Deep Learning for Object Relationships: Applications to Road Safety and Bin Picking

by

Auguste Lawrence Whelan Koh

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Systems Design Engineering

Waterloo, Ontario, Canada, 2024

Author's Declaration

This thesis consists of material all of which I authored or co-authored: see Statement of Contributions included in the thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Statement of Contributions

The bin-picking project on which I worked, treated in Chapter 3, was started as part of the Federated Learning for Robot Picking (FLAIROP) project and was done under the supervision of Professor Paul Fieguth and the mentorship of Research Assistant Professor Yuhao Chen.

Elements of the work treated in an extended abstract [1] submitted for the 8th Annual Conference on Vision and Imaging Systems (CVIS 2022) and treated in an associated poster session [2] are presented in Chapter 4 of this thesis. The co-authors of the aforementioned extended abstract and poster are myself, Jinman Park, and Paul Fieguth. The work treated in Chapter 4 was performed under the supervision of Professor Paul Fieguth and was done as part of a collaboration with Miovision Technologies Incorporated.

Abstract

Estimating the relationships between objects is fundamental to certain problems that require an understanding of a scene captured by a camera. This object-relationship theme is explored in two contexts in this thesis: (i) the task of identifying relative placements of objects for bin picking in potential clutter and (ii) the task of estimating distances between vehicles in 3D given some wide-angle video.

Bin picking generally refers to the task of picking up an object in a bin with a robotic arm, given some measurement of the scene. This problem can have a number of challenges and difficulties, one of which being the potential for the objects in the scene to be piled up or in a clutter. If the target object to manipulate is partially occluded by other objects in the scene, there can be some difficulty not only in terms of *detecting* the object, but also in terms of deciding how to clear the way to this target object or how to grasp and lift it appropriately without damaging or excessively displacing the neighbouring objects. Herein is presented a deep-learning module to help deal with potentially cluttered scenes: To account for neighbouring objects when estimating the relationship between two objects, a *graph-network* architecture was designed and implemented. This architecture relies on the bounding boxes and feature maps that would be outputted by an upstream detector to estimate the relationships for pairs of detected objects. The starting edge and vertex attributes of this proposed graph-network architecture are bounding box coordinates (or values derived from such coordinates) and feature-map crops. In addition to this architecture, some definitions for precision and recall that are tailored to this problem are proposed for comparing a ground-truth graph to a predicted graph. Finally, the proposed architecture was evaluated against a baseline model using existing datasets: one containing computer-rendered images, and one with real images.

The problem of estimating distances between vehicles is motivated by the more general problem of estimating the risk of accidents at any given intersection or road segment. The number of traffic accidents per year in Canada, albeit generally decreasing, is still substantial. Estimating the risk of accident at any given traffic intersection or road segment could provide insight and actionable information to municipalities to help determine which intersection or road segment should be prioritized and potentially improved in order to increase road safety. To estimate this risk of accidents, tracking the number of *close calls* or *near misses* is more desirable than merely tracking the number of accidents, as it does not require the observer to wait for accidents to occur, and close calls are presumably much more frequent than actual accidents. In order to determine whether a close call has occurred, one could simply refer to the distance between any two given vehicles; although this is not a perfect metric for detecting close calls, it is a starting point and a metric

that is simple and easy to interpret. As such, this thesis addresses the more specific problem of estimating distances between any two detected vehicles from wide-angle videos. The *wide-angle* nature of images or videos introduces a difficulty, as it challenges a core assumption of normal convolutional neural networks—that of *translational equivariance*. A size-estimation model which uses spherical convolutions was evaluated on a simple, artificial dataset, and results showed that the use of spherical convolutions, as opposed to normal planar convolutions, was able to offer better performance in the tested scenario. In addition to this work, a deep-learning module to estimate distances between vehicles, given some bounding box coordinates and an image, is proposed. An ablation study was performed on this distance-estimating architecture, the results of which quantified the amount of performance gain that could be attributed to the use of pixel information in addition to bounding-box coordinates.

Acknowledgements

I would first like to thank my MASc advisor, Professor Paul Fieguth, for his mentorship and support during this thesis work. In addition, I would like to thank Research Assistant Professor Yuhao Chen for his guidance and help on the bin picking project.

I thank Jinman (Eddie) Park and Miovision Technologies Incorporated for their collaboration and help on the traffic safety project. I also acknowledge the financial support of Miovision Technologies Incorporated, NSERC-Alliance, and OCI-VIP II.

I thank Emily Zhixuan Zeng for the help that she has given me toward using the MetaGraspNet datasets [3, 4] for the bin picking project.

Finally, I am grateful for all the support that I have received from family and friends throughout my studies.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Since the rise in popularity of deep learning [5–7], object detection has met a great deal of success through substantial improvements in detection accuracy. From R-CNN [8] to CenterNet [9] and DETR [10], deep learning methods have repeatedly surpassed classical methods in terms of object detection accuracy [11]. A derived field [12] which directly relies on the ability to detect objects is the prediction of *relationships between* detected objects in a scene; this task is generally referred to as *scene graph generation* [12–14]. This field is important in applications where there is an interest in understanding the general structure of a scene rather than characteristics of each object on its own. This thesis will focus on the applications of deep learning for predicting relationships between objects for two applications: bin picking and traffic road safety.

1.1 Motivations and Problem Descriptions

Motivations and brief problem statements for two problems—bin picking in clutter and detecting close calls in road traffic—will be given in the two subsections that follow. The problem formulations will become more precisely defined in Chapter 3 and Chapter 4.

1.1.1 Bin Picking in Clutter

Bin picking [15–19] is the task of grasping, with a robotic arm, objects placed in a bin; a description of this problem is provided in the background in Section 2.1. In the presence of clutter, some objects may be occluded from the perspective of the camera. The presence

of such occluding objects can not only make object detection more challenging, but can also lead to challenges or shortcomings with respect to the grasping process if the relative placements of the objects neighbouring the targeted object are not accounted for. Three such issues are described below.

Collision between the arm and non-targeted objects: Upon attempting to grasp a targeted object, the robotic arm may come into physical contact with neighbouring objects in the scene. This could cause the grasp attempt to fail, as the non-targeted objects could either prevent the robotic arm from placing the end effector in the correct position or, when using a gripper, a non-targeted object could prevent the end effector from closing on and grasping the object effectively.

Accidentally lifting non-targeted objects: Assuming that a grasp attempt was successful and that the end effector has securely gripped the targeted object: Upon lifting the targeted object, the lifting motion of the arm may partly lift some neighbouring objects that were physically supported by the targeted object. Inadvertently moving such neighbouring objects may be undesirable for two reasons: first, after being inadvertently moved, these other objects in the scene might not return to their original positions, and the model might require a new picture of the scene to be taken in order to detect, once again, the remaining objects; and second, some neighbouring objects could be lifted to such an extent as to fall out of the bin or out of the field of view of the camera, which could damage these objects or make them not visible to the system.

Lower overall speed due to unnecessary grasps: Even if the grasping system is aware that the targeted object is occluded, without knowledge of exactly which objects are obstructing the way to the targeted object, clearing the scene in order to access the non-targeted object might be done in a suboptimal sequence of grasps by unnecessarily removing some objects from the bin.

Together, these issues call for a bin-picking model that is able to reason about the relative placements of objects in a scene, that is able to understand which objects might get moved as a consequence of a targeted object being lifted, and that can find a suitable order in which to remove objects which are obstructing the targeted object.* In particular, a part of this thesis focuses on a proposed deep-learning model that can learn to understand which objects are subject to being displaced when the targeted object is picked up.

*Note that once a neighbouring object is identified as one that must be picked up before the targeted object, this neighbouring object itself temporarily becomes the targeted object, and it might also be occluded by other objects; there is thus a recursive nature to this problem.

The scope of this problem addressed in this thesis is the following: *Given an image of a cluttered set of objects in a bin or on a flat surface, for any given object in the scene (the targeted object), design a model to predict which detected objects must be moved before grasping and lifting the targeted object.*

1.1.2 Detecting Close Calls in Road Traffic

Although there has been a general trend of decrease in the number of collisions resulting in injury or death in the past two decades in Canada (156 415 reported for 2002, 124 682 reported for 2012, and 79 563 reported for 2021 [20]), the number of total injuries in Canada is still substantial: 108 018 injuries (excluding deaths) caused by motor vehicle collisions were reported for 2021 [20]. As such, there is an interest in making roads safer by first identifying the main causes of accidents and any road segments or intersections that are particularly dangerous to drivers, cyclists, and pedestrians. This can be done by referring to the number of reported accidents for any area of interest.

However, although the number of accidents per year is substantial, the number of road segments and intersections, especially in large cities, is considerable. Thus, the number of accidents per year *per intersection* can be relatively low for some intersections, especially for those that are less frequently used. Furthermore, merely referring to accident statistics to determine the level of risk of a certain road segment or intersection means that it is impossible to reliably identify high-risk areas or find recurring causes of accidents *before* any accidents occur! As such, there is motivation to instead refer to some other metrics that can be collected before accidents ever happen. Close calls (or near misses) that occur between a vehicle and a pedestrian, cyclist, or other vehicle are presumably more common than actual accidents* and strongly correlated to the risk of accidents. Thus, referring to the prevalence of close calls seems, *a priori*, to be more desirable than only tracking the prevalence of actual accidents when assessing the degree of risk of a given area.

Miovision Technologies Incorporated, a company which has collaborated on this project, offers products for traffic monitoring and data analysis. In order to monitor traffic, wide-angle monocular RGB cameras have been installed on tall structures at a number of intersections in such a way that they are placed above pedestrians, cyclists, and typical

*Even if close calls do not occur much more often than accidents, they supplement the accidents data, and they should allow a faster identification of those road segments and intersections that are most dangerous.

vehicles. Consequently, assumptions about the data being collected and the equipment used for this road-safety project have been based on the real-world circumstances and constraints under which data is collected by Miovision.

The high-level objective of this project is to identify close calls and near misses based on RGB footage produced by these fixed wide-angle cameras. In this thesis, this project is focused on estimating distances between two vehicles, which is a simple and easy-to-interpret indicator of whether a close call took place between those two vehicles. The problem formulation of this project in this thesis is thus the following: *Given an RGB video from a wide-angle, monocular, fixed camera of known specifications, design a model to estimate the 3D distance between any two vehicles that are simultaneously visible in this video.*

1.2 Contributions

Generally, the objectives for this thesis are to provide, for each of the two main problems, insight into limitations of past methods, and why some proposed methods should be well-suited to each problem. The main contributions of this thesis, in the order that they are presented, are as follows:

1. A graph-network-based architecture for predicting relative object placements in a scene.
2. Formalization of performance metrics for object relationship prediction models in the context of bin picking, as rigorous metric definitions suited to this problem appear to be lacking in the related literature.
3. An evaluation of the suitability of spherical convolutions for 3D distance estimation in wide-angle images.
4. A model to estimate distance between vehicles on a fixed plane, based on the output of a detector.

The remainder of the thesis contains four chapters. Chapter 2 provides some background theory on four major topics: the bin picking problem, object detection using deep learning, graph networks, and spherical convolutional networks. The background content on the bin picking problem, object detection, and graph networks includes information germane to the problem of bin picking in clutter introduced in Section 1.1.1, while the background on

spherical convolutional networks and the background on object detection are relevant to the distance-estimation problem introduced in Section 1.1.2. Chapter 3 further formulates the problem of bin picking in clutter described in Section 1.1.1, and it presents a proposed model along with some performance results. Chapter 4 focuses on the distance-estimation problem introduced in Section 1.1.2 by providing motivation for the use of spherical convolutions and by presenting a module architecture, designed to be placed downstream of a detector, for estimating distances between detected vehicles. Finally, Chapter 5 concludes both Chapter 3 and Chapter 4 and includes suggestions for any continuations of these projects.

Chapter 2

Background

Before delving into each of the two main computer vision problems of this thesis, some background concepts are given in this current chapter in order to provide context to the reader. Some fields and methods used for this thesis work already have a large presence in the literature, and before starting to use any of them, the reader might benefit from first getting a summary for each, some relevant definitions, as well as an introduction to the notation that will later be used in some subsequent chapters.

2.1 The Bin Picking Problem

Bin picking [15–19] generally refers to the task, for a robotic system, of picking up an object in a scene with a robotic arm; as illustrated in Figure 2.1, the robotic system typically includes one or more imaging sensors (such as an RGB camera and/or depth camera) that provide measurements of the scene. Thus, the problem of robotic object grasping is a compound one, which can include the following sub-tasks (depending on what type of approach is used): scene analysis, object detection, pose estimation, and grasp and path planning (how the robotic arm should be placed on the object) [15, p. 3].

While a large portion of object-grasping approaches rely on estimating the poses of objects and using geometric models (e.g., CAD models of the objects to manipulate), there has recently been an increase in popularity in data-driven approaches which do not rely on predetermined geometric models of the objects to manipulate [16, 21], which offers flexibility. Furthermore, model-free approaches have been reported to allow for generalization to objects not previously seen by the grasping system [21, 22].

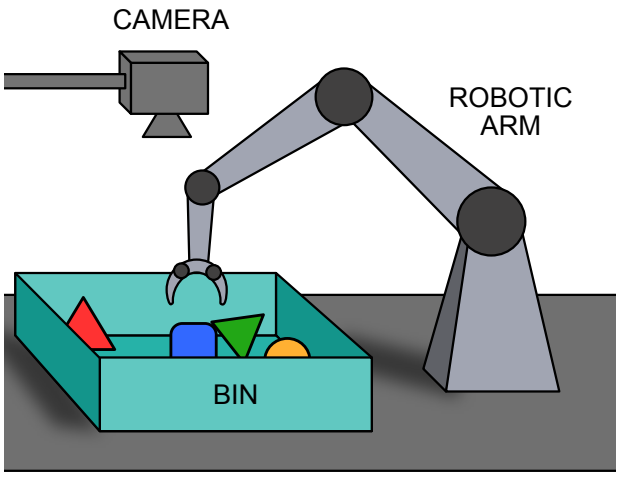


Figure 2.1: Illustration of a bin-picking system comprising of a robotic arm and a camera pointed towards the bin. At this point in the process, the camera has already captured an image, an object has been detected, a path has been planned, and the robotic arm is in the process of positioning the end effector around an object to grasp it.

In addition to the possibility of dealing with objects without knowing their geometric models, or dealing with novel objects, bin picking and object grasping can include additional challenges: The objects to manipulate may be shiny, transparent, or translucent, which introduces difficulty at the perception level. Challenges can also be present at the object manipulation level: Objects may be fragile or shaped in a way which greatly restrict the range of acceptable ways in which to grasp the object, or multiple objects to manipulate may be piled up or in a clutter. This last difficulty motivates, and is partly addressed in, Chapter 3.

The challenge of dealing with clutter in bin picking has a history which dates at least as far back as 1983 [23]. This work by K. Ikeuchi et al. [23] presented a system which segmented the scene based on photometric stereo measurements, estimated the attitude and position of the object to grasp, and determined where to grasp the object. As summarized in [24], later works related to this challenge of dealing with clutter have investigated different

approaches, ranging from using supervised machine learning to learn how to interact with the model [25, 26], to physically interacting with the pile of objects in order to either probe or change the arrangements of the objects [27–29].

Robotic object grasping is generally a broad and fundamental problem in autonomous robotics, and there are multiple applications that motivate research in this field [30]: these include anthropomorphic service robots for household scenarios [18], picking objects in industrial, automotive settings [31], and warehouse automation [32].

2.2 Object Detection Using Deep Learning

In computer vision, the task of an object-detection model (or *detector*) consists in *locating* and *classifying* some physical *objects* that appear in an RGB *image or video* (possibly including pixel-wise depth information, D) [11, 33, 34]. For greater clarity and precision, a description for each of the main concepts in the preceding statement is given:

- *Objects*: A (physical) object is to be interpreted in the usual meaning of the word, including animate objects: e.g., a banana, a car, a pair of scissors, etc.
- *Images or videos*: Using a notation and mathematical framework similar to that described in [35], let an image of c channels ($c = 3$ for an RGB image, and $c = 4$ for an RGB-D image) be mathematically described as a function which maps from a 2D integer lattice (set of discrete coordinates), $\mathbb{Z}^2$, to a set of three-dimensional real-valued vectors, the elements of which are all non-negative and no greater than 1:

  $$\mathbb{Z}^2 \to \left\{ v \in \mathbb{R}^c \;\middle|\; 0 \leq \min(v) \wedge \max(v) \leq 1 \right\}.$$

  This set of coordinates $\mathbb{Z}^2$ is for specifying two spatial dimensions: a horizontal direction $\hat{\jmath}$ and a vertical direction $\hat{k}$. This definition of an image can be extended to also encompass videos (lists of images); this results in a function with the same target set, but a different domain: $\mathbb{Z}^3$. Thus, a video is such a function where the domain is a set of discrete coordinates, each specifying a horizontal position (along $\hat{\jmath}$), a vertical position (along $\hat{k}$), and a temporal position (along $\hat{t}$).
- *Model*: An object detection model is the *implementation* of some function $\mathscr{D}$ which maps from the set of all images (or videos), $\mathcal{M}$, to the set of all multisets of pairs in which the first element is a bounding box and the second element is a class:

  $$\mathscr{D} \colon \mathcal{M} \to \mathcal{P}'(\mathcal{B} \times \mathcal{C}),$$

where $\mathcal{P}'(\square)$ is the set of all multisets which have a subset of $\square$ as their underlying set.* The wording "implementation" is used here because a model is not merely a function, since the general architecture and algorithm used for making an inference are defining characteristics of the model. In contrast, a function can be entirely described as simply a "set of ordered pairs" [36, p. 42].

- *Locating*: Expressing the location of an object is usually carried out by placing *bounding boxes* (rectangles) on the 2D planes of the images or video frames. The set of all possible 2D bounding boxes is denoted by $\mathcal{B}$:

$$\mathcal{B} \coloneqq \left\{ \begin{bmatrix} b_{1,1} & b_{1,2} \\ b_{2,1} & b_{2,2} \end{bmatrix} \in \mathbb{R}^{2\times 2} \;\middle|\; b_{1,1} < b_{1,2} \wedge b_{2,1} < b_{2,2} \right\},$$

  where $[b_{1,1} \;\; b_{2,1}]^{\mathrm{T}}$ is the bottom-left corner vertex of the bounding box and $[b_{1,2} \;\; b_{2,2}]^{\mathrm{T}}$ is the top-right corner vertex.

- *Classifying*: In this context, classification is the process of assigning some categorical label, i.e., a *class*, out of all classes $\mathcal{C} = \{c_1, c_2, \ldots, c_k\}$ considered, to each of the estimated locations (bounding boxes). Generic classes might include, for example, "person", "chair", "fruit", "vehicle", etc. [37]; the specific classes would depend on the dataset and the application for which the detection algorithm is used. The definition of a class itself can be approached in a variety of ways, but it is reasonable to restrict its definition to only depend on properties that can have a direct effect on the measured image or video (e.g., shape, colour, degree of scattering, degree of absorption, etc.).

The field of object detection has existed since at least the 1990s [11, 33], but the use of deep learning for this purpose has become particularly popular [11] following the introduction [38] of AlexNet [39] in 2012. The year 2014 marks the start of a period of widespread use of deep-learning methods for object detection [11, 33, 34], with the introduction of R-CNN [8] and, in 2015, the introduction [40] of YOLO [41]. Around this time, a distinction between two categories of such models becomes apparent: "two-stage detectors" and "one-stage detectors" [11]. A two-stage detector, such as Faster R-CNN [42], includes a region proposal stage, $\mathscr{D}_{\mathrm{RP}} \colon \mathcal{M} \to \mathcal{P}'(\mathcal{B})$, which generates a large number of proposed regions of interest (RoIs) [43]. Each proposed RoI is then inputted into the

*The idea of *multiset* is used here instead of simply *set* because it could technically be possible for a detector to output two bounding boxes with the same coordinates, and with the same classes assigned to both of them, indicating that two objects of the same class have the same location on the image (e.g., two trees, one behind the other, that have the same height, width, and location on the image).

second stage, $\mathscr{D}_{\mathrm{H}} \colon \mathcal{P}'(\mathcal{B}) \to \mathcal{P}'(\mathcal{B} \times \mathcal{C})$, which assigns a class to each proposal and refines the bounding-box coordinates [42, 43]. A two-stage detector is thus an algorithm for a function of the form $\mathscr{D}_{\mathrm{H}} \circ \mathscr{D}_{\mathrm{RP}}$, with distinct implementations (algorithm sections) for each of $\mathscr{D}_{\mathrm{RP}}$ and $\mathscr{D}_{\mathrm{H}}$. One-stage detectors, such as YOLO [41], produce both bounding-box coordinates and assign a class to each bounding box in a single stage, without intermediate proposals [43]. Thus, a one-stage detector is not composed of distinct algorithms for $\mathscr{D}_{\mathrm{RP}}$ and $\mathscr{D}_{\mathrm{H}}$.

Subsections 2.2.1 and 2.2.2 gives a brief overview of these two categories of detectors, with a greater focus on the two-stage approaches, as this is the category of deep-learning models that were used in the main studies for this thesis.

2.2.1 Two-Stage Approaches

The two stages involved in these approaches are the following, in the order that the processing is performed [33]:

1. **A region proposal stage**, which proposes many bounding boxes in the image regardless of the apparent classes of the objects that are detected—this stage is *class agnostic*.

2. **A prediction stage** which gives a *classification* output [33], assigning an object class to each region proposal. This stage may also refine the bounding-box coordinates of the selected region proposals from the first stage [33].

Region Proposal Stage

Fundamentally, the purpose of a region proposal stage is to produce a large number of bounding boxes as hypotheses for object locations. As such, an adequate region proposal stage should have high *recall* and make proposals that are *class agnostic*.

The region proposal stage can either be implemented using classical methods, such as selective search [8, 44] and multiscale combinatorial grouping [45], or using deep-learning approaches [42, 46, 47]. An advantage of using deep-learning for the region proposal stage is that it allows for end-to-end training with the weights of the prediction stage [42].

The Region Proposal Network (RPN) introduced in [42], as part of Faster R-CNN, is the first such instance of a deep-learning approach to generating region proposal. The processing steps of this RPN are the following, based on [42]:

1. The image is first fed into a convolutional feature extractor which outputs a feature map. (This feature map can then be re-used by the prediction stage of the detector.)

2. A small fully-connected network is applied in a sliding-window fashion along the two spatial dimensions of the output feature map—this is effectively a convolutional layer. Spatial overlap between windows is permitted.

3. To each window is assigned a set of predefined reference boxes, named *anchors*. These anchors include a variety of predefined scales and aspect ratios.

4. For each anchor, two predictions are performed: (a) the probability that the anchor corresponds to an object, and (b) regression in each of the four bounding-box parameter dimensions, corresponding to the amount of offset by which to adjust the anchor coordinates to fit the bounds of object that has been detected.

The number of resulting proposals can then be adjusted by selecting a threshold for the probability scores of step 4(a).

Prediction Stage

Given the output of the region proposal stage and the original image (or feature map from step 1 of the RPN), the role of the prediction stage is to further refine the bounding-box coordinates and to assign to each bounding box an object class [48]. This stage typically starts by producing a crop of the image or feature map for each bounding box received, and then performs pooling on each crop [48]. In the case of [48], the pooled features-map crop is then fed into some fully-connected layers, which then branch off to a classification module and a bounding-box regression module.

2.2.2 One-Stage Approaches

The defining characteristic of one-stage detectors is that they do not rely on using bounding-box proposals that were dynamically created by an upstream module; instead, object classification (specifying the type of object) is performed without directly using the predicted bounding-box coordinates [43]. These detectors often work in a grid-search-like fashion, with the image being processed at different locations [49] (and sometimes in combination with different scales and aspect ratios; e.g., [50]).

One of the main appeals of one-stage detectors is their inference speed, which is often higher than two-stage detectors (this depends on the number of proposals to process), albeit at the cost of prediction accuracy [41].

Popular one-stage detectors include the following [11]: YOLO [41] (and various derivatives with improvements in inference speed and accuracy; e.g., YOLOv2 [51], YOLOv3 [52], YOLOv4 [53], and YOLOv7 [54]), Single Shot Multibox Detector [50], RetinaNet [49], CornerNet [55], and CenterNet [9].

Since the two main projects covered by this thesis do not require real-time inference, the two-stage class of detectors was selected.

2.3 Graph Networks

Graph networks [56–59], a generalization of graph neural networks [60–64], are models that process data that has a graph structure [56]. Any graph can be defined by a set of vertices $\mathcal{V}$ and a set of edges $\mathcal{E}$ which is a subset of $\mathcal{V}^2$. As such, a graph $\mathcal{G}$ can simply be defined as $(\mathcal{V}, \mathcal{E})$. In the context of graph networks, there is also the concept of an *attribute*, which is defined per graph-processing iteration and is associated to each vertex, edge, and the graph as a whole [56]. For a state $i \in \mathbb{N}$ and a graph $\mathcal{G}$, this concept of *attributes* can be represented by a $\mathcal{V} \cup \mathcal{E} \cup \{\mathcal{G}\} \to \cup_{n=1}^{\infty} \mathbb{R}^n$ function $\mathbf{f}_{\mathcal{G}}^{i}$ which maps any vertex or edge of the graph $\mathcal{G}$ (or maps $\mathcal{G}$ itself) to a feature vector of some dimensionality. (As an abuse of notation, the more succinct form $\mathbf{f}^i$ will frequently be used, especially when there is no ambiguity as to which graph this function corresponds to.) A graph network processes and updates all of these attributes in iterations, each of which has its update performed by a dedicated "GN block" [56] (a processing & update block). A GN block responsible for processing a graph state i (and producing state $i+1$) is thus a function $\mathbf{f}^i \mapsto \mathbf{f}^{i+1}$, which processes all the attributes of the edges, the vertices, and the graph. A GN block is composed of six fundamental functions [56]:

1. The edge-update function: ϕ_{e}
2. The vertex-update function: ϕ_{v}
3. The global-update function: ϕ_{g}
4. The local edge-aggregation function: $\rho_{\mathrm{e}\to\mathrm{v}}$
5. The global edge-aggregation function: $\rho_{\mathrm{e}\to\mathrm{g}}$

6. The vertex-aggregation function: $\rho_{\mathrm{v}\to\mathrm{g}}$

Any of the three update functions could be implemented as artificial neural networks, while the three aggregation functions would need to be able to accept a variable input size (but they could still include some processing component implemented using some artificial neural network) [56].

The output of the $\rho_{\mathrm{e}\to\mathrm{v}}$ aggregation function for an input $\mathbf{f}^i(v_t)$ (for some vertex v_t) is independent of the attributes of the edges that do not have vertex v_t as their heads; i.e., if $\{\!\{\square_1, \square_2, \dots\}\!\}$ denotes a *multiset* with elements $\square_1, \square_2, \dots$, then $\mathbf{f}^{i+1}(v_t)$ is independent of $\{\!\{\mathbf{f}^i(v_1, v_2) \mid v_1 \in \mathcal{V} \wedge v_2 \neq v_t\}\!\}$. Since $\rho_{\mathrm{e}\to\mathrm{v}}$ operates on multisets of edge attributes, it is convenient to define a new function to refer to the multiset of attributes of a set of edges: Let $\mathbf{F}^i_{\mathcal{G}}$ be a function analogous to $\mathbf{f}^i_{\mathcal{G}}$ that can be applied to any subset of the domain of $\mathbf{f}^i_{\mathcal{G}}$ (and let $\mathbf{F}^i$ be a more succinct form); i.e.,

$$\mathbf{F}^i_{\mathcal{G}} \colon \mathcal{P}\big(\mathcal{V} \cup \mathcal{E} \cup \{\mathcal{G}\}\big) \to \cup_{n=1}^{\infty} \mathcal{P}'(\mathbb{R}^n)$$

$$S \mapsto \{\!\{\mathbf{f}^i_{\mathcal{G}}(s) \mid s \in S\}\!\},$$

where $\mathcal{P}(\square)$ is the power set of $\square$ and $\mathcal{P}'$ is the multiset counterpart (defined on page 9). Further, let $\mathfrak{e}_{\to}$ be the function which, for an input vertex v_t, outputs the set of edges that have vertex v_t as their heads:

$$\mathfrak{e}_{\to} \colon \mathcal{V} \to \mathcal{P}(\mathcal{E})$$

$$v_t \mapsto \big\{(v_1, v_2) \;\big|\; v_1 \in \mathcal{V} \wedge v_2 = v_t\big\}.$$

At last, with these notations defined and using the framework from [56], for any given edge $e \coloneqq (v_1, v_2)$ and $\forall i \in \mathbb{N}$, the updated attributes are defined in a recursive fashion and can be expressed using the six fundamental functions:

$$\mathbf{f}^{i+1}(e) = \phi_{\mathrm{e}}\Big(\mathbf{f}^i(e),\ \mathbf{f}^i(v_1),\ \mathbf{f}^i(v_2),\ \mathbf{f}^i(\mathcal{G})\Big)$$

$$\mathbf{f}^{i+1}(v) = \phi_{\mathrm{v}}\Big(\rho_{\mathrm{e}\to\mathrm{v}} \circ \mathbf{F}^{i+1} \circ \mathfrak{e}_{\to}(v),\ \mathbf{f}^i(v),\ \mathbf{f}^i(\mathcal{G})\Big)$$

$$\mathbf{f}^{i+1}(\mathcal{G}) = \phi_{\mathrm{g}}\Big(\rho_{\mathrm{e}\to\mathrm{g}} \circ \mathbf{F}^{i+1}(\mathcal{E}),\ \rho_{\mathrm{v}\to\mathrm{g}} \circ \mathbf{F}^{i+1}(\mathcal{V}),\ \mathbf{f}^i(\mathcal{G})\Big).$$

One notable aspect of updating attributes in this fashion is that after n iterations, information from the attribute of any vertex v_a can have an effect on the attributes of all edges and vertices along all *walks* of length n [56]. Conversely, if the shortest path from a

node v_a and a node v_b is of length $n+1$, then it is guaranteed that after only n (or fewer) iterations, the attribute of v_a will have neither affected that of v_b and nor that of any edge which has v_b as its head.

Graph-network or graph-network-like models have been previously applied on tasks such as molecular design [59], road traffic forecasting [65, 66], and visual scene understanding [67, 68]. Such networks are generally suitable when the data on which to make a prediction has the structure of a graph, but where the topology of the graph is not constant across the dataset. This idea of variable topology is relevant to both problems addressed in Chapter 3 and Chapter 4: indeed, in the case of the bin-picking problem, the number of *objects* in a bin can vary across images, and in the case of the close-call detection problem, the number of *vehicles, pedestrians, and cyclists* in a scene can vary across videos and even frames.

2.4 Spherical Convolutional Neural Networks

Spherical convolutional neural networks (S^2-CNNs) [69] are convolutional neural networks (CNNs) [70] which use spherical convolutional layers instead of regular 2-dimensional (planar) convolutional layers that operate on a 2-dimensional Euclidean space $\mathbb{E}^2$. Thus, an image processed by an S^2-CNN is assumed to lie on a sphere; in contrast to a typical (planar) image where to each point in $\mathbb{R}^2$ is associated an RGB value (a triplet), a spherical image can be described as an association of each point on a 2-dimensional sphere S^2 with an RGB value. Formally, and as a generalization of definitions of *image* given in [35, 69], a k-channel image (k-dimensional signal) on a space $\mathbb{S}$ can be described as an $\mathbb{S} \to \mathbb{R}^k$ function, where an input is the coordinates of a point on the image, and the output is the pixel value at that point. As such, a continuous-domain RGB image on $\mathbb{E}^2$ can be described as an $\mathbb{R}^2 \to \mathbb{R}^3$ function (or a restriction of this function, as applicable), with its discrete-domain counterpart being described as an $\mathbb{Z}^2 \to \mathbb{R}^3$ function [35]. Similarly, a continuous-domain RGB image on a 2-sphere S^2 can be described as an $S^2 \to \mathbb{R}^3$ function [69].

The *convolution* operator "$*$" is an $\mathbb{S}^{(\mathbb{R}^k)} \times \mathbb{S}^{(\mathbb{R}^k)} \to \mathbb{S}^{\mathbb{R}}$ function. When $\mathbb{S} = \mathbb{R}^2$, the convolution operation between an image f and a kernel κ is defined as follows [71, p. 95]:

$$(\kappa * f)(x, y) \coloneqq \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(t_x, t_y) \cdot \kappa(x - t_x, y - t_y) \, \mathrm{d}t_x \, \mathrm{d}t_y \, .$$

Using a vector notation, the following definition is equivalent:

$$(\kappa * f)(\vec{x}) = \int_{\mathbb{R}^2} f(\vec{t})^{\mathrm{T}} \kappa(\vec{x} - \vec{t}) \, \mathrm{d}\vec{t} \, . \tag{2.1}$$

Now, what would convolution on the sphere look like? Over what space would integration be done, and how would one deal with the $\vec{x} - \vec{t}$ component, which is not applicable to coordinates on a sphere? In order to choose a definition which is adequate, it is essential to first identify what properties of the convolution operator should be maintained. The convolution operator has a number of properties [35, 69] that have contributed to its success in deep learning for computer vision:

1. It is equivariant with respect to translation. More rigorously, let $\theta_\kappa \colon f \mapsto \kappa * f$ for any kernel $\kappa \in (\mathbb{R}^2)^{(\mathbb{R}^k)}$. Thus, $\theta_\kappa(f)$ is simply the result of convolving some given kernel κ with the input image f. Such θ_κ functions are equivariant with respect to translation; i.e., if translation is described as a group action τ of the group $T = (\mathbb{R}^2, +)$ on $(\mathbb{R}^2)^{(\mathbb{R}^k)} \cup (\mathbb{R}^2)^{(\mathbb{R})}$ such that $\tau(\vec{t}, f) = (x \mapsto f(x + \vec{t}))$, then the translation equivariance property is as follows:

$$\forall \vec{t} \; \forall \kappa \; \forall f \quad \left(\; \theta_\kappa\Big(\tau\Big(\vec{t}, f\Big)\Big) \;\; = \;\; \tau\Big(\vec{t}, \theta_\kappa(f)\Big) \; \right).$$

   This same idea can be expressed in a commutative diagram [72], clearly showing that applying convolution via θ_κ followed by some translation $\tau_{\vec{t}} \colon f \mapsto \tau(\vec{t}, f)$ is equivalent to applying translation $\tau_{\vec{t}}$ followed by convolution via θ_κ:

$$\begin{array}{ccc} (\mathbb{R}^2)^{(\mathbb{R}^k)} & \xrightarrow{\theta_\kappa} & (\mathbb{R}^2)^{\mathbb{R}} \\ \Big\downarrow{\scriptstyle \tau_{\vec{t}}} & & \Big\downarrow{\scriptstyle \tau_{\vec{t}}} \\ (\mathbb{R}^2)^{(\mathbb{R}^k)} & \xrightarrow{\theta_\kappa} & (\mathbb{R}^2)^{\mathbb{R}} \end{array}$$

2. The input to θ_κ and its output are both signals on the same space, $\mathbb{E}^2$; thus, it is easy to chain convolutional layers by simply using a dedicated θ_κ function for each of the k desired output channels.

3. It is simple to discretize θ_κ, as $\mathbb{E}^2$ (the domain of both kernel κ and feature map f) can be discretized with a tessellation that is symmetric with respect to translation (with square tiles forming a grid)—a *regular* tessellation.

Ideally, for convolution operations on the sphere, the three properties listed above should be satisfied; however, working with images on S^2 comes with some complications which require substantial adjustments to the definition and implementation of regular convolutions. Two such complications are presented below:

(A) **The space of rotations is not isomorphic to S^2** [69]: The $\mathbb{E}^2$ space possesses multiple symmetries; notably, its symmetry group E_2 includes the subgroup T_2 of all *translations* in 2D, which is isomorphic to $\mathbb{E}^2$ itself [73, p. 19]. Practically, this means that a convolution kernel moving over $\mathbb{E}^2$ can do so in a way that will result in a signal on $\mathbb{E}^2$ and that the input and output of a convolution on $\mathbb{E}^2$ are both signals on $\mathbb{E}^2$. This is convenient because such convolution can be composed (chained) with itself without any change in the underlying space on which the signal lies. Further this also allows for translational equivariance. While allowing for self-composition is feasible, preserving translational equivariance is not possible on the sphere, as translation is not part of the symmetry group of the sphere. As rotations of the sphere are analogous to translations on $\mathbb{E}^2$, one could consider defining a spherical convolution operator that is equivariant to rotations on the sphere; however, the symmetry group of the sphere, ignoring reflections, is the three-dimensional group SO(3), which is *not* isomorphic to S^2 (different numbers of dimensions), and the output of spherical convolution of two signals on S^2 is three-dimensional [69]. Although one could conceive of a spherical convolution operator which outputs a two-dimensional signal, this operator would have to be, for each location on the sphere, non-equivariant to a certain rotation (since there would be one fewer rotation dimension)—such as the one in [74], as pointed out by [69].

(B) **Very limited tessellation options**: Since, practically, it is convenient to store an image $f\colon \mathbb{S} \to \mathbb{R}^k$ in memory in an array structure, computation of convolution would typically be done on a discretized version of f. In the case that $\mathbb{S} \coloneqq \mathbb{E}^2$, this poses no major problem; however, in the case of S^2, there is *no non-degenerate regular tessellation* on the sphere other than the five tessellations which are each analogous to one of the five platonic solids [75, pp. 128–130]. The platonic solid with the most faces has 20 faces [75, p. 130], which is clearly too coarse-grained for many computer vision applications. This calls for the need to interpolate values when performing spherical convolution at resolutions of more than 20 pixels over the entire sphere.

T. S. Cohen et al. [69] have proposed two definitions for correlation*—one for signals on S^2 and one for signals on SO(3). Their definitions, after adapting them to be consistent

*For real-valued signals, cross-correlation and convolution are equivalent, up to some reflections of one of the two input functions; in contemporary computer vision, it is common to use the term "convolution"

with the notation used in (2.1), are as follows, for a spherical image (of feature map) f and a spherical kernel κ:

$$\forall R \in \mathrm{SO}(3) \quad \forall \kappa, f \in (S^2)^{(\mathbb{R}^k)} \quad \left(\quad (\kappa * f)(R) \coloneqq \int_{S^2} f(\vec{t})^{\mathrm{T}} \kappa(R^{-1}\vec{t}) \, \mathrm{d}\vec{t} \quad \right), \tag{2.2}$$

$$\forall R \in \mathrm{SO}(3) \quad \forall \kappa, f \in (\mathrm{SO}(3))^{(\mathbb{R}^k)} \quad \left(\quad (\kappa * f)(R) \coloneqq \int_{\mathrm{SO}(3)} f(Q)^{\mathrm{T}} \kappa(R^{-1}Q) \, \mathrm{d}Q \quad \right), \tag{2.3}$$

where $\mathrm{d}Q$ is an integration measure on SO(3) (the definition of which is omitted here) [69]. There are three notable points to emphasize from these definitions [69]:

- The R of (2.2) and (2.3) is a 3-by-3 rotation matrix that is analogous to the $\vec{x}$ in (2.1); an input to the function $\kappa * f$ is a *rotation* matrix. Thus, any kernel placement on the sphere is defined by a rotation, and any pixel of a feature map on SO(3) is positioned at a particular rotation.
- The input signals in (2.2) are on S^2, while the output signal is on SO(3); the input signals in (2.3) and the output signal are both on SO(3). Thus, the convolution of (2.2) cannot be naturally chained with itself, while the convolution of (2.3) can.
- SO(3) is a *three-dimensional* manifold, and thus the space over which the kernel travels in (2.2) and (2.3) is three-dimensional, and the kernel and feature maps in (2.3) are three-dimensional. This can substantially increase the computational complexity of a spherical convolution algorithm [69].

Despite the popularity and relative simplicity of convolutions on $\mathbb{E}^2$, which are contrasted by the practical complications that can arise from spherical convolutions, this latter type of convolution does have its place in some applications where the signal on which to make some inference has a spherical geometry. The concept of spherical convolutions and its limitations will become relevant in Chapter 4, and the use of spherical convolutions will be justified therein, as it will be shown that the images collected by a wide-angle camera are inherently spherical due to the geometry of the measurement system.

---

when actually referring to the cross-correlation operation, such as in the name for CNNs (as mentioned in [69]).

Chapter 3

Bin Picking for Cluttered Scenes

This chapter will more precisely define the bin-picking problem introduced in Chapter 1. It will also outline and discuss some of the main challenges associated with the bin-picking problem in cases of potential clutter in a scene, and it will describe a proposed model architecture for improving detection performance on highly realistic datasets such as the MetaGraspNetV2 dataset [4].

3.1 Problem Formulation

As explained in Section 2.1, the bin picking problem generally refers to the task of using a robotic arm to pick up some targeted object in a bin based on some measured data, such as RGB or RGB-D images. This current section will focus on the problem of interpreting the scene and, in particular, on understanding the relative arrangements of objects in the scene, especially in cases of possible clutter. In situations in which objects are cluttered in a pile, identifying which object to grasp first is not necessarily obvious. Some objects might be physically obstructing the path that the robotic arm might follow to grasp the targeted object, or neighbouring objects might be visually occluding the targeted object. In order to reason about the scene and about how to proceed in cases of clutter, it is important for the bin-picking system to first be able to identify the occlusion or obstruction relationships between the objects in the scene (and thus, their relative placements). A formulation is given below for this identification problem.

For a given image of a scene containing N objects $V \coloneqq \{v_0, v_1, \ldots, v_N\}$ that are visible to a camera, the relationship between objects V can be specified by a function

$$c_{2,\rightarrow} \colon E_{\rightarrow} \rightarrow \mathcal{K}_{2,\rightarrow} \tag{3.1}$$

where

$$E_{\rightarrow} = \{(v_k, v_m) \in V \times V \;\mid\; v_k \neq v_m\} \tag{3.2}$$

are all the possible ordered pairs of distinct objects in the given image and where $\mathcal{K}_{2,\rightarrow} = \{\text{“obstructing”}, \text{“not obstructing”}\}$ are the possible classes for the edges. Notice the subscript “$_{\rightarrow}$” used in variable notation to indicate that a variable corresponds to the directed-edges case. Similarly, the subscript “$_{\leftrightarrow}$” will be used to indicate correspondence the undirected-edges cases. With this formulation,

$$G_{\rightarrow} \coloneqq (V, E_{\rightarrow}) \tag{3.3}$$

is a *directed simple graph* that is *complete*. When an edge (v_k, v_m) is illustrated by an arrow (such as in Figure 3.1), this arrow points *from* v_k *to* v_m. Thus, for any $(v_k, v_m) \in E_{\rightarrow}$, if any part of the object v_k is physically obstructing any part of the object v_m, then $c_{2,\rightarrow}(v_k, v_m) =$ “obstructing”, and if no such obstruction is present, then $c_{2,\rightarrow}(v_k, v_m) =$ “not obstructing”.

Since $\mathcal{K}_{2,\rightarrow}$ only contains two elements (classes), the relationships between any two distinct vertices in V can also be described by the *spanning subgraph*

$$G^s_{\rightarrow} \coloneqq (V, E^s_{\rightarrow}) \tag{3.4}$$

of $G_{\rightarrow}$ where

$$E^s_{\rightarrow} \coloneqq \{(v_k, v_m) \in E_{\rightarrow} \;\mid\; c_{2,\rightarrow}(v_k, v_m) = \text{“obstructing”}\}\,. \tag{3.5}$$

That is, for any two objects v_k and v_m, some part of the object v_k is physically obstructing some part of v_m if and only if the directed edge (v_k, v_m) is in $E^s_{\rightarrow}$.

It is with purpose that these *two* (equivalent) formulations have been introduced above: the formulation of $G_{\rightarrow}$—consisting of (3.1), (3.2), and (3.3)—is considered more suitable when discussing the nature of a relation between any two given vertices, while the formulation of $G^s_{\rightarrow}$—consisting of (3.4) and (3.5)—is more succinct and simpler to represent in the form of a graph diagram. The problem on which this chapter focuses is the design of a model $\mathscr{M}_{G,\rightarrow}$ which, given an RGB or RGB-D image of a scene containing $N \geq 0$ objects, estimates the graph $G^s_{\rightarrow}$ corresponding to that image—or, equivalently, estimates $G_{\rightarrow}$ and

Figure 3.1: An RGB image from a simulation of a set of objects in a scene, with the relationships between objects shown with annotations illustrating $G^s_{\rightarrow}$. Object E is obstructed by object C and obstructed by object A, which is obstructed by object B. The image without annotation is from the MetaGraspNetV2 [4] dataset.

its corresponding $c_{2,\rightarrow}$. This should *not* be considered as simply a (binary) *classification* problem on the directed edges $E_{\rightarrow}$ of the complete graph $G_{\rightarrow}$. Rather, for any given image $\vec{m}$, the model $\mathscr{M}_{G,\rightarrow}$ should *detect* the vertices and *assign a class* to each ordered pair of vertices. The model $\mathscr{M}_{G,\rightarrow}$ can thus be separated into two modules: an object (i.e., vertex) detector $\mathscr{D}_V$ that outputs an estimate $\widehat{V}$ of V and a module $\mathscr{C}_{E,\rightarrow}$ that outputs an estimate $\widehat{c_{2,\rightarrow}}$ of the function $c_{2,\rightarrow}$ that specifies the class of all the edges of the complete graph of V. The model $\mathscr{M}_{G,\rightarrow}$ essentially predicts a set of vertices and equips the complete graph of these vertices with a $\widehat{c_{2,\rightarrow}}$ function; this is equivalent to mapping each of the predicted

edges $\widehat{E_{\rightarrow}}$ to a class in $\mathcal{K}_{2,\rightarrow}$.

$$\mathscr{D}_V(\vec{m}) = \widehat{V}$$

$$\mathscr{C}_{E,\rightarrow}(\widehat{E_{\rightarrow}}) = \widehat{c_{2,\rightarrow}}$$

$$\mathscr{M}_{G,\rightarrow}(\vec{m}) = \left(\widehat{V}, \widehat{E_{\rightarrow}}, \widehat{c_{2,\rightarrow}}\right)$$

H. Zhang et al. [76, 77] refer to the concept of a "tree-like structure" [76, 77] for representing relationships between objects, and the use of a tree structure (although perhaps not in the strict sense) for representing relationships is also mentioned in [78] and [79]. In [80] and [81], the relationship structures are described as graphs, but it seems that the possibility of the presence cycles is analyzed neither in those works, nor in [76–79] (perhaps because cycles are not possible in the dataset(s) that those authors used or because they considered that there must always be at least one object that can be picked up before any other object). For the problem formulation given above in this current section, using a graph allows for the representation of relevant scenarios that a tree representation would fail to capture (unless duplicate nodes can appear in the tree): as is shown in Section 3.2, cycles—which cannot exist in a tree—can indeed be present in a scene.

3.2 Dataset Challenges

The MetaGraspNet datasets [3, 4] were the main datasets of focus for the relationship-prediction problem described in Section 3.1. The MetaGraspNetV2 dataset contain RGB images with depth information and different types of labels [4]. The dataset is composed of a real-world part and a simulated part: over 3 200 real images and over 296 000 synthetic (computer-rendered) images [4]. Each image is of a bin with zero or more objects (out of 82 classes of objects) in positions that result from simulating the process of dropping the objects into the bin [4]. For the purpose of the work in this thesis, only the synthetic images are used, as they are more abundant, and as their ground-truth labels are presumably more consistent*.

*It appears that synthetic part of the MetaGraspNetV2 [4] dataset contains some issues in the object-ordering labels used for this project, where, for some RGB images, the number of objects present according to these ordering labels differs from the number objects that are actually visible in the bin. For at least some of these cases, the relationships (ordering) of objects were erroneous or misinterpreted. Although this may impact the results for some of those affected images, it seems that less than 0.5 % of the synthetic RGB images of MetaGraspNetV2 are affected.

As a consequence of the randomness in the positions of the objects, at least three categories of difficult scenarios can arise in this dataset:

1. **Cycles**: In a set $S \subseteq V$ of at least two objects, all objects in S can reciprocally obstruct each other in such a way that there is no object in S that is unobstructed by all other objects in S. In such a case, all vertices in S must be forming one or multiple cycles, and thus a cycle would be present in $G^s_{\rightarrow}$. (Indeed, if a vertex v_k were not part of a cycle in S, then there would exist a path $(v_1, v_2, \ldots, v_k, \ldots)$ of one or more vertices in S with v_1 not being obstructed by any other object in S.)

2. **Occluded Boundary**: This refers to cases where one object is obstructing an other object, but there is a third object which partially or completely occludes the occlusion boundary between the first two objects. This is a special case of a situation where there are three objects (v_a, v_b, v_c) such that $(v_a, v_b) \in E^s_{\rightarrow}$, $(v_c, v_a) \in E^s_{\rightarrow}$, and $(v_c, v_b) \in E^s_{\rightarrow}$.

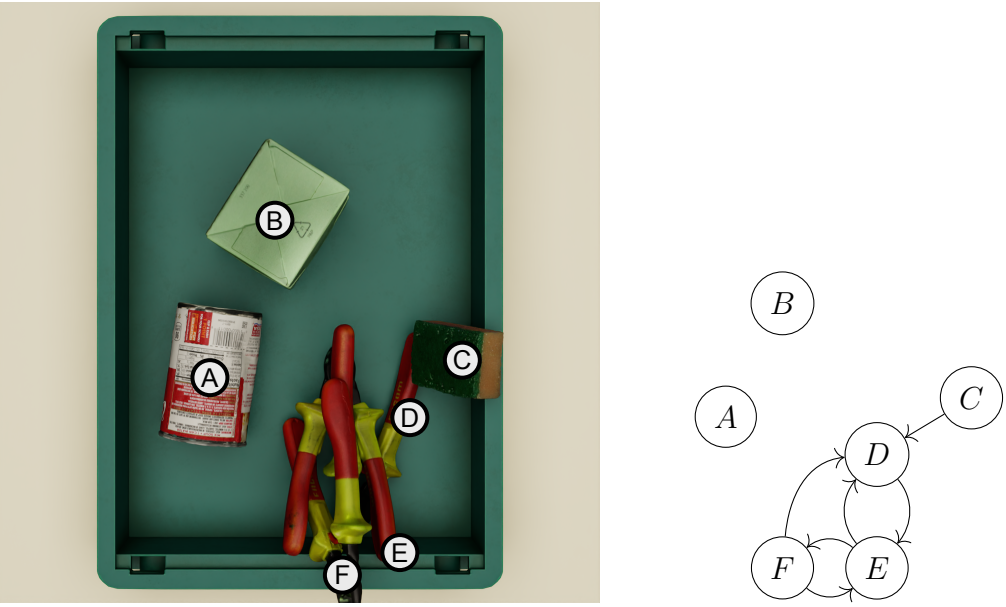
3. **Small Overlap**: In some cases, one object does obstruct an other object, but the area of overlap (or obstruction) between the two objects is very small.

Each of these categories of problems are discussed and illustrated below.

3.2.1 Cycle Problem

Clutter can lead to situations where each object within a set is obstructed by an other object in this set. Such a situation indicates the presence of cycles in the graph representation of the scene. It should be noted that, as a result of this, *trees* are too restrictive to represent the arrangement of objects in a scene (especially for cycles of more than two vertices), and thus, there may at times be no obvious order in which the objects in a cluttered scene should be picked up. In some of the most extreme cases of this, due to the shapes of some of the objects that can appear in the scene, it is possible for only *two* objects to reciprocally obstruct each other (i.e., a *2-cycle*, a cycle of two vertices, is present). Figure 3.2 illustrates a case where two 2-cycles are present in a scene.

The presence of cycles is relevant as it is a situation in which picking any of the objects in the cycle may cause an other object to move in the process. Depending on the requirements or specifications desired for the bin-picking robot, this situation could call for different courses of action (e.g., human intervention, picking the object that is the least obstructed, etc.), but in a lot of cases, a first important step would still be the detection of such a scenario.



(a) (b)

Figure 3.2: Cycles of two or more vertices can be present: (a) Example of a scene containing two 2-cycles and one 3-cycle, which are each formed by wire cutters (original image, without annotations, is from the MetaGraspNetV2 [4] dataset). (b) Illustration of the corresponding graph $G^s_{\rightarrow}$.

As a remark: This situation could be partly addressed by the use of weighted edges in the predicted graph, where the weight of each edge would indicate an estimate for the area of overlap between the two objects.

3.2.2 Occluded-Boundary Problem

In a case where an object v_a occludes an other object v_b, it is possible for a third object v_c to partially or fully occlude the overlap area between v_a and v_b. Based on some of the results in [77, 80] and intuition, it would be expected that this overlap area would be highly relevant for determining $c_{2,\rightarrow}(v_a, v_b)$ and $c_{2,\rightarrow}(v_b, v_a)$. Figure 3.3 illustrates an example of a fully-occluded boundary.

The case of full occlusion might be too difficult to deal with in practice, but it is expected that it should be possible to properly classify cases where a significant part of the

Figure 3.3: Example of a fully occluded boundary. Two separate yellow objects are present in this scene, but they are both partially obstructed by a same object. The area of overlap and the occlusion boundary are both fully occluded. This image is from the MetaGraspNetV2 [4] dataset.

occlusion boundary is still visible. In the case of partial occlusion of an occlusion boundary, it would be important to consider objects other than v_a and v_b when making a prediction about the edges (v_a, v_b) and (v_b, v_a); in particular, it would be important to account for the object(s) which are partially occluding this occlusion boundary. This motivates the use of a *graph network* to predict $c_{2,\rightarrow}(v_a, v_b)$ of any pair $(v_a, v_b) \in E_{\rightarrow}$, as graph networks allow for *message passing* [56] between vertices in a graph: Information from the features of a vertex v_c that is occluding the boundary between v_a and v_b could be thus taken into account when predicting $c_{2,\rightarrow}(v_a, v_b)$ and $c_{2,\rightarrow}(v_b, v_a)$.

3.2.3 Small-Overlap Problem

In the case of one object v_a occluding an other object v_b, the amount of occluded area can be very small. Figure 3.4 shows such an example of small overlap. Correctly classifying the type of relationship in this type of situation can be particularly challenging, especially if the resolution of the input image is reduced at preprocessing time or when convolutions are performed. Mis-classification of small overlap scenarios might not always be inconsequential, as even a small amount of overlap could be sufficient to lift an occluding object and move it substantially. Thus, in the absence of an indicator of the probability that v_a would be excessively moved if v_b were to be picked up, it is more prudent to assume that no amount of (non-zero) overlap should be ignored when determining whether v_b should be lifted directly.

(a) Original image.

(b) Enlarged crop of the original image.

Figure 3.4: Example of image with small overlap between two objects. The image in the right panel is included to more clearly show the small area of overlap. (The original image, left, is from the MetaGraspNetV2 dataset [4].)

3.3 Baseline Model

The task of predicting relative placements of objects for the purposes of robotic grasping has been studied in [76], which has been cited in some related works [78, 79, 81], and as such, it is deemed to be a suitable starting point for a baseline deep-learning model for predicting relationships between objects in a scene. In this current section, the "Visual Manipulation Relationship Network" (abbreviated as "VMRN") [76] model will be discussed, as it was used as a basis for the implementation of a baseline relationship-prediction model which was used for comparison against the model which will be presented in Section 3.4.

The architecture of the VMRN model consists of a backbone ("feature extractor" [76]), an object detector, and a relationship-prediction module dedicated to predicting the relationships of pairs of objects appearing in the scene (the "Object Pairing Pooling Layer" [76], together with the "Manipulation Relationship Predictor" [76]). The VMRN makes predictions on *unordered* pairs of objects, and there are three possible classes [76]:

$\mathcal{K}_{3,\leftrightarrow} \coloneqq \{$"A is obstructing B", "B is obstructing A", "No relationship"$\}$*. The backbone of the VMRN is a function of type $\mathcal{M} \to \mathcal{M}$, and it is composed of convolutional layers. The detector of the VMRN uses the output of the backbone as input, and the detector is a function of type $\mathcal{M} \to \mathcal{P}'(\mathcal{B})$. The VMRN's relationship-prediction module takes as input pairs of bounding boxes along with the feature map produced by the backbone; thus, it is a function of type $\mathcal{B} \times \mathcal{B} \times \mathcal{M} \to \mathcal{K}_{3,\leftrightarrow}$. If the bounding box $B(v_a) \in \mathcal{B}$ of an object v_a is written as

$$B(v_a) = \begin{bmatrix} b^a_{1,1} & b^a_{1,2} \\ b^a_{2,1} & b^a_{2,2} \end{bmatrix}$$

with $[b^a_{1,1} \;\; b^a_{2,1}]^{\mathrm{T}}$ corresponding to the bottom-left corner of the bounding box and $[b^a_{1,2} \;\; b^a_{2,2}]^{\mathrm{T}}$ corresponding to the top-right corner, then the union bounding box [76] of the bounding boxes

$$B(v_a) = \begin{bmatrix} b^a_{1,1} & b^a_{1,2} \\ b^a_{2,1} & b^a_{2,2} \end{bmatrix} \qquad B(v_b) = \begin{bmatrix} b^b_{1,1} & b^b_{1,2} \\ b^b_{2,1} & b^b_{2,2} \end{bmatrix}$$

of two objects v_a and v_b is

$$B_{\cup}(v_a, v_b) \coloneqq \begin{bmatrix} \min\limits_{\alpha}(b^\alpha_{1,1}) & \max\limits_{\alpha}(b^\alpha_{1,2}) \\ \min\limits_{\alpha}(b^\alpha_{2,1}) & \max\limits_{\alpha}(b^\alpha_{2,2}) \end{bmatrix}. \tag{3.6}$$

In [76], for each pair of objects $\{v_a, v_b\}$ on which to make a prediction, three crops of the feature map obtained from the backbone are used together to classify the object pair $\{v_a, v_b\}$:

1. a feature-map crop corresponding to the bounding box of object v_a,
2. a feature-map crop corresponding to the bounding box of object v_b, and
3. a feature-map crop corresponding to the union bounding box of objects v_a and v_b.

In order for ternary classification (with classes $\mathcal{K}_{3,\leftrightarrow}$) to be meaningful, there must be a way to know what A and B each refer to in the names of the classes in $\mathcal{K}_{3,\leftrightarrow}$. Thus, for the discussion that follows, it shall be assumed that for any given set of objects V in a given

*The quotation marks here around each of the three class names are not meant to indicate quotations from [76]; the names of the classes given here are intended to be consistent with the wording in Section 3.1.

image, there exists *some* way to order the vertices; i.e., that for any $\{v_1, v_2, \ldots, v_n\} \subseteq V$, there is some *strict total order relation* denoted "$<$" (the definition of which is unimportant) such that $v_1 < v_2 < \cdots < v_n$.

For a given V, ternary classification on *unordered* pairs of objects is nearly equivalent to binary classification on $G_{\rightarrow}$: with the complete directed graph $G_{\rightarrow} = (V, E_{\rightarrow})$ can be defined a corresponding complete undirected counterpart $G_{\leftrightarrow} \coloneqq (V, E_{\leftrightarrow})$ containing the same vertices V, where

$$E_{\leftrightarrow} \coloneqq \{\{v_a, v_b\} \mid (v_a, v_b) \in E_{\rightarrow}\}\,.$$

This definition of $E_{\leftrightarrow}$ halves the number of edges, but it still gives a bijection $b_{\mathcal{E}} \colon \mathcal{E}_{\rightarrow} \to \mathcal{E}_{\leftrightarrow}$ between the set $\mathcal{E}_{\rightarrow}$ of all edge sets of the form $E_{\rightarrow}$ and the set $\mathcal{E}_{\leftrightarrow}$ of all edge sets of the form $E_{\leftrightarrow}$ (that is, an element of $\mathcal{E}_{\rightarrow}$ corresponds to the all the edges of a particular complete directed graph, and an element of $\mathcal{E}_{\leftrightarrow}$ to those of a particular complete undirected graph). With the undirected edges $E_{\leftrightarrow}$, it is still possible to define a counterpart to $c_{2,\rightarrow}$ for undirected edges, without loss of information: Let the function $c_{4,\leftrightarrow}$ for any $G_{\leftrightarrow}$ be defined as

$$c_{4,\leftrightarrow} \colon E_{\leftrightarrow} \to \mathcal{K}_{2,\rightarrow} \times \mathcal{K}_{2,\rightarrow}$$

$$\{v_a, v_b\} \mapsto \begin{cases} (t_1, t_2) & \text{if } v_a < v_b \\ (t_2, t_1) & \text{if } v_b < v_a \end{cases} \quad \text{with } (t_1, t_2) \coloneqq \big(c_{2,\rightarrow}(v_a, v_b), c_{2,\rightarrow}(v_b, v_a)\big)\,.$$

With such a definition, one can obtain any $c_{4,\leftrightarrow}$ from any $c_{2,\rightarrow}$, and vice versa: this definition gives a bijection $b_{\mathcal{C}} \colon \mathcal{C}_{2,\rightarrow} \to \mathcal{C}_{4,\leftrightarrow}$ between the set $\mathcal{C}_{4,\leftrightarrow}$ of functions of type $c_{4,\leftrightarrow}$ and the set $\mathcal{C}_{2,\rightarrow}$ of functions of type $c_{2,\rightarrow}$. Therefore, using a module $\mathscr{C}_{E,\rightarrow}$ to predict $c_{2,\rightarrow}$ of the edges $E_{\rightarrow}$ of a complete directed graph is the same as converting $E_{\rightarrow}$ to its undirected version $E_{\leftrightarrow}$, then using a module $\mathscr{C}_{E,\leftrightarrow}$ to find $c_{4,\leftrightarrow}$, and then converting $c_{4,\leftrightarrow}$ to $c_{2,\rightarrow}$:

$$\mathscr{C}_{E,\rightarrow} = {b_{\mathcal{C}}}^{-1} \circ \mathscr{C}_{E,\leftrightarrow} \circ b_{\mathcal{E}}\,.$$

Quaternary classification, through $c_{4,\leftrightarrow}$, on undirected edges can thus be considered equivalent to performing binary classification on directed edges; however, *ternary* classification on undirected edges must result in a loss of information, as $c_{4,\leftrightarrow}$ could be surjective and therefore it is not possible to always recover it from a function $c_{3,\leftrightarrow} \colon E_{\leftrightarrow} \to \mathcal{K}_{3,\leftrightarrow}$ which assigns one of three classes to each undirected edge. (For a fixed V, there is no bijection between $\mathcal{C}_{4,\leftrightarrow}$ and $\mathcal{C}_{3,\leftrightarrow}$, since there is a difference in cardinality.) Therefore, this motivates the estimation of $c_{2,\rightarrow}$ or $c_{4,\leftrightarrow}$ rather than $c_{3,\leftrightarrow}$.

The baseline relationship-prediction model which was heavily based on the VMRN of [76] was implemented for the purpose of comparing its performance to the model which will be presented in Section 3.4. A major aspect of this baseline model that differs from the architecture described in [76] is that the baseline model was designed to perform binary classification of directed edges rather than ternary classification of undirected edges (i.e., prediction, for each image, of a $c_{2,\rightarrow}$ rather than a $c_{3,\leftrightarrow}$). This is consistent with the problem formulation from Section 3.1, which specifies the classification of directed edges and allows for 2-cycles to be present in any given scene.

3.4 Proposed Model

In this section, a model architecture to estimate $G^s_{\rightarrow}$ of a scene in a given image will be presented, with the intent of addressing, at least partly, each of the three main categories of problems described in Section 3.2.

In contrast to the architecture described in [76], the model being proposed in this current thesis chapter uses a *graph-network architecture*[*] and makes prediction on *ordered* pairs of objects. The related works in which a graph-based architecture is presented include [80, 81]. In addition, to make a prediction on any given edge (v_a, v_b), instead of using the *union* bounding boxes of v_a and v_b, it uses the *intersection-with-margin* bounding box (defined later) of the two objects under consideration (incidentally, the use of the intersection was recently proposed by [79]). Furthermore, the proposed model uses bounding box coordinates (or coordinate embeddings) as features of the edges or vertices of the graph. Descriptions and motivations for each of these main architectural features are given below. In addition to these major changes, instead of setting the feature extractor to output a 512-channel feature map as in [76], the number of such channels was reduced set to only 256, and the resolution of the cropped feature maps was set to 14×14, instead of 7×7 as in [76].

Graph network architecture: With the use of graph networks come some potentially favourable characteristics, namely message passing and global features:

- Message passing: By allowing information to flow from one vertex to another and from vertices to vertices, it is hypothesized that the model would be able to better account for the presence and potential effects that neighbouring objects $(v_c, v_d, \dots)$ have on

[*]Although it might be the case that the model in [76] could technically fall under the definition of a graph network, it would be a rather simple graph network, as it is missing common aspects of a graph network, such as global features and message passing.

the camera image representation of the two objects (v_a, v_b) under consideration. This is particularly relevant when a neighbouring object v_c is affecting the content of each of the three cropped feature maps for (v_a, v_b): e.g., v_c can affect the representation of v_a by partially occluding it, and v_a could also affect the amount by which the occlusion boundary between v_a and v_b is visible (this is the occluded-boundary problem described in Subsection 3.2.2).

- Global feature map: A global feature map was used with the motivation of allowing the model to account for global image properties such as the angle of the camera with respect to the bin, the distance between the camera and the bin, or some properties of the lighting in the scene (e.g., angle, magnitude, colour, and number of sources).

Intersection-with-margin bounding box: The intersection-with-margin, defined below with (3.9), was used instead of the union of the bounding boxes in order to allow the model to be more sensitive to small overlaps between two objects v_a and v_b. If the same notation system as in (3.6) is used, then the intersection bounding box of two objects (v_a, v_b) is

$$B_{\cap}(v_a, v_b) \coloneqq \begin{bmatrix} \max\limits_{\alpha}(b^{\alpha}_{1,1}) & \min\limits_{\alpha}(b^{\alpha}_{1,2}) \\ \max\limits_{\alpha}(b^{\alpha}_{2,1}) & \min\limits_{\alpha}(b^{\alpha}_{2,2}) \end{bmatrix}, \tag{3.7}$$

$$\text{only defined when} \quad \forall i \ \max_{\alpha}(b^{\alpha}_{i,1}) < \min_{\alpha}(b^{\alpha}_{i,2})\,. \tag{3.8}$$

Similarly, the intersection-with-margin bounding box is

$$B^{\mathrm{m}}_{\cap}(v_a, v_b) \coloneqq \begin{bmatrix} \max\limits_{\alpha}(b^{\alpha}_{1,1}) & \min\limits_{\alpha}(b^{\alpha}_{1,2}) \\ \max\limits_{\alpha}(b^{\alpha}_{2,1}) & \min\limits_{\alpha}(b^{\alpha}_{2,2}) \end{bmatrix} + \epsilon_m \begin{bmatrix} -1 & 1 \\ -1 & 1 \end{bmatrix}, \tag{3.9}$$

$$\text{only defined when} \quad \forall i \ \max_{\alpha}(b^{\alpha}_{i,1}) - \epsilon_m < \min_{\alpha}(b^{\alpha}_{i,2}) + \epsilon_m \tag{3.10}$$

(for some given, relatively small hyperparameter ϵ_m that sets the size of the margin; in practice, a rule is also included to prevent $B^{\mathrm{m}}_{\cap}(v_a, v_b)$ from having a corner fall outside of the bounds of the image).

The motivation for using the intersection-with-margin is as follows: Since the 2D representation of an object on the image must be entirely contained within its (ground-truth) bounding box, any occlusion area and occlusion boundary between v_a and v_b must be entirely contained within the (ground-truth) intersection bounding box $B_{\cap}(v_a, v_b)$. Since

the rest of object v_a and v_b can already be captured or represented by the features falling exclusively in $B(v_a)$ or $B(v_b)$, using the union bounding box might introduce redundancy in the information given to the relationship-prediction module. Furthermore, avoiding this redundancy allows for a greater resolution for the area of overlap between the two objects without increasing the size of the (discrete) convolution kernel or reducing the convolution stride—i.e., without increasing the number of training parameters.

Bounding box coordinates: Only using the cropped feature maps forgoes information that could easily be used to estimate the relative position of the projections of v_a and v_b on the 2D image (although it might be possible to infer their relative positions through the feature maps alone). Information about the relative positions of the projections of the objects is presumably relevant, and it is expected that it would reduce the amount of information that the feature-map layers would have to infer.

In Section 2.3, a brief summary of graph networks was given, along with the mention that attributes, given by the function $\mathbf{f}^i$, are defined for the vertices, a subset (see below) of the edges, and the entire graph. In the particular implementation herein proposed, these attributes in a graph G are as follows:

- For an edge e, if $B^{\mathrm{m}}_{\cap}(e)$ is defined according to (3.10), $\mathbf{f}^i_{\mathrm{vec}}(e)$ is a feature vector (not a 2D spatial feature map) which is the bounding box coordinates or an embedding of these coordinates of $B^{\mathrm{m}}_{\cap}(e)$, and $\mathbf{f}^i_{\mathrm{map}}(e)$ is a feature map which is defined for each point that is present within the bounding box $B^{\mathrm{m}}_{\cap}(e)$. If $B^{\mathrm{m}}_{\cap}(e)$ is not defined, then e has no attributes.

- Similarly, for a vertex v, $\mathbf{f}^i_{\mathrm{vec}}(v)$ is a feature vector for the embeddings of the bounding box coordinates of $B(v)$, and $\mathbf{f}^i_{\mathrm{map}}(v)$ is a feature map which is defined at each point within $B(v)$.

Prior to describing the attribute-update mechanism, some notation will be introduced. For any feature vector or feature map $\vec{x}$ being processed by the i^{th} GN block, let $L_{i,j}\vec{x}$ denote

$$\begin{bmatrix} M_{i,j} & \vec{\beta}_{i,j} \end{bmatrix} \begin{bmatrix} \vec{x} \\ 1 \end{bmatrix},$$

where $M_{i,j}$ is some matrix of trainable parameters and $\vec{\beta}_{i,j}$ is some vector of trainable parameters, both proper to the i^{th} GN block (here, $M_{i,j}$ does *not* refer to some element at indices (i, j) in M). The variable j here is simply an indexing variable to make distinctions

between each matrix or each vector of the i^{th} GN block. For any feature vector $\vec{x}$, let $\text{bn}_{i,j}\,\vec{x}$ denote a (trainable) batch-normalization operation on $\vec{x}$, proper to the i^{th} GN block. The notation $\text{bn}_{i,j}\,L_{i,j}\vec{x}$ should be read as $\text{bn}_{i,j}\big(L_{i,j}\vec{x}\big)$. Let σ be the ReLU activation function. A "crop" function was used in the update mechanism of the edge and vertex feature maps; this function is intended to be the same as or very similar to the cropping mechanism used in [76]. For clarity, the "crop" function used in the model proposed herein is defined below, with (3.11) and (3.12). Since cropping operations are done along spatial dimensions but fully-connected layers operate on finite-dimensional vectors, it may be useful to introduce the notation "cont" (with an inverse, "dcr"), where for any discrete-domain feature map $\vec{x}_{\text{map}}$, the notation $\text{cont}\,\vec{x}_{\text{map}}$ refers to a continuous-domain vector field corresponding to $\vec{x}_{\text{map}}$. Using this notation, for some feature map $\vec{x}_{\text{map}}$, some vertex v, and some edge e,

$$\text{crop}\big(\vec{x}_{\text{map}}, B(v), B(e)\big) = \text{dcr}(f)\,, \tag{3.11}$$

$$\text{where } f\colon \vec{z} \mapsto \begin{cases} \vec{0} & \text{if } \vec{z} \text{ is not within } B(v) \\ \big(\text{cont}\,\vec{x}_{\text{map}}\big)(\vec{z}) & \text{otherwise} \end{cases}, \tag{3.12}$$

defined for any $\vec{z}$ that is within $B(e)$.

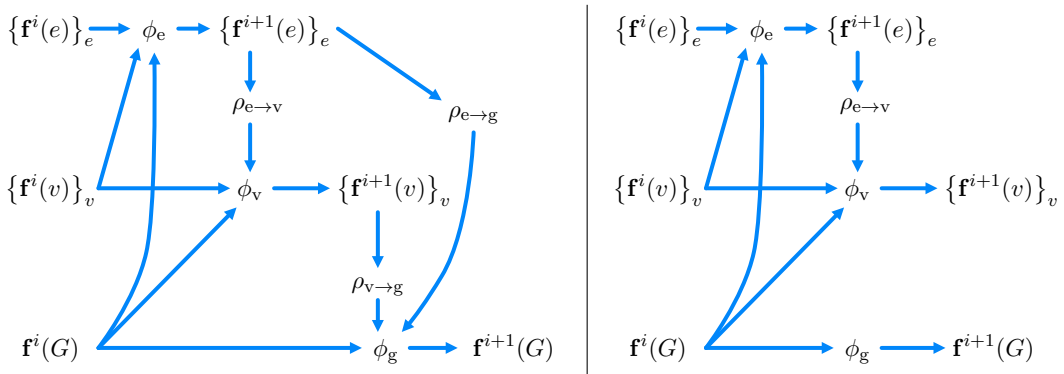
Edge, vertex, and global attributes of the graph G are updated as follows:

- For the attributes of an edge $e = (v_a, v_b)$, the update function ϕ_{e} (introduced in Section 2.3) of a GN block can be, in this specific implementation, described by giving separate expressions for $\mathbf{f}^{i+1}_{\text{vec}}(e)$ and $\mathbf{f}^{i+1}_{\text{map}}(e)$. The ϕ_{e} of the first GN block simply consists of a fully-connected layer for $\mathbf{f}^{1}_{\text{vec}}(e)$ and a convolutional layer for $\mathbf{f}^{1}_{\text{map}}(e)$:

$$\mathbf{f}^{2}_{\text{vec}}(e) = \text{bn}_{1,1}\,\sigma\Big(L_{1,1}\,\mathbf{f}^{1}_{\text{vec}}(e)\Big)$$

$$\mathbf{f}^{2}_{\text{map}}(e) = \text{bn}_{1,4}\,\sigma\Big(L^{\text{conv}}_{1,4}\,\mathbf{f}^{1}_{\text{map}}(e)\Big)\,,$$

  where $L^{\text{conv}}_{1,4}$ applies a convolution operation (which can be described in matrix form). For subsequent GN blocks, except the final one, the ϕ_{e} of the i^{th} GN block is as



(a) Full GN block [56]. (b) Proposed GN block.

Figure 3.5: The proposed GN block, compared to the full GN block [56], is missing a $\rho_{e\to g}$ function and a $\rho_{v\to g}$ function. This figure is based on and adapted from Figure 4 of [56].

follows:

$$\mathbf{f}^{i+1}_{\text{vec}}(e) = \text{bn}_{i,2}\, L_{i,2}\Big(\text{bn}_{i,1}\, \sigma\big(L_{i,1}\vec{c}_{\text{vec}}\big)\Big)$$

$$\vec{c}_{\text{vec}} \coloneqq \text{cat}\Big(\mathbf{f}^i_{\text{vec}}(e), \mathbf{f}^i_{\text{vec}}(v_a), \mathbf{f}^i_{\text{vec}}(v_b)\Big)$$

$$\mathbf{f}^{i+1}_{\text{map}}(e) = \text{bn}_{i,5}\, \sigma\Bigg(L^{\text{conv}}_{i,4}\vec{c}_{\text{map}} + \text{extend}\Big(L_{i,3}\, \mathbf{f}^i_{\text{map}}(G)\Big) + \text{extend}\Big(L_{i,5}\, \mathbf{f}^i_{\text{vec}}(e)\Big)\Bigg)$$

$$\vec{c}_{\text{map}} \coloneqq \text{cat}\Bigg(\mathbf{f}^i_{\text{map}}(e), \text{crop}\Big(\mathbf{f}^i_{\text{map}}(v_a), B(v_a), B(e)\Big) - \text{crop}\Big(\mathbf{f}^i_{\text{map}}(v_b), B(v_b), B(e)\Big)\Bigg),$$

where $L^{\text{conv}}_{i,4}$ applies a convolution operation, where "extend(□)" denotes a feature map with value □ at each point of the map, and where "cat" denotes concatenation of feature maps along the array axis that indexes the channels of the feature map. The final GN block simply flattens the feature map of each edge and applies a series of fully-connected layers (with a non-linear activation function and batch normalization) to obtain per-edge classification predictions.

- For the attributes of a vertex v, let N_v be a subset of the edges that are adjacent and directed to v, defined as

$$N_v = \Big\{(v_k, v) \;\Big|\; (v_k, v) \in E^s_{\to} \;\wedge\; \big(B^{\text{m}}_{\cap}(v_k, v) \text{ is defined according to (3.10)}\big)\Big\}.$$

Thus, for a vertex v, the update function ϕ_{v} (introduced in Section 2.3) can be described in this specific implementation with expressions for $\mathbf{f}_{\mathrm{vec}}^{i+1}(v)$ and $\mathbf{f}_{\mathrm{map}}^{i+1}(v)$. For the first GN block, ϕ_{v} simply applies a fully-connected layer on $\mathbf{f}_{\mathrm{vec}}^{i+1}(v)$ and a convolutional layer on $\mathbf{f}_{\mathrm{map}}^{i+1}(v)$:

$$\mathbf{f}_{\mathrm{map}}^{2}(v) = \mathrm{bn}_{1,7}\,\sigma\Big(L_{1,7}^{\mathrm{conv}}\,\mathbf{f}_{\mathrm{map}}^{1}(v)\Big)$$

$$\mathbf{f}_{\mathrm{vec}}^{2}(v) = \mathrm{bn}_{1,8}\,\sigma\Big(L_{1,8}\,\mathbf{f}_{\mathrm{vec}}^{1}(v)\Big)\,,$$

where $L_{1,7}^{\mathrm{conv}}$ applies a convolution operation. For subsequent GN blocks, except the final one (where no further processing is performed for the edges), the ϕ_{v} of the i^{th} GN block is as follows:

$$\mathbf{f}_{\mathrm{map}}^{i+1}(v) = \mathrm{bn}_{i,7}\,\sigma\Bigg(L_{i,7}^{\mathrm{conv}}\vec{c}_{\mathrm{map}} + \mathrm{extend}\Big(L_{i,6}\,\mathbf{f}_{\mathrm{map}}^{i}(G)\Big)\Bigg)$$

$$\vec{c}_{\mathrm{map}} \coloneqq \mathrm{cat}\Big(\mathbf{f}_{\mathrm{map}}^{i}(v), \vec{a}_{\mathrm{map}}\Big)$$

$$\vec{a}_{\mathrm{map}} \coloneqq \sum_{e_k \in N_v} \frac{1}{|N_v|}\,\mathrm{crop}\Big(\mathbf{f}_{\mathrm{map}}^{i+1}(e_k), B(e_k), B(v)\Big)$$

$$\mathbf{f}_{\mathrm{vec}}^{i+1}(v) = \mathrm{bn}_{i,8}\,\sigma\Big(L_{i,8}\,\mathbf{f}_{\mathrm{vec}}^{i}(v)\Big)\,,$$

where $L_{i,7}^{\mathrm{conv}}$ applies a convolution operation. Note that this expression for $\vec{a}_{\mathrm{map}}$ corresponds to the local edge-aggregation function $\rho_{\mathrm{e}\to\mathrm{v}}$ (mentioned in Section 2.3), as it *aggregates* some features from incident edges (pointing toward v), and it is defined for any positive number (or zero) of such incident edges, for a given v (it is $\vec{0}$ if there are no incident edges).

- The global attributes of the entire graph G are only updated once, at the beginning of the graph-network module. This is because the global attributes are updated without using the attributes of the vertices and edges (see Figure 3.5), so processing the global features through, say, N_g different GN blocks, each applying a separate fully connected layer, is equivalent to simply applying these N_g fully connected layers at once at the beginning, in the first GN block. The global update function ϕ_{g} (introduced in Section 2.3), for this particular implementation, is described below:

$$\mathbf{f}_{\mathrm{map}}^{2}(G) = \mathrm{bn}_{1,11}\,\sigma\Bigg(L_{1,11}^{\mathrm{conv}}\,\mathrm{bn}_{1,10}\,\sigma\Big(L_{1,10}^{\mathrm{conv}}\,\mathrm{bn}_{1,9}\,\sigma\Big(L_{1,9}^{\mathrm{conv}}\,\mathbf{f}_{\mathrm{map}}^{1}(G)\Big)\Big)\Bigg)\,,$$

where $L_{1,9}^{\text{conv}}$, $L_{1,10}^{\text{conv}}$, and $L_{1,11}^{\text{conv}}$ each apply a convolution operation.

Figure 3.6 and Figure 3.7 illustrate a simplified overview of the main edge-update and vertex-update mechanisms. Note that $\rho_{\text{e}\to\text{g}}$ and $\rho_{\text{v}\to\text{g}}$ (mentioned in Section 2.3) are purposely missing and undefined, as the essential representation and processing was intended to be at the level of vertex and edge attributes (since the model was designed to predict classes of *edges*); thus, it was presumed that updating the global attributes using attributes of individual vertices and edges would not be significantly advantageous.

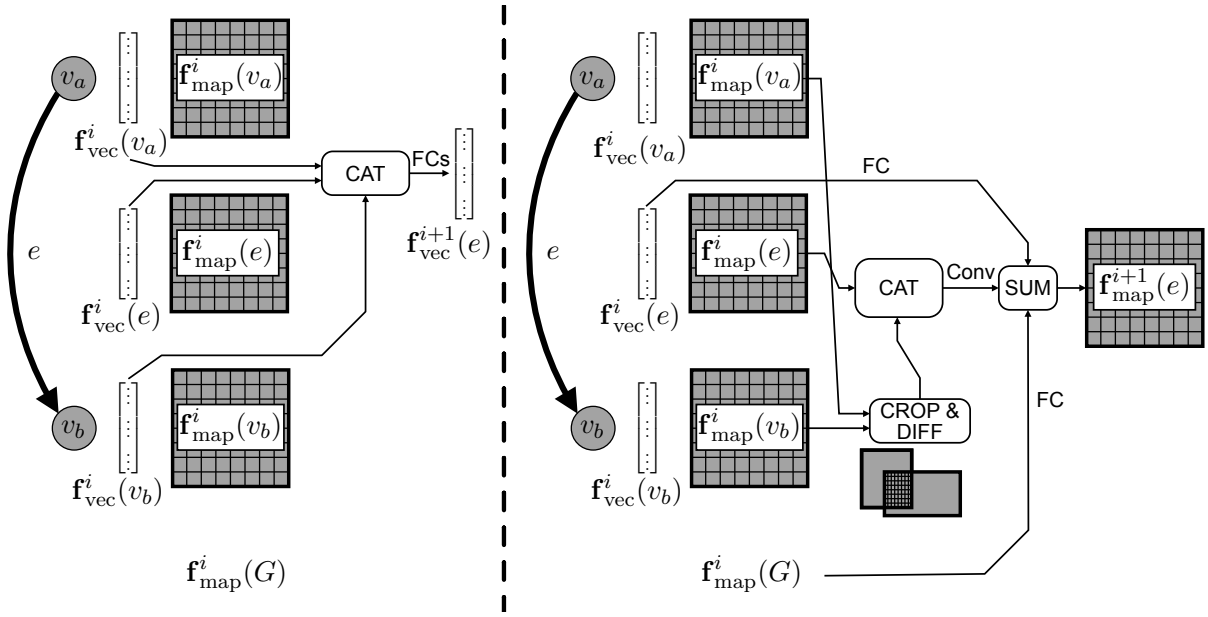


Figure 3.6: Overview of the main edge-update mechanism for an edge e between vertices v_a and v_b. In each of the two panels of the figure, the edge e and the vertices v_a and v_b are illustrated on the left. The left panel of the figure illustrates the update mechanism which outputs $\mathbf{f}_{\text{vec}}^{i+1}(e)$. The right panel illustrates the mechanism for $\mathbf{f}_{\text{map}}^{i+1}(e)$.

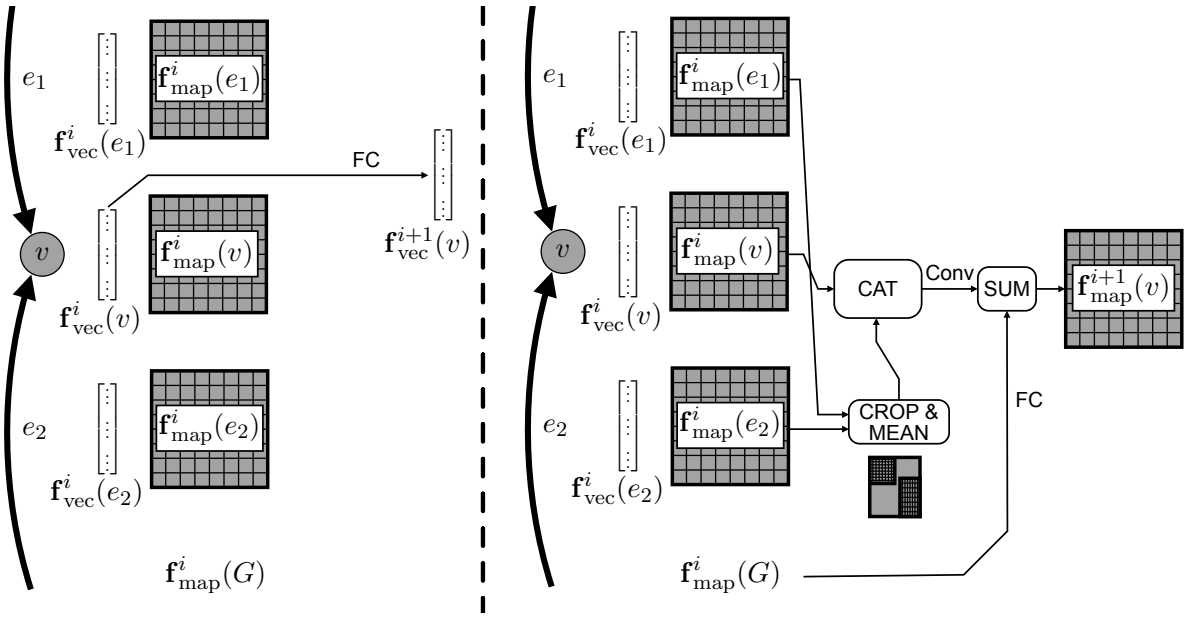


Figure 3.7: Overview of the main vertex-update mechanism for a vertex v with two incident edges e_1 and e_2. In each of the two panels of the figure, e_1, e_2, and v are illustrated on the left. The left panel of the figure illustrates the update mechanism which outputs $\mathbf{f}^{i+1}_{\text{vec}}(v)$. The right panel illustrates the mechanism for $\mathbf{f}^{i+1}_{\text{map}}(v)$.

3.5 Performance Metrics

This section will focus on describing and justifying the precise metric used to evaluate the proposed model. In a cluttered scene, it is often the case that $|E_{\rightarrow} \setminus E^s_{\rightarrow}| \gg |E^s_{\rightarrow}|$; i.e., that for any two given objects (v_a, v_b) in a scene, it is much more likely that $c_{2,\rightarrow}(v_a, v_b) =$ "not obstructing" than $c_{2,\rightarrow}(v_a, v_b) =$ "obstructing". As such, without knowledge of the ratio $|E_{\rightarrow} \setminus E^s_{\rightarrow}|/|E^s_{\rightarrow}|$, which is an indicator the degree of class imbalance, keeping track of the recall and precision of the model is more meaningful and insightful than simply the accuracy of the model. To compute the precision and recall, the positive class is considered to be "obstructing", while "not obstructing" is considered to be the negative class. I.e., an instance is positive if and only if it is in $E^s_{\rightarrow}$, which is either that of the ground-truth graph or that of the predicted graph (depending on context). At a first glance, this description of positive and negative classes may appear sufficient to computing precision and recall; however, some complications may arise when using *predicted* bounding boxes. Some recall and precision metrics were mentioned in the works [76–81]; however, it is unclear to the author how, in those aforementioned works, the recall and precision metrics were calculated for the (presumably numerous) cases in which the detector underestimated or overestimated of number of objects in the scene. Thus, these cases of underestimation and overestimation in the number of vertices are discussed below, and a precise computation method for recall and precision are described below.

Some complications arise when the number of predicted bounding boxes outputted by the detector $\mathscr{D}_V$ is not equal to the number of ground-truth bounding boxes; i.e., when the number $|\widehat{V}|$ of vertices in the predicted graph $\widehat{G^s_{\rightarrow}}$ differs from the number $|V|$ of vertices in the ground-truth graph $G^s_{\rightarrow}$. In such a situation, after the assignment problem between $\widehat{V}$ and V has been resolved, if $|\widehat{V}| > |V|$, there will be $|\widehat{V}| - |V|$ vertices in $\widehat{V}$ which have not been assigned to a vertex in V; otherwise, if $|\widehat{V}| < |V|$, then there will be $|V| - |\widehat{V}|$ vertices in V which have not been assigned to a vertex in $\widehat{V}$. Described in terms of the edges, resolving the assignment problem between the vertices implicitly gives an assignment between the edges $\widehat{E_{\rightarrow}}$ of the predicted graph and the edges $E_{\rightarrow}$ of the ground-truth graph. Once this assignment problem is resolved, there are four cases of interest, each described in an order that roughly follows the illustrations in Figure 3.8:

(A) For any edge $\widehat{e} = (\widehat{v_a}, \widehat{v_b}) \in \widehat{E^s_{\rightarrow}}$, either $\widehat{e}$ has been assigned to an edge $e \in E^s_{\rightarrow}$, which corresponds to a *true positive*, or it has not been assigned to an edge e, which corresponds to a *false positive*. A false positive can be obtained in only four ways:

(Type 1) $\widehat{v_a}$ and $\widehat{v_b}$ have each been assigned to some ground-truth vertices v_a and v_b, but $\widehat{e} \notin \widehat{E^s_{\rightarrow}}$ as a result of the classification performed by $\mathscr{C}_{E,\rightarrow}$.

(Type 2) $\widehat{v_b}$ has been assigned to some ground-truth vertex v_b, but $\widehat{v_a}$ has not been assigned to some ground-truth vertex. Thus, a false-positive *must* have occurred, regardless of the ground-truth classes given by $c_{2,\rightarrow}$.

(Type 3) $\widehat{v_a}$ has been assigned to some ground-truth vertex v_a, but $\widehat{v_b}$ has not been assigned to some ground-truth vertex. Similarly to (Type 2), this must mean that a false-positive has occurred.

(Type 4) Neither $\widehat{v_a}$ nor $\widehat{v_b}$ have been assigned to some ground-truth vertices, and thus a false-positive must have occurred.

(B) For any edge $e = (v_a, v_b) \in E^s_\rightarrow$, either e has been assigned to a predicted edge $\widehat{e} \in \widehat{E^s_\rightarrow}$, which corresponds to a *true positive*, or it has not been assigned to an edge e, which corresponds to a *false negative*. A false negative can be obtained in only four ways, which are analogous to the four ways of obtaining a false positive described in (A).

(C) For any edge $e = (v_a, v_b) \in E_\rightarrow \setminus E^s_\rightarrow$ (this is a ground-truth edge for which the ground-truth class is negative), either e has been assigned to a predicted edge $\widehat{e} \in \widehat{E^s_\rightarrow}$, which corresponds to a false positive case already addressed above, or e has been assigned to a predicted edge $\widehat{e} \in \widehat{E_\rightarrow} \setminus \widehat{E^s_\rightarrow}$, which corresponds to a Type 1 *true negative*, or it has been neither assigned to an $\widehat{e} \in \widehat{E^s_\rightarrow}$ nor to an $\widehat{e} \in \widehat{E_\rightarrow} \setminus \widehat{E^s_\rightarrow}$, in which case this *still corresponds to a true negative* (types 2–4), since the model is not claiming that v_a is obstructing v_b (because there is no $\widehat{v_a}$ assigned to v_a or there is no $\widehat{v_b}$ assigned to v_b), and there is also no such obstruction according to the ground-truth graph.

(D) For any edge $\widehat{e} = (\widehat{v_a}, \widehat{v_b}) \in \widehat{E_\rightarrow} \setminus \widehat{E^s_\rightarrow}$, either $\widehat{e}$ has been assigned to a ground-truth edge $e \in E^s_\rightarrow$, which corresponds to a false negative case already addressed above, or $\widehat{e}$ has not been assigned to such an edge, which corresponds to a true negative (types 5–7). Note that these types of true negatives (types 5–7) are deemed to be less relevant than true negatives of types 1–4 because they essentially result from a vertex detector that "detects" vertices that are not actually present and an edge-classification model that does not claim than these extra vertices are obstructed by or obstructing any other object. Ultimately, these types of true negatives are not providing *any* information about the scene, and so a model which generates more true negatives of types 5–7 should not be considered to have better performance than a model giving no true negatives of types 5–7.

Considering all of these types of edge-prediction cases, the following precision and recall formulas are proposed, based on the normal definition of precision and recall:

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP1} + \text{FP2} + \text{FP3} + \text{FP4}}$$

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN1} + \text{FN2} + \text{FN3} + \text{FN4}} \,,$$

where TP is the number of true positives, FPk is the number of false positives of type k, and FNk is the number of true negatives of type k. Then, the usual formula for the F1 score can be used:

$$\text{F1} = \frac{2}{(\text{recall})^{-1} + (\text{precision})^{-1}} \,.$$

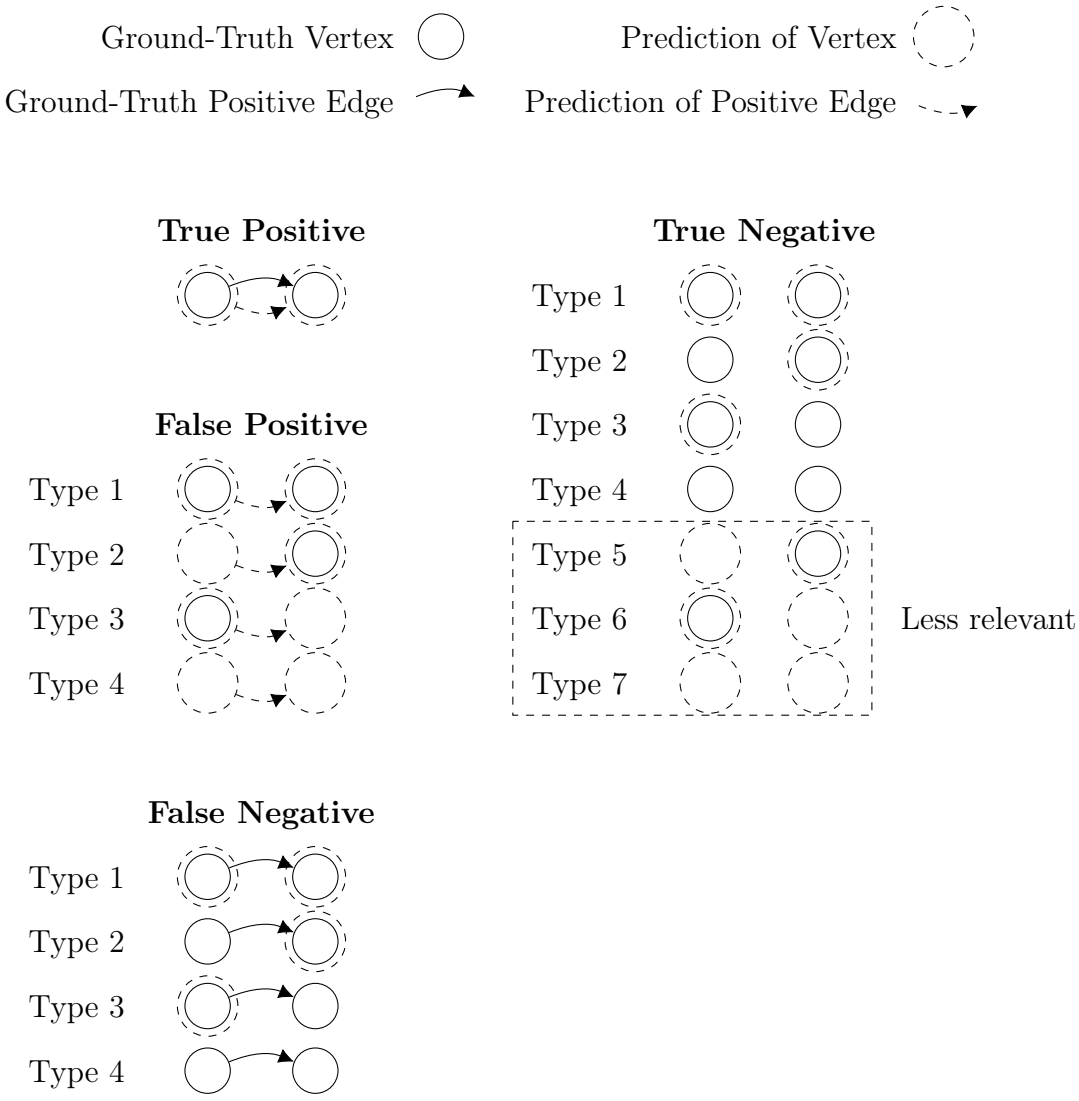


Figure 3.8: Types of edge detection errors between a ground-truth and a predicted graph. Two concentric circles (one dashed and one solid) together illustrate that a predicted vertex has been assigned to a ground-truth vertex. A single circle without a corresponding concentric circle indicates that it is not assigned to another vertex.

3.6 Results

In order to evaluate the effect of the proposed model architecture changes in Section 3.4, the baseline model architecture described in Section 3.3 and the proposed model architecture were compared using the performance metrics specified in Section 3.5 by training and evaluating on the MetaGraspNetV2 [4] dataset, using its synthetic RGB images (without depth) as inputs.

In addition, the VMRD dataset version "V2" [82] was also used for separate training and evaluation; this dataset was selected because has been used by other groups in related works [76, 77, 79–81] (the VMRD dataset version used may vary between these works). The "V2" version of this dataset contains 4 683 colour images [77, 82].

Based on Figure 3.9, on the MetaGraspNetV2 dataset, the proposed model architecture (labelled "GNN") outperformed the baseline model in terms of the F1 score on the validation dataset partition, and the model appeared to consistently outperformed the baseline model for 10^5 training iterations. Similarly, for the VMRD dataset, the results in Figure 3.10 indicate that the proposed model architecture provides at least a modest improvement over the baseline model.

One might note the very large gap between the "Training" F1 scores and the "Validation (Pred. BBoxes)" F1 scores of the models in both Figure 3.9 and Figure 3.10; this large gap is explained partly by the use of *predicted* bounding boxes at validation time in contrast to the use of *ground-truth* bounding boxes as training time. The performance when the ground-truth bounding boxes were used for validation is shown by the "Validation (G.T. BBoxes)" curves.

In order to further investigate the extent to which any of the architecture features proposed in Section 3.4 positively or negatively the F1 score, a detailed ablation study is recommended. In particular, each of the 2^3 combinations of inclusion or exclusion of the main architecture changes could be tested: (i) the use of $B_{\cap}^{\mathrm{m}}$ instead of $B_{\cup}$, (ii) the use of bounding-box coordinate embeddings in the vertex and node attributes, and (iii) message passing between vertices or edges in the graph, with the implementation of $\rho_{\mathrm{e}\to\mathrm{v}}$ for locally aggregating edges and the use of the "crop" and "cat" functions in the implementation of ϕ_{e} for combining feature maps, as described in Section 3.4.

Finally, another recommendation for a continuation of this work would be to compare the performance of the baseline model and the proposed model for each of the three categories of difficulties outlined in Section 3.2. A shortcoming of the results presented in Figure 3.9 and Figure 3.10 is that the MetaGraspNetV2 dataset and the version of the VMRD dataset used seemed to be missing some or all 2-cycles in the ground-truth labels. In the case of

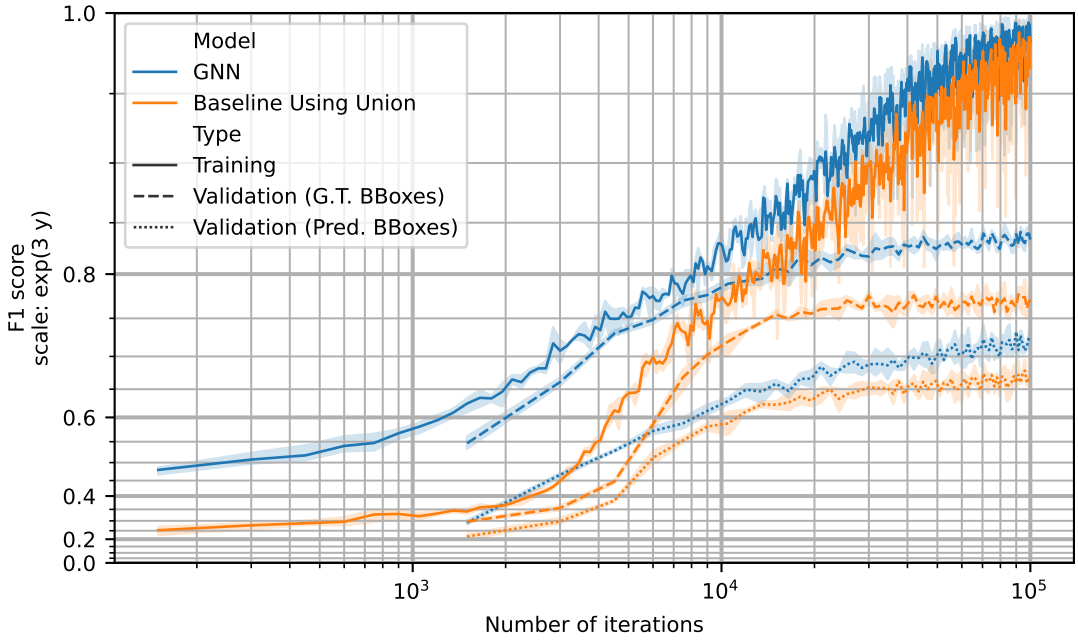


Figure 3.9: Results for three training runs with synthetic RGB images of the MetaGraspNetV2 [4] dataset. The "Validation (Pred. BBoxes)" curves indicate the true validation performance, while the "Validation (G.T. BBoxes)" curves show performance when the ground-truth bounding boxes are used by the relationship-prediction modules. A non-linear vertical scale was used to emphasize differences at higher F1 scores. The shaded area shows the range (between minimum and maximum) of performance across the three runs, for a given number of iterations.

the MetaGraspNetV2 dataset, this is simply due to the method by which the dataset was labelled, whereas in the case of the VMRD dataset, it could be presumed that the absence of 2-cycles in the ground-truth labels might have simply been a design choice that was made upon the creation of the VMRD dataset. In any case, the presence of 2-cycles is a real issue which can arise, and it should not be ignored merely due to its absence in some dataset labels. As such, a recommendation is to amend the labels of the MetaGraspNetV2 dataset in order to represent, in the ground-truth labels, all the 2-cycles that were actually present in each image.

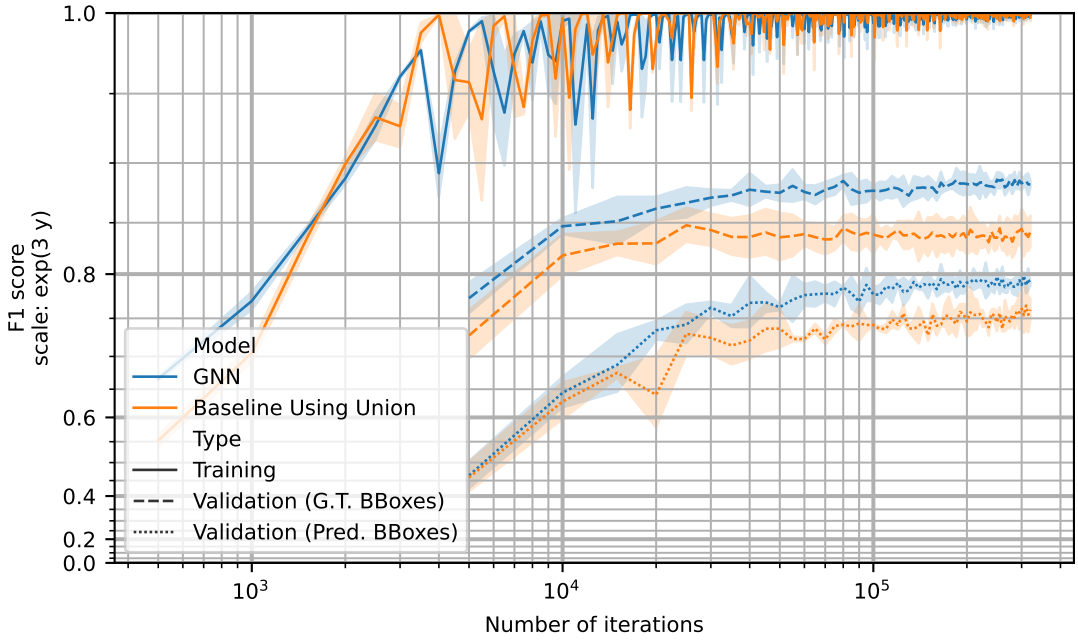


Figure 3.10: Results for two training runs with the VMRD dataset. (With 18 or fewer images removed due to apparent issues with the ground-truth labels.) A non-linear vertical scale was used, just as in Figure 3.9. As in Figure 3.9, the shaded area shows the range of performance across the two runs.

This current chapter focused on classifying relationships between objects in a scene, or more generally, the task of predicting a directed graph representing the relationships (relative placements) of objects in a scene, for the purpose of the bin picking problem in potentially cluttered scenes. Next, in Chapter 4, a problem which also relates to making inferences about object relations from images will be treated: the problem of estimating distances between two vehicles, given some wide-angle RGB videos of road traffic, for the purpose of detecting close-call events.

Chapter 4

Predicting Distances Between Vehicles

4.1 Problem Formulation

The scene being monitored by a camera is a part of the world being observed. Consider this world $\mathcal{W}$ to be a smooth four-dimensional manifold with boundary, in a space with coordinate system $(\hat{\imath}, \hat{\jmath}, \hat{k}, \hat{t})$, with the first three base vectors corresponding to spatial dimension, and $\hat{t}$ corresponding to the temporal dimension. Gravitation is present, and it has a corresponding vector $\vec{g}$ which points in the direction of gravity ("downward"). The camera observing the scene is defined to be infinitesimally small and located at $\vec{0}$ and pointing in direction $\hat{\imath}$, which is not orthogonal to $\vec{g}$; thus, $\hat{\imath} \cdot \vec{g} > 0$ (in addition, the horizontal direction of the camera is to be interpreted to be along $\hat{\jmath}$). Under these restrictions, the position and orientation of the camera is fixed throughout time. The world is composed of connected components, classified into three categories:

1. The ground on which the agents travel: roads and sidewalks.
2. Autonomous agents: vehicles, cyclists, and pedestrians.
3. Other structures which may obstruct the agents from the point of view of the camera: e.g., buildings, trees, traffic lights and signs, etc.

A description of each of these categories of components is provided below.

The ground is considered to be a 2D plane embedded in a 3D space $\mathbb{E}^3$: it is the plane $\{\vec{x} \mid \vec{g} \cdot \vec{x} = 0\}$. Note that since we are dealing with *videos* rather than still images, it is logical to also consider the passage of time; thus, it is possible to extend this definition to a plane embedded in a space with 4-dimensions (of the form $(\hat{\imath}, \hat{\jmath}, \hat{k}, \vec{t})$), and the definition of the ground stays the same (since the ground is assumed to be fixed in time).

The autonomous agents are connected components which correspond to either a vehicle, cyclist, or pedestrian. They are deemed to each have a *shape* that is variable with time, and which can differ between agents. Furthermore, each agent can *travel* in the scene, which is distinct from a change in shape. Each agent can thus be defined as a four-dimensional manifold a embedded in a four-dimensional space $\mathbb{R}^4$: three spatial dimensions and one temporal dimension. The shape of an agent a at a particular time t_i is $a_{t_i} \coloneqq \{[x_1, x_2, x_3, x_4]^{\mathrm{T}} \in a \mid x_4 = t_i\}$. In order to define the concept of *travelling*, it is important to define the *position* of an agent; as such, let the position of agent a at time t_i be defined as simply the centroid of a_{t_i} in space: let $p(a, t_i)$ denote the centroid of a_{t_i}.

The other structures can be defined, just like the autonomous agents, except that they have the special property that their shapes are constant with respect to time.

Of all the points in this world $\mathcal{W}$, only a subset is visible to the camera. A point $\vec{x} \in \mathbb{R}^4$ is visible to the camera if and only if $\vec{x} \cdot \hat{\imath} > 0 \ \wedge \ \nexists k \in \mathbb{R}_{>0} \ (k < 1 \ \wedge \ k\vec{x} \in \mathcal{W})$ (this theoretical camera is at $\vec{0}$ and has a field of view of 180 degrees). Note that this formulation does not allow for semi-transparent or translucent material. To this set of visible points $\mathcal{P}_v \subset \mathbb{R}^4$ can be associated a function $\mathbf{f} \colon \mathcal{P}_v \to \mathbb{R}^3$ which gives the RGB value $\mathbf{f}(\vec{x})$ of a point $\vec{x}$ which is visible the camera. The camera, however, only associates three dimensions to each measured pixel (two spatial and one temporal dimension), and thus this signal $\mathbf{f}$ is not directly given by the camera; instead, a derived signal on the projection $\mathcal{P}'_v$ of $\mathcal{P}_v$ onto the camera's sensor is available: $f \colon \mathcal{P}'_v \to \mathbb{R}^3$. This has some important consequences which will be discussed in Section 4.3.

The problem to be addressed in this chapter is to estimate, from f, the distance between any given pair of detected agents (a_1, a_2) for a given time t_i: $\|p(a_2, t_i) - p(a_1, t_i)\|$. Section 4.2 will give a short overview of the CARLA simulator [83], and Section 4.5 will present an architecture to estimate distances between vehicles in images outputted by this CARLA simulator, given some bounding box information.

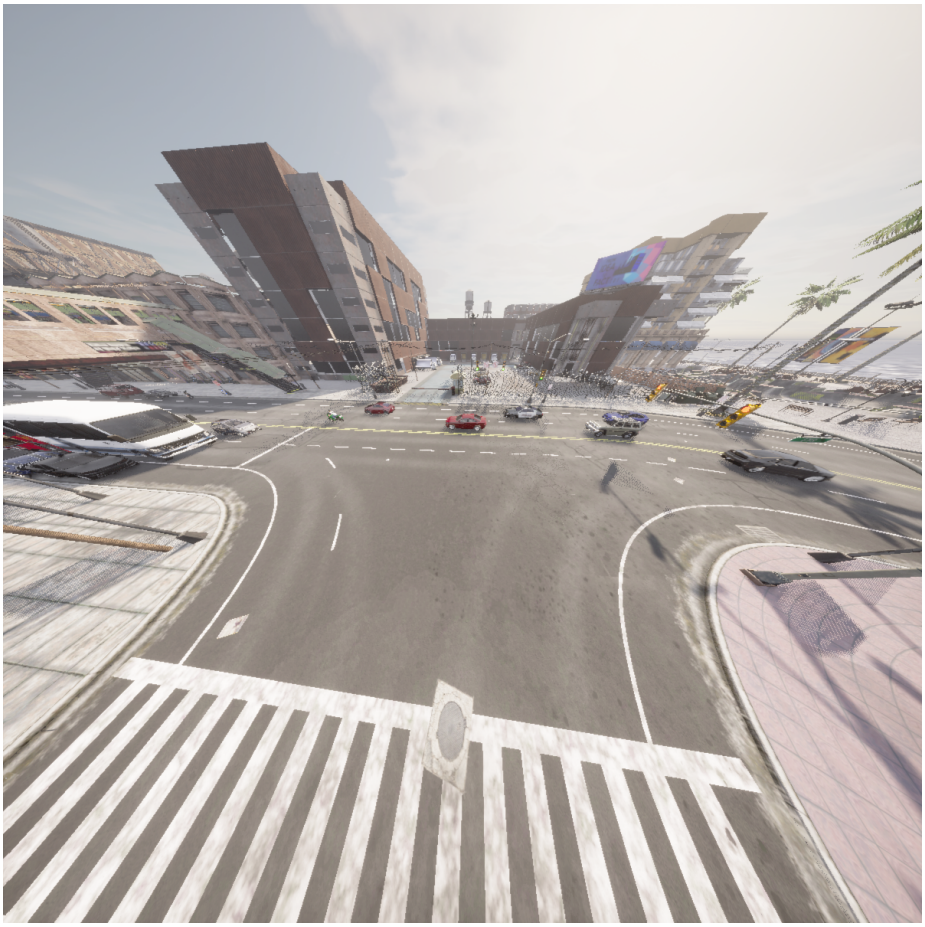


Figure 4.1: Image from a CARLA [83] simulation of traffic at an intersection, with a wide field of view. The resulting images are reasonably realistic. The agents in the scene are able to travel autonomously in the scene, and their movements and positions can be controlled in the simulation.

4.2 Dataset Creation with the CARLA Simulator

In order to train and evaluate a model to perform such a distance-estimation task, the CARLA [83] simulator was used. This publicly-available simulator is a fork of Unreal Engine 4*, and it was originally targeted towards autonomous-driving applications, but fortunately, there is much overlap between the autonomous-driving research field and the problem at hand in this chapter. Many of the features made available in CARLA are relevant to problem addressed here. There are some striking benefits to using a simulated environment; these include

- the ability to get perfect or near-perfect ground-truth information such as segmentation masks, bounding boxes, positions of agents, classes of agents;
- a high degree of control over the behaviours of the agents in the scene;
- the option to set a number of simulation parameters such as weather conditions, the position of the sun in the sky, the position of the camera, the number of pedestrians, cyclist, and vehicles; and
- the ability to generate data relatively quickly, consistently, and in large quantities (depending on computation power and efficiency).

However the CARLA simulator does have some limitations, which include

- lack of realism in some physical interactions between objects;
- less diversity, or randomness that is not representative of the real-world random distribution in the behaviour of real agents;
- limited choice in the types of camera lenses or projections available; and
- lack of some weather conditions (which are relatively difficult to simulate), such as heavy rain, thunderstorms, and notably snowfall and snow accumulation on the ground.

In order to generate simulate video footage, a (virtual) camera with a 150-degree field of view was placed in the scene such that it was above street lights and pointing downward, at an angle of 45 degrees. Figures 4.1 is an example of a wide-angle image generated using the CARLA simulator.

*Unreal Engine 4 is part of a popular series of gaming engines generally under the name *Unreal Engine*; it is capable of generating realistic renderings of 3D scenes.

4.3 Signal Acquired by a Wide-Angle Camera

The process of projecting any point $\vec{x} = [x_i, x_j, x_k, x_t]^{\mathrm{T}} \in \mathbb{R}^4$ onto a fixed camera sensor is completely independent of its time component x_t. As such, when analyzing the effect of projecting a set of points in $\mathbb{R}^4$ onto a camera sensor, the temporal dimension of each of these points can be ignored, and thus the analysis can be carried out in the smaller space that is $\mathbb{R}^3$, and conclusions derived from it can then also be applied to $\mathbb{R}^4$. Therefore, in this current section, the temporal dimension of the signal acquired by the camera will be ignored for discussing the effects of performing a projection in space from $\mathcal{P}_v$ to $\mathcal{P}'_v$.

The objective is to make an inference on sets of 3D points—the agents. More specifically, this inference is to be made on the function $f \coloneqq \mathbb{R}^2 \to \mathbb{R}^3$ (an RGB image) which is derived from $\mathbf{f}$. The image is a projection onto a 2D manifold, which implies that there is a loss of information. However, the metric that we are interested in (distance in 3D) is defined based on (spatial) 3D coordinates; thus, the model would have the task—at least implicitly—to estimate the lost dimension. In addition to this, the nature of the projection introduces some distortion in the image (compared to an orthographic projection); see Figure 4.2 and Figure 4.3. By embedding some prior knowledge about the geometry of the projection into the model (in the form of an inductive bias), it can be expected that the model would outperform a model in which information about the projection properties is not embedded.

To analyze the signal that is acquired, it might be helpful to start from the point of view of the flat sensor. For an RGB image, the signal acquired by a flat sensor is a function $f \coloneqq \mathbb{R}^2 \to \mathbb{R}^3$ which, for a given *pixel location* (x, y), outputs the RGB value $f(x, y)$. Now, there is emphasis on $f(x, y)$ corresponding to the RGB value of a specified *pixel* location rather than the location of the original scattering source.

Assuming that the lens of the camera is equivalent to a pinhole (which has an infinitely large depth of field), then under the ray model of light, the location at which any given ray of light entering the lens hits the sensor is solely dependent on the (angular) orientation of this ray. If the distance between the sensor and the lens is known, then the mapping between ray orientations and pixel locations can be determined. Let this mapping be denoted $l \colon \Theta \times \Phi \to \mathbb{R}^2$, where $\Theta \coloneqq \left]\frac{-\pi}{2}, \frac{\pi}{2}\right[$* and $\Phi \coloneqq]0, \pi[$, such that a ray of light with orientation (θ, ϕ) hits the sensor at the location $l(\theta, \phi)$.

*The notation $]\square_1, \square_2[$ refers to the open interval from the lower $\square_1$ to the upper bound $\square_2$: $\forall \square_1\ \forall \square_2\ \big(]\square_1, \square_2[= \{\square \mid \square_1 < \square \wedge \square < \square_2\}\big)$. Similarly, $]\square_1, \square_2] = \{\square \mid \square_1 < \square \leq \square_2\}$, and $[\square_1, \square_2[= \{\square \mid \square_1 \leq \square < \square_2\}$. Some authors use parentheses to denote an open interval: $(\square_1, \square_2)$; but this could be confused for the pair $(\square_1, \square_2)$.

(a) Near-orthographic projection (narrow field of view).

(b) Left-hand-side crop.

(c) Right-hand-side crop.

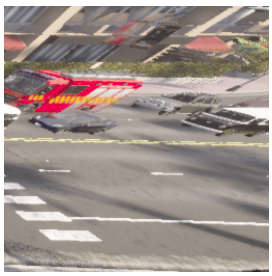
(d) Top-region crop.

Figure 4.2: An orthographic projection of 3D objects on a 2D plane is equivariant to translation of the object in 3D. (a): Illustration of a near-orthographic projection. (b), (c), & (d): The shapes of the vehicles appear the same, regardless of whether they are at the edges or at the centre.

Thus, the signal f is a signal on $\mathbb{R}^2$ and the signal $f \circ l$ is a signal on $\Theta \times \Phi$. The signal $f \circ l$ can be interpreted as a spherical image, as its domain is a set of *angular* coordinates. Now, considering the complications that could arise from dealing with a signal on a sphere, why would $f \circ l$ be used over f? Since the objective is to make some conclusions about the world *outside* of the camera and the lens, it is logical to consider using $f \circ l$ rather than f. By only using f directly, the design of a model would fail to incorporate the (completely known) cause of the image distortion, and the model would be left with the challenge of learning l, a function which is not relevant to making bounding box regression or class prediction. Let $\text{proj}_P \colon \mathbb{R}^3 \to \mathbb{R}^3$ be the function that performs a perspective (rectilinear) projection of a 3D point onto a plane $P \coloneqq \{\vec{x} \mid \vec{x} \cdot \hat{\imath} = D\}$, with D being the constant distance between the camera and the sensor represented by P. Let d_P be the distance

(a) Rectilinear projection (wide field of view).

(b) Left-hand-side crop.



(c) Right-hand-side crop.

(d) Crop near centre.

Figure 4.3: A rectilinear projection (with a large field of view) is not equivariant to translation of objects. The 2D projection of objects near the edge of the field of view is different than the projection of the same objects near, for example, the centre of the field of view.

between the projections of two given 3D points (on plane P):

$$d_P \colon \mathbb{R}^3 \times \mathbb{R}^3 \to \mathbb{R}$$

$$(\vec{x}, \vec{y}) \mapsto \|\mathrm{proj}_P(\vec{x}) - \mathrm{proj}_P(\vec{y})\|_2 \,.$$

Since we want to estimate the distance between two points, it would be ideal if d_P were invariant to changes in position of the camera (or positions of pairs of vehicles relative to the camera)—i.e., transformations in the special Euclidean group, SE(3)—because it would mean that the model would only need to learn to measure distances on the projected image directly (up to some scalar). Unfortunately, a perspective projection does not preserve $d_P(\vec{x}, \vec{y})$ for any two points $(\vec{x}, \vec{y})$ under all transformations $(R, \vec{\tau}) \in \mathrm{SE}(3)$. Only certain transformations in SE(3) preserve $d_P(\vec{x}, \vec{y})$: for a perspective projection, this is two one-dimensional manifolds corresponding to the set of rotations about $\hat{\imath}$, whereas for a spherical projection, this is a three-dimensional manifold corresponding to SO(3). *This*

means that there are two fewer dimensions of non-invariance to learn in the case of a spherical projection compared to a perspective projection.

Let the group SE(3) *act* on the set $\mathbb{R}^3$, with the group action being defined as

$$\mathrm{SE}(3) \times \mathbb{R}^3 \to \mathbb{R}^3$$

$$\big((R, \vec{\tau}), \vec{x}\big) \mapsto R\vec{x} + \vec{\tau}\,,$$

and which, $\forall g \in \mathrm{SE}(3)\ \forall \vec{x} \in \mathbb{R}^3$, is denoted by $g \cdot \vec{x}$. The operator notation "$\cdot$" is also used for the operator of the group SE(3), but what "$\cdot$" denotes in each case should be clear from the nature of each of its two arguments.

Proposition 4.1. *A transformation $g \in \mathrm{SE}(3)$ preserves distances after performing a perspective projection onto plane P if and only if it is a rotation about $\{a\hat{\imath} \mid a \in \mathbb{R}\}$ (with no translation) or it is a rotation about an axis orthogonal to $\hat{\imath}$, with angle $\|\vec{n}\| = \pi$ (with no translation); i.e., the set of transformations that preserve distances,*

$$\mathcal{T}_{\text{inv}} \coloneqq \Big\{ g \in \mathrm{SE}(3) \ \mid\ \forall (\vec{x}, \vec{y}) \in \mathbb{R}^3 \times \mathbb{R}^3 \ \big(d_P(\vec{x}, \vec{y}) = d_P(g \cdot \vec{x}, g \cdot \vec{y})\big) \Big\}\,,$$

is equal to

$$\Big\{ (R, \vec{0}) \in \mathrm{SE}(3) \ \mid\ \forall \vec{x} \in \mathbb{R}^3 \ \big(\vec{x} \cdot \hat{\imath} = \|\vec{x}\| \implies R\vec{x} = \pm\vec{x}\big) \Big\}\,.$$

Proposition 4.1 is meant to emphasize that there is only one rotation dimension (for each of the two one-dimensional manifolds), in the six-dimensional manifold SE(3), along which projected distance is invariant. This proposition will be proven below, starting on page 55, after introducing and proving some other (intermediate) propositions. Although a series of propositions and proofs are presented in the current chapter in order to support an argument and motivation for the use of spherical convolutions, these propositions and proofs are not intended to be a main result of this thesis work, and the author acknowledges that it is extremely likely that equivalent or stronger propositions may already have been published and proven. Some examples of related works include [84, 85].

Proposition 4.2. *For any rotation R (with rotation vector $\vec{n}$ and rotation angle $\|\vec{n}\|$) satisfying the condition*

$$0 < \|\vec{n}\| < \pi \ \wedge\ \vec{n} \cdot \hat{\imath} = 0\,, \tag{4.1}$$

and for any pair of distinct points $(\vec{b}, \vec{c})$ *satisfying the condition*

$$\|\vec{b}\| = \|\vec{c}\| \quad \wedge \quad \forall \vec{x} \in \left\{\vec{b}, \vec{c}\right\} \left(\vec{x} \cdot \vec{n} = 0 \ \wedge \ \vec{x} \cdot (\hat{\imath} \times \vec{n}) > 0 \ \wedge \ \vec{x} \cdot \hat{\imath} > 0\right) \tag{4.2}$$

(see Figure 4.4), the following is true:

$$R\vec{b} \cdot (\hat{\imath} \times \vec{n}) > 0 \quad \wedge \quad R\vec{c} \cdot (\hat{\imath} \times \vec{n}) > 0 \quad \Longrightarrow \quad d_P(R\vec{b}, R\vec{c}) < d_P(\vec{b}, \vec{c}) \,.$$

Proof of Proposition 4.2. Consider any two distinct points $\vec{b}$ and $\vec{c}$ satisfying condition (4.2) and with $\vec{b} \cdot (\hat{\imath} \times \vec{n}) > \vec{c} \cdot (\hat{\imath} \times \vec{n})$, as illustrated in Figure 4.4, and their respective projections $\vec{b^P} = [b^P_{\hat{\imath}} \ \ b^P_{\hat{\jmath}} \ \ b^P_{\hat{k}}]^{\mathrm{T}}$ and $\vec{c^P} = [c^P_{\hat{\imath}} \ \ c^P_{\hat{\jmath}} \ \ c^P_{\hat{k}}]^{\mathrm{T}}$ onto plane P (which is at a distance D from the origin $\vec{0}$). Let

$$\Delta \ : \ \varepsilon \mapsto \frac{D}{\tan(\varepsilon)} \,, \tag{4.3}$$

then

$$b^P_{\hat{\imath}} = \Delta(\varepsilon_1) \tag{4.4}$$

$$c^P_{\hat{\imath}} = \Delta(\varepsilon_1 + \varepsilon_2) \tag{4.5}$$

$$d_P(\vec{b}, \vec{c}) = b^P_{\hat{\imath}} - c^P_{\hat{\imath}} = \Delta(\varepsilon_1) - \Delta(\varepsilon_1 + \varepsilon_2) \,. \tag{4.6}$$

Applying R on $\vec{b}$ and $\vec{c}$ is equivalent to increasing ε_1. As ε_1 gets closer to $\frac{\pi}{2} - \varepsilon_2$, the scalar $d_P(\vec{b}, \vec{c})$ becomes smaller, since Δ' (first derivative of Δ) is negative over $]0, \pi/2[$ and Δ'' is positive over this same interval. ■

Proposition 4.3. *For any rotation* R *satisfying condition (4.1), there exists a triplet of distinct points* $(\vec{a}, \vec{b}, \vec{c})$ *for which*

$$d_P\left(\vec{a}, \vec{b}\right) = d_P\left(R\vec{b}, R\vec{c}\right) > d_P\left(\vec{b}, \vec{c}\right) = d_P\left(R\vec{a}, R\vec{b}\right) . \tag{4.7}$$

Proof of Proposition 4.3. The objective here is to identify three points which are on a circle centred at $\vec{0}$, and which satisfy (4.7). Consider any rotation R satisfying condition (4.1), and let $\theta \coloneqq \|\vec{n}\|$, where $\vec{n}$ is the rotation vector of R. It will be shown that for any such R,

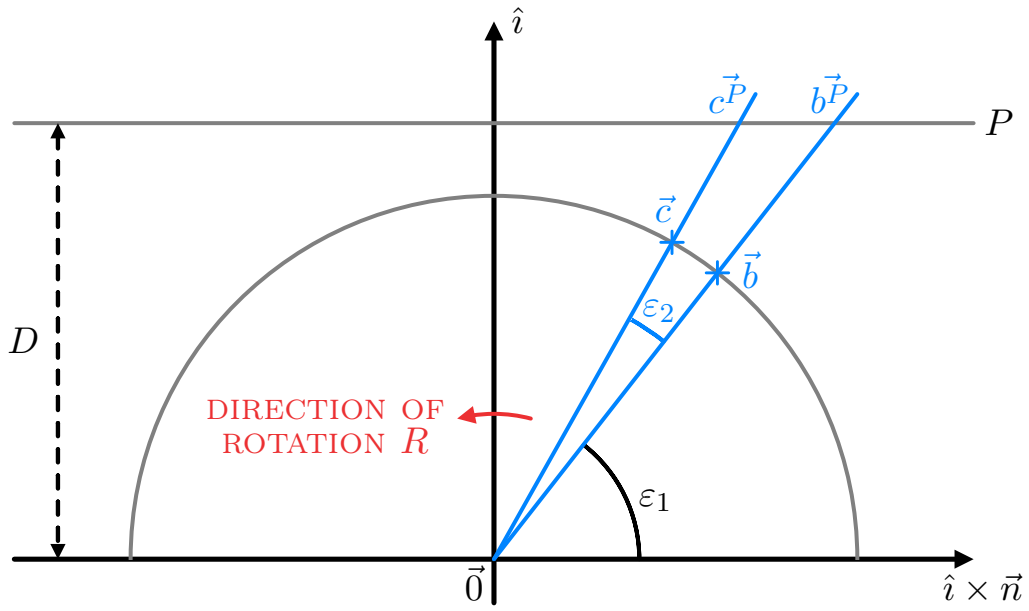


Figure 4.4: Illustration for Proposition 4.2.

there exists values ε_1 and ε_2 satisfying all of the following conditions simultaneously:

$$0 < \varepsilon_2 < \varepsilon_1 \tag{4.8}$$

$$\varepsilon_1 + \theta + \varepsilon_2 < \pi \tag{4.9}$$

$$\frac{\pi}{2} < \theta + \varepsilon_1 - \varepsilon_2 \tag{4.10}$$

$$\varepsilon_1 + \varepsilon_2 < \frac{\pi}{2}\,. \tag{4.11}$$

(Notice, by inspecting Figure 4.5, that when (4.8), (4.9), (4.10), and (4.11) are satisfied, $\vec{a}$, $\vec{b}$, and $\vec{c}$ are guaranteed to be in the top right quadrant of the figure, $\vec{a'}$, $\vec{b'}$, and $\vec{c'}$ are guaranteed to be in the top left quadrant of the figure, and it is guaranteed that $\vec{a} \cdot (\hat{\imath} \times \vec{n}) > \vec{b} \cdot (\hat{\imath} \times \vec{n}) > \vec{c} \cdot (\hat{\imath} \times \vec{n})$.) Assume that

$$\varepsilon_1 = \frac{\pi - \theta}{2} \quad \wedge \quad \varepsilon_2 \in \,]0, u_2[\,, \tag{4.12}$$

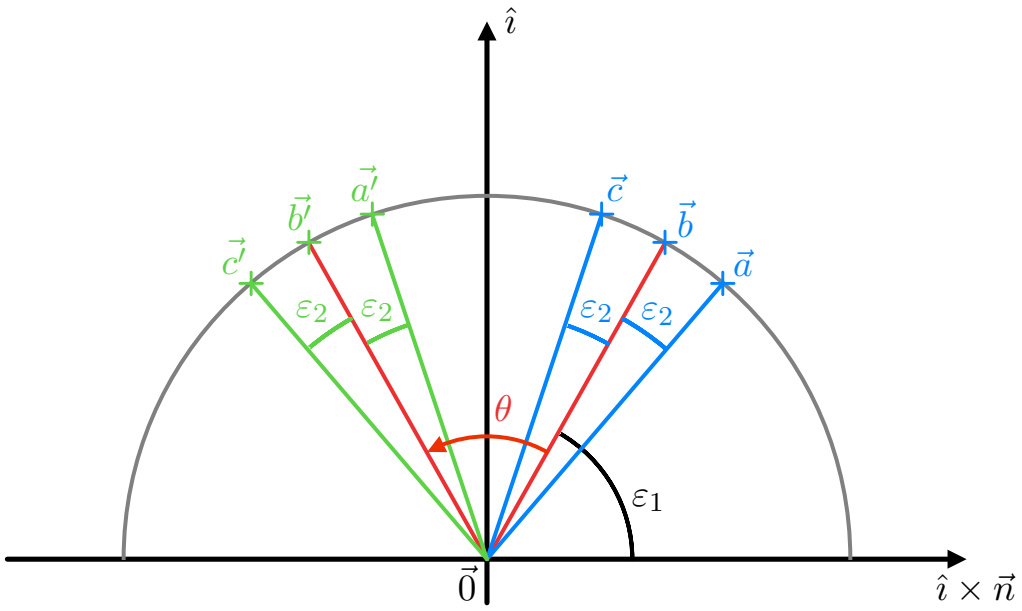


Figure 4.5: Illustration for Proposition 4.3.

where

$$u_2 \coloneqq \min\left(\varepsilon_1, \varepsilon_1 + \theta - \frac{\pi}{2}\right).$$

Any such $(\varepsilon_1, \varepsilon_2)$ satisfies (4.8), since $0 < \varepsilon_2 < u_2 \leq \varepsilon_1$. Condition (4.9) is also satisfied since

$$\begin{cases} \varepsilon_1 = \dfrac{\pi - \theta}{2} \\ \varepsilon_2 < \min\left(\varepsilon_1, \varepsilon_1 + \theta - \dfrac{\pi}{2}\right) \end{cases} \implies \varepsilon_1 + \varepsilon_2 < 2\varepsilon_1 = \pi - \theta$$

$$\implies \varepsilon_1 + \theta + \varepsilon_2 < \pi\,.$$

Condition (4.10) is satisfied, as

$$\varepsilon_2 < \min\left(\varepsilon_1, \varepsilon_1 + \theta - \frac{\pi}{2}\right) \leq \theta + \varepsilon_1 - \frac{\pi}{2}\,.$$

Finally, condition (4.11) is also satisfied:

$$\varepsilon_2 < u_2 \leq \varepsilon_1 = \frac{\pi-\theta}{2} \implies \varepsilon_1 + \varepsilon_2 < \frac{\pi-\theta}{2} + \min\left(\frac{\pi-\theta}{2}, \frac{\theta}{2}\right) = \min\left(\pi-\theta, \frac{\pi}{2}\right) \leq \frac{\pi}{2}\,.$$

Note that the interval for ε_2 in (4.12) is non-empty for $0 < \theta < \pi$, since $0 < \varepsilon_1$ and, for $\varepsilon_1 = (\pi-\theta)/2$,

$$\varepsilon_1 + \theta - \frac{\pi}{2} = \frac{\theta}{2} > 0\,.$$

Therefore, $\exists\varepsilon_1\ \exists\varepsilon_2\ ((4.8)\ \wedge (4.9)\ \wedge (4.10)\ \wedge (4.11))$.

For such a pair $(\varepsilon_1, \varepsilon_2)$, consider points $\vec{a}$, $\vec{b}$, and $\vec{c}$ illustrated in Figure 4.5. Based on Proposition 4.2,

$$d_P(\vec{a},\vec{b}) > d_P(\vec{b},\vec{c}) \tag{4.13}$$

(consider a rotation R_{ε_2} with rotation vector $\varepsilon_2\vec{n}/\|\vec{n}\|$) and

$$d_P(\vec{a'},\vec{b'}) < d_P(\vec{b'},\vec{c'}) \tag{4.14}$$

(consider the rotation $R_{\varepsilon_2}^{-1}$). Notice that there is reflection symmetry about the plane $\left\{\vec{x} \;\middle|\; \vec{x}\cdot(\hat{\imath}\times\vec{n}) = 0\right\}$ between $\vec{a}$ and $\vec{a'}$, between $\vec{b}$ and $\vec{b'}$, and between $\vec{c}$ and $\vec{c'}$ (since $\varepsilon_1 + \theta/2 = \pi/2$); thus,

$$d_P(\vec{b'},\vec{c'}) = d_P(\vec{a},\vec{b})\ \wedge\ d_P(\vec{b},\vec{c}) = d_P(\vec{a'},\vec{b'})\,.$$

Since $\vec{a'} = R\vec{a}$, $\vec{b'} = R\vec{b}$, and $\vec{c'} = R\vec{c}$, Proposition 4.3 is true.

■

Proof of Proposition 4.1. First, note that any transformation $(R, \vec{\tau}) \in \mathrm{SE}(3)$ can be decomposed. In particular, consider the rotation component $R_{\hat{\imath}} \in \mathcal{R}_{\hat{\imath}}$ (where $\mathcal{R}_{\hat{\imath}}$ is the set of rotations with rotation axis parallel to $\hat{\imath}$) and consider the translation component $\vec{\tau}_{\hat{\imath}}$ such that $\vec{\tau}_{\hat{\imath}} \times \hat{\imath} = \vec{0}$. Let $R_{\mathrm{r}} \coloneqq RR_{\hat{\imath}}^{-1}$ and $\vec{\tau}_{\mathrm{r}} \coloneqq \vec{\tau} - \vec{\tau}_{\hat{\imath}}$ correspond to the remaining rotation and

remaining translation respectively. Thus, any $(R, \vec{\tau})$ can be decomposed as follows:

$$(R, \vec{\tau}) = (R_{\mathrm{r}} R_{\hat{\imath}}, \vec{\tau}_{\mathrm{r}} + \vec{\tau}_{\hat{\imath}}) = (I, \vec{\tau}_{\mathrm{r}} + \vec{\tau}_{\hat{\imath}}) \cdot (R_{\mathrm{r}} R_{\hat{\imath}}, \vec{0}) = (I, \vec{\tau}_{\mathrm{r}}) \cdot (I, \vec{\tau}_{\hat{\imath}}) \cdot (R_{\mathrm{r}}, \vec{0}) \cdot (R_{\hat{\imath}}, \vec{0}) \tag{4.15}$$

$$\mathcal{C}_4 \coloneqq \left\{ (I, \vec{\tau}_{\mathrm{r}}) \;\middle|\; \vec{\tau}_{\mathrm{r}} \cdot \hat{\imath} = 0 \right\} \quad \mathcal{C}_3 \coloneqq \left\{ (I, \vec{\tau}_{\hat{\imath}}) \;\middle|\; \vec{\tau}_{\hat{\imath}} \cdot \hat{\imath} = \|\vec{\tau}_{\hat{\imath}}\| \right\}$$

$$\mathcal{C}_2 \coloneqq \left\{ (R_{\mathrm{r}}, \vec{0}) \;\middle|\; \forall \vec{x} \in \mathbb{R}^3 \; (\vec{x} \cdot \hat{\imath} = 0 \implies R_{\mathrm{r}} \vec{x} = \vec{x}) \right\}$$

$$\mathcal{C}_1 \coloneqq \left\{ (R_{\hat{\imath}}, \vec{0}) \;\middle|\; \forall \vec{x} \in \mathbb{R}^3 \; \left(\vec{x} \cdot \hat{\imath} = \|\vec{x}\| \implies R_{\hat{\imath}} \vec{x} = \vec{x}\right) \right\}$$

$$\mathcal{C}_{4,3,2,1} \coloneqq \mathcal{C}_4 \times \mathcal{C}_3 \times \mathcal{C}_2 \times \mathcal{C}_1$$

$$\forall (R, \vec{\tau}) \in \mathrm{SE}(3) \; \exists (c_4, c_3, c_2, c_1) \in \mathcal{C}_{4,3,2,1} \; \left((R, \vec{\tau}) = c_4 \cdot c_3 \cdot c_2 \cdot c_1 \right). \tag{4.16}$$

Second, note that any $(R_{\hat{\imath}}, \vec{0}) \in \mathcal{C}_1$ does not change the distance after projection, since $\forall \vec{x} \in \mathbb{R}^3 \; (\mathrm{proj}_P(R_{\hat{\imath}} \vec{x}) = R_{\hat{\imath}} \, \mathrm{proj}_P(\vec{x}))$ and transformations from SE(3) do not change distances in 3D, thus $\forall \vec{x} \in \mathbb{R}^3 \; \forall \vec{y} \in \mathbb{R}^3$

$$\left\| \mathrm{proj}_P(R_{\hat{\imath}} \vec{y}) - \mathrm{proj}_P(R_{\hat{\imath}} \vec{x}) \right\|_2 = \left\| R_{\hat{\imath}} \, \mathrm{proj}_P(\vec{y}) - R_{\hat{\imath}} \, \mathrm{proj}_P(\vec{x}) \right\|_2 = \left\| \mathrm{proj}_P(\vec{y}) - \mathrm{proj}_P(\vec{x}) \right\|_2 . \tag{4.17}$$

Therefore,

$$\forall (\vec{x}, \vec{y}) \in \mathbb{R}^3 \times \mathbb{R}^3 \; \left(d_P(R_{\hat{\imath}} \vec{x}, R_{\hat{\imath}} \vec{y}) = d_P(\vec{x}, \vec{y}) \right). \tag{4.18}$$

This shows that component c_1 (of (4.16)) does not change distances between any two projected points:

$$\forall c_1 \in \mathcal{C}_1 \; \forall (\vec{x}, \vec{y}) \; \left(d_P(c_1 \cdot \vec{x}, c_1 \cdot \vec{y}) = d_P(\vec{x}, \vec{y}) \right). \tag{4.19}$$

For the terms c_2 and c_3 of (4.16): let $\vec{n}_2$ be the rotation vector of R_r (of c_2). Based on Proposition 4.3, a c_2 with $0 < \|\vec{n}_2\| < \pi$ *does* change the distance between at least two points; more specifically, for a triplet of distinct points $(\vec{a}, \vec{b}, \vec{c})$ satisfying (4.7), $0 < \|\vec{n}_2\| < \pi$ implies that

$$d_P\Big(c_2 \cdot \vec{b}, c_2 \cdot \vec{c}\Big) > d_P\Big(\vec{b}, \vec{c}\Big) \tag{4.20}$$

$$d_P\Big(c_2 \cdot \vec{a}, c_2 \cdot \vec{b}\Big) < d_P\Big(\vec{a}, \vec{b}\Big). \tag{4.21}$$

Thus, such a c_2 *increases* the projected distance between $\vec{b}$ and $\vec{c}$, and it *decreases* the projected distance between $\vec{a}$ and $\vec{b}$. If $\|\vec{n}_2\| = \pi$, then the projected distances are unchanged, since $\forall \vec{x_1} \in \mathbb{R}^3 \; \forall \vec{x_2} \in \mathbb{R}^3$, the points $R_r\vec{x_1}$ and $R_r\vec{x_2}$ will have the same projections as $R_{\hat{\imath},\pi}\vec{x_1}$ and $R_{\hat{\imath},\pi}\vec{x_1}$ respectively, where $R_{\hat{\imath},\pi}$ is a rotation of π around $\hat{\imath}$ (note that $R_r\vec{x_1}$, $R_{\hat{\imath},\pi}\vec{x_1}$, and $\vec{0}$ are aligned, and $R_r\vec{x_2}$, $R_{\hat{\imath},\pi}\vec{x_2}$, and $\vec{0}$ are aligned).

Note that for c_3, based on (4.3) and (4.6), if $\vec{\tau}_{\hat{\imath}} \cdot \hat{\imath} > 0$, then c_3 decreases *both* projected distances (this corresponds to a decrease in D in (4.3)), and if $\vec{\tau}_{\hat{\imath}} \cdot \hat{\imath} < 0$, then c_3 increases *both* projected distances. *In either case,* (4.20) *or* (4.21) *(or both) will still be true after applying any* $c_3 \in \mathcal{C}_3$:

$$\begin{cases} \vec{\tau}_{\hat{\imath}} \cdot \hat{\imath} \geq 0 \\ (4.21) \end{cases} \implies d_P\left(c_3 \cdot c_2 \cdot \vec{a}, c_3 \cdot c_2 \cdot \vec{b}\right) \leq d_P\left(c_2 \cdot \vec{a}, c_2 \cdot \vec{b}\right) < d_P\left(\vec{a}, \vec{b}\right)$$

$$\begin{cases} \vec{\tau}_{\hat{\imath}} \cdot \hat{\imath} \leq 0 \\ (4.20) \end{cases} \implies d_P\left(c_3 \cdot c_2 \cdot \vec{b}, c_3 \cdot c_2 \cdot \vec{c}\right) \geq d_P\left(c_2 \cdot \vec{b}, c_2 \cdot \vec{c}\right) > d_P\left(\vec{b}, \vec{c}\right).$$

Therefore, for a triplet of distinct points $(\vec{a}, \vec{b}, \vec{c})$ satisfying (4.7), there is no $c_3 \cdot c_2$ that simultaneously preserves the projected distances of $(\vec{a}, \vec{b})$ and $(\vec{b}, \vec{c})$, except when (i) c_3 is the identity element, e, of SE(3), and (ii) $c_2 = e$ or the angle of rotation of c_2 is π.

For the term c_4 of (4.16): Note that for any pair of distinct points $(\vec{x}, \vec{y})$ in $\{\vec{x} \mid 0 \neq |\vec{x} \cdot \hat{\imath}| = \|\vec{x}\|\}$, the projected distance is affected by c_4 but unaffected by c_3, c_2, and c_1.

Therefore, for any transformation $(R, \vec{\tau}) = c_4 \cdot c_3 \cdot c_2 \cdot c_1$ in SE(3) that preserves distances after projection,

- $c_4 = e$ because any two distinct points on $\{\vec{x} \mid 0 \neq |\vec{x} \cdot \hat{\imath}| = \|\vec{x}\|\}$ have their projected distances unaffected by $c_3 \cdot c_2 \cdot c_1$ but have their projected distances affected by c_4 (on its own) when $c_4 \neq e$ and therefore must have their projected distances affected by $c_4 \cdot c_3 \cdot c_2 \cdot c_1$ when $c_4 \neq e$;
- c_1 does not affect projected distance, so there is no restriction on c_1 (other than being in $\mathcal{C}_1$); and
- $c_3 \cdot c_2$ must not, on its own, affect projected distances, which is the case if and only if
  1. $c_3 = e$ and
  2. $c_2 = e$ or the angle of rotation of c_2 is π.

In summary, for any transformation $c \in \mathrm{SE}(3)$ decomposed as $c_4 \cdot c_3 \cdot c_2 \cdot c_1$, and where $\vec{n_2}$ denotes the rotation vector of c_2,

$$d_P(x, y) = d_P(c \cdot x, c \cdot y) \implies (c_2 = e \ \vee \ \|\vec{n_2}\| = \pi) \ \wedge \ c_3 = e \ \wedge \ c_4 = e \,.$$

The converse of this statement is also true since any $c_1 \in \mathcal{C}_1$ does not affect projected distance; thus, Proposition 4.1 is true. ■

This current section has focused on analyzing the nature of the signal acquired by a wide-angle camera, and it was concluded that for a rectilinear projection, there is only one rotational degree of freedom along which projected distances are invariant (more precisely, along two one-dimensional manifolds, one of which is irrelevant in practice), and in contrast, for a spherical projection there are two rotational degrees of freedom along which projected distances are invariant. This is an important consideration, as the distance-prediction model should be invariant to any type of rotation in 3D, and thus, if the signal is interpreted by the model as being spherical rather than the result of a rectilinear projection onto a plane, then invariance to an additional rotational degree of freedom would already be embedded in the model, before any training even begins. Next, in Section 4.4, experimental results to investigate the potential gain in performance from the use of spherical convolutions will be presented.

4.4 Spherical Convolutional Layers

In order to experimentally study any improvement in performance that is achievable using spherical convolutions instead of planar convolutions, one could define an estimation task to be performed on images acquired by a pinhole camera and compare model performance with spherical convolutions to model performance with regular planar convolutions, i.e., when the model treats the signal as spherical or when it treats the signal as planar. Since the problem of this current chapter focuses on distance estimation, it would be appropriate to compare models which estimate distances. For a simple and well-controlled evaluation, a dataset for such a task should have a well-defined "distance" to estimate for each image it contains. To this end, a synthetic dataset was created in which each image was a binary mask of a two-dimensional disk of random curved diameter (in the range of $[\pi/8, \pi/2[\,)$ on a sphere. (Here, "disk" actually refers to a spherical cap, and "diameter" refers to the arc length on the sphere; i.e., a disk of diameter d centred at $\hat{\imath}$ on the unit sphere is the set $\{\vec{x} \ \mid \ \angle(\hat{\imath}, \vec{x}) < d\}$, where $\angle(\vec{a}, \vec{b})$ denotes the angle between $\vec{a}$ and $\vec{b}$ in radians.) Although disks on the sphere could be drawn at any angle, the possible positions of the centres of the

disks on the sphere were limited to be within $\pm 45°$ in latitude and $\pm 72°$ in longitude in order limit distortions to levels closer to what would be expected with a typical wide-angle camera. In order to store and process the image in a discrete fashion, the sphere was pixelated according to the Driscoll and Healy grid [69, 74] such that each image was 32 by 32 pixels. As illustrated in Figure 4.6, when projected onto a plane according to the Driscoll and Healy grid, these disks appear distorted, especially at the top and bottom edges of the frame.

To evaluate the ability of a detection model in estimating *distances*, the task was defined to be the estimation, for each image, of the (curved) diameter of the disk on the sphere. By processing these images on the sphere, it would be expected that these distances would be learned more readily, as they should be invariant to the position of the disks, while on the plane, these distances could be different for two disks of the same diameter but at different locations.

The architecture used is illustrated in Figure 4.7; this architecture was based on the RPN described in Faster-RCNN [42], but it includes some modifications. Two RPN versions were implemented, both of which were using bounding *circles* rather than bounding *boxes*, as representations and coordinates of circles on the sphere are simpler to process. The key difference between these two models was the use of spherical convolutional layers in one model and planar convolutional layers in the other model: the layers which differed between these models are labelled in red in Figure 4.7. Note that the planar convolutions were directly preformed on the planar representation of the Driscoll and Healy grid, which is not the rectilinear projection described in Section 4.3 (and thus not the result of a pinhole camera); however, in the planar representation of the Driscoll and Healy grid, disks are still distorted, so conclusions about differences in performance using planar convolutions versus spherical convolutions could still be used to support arguments in favour of or against spherical convolutions.

Figure 4.8 summarizes the results obtained for each model across multiple runs. With the percentage of absolute error in the vertical axis and the amount of training time in the horizontal axis, a model with higher performance (in terms of training time to reach a given error threshold) would have training curves that are closer to the bottom-left corner of the plot.

Since substantial variability can be present between runs of a same model due to randomness, simply studying the average curve might not be as insightful as studying the Pareto front of the model over several runs. In order to mitigate the effects of randomness in the initialization of the weights of the model as well as the generated images, the 20^{th} percentile of error for each epoch was plotted. For each model, different numbers of total

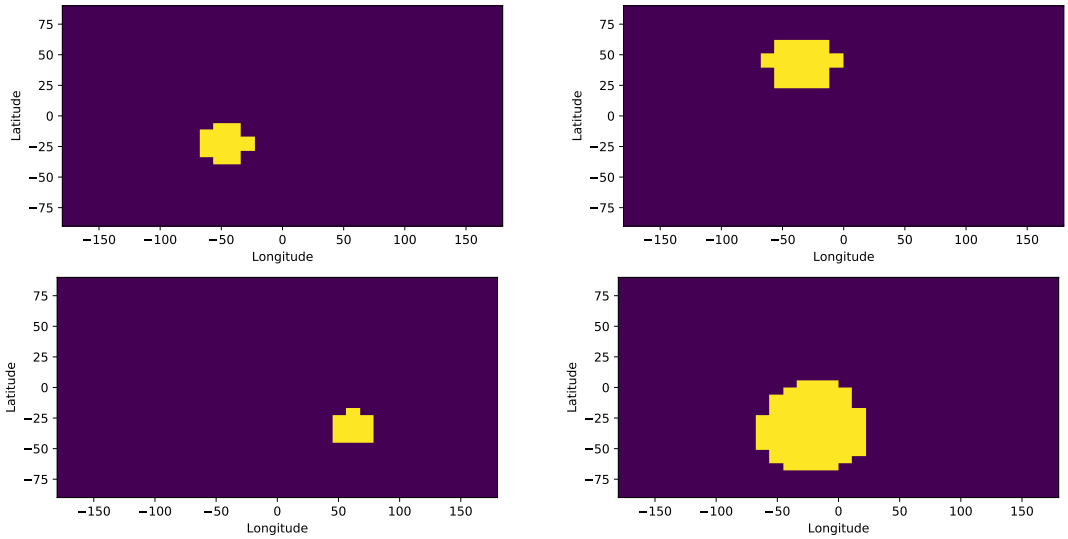


Figure 4.6: Examples of generated images for training and evaluation: Circles centers within $\pm 45°$ in latitude and $\pm 72°$ in longitude.

trainable parameters were tested by changing the number of channels in the convolutional layers, which was parameterized through a channel hyperparameter. See Table 4.1 for the correspondence between this channel hyperparameter and the total number of trainable parameters in each model.

Upon observing Figure 4.8, one can note that, for the tested resolution, the spherical version of the RPN generally outperformed the planar RPN model, in spite of the spherical model being more computationally demanding per epoch (the planar RPN plot corresponds to three times as many iterations per run than the spherical RPN plot). For example, for only 10 seconds of training, the 20^{th} percentile of error was at about 7% for the best channel hyperparameter in the case of the spherical RPN, while for the same amount of training time, the 20^{th} percentile of error was at more than 10% for the planar RPN.

To conclude this section, substantial improvements in distance estimation accuracy were observed for the task of estimating curved diameters of disks on a sphere (spherical caps) for images of 32 by 32 pixels when using spherical convolutional layers instead of planar convolutional layers. Section 4.5 will introduce a proposed module architecture to estimate distances between vehicles given images and bounding boxes outputted by CARLA.

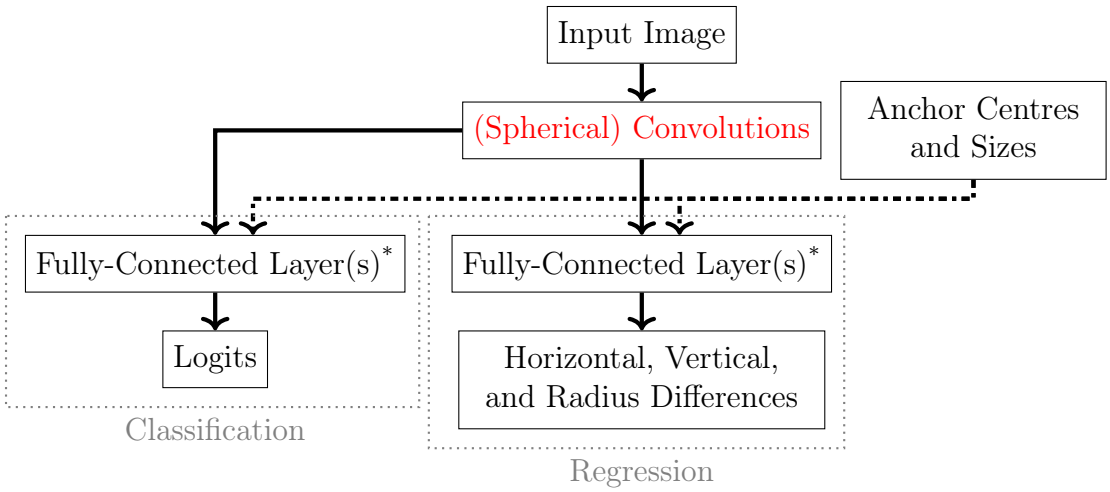
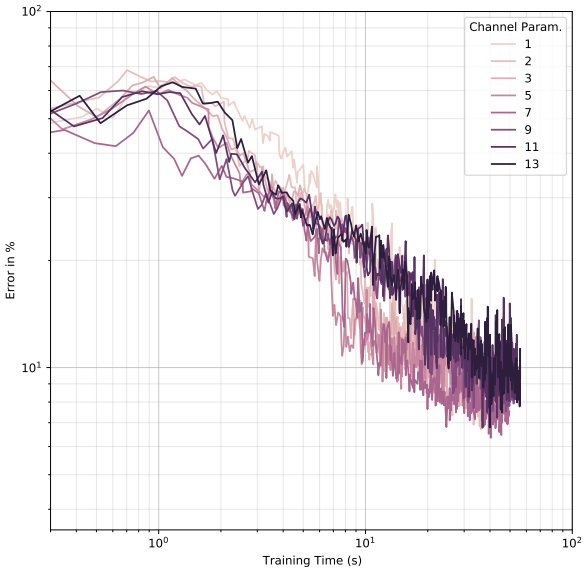


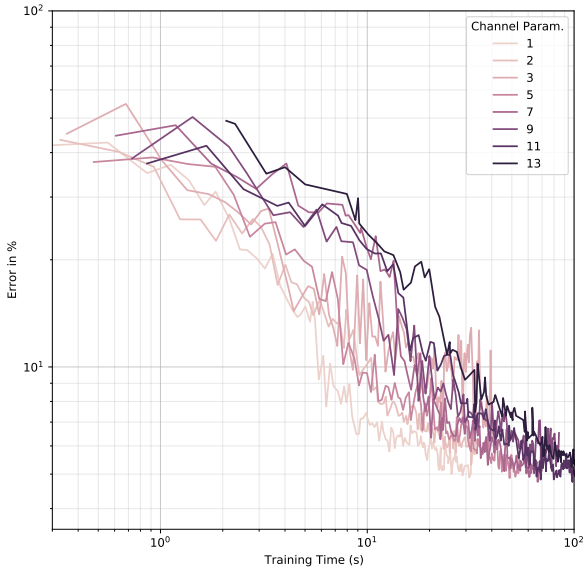
Figure 4.7: RPN Architecture used. The two RPN model versions compared mainly differed at the layers labelled in red. In one version, the convolutions were planar; in the other, the convolutions were spherical. *For each anchor position; implemented as convolution layers with kernel sizes of 1.

Table 4.1: Mapping between channel hyperparameter and trainable-parameter count; a higher channel hyperparameter value corresponds to more trainable parameters. A channel hyperparameter of 3 for the spherical model produces about the same number of trainable parameters as a channel hyperparameter of 11 for the planar model.

	Number of trainable parameters	
Channel hyperparam.	Spherical model	Planar model
1	17 833	5 417
2	19 567	5 864
3	21 685	6 409
5	32 095	9 122
7	45 577	12 619
9	62 131	16 900
11	81 757	21 965
13	104 455	27 814



(a) Planar RPN: 20^{th} percentile of error for each epoch.



(b) Spherical RPN: 20^{th} percentile of error for each epoch.

Figure 4.8: Planar vs. spherical RPN. The spherical RPN architecture is generally able to achieve lower absolute error in curved diameter estimation for a fixed amount of training time.

4.5 Distance Estimation from Bounding Boxes and Images

In an effort to develop a deep-learning model to estimate distances between vehicles appearing in a CARLA image, a module $\mathcal{D}_d$ (without a detector) was developed to predict 3D positions and distance between two detected vehicles, given access to bounding-box information and the original image; this was assuming that reasonably accurate bounding boxes could eventually be produced by a state-of-the-art detector. For a pair of vehicles (v_A, v_B), the distance-estimation module's inputs were as follows: the RGB image, the bounding-box coordinates $B(v_A) \in \mathcal{B} \subset \mathbb{R}^4$ of v_A, and the bounding-box coordinates $B(v_B) \in \mathcal{B} \subset \mathbb{R}^4$ of v_B. The outputs of the module $\mathcal{D}_d$ are the estimated 3D position of v_A (i.e., $\hat{p}(v_A)$), the estimated 3D position of v_B (i.e., $\hat{p}(v_B)$), and the estimate, $\hat{d}(v_A, v_B)$, of the distance $d(v_A, v_B) \coloneqq \|p(v_B) - p(v_A)\|$ between the two vehicles. The architecture of $\mathcal{D}_d$ is summarized in Figure 4.9.

$$\mathcal{D}_d \colon \mathcal{M} \times \mathcal{B} \times \mathcal{B} \to \mathbb{R}^3 \times \mathbb{R}^3 \times \mathbb{R}$$

The main questions that were investigated experimentally with this model architecture were the following:

1. How much precision in distance estimation can be obtained from only using bounding boxes as inputs, and how much gain in precision can be attributed to the use of the pixel information enclosed by the bounding boxes?

2. How sensitive would such a model be to error or noise in the bounding-box coordinates, and to what extent would sensitivity to noise be dependent on the use or omission of pixel information for inference?

In order to train and evaluate the model, only vehicles no more than 30 metres away from the camera were considered to be detected, and their ground-truth 2D bounding boxes were derived from CARLA. The CARLA simulator provides access to oriented 3D bounding boxes for vehicles: for any vehicle v_a, this is eight points $B_{3\mathrm{D}}(v_a) = (\vec{b}_1, \vec{b}_2, \ldots, \vec{b}_8)$ in 3D defining a hexahedron within which all of v_a is contained. From this, non-oriented 2D four-vertex bounding boxes (described in the 2D coordinate system $(\hat{\jmath}, \hat{k})$) of given vehicles were computed. As these 2D four-vertex bounding boxes are non-oriented and their edges are each either parallel to $\hat{\jmath}$ or $\hat{k}$, they can each be described by simply two vectors, each in

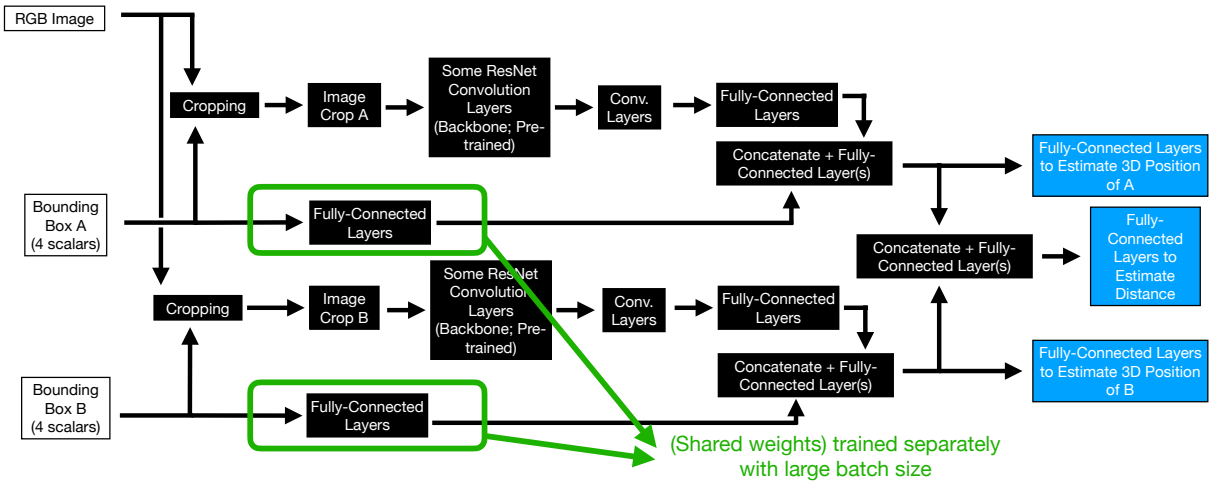


Figure 4.9: Architecture of $\mathcal{D}_d$. The two branches illustrated (each for one of the two vehicles) are in fact the same branch (in code) and share the same trainable parameters. Pre-trained ResNet convolutions were used, while the fully-connected layers identified in green were trained separately by omitting the RGB image data, which allowed for larger batch sizes to be easily used.

2D: $\vec{b}_{\text{tr}}$ at the top-right corner in the $(\hat{\jmath}, \hat{k})$ 2D coordinate system, and $\vec{b}_{\text{bl}}$ at the bottom-left corner:

$$\vec{b}_{\text{bl}} = \begin{bmatrix} \min\left(\hat{\jmath} \cdot \text{proj}_P(\vec{b}_1), \hat{\jmath} \cdot \text{proj}_P(\vec{b}_2), \ldots, \hat{\jmath} \cdot \text{proj}_P(\vec{b}_8)\right) \\ \min\left(\hat{k} \cdot \text{proj}_P(\vec{b}_1), \hat{k} \cdot \text{proj}_P(\vec{b}_2), \ldots, \hat{k} \cdot \text{proj}_P(\vec{b}_8)\right) \end{bmatrix}$$

$$\vec{b}_{\text{tr}} = \begin{bmatrix} \max\left(\hat{\jmath} \cdot \text{proj}_P(\vec{b}_1), \hat{\jmath} \cdot \text{proj}_P(\vec{b}_2), \ldots, \hat{\jmath} \cdot \text{proj}_P(\vec{b}_8)\right) \\ \max\left(\hat{k} \cdot \text{proj}_P(\vec{b}_1), \hat{k} \cdot \text{proj}_P(\vec{b}_2), \ldots, \hat{k} \cdot \text{proj}_P(\vec{b}_8)\right) \end{bmatrix}$$

$$B_{\text{2D}}(v_a) = \left(\vec{b}_{\text{bl}}, \vec{b}_{\text{tr}}\right).$$

Note that this method of constructing the 2D non-oriented bounding boxes is not perfect and will typically generate 2D bounding boxes which are too large (but not too small)—as an

comparable example, imagine a unit sphere bounded by the octahedron $\{\vec{x} \in \mathbb{R}^3 \mid \|\vec{x}\|_1 = \sqrt{3}\}$; by applying a method similar to the one above (but using an orthogonal projection), the 2D non-oriented bounding box would be $b_{\text{tr}} = [-\sqrt{3} \ \sqrt{3}]^{\text{T}}$, $b_{\text{bl}} = [\sqrt{3} \ -\sqrt{3}]^{\text{T}}$ instead of a square with edges of lengths 1.

As these bounding boxes were, at inference time, meant to be the simulated output of a detector, a hyperparameter to adjust the amount of noise in the bounding box coordinates was introduced:

$$B_{\text{2D}}^{\text{noisy}}(v_a) = \left(\vec{b}_{\text{bl}} + k_{\text{noise}} \times \vec{\varepsilon}_{\text{uniform}}, \vec{b}_{\text{tr}} + k_{\text{noise}} \times \vec{\varepsilon}_{\text{uniform}}\right)$$

$$\vec{\varepsilon}_{\text{uniform}} \sim \mathcal{U}\left(\vec{b}_{\text{bl}} - \vec{b}_{\text{tr}}, \vec{b}_{\text{tr}} - \vec{b}_{\text{bl}}\right),$$

where k_{noise} is a hyperparameter that can be adjusted, and where $\vec{x} \sim \mathcal{U}(\vec{x}_l, \vec{x}_u)$ indicates that the i^{th} coordinate of $\vec{x}$ follows a uniform distribution in $[\vec{x}_{l,i}, \vec{x}_{u,i}[$ (interval bounded by the i^{th} coordinate of $\vec{x}_l$ and the i^{th} coordinate of $\vec{x}_u$).

For a given pair of agents (v_a, v_b) in image m, the image was cropped for each of the two bounding boxes, generating $\text{crop}(m, B_{\text{2D}}(v_a))$ and $\text{crop}(m, B_{\text{2D}}(v_a))$. The croping of m at a bounding box is the process of keeping only the pixel information that is contained within this bounding box (with some rescaling and interpolation, as applicable).

The model $\mathcal{D}_d$ consists of sub-modules $\mathcal{D}_{\text{bbox}}$, $\mathcal{D}_{\text{pixels}}$, and $\mathcal{D}_{\text{head}}$, which each respectively correspond to a module for processing bounding boxes, a module for processing cropped images, and a module for the final outputs. For two vehicles (v_a, v_b) in image m,

$$\mathcal{D}_{\text{combined}} \colon (m, B) \mapsto \left(\ \mathcal{D}_{\text{bbox}}(B), \ \ \mathcal{D}_{\text{pixels}}\big(\text{crop}(m, B)\big) \ \right)$$

$$\mathcal{D}_d\big(m, B_{\text{2D}}(v_a), B_{\text{2D}}(v_b)\big) = \mathcal{D}_{\text{head}}\Big(\mathcal{D}_{\text{combined}}\big(m, B_{\text{2D}}(v_a)\big), \mathcal{D}_{\text{combined}}\big(m, B_{\text{2D}}(v_b)\big)\Big).$$

Figure 4.9 illustrates the connections between $\mathcal{D}_{\text{bbox}}$, $\mathcal{D}_{\text{pixels}}$, and $\mathcal{D}_{\text{head}}$. Although one branch per vehicle is illustrated in that figure, they are in fact one same module and have the same parameters; i.e., at any point during training there is only one set of trainable parameters for $\mathcal{D}_{\text{bbox}}$ and only one set of trainable parameters for $\mathcal{D}_{\text{pixels}}$.

To quantify the performance gain offered by the use of pixel information in $\mathcal{D}_d$, a variant of this model was also evaluated: $\mathcal{D}_d^{\text{no-pixel}}$. The implementation of this modified model

amounted to defining it as follows:

$$\mathcal{D}_{\text{combined}}^{\text{no-pixel}} \colon (m, B) \mapsto \Big(\ \mathcal{D}_{\text{bbox}}(B), \ \ \vec{0} \ \Big)$$

$$\mathcal{D}_{d}^{\text{no-pixel}}\big(m, B_{\text{2D}}(v_a), B_{\text{2D}}(v_b)\big) = \mathcal{D}_{\text{head}}\Big(\mathcal{D}_{\text{combined}}^{\text{no-pixel}}\big(m, B_{\text{2D}}(v_a)\big), \mathcal{D}_{\text{combined}}^{\text{no-pixel}}\big(m, B_{\text{2D}}(v_b)\big)\Big) \, .$$

Both $\mathcal{D}_d$ and $\mathcal{D}_d^{\text{no-pixel}}$ were trained with $k_{\text{noise}} = 0.2$. Figure 4.10 and Figure 4.11 illustrate the performance of each model $\mathcal{D}_d$ and $\mathcal{D}_d^{\text{no-pixel}}$ during training, with the evaluation performance also being measured with $k_{\text{noise}} = 0.2$. These results indicate that $\mathcal{D}_d$ outperformed $\mathcal{D}_d^{\text{no-pixel}}$ in absolute validation performance and that the use of pixel information was valuable in decreasing the distance estimation error.

In order to investigate the sensitivity of these models (trained with $k_{\text{noise}} = 0.2$), the models were tested at values of k_{noise} ranging from 0 to 0.4, with increments of 0.05. Figure 4.12 illustrates the performance of the trained $\mathcal{D}_d$ and $\mathcal{D}_d^{\text{no-pixel}}$ as a function of the noise introduced at evaluation time. From this figure, it can be deduced that both models are relatively robust to small-to-moderate noise, with the trained $\mathcal{D}_d$ consistently outperforming $\mathcal{D}_d^{\text{no-pixel}}$.

To conclude this section, a model $\mathcal{D}_d$ that estimates the distance between two vehicles (given their bounding boxes and the image in which they appear) was evaluated and compared to $\mathcal{D}_d^{\text{no-pixel}}$, which omitted pixel information. Both of these models were trained on bounding boxes with the noise parameter k_{noise} set to 0.2. Using $\mathcal{D}_d$, distances between two vehicles in a CARLA wide-angle image can be estimated within an average error of roughly one metre when $k_{\text{noise}} = 0.2$. The use of pixel information appeared to provide a significant but not substantial improvement in estimation accuracy, and it appeared to offer a slight increase in robustness to noise. This architecture exploration and these results are only experimental and have some notable limitations:

- As previously mentioned, the derived 2D four-vertex bounding boxes were not ideal and were typically too large. One could envision using other information from CARLA in order to obtain more precise 2D bounding boxes.

- To estimate distances between vehicles, the ground-truth positions or centres of the vehicles were defined (in the computer program) to be the average of the 3D eight-vertex bounding boxes. However, it would be possible in CARLA to better estimate the closest distance between any two given vehicles and train $\mathcal{D}_d$ to estimate this distance instead, which would be expected to be more representative of whether a close

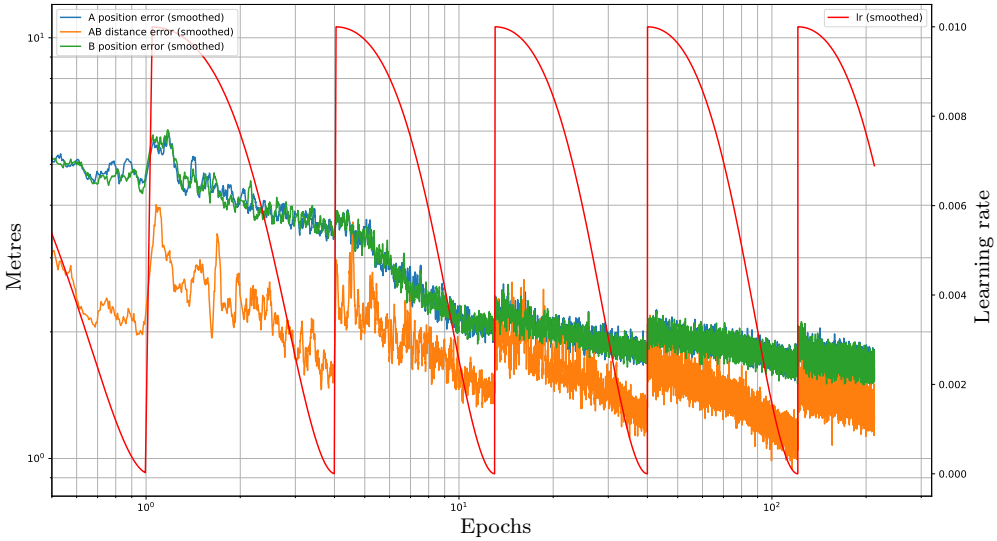
Table 4.2: Best epoch (with total) and corresponding mean absolute validation error, for multiple training runs, with $k_{\text{noise}} = 0.2$. This shows that architecture $\mathcal{D}_d$ has consistently outperformed $\mathcal{D}_d^{\text{no-pixel}}$. (The difference in number of runs between the architectures is due to the difference in training time.)

	$\mathcal{D}_d^{\text{no-pixel}}$			$\mathcal{D}_d$		
		Epochs			Epochs	
Run Index	Error (metres)	Best	Total	Error (metres)	Best	Total
1	1.176	120	212	0.960	107	131
2	1.182	116	150	0.944	114	135
3	1.146	115	150	0.873	121	140
4	1.149	117	150			
5	1.171	117	150			
6	1.191	120	150			

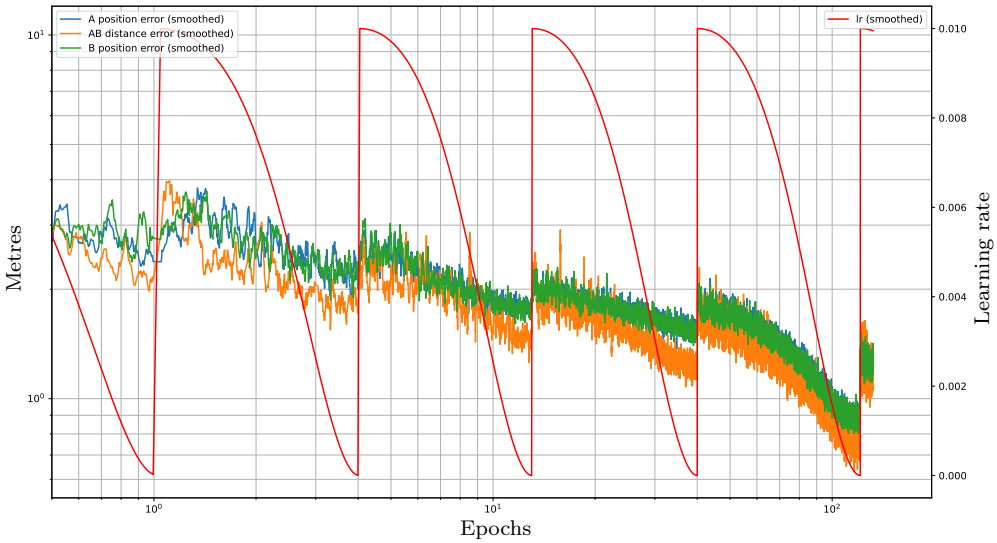
call has occured—indeed, two long trucks might nearly collide (bumper to bumper) without their centre-to-centre distance ever being below a few metres, whereas such a distance between the centres of two small motorcycles would be less indicative of a close call.

- Planar CNN layers were used for $\mathcal{D}_d$. Section 4.3 raised arguments toward the use of spherical convolutions for rectilinear projections, but planar convolutions were used for $\mathcal{D}_d$, as they are substantially less computationally demanding and they are simpler to use (especially considering the prevalence and maturity of existing implementations such as Conv2D of the PyTorch library [87]).

- The model $\mathcal{D}_d$ processes frames without taking into consideration the temporal relationships and ordering of the frames. This is a disadvantage because for any given frame, the frames preceding it and following it in time would be expected to contain some prior information that could be useful in better estimating the distance between two vehicles. As an example, consider a case in which prediction is performed on image at time t_i, but in which one of the two vehicles is obstructed; if this vehicle is less obstructed in either the frame at t_{i-1} or the frame at t_{i+1}, then knowing that the speed and acceleration of a vehicle is limited, there would be some prior information from these neighbouring frames that could be used to better estimate distance at t_i.

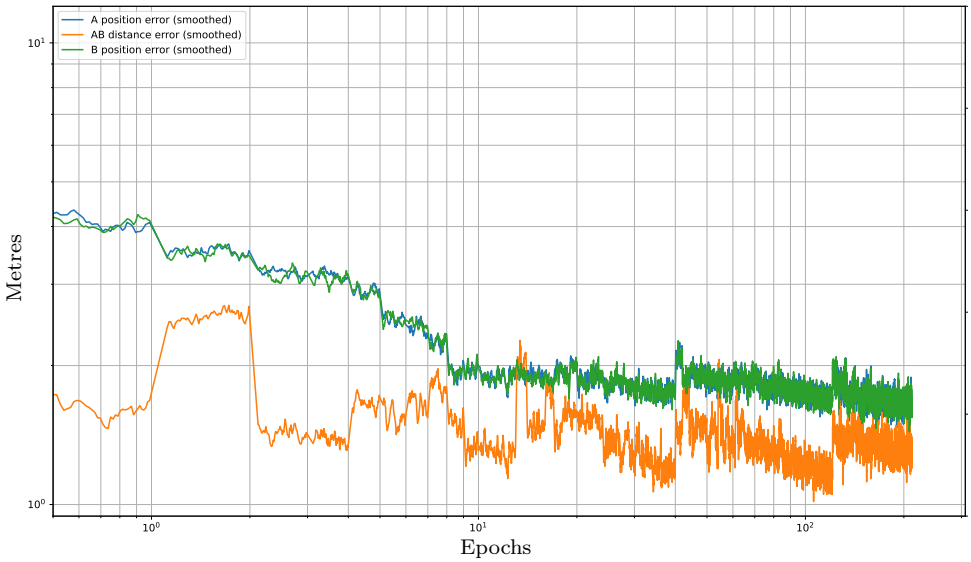


(a) Without using pixel information.

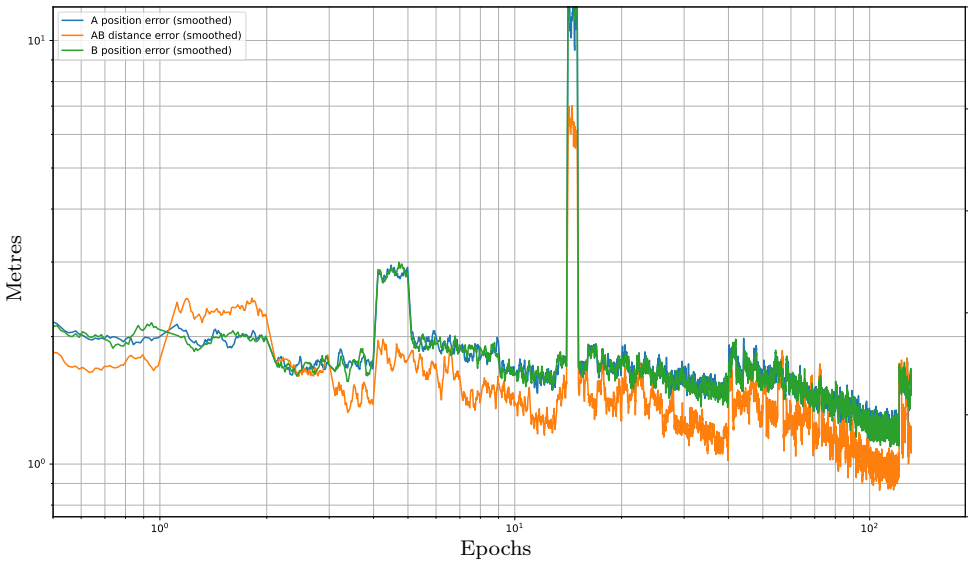


(b) Using pixel information.

Figure 4.10: Training performance for one training run per architecture. L^2 norm (blue) between predicted and ground-truth position for vehicle A, same for vehicle B (green), and absolute error between predicted and ground-truth distance between A and B (orange). The learning rate (red) varied with number of epochs according to a cosine annealing scheduler with warm restarts [86]. A rolling average was applied before plotting.



(a) Without using pixel information.



(b) Using pixel information.

Figure 4.11: Same as Figure 4.10, but for performance on validation data instead of training data (and without showing learning-rate curves).

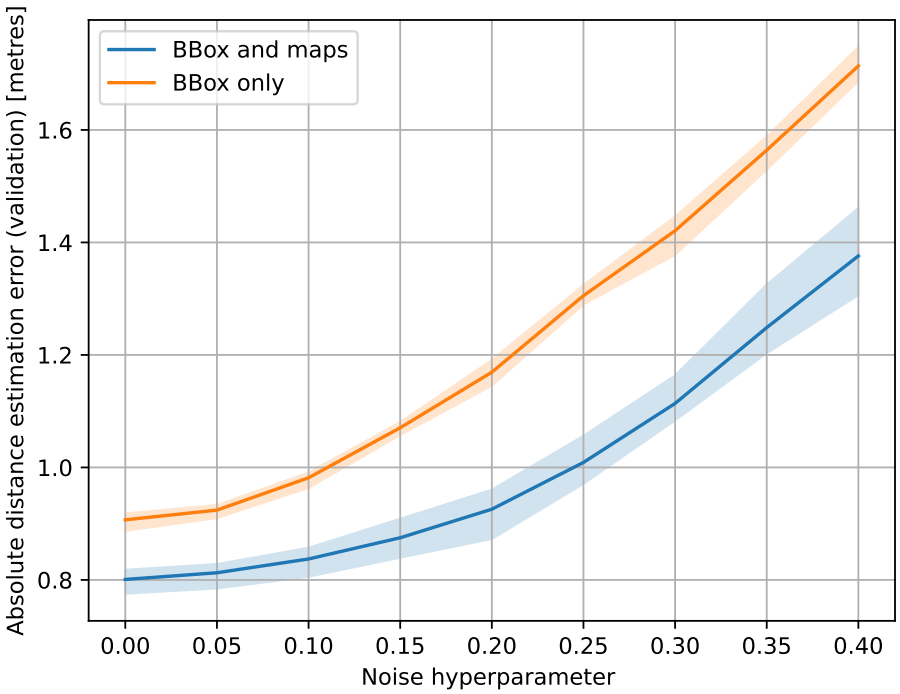


Figure 4.12: Validation performance as a function of k_{noise} (at validation time, ranging from 0 to 0.4) of models trained with $k_{\text{noise}} = 0.2$. In blue ("BBox and maps"): results across three trained instances of $\mathcal{D}_d$. In orange ("BBox only"): results across six trained instances of $\mathcal{D}_d^{\text{no-pixel}}$. The shaded area shows the range (between minimum and maximum) of performance across multiple trained model instances for a same architecture, for a given k_{noise} value. For both architectures, the estimation error increased with more noise, at similar rates. The $\mathcal{D}_d^{\text{no-pixel}}$ architecture consistently performed more poorly than $\mathcal{D}_d$.

Chapter 5

Conclusion

Two main problems were presented in this thesis: one which relates to the bin-picking problem, more specifically in the case of highly cluttered scenes, and one which relates to identifying close calls by estimating distances between vehicles, given some wide-angle RGB footage of road traffic.

In Chapter 3, the problem of estimating the relative placements of objects in clutter was formulated. Argumentation toward the use of *directed* simple graphs to represent the relations between objects in any given scene was put forward, and difficulties that may arise in realistic bin-picking datasets were discussed. A graph-network model architecture was proposed and implemented, and a baseline model which was based on an architecture presented in [76] was used for comparison. This graph-network model and this baseline model were both tested and compared using the synthetic part of the MetaGraspNetV2 [4] dataset and on the Visual Manipulation Relationship Dataset (VMRD) [76, 77, 82]. In addition, for this particular object-relationship problem, some rigorous definitions for precision and recall were both given, which account for the possibility of missed (undetected) objects on the part of the object detector, or of erroneous detection of nonexistent objects (resulting in extra bounding boxes). Based on the results obtained with the MetaGraspNetV2 and the VMRD datasets, it was observed that the proposed graph-based model architecture outperformed the baseline architecture in terms of F1 score (when using the given definitions for precision and recall).

As mentioned in Section 3.6, one of the potential future directions for this work would be to perform a careful ablation study of the proposed graph-network model, where each of the proposed architecture features would be ablated to measure the impact that it has on the overall performance of the model. Such an ablation study could be combined with a

study of the performance of the model for images specifically selected to contain each of the dataset challenges described in Section 3.2. This would provide insight into whether each of the proposed changes is indeed effective (or not) at addressing the difficulty for which it was designed. Another direction for future work would be an analysis of model performance in terms of prediction time. Indeed, as the problem described concerns real-time bin-picking applications, there would typically be an incentive to use a model that is able to reason about the scene, make decisions, and grasp objects more quickly. Computational complexity of prediction (and training) time could be analyzed both theoretically and empirically, as a function of the number of objects in the scene, and compared to architectures proposed by other researchers.

Chapter 4 formulated and analyzed the problem of estimating the distance between two given vehicles appearing in a wide-angle RGB video of traffic at any given intersection. This compound problem, which implicitly involves both detection and regression tasks for any given pair of detected vehicles, was separated into two parts: one focusing on planar and spherical convolutions (Section 4.3 and Section 4.4) and one focusing on estimating the distance between two vehicles, given an image and predetermined bounding boxes (Section 4.5).

Section 4.3 presented a fundamental disadvantage with the use of regular planar convolutions on wide-angle images, especially for estimating distances between objects in 3D, and it instead argued toward the use of spherical convolutions. Some experimental results to support this argument were presented in Section 4.4, with the use of two versions of a region proposal network: one which used planar convolutions and one which used spherical convolutions to detect and estimate the diameter of a spherical cap appearing in an image.

Section 4.5 presented some work that focused on a module which is intended to be used downstream of a vehicle detector. In addition to simplifying the experimental process, assuming that the bounding boxes are already detected has the added benefit of allowing for more control over the amount of error in the bounding boxes given to the distance-estimation model; this also allows for a controlled evaluation of the robustness of the model as a function of bounding box error. To quantify the amount of distance-estimation accuracy which can be obtained by this module from bounding-box coordinates alone (i.e., without the use of pixel information), a simple ablation study was conducted in which the distance-estimation module was modified to ignore pixel information in the image. When using pixel information, the module consistently performed better than when it ignored the pixel information.

As potential long-term directions for any continuations of this project, one might consider other accident-risk metrics to estimate. Ultimately, the objective is to provide an insightful

and actionable indicator of the risk of accident at any given intersection or road segment. For this, a more sophisticated metric, other than distance, could be considered. One such metric might be an indicator of the likelihood that an accident would have occurred at time t_i when only considering images for times $t < t_i$ (indeed, estimating the likelihood that an accident would have occurred with a model that has access to the future frames which show whether an accident has truly occurred does not seem logical). Furthermore, there might be more value in estimating some high-level metrics that are not limited to making conclusions about *particular* pairs of vehicles, but which rather focus on indicating the *causes* or common *locations* of close calls within an intersection of high risk; such metrics would presumably be much more insightful and actionable from the perspective of a municipality which wishes to make concrete improvements to dangerous intersections.

References

[1] A. L. W. Koh, J. Park, and P. Fieguth, "Challenges in detection of rare close-call events from vehicle-traffic videos", 8th Annual Conference on Vision and Imaging Systems (CVIS 2022), 2022.

[2] A. L. W. Koh, J. Park, and P. Fieguth, *Challenges in detection of rare close-call events from vehicle-traffic videos*, 8th Annual Conference on Vision and Imaging Systems (CVIS 2022), 2022, (poster session).

[3] M. Gilles, Y. Chen, T. R. Winter, E. Z. Zeng, and A. Wong, *MetaGraspNet: A large-scale benchmark dataset for scene-aware ambidextrous bin picking via physics-based metaverse synthesis*, 2022. arXiv: 2208.03963 [cs.CV].

[4] M. Gilles *et al.*, "MetaGraspNetV2: All-in-one dataset enabling fast and reliable robotic bin picking via object relationship reasoning and dexterous grasping", *IEEE Transactions on Automation Science and Engineering*, pp. 1–19, 2023. DOI: 10.1109/TASE.2023.3328964.

[5] J. Schmidhuber, "Deep learning in neural networks: An overview", *Neural Networks*, vol. 61, pp. 85–117, 2015, ISSN: 0893-6080. DOI: 10.1016/j.neunet.2014.09.003. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0893608014002135.

[6] L. Deng, "Artificial intelligence in the rising wave of deep learning: The historical path and future outlook [perspectives]", *IEEE Signal Processing Magazine*, vol. 35, no. 1, pp. 180–177, 2018. DOI: 10.1109/MSP.2017.2762725.

[7] A. Voulodimos, N. Doulamis, A. Doulamis, and E. Protopapadakis, "Deep learning for computer vision: A brief review", *Computational Intelligence and Neuroscience*, vol. 2018, 2018. DOI: 10.1155/2018/7068349.

[8] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation", in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2014.

[9] K. Duan, S. Bai, L. Xie, H. Qi, Q. Huang, and Q. Tian, "CenterNet: Keypoint triplets for object detection", in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, Oct. 2019.

[10] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers", in *Computer Vision – ECCV 2020*, A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, Eds., Cham: Springer International Publishing, 2020, pp. 213–229, ISBN: 978-3-030-58452-8.

[11] Z. Zou, K. Chen, Z. Shi, Y. Guo, and J. Ye, "Object detection in 20 years: A survey", in *Proceedings of the IEEE*, vol. 111, 2023. DOI: 10.1109/JPROC.2023.3238524.

[12] X. Chang, P. Ren, P. Xu, Z. Li, X. Chen, and A. Hauptmann, "A comprehensive survey of scene graphs: Generation and application", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 1, pp. 1–26, 2023. DOI: 10.1109/TPAMI.2021.3137605.

[13] D. Xu, Y. Zhu, C. B. Choy, and L. Fei-Fei, "Scene graph generation by iterative message passing", in *30TH IEEE CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION (CVPR 2017)*, ser. IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 3097–3106, ISBN: 978-1-5386-0457-1. DOI: 10.1109/CVPR.2017.330.

[14] Y. Li, W. Ouyang, B. Zhou, K. Wang, and X. Wang, "Scene graph generation from objects, phrases and region captions", in *2017 IEEE INTERNATIONAL CONFERENCE ON COMPUTER VISION (ICCV)*, 2017, pp. 1270–1279, ISBN: 978-1-5386-1032-9. DOI: 10.1109/ICCV.2017.142.

[15] D. Buchholz, *Bin-Picking, New Approaches for a Classical Problem*. Springer, 2016, ISBN: 978-3-319-26498-1. DOI: 10.1007/978-3-319-26500-1.

[16] A. Cordeiro, L. F. Rocha, C. Costa, P. Costa, and M. F. Silva, "Bin picking approaches based on deep learning techniques: A state-of-the-art survey", in *2022 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, 2022, pp. 110–117. DOI: 10.1109/ICARSC55462.2022.9784795.

[17] M. Alonso, A. Izaguirre, and M. Graña, "Current research trends in robot grasping and bin picking", in *International Joint Conference SOCO'18-CISIS'18-ICEUTE'18*, M. Graña *et al.*, Eds., Cham: Springer International Publishing, 2019, pp. 367–376, ISBN: 978-3-319-94120-2.

[18] M. Nieuwenhuisen *et al.*, "Mobile bin picking with an anthropomorphic service robot", in *2013 IEEE International Conference on Robotics and Automation*, 2013, pp. 2327–2334. DOI: 10.1109/ICRA.2013.6630892.

[19] K. Rahardja and A. Kosaka, "Vision-based bin-picking: Recognition and localization of multiple complex objects using simple visual cues", in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems. IROS '96*, vol. 3, 1996, pp. 1448–1457. DOI: 10.1109/IROS.1996.569005.

[20] Government of Canada, *Canadian motor vehicle traffic collision statistics*, 2023. [Online]. Available: https://tc.canada.ca/en/road-transportation/statistics-data/canadian-motor-vehicle-traffic-collision-statistics-2021.

[21] K. Kleeberger, R. Bormann, W. Kraus, and M. F. Huber, "A survey on learning-based robotic grasping", *Current Robotics Reports*, 2020. DOI: 10.1007/s43154-020-00021-6.

[22] A. Saxena, J. Driemeyer, and A. Y. Ng, "Robotic grasping of novel objects using vision", *The International Journal of Robotics Research*, Feb. 2008. DOI: 10.1177/0278364907087172.

[23] K. Ikeuchi, B. K.P. Horn, S. Nagata, T. Callahan, and O. Feingold, *Picking up an object from a pile of objects*, 1983.

[24] B. Sauvet, F. Lévesque, S. Park, P. Cardou, and C. Gosselin, "Model-based grasping of unknown objects from a random pile", *Robotics*, vol. 8, no. 79, 2019. DOI: 10.3390/robotics8030079.

[25] D. Katz, A. Venkatraman, M. Kazemi, J. A. Bagnell, and A. Stentz, "Perceiving, learning, and exploiting object affordances for autonomous pile manipulation", *Autonomous Robots*, 2014. DOI: 10.1007/s10514-014-9407-y.

[26] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen, "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection", *The International Journal of Robotics Research*, vol. 37, no. 4-5, pp. 421–436, 2018. DOI: 10.1177/0278364917710318.

[27] L. Chang, J. R. Smith, and D. Fox, "Interactive singulation of objects from a pile", *2012 IEEE International Conference on Robotics and Automation*, 2012. DOI: 10.1109/ICRA.2012.6224575..

[28] D. Katz, M. Kazemi, J. A. Bagnell, and A. Stentz, "Clearing a pile of unknown objects using interactive perception", in *2013 IEEE International Conference on Robotics and Automation*, 2013, pp. 154–161. DOI: 10.1109/ICRA.2013.6630570.

[29] T. Hermans, J. M. Rehg, and A. Bobick, "Guided pushing for object singulation", in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 4783–4790. DOI: 10.1109/IROS.2012.6385903.

[30] M. Fujita *et al.*, "What are the important technologies for bin picking? technology analysis of robots in competitions based on a set of performance metrics", *Advanced Robotics*, vol. 34, no. 7-8, pp. 560–574, 2020. DOI: 10.1080/01691864.2019.1698463.

[31] C. Martinez, R. Boca, B. Zhang, H. Chen, and S. Nidamarthi, "Automated bin picking system for randomly located industrial parts", in *2015 IEEE International Conference on Technologies for Practical Robot Applications (TePRA)*, 2015, pp. 1–6. DOI: 10.1109/TePRA.2015.7219656.

[32] N. Correll *et al.*, "Analysis and observations from the first amazon picking challenge", *IEEE Transactions on Automation Science and Engineering*, vol. 15, no. 1, pp. 172–188, 2018. DOI: 10.1109/TASE.2016.2600527.

[33] Y. Xiao *et al.*, "A review of object detection based on deep learning", *Multimedia Tools and Applications*, 2020. DOI: 10.1007/s11042-020-08976-6.

[34] Z.-Q. Zhao, P. Zheng, S.-T. Xu, and X. Wu, "Object detection with deep learning: A review", *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 11, pp. 3212–3232, 2019. DOI: 10.1109/TNNLS.2018.2876865.

[35] T. Cohen and M. Welling, "Group equivariant convolutional networks", in *Proceedings of The 33rd International Conference on Machine Learning*, M. F. Balcan and K. Q. Weinberger, Eds., ser. Proceedings of Machine Learning Research, vol. 48, New York, New York, USA: PMLR, Jun. 2016, pp. 2990–2999. [Online]. Available: https://proceedings.mlr.press/v48/cohenc16.html.

[36] H. B. Enderton, *Elements of Set Theory*. Academic Press, Inc., 1977, ISBN: 0-12-238440-7.

[37] T.-Y. Lin *et al.*, "Microsoft COCO: Common objects in context", in *EECV*, 2014. DOI: 10.1007/978-3-319-10602-1_48.

[38] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks", in *Advances in Neural Information Processing Systems*, F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, Eds., vol. 25, Curran Associates, Inc., 2012. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

[39] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks", *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, May 2017, ISSN: 0001-0782. DOI: 10.1145/3065386. [Online]. Available: https://doi.org/10.1145/3065386.

[40] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection", *Computing Research Repository*, vol. abs/1506.02640, 2015. arXiv: 1506.02640. [Online]. Available: http://arxiv.org/abs/1506.02640.

[41] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection", in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016.

[42] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Toward real-time object detection with region proposal networks", *Computing Research Repository*, Jan. 2016. DOI: 10.48550/arXiv.1506.01497.

[43] M. Carranza-García, J. Torres-Mateo, P. Lara-Benítez, and J. García-Gutiérrez, "On the performance of one-stage and two-stage object detectors in autonomous vehicles using camera data", *Remote Sensing*, 2020. DOI: 10.3390/rs13010089.

[44] J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, and A. W. M. Smeulders, "Selective search for object recognition", *International Journal of Computer Vision*, 2013. DOI: 10.1007/s11263-013-0620-5.

[45] P. Arbeláez, J. Pont-Tuset, J. T. Barron, F. Marques, and J. Malik, "Multiscale combinatorial grouping", in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2014.

[46] T. Kong, A. Yao, Y. Chen, and F. Sun, "HyperNet: Towards accurate region proposal generation and joint object detection", in *Conference on Computer Vision and Pattern Recognition*, 2016.

[47] J. Wang, K. Chen, S. Yang, C. C. Loy, and D. Lin, "Region proposal by guided anchoring", in *Conference on Computer Vision and Pattern Recognition*, 2019.

[48] R. Girshick, "Fast R-CNN", in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Dec. 2015.

[49] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection", in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct. 2017.

[50] W. Liu *et al.*, "SSD: Single shot multibox detector", in *Computer Vision – ECCV 2016*, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds., Springer International Publishing, 2016, pp. 21–37, ISBN: 978-3-319-46448-0.

[51] J. Redmon and A. Farhadi, "YOLO9000: Better, faster, stronger", in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul. 2017.

[52] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement", *Computing Research Repository*, Apr. 2018. DOI: 10.48550/arXiv.1804.02767.

[53] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: Optimal speed and accuracy of object detection", *Computing Research Repository*, Apr. 2020. DOI: 10.48550/arXiv.2004.10934.

[54] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors", in *IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2023.

[55] H. Law and J. Deng, "CornerNet: Detecting objects as paired keypoints", in *Proceedings of the European Conference on Computer Vision (ECCV)*, Sep. 2018.

[56] P. W. Battaglia *et al.*, *Relational inductive biases, deep learning, and graph networks*, 2018. arXiv: 1806.01261 (cs.LG).

[57] P. de Haan, T. Cohen, and M. Welling, "Natural graph networks", in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, Eds., vol. 33, Curran Associates, Inc., 2020, pp. 3636–3646. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2020/file/2517756c5a9be6ac007fe9bb7fb92611-Paper.pdf.

[58] A. Sanchez-Gonzalez, J. Godwin, T. Pfaff, R. Ying, J. Leskovec, and P. Battaglia, "Learning to simulate complex physics with graph networks", in *Proceedings of the 37th International Conference on Machine Learning*, H. Daumé III and A. Singh, Eds., ser. Proceedings of Machine Learning Research, vol. 119, PMLR, 2020, pp. 8459–8468. [Online]. Available: https://proceedings.mlr.press/v119/sanchez-gonzalez20a.html.

[59] R. Mercado *et al.*, "Graph networks for molecular design", *Machine Learning: Science and Technology*, vol. 2, no. 2, Mar. 2021. DOI: 10.1088/2632-2153/abcf91. [Online]. Available: https://dx.doi.org/10.1088/2632-2153/abcf91.

[60] M. Gori, G. Monfardini, and F. Scarselli, "A new model for learning in graph domains", in *2005 IEEE International Joint Conference on Neural Networks*, vol. 2, 2005, pp. 729–734. DOI: 10.1109/IJCNN.2005.1555942.

[61] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "Computational capabilities of graph neural networks", *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 81–102, 2009. DOI: 10.1109/TNN.2008.2005141.

[62] J. Zhou *et al.*, "Graph neural networks: A review of methods and applications", *AI Open*, vol. 1, pp. 57–81, 2020, ISSN: 2666-6510. DOI: 10.1016/j.aiopen.2021.01.001. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2666651021000012.

[63] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" In *International Conference on Learning Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=ryGs6iA5Km.

[64] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model", *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 61–80, 2009. DOI: 10.1109/TNN.2008.2005605.

[65] Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Diffusion convolutional recurrent neural network: Data-driven traffic forecasting", in *International Conference on Learning Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=SJiHXGWAZ.

[66] Z. Cui, K. Henrickson, R. Ke, and Y. Wang, "Traffic graph convolutional recurrent neural network: A deep learning framework for network-scale traffic learning and forecasting", *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 11, pp. 4883–4894, 2020. DOI: 10.1109/TITS.2019.2950416.

[67] D. Raposo, A. Santoro, D.G.T. Barrett, R. Pascanu, T. Lillicrap, and P. Battaglia, "Discovering objects and their relations from entangled scene representations", in *ICLR 2017*, 2017. [Online]. Available: https://openreview.net/pdf?id=rkrjrvmKl.

[68] A. Santoro *et al.*, "A simple neural network module for relational reasoning", in *Advances in Neural Information Processing Systems*, I. Guyon *et al.*, Eds., vol. 30, Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2017/file/e6acf4b0f69f6f6e60e9a815938aa1ff-Paper.pdf.

[69] T. S. Cohen, M. Geiger, J. Köhler, and M. Welling, *Spherical CNNs*, 2018. arXiv: 1801.10130 [cs.LG].

[70] J. Gu *et al.*, "Recent advances in convolutional neural networks", *Pattern Recognition*, vol. 77, pp. 354–377, 2018, ISSN: 0031-3203. DOI: https://doi.org/10.1016/j.patcog.2017.10.013. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0031320317304120.

[71] G. J. Awcock and R. Thomas, *Applied Image Processing*. MACMILLAN PRESS LTD, 1995, ISBN: 978-0-333-58242-8. DOI: 10.1007/978-1-349-13049-8.

[72] R. Kondor and S. Trivedi, "On the generalization of equivariance and convolution in neural networks to the action of compact groups", in *Proceedings of the 35th International Conference on Machine Learning*, J. Dy and A. Krause, Eds., ser. Proceedings of Machine Learning Research, vol. 80, PMLR, Jul. 2018, pp. 2747–2755. [Online]. Available: https://proceedings.mlr.press/v80/kondor18a.html.

[73] A. Baker, *Matrix Groups, An Introduction to Lie Group Theory*. Springer-Verlag London Limited, 2002, ISBN: 978-1-85233-470-3.

[74] J. R. Driscoll and D. M. Healy, Jr., "Computing fourier transforms and convolutions on the 2-sphere", *Advances in Applied Mathematics*, vol. 15, no. 2, pp. 202–250, 1994, ISSN: 0196-8858. DOI: https://doi.org/10.1006/aama.1994.1008. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0196885884710086.

[75] D.L. Johnson, *Symmetries*. Springer-Verlag London Limited, 2001, ISBN: 978-1-85233-270-9. DOI: 10.1007/978-1-4471-0243-4.

[76] H. Zhang, X. Lan, X. Zhou, Z. Tian, Y. Zhang, and N. Zheng, "Visual manipulation relationship network for autonomous robotics", *2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids)*, 2018. DOI: 10.1109/HUMANOIDS.2018.8625071.

[77] H. Zhang, X. Lan, X. Zhou, Z. Tian, Y. Zhang, and N. Zheng, "Visual manipulation relationship recognition in object-stacking scenes", *Pattern Recognition Letters*, vol. 140, pp. 34–42, 2020, ISSN: 0167-8655. DOI: https://doi.org/10.1016/j.patrec.2020.09.014. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167865520303445.

[78] Z. Wu, J. Tang, X. Chen, C. Ma, X. Lan, and N. Zheng, *Prioritized planning for target-oriented manipulation via hierarchical stacking relationship prediction*, 2023. arXiv: 2303.07828 [cs.RO].

[79] M. Dong, Y. Bai, S. Wei, and X. Yu, *A single multi-task deep neural network with a multi-scale feature aggregation mechanism for manipulation relationship reasoning in robotic grasping*, 2023. arXiv: 2305.13591 [cs.RO].

[80] G. Zuo, J. Tong, H. Liu, W. Chen, and J. Li, "Graph-based visual manipulation relationship reasoning network for robotic grasping", *Frontiers in Neurorobotics*, vol. 15, 2021, ISSN: 1662-5218. DOI: 10.3389/fnbot.2021.719731. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fnbot.2021.719731.

[81] M. Ding, Y. Liu, C. Yang, and X. Lan, "Visual manipulation relationship detection based on gated graph neural network for robotic grasping", in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022, pp. 1404–1410. DOI: 10.1109/IROS47612.2022.9981077.

[82] https://gr.xjtu.edu.cn/zh/web/zeuslan/dataset, accessed on 2023-11-06.

[83] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator", in *Proceedings of the 1st Annual Conference on Robot Learning*, 2017, pp. 1–16.

[84] P. Gros and L. Quan, "Projective invariants for vision", Laboratoire d'Informatique Fondamentale et d'Intelligence Artificielle, Institut National Polytechnique de Grenoble, 46, avenue Félix Viallet 38031 Grenoble, France, Technical Report, Dec. 1992, pp. 1–46. [Online]. Available: https://inria.hal.science/inria-00590013.

[85] L. J. V. Gool, T. Moons, E. Pauwels, and J. Wagemans, "Invariance from the euclidean geometer's perspective", *Perception*, vol. 23, no. 5, pp. 547–561, 1994. DOI: 10.1068/p230547.

[86] I. Loshchilov and F. Hutter, *SGDR: Stochastic gradient descent with warm restarts*, 2017. arXiv: 1608.03983 [cs.LG].

[87] *PyTorch*, https://pytorch.org, accessed on 2023-10-27.