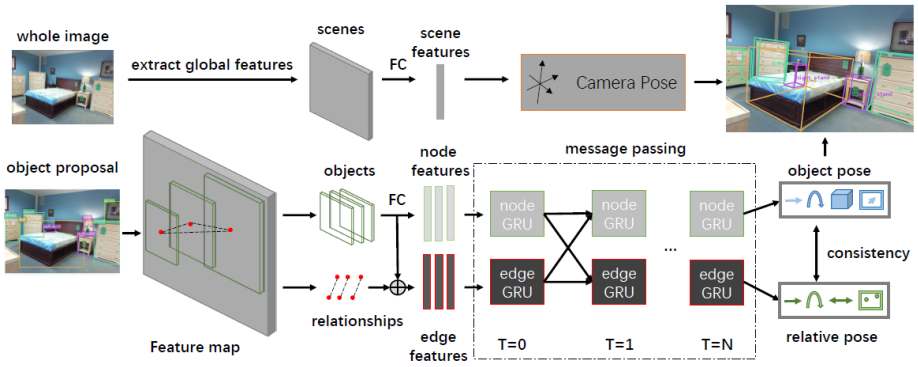


Figure 1.4: Overview of the Holistic Pose Graph Architecture [26]

1.4 Holistic Pose Graph

Holistic Pose Graph[26] is a research paper proposing a new pipeline for Monocular 6D Pose Estimation at categorical level.

The proposed architecture in Figure 1.4 consists of two branches that predict the pose of the camera and the pose of each object, respectively. The first branch uses a ResNet-34[9] network as features extractor starting from the whole image to estimate the camera pose.

The second branch first computes the 2D bounding box, then each detected object proposal is cropped and fed to a ResNet-34 to extract its features. A fully connected graph that models all geometric relationships between each pair of objects is built by representing each detected object as a node. To integrate geometric information, a message passing mechanism is used to pass the information along the graph structure and update the node GRUs[3] and the edge GRUs. The GRUs of nodes are initialized with the features previously extracted for each object, while the GRUs of edges are initialized with the features created by concatenating the features of the objects involved in the relationship. All node GRUs share the same weights, and all edge GRUs share another set of weights. During each iteration of message passing, each node GRU is updated with messages from all related edges, while each edge GRU is updated with messages from its subject node and object node. The features

of each final object node are used to regress the 6D object pose, while the relative pose between the two connected objects is regressed for each edge node. A term is also added to the loss function to enforce the consistency between the object pose and the relative pose.

Chapter 2

Architecture

2.1 Overview

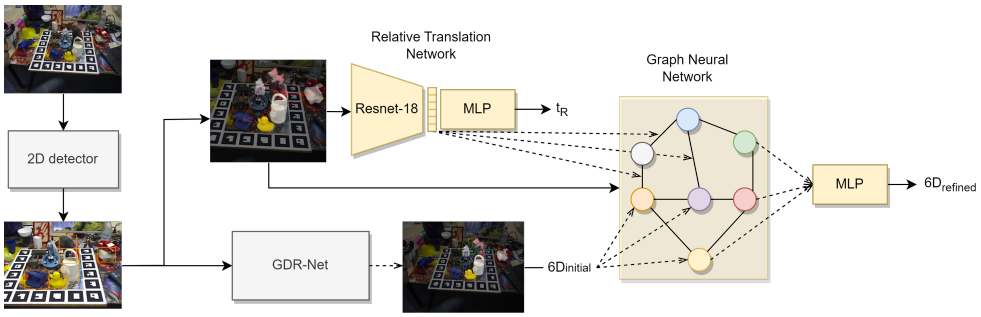


Figure 2.1: Overview of the proposed architecture

The proposed architecture is built on the premise that having a deep understanding of a scene's characteristics, particularly in terms of the geometric relationships between objects, can significantly enhance the accuracy when estimating the 6D pose of each individual object. Therefore, the architecture is designed to identify object pairs within the scene, extract their spatial relationships, and use this information to process the entire scene as a whole.

The proposed architecture, as illustrated in Figure 2.1, includes four building components:

- **2D detector** → the 2D detector is implemented using YOLOX [8]. Its function is to detect objects present in the image by estimating their 2D bounding box and their identifiers. The 2D bounding box information is essential for all the subsequent parts of the architecture;

- **6D object pose estimator** → the 6D object pose estimator is implemented using GDR-Net [23]. It operates at the instance level, utilizing only a single RGB image to estimate the initial 6D poses for each object detected by the 2D detector. This initial estimation is then refined by the other parts of the network;

- **Relative Translation Network (RTN)** → the third component, the Relative Translation Network, encodes the relative geometry between pairs of neighboring objects. It takes pairs of objects and extracts their spatial relationships, which are used to refine the initial 6D poses;

- **Pose Graph Neural Network (PGNN)** → the Pose Graph Neural Network leverages the initial 6D estimation along with the pairs embedding from the Relative Translation Network and performs graph convolution obtaining the refined 6D poses for each object.

In the following sections, we will provide a more detailed explanation of each component of the architecture.

2.2 YOLOX

YOLOX[8] is an object detection model that was introduced in 2021 as an improvement over the YOLO series. The authors build it on top of the most widely used detectors in the industry, YOLOv3[19] by introducing several modifications that we summarize in the following:

- **Decoupled head** → in YOLOv3, the detection head is a single branch

that outputs both the class probabilities and the bounding box coordinates for each object. This means that the head has to perform both classification and regression tasks, which can lead to suboptimal performance in both tasks. In YOLOX, the detection head is decoupled into two separate branches: a classification branch and a regression branch. The classification branch predicts the probability of an object belonging to each class, while the regression branch predicts the bounding box as four values (top-left corner and height and width). This decoupling of the head allows for more efficient and accurate predictions, as the two tasks can be optimized separately;

- **Strong data augmentation** $\rightarrow$ with respect to YOLOv3, YOLOX introduces two strong data augmentation techniques called Mosaic and MixUp. Mosaic is a data augmentation technique that combines four images into a single mosaic image that is then used as input to the network. Specifically, the four images are randomly cropped and resized and then combined into a single mosaic image together with the adjusted object annotations. MixUp is an augmentation technique that involves blending two images and their corresponding labels to create a new training example. Specifically, this technique involves taking two random images and blending their pixels together using a predetermined ratio. In addition, the one-hot labels are also mixed up using the same ratio;

- **Anchor-Free approach** $\rightarrow$ in the anchor-based approach used in YOLOv3, the detection head predicts the offsets of predefined anchor boxes, which are fixed in size and aspect ratio. The network then adjusts the anchor boxes based on the predicted offsets and assigns each object to the anchor box with the highest Intersection-over-Union (IoU) overlap. Although this approach works well in practice, it requires careful selection of anchor boxes and complicates the overall pipeline. In the

anchor-free approach used in YOLOX, based on FCOS [21], the detection head directly predicts the coordinates of the bounding box for each object, without the need for predefined anchor boxes. Specifically, the detection head predicts four values for each object: the coordinates of the top-left corner and the width and height of the bounding box;

- **SimOTA** → during training the method for determining positive and negative samples for detection is referred to as label assignment strategy. In recent years advanced label assignment has become an important improvement for object detection, and YOLOX proposes SimOTA. SimOTA is based on OTA[7], and approaches the problem from a global perspective, viewing it as an Optimal Transport (OT) problem. Instead of using the computationally expensive Sinkhorn-Knopp algorithm, SimOTA utilizes a dynamic top-k strategy that can generate an approximate solution for the OT problem.

2.3 GDR-Net

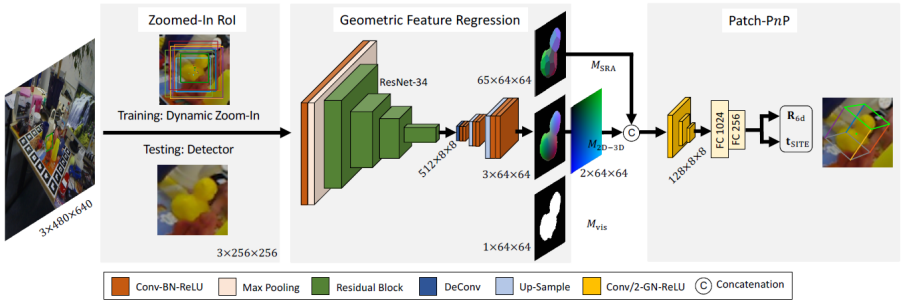


Figure 2.2: Overview of the GDR-Net architecture[23]

GDR-Net [23] is a state-of-the-art model for monocular 6D pose estimation at instance level. Differently from the majority of the work on this task, GDR-Net doesn't use P*n*P/RANSAC to find 2D-3D correspondences but it uses a method fully based on deep neural network. The usage of P*n*P/RANSAC is proven to be very effective but it can be very time-consuming and it also

makes the pipeline more complex as it has to decouple the problem into two separate steps, one of which is not differentiable preventing the network to be trained in an end-to-end fashion.
The GDR-Net architecture, depicted in Figure 2.2, zooms into the region of interest corresponding to each object detection and utilizes it as input to predict multiple intermediate geometric feature maps. The 6D object pose is directly regressed from the dense correspondence-based intermediate geometric features. In particular, the network predicts three intermediate feature maps:

- **Dense Correspondences Map**: the Dense Correspondences Map is obtained by stacking the Dense Coordinates Map onto a map containing the corresponding 2D pixel coordinates from the original image. The Dense Coordinates Map itself is generated through the rendering of the 3D object model in its associated pose;
- **Surface Region Attention Map**: starting from the Dense Coordinates Maps, the Surface Region Attention Map can be obtained by employing farthest points sampling;
- **Visible Object Mask** : the Visible Object Mask is a binary mask that indicates which pixels in the cropped image correspond to the visible surface of the 3D object.

As these intermediate geometric feature maps are organized as 2D-3D correspondences with respect to the image, a 2D convolution is used to estimate the 6D object pose. This module is referred to as Patch-P*n*P.

GDRNPP

In our network, we use an updated version of GDR-Net, called GDRNPP[14], which is the winner (most of the awards) of the BOP Challenge 2022 3.1. The differences with GDR-Net include:

- **Domain Randomization**: usage of stronger domain randomization;

- **Backbone**: usage of the more powerful ConvNeXt[15] backbone instead of ResNet-34[9];
- **Decoupled Head**: usage of two mask heads for predicting amodal mask and visible mask separately;
- **Hyperparameters**: usage of different learning rate, weight decay, and other hyperparameters.

2.4 Relative Translation Network

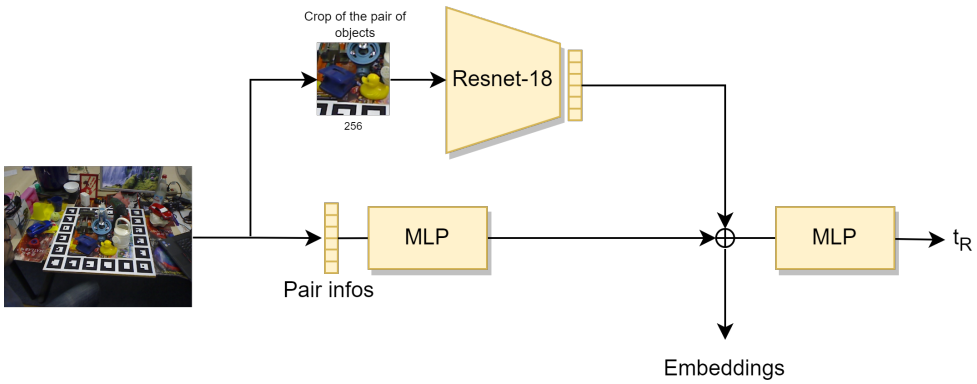


Figure 2.3: Relative Translation Network architecture

2.4.1 Preprocessing

Pair selection

The process of selecting pairs of objects in the proposed method follows the same algorithm described in MonoPair[2]. The approach involves selecting pairs of neighboring objects that have no multiple objects between them. Specifically, the process begins by randomly selecting two objects and considering the centers of their respective 2D bounding boxes. A pair of objects is then defined as neighbors if no other object center falls within the circumference, which has the mean point between the two objects' centers as its center

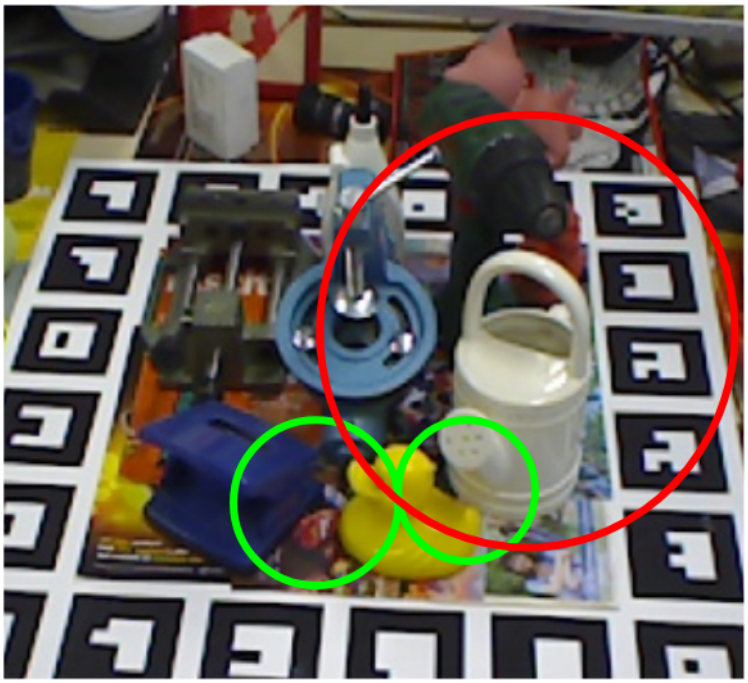


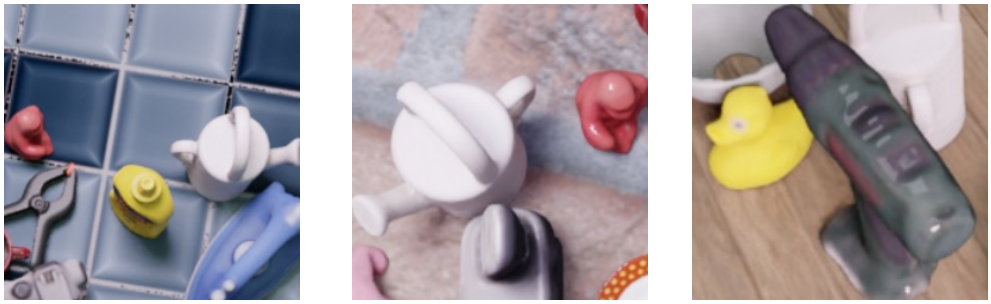
Figure 2.4: Pair selection algorithm. The green circles identify two neighboring objects. The red circle has other object centers that fall in it, resulting in the classification of the yellow duck and the pink cat as not neighbors.

and the distance between the mean and one of the objects as its radius.

Crop and Pair Infos

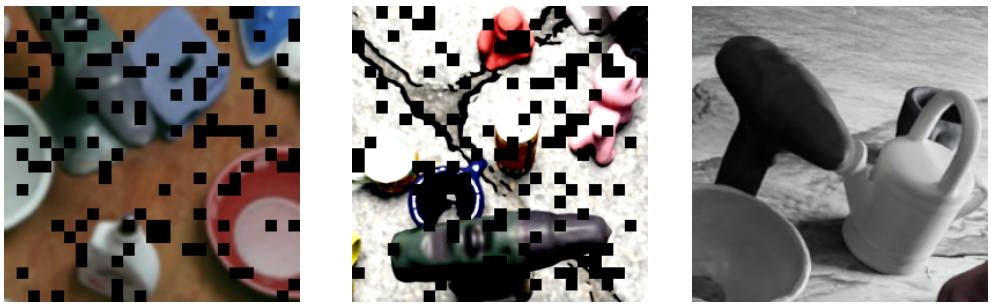
After selecting neighboring object pairs from the input image, the proposed method extracts the input for the network. To begin, the RGB image is cropped to a square of size 256×256 that includes both objects. In order to avoid deformation of the image that could lead to degraded results, the square crop is zoomed in or out until the two objects are included, padding with zeros if necessary. The resulting image is then normalized using a mean of $(0.485, 0.456, 406)$ and a standard deviation of $(0.229, 0.224, 0.225)$.

In addition to the cropped image, information about each object is concatenated together for each pair. This includes the 2D bounding box for each object, its unique object ID, and information about the object model, such as



(a) (b) (c)

Figure 2.5: Crop of images containing pair of objects



(a) (b) (c)

Figure 2.6: Example of augmentation techniques applied to crops of the image

its diameter and size.

Augmentation

As described in Section 3.2, during training the input images have to be exclusively synthetic, while for testing they are real-world images; this can result in a mismatch between the performance of the network during training and testing. This problem is known in the relevant literature as domain gap, and the problem arises from the fact that the synthetic and real-world images live in the same feature space but have different distributions. The problem is partially addressed by the way synthetic images are produced. In fact, as described in Section 3.3, thanks to the physically-based renderer the produced images are almost photorealistic, resulting in a distribution that is closer to the real images. However, even if the differences with real images are only slightly noticeable, the domain gap is still there. To further tackle the problem

we propose to apply strong data augmentation. Data augmentation refers to the process of creating new training data from the original dataset by applying various transformations to the images. It is a standard technique, not only for computer vision tasks, that is used to improve the performance and generalization capabilities of the model but also to tackle the domain gap problem, as in our case. We will list all the pose invariant transformations that we have applied in the following:

- **Cutout**: sets rectangular areas within images to zero;
- **Gaussian Blur**: blur images using the Gaussian kernel;
- **Enhance Sharpness**;
- **Enhance Contrast**;
- **Enhance Brightness**;
- **Enhance Color**;
- **Add**: add a value to all pixels in an image;
- **Invert**: inverts all values in images, i.e. sets a pixel from value v to $255 - v$;
- **Multiply**: multiply all pixels in an image with a specific value;
- **Additive Gaussian Noise**: add noise sampled from gaussian distributions elementwise to images;
- **Linear Contrast**: scale each pixel to $127 + alpha \cdot (v - 127)$, where v is the value of a pixel and alpha is sampled uniformly from an interval;
- **Grayscale**: convert the image to the grayscale version of the image with varying strengths.

Each augmented image is the result of a random subset of all this transformation, both in order and in number.

2.4.2 Architecture

The proposed architecture for the Relative Translation Network (RTN) is shown in Figure 2.3 and has the purpose of encoding the spatial relationships between pairs of objects within the scene.

To achieve this, the RTN takes as input a crop of pairs of objects, which is obtained using the method described in 2.4.1. The crop is then processed by ResNet-18[9], a widely used neural network architecture for computer vision tasks.

In addition, the RTN processes in a second branch the pair's information using a fully connected layer. The output, together with the output of the ResNet-18 are concatenated and used as embeddings in the Pose Graph Neural Network 2.5.

Furthermore, the concatenation is also used as the input for a fully connected layer that regresses the relative position between the two objects as a three-dimensional translation vector.

2.5 Pose Graph Neural Network

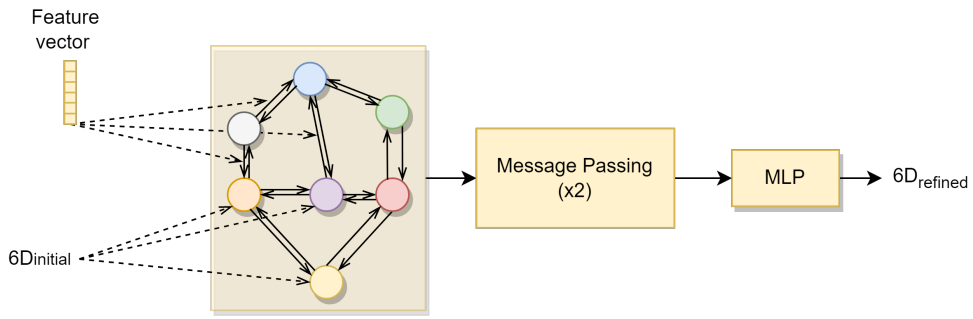


Figure 2.7: Architecture of the Pose Graph Neural Network

The proposed architecture for the Pose Graph Neural Network is inspired by Graph R-CNN[27], in particular by the way in which the attentional graph neural network is used, as described in 1.1.1.

The graph is constructed by considering each detected object as a node, while

the relations between the objects are defined by applying to each possible pair of objects the algorithm presented in 2.4.1. The relations are also represented as nodes, an edge is inserted between the relation nodes and their associated objects. An edge is also added between each object, resulting in an even greater information flow. The constructed graph is directional but the information can always flow in both directions (e.g., from object node to relation node and also from relation node to object node), so for each relation, we will add two edges.

The object nodes are initialized with the 6D pose computed by GDR-Net[23] while the relation node with the embedding estimated by the Relative Translation Network 2.4.

After the data in the graph is processed as described in 1.1.1 a shared head for each object node is employed as two MLPs to regress the refined 3D rotation and 3D translation for each object.

Chapter 3

Evaluation Framework

In this chapter, we will delve deeper into the chosen framework to evaluate our architecture. We will provide a comprehensive overview of the task, the utilized dataset, and the employed evaluation metrics. All of these elements will be described within the context of the Benchmark for 6D Object Pose Estimation (BOP)[10], which we will discuss in the next section.

3.1 BOP

The BOP (Benchmark for 6D Object Pose Estimation)[10] is a benchmark evaluation framework for 6D object pose estimation algorithms. It provides researchers with a standardized evaluation platform including common formats, datasets, and metrics to fairly compare the performance of their algorithms. In recent years, to accelerate the research process, a challenge with a rewarding prize has been established attracting a significant number of submissions, with hundreds of entries received, reflecting a high level of interest among the research community. In the following sections, we will describe in detail every aspect of the challenge.

3.2 Task

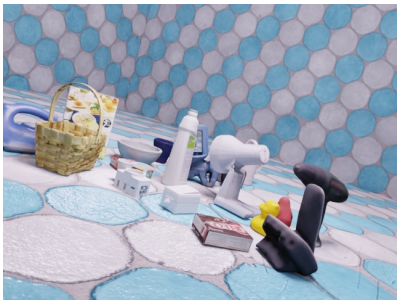
In this study, we address one of the tasks defined by the BOP challenge. The objective is to accurately estimate the 6D pose of an arbitrary number of objects present in a single RGB image. The constraint is that the training data must consist solely of synthetic images and the use of depth information is not allowed. However, the 3D mesh models of the objects can be used during training and testing. The expected output is a 3×3 rotation matrix $\mathbf{R}$ and a 3×1 translation vector $\mathbf{t}$ for each detected object. The matrix $\mathbf{P} = [\mathbf{R}|\mathbf{t}]$ represents the rigid transformation from the 3D coordinate system of the object model to the 3D coordinate system of the camera.

3.3 Dataset

The dataset that we have chosen is Linemod-Occluded(LM-O)[10], one of the core datasets of BOP. The reason for this choice is that it is the only one among the core datasets that have annotations for multiple objects in the same scene during testing; this is very important because our method's core contribution is the use of multiple objects to refine the position of each of them. Moreover, the dataset offers challenging test cases with various levels of occlusion. LM-O consists of 50k synthetic images for training Figure 3.1(a) and 200 real images for testing Figure 3.1(b). It features 8 objects of common use or little toys: a toy ape, a can, a toy cat, a driller, a duck, an eggbox, a glue, and an holepuncher.

The synthetic training images are photorealistic images generated and automatically annotated by BlenderProc4BOP [5], which is a physically-based renderer (PBR), able to accurately simulate the flow of light energy in the scene by ray tracing. To generate a high degree of variability, the objects in the dataset are randomly placed within an empty cube. A randomly selected texture from a library is assigned to the wall of the cube. Furthermore, the

(a)



(b)

Figure 3.1: Examples taken from the LM-O dataset[1] of real-world image(a) and synthetic PBR image(b)

properties of the light source, including intensity, color and position, are also randomly chosen to further increase the diversity of the dataset.

The real-world images used for testing in this dataset were captured within the same environment, a work table, where objects were placed randomly, often with significant occlusion. The images were taken under different lighting conditions and with varying camera positions, further increasing the complexity and realism of the test environment.

The validation set is not predefined in the LM-O, so we have taken 1k images from the 50k training ones as validation images, we also extracted another 1k image to test on synthetic images, resulting in 48k images for training 1k images for validation, 1k synthetic images for testing, and 1k real images also for testing.

Every image is annotated following the BOP format[1], each image includes the ground truth bounding box position of each object, its rotation and translation matrix.

[1] `https://github.com/thodan/bop_toolkit/blob/master/docs/bop_datasets_format.md`

3.4 Metrics

BOP uses three different metrics to compute the error of an estimated 6D pose **P** with respect to the ground-truth 6D pose $\hat{\mathbf{P}}$. In the following, we will explain more in detail the Visible Surface Discrepancy (VSD), the Maximum Symmetry-Aware Surface Distance (MSSD), and the Maximum Symmetry-Aware Projection Distance.

3.4.1 Visible Surface Discrepancy

The VSD (Visible Surface Discrepancy) loss measures the alignment of the estimated pose with the ground truth pose only where the object is visible:

$$e_{VSD}(\hat{D}, \overline{D}, \hat{V}, \overline{V}, \tau) = avg_{p \in \hat{V} \cup \overline{V}} \begin{cases} 0 & \text{if } p \in \hat{V} \cap \overline{V} \wedge |\hat{D}(p) - \overline{D}(p)| < \tau \\ 1 & \text{otherwise} \end{cases}$$

$\hat{D}$ and $\overline{D}$ are distance maps, at each pixel *p* is stored the distance from the camera center to the 3D point x_p that is projected to the pixel *p*. In particular, $\hat{D}$ and $\overline{D}$ are obtained by rendering the object model in the estimated pose and in the ground-truth pose respectively. For each visible pixel, both in the estimated and in the ground truth position, the VSD output is 0 if the pixel is visible both in the estimated and in the ground truth position and the difference of the distance maps is less than a certain tolerance; otherwise, the output is 1. The output for each pixel is in the end averaged, resulting in a loss that ranges between 0 and 1.

3.4.2 Maximum Symmetry-Aware Surface Distance

The MSSD (Maximum Symmetry-Aware Surface Distance) loss measures the largest distance between the vertices of the model in the estimated position and in the ground truth position, taking into account the symmetry of the object

model:

$$e_{MSSD}(\hat{P}, \overline{P}, S_M, V_M) = \min_{\mathbf{S} \in S_M} \max_{x \in V_M} ||\hat{\mathbf{P}}x - \overline{P}Sx||_2$$

where S_M contains global symmetry transformations of the object model M, and V_M contains the model vertices. For each symmetry, we compute the maximum Euclidean distance of the vertexes for the object in the estimated position and in the ground truth position. Among all the maximum distances for each symmetry, we choose the smaller one.
This loss function is especially significant in the context of robotic manipulation tasks, as the maximum deviation between the surfaces directly indicates the likelihood of a successful grasp.

3.4.3 Maximum Symmetry-Aware Projection Distance

The MSPD (Maximum Symmetry-Aware Projection Distance) loss measures the largest distance between the projected vertices in the 2D plane of the model in the estimated position and in the ground truth position:

$$e_{MSPD}(\hat{P}, \overline{P}, S_M, V_M) = \min_{\mathbf{S} \in S_M} \max_{x \in V_M} ||\text{proj}(\hat{P}x) - \text{proj}(\overline{P}Sx)||_2$$

The symbols meaning is the same as in MSSD 3.4.2, the function proj(·) is the projection of the 3D point of the model onto the 2D image plane.
As this loss function only takes into account the perceivable discrepancy and does not consider the depth axis, it is relevant for augmented reality applications and ideal for evaluating methods that use RGB data exclusively.

3.4.4 Accuracy Score

For each of the described losses, we computed the Recall value, the fraction of annotated object instances for which a correct pose is estimated. A pose estimate is considered correct in relation to a pose-error function *e* if it is less

than a certain threshold of correctness $e < \theta_e$. The Average Recall (AR) with respect to a loss is defined as the average of recall values corresponding to different thresholds of correctness and also for varying misalignment tolerances τ in the case of e_{VSD}. In particular, they are the average of the following:

- $AR_{VSD} \rightarrow \tau$ ranging from 5% to 50% of the object diameter with a step of 5% and for θ_{VSD} ranging from 0.05 to 0.5 with a step of 0.05.
- $AR_{MSSD} \rightarrow \theta_{MSSD}$ ranging from 5% to 50% of the object diameter with a step of 5%
- $AR_{MSPD} \rightarrow \theta_{MSPD}$ ranging from 5*r* to 50*r* with a step of 5*r*, where $r = w/640$, where w is the image width in pixels.

As a single general metric is proposed the average of the ARs of VSD, MSSD, and MSPD:

$$AR_D = (AR_{VSD} + AR_{MSSD} + AR_{MSPD})/3$$

Chapter 4

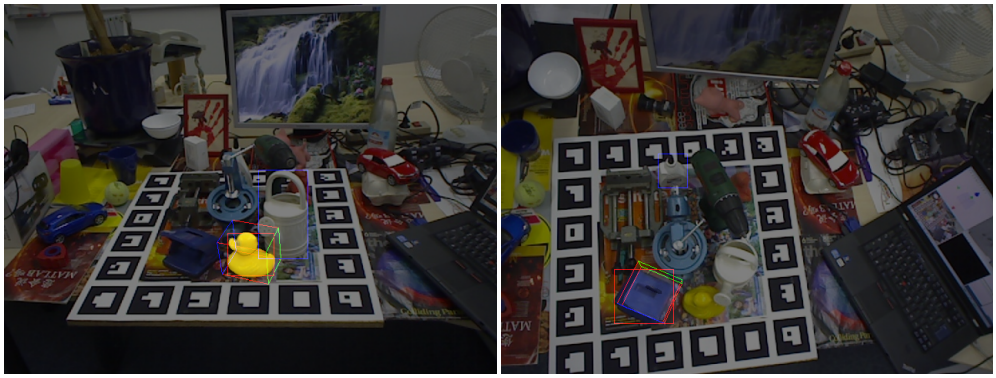
Experiments and Training Setup

We adopted a two-stage training strategy for our network: first, we trained the Relative Translation Network (RTN) alone, then the whole network using the pre-trained RTN. This strategy yielded better results than training the entire network from scratch. We did not train YOLOX and GDR-Net as we used the checkpoints published by the original authors of GDR-Net and kept their weights fixed throughout our experiments. In this section, we provide more details on how we trained our proposed network, including the training environment, the hyperparameters, and the losses for each stage of the training.

4.1 Environment

All of the experiments were done on the HPC cluster[1] offered by the University of Bologna, that is equipped with NVIDIA 2080Ti® GPUs.
The chosen programming language is Python as it has increasingly become one of the most used programming languages for machine learning tasks resulting in a big community of developers and researchers and a lot of available libraries and frameworks. For this project, the framework chosen for building the Neural Networks is PyTorch[18], a popular open-source deep learning

[1] `https://disi.unibo.it/it/dipartimento/servizi-tecnici-e-amministrativi/servizi-informatici/utilizzo-cluster-hpc`



(a) (b)

Figure 4.1: In the images, two bounding boxes are shown, one in blue identifying the starting object, while the other in red is the target object. A rendering of the object in the estimated position is shown.

framework that is easy to use, flexible, and well integrated with other popular libraries.

4.2 Relative Translation Network

As anticipated, the first phase of the training was dedicated to the Relative Translation Network, whose architecture was described in 2.4. The objective of this part of the network is to extract, from the crop of pairs of objects, relevant geometric embedding. To do this we train the network to regress the relative position between the pair of objects as shown in Figure 4.1, the ground truth is computed by subtracting the ground truth position of the first object from the ground truth position of the second one. The used loss is the Mean Square Error (MSE):

$$L_{RTN} = MSE((t_{obj2} - t_{obj1}), (\tilde{t}_{obj2} - \tilde{t}_{obj1}))$$

To evaluate the performance of the model we use a custom metric, guided by the idea that the error committed by the network should be normalized by how

Model	Augmentation	Test Synth	Test Real
RTN	Yes	16.35	25.6
RTN	No	15.51	29.89

Table 4.1: The table shows the results of the RTN model, tested on both real images and synthetic ones. A comparison of the usage of augmentation is also presented.

far apart the two objects are:

$$e_{RTN} = \frac{|(t_{obj2} - t_{obj1}) - (\tilde{t}_{obj2} - \tilde{t}_{obj1})|_2}{|\tilde{t}_{obj2} - \tilde{t}_{obj1})|_2}$$

The results shown in Table 4.1 suggest that even when data augmentation is used, the performance of the network on synthetic images is much better than on real images. The usage of augmentation makes the difference lower but does not solve the problem. This problem will also be reflected in the final results 4.3.3.

4.2.1 Setup

The network was trained for 30 epochs using the Adam optimizer [11], weight decay of $1e-3$, and a batch size of 128. To adjust the learning rate during training, we employed a scheduler from PyTorch called StepLR, which reduces the learning rate by a factor of 0.5 every 5 epochs starting from $1e-3$. Additionally, we implemented early stopping as a means of preventing the model from overfitting with patience equal to 5.

4.3 Graph Pose Network

The second phase of the training is dedicated to the entire network. The Relative Translation Network part of the architecture is initialized with the pretrained weights. We start the training by keeping the Relative Translation Network weights frozen and train only the graph neural network part of the

architecture. In the final part of the training also the RTN weights are unfrozen to further get the most from the model. The regression target for the network is the 3D translation and 3D rotation for each object. The 3D translation is directly expressed as a vector of length three representing the x,y, and z coordinates of the object in the camera reference system. For the rotation, different representations can be chosen but many of them exhibit ambiguities, meaning that even if $\mathbf{R}_i \neq \mathbf{R}_j$ it can still happen that they describe the same rotation. A possible solution is to use representations that are unique, like unit quaternions [25], however representation with four or fewer dimensions for 3D rotation shows discontinuities in the Euclidian space. The representation that we have chosen is proposed by [29] and it solves the discontinuities problem, specifically, the rotation can be expressed as a 6-dimensional vector

$$\mathbf{R}_{6d} = [r_1|r_2]$$

the rotation matrix $\mathbf{R} = [R_1|R_2|R_3]$ can be computed as:

$$\begin{cases} \mathbf{R}_1 = \phi(r_1) \\ \mathbf{R}_2 = \phi(R_1 \times r_2) \\ \mathbf{R}_3 = R_3 \times R_1 \end{cases}$$

where $\phi(\cdot)$ is the vector normalization operation.

4.3.1 Loss

It is crucial to find the appropriate loss function to regress the 6D pose, in order to achieve optimal results. Essentially, we are attempting to optimize two sub-objectives: the 3D translation and the 3D rotation. There are two different approaches that can be employed. One involves separately supervising translation and rotation, using methods such as the angular distance for rotation and the L_2 distance for translation. The other approach is to couple them together,

resulting in a single expression loss. We have chosen the second option because it is less complex having only one loss function rather than dealing with the sum of two losses and their relative weights. Additionally, since we must modify the loss to suit our particular task, having only one expression is easier to manage.

The chosen coupled loss is called Point Matching Loss, proposed in the article DeepIM [13]. Given the ground truth pose $\mathbf{p} = [\mathbf{R}|\mathbf{t}]$ and the estimated pose $\tilde{\mathbf{p}} = [\tilde{\mathbf{R}}|\tilde{\mathbf{t}}]$ we compute the loss as:

$$L_{\text{pose}}(p, \tilde{p}) = \frac{1}{n} \sum_{i=1}^{n} ||(\mathbf{R}x_i + \mathbf{t}) - (\tilde{\mathbf{R}}x_i + \tilde{\mathbf{t}})||_2$$

where x_i are randomly selected points of the model.

We propose a slight modification to the loss function, where the focus is on improving the initial 6D pose of GDR-Net rather than the absolute values. Specifically, our objective is to distinguish between evaluating the same absolute improvement from a poor initial position versus an already good initial estimate. This is motivated by the fact that improving an already good position is more challenging than doing so from a bad initial position. If we don't make this distinction, a significant portion of the loss magnitude will be composed of objects with a poor initial position, such as heavily occluded objects or outliers. In this case, the network will only attempt to learn to refine their positions without the ability to do so successfully, as our network can only make small modifications to already good initial positions. Therefore, the loss will be computed as follows:

$$L_{\text{pose}}(p, \tilde{p}) = \frac{1}{n} \sum_{i=1}^{n} \frac{||(\mathbf{R}x_i + \mathbf{t}) - (\tilde{\mathbf{R}}x_i + \tilde{\mathbf{t}})||_2}{||(\mathbf{R}_{initial}x_i + \mathbf{t}_{initial}) - (\tilde{\mathbf{R}}x_i + \tilde{\mathbf{t}})||_2}$$

4.3.2 Setup

The network was trained for 150 epochs using the Adam optimizer [11], weight decay of $1e-3$, and a batch size of 4. To adjust the learning rate during training, we used a step decay scheduler, which reduces the learning rate by a factor of 0.5 every 10 epochs starting from $1e-4$ and with a minimum of $1e-5$. Additionally, we implemented early stopping as a means of preventing the model from overfitting with patience equal to 10.

4.3.3 Results

The results of our study, as presented in Table 4.2, indicate that our model outperforms GDR-Net when tested on synthetic images but performs poorly on real images, as we had anticipated. This issue is commonly referred to as the *domain gap* and we have identified the RTN network as the root cause of the problem, as can be seen in Table 4.1. Despite implementing various augmentation techniques to improve the generalization to the real domain of the embeddings, this issue persists. While there are domain adaptation techniques that may potentially resolve this issue, addressing it was not the primary objective of our thesis and was left for future research.

	Synth		Real	
	GDR-Net	Ours	GDR-Net	Ours
AR_{MSPD}	83.76	**84.39**	**87.50**	84.85
AR_{MSSD}	75.66	**76.49**	**66.47**	64.46
AR_{VSD}	73.72	**77.70**	**51.78**	50.24
AR_{D}	77.71	**79.52**	**68.58**	66.51

Table 4.2: Comparison of GDR-Net and Ours on PBR and Real test set

Conclusion

In this work, we have investigated a technique to refine the 6D object pose estimation of multiple objects using a monocular image. Our approach incorporates the context of the scene and the relationship between objects to gain a better understanding of the scene. The idea is to develop a consistent pose evaluation for each object using this additional information. Specifically, starting from the 6D object pose estimated by GDR-Net[23], we have constructed a graph of the scene representing each object with a node and with edges connecting neighboring objects. We designed a network to extract geometric features from the connected object pairs and utilized these features in a graph neural network together with the objects' initial 6D poses to get the refined poses.

We have trained the proposed architecture solely on synthetic images and we have tested it on both synthetic and real images. Our finding indicates that the model produced better results compared to the initial 6D poses when tested on synthetic images, but performed worst on real images. This discrepancy can be attributed to the domain gap problem, where the model is evaluated on data that differ in distribution from the training data, leading to sub-optimal performance. We attempted to solve this issue using data augmentation, but the improvements were only marginal.

In conclusion, our approach looks promising, but further investigation is necessary to confirm its effectiveness. Future work may explore domain adaptation techniques to determine whether the results obtained in the synthetic domain can be replicated in the real domain, as well as train the network on

other datasets that contain both synthetic and real images in the training set.

Bibliography

[1] E. Brachmann. 6D Object Pose Estimation using 3D Object Coordinates [Data]. Version V1, 2020. DOI: 10.11588/data/V4MUMX. URL: https://doi.org/10.11588/data/V4MUMX.

[2] Y. Chen, L. Tai, K. Sun, and M. Li. Monopair: monocular 3d object detection using pairwise spatial relationships, 2020. DOI: 10.48550/ARXIV.2003.00504. URL: https://arxiv.org/abs/2003.00504.

[3] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation, 2014. DOI: 10.48550/ARXIV.1406.1078. URL: https://arxiv.org/abs/1406.1078.

[4] A. Collet, M. Martinez, and S. S. Srinivasa. The moped framework: object recognition and pose estimation for manipulation. *The international journal of robotics research*, 30(10):1284–1306, 2011.

[5] M. Denninger, M. Sundermeyer, D. Winkelbauer, D. Olefir, T. Hodan, Y. Zidan, M. Elbadrawy, M. Knauer, H. Katam, and A. Lodhi. Blenderproc: reducing the reality gap with photorealistic rendering, 2020.

[6] K. Duan, S. Bai, L. Xie, H. Qi, Q. Huang, and Q. Tian. Centernet: keypoint triplets for object detection. *CoRR*, abs/1904.08189, 2019. arXiv: 1904.08189. URL: http://arxiv.org/abs/1904.08189.

[7] Z. Ge, S. Liu, Z. Li, O. Yoshie, and J. Sun. Ota: optimal transport assignment for object detection, 2021. DOI: 10.48550/ARXIV.2103.14259. URL: https://arxiv.org/abs/2103.14259.

[8] Z. Ge, S. Liu, F. Wang, Z. Li, and J. Sun. YOLOX: exceeding YOLO series in 2021. *CoRR*, abs/2107.08430, 2021. arXiv: 2107.08430. URL: https://arxiv.org/abs/2107.08430.

[9] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition, 2015. DOI: 10.48550/ARXIV.1512.03385. URL: https://arxiv.org/abs/1512.03385.

[10] T. Hodaň, F. Michel, E. Brachmann, W. Kehl, A. Glent Buch, D. Kraft, B. Drost, J. Vidal, S. Ihrke, X. Zabulis, C. Sahin, F. Manhardt, F. Tombari, T.-K. Kim, J. Matas, and C. Rother. BOP: benchmark for 6D object pose estimation. *European Conference on Computer Vision (ECCV)*, 2018.

[11] D. P. Kingma and J. Ba. Adam: a method for stochastic optimization, 2014. DOI: 10.48550/ARXIV.1412.6980. URL: https://arxiv.org/abs/1412.6980.

[12] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. *CoRR*, abs/1609.02907, 2016. arXiv: 1609.02907. URL: http://arxiv.org/abs/1609.02907.

[13] Y. Li, G. Wang, X. Ji, Y. Xiang, and D. Fox. DeepIM: deep iterative matching for 6d pose estimation. *International Journal of Computer Vision*, 128(3):657–678, November 2019. DOI: 10.1007/s11263-019-01250-9. URL: https://doi.org/10.1007%2Fs11263-019-01250-9.

[14] X. Liu, R. Zhang, C. Zhang, B. Fu, J. Tang, X. Liang, J. Tang, X. Cheng, Y. Zhang, G. Wang, and X. Ji. Gdrnpp. https://github.com/shanice-l/gdrnpp_bop2022, 2022.

[15] Z. Liu, H. Mao, C. Wu, C. Feichtenhofer, T. Darrell, and S. Xie. A convnet for the 2020s. *CoRR*, abs/2201.03545, 2022. arXiv: 2201.03545. URL: https://arxiv.org/abs/2201.03545.

[16] E. Marchand, H. Uchiyama, and F. Spindler. Pose estimation for augmented reality: a hands-on survey. *IEEE transactions on visualization and computer graphics*, 22(12):2633–2651, 2015.

[17] K. O'Shea and R. Nash. An introduction to convolutional neural networks. *arXiv preprint arXiv:1511.08458*, 2015.

[18] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: an imperative style, high-performance deep learning library, 2019. DOI: 10.48550/ARXIV.1912.01703. URL: https://arxiv.org/abs/1912.01703.

[19] J. Redmon and A. Farhadi. Yolov3: an incremental improvement. *CoRR*, abs/1804.02767, 2018. arXiv: 1804.02767. URL: http://arxiv.org/abs/1804.02767.

[20] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: towards real-time object detection with region proposal networks, 2015. DOI: 10.48550/ARXIV.1506.01497. URL: https://arxiv.org/abs/1506.01497.

[21] Z. Tian, C. Shen, H. Chen, and T. He. FCOS: fully convolutional one-stage object detection. *CoRR*, abs/1904.01355, 2019. arXiv: 1904.01355. URL: http://arxiv.org/abs/1904.01355.

[22] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio. Graph attention networks, 2017. DOI: 10.48550/ARXIV.1710.10903. URL: https://arxiv.org/abs/1710.10903.

[23] G. Wang, F. Manhardt, F. Tombari, and X. Ji. Gdr-net: geometry-guided direct regression network for monocular 6d object pose estimation. *CoRR*, abs/2102.12145, 2021. arXiv: 2102.12145. URL: https://arxiv.org/abs/2102.12145.

[24] D. Wu, Z. Zhuang, C. Xiang, W. Zou, and X. Li. 6d-vnet: end-to-end 6-dof vehicle pose estimation from monocular rgb images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 0–0, 2019.

[25] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox. Posecnn: a convolutional neural network for 6d object pose estimation in cluttered scenes, 2017. DOI: 10.48550/ARXIV.1711.00199. URL: https://arxiv.org/abs/1711.00199.

[26] J. Xiao, R. Wang, and X. Chen. Holistic pose graph: modeling geometric structure among objects in a scene using graph inference for 3d object prediction:12697–12706, 2021. DOI: 10.1109/ICCV48922.2021.01248.

[27] J. Yang, J. Lu, S. Lee, D. Batra, and D. Parikh. Graph R-CNN for scene graph generation. *CoRR*, abs/1808.00191, 2018. arXiv: 1808.00191. URL: http://arxiv.org/abs/1808.00191.

[28] C. Zhang, Z. Cui, Y. Zhang, B. Zeng, M. Pollefeys, and S. Liu. Holistic 3d scene understanding from a single image with implicit representation. *CoRR*, abs/2103.06422, 2021. arXiv: 2103.06422. URL: https://arxiv.org/abs/2103.06422.

[29] Y. Zhou, C. Barnes, J. Lu, J. Yang, and H. Li. On the continuity of rotation representations in neural networks, 2018. DOI: 10.48550/ARXIV.1812.07035. URL: https://arxiv.org/abs/1812.07035.

Acknowledgements

This thesis is the result of the period I spent at the Technical University of Munich (TUM) for the thesis abroad program.

I am deeply grateful to Prof. Samuele Salti and Prof. Federico Tombari for providing me with this opportunity, and for their support and guidance.

A special thanks go to Dr. Fabian Manhardt and Dr. Yan Di for their mentorship and feedback that shaped this work.

I would also like to express my sincere gratitude to the CAMP research group, who welcomed me and provided resources and support to make the most of my stay.

I am forever grateful to my parents, friends, and girlfriend for their invaluable support and encouragement throughout this journey.