Streamlining Warehouse Operations: Monocular Multi-View Layout Estimation and Intelligent Visual Servoing for Robotic Tasks

Thesis submitted in partial fulfillment
of the requirements for the degree of

Master of Science
in
Electronics and Communication Engineering
by Research

by

Pranjali Pramod Pathre
2019112002
pranjali.pathre@research.iiit.ac.in

International Institute of Information Technology
Hyderabad - 500 032, INDIA
June 2024

International Institute of Information Technology
Hyderabad, India

CERTIFICATE

It is certified that the work contained in this thesis, titled "**Streamlining Warehouse Operations: Monocular Multi-View Layout Estimation and Intelligent Visual Servoing for Robotic Tasks**" by **Pranjali Pramod Pathre**, has been carried out under my supervision and is not submitted elsewhere for a degree.

Date	Adviser: Prof. K. Madhava Krishna

To *My Parents* and *Mentors*

Acknowledgments

I would like to express my deepest gratitude to my research advisor, Prof. K. Madhav Krishna, for his unwavering encouragement and guidance throughout my research journey. His mentorship and constructive feedback have been instrumental in pushing my boundaries, exposing me to diverse domains within robotics, and fostering my personal and academic growth. I would also like to thank Dr. Samarth Brahmbhatt for his invaluable feedback and constructive input throughout the project's development. His insightful comments and suggestions have played a significant role in shaping the clarity and quality of the research. I would like to acknowledge Dr. Harit Pandya for his guidance, which has deepened my understanding of the subject and motivated me to tackle challenges with curiosity.

Big thanks to Md. Nomaan Qureshi for his insightful perspectives and keen analytical skills in problem formulation. His thoughtful contributions have been crucial in defining the scope and direction of the second part of this study, setting a solid foundation for our investigation. I'm thankful for the awesome teamwork and dedication shown by my co-authors, Gunjan Gupta and Anurag Sahu, at every stage of the research process. I'm grateful to my fellow contributors - Ashwin Rao, Avinash Prabhu, Meher Shashwat Nigam, Tanvi Karandikar and Mandyam Brunda for their unwavering commitment and their tireless efforts have been instrumental in achieving our common goals. Their innovative ideas and thorough analysis have enriched our discussions and propelled us toward meaningful insights and discoveries.

I am grateful to my batchmates and fellow research students at RRC for their consistent contributions of ideas and essential information, which have significantly enriched my research work and validated my ideas. A big thanks to my friends Sasanka GRS and Samruddhi Shastri for all the laughs, understanding, and great times we've had together. And finally, I express my deepest gratitude to my parents for their unwavering dedication and support throughout my academic journey, even in the face of difficulties. Their unconditional love, sacrifices, and encouragement have been the cornerstone of my success, giving me the strength and motivation to overcome obstacles and pursue my dreams.

Abstract

This thesis is driven by the imperative to enhance efficiency and precision in warehouse management systems. The first part introduces *MVRackLay*, an innovative solution employing multi-view analysis to accurately estimate complex layouts of racks and shelves in warehouses, offering a comprehensive 3D rendering of the scene from a single monocular camera. In the second part, *Imagine2Servo* revolutionizes visual servoing algorithms by generating intermediate goal images through diffusion-based editing techniques, enabling precise control in object-reaching tasks in warehouses and tasks like long-range navigation and manipulation. Through real-world validation, these innovations mark significant advancements in warehouse automation and robotic control systems, promising transformative impacts across various domains.

In the first part of this thesis (chapter 3), we showcase *MVRackLay*, a monocular multi-view layout estimation for warehouse racks and shelves. Unlike typical layout estimation methods, *MVRackLay* estimates multi-layered layouts across multiple views, wherein each layer corresponds to the layout of a shelf within a rack. Given a sequence of images of a warehouse scene, our model outputs segmented racks and each shelf's front and top view layout within a rack. Further, *MVRackLay* shows superior performance vis-a-vis its single view counterparts in layout accuracy, quantized in terms of the mean IoU and mAP metrics. We also showcase multi-view stitching of the 3D layouts, resulting in a representation of the warehouse scene concerning a global reference frame akin to a rendering of the scene from a SLAM pipeline. To the best of our knowledge, this is the first such work to portray a 3D rendering of a warehouse scene in terms of its semantic components - Racks, Shelves and Objects - all from a single monocular camera.

In the second part of this thesis (chapter 4), we introduce *Imagine2Servo*, an innovative approach leveraging diffusion-based image editing techniques to enhance visual servoing algorithms by generating intermediate goal images. Visual servoing, the method of controlling robot motion through feedback from visual sensors, has seen significant advancements with the integration of optical flow-based methods. However, its application remains limited by inherent challenges, such as the necessity for a target image at test time, the requirement of substantial overlap between initial and target images, and the reliance on feedback from a single camera. *Imagine2Servo* allows for the extension of visual servoing applications beyond traditional constraints, enabling tasks like long-range navigation and manipulation without pre-defined goal images. We show its applicability in precisely performing different warehouse tasks, navigation tasks and manipulation tasks. We propose a pipeline that synthesizes subgoal images

grounded in the task at hand, facilitating servoing in scenarios with minimal initial and target image overlap and integrating multi-camera feedback for comprehensive task execution. Our contributions demonstrate a novel application of image generation to robotic control, significantly broadening the capabilities of visual servoing systems. Real-world experiments validate the effectiveness and versatility of the *Imagine2Servo* framework in accomplishing a variety of tasks, marking a notable advancement in the field of visual servoing.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

In today's fast-paced world of technology and efficiency demands, there's a clear call for continuous research in warehouse automation. As businesses worldwide struggle to meet rising consumer expectations and manage intricate supply chains, finding smart solutions to streamline warehouse operations is crucial. The push for warehouse automation is growing rapidly, with predictions that robots could soon manage entire warehouses with minimal human intervention [10]. Yet, it's notable that around 30% of warehouses still lack essential warehouse management systems (WMS) [11].

The issue of warehouse automation carries equal significance both in warehouses lacking WMS and in situations where automated robotic agents interact with shelf spaces. Performing actions like retrieving an item from a shelf, given the language command, can be significantly expedited through automated systems. Before engaging with an item on the shelf, the initial step involves reaching the object from within the warehouse. When provided with language instruction, the robot is expected to autonomously navigate and reach the desired target without human intervention. Motivated by this need, in Chapter 4, we introduce *Imagine2Servo*[†], an intelligent visual servoing-based controller designed to guide robots in reaching specific stacks of objects on shelves. This model isn't solely intended for warehouse automation; it's also tailored for various manipulation and long-range navigation tasks. Another key motivation behind the development of the Imagine2Servo model is to offer a significant enhancement to Image-based Visual Servoing (IBVS) algorithms. Traditional visual servoing algorithms rely on having a predefined goal image during testing. In Imagine2Servo, we innovate by generating a task-specific sub-goal image, which is then utilized by an IBVS controller to perform actions aimed at achieving the desired objective.

While an autonomous agent operates within a warehouse space, it is also crucial to complement its actions with a semantic comprehension of the rack layout. This understanding, particularly regarding objects positioned on shelves, aids in addressing subsequent tasks such as determining object counts and estimating available space. To tackle this challenge, layout prediction comes into play, entailing

[†]Project Page: `https://brunda02.github.io/RRC/`

the segmentation of all shelves within a rack at varying heights. While RackLay [1] provides layout estimation for racks, its scope is limited to predicting the layout of the predominant rack visible in the image. While suitable for constrained scenarios, this method inherently lacks scalability and adaptability, even after finetuning it for a dynamic warehouse setting where multiple racks may be concurrently present and only partially visible across a sequence of images. To tackle these challenges comprehensively, in chapter 3, we introduce *MVRackLay*[§], a *multi-view* layout estimation for all partly or wholly visible racks in the image. Furthermore, we leverage past observations to integrate spatial data over time, enhancing the accuracy of layout estimation.

1.2 Research problems Tackled

We discuss two research problems tacked in this thesis below. First, the problem of predicting top-view and front-view occupancy layouts for all racks and shelves from an input sequence of monocular RGB warehouse images is addressed in Sec. 1.2.1. The second problem of generating intermediate goal images for the IBVS algorithms to servo to the desired target view based on the language instruction is addressed in Sec. 1.2.2.

1.2.1 MVRackLay: Monocular Multi-View Layout Estimation for Warehouse Racks and Shelves

We introduce a simple yet efficient network architecture called *MVRackLay* (chapter 3), which generates both the *top-view* and *front-view* layouts of all shelves within each rack. These layouts are derived from monocular RGB images of a warehouse, which constitute frames of a video sequence. It's important to note that racks may only be partially visible in some frames (Fig. 1.1). The network learns layouts in the canonical frame centered on the shelf, called the *shelf-centric* layout.

An essential point to note is that the problem is not a direct application of a standard formulation of object recognition, semantic segmentation or layout estimation. While the above methods can be applied to objects on rack shelves [12, 9], present methods cannot be adapted directly to localize rack shelves, as shown in our baseline comparisons in Sec. 3.4.3. While typical layout formulations estimate layouts with reference to a single dominant plane (such as the ground plane) [13], warehouse rack shelves take the form of disconnected planar segments, each present at different heights above the ground plane. Furthermore, they often appear occluded and diffused in warehouse scenes. Thus, an important novelty of our formulation is the adaptation of deep architectures to the problem of multi-view layout estimation over multiple shelves and racks, as well as shelf contents.

MVRackLay leverages, using Convolutional LSTM layers, the temporal data across images and extends the layout prediction problem to a wider scope in a warehouse setting. Unlike *RackLay[1]*, where the model predicts the layouts only for the rack in focus, we design our model to predict the layouts

[§]Project Page: `https://pranjali-pathre.github.io/MVRackLay/`

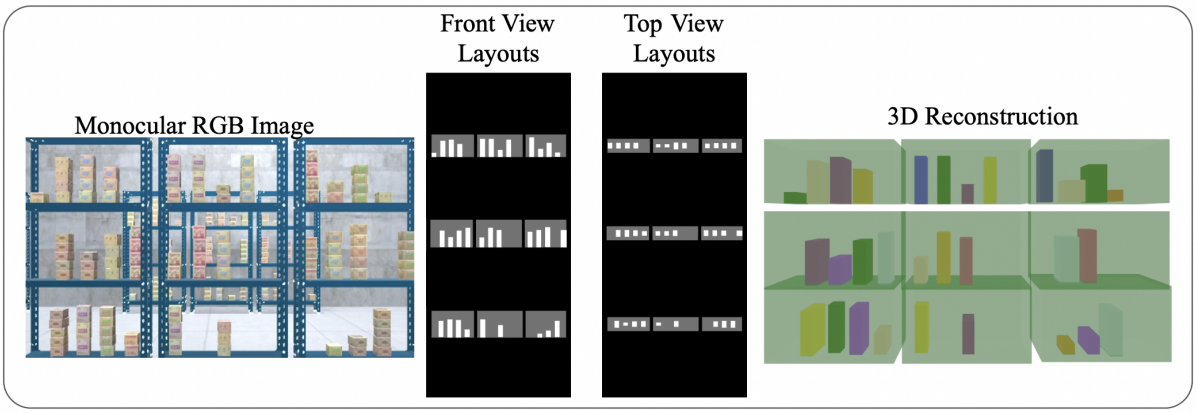

Figure 1.1: We introduce **MVRackLay**, a deep neural architecture, which, given a sequence of monocular RGB images of racks in a warehouse scene, predicts the *top-view* and *front-view* occupancy layouts for all the racks and shelves, partly or completely visible, in a *shelf-centric* frame. Fusing these *shelf-centric* layouts provides a 3D reconstruction of the rack, and stitching together all the racks across frames produces a 3D reconstruction of the warehouse.

for all the racks in the image, whether visible fully or partially in frame. Additionally, to leverage the temporal information across image sequences, we propose the multiview and multilayer layout prediction mechanism for *MVRackLay*, where layouts are predicted over a sequence of images. Furthermore, through the downstream application of layout stitching across multiple views, we demonstrate that an explicit 3D model of the warehouse can be reconstructed from the predicted layouts.

Real-world public datasets for warehouses are few and not comprehensive. To alleviate the issue of obtaining sufficient training data, we utilize the open-sourced synthetic data generation pipeline *WareSynth* proposed in [14], an extended and improved version of the pipeline introduced in [1]. Using domain randomization in *WareSynth* we generate a huge variety of synthetic warehouses, suitably emulating any real-world warehouse. We train and evaluate the performance of *MVRackLay* on such synthetic warehouse scenes. The monocular image sequences obtained from *WareSynth* involve translation of the camera along a predefined path, facing a row of racks and at a fixed distance from them. The annotations for these sequences are fed into the deep *MVRackLay* network.

1.2.1.1 Contributions

Our contributions in Chapter 3 are summarised as follows:

- It presents a notable improvement over the formerly proposed *RackLay* [1] architecture, the keynote of which is the use of Convolutional LSTM Layers, which enable the network to train on a monocular image sequence, rather than discrete images of racks. The network uses this previously-missing spatial data to improve *shelf-centric* layout predictions in each frame, specifically when racks and shelves are only partially visible in the image.

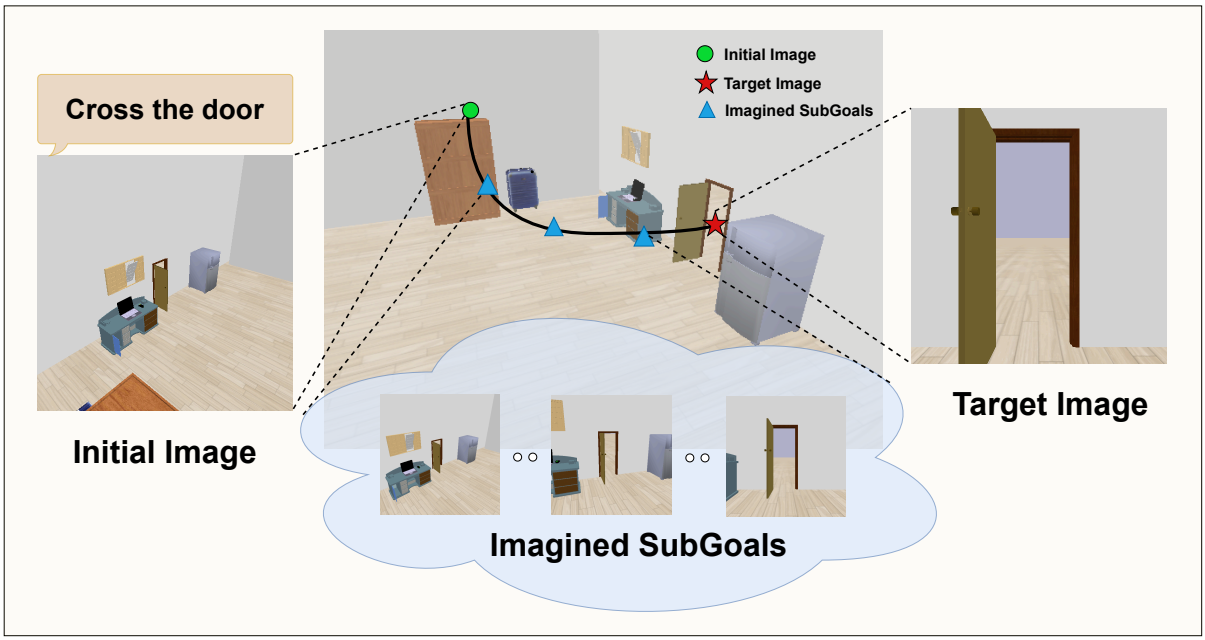

Figure 1.2: Given a single eye-in-hand camera input and the language instruction to perform a task, we introduce **Imagine2Servo**, a pipeline that generates intermediate goal images for the image-based visual servoing (IBVS) algorithms to servo to the desired target location. The loop of generating a sub-goal image and execution of the action by an IBVS controller is continued until we reach the final target. We show the application of our pipeline to long-range navigation as well as manipulation tasks.

- Moreover, we showcase an application that employs these layout representations. Layout-enabled multi-view 3D warehouse reconstruction is a novel use case discussed in Sec. 3.5. Most importantly, we show that the use of Convolutional LSTM layers to predict 3D representations of racks in each frame enables them to be stitched together into a 3D reconstruction of the warehouse in a global reference frame, similar to the rendering of a scene from a SLAM pipeline.

- We demonstrate significant performance gain on popular rubrics compared to previous methods [8, 9] adapted to the estimation of shelf layouts. Equally important, we showcase notable improvement in layout estimation over the single-view estimator [1]. Moreover, we compile and present results of several ablations involving variations in architecture which establish the superiority of *MVRackLay*.

1.2.2 Imagine2Servo: Intelligent Visual Servoing with Diffusion-Driven Goal Generation for Robotic Tasks

The visual servoing problem [15] involves the challenge of controlling the motion of a robot by utilizing feedback from visual sensors, typically cameras, to adjust its actions in real-time. This process

entails the robot's ability to interpret visual data to determine its relative position and orientation with respect to target objects or locations within its environment. The core objective is to enable the robot to perform precise movements or reach specific goals by continuously comparing the current visual scene against a desired configuration or outcome. This approach requires sophisticated algorithms for image processing and control theory to bridge the gap between visual perception and mechanical action, thereby allowing the robot to adapt its movements based on the visual feedback it receives.

Recently, there has been a lot of progress in optical flow-based visual servoing methods [2, 16, 17, 18, 19]. These methods are shown to be highly precise in reaching their goals with some guarantees of convergence. However, the utility of visual servoing has remained limited due to major limitations common to all servoing algorithms: 1) They necessarily require a goal image during test time. This makes it quite tough for visual servoing algorithms to be applied in real-world navigation or manipulation since if we already have a map of the environment, there are better ways to reach the goal pose than through a target image. 2) Visual Servoing cannot work if there is not much overlap between the initial and target image. 3) Visual Servoing can only accommodate feedback from a single camera. To tackle these challenges, we introduce Imagine2Servo, a diffusion-guided visual servoing approach. We demonstrate its effectiveness not only in reaching tasks within warehouse environments but also in a wider array of navigation and manipulation tasks.

Solving the problems faced by visual servoing can greatly enhance the utility of visual servoing methods. For example, solving the problem of final image generation based on the skill the robot is executing can make servoing quite useful for real-world tasks. Imagine a drone mounted with a monocular camera trying to cross a door. The robot will first have to visualise the approximate position of the door just before it crosses it; then, it will have to visually servo to the imagined image and then apply a simple hardcoded skill to cross the door.

This pattern of imagining a goal, servoing to the goal and applying a hardcoded skill can be repeated for quite a lot of skills, both in navigation as well as manipulation. Another good example of a skill we can solve using servoing is the 'reaching' skill, which is a part of several manipulation tasks. For instance, take the example of 'unplugging the charger', where the robot has to 'reach' a particular grasping pose before applying a hardcoded policy. In chapter 4, we leverage the recent advancements in diffusion-based image editing to provide a much-needed and major update to servoing algorithms.

1.2.2.1 Contributions

Our contributions in Chapter 4 can be stated as follows:

- We first introduce the notion of final image generation for servoing, grounded in the skill we are trying to execute. We then introduce our *Imagine2Servo* pipeline, which takes as input the current view of the camera and outputs a subgoal for the servoing algorithm to reach. We also demonstrate how feedback from multiple cameras can be incorporated.

- We show that the *Imagine2Servo* pipeline can be used to solve the problems where traditional servoing fails even with a target image in place. This is especially pertinent when the initial image and target image don't have much overlap.
- We then go on to show how we can apply the same framework to solve multiple other skills which follow the reach and act framework. We show the applicability of *Imagine2Servo* to long-range servoing, reaching tasks in warehouses, and different navigation and manipulation tasks in a variety of simulators such as Habitat [4], RLBench [5], PyBullet [3] and Unity [6].
- We show real-robot results of our method to further demonstrate its efficacy.

1.3 Thesis Layout

C1 This chapter (chapter 1) serves as an introduction, outlining the scope of the research conducted in this thesis within the realm of monocular layout estimation for warehouse shelves and the challenges encountered by visual servoing methods. We delineate the issues we aim to resolve and provide rationale for the methodologies we will develop in subsequent chapters.

C2 Chapter 2 provides an introduction to the essential background information and terminology utilized throughout this study.

C3 Chapter 3 introduces *MVRackLay: Monocular Multi-View Layout Estimation for Warehouse Racks and Shelves*, a method designed to forecast the occupancy layouts, both top and front views for all racks and shelves. Through comprehensive experimentation within the Unity simulator [6], we showcase its enhanced performance compared to its single-view counterpart, RackLay[1]. Additionally, we present a 3D reconstruction of the entire warehouse utilizing the predicted layouts.

C4 Chapter 4 delves into the *Imagine2Servo: Intelligent Visual Servoing with Diffusion-Driven Goal Generation for Robotic Tasks* method, focusing on its innovative approach to generating subgoals for visual servoing tasks. Through thorough experimentation across various simulators, we demonstrate significant enhancements compared to alternative baseline methods.

C5 Chapter 5 ends with a recapitulation of the methods and results covered in this thesis, the practical applications of these methodologies, and the potential avenues for extending this research in the future.

Chapter 2

Background

In this chapter, we will provide a concise overview of the current body of literature concerning layout estimation, image-based diffusion for visual servoing, and other relevant research pertaining to the addressed problems.

2.1 Layout Estimation

In recent times, there has been a surge in interest surrounding the learning of scene layouts and the direct acquisition of volumetric representations from RGB images. Deep learning techniques have significantly improved in reliability and accuracy across various computer vision tasks, including semantic segmentation, object detection, and depth estimation. However, even the amalgamation of these fundamental solutions falls short when addressing more complex tasks such as shelf-layout estimation within warehouse management systems, which demands multi-layer top-view layout estimation. In light of this, we review existing approaches and highlight the distinctions of our methodology from others in the field.

2.1.1 Related work

2.1.1.1 Object detection methods

The literature review of object detection methods for warehouse estimation reveals various techniques to optimize inventory management and operational efficiency. Traditional approaches, such as handcrafted feature extraction coupled with classifiers like Support Vector Machines (SVM)[20] or Random Forests [21], have paved the way for more sophisticated deep learning architectures. Convolutional Neural Networks (CNNs) have emerged as dominant players in this field, demonstrating remarkable performance in detecting and localizing objects within warehouse environments. Researchers have explored various CNN architectures, including Faster R-CNN [22], YOLO (You Only Look Once) [23], and SSD (Single Shot MultiBox Detector) [24], tailoring them to the unique challenges posed by warehouse settings such as varying lighting conditions, occlusions, and diverse object types.

Furthermore, recent studies have focused on leveraging transfer learning and domain adaptation techniques to enhance the generalization capabilities of object detection models across different warehouse scenarios. By pre-training large-scale datasets like COCO (Common Objects in Context) [25] or ImageNet [26] and fine-tuning domain-specific data, these approaches have shown promising results in improving detection accuracy and robustness. Additionally, integrating sensor fusion methodologies, combining data from multiple sources such as RGB cameras, LiDAR, and depth sensors, has garnered attention for its potential to provide richer contextual information and mitigate challenges associated with individual sensor modalities. A significant part of our problem involves localizing semantic classes like shelves and boxes/cartons in a 3D scene. Several approaches exist to detect object layouts in 3D. Some of these [27, 28] combine information from images and LiDAR, while others [13, 7] first convert images to *bird's eye view* representations, followed by object detection.

2.1.1.2 Bird's eye view (BEV) representation

Bird's eye view estimation is crucial in warehouse management systems, enabling efficient resource allocation, inventory tracking, and layout planning. Various computer vision and sensor fusion techniques have been explored to generate accurate bird's eye views of warehouse environments. These methods often involve integrating data from overhead cameras, LiDAR, and other sensors to reconstruct a top-down perspective of the warehouse space. For instance, approaches based on structure-from-motion (SfM) algorithms utilize images captured from multiple viewpoints to estimate the 3D structure of the scene, subsequently transforming it into a bird's eye view representation. Schulter [29] proposed one of the first approaches to estimate an occlusion-reasoned BEV road layout from a single color image. Wang [30] build on top of [29] to infer parameterized road layouts. In contrast, our approach is non-parametric and, hence, more flexible than such parametric models, which may not account for all possible layouts. We take inspiration from MonoLayout [31] (which can be trained end to end on colour images, reasons beyond occlusion boundaries and being non-parameterized, need not be actuated with these additional inputs) and extend it to multiple planes.

2.1.1.3 Single-view layout estimation

RackLay [1] introduced a layout estimation technique tailored to the specific scenario where only one rack is in focus and completely visible within a single monocular image. While effective for such constrained setups, this approach inherently limits scalability and adaptability, particularly in dynamic warehouse environments where multiple racks may be simultaneously present and partially visible across a sequence of images. We propose MVRackLay, a more flexible and accurate solution for shelf layout prediction in monocular image sequences, to address this limitation. MVRackLay offers enhanced flexibility by extending the capability to predict shelf layouts not only for racks fully visible in a single frame but also for those partially visible or obscured by objects or other racks. By leveraging information from consecutive frames in the input video sequence, MVRackLay incorporates spatial data

of warehouse racks over time, enabling the network to infer more comprehensive and accurate layout estimations across multiple racks simultaneously.

This incorporation of temporal information allows MVRackLay to capture the dynamic nature of warehouse environments, where objects may be moved, added, or removed over time. By analyzing the evolution of rack configurations across frames, MVRackLay can adaptively adjust its predictions, resulting in more robust and reliable shelf layout estimations compared to methods solely reliant on single-frame analysis. In summary, MVRackLay represents a significant advancement over RackLay [1], offering greater flexibility and accuracy in predicting shelf layouts across monocular image sequences of warehouse environments. By harnessing spatial data from consecutive frames, MVRackLay demonstrates superior performance in handling dynamic scenarios, making it a valuable tool for warehouse management and optimization tasks.

2.1.1.4 Warehouse Datasets

Warehouse datasets play a pivotal role in the development and evaluation of algorithms and models for various warehouse-related tasks, including object detection, layout estimation, and inventory management. Publicly available datasets for warehouse settings are far and in between. Real-world datasets like LOCO [32] exist for scene understanding in warehouse logistics, but they provide a limited number of images, along with corresponding 2D annotations. Furthermore, there are only a handful of general-purpose synthetic data simulators for generating photo-realistic images, like NVIDIA Isaac [33], which provide warehouse scenes. However, there is no straightforward way to modify them to generate annotations needed for the task at hand.

2.2 Diffusion Models and Visual Servoing

The application of diffusion models for image-to-image diffusion tasks has garnered substantial attention in recent years. Pioneering works in this domain have demonstrated the efficacy of diffusion-based frameworks for various image-processing tasks. These works underscore the versatility and effectiveness of diffusion models for a wide range of image-to-image diffusion tasks, paving the way for further advancements in the field.

Advancements in image-based visual servoing have significantly enhanced the capabilities and applicability of robotic systems across various domains. One notable advancement lies in integrating deep learning techniques, revolutionizing visual perception by enabling robots to interpret and understand complex visual scenes with unprecedented accuracy and robustness. In image-based visual servoing, the target image is a crucial reference for guiding the robotic system towards achieving a desired task or goal. Without a target image, the system lacks a clear objective to pursue, hindering its ability to perform tasks effectively.

2.2.1 Related work

2.2.1.1 Diffusion Models

In recent years, the landscape of diffusion models has undergone a remarkable evolution, marked by rapid advancements and a proliferation of applications. Building upon foundational contributions laid out in seminal works such as [34, 35, 36, 37], diffusion models have emerged as versatile tools with broad utility across diverse domains. These models have been effectively harnessed for an array of tasks, spanning text-driven image generation [38, 39], depth estimation [40, 41], image inpainting [42], and even video editing [43]. Notably, the scope of diffusion models has expanded beyond traditional domains, with the burgeoning interest in their application within the realm of robot learning [44, 45, 46, 47]. However, amidst this expansive landscape, recent attention has been particularly drawn towards the fusion of diffusion techniques with smart data collection strategies, yielding promising outcomes in the domain of image editing [48, 49, 50].

In this study, we delve into the niche of image editing models that leverage diffusion techniques, with a keen focus on their potential to drive high-quality image synthesis. Specifically, we explore the efficacy of instruct-pix2pix [48], among other innovative approaches, as a means to produce images that serve as exemplary representations of desired objectives for servoing models to pursue. By amalgamating the power of diffusion models with strategic data collection methodologies, we aim to unravel novel pathways towards the creation of visually compelling and goal-oriented images, thereby paving the way for enhanced performance and adaptability in the realm of image-driven robotic tasks.

2.2.1.2 Visual Servoing

Image-based visual servoing (IBVS) has gained significant attention in robotics for its ability to control the motion of a robotic system based on visual feedback from cameras. Numerous studies have explored different aspects of IBVS, including feature extraction methods [51], control strategies [52], camera calibration techniques [53], and applications in various robotic tasks [54]. Recent advancements have focused on improving the robustness and efficiency of IBVS systems through techniques such as deep learning-based visual features [55], adaptive control algorithms [56], and real-time implementation frameworks [57]. Despite these advancements, challenges remain in handling complex environments, handling occlusions, and ensuring real-time performance.

Deep learning-based visual servoing approaches have emerged as an alternative to traditional methods reliant on hand-crafted features. In the last few years, optical flow-based visual servoing algorithms [2, 16, 17, 18, 19] have gained a lot of traction. They use off-the-shelf optical flow networks [58, 59] as a feature for the servoing controller to optimize. These algorithms have been shown to be quite robust and have been applied to a variety of tasks, from large building avoidance [60] to dynamic grasping [61]. However, they still require a target image to servo to, which limits their applicability in a lot of real-world tasks where the goal image is not available. In contrast, our Imagine2Servo framework generates the subgoal for the servoing to reach to. We show in our results that not only our

skill-grounded Imagine2Servo framework is able to beat traditional servoing algorithms, even in cases where the servoing methods have a final image to go to, but can also be applied to other tasks which fall in the reach and execute category.

2.2.1.3 Other Related Works

A work notably related to ours is SuSIE [62], which employs the InstructPix2Pix [48] architecture in tandem with reinforcement learning (RL) policies for solving manipulation tasks. While SuSIE represents a significant contribution, our research primarily focuses on providing a substantial update to servoing models. Furthermore, RL approaches are known for their considerable computational demands and sample inefficiency, posing challenges in training models without assured convergence towards the objective. In contrast, our servoing controller is designed for broader applicability without the need for fine-tuning to specific datasets. This attribute significantly enhances its deployability across various environments and real-world scenarios, offering a more efficient and flexible solution. Visual servouing algorithms

Another pertinent work is Robotap [63], which employs keypoint tracking combined with a visual servoing algorithm for certain manipulation challenges. However, Robotap's application scope is limited by its reliance on well-defined key points, rendering it ineffective for tasks where key points cannot be easily identified, such as navigating through a door. In contrast, our model's versatility allows it to address a broader spectrum of tasks, bypassing the limitations inherent in keypoint-dependent approaches. Visual foresight models [64, 65] are another line of work which are similar to our primary motivation; however, they require to learn a model, which is hard to do in image space. We employ a simpler yet effective model-free approach to combine diffusion-based foresight with servoing, which not only enhances the utility of servoing but also provides a pipeline which can be used to solve a lot of skills. In the realm of diffusion-based navigation, several recent works have contributed significantly to the field, showcasing novel approaches and methodologies. Notably, the NoMaD [66] framework stands out as a parallel work in this domain. NoMaD leverages diffusion-based strategies to navigate complex environments effectively.

Chapter 3

MVRackLay: Monocular Multi-View Layout Estimation for Warehouse Racks and Shelves

3.1 Introduction

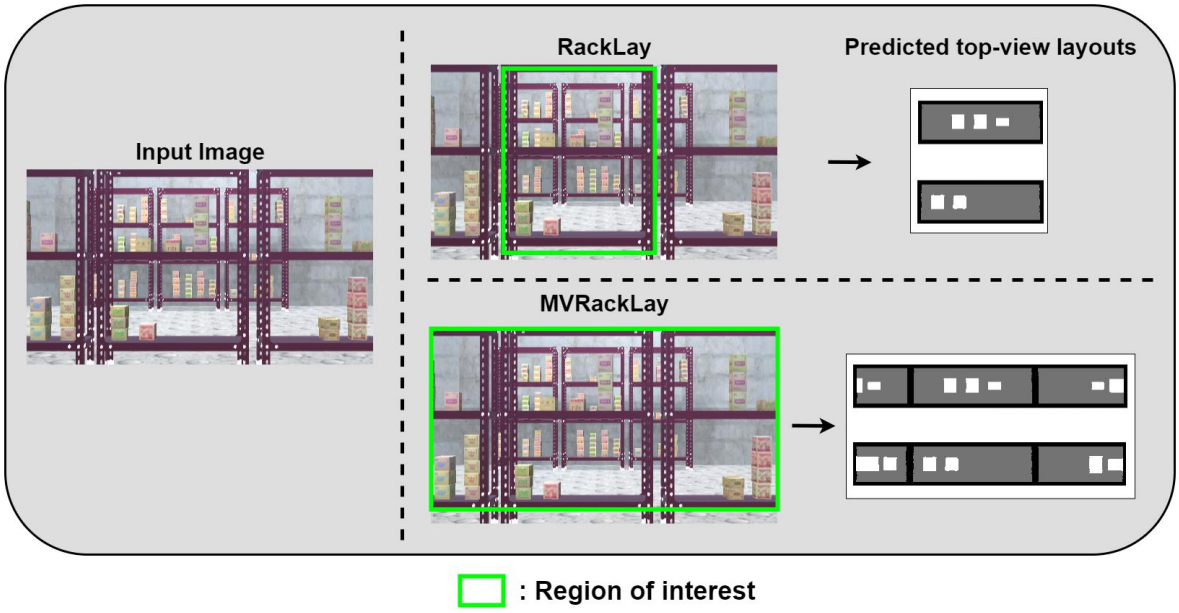

Figure 3.1: Qualitative depiction of the regions of interest targeted by Racklay [1] and our model MVRackLay. In contrast to RackLay, which exclusively predicts the layout of the dominant rack in the input image, MVRackLay extends this capability to encompass layout prediction for all racks depicted within the image while also inferring the individual rack boundaries. Additionally, we enhance the accuracy of layout prediction by integrating temporal data from previous frames.

This chapter presents MVRackLay, a deep neural architecture designed for Monocular Multi-View Layout Estimation in warehouse environments. Although RackLay [1] offers layout estimation specifi-

cally for racks, its functionality is restricted to predicting the layout of the most prominent rack visible in an image (refer to fig. 3.1). While effective in certain constrained situations, this approach inherently lacks scalability and adaptability, particularly in dynamic warehouse environments where numerous racks may coexist simultaneously and only be partially visible across a sequence of images. Given a sequence of monocular RGB images capturing racks within a warehouse scene, MVRackLay predicts the occupancy layouts for both top-view and front-view perspectives for all visible or partially visible racks and shelves.

We commence with an overview of our approach, including problem formulation and architecture details. Subsequently, we detail the dataset used for training and testing, followed by a thorough examination of experimental outcomes and results analysis. Our discussion encompasses the model's performance across diverse warehouse scenarios alongside a comparative study against relevant methods and an ablation analysis. Lastly, we demonstrate the practical utility of our model through 3D reconstruction facilitated by the generated layouts.

3.2 Approach

3.2.1 Problem Formulation

Given a sequence of RGB images $\mathcal{I}_1$, $\mathcal{I}_2$,..., $\mathcal{I}_n$ of racks in warehouses in perspective view, we aim to predict the top-view (bird's eye view) and front-view layout for each rack present in each frame of the input video sequence.

We consider **R** to be a rectangular area in a top-down orthographic view of the scene. The camera is placed at the mid-point of the lower side of the rectangle, directly facing the racks such that the image plane is orthogonal to the ground plane. (Fig. 3.2). Concretely, we want our network to generate top-view and front-view layouts for all the racks visible in each frame $\mathcal{I}_t$, within a region of interest Ω. Our network predicts shelf-centric layouts where we map Ω to a rectangular area. In **shelf-centric** layout representation, we consider Ω to be a rectangular area, positioned such that its centre coincides with the centre of the shelves spanning across all the racks visible in the image, as shown in Fig. 3.2. This layout is, hence with respect to the rack and is view-point agnostic.

Our model predicts **top-view** and **front-view** layout representations for each frame in the sequence. Top-view layouts predict the bird's eye view occupancy of each shelf on the rack. Each pixel in the layout can either be classified as *occupied*, *unoccupied*, or *background*. A pixel is said to be *occupied* when it is a part of the object on the shelf, *unoccupied* when it represents the empty space on the rack, and *background* when it denotes the region which is not occupied by the shelf.

Consider a right-handed coordinate frame where the X axis points to the right, the Y axis points downwards, and the Z axis points into the plane. For top-view layouts, the coordinate frame is positioned at the centre of the shelves, spanning across all racks for layouts. Hence, the top-view layouts are parallel to the X-Z plane, as in fig. 3.2 (and the ground plane) at corresponding shelf heights. In the case of

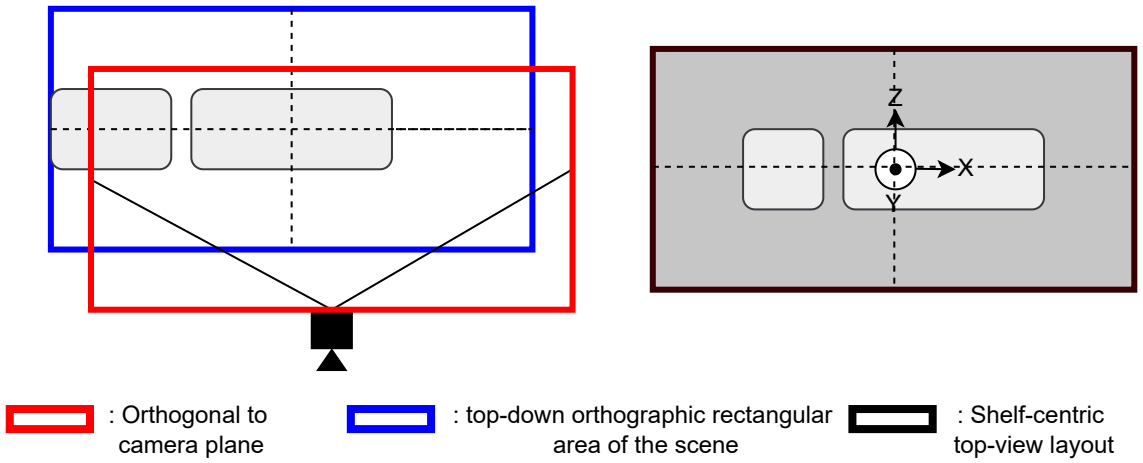

Figure 3.2: **Top-view layout representation:** The plane orthogonal to the camera plane (red box) is mapped to the shelf frame. In the shelf frame, the centre of the layout coincides with the centre shelves spanning across all the racks visible in the image.

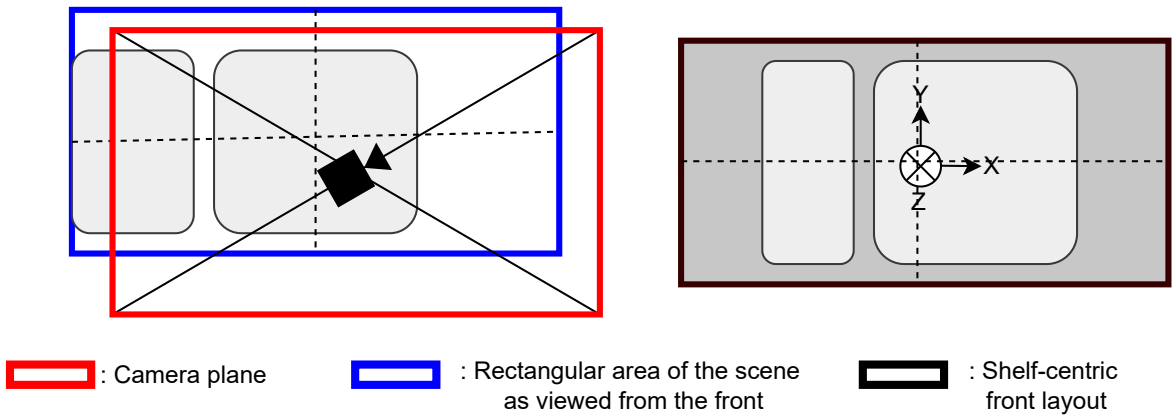

Figure 3.3: **Front-view layout representation:** The image plane is mapped to the shelf frame. In the shelf frame, the centre of the layout coincides with the centre shelves spanning across all the racks visible in the image.

front-view layouts, the centre of the coordinate frame is positioned at the centre of shelves in X and Y directions. Front view layouts are, therefore, orthogonal to the ground plane, as in fig. 3.3.

As an additional task, we demonstrate **Multi-view Stitching** - combining the representation from the top-view and front-view layouts to obtain a 3D reconstruction of all racks in the warehouse in the global shelf frame (Fig. 3.9). This can be further used for 3D spatial reasoning tasks. We infer the X and Z coordinates from the top view and the Y coordinates from the front view. We further explore these applications in Sec. 3.5.

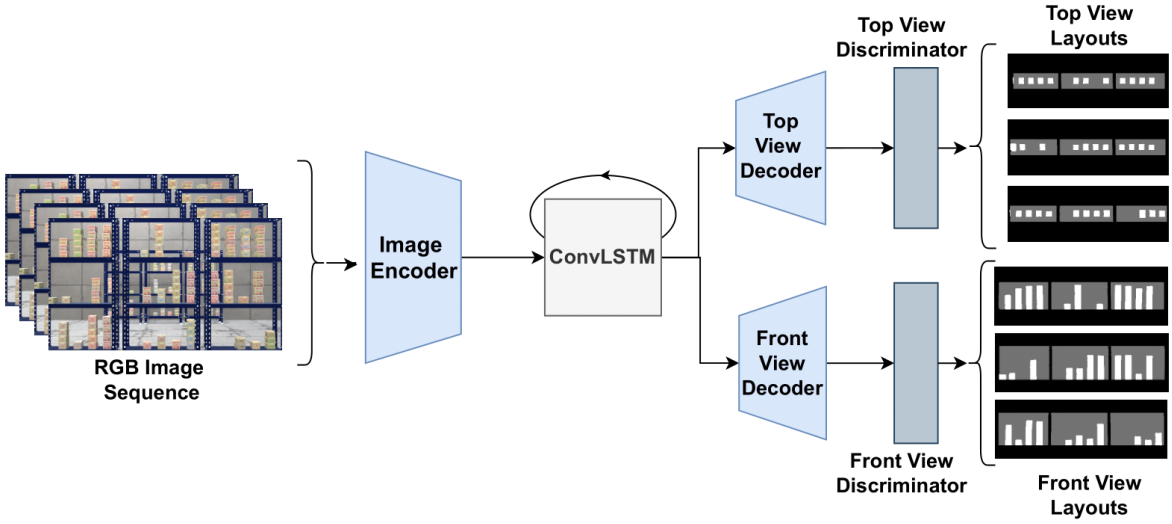

Figure 3.4: **Architecture:** The figure shows the architecture diagram for *MVRackLay-disc*. It comprises a context encoder, a Convolutional LSTM for encoding temporal information, and multi-channel decoders and adversarial discriminators. (Refer to Sec. 3.2.2).

3.2.2 MVRackLay Architecture

We build a double-decoder MVRackLay architecture (Fig. 3.4) which takes as input a sequence of RGB images $\mathcal{I}_1$, $\mathcal{I}_2$,..., $\mathcal{I}_n$ and predicts the top-view and front-view layouts. The components of the model are described in detail below.

1) A **context encoder** which uses a ResNet-18 backbone pre-trained on ImageNet [67] to retrieve relevant 3D scene and semantic components from the monocular input $\mathcal{I}_t$. This feature extractor learns low-level features $\mathcal{C}_1$, $\mathcal{C}_2$,..., $\mathcal{C}_n$ that help reason about the occupancy of the scene points.

2) A stacked **Convolutional LSTM** submodule uses the encoder extracted features $\mathcal{C}_1$, $\mathcal{C}_2$,..., $\mathcal{C}_t$ and in turn encodes a temporal representation to capture motion across input frames. We use this Spatio-temporal prediction to estimate consistent layouts by consolidating the information from the past frames to predict the current frame. The number of previous frames used in this prediction is a hyperparameter, the value of which is varied in our ablation studies (Refer to Sec. 3.4.4). The output of this block is an encoded representation that better reasons the scene points as *occupied*, *unoccupied*, and *background*.

3) A **top-view decoder** and **front-view decoder** that generates layouts respectively for top-view and front-view from the temporal representation learned by the Convolutional LSTM submodule. It consists of upsampling layers to output an $\mathcal{R} \times \mathcal{D} \times \mathcal{D}$ grid which represents the layout, where $\mathcal{R}$ is the number of output channels and $\mathcal{D}$ is the resolution of the output layouts.

4) Identical **discriminators** following top-view and front-view decoders, respectively, are adversarial regularizers that rectify the layouts further by homogenizing their distributions to be similar to the true distribution of plausible layouts. The layout predicted by the decoder is the input to this submodule which outputs the final refined predictions.

3.2.3 Loss function

We describe here the loss function of our *MVRackLay* architecture for top-view layout estimation. We use stochastic gradient descent to optimize over the network parameters ϕ, ψ, θ of the context encoder, convolutional LSTM, and the decoder.

$$L_{sup}(\widehat{\mathcal{T}};\phi,\psi) = \sum_{j=1}^{N}\sum_{i=1}^{\mathcal{R}} f\left(\widehat{\mathcal{T}}_i^j, \mathcal{T}_i^j\right)$$

$$L_{adv}(\widehat{\mathcal{T}};\phi,\psi,\theta) =_{\theta\sim p_{fake}} [(\widehat{\mathcal{T}}(\theta)-1)^2]$$

$$L_{short}(\widehat{\mathcal{T}};\phi,\psi) = \sum_{j=1}^{N}\sum_{i=1}^{seqlen-1} f\left(\widehat{\mathcal{T}}_i^j, \widehat{\mathcal{T}}_i^{j+1}\right)$$

$$L_{long}(\widehat{\mathcal{T}};\phi,\psi) = \sum_{j=1}^{N}\sum_{i=1}^{seqlen-1}\sum_{k=j+2}^{seqlen} f\left(\widehat{\mathcal{T}}_i^j, \widehat{\mathcal{T}}_i^k\right)$$

$$L_{discr}(\widehat{\mathcal{T}};\theta) =_{\theta\sim p_{true}} [(\widehat{\mathcal{T}}(\theta)-1)^2] +_{\theta\sim p_{fake}} [(\widehat{\mathcal{T}}(\theta))^2]$$

$$L_{total} = L_{sup} + L_{short} + L_{long} + L_{adv} + L_{discr}$$

Here $\widehat{\mathcal{T}}$ is the predicted top-view layout of each shelf, $\mathcal{T}$ is the ground truth top-view layout of each shelf, $\mathcal{R}$ is the maximum number of shelves considered, and N is the mini-batch size.

L_{sup} is the per-pixel cross-entropy loss which penalizes variation of the predicted output labels ($\widehat{\mathcal{T}}$) from corresponding ground-truth values ($\mathcal{T}$). The adversarial loss L_{adv} enables the distribution of layout estimates from the top-view decoder (p_{fake}) to be similar to the actual data distribution (p_{true}). L_{discr} enforces the discriminator to accurately classify the network-generated top-view layouts sampled from the true data distribution [68]. L_{short} is the short-range consistency loss, and L_{long} is the long-range consistency loss. Finally, we minimize the total loss over the network parameters ϕ, ψ, θ and use it to back-propagate gradients through the network. Equivalent expressions are defined for front-view layout prediction as well.

3.3 Warehouse Dataset

For training and testing our network, we generated a diverse dataset with 20k images spanning 400 sequences with 50 images per sequence, using *WareSynth* described in [14]. It is split into 360/20/20 for training/testing/validation. All the results discussed are on the test set of this dataset. The dataset is highly varied and spans multiple warehouse scenes. The variety demonstrates the generality of *MVRack-Lay* and is useful to evaluate the performance of our model in varied warehouse settings. The video sequences captured using *WareSynth* resemble the data captured by a manual camera movement or a

mechanized system performing the task in an actual warehouse. Various scene elements were diversified during data generation to impart assortment in generated scenes so that the synthetically developed warehouses mimic their real-world counterparts. We describe these randomizations below.

Domain Randomization: We show the randomizations we introduce using 3 randomly selected images from our dataset through Fig. 3.5 (referred throughout this section):

- Boxes exhibit variability in terms of size, texture, rotations, colour, and reflective properties.
- Placement of boxes varies from densely packed arrangements to moderate spacing to sparse configurations.
- Racks' colour and texture are randomized to introduce diversity.
- The vertical stacking height of boxes is randomized to simulate realistic stacking scenarios.
- Background settings include options such as a solid wall backdrop or a bustling warehouse environment.
- Floors and walls feature randomized colours and textures to add visual complexity.
- The camera's positioning relative to the rack is adjusted within a specified range to capture different numbers of shelves in each frame.
- Camera movement is orchestrated to ensure varying numbers of racks are visible across consecutive frames, enhancing the diversity of captured scenes.

The extensive diversity within our dataset has proven instrumental in preventing the network from overfitting solely to synthetic data domains. Instead, it has facilitated the learning of features that closely mimic real-world scenes..

3.4 Experiments and Analysis

3.4.1 Evaluated Methods and Metrics

We compare the following variants of MVRackLay:

- *MVRackLay-Disc-4*: Double decoder architecture with discriminators for both front-view *and* top-view, with a sequence length of 4 used in the ConvLSTM module.
- *MVRackLay-Disc-8*: Double decoder architecture with discriminators for both front-view *and* top-view, with a sequence length of 8 used in the ConvLSTM module.
- *MVRackLay-4*: Double decoder architecture for both front-view *and* top-view without discriminators, with a sequence length of 4 used in the ConvLSTM module.

We report Mean Intersection-Over-Union (mIoU) and Mean Average-Precision (mAP) scores in this task as the previously proposed methods also evaluate the model on the same criterion.

3.4.2 Results

We first trained *MVRackLay-4* for top-view and front-view. Having achieved superior results compared to baselines, we further trained *MVRackLay-Disc-4* for both front-view and top-view to capture the distribution of our layouts, which led to the best results. We observed performance gains as discussed in Sec. 3.4.4. We further trained *MVRackLay-Disc-8* to capture the performance of the model for higher sequence length (discussed in Sec. 3.4.4). Overall, *MVRackLay-Disc-4* showed the best overall performance (refer Table 3.1).

Fig. 3.5 summarizes the results of *MVRackLay-disc-4* tested on domain randomized data. Our best network is able to predict all the racks present in the image with clean boundaries separating them and precisely estimate the space between two racks and two objects on the shelf. *MVRackLay* can rightly predict the layouts for the racks in the foreground, even in the presence of background clutter in the image. It can also reason for the narrow spaces in densely packed shelves.

The results show that *MVRackLay* can significantly benefit downstream warehouse tasks. Sec. 3.5 demonstrates one such task of 3D warehouse reconstruction using a multiview layout predicted by the network. The generality and superiority of the results show that our model can easily be transferred to warehouses with diverse arrangements of racks and objects on shelves.

3.4.3 Comparison with baselines

RackLay: RackLay [1] was proposed to solve the problem of layout prediction for the dominant rack present in the input RGB image. We trained it for the task of layout prediction for all the racks present over the video sequence. From Table 3.1 it is clear that *MVRackLay-Disc-4* performs significantly better than *RackLay-D-disc* quantitatively. Comparing qualitatively, Fig. 3.6 summarizes the improvements of our network over *RackLay*. *RackLay* often fails to demarcate an exact boundary between two racks present in an image (row 1). In row 2, observe that *RackLay* is unable to predict a sharp boundary of both box and shelf as in the corresponding output of *MVRackLay-Disc-4*. *RackLay* often incorrectly predicts the presence of the box on racks (rows 1, 2) or suffers from noisy predictions of boxes in regions with no objects as well as of the shelf (row 2). Our model not only extends *RackLay*'s functionality to image sequences but also improves upon its liabilities. We predict sharper and more accurate shelf and object boundaries and reduce false box predictions.

PseudoLidar-PointRCNN: PointRCNN [8] is a 3D object detector that uses the raw point cloud as input. Hence we use the PseudoLidar[7] information to detect 3D objects using PointRCNN. As this method considers a single dominant plane and is employed for birds-eye view prediction, we report metrics for the bottommost shelf (refer to Table 3.1) and the top-view prediction only. Accounting for the single-dominant layer assumption, it is clear from Table 3.1 that our model performs better as it

	Top View				Front View			
	Rack		Box		Rack		Box	
Method	**mIoU**	**mAP**	**mIoU**	**mAP**	**mIoU**	**mAP**	**mIoU**	**mAP**
MVRackLay-Disc-4 (ours)	**96.44**	98.01	**86.89**	**92.70**	**94.98**	**97.14**	**88.02**	**93.49**
MVRackLay-Disc-8 (ours)	95.84	**98.12**	85.98	88.35	93.78	96.98	86.49	88.76
MVRackLay-4 (ours)	95.94	97.94	86.66	89.31	93.73	96.58	86.72	89.16
RackLay-D-disc	93.44	94.98	82.80	85.47	91.75	93.10	83.28	85.49
PseudoLidar-PointRCNN[8]	73.28	77.40	55.77	81.26	–	–	63.05	89.45
MaskRCNN-GTdepth[9]	–	–	35.57	47.44	–	–	76.48	82.48

Table 3.1: **Quantitative results**: We benchmark the 3 different versions of our network - *MVRackLay-Disc-4*, *MVRackLay-Disc-8* and *MVRackLay-4*, along with three baselines- *RackLay-D-disc*[1] *PseudoLidar-PointRCNN*[7, 8] and *MaskRCNN-GTdepth*[9] (as described in Sec. 3.4.1).

performs multilayer, front-view, and top-view layout predictions that can also reason about the inter-shelf distance.

Mask R-CNN: We select Mask R-CNN[9] as one of our baselines to test the instance segmentation method for multi-layer layout prediction in the warehouse setting for the sequential data. We subsequently integrate the Mask R-CNN segmented instances with the depth maps and project on a horizontal plane to segment the boxes shelf-wise, using the fact that a set of boxes on a particular shelf will be situated on the same plane located at some elevation from the ground plane. From the experiment, it is observed that Mask R-CNN fails to detect the precise boundary of the rack due to its thin structures. If multiple racks are present in the image, Mask R-CNN also fails to mark a clear distinction between them. The results summarized in Table 3.1 prove that our model performs better quantitatively too. Since Mask R-CNN can only claim regarding the points visible in the image, it is evident that our model accomplishes better results with amodal estimation to reason beyond the visual elements that structurally characterize racks and objects.

3.4.4 Ablation studies

To thoroughly comprehend the underlying effect of different components, we perform ablation studies over the model's architecture and examine its effect on performance.

1) Convolutional LSTM sequence length: We varied the time steps used in the stacked Convolutional LSTM submodule. We observed that *MVRackLay-Disc-4* was able to converge faster and improve qualitatively over *MVRackLay-Disc-8*. Although quantitatively *MVRackLay-Disc-4* and *MVRackLay-Disc-8* perform alike (refer to Table 3.1), from fig. 3.7 considerable qualitative improvements can be

observed. *MVRackLay-Disc-4* performs better in identifying precise object boundaries and distinguishing between the closely spaced objects on the rack. A lower sequence length enabled the model to compile only relevant details from past frames and avoid spurious noise.

2) Adversarial learning: In *MVRackLay-Disc-4*, we add discriminators after decoders in *MVRackLay-4* to capture the distribution of *plausible* layouts. We observed a substantial improvement both quantitatively (refer to Table 3.1) and qualitatively (Fig. 3.8). Layouts have become sharper, and most notably, using a discriminator diminished the stray pixels wrongly categorized as boxes. The actual distance between the boxes positioned near the end of the shelf is difficult to estimate as they are imaged obliquely. In such cases, *MVRackLay-Disc-4* remarkably improved the prediction of the boxes and generated cleaner layouts.

3.5 Applications

Multi View Stitching: We illustrate the process of 3D warehouse reconstruction using the layout predictions of MVRackLay-Disc-4. This task also aids in subsequent tasks, including estimating free space and determining object counts on the racks. From the layout prediction of a particular frame $\mathcal{I}_t$, we first obtain the 2D bounding boxes of all shelves and boxes detected in the front-view and top-view layout. The detections from the top-view and front-view layouts correspond to identify the matching. Once we have a map, we generate the 3D bounding box for all the mapped racks and objects using the dimension information from front-view and top-view layout predictions. Finally, we combine these representations of each shelf to get a 3D reconstruction f_t of all the racks in the frame. This process is repeated for all the frames in the sequence.

Given two consecutive 3D reconstructions f_t and f_{t+1}, we initially find all the corresponding matching boxes. We then calculate the shift between them; using this shift, we discern the direction of the motion. Finally, we consider the last box in frame f_t in the direction of motion and check its corresponding box in frame f_{t+1}. If the size of the box in f_{t+1} is larger, we increase the size of the shelf and boxes in f_t accordingly. If there is an addition of a new box or shelf in f_{t+1}, the same is included in the f_t reconstruction. Eventually, we obtain the merged layouts of f_t and f_{t+1} in f_t frame. If F_t denotes the merged 3D representation from f_1 to f_t, f_{t+1} is merged into F_t using the method described above. Fig. 3.9 shows the 3D reconstruction of a single warehouse with 4 racks, obtained from the multi-view stitching of predicted layouts of 4 sequences with 70 frames per sequence.

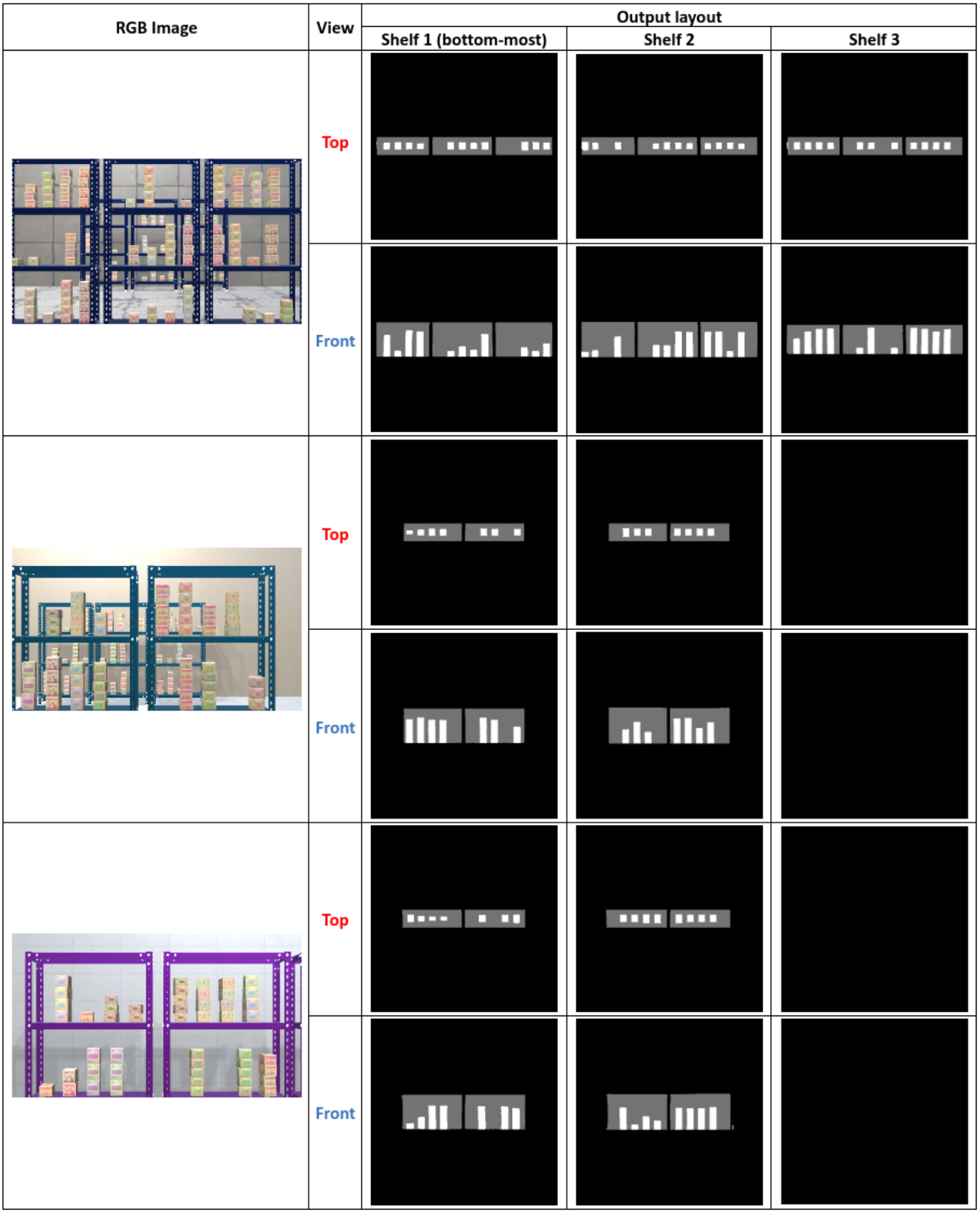

Figure 3.5: ***MVRackLay-Disc-4* Results**: Here, we present the results of our network tested on domain randomized data. The bottom-most shelf layout is shown in the left-most column, followed by the middle and top shelf (if visible). Observe the diversity of warehouse scenes captured (detailed in 3.3) and the top-view and front-view layouts predicted for the same.

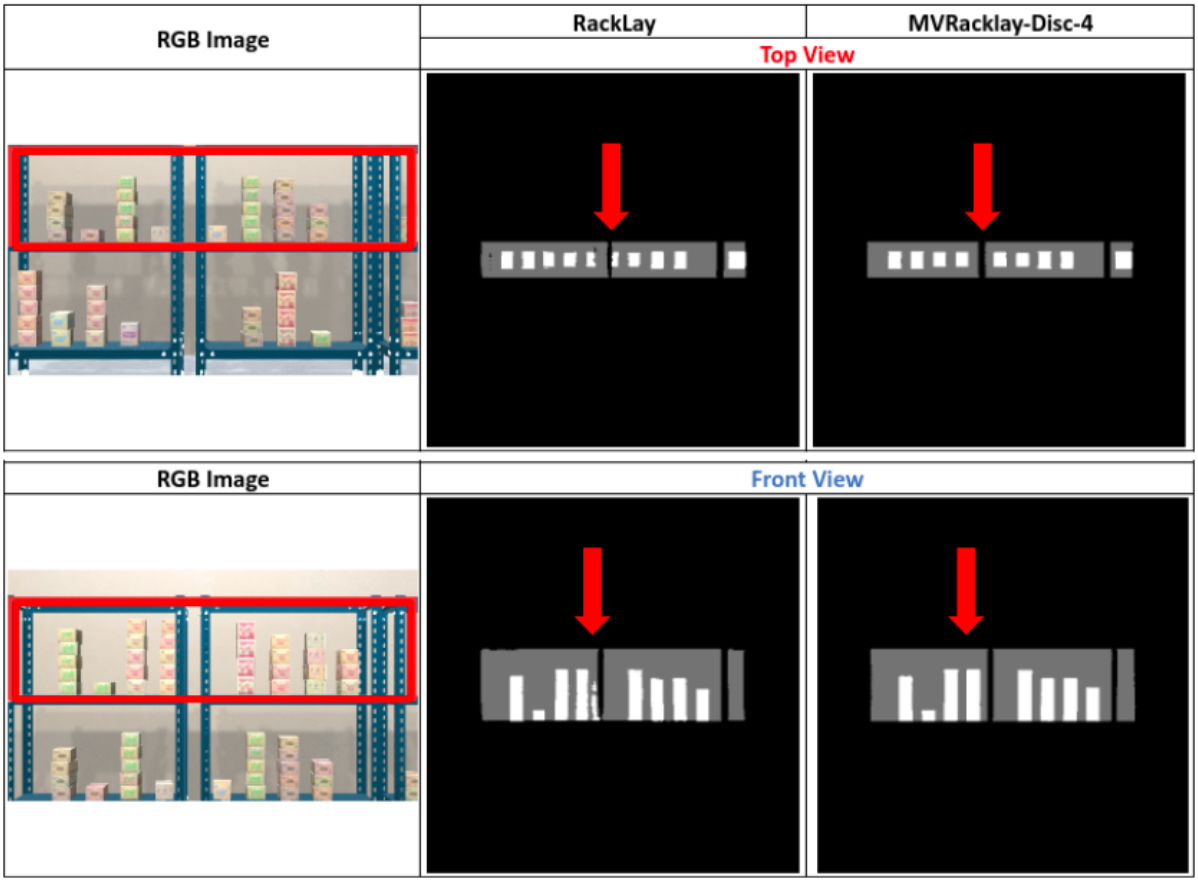

Figure 3.6: ***RackLay* vs. *MVRackLay-Disc-4***: Above, we compare qualitatively the results of *RackLay* and our *MVRackLay-Disc-4*. The shelf in focus is highlighted with a red border. Observe that our model removes the false positive in row 1, removes noise in row 2, and increases the sharpness of both box boundaries (both rows) and shelf edges (row 2).

.

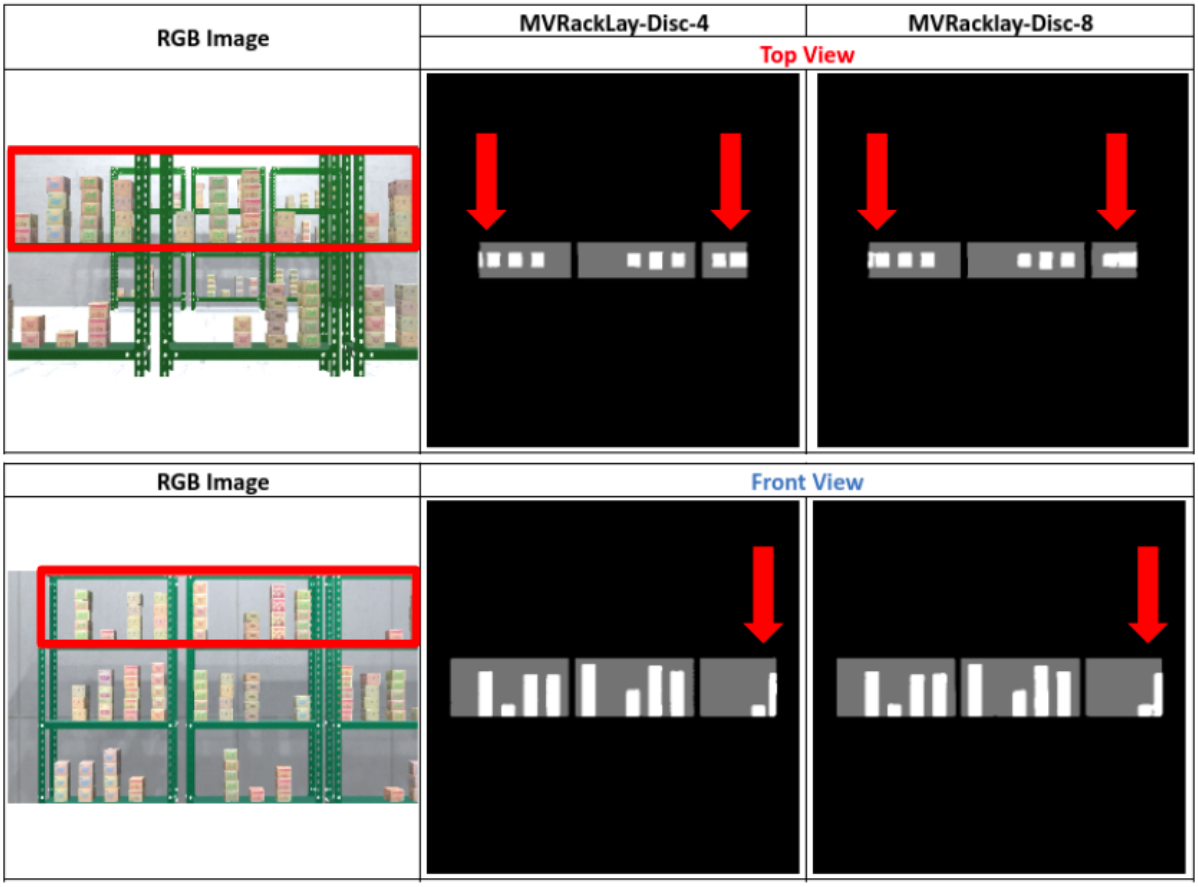

Figure 3.7: ***MVRackLay-Disc-4* vs. *MVRackLay-Disc-8***: The shelf in focus is highlighted with a red border. Better demarcations between adjoining boxes and less joining of abreast layouts are observed in the output of *MVRackLay-Disc-4* compared to its counterpart.

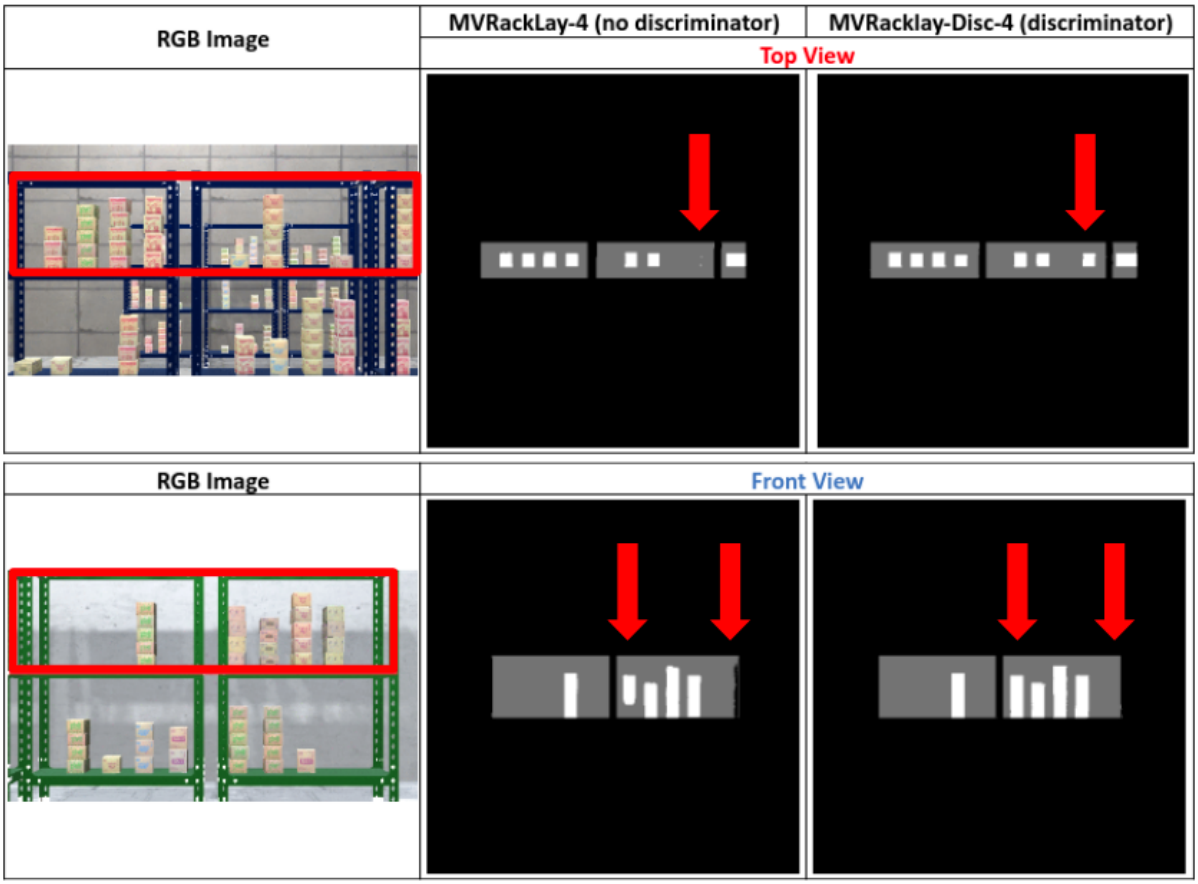

Figure 3.8: ***MVRackLay-4* vs. *MVRackLay-Disc-4***: The shelf in focus is highlighted with a red border. Observe how using a discriminator fixes the false negative in row 1 and improves predicted box boundaries and shelf boundaries in row 2.

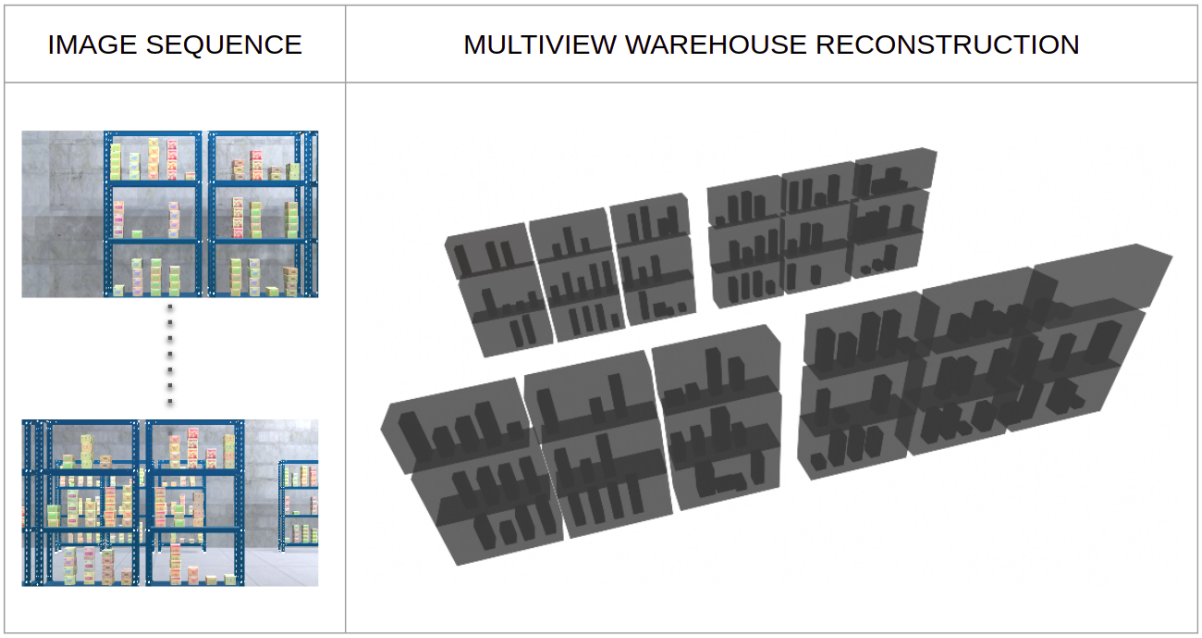

Figure 3.9: **Multi View Reconstruction** of the entire warehouse using four sequences covering 280 frames (70 frames each), using the layouts predicted by *MVRackLay-Disc-4*.

Chapter 4

Imagine2Servo: Intelligent Visual Servoing with Diffusion-Driven Goal Generation for Robotic Tasks

4.1 Introduction

In the preceding chapter, our focus was on addressing the challenge of estimating layouts for racks within warehouse settings. In this chapter, we broaden our scope by delving into the automation of object-reaching tasks within warehouses. Moreover, we extend our pipeline beyond the confines of warehouses to encompass manipulation and long-range navigation tasks, offering a more generalized approach. This chapter presents *Imagine2Servo*, an innovative framework for Intelligent Visual Servoing featuring Diffusion-Driven Goal Generation tailored for various robotic and warehouse tasks. This pipeline facilitates the generation of intermediate goal images, optimizing the image-based visual servoing (IBVS) algorithms to effectively guide robotic systems towards desired target locations.

We commence with a concise introduction outlining our approach, problem formulation, and the architectural components of *Imagine2Servo*, delving into training and implementation specifics. Subsequently, we introduce the datasets generated and utilized for training and testing purposes, followed by a comprehensive examination of experimental outcomes and results analysis. Our discussion encompasses both quantitative and qualitative assessments of the model's performance across diverse manipulation and navigation tasks. Furthermore, we conduct a thorough baseline study of methods pertinent to our work and showcase real-world experiments, culminating in an ablation study to further elucidate the efficacy of our approach.

4.2 Approach

4.2.1 Problem Formulation

At any given moment, our system receives an RGB image I_t from the robot's camera sensor, along with a textual prompt P that specifies the task to be accomplished. Our objective is to accurately predict

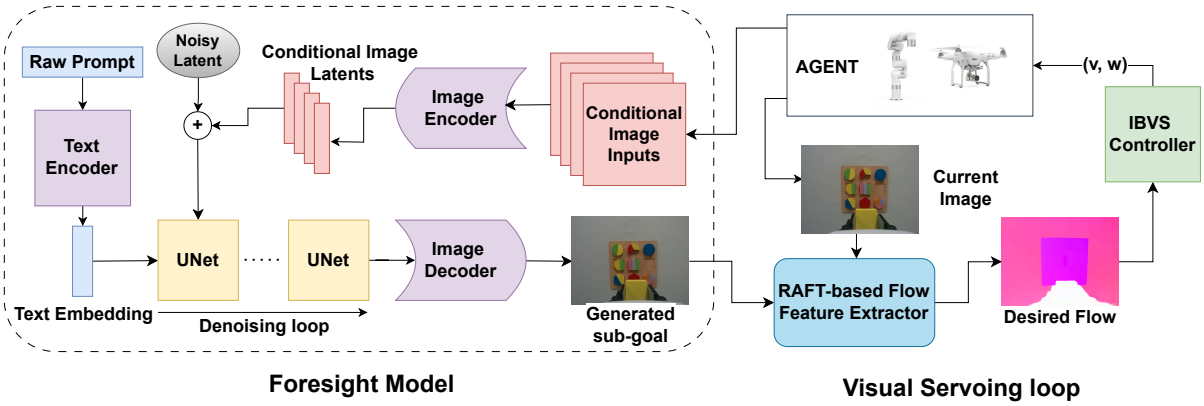

Figure 4.1: Our method employs an alternating loop of intermediate goal generation by the foresight model and execution of actions by the flow-based IBVS controller to perform the action described in the language instruction. The foresight model is a diffusion-based image-to-image translation model conditioned on the current monocular eye-in-hand camera input and any additional image observations available in the scene. The visual servoing controller consists of a flow-feature-based RTVS [2] controller that predicts subsequent actions in 6 DOF to reach the intermediate goal. This loop is continued until we solve the task.

and implement the necessary control commands, denoted as $[v_t, \omega_t]$, to fulfil the task described in the prompt. Tasks vary, ranging from navigating through a doorway to disconnecting a charger. Note that in traditional servoing algorithms, the final image I_g is usually provided to perform the task. This greatly limits the utility of the visual servoing models. To address this challenge, we divide our approach into two distinct phases. Initially, our framework is designed to conceptualize a subgoal I_g based on the task and current feedback from the camera I_t. Subsequently, we aim to attain this subgoal image I_g by employing a servoing algorithm. In the forthcoming sections, we will elaborate on the components of our Imagine2Servo framework. We begin by introducing our foresight model in Section 4.2.2.1, which leverages an image-editing algorithm to predict a subgoal image. Following this, we discuss the mechanics of our servoing framework 4.2.2.2, which facilitates reaching the subgoal. We then describe our overall framework in 4.2.3, our dataset in 4.3 and training details in 4.2.4. Our overall pipeline is summarised using Fig. 4.1.

4.2.2 Imagine2Servo Architecture

4.2.2.1 Visual foresight model

We see the task of generating the next subgoal for the servoing as editing the pixels of the current input image. Given the current image I_t and task description P, we aim to generate the subgoal image I_g using our foresight model or imagination module $p(I_g|I_t, P)$. To achieve this task, we use Instructpix2pix [48], an image editing framework, though any other image editing model can be used.

Instructpix2pix gives a good initialisation of the denoising model conditioned on the text prompt and the input image. We utilize the pretrained weights of this model and fine-tune it further to fit our task of learning visual foresight.

One significant benefit of this approach is the ability to integrate auxiliary camera data into the denoising process. This enhancement proves invaluable in scenarios where a solitary camera's perspective is inadequate for generating the necessary subgoal. An illustrative example of this is when the camera is affixed to the end-effector of a manipulator; in such instances, auxiliary image information I_{aux} from alternative viewpoints can significantly augment the planning of subgoals. Our empirical results demonstrate that the inclusion of auxiliary views markedly enhances the algorithm's efficacy, particularly in manipulation tasks. This methodology facilitates the incorporation of multi-camera feedback into the servoing algorithm, a capability that traditional servoing frameworks do not possess.

4.2.2.2 Image Based Visual Servoing Controller

In our approach, we employ RTVS [2] (Real-Time Visual Servoing) as the Image-Based Visual Servoing (IBVS) controller to reach the subgoals predicted by the foresight module. We use the RTVS servoing algorithm without any fine-tuning to the newer environments. RTVS is an iterative algorithm. At every timestep k, it seeks to optimize camera motion velocities (linear and angular) $\mathbf{v}_k$ to minimize the difference between the velocity-induced optical flow $\widehat{\mathcal{F}}(\mathbf{v}_k)$, and $\mathcal{F}(I_k, I_g)$, the optical flow of the current image I_k to the given subgoal image I_g.

$$\mathbf{v}_k^* = ||\widehat{\mathcal{F}}(\mathbf{v}_k) - \mathcal{F}(I_k, I_g)||_2^2 \tag{4.1}$$

In RTVS, this optimization is performed by inference on a pre-trained motion generation network (see [2] for details). $\widehat{\mathcal{F}}(\mathbf{v}_k)$ is calculated using the interaction matrix $L(Z_t)$ (described in Equation 4.2), which relates the camera velocity to the corresponding flow in the image plane.

$$L(Z_t) = \begin{bmatrix} -1/Z_t & 0 & x/Z_t & xy & -(1+x^2) & y \\ 0 & -1/Z_t & y/Z_t & 1+y^2 & -xy & -x \end{bmatrix} \tag{4.2}$$

This matrix is a function of the depth Z_t and the image coordinates (x, y), encapsulating the geometric relationship between motion in the robot's operational space and its projection onto the image plane. We used the FlowDepth trick described in [2] to use flow as a proxy for depth, thereby preventing the need for depth information for our pipeline. The cycle of optimizing the camera velocity, moving the camera by that velocity, and then computing the optical flow from the new observed image to the subgoal image is repeated until convergence, i.e. until the photometric error between the current observed image and the subgoal image is less than a threshold.

4.2.3 Overall Imagine2Servo Framework

Our Imagine2Servo framework is described in Algorithm 1. Given a task prompt P that specifies the desired task in natural language, the algorithm leverages the current observation I_t to initiate a process that dynamically guides the robot towards achieving the specified task. This is facilitated through the use of an Image Foresight Model $p(I_g|I_t, P)$ described in Section 4.2.2.1, which generates a series of intermediate goal images I_g. These images act as visual targets for the robot to pursue, effectively bridging the gap between the high-level task description and the robot's moment-to-moment actions.

At each time step, the algorithm assesses the alignment between the generated goal image and the current observation using a predefined convergence criterion ϵ_p. If the current goal is deemed to have been reached, the model generates a new goal image based on the updated observation and the initial task prompt P. This process repeats, with the robot making progress towards the final objective through a series of intermediate steps.

The action execution phase is managed by an Image-Based Visual Servoing (IBVS) controller $\phi(I_t, I_g^*)$, which adjusts the robot's actions to minimize the difference between its current state and the target state represented by the goal image as described in Section 4.2.2.2. This methodology allows the robot to navigate and manipulate objects within its environment effectively, translating high-level language prompts into a sequence of actionable visual goals. By continually updating these goals based on real-time feedback from the environment, our algorithm ensures adaptability and efficiency in the robot's task execution, demonstrating a novel approach to integrating language-based task instructions with robotic control mechanisms.

4.2.4 Training and implementation details

Imagine2Servo is trained on input-output image tuples with a corresponding text description. We use the pre-trained weights of InstructPix2Pix [48] and fine-tune their model. We train our model on a single 12 GB RTX 3090 GPU for 30,000 training steps with a learning rate of 5×10^{-5} and 500 learning rate warm-up steps. We use a batch size of 4 with 4 gradient accumulation steps. We use images of dimensions of 256 x 256, and the rest of the hyperparameters are similar to InstructPix2Pix. At test time, we use 100 inference steps with 1.5 and 7.5 as our image and text guidance parameters, respectively.

4.3 Dataset

Now, we describe the data collection process for training our *Imagine2Servo* framework. We collect our dataset using a variety of environments in different simulators in Pybullet [3], Unity [6], Habitat [4] and RLbench [5]. 1) *Door crossing*: Using the door samples from the PartNet-Mobility dataset [69] and the scenes in Habitat-sim[4], we generate a dataset of 720 and 400 video sequences, respectively, of a drone crossing the door, with 9 frames per sequence. The captured video sequences resemble that of a drone trying to cross a door frame. From every consecutive image pair of a trajectory, we sample input-

Algorithm 1 Imagine and Servo, Test-Time execution

Require: Test time t, language prompt *P*, current observation I_t, Image foresight model $p(I_g^*|I_t,l)$, IBVS controller $\phi(I_t, I_g^*)$, convergence constant for foresight model ϵ_p, convergence constant for IBVS controller ϵ_ϕ

$t \leftarrow 0$

$dist \leftarrow \infty$

Sample $I_g \sim p(I_g|I_t,\text{P})$ ▷ *Generate next intermediate goal*

while $dist \geq \epsilon_p$ **do** ▷ *Check if the last generated goal is the same generated goal.*

  $I_g^{last} \leftarrow I_g$

  while $||I_g - I_t|| \leq \epsilon_\phi$ **do** ▷ *Convergence criterion for IBVS*

    $\mathcal{F}_t :=$ predict-target-flow (I_g, I_t) ▷ *Predict the flow using the flow network*

    $\mathcal{L}_t :=$ calculate-interaction-data $(\mathcal{F}(I_g, I_t))$ ▷ *Calculate the interaction matrix*

    Execute $v_{t+1} :=$ relative-pose-network $(\mathcal{L}_t, \mathcal{F}_t)$ ▷ *Predict and execute action on the robot*

    $I_{t+1} \leftarrow$ robot-observation ▷ *Get the current observation after action execution*

    $t \leftarrow t + 1$

  end while

  Sample $I_g \sim p(I_g|I_t,\text{l})$ ▷ *Generate the next intermediate goal.*

  $dist \leftarrow ||I_g^{last} - I_g|$

end while

output image tuples. The text prompt is expected by the InstructPix2Pix backbone, which describes the task being performed, for example, to "cross the door". We diversify the scene elements during video creation to create a synthetic scene resembling its real-world counterpart. We randomly sample the initial pose and randomize lighting conditions, wall textures, and floors in every video we generate. 2) *Reaching tasks in RLBench [5]*: We choose 17 tasks from RLbench and collect 100 demonstrations for every variation of a task. Every task in RLBench is a composition of reaching and manipulating actions. So, we collect data to learn reaching by discarding manipulation actions like grasping from the demonstrations. The text prompt is set to "reach the *object*". *Reaching tasks in Unity [6]*: Our methodology involves the creation of a dataset comprising 8,000 trajectories, each consisting of 15 frames per trajectory. Utilizing textual descriptions such as "Go to the first stack on the second shelf of the third rack", we generate trajectories simulating a drone's approach to the specified target from various starting positions. Our model was fine-tuned for 70,000 steps on this data.

We sample 20 novel eye-in-hand camera viewpoints in Habitat and pybullet (with domain randomization) during test time evaluation. For RLbench, we use unseen demonstrations during training. We sample 20 novel arrangements for each variation of a task during test-time evaluation. For warehouse scenes in Unity, We generate 10 new warehouses with randomized conditions (as described in 3.3) and test on 20 unseen viewpoints (2 per warehouse scene).

4.4 Experiments and Analysis

We evaluate the performance of *Imagine2Servo* for two different tasks: 1) *Door crossing*. This tests the ability of our algorithm to perform long-range servoing. Experiments are conducted in two simulation environments: PyBullet [3] and Habitat [4]. 2) *Reaching*. This tests the applicability of our algorithm to manipulation tasks in RLBench [5] and on a real robot. Experiments are performed both in simulation (RLBench [5]) and on a real robot setup. Furthermore, we demonstrate the utility of our approach in simulating object-reaching tasks within warehouse environments in the Unity [6] simulator. Please note that we select three equally spaced sub-goals generated by the foresight model to demonstrate the qualitative results of our model in fig 4.3, 4.4, 4.8, 4.5.

4.4.1 Baselines

We compare our *Imagine2Servo* framework with the following approaches:

- **RTVS** [2] is an MPC-based formulation with multi-step ahead prediction. To RTVS, we provide the privileged real final image as the target image and report the success rate.

- In **Pose-Diffusion**, we implement a simplified version of the Diffusion Policy [70] to predict the next action, conditioned on the current eye-in-hand camera observation.

	PyBullet	Habitat-Sim	RLBench			Unity		
	Success Rate		Success Rate	Trans. Error	Rot. Error	Success Rate	Trans. Error	Rot. Error
RTVS[2]	0.45	0.5	0.25	0.18	0.38	0.35	0.15	0.26
Pose-Diffusion	0.4	0.2	0.2	0.27	0.45	0.25	0.31	0.29
Cam-Axis	0.2	0.1	0.1	0.39	0.71	0.15	0.35	0.26
Imagine2Servo-Single-View (Ours)	**0.9**	**0.95**	0.65	0.09	0.08	**0.75**	**0.12**	**0.05**
Imagine2Servo-Multi-View (Ours)	-	-	**0.75**	**0.03**	**0.02**	-	-	-

Table 4.1: **Quantitative results**: We benchmark the two different versions of our network - *Imagine2Servo-Single-View* and *Imagine2Servo-Multi-View*, along with three baselines- *RTVS*[2], *Pose-Diffusion* and *Cam-Axis* (as described in Section 4.4). We report the average success rate for all tasks, and additionally, 6 DOF pose error for manipulation tasks in RLBench[5]. Imagine2Servo achieves the best success rate, demonstrating its applicability in different tasks. *Imagine2Servo-Single-View* outperforms other baselines in navigation task while *Imagine2Servo-Multi-View* performs the best in the manipulation tasks in RLBench [5].

- In the **Cam-Axis** baseline, we implement a simple controller that moves the camera along the camera axis from the initial pose.

We compare these baselines to two versions of our algorithm. **Imagine2Servo-Single-View** uses the current eye-in-hand camera view as the only conditional image input. In **Imagine2Servo-Multi-View**, we train Imagine2Servo to use additional input from the overhead camera as the conditional input to the foresight model, along with the eye-in-hand monocular camera input.

4.4.2 Quantitative Comparison

We measure the performance of all algorithms in terms of success rate. For door crossing, the camera is modelled as if it were mounted on a cylindrical drone of radius 15 cm. If the camera crosses the door, the trial is marked successful. For reaching tasks, we report the linear and angular errors as well as the success rate. The angular error is measured as the norm of quaternion difference. If the linear error < 3 cm and angular error < 0.03, the trial is considered successful. For both tasks, we also check for collision after every controller step and, upon collision, declare the trial a failure. We show success rate comparisons in table 4.1.

Imagine2Servo-Single-View outperforms all the other baselines with a significant improvement in navigation tasks. It attains 90% and 95% success rates, respectively, in navigating doors in PyBullet [3] and Habitat-Sim [4] environments, with 45% absolute improvement over RTVS[2], which employs a single target view to servo to the target. Additionally, we attain a 75% success rate in object-reaching tasks within the warehouse scenes simulated in Unity [6]. Imagine2Servo-Single-View also outper-

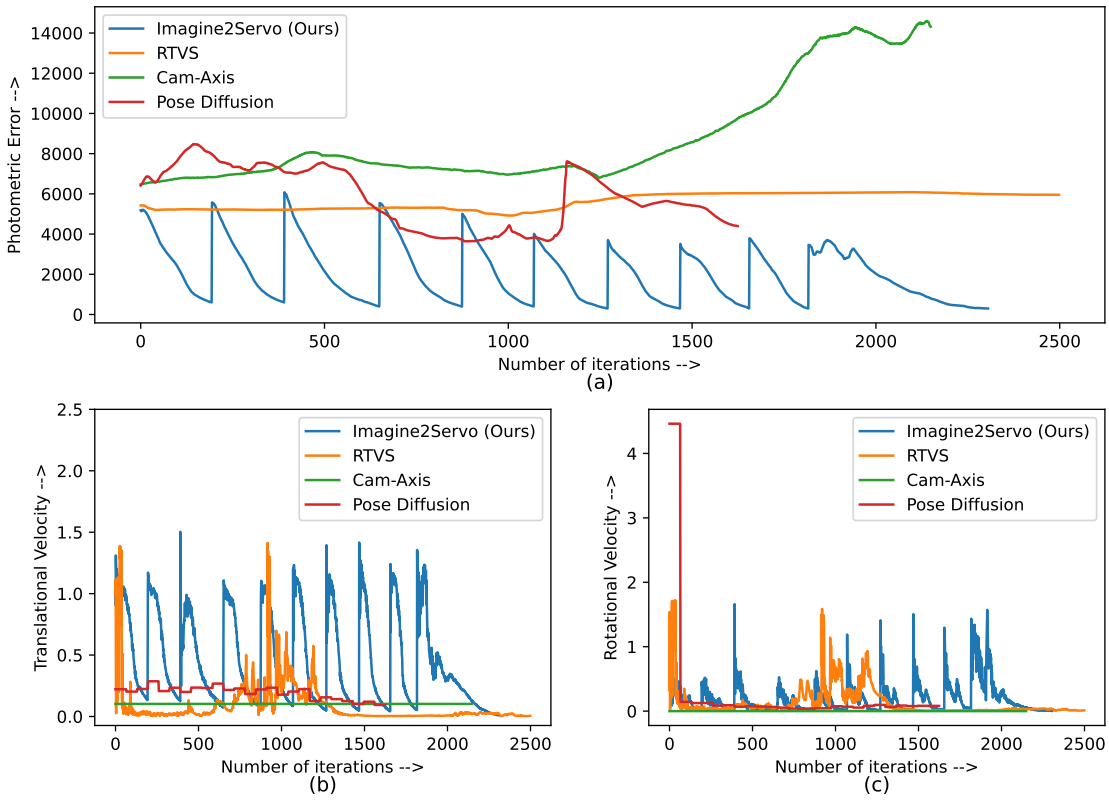

Figure 4.2: **(a)** We compare the photometric error convergence for each baseline. *Imagine2Servo* converges to each of the subgoals (where the error becomes close to zero) while other methods fail to converge. **(b)** Evolution of the magnitude of the translational velocity in each step. **(c)** Evolution of the magnitude of the rotational velocity in each step.

forms Pose-Diffusion, which directly predicts the next action instead of the next image. Since our model reasons in image space, it has the advantage of using a strong diffusion backbone trained on millions of images which is not the case for pose-diffusion baseline. For reach tasks in RLBench [5], Imagine2Servo-Multiview-View outdoes other baselines with a 75% success rate, a 10% increase over Imagine2Servo-Single-View owing to the extra information provided by the additional camera views. It achieves the lowest linear and angular error, thereby performing precise alignments, which is an important factor in manipulation tasks.

To further analyze the overall performance of our model, we perform a comparative study of photometric error convergence (fig. 4.2(a)) and evolution of translational and rotational velocities (fig. 4.2(b),(c)) in each step. These experiments are performed on HM3D [71] scene (e.g. 3 in fig. 4.3). The photometric error of Imagine2Servo smoothly converges without any oscillations to below 300 (for all sub-goals), while other baselines fail to converge beyond the error of 4000. We also observe a bounded velocity profile in Imagine2Servo while sudden accelerations are observed in RTVS [2]. The initial tra-

jectory along the Cam-Axis baseline tries to move in the right direction towards the goal but eventually collides with the front wall. Pose-Diffusion at first converges, but as it approaches the target, it fails to generate appropriate velocity commands and stalls in the same place before diverging. Imagine2Servo gradually decelerates as it approaches the sub-goal image, giving us a smooth motion profile.

4.4.3 Qualitative Results

Fig. 4.3 and 4.4 show qualitative results for door crossing and reaching, respectively. Despite encountering significant variations in translations, with distances spanning up to 10 meters and rotations up to 80 degrees in roll, pitch, and yaw, our system reliably accomplishes the challenging task of crossing the door. This resilience demonstrates the robustness and adaptability of our approach to handling diverse real-world conditions, paving the way for versatile applications in various environments. For manipulation tasks, we conduct experiments in RlBench [5], which is a well-established benchmark for manipulation policies. We pick 3 RLBench tasks with varying complexities to demonstrate the wider applicability of our model. With sub-goal guidance, even when the window is partially visible in the initial image (e.g. 2 in fig. 4.4), Imagine2Servo-Single-View achieves precise final alignment of the end-effector. It also correctly interprets the text input to perform necessary action when multiple objects are present in the scene. For the task of shape sorter, the foresight model is able to correctly reason between different objects (e.g. 3, 4 in fig. 4.4) to generate the next sub-goal.

Figure 4.5 provides a comprehensive overview of the outcomes yielded by our model within the Unity [6] Simulator environment. We present results obtained across diverse warehouse scenes, each characterized by distinct lighting conditions. Notably, Imagine2Servo is tasked with accurately deducing the layout of racks, shelves, and stacks within the given instruction to successfully execute the assigned task. Our model adeptly identifies the pertinent elements, including the rack structure, the specific shelf situated on the rack, and the targeted stack as specified in the textual instruction. Noteworthy is our model's capability to navigate successfully even when confronted with limited visibility of details in the initial view, as exemplified in row 3 of Figure 4.5. Through the strategic utilization of sub-goals generated by the foresight model, our framework demonstrates iterative convergence towards the desired target, showcasing its robustness and adaptability in navigating complex warehouse environments.

Fig 4.6 summarizes the improvements of Imagine2Servo-Multi-View over the Imagine2Servo-Single-View baseline. Imagine2Servo-Multi-View correctly predicts the next sub-goal when the target object is not visible (row 2, 3, 4 in fig 4.6) in the current view or is occluded behind other objects (row 1 in fig 4.6) in the scene. In such cases, Imagine2Servo-Single-View diverges as the foresight model fails to generate the next relevant sub-goal. Using the auxiliary information from the overhead camera view, the multi-view model correctly generates the next intermediate view and guides the controller towards the target.

The qualitative results described above show the applicability of Imagine2Servo in navigation as well as manipulation tasks. Our model can significantly benefit applications where a robot needs to precisely navigate the space with occlusion. Even in poorly textured environments, our flow-based

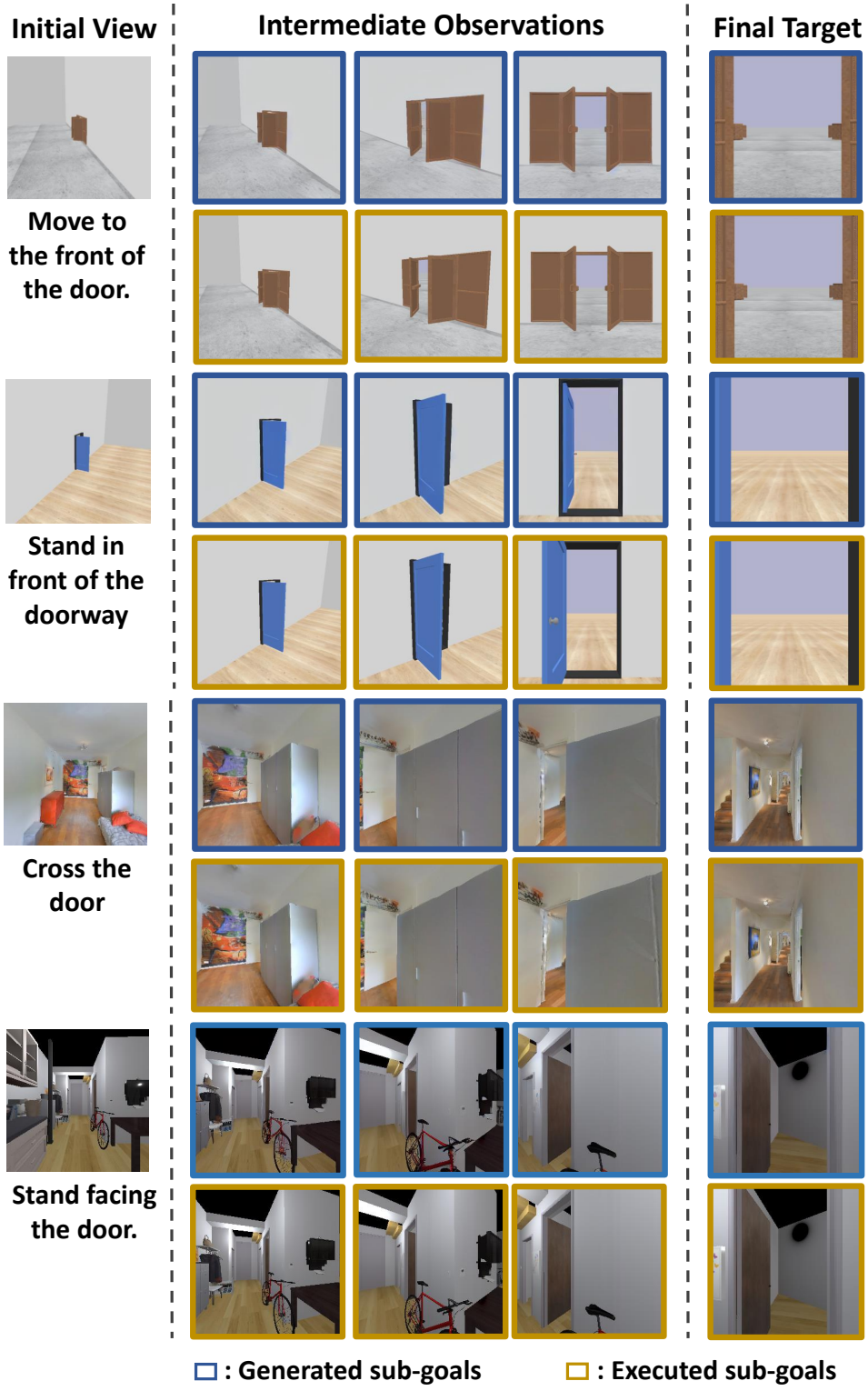

Figure 4.3: We visualize the foresight model generated sub-goals and the pose reached after the execution of actions by the controller for the task of navigation through a door in PyBullet [3] and Habitat-Sim [4]. *Imagine2Servo-Single-View* successfully converges for large translations and rotations of the initial view with respect to the target pose.

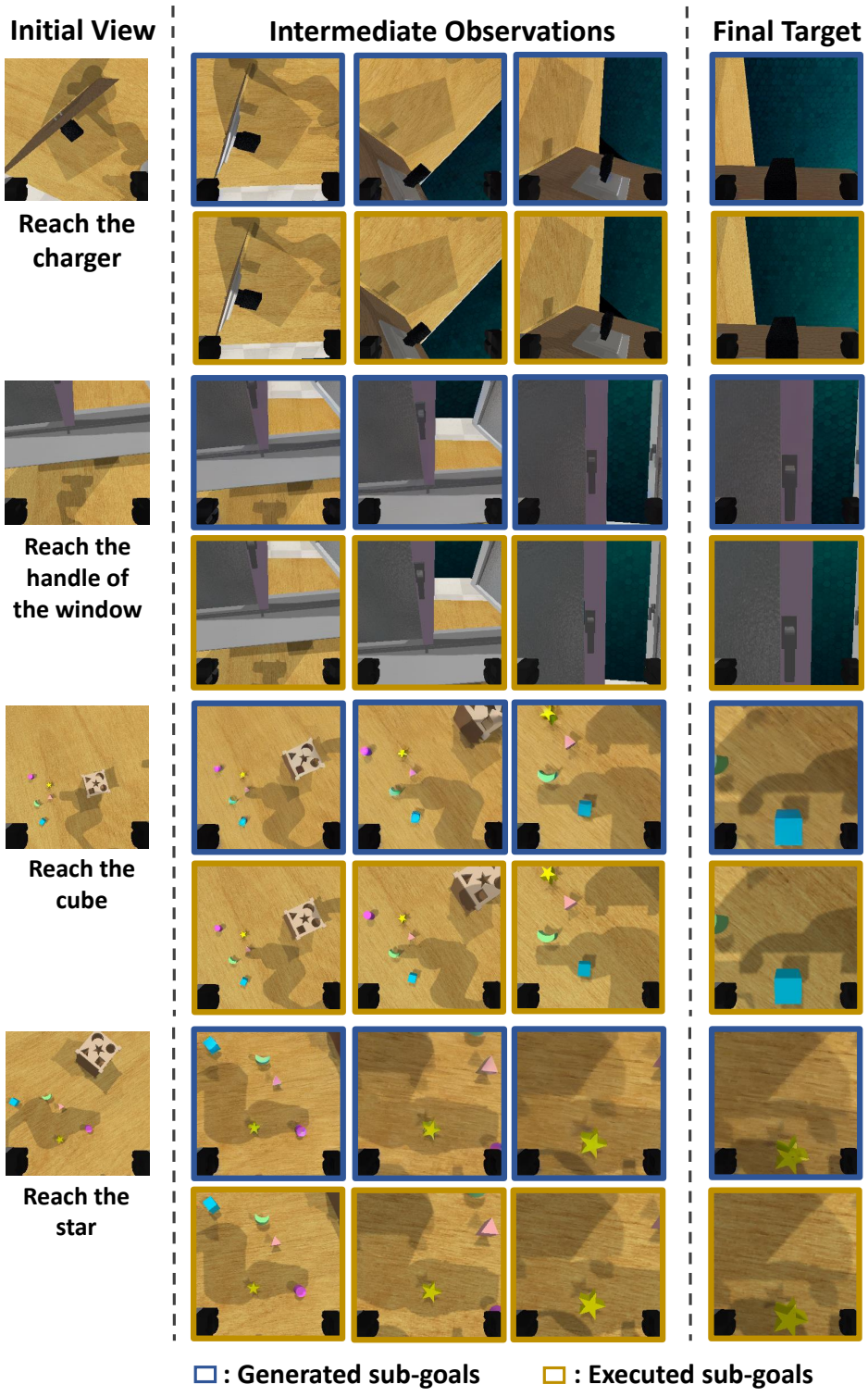

Figure 4.4: We visualize the sub-goals generated by the foresight model and the pose reached after the execution of actions by the controller for three different tasks in RLBench [5]. *Imagine2Servo-Single-View* converges even when the target object is partially visible in the scene. When multiple objects are present in the scene, we correctly differentiate between ”cube” and ”star” to reach the target object.

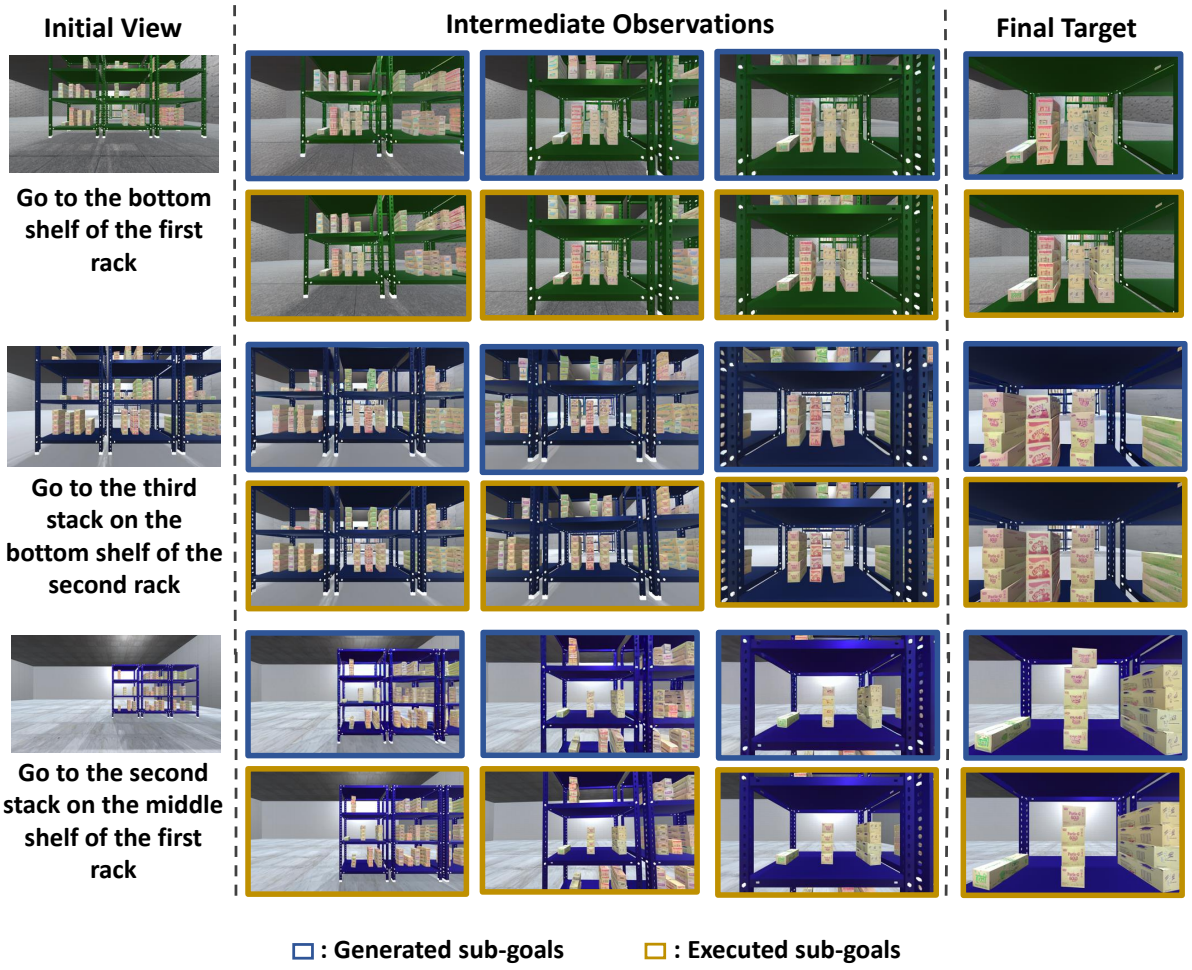

Figure 4.5: We visualize the sub-goals generated by the foresight model and the pose reached after the execution of actions by the controller for three different instructions in Unity simulator [6]. *Imagine2Servo-Single-View* identifies the correct rack and shelf and generates the correct subgoals to reach the target.

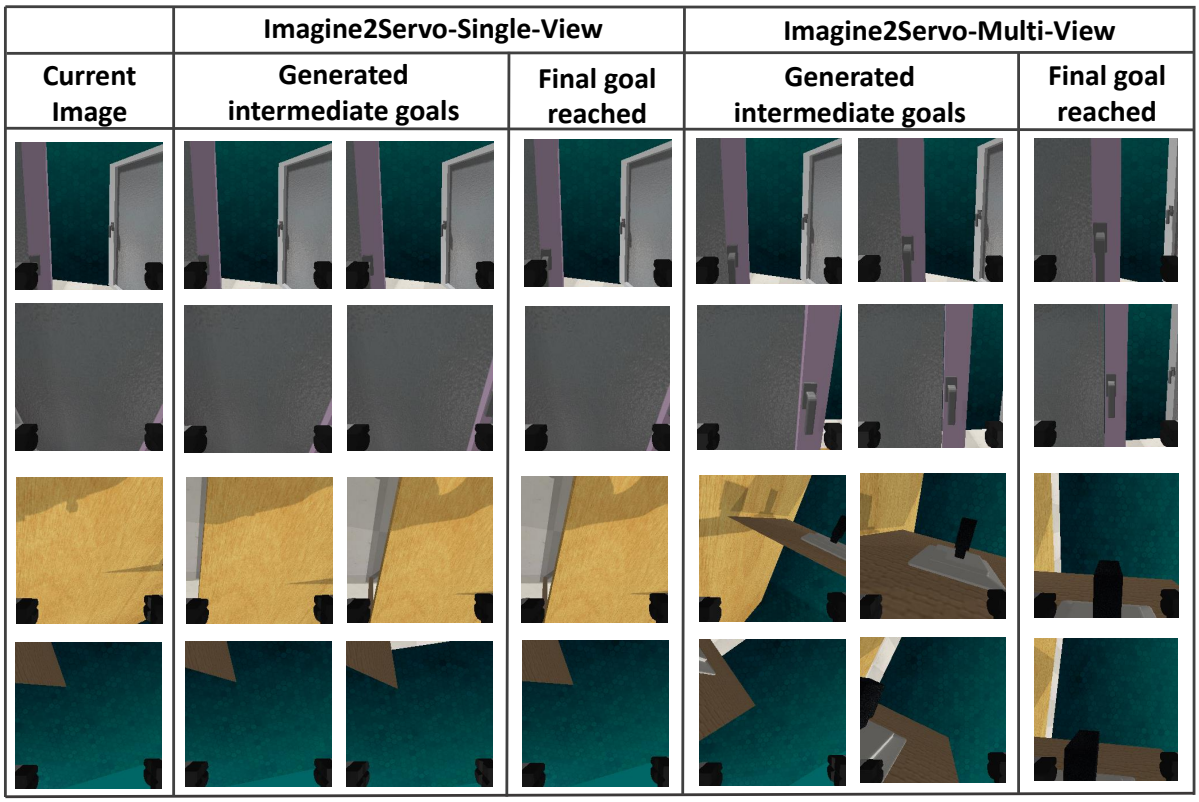

Figure 4.6: We qualitatively compare the performance of *Imagine2Servo-Single-View* and *Imagine2Servo-Multi-View* for the task of "Reach the handle of the window" (row 1, 2) and "reach the charger charger" (row 3, 4) in RLBench [5]. When the target is not visible or is occluded behind other objects in the initial image, *Imagine2Servo-Single-View* fails to converge, whereas *Imagine2Servo-Multi-View* generates sub-goals that converge to the final target.

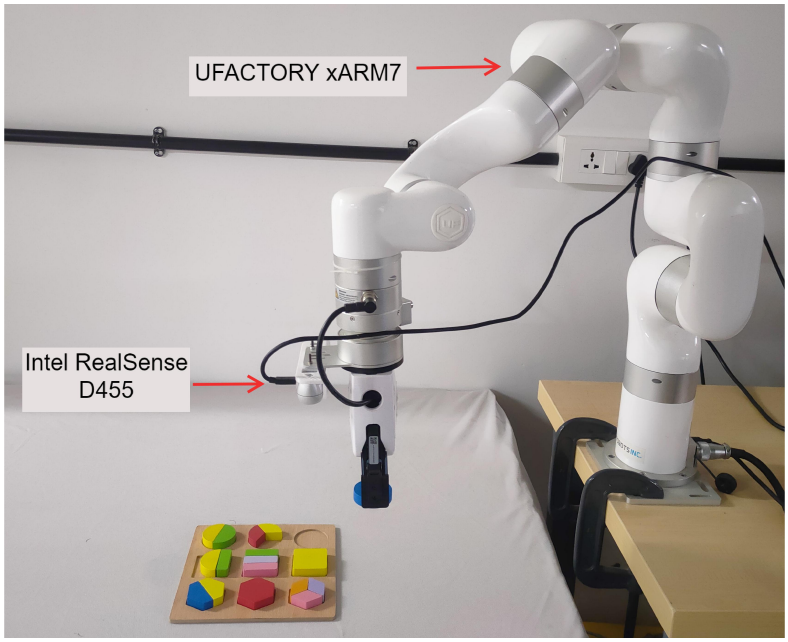

Figure 4.7: Real-world setup: We use UFACTORY xARM7 and eye-in-hand Intel RealSense D455 sensor input for real-world experiments.

IBVS controller extracts the necessary features to perform the required task. Flow-based IBVS controller compensates for errors in the foresight model and executes necessary actions to complete the task successfully.

4.4.4 Real World

We set up manipulation experiments* in the real world on UFACTORY xARM7 with an Intel RealSense D455 sensor fixed on its end-effector (fig 4.7). We demonstrate the following tasks: **1.** Placing a shape in the shape sorter **2.** Shape stacking. We generate 300 trajectories with 15 frames per trajectory for each shape and finetune our model by sampling input-output image tuples from the above video demonstrations with a language prompt explaining the task. Unseen poses and randomized positions are used during test time. Our model performs reasonably well, achieving success rates of 8/10 for task 1 and 7/10 for task 2. Please see the supplementary material for videos.

Imagine2Servo captures the semantic information in the scene and correctly sorts circles and squares in the respective shape sorters (rows 1, 2 in fig 4.8). For the task of stacking the shape where precise alignment is important to successfully stack the object, our model performs exceptionally well. Even when the stacker and the sorter are partially visible in the initial image, the foresight model generates

*Would like to acknowledge Gunjan Gupta for establishing the servoing pipeline on xARM7 and for overseeing the data collection process for various tasks on the physical robot. Gunjan's contributions have been invaluable in conducting a range of experiments in real-world settings.

relevant sub-goals and converges to the final target. Despite the actuation noise in the real-world setup, our method adapts well and converges successfully in the execution of the task.

4.4.5 Ablation Study

We perform ablation experiments to comprehend the advantage of using sub-goals in image-based servoing. At test time, we expect the flow-based IBVS controller to proficiently match the features of the current and the goal images generated by the foresight model. Optical flow calculations are constrained by the motion between two consecutive image frames. Therefore foresight model should be trained to predict the next sub-goal, which would give us good features for servoing. We study the convergence of Imagine2Servo as we uniformly sample *n* frames per trajectory to form the training set $S = \{I_1, ..., I_n\}$. The consecutive frames from this subset are then used to train the image foresight model.

From fig. 4.9, we see that as *n* increases, the optical flow features between the current and the generated image become more prominent and lead to an increase in success rate (fig. 4.10). For $n = 2$, we directly predict the final goal image without predicting any sub-goals. In this case, the target image is often incorrect as the foresight doesn't have enough information from the current view to directly generate the target view. Even if the target view is correct, the IBVS controller fails to converge to the desired target location. With the increase in the value *n*, from n=4, the foresight model output becomes more grounded, yet the IBVS controller fails to converge due to the lack of good flow features. For *n* = 9, the IBVS controller converges to the final target with balanced in-range sub-goal generations by the foresight model. This shows that predicting sub-goals instead of one final goal helps ground the foresight model and makes the controller more robust as it also learns to compensate for any error in the generated sub-goals (e.g. 2 in fig 4.3). Further increasing the value of *n* leads to very little improvement in the success rate as it stabilizes for values of $n \geq 9$.

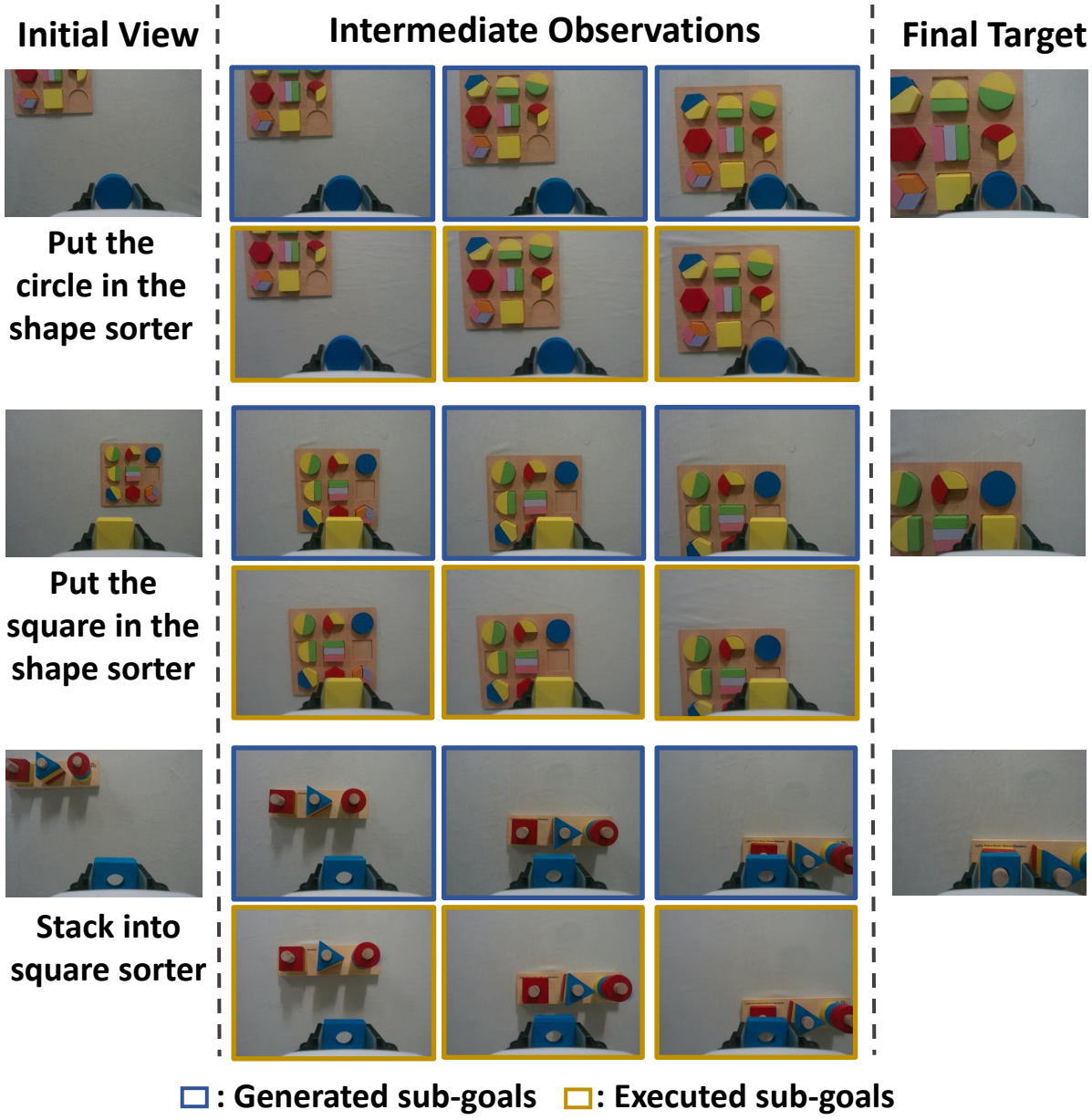

Figure 4.8: For real-world tasks, visualize the target reached with the sub-goals and executions for the task of placing the shape in the shape sorter and stacking the shape. We achieve a precise end-effector pose and drop the shape correctly in its shape sorter.

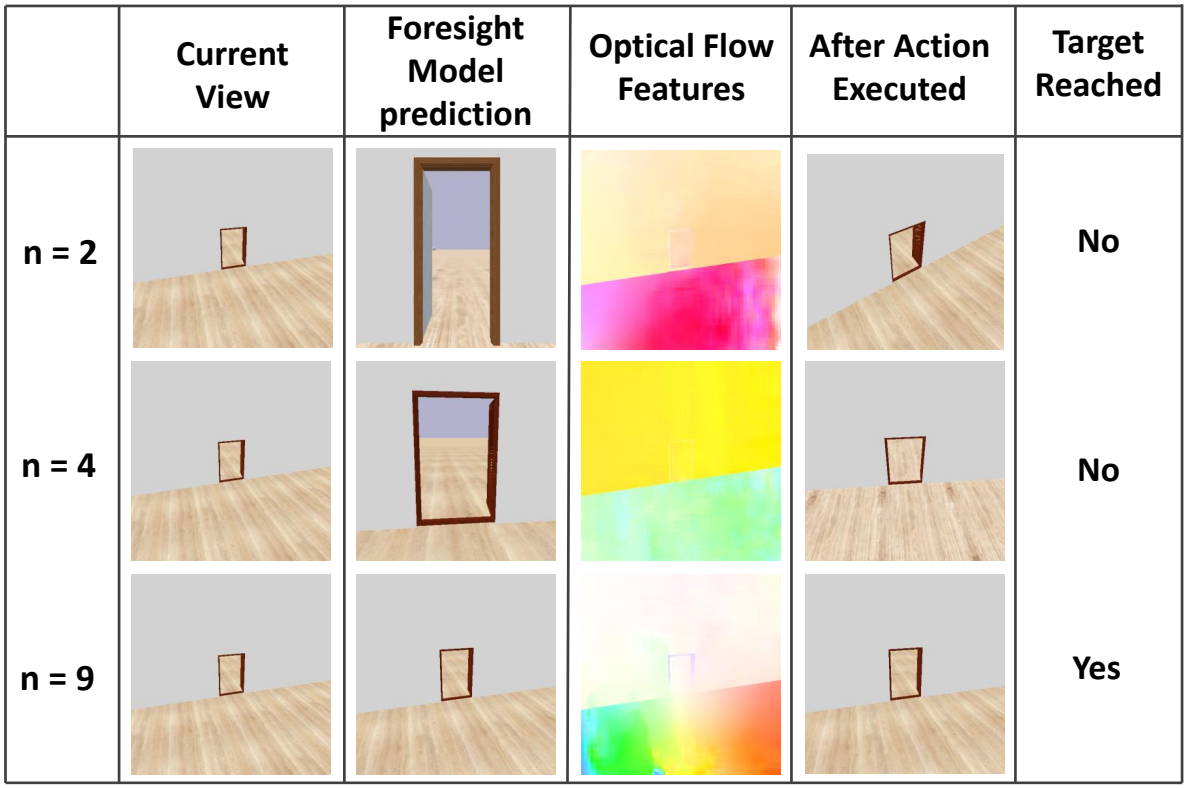

Figure 4.9: Results of qualitative comparison of sub-goal generations and IBVS controller convergence with the change in the value of n (n is the sampling frequency used to generate the training set). We show results for the task of reaching the door in PyBullet [3]. For a given current view, we compare the foresight model predictions for varying values of n. As n increases, the accuracy of the foresight model predictions improves. For $n = 2, 4$, the IBVS controller fails to reach the sub-goal, leading to divergence from the final target. However, as n increases, the state achieved after the execution of the IBVS controller aligns more closely with the target image generated by the foresight model. Refer to 4.4.5 for more details.

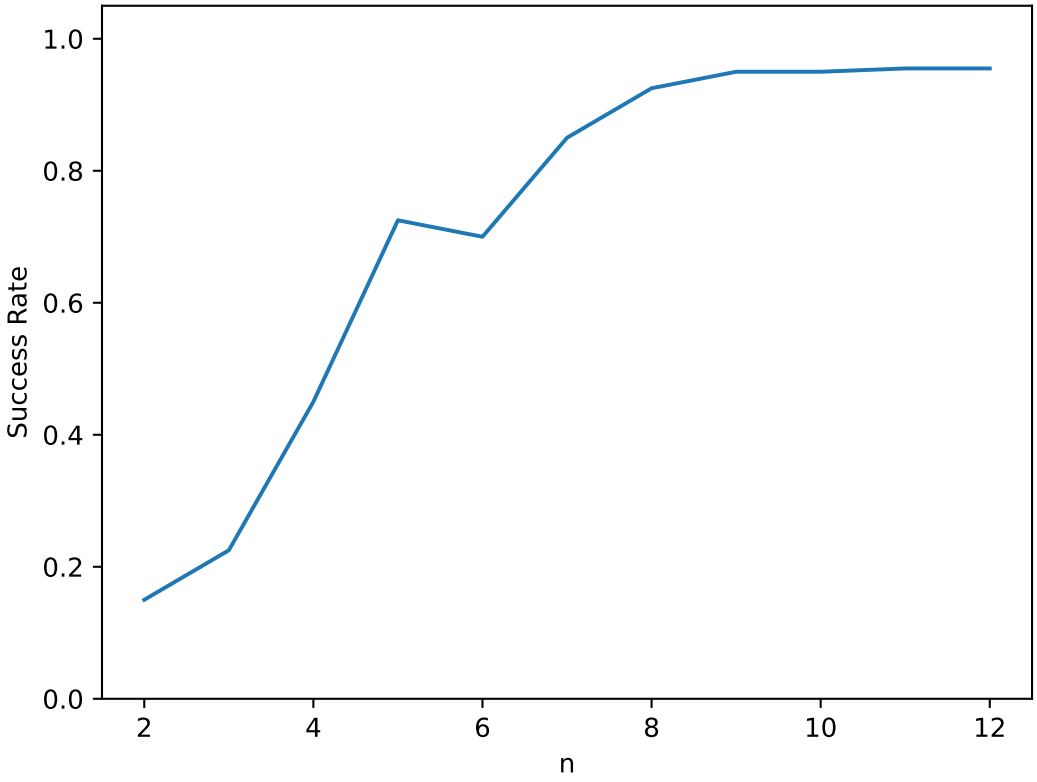

Figure 4.10: Ablation study to elucidate the impact of altering the sampling frequency of the training set, denoted by n, on the success rate (refer to Section 4.4.5).

Chapter 5

Conclusions

In the initial segment of this thesis, we introduce MVRackLay, an innovative framework tailored to undertake multi-view and multi-layered layout estimation for all racks discernible in each frame of a monocular image sequence. Setting itself apart from existing approaches, MVRackLay leverages temporal information across image frames to augment the accuracy of layout predictions. Additionally, we unveil a comprehensive pipeline designed for reconstructing the entire warehouse in 3D based on the predicted shelf-centric layouts. The versatility of MVRackLay is underscored by experimental findings across a spectrum of warehouse scenarios, demonstrating its substantial superiority over previous baselines adapted for comparable tasks.

Improving MVRackLay's resilience to the dynamic shifts inherent in warehouse settings, such as the movement of objects, stands as a fundamental requirement for its effective deployment in real-world contexts. Subsequent research endeavours may concentrate on crafting algorithms adept at identifying and adjusting to these dynamic elements within the scene, thereby ensuring precise layout estimations even amidst challenging circumstances. Additionally, forthcoming investigations might delve into methodologies that jointly infer the arrangement of racks while pinpointing the exact locations of individual items within the scene, drawing upon both geometric principles and appearance-driven insights.

Moreover, enhancing MVRackLay's capacity to comprehend semantic contexts within warehouse scenes, including the discernment and categorization of diverse objects and spatial regions, holds promise for facilitating more nuanced applications. Prospective efforts could integrate advanced techniques like semantic segmentation and object recognition into the framework, enriching scene annotations and streamlining higher-level decision-making processes. Thorough validation and benchmarking initiatives, conducted on expansive real-world datasets encompassing a spectrum of warehouse environments, offer an avenue to glean valuable insights into MVRackLay's adaptability and performance across varied conditions. This future research may entail the meticulous curation and annotation of datasets representative of diverse warehouse layouts, object configurations, and environmental factors, thus providing a comprehensive evaluation of MVRackLay's robustness and scalability.

In the latter portion of this thesis, we unveil Imagine2Servo, an innovative framework poised to substantially enhance the capabilities of visual servoing. By dynamically generating intermediate goal

images, Imagine2Servo enables robust task execution across diverse environments, eliminating the need for predefined goal imagery. This approach effectively mitigates key limitations inherent in traditional visual servoing methods, including the necessity of goal images, challenges stemming from limited overlap between initial and target views, and reliance on single-camera feedback. Our empirical findings showcase the vast potential of Imagine2Servo in elevating robotic navigation and manipulation abilities, leveraging diffusion-based image editing to generate subgoals with precision. This advancement positions Imagine2Servo as a pioneering benchmark in the field of visual servoing, charting a course towards more autonomous and adaptable robotic systems.

While Imagine2Servo shows promising results in basic tasks, expanding its capabilities to handle complex manipulation scenarios could significantly enhance its performance and adaptability in real-world applications. Future research could focus on joint training of the foresight model and IBVS controller to make diffusion more aware of the capabilities of the IBVS controller and vice versa. Additionally, investigating the integration of machine learning techniques presents an opportunity to leverage past experiences and adapt the strategies of a model accordingly. This could entail training neural networks to anticipate optimal subgoals based on environmental cues and robot dynamics, thereby enhancing task execution efficiency and enabling adaptation to novel scenarios. Moreover, optimizing Imagine2Servo for real-time performance is imperative for its practical deployment in real-world robotic systems. Future efforts could concentrate on developing streamlined algorithms and parallel computing techniques to facilitate rapid computation of intermediate goal images, ensuring seamless integration with high-speed robotic control systems.

While empirical evidence showcases Imagine2Servo's efficacy in navigation and basic manipulation tasks, there's potential to broaden its capabilities to encompass more intricate manipulation scenarios. This expansion could encompass tasks involving object interactions, tool manipulation, or delicate object handling, requiring the formulation of strategies to generate subgoals that factor in object dynamics, contact constraints, and task-specific objectives. Additionally, despite Imagine2Servo's advancements over single-camera feedback systems, there's room for further improvement through the integration of feedback from multiple sensors such as depth sensors, LiDAR, or inertial measurement units (IMUs). Exploring fusion techniques to seamlessly incorporate multi-sensor data into the framework could bolster Imagine2Servo's robustness and adaptability in complex environments, thus enhancing its overall performance and utility in real-world applications.

Related Publications

1. **Pranjali Pathre**, Anurag Sahu, Ashwin Rao, Avinash Prabhu, Meher Shashwat Nigam, Tanvi Karandikar, Harit Pandya, and K. Madhava Krishna. "MVRackLay: Monocular Multi-View Layout Estimation for Warehouse Racks and Shelves." In *2022 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pp. 1476-1482. IEEE, 2022

2. **Pranjali Pathre**, Gunjan Gupta, M. Nomaan Qureshi, Mandyam Brunda, Samarth Brahmbhatt, K. Madhava Krishna. "Imagine2Servo: Intelligent Visual Servoing with Diffusion-Driven Goal Generation for Robotic Tasks." Under review and in pre-proceedings of the *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

Bibliography

[1] Meher Shashwat Nigam, Avinash Prabhu, Anurag Sahu, Tanvi Karandikar, Puru Gupta, N. Sai Shankar, Ravi Kiran Sarvadevabhatla, and K. Madhava Krishna. Monocular multi-layer layout estimation for warehouse racks. In *Proceedings of the Twelfth Indian Conference on Computer Vision, Graphics and Image Processing*, ICVGIP '21, New York, NY, USA, 2021. Association for Computing Machinery.

[2] M. Nomaan Qureshi, Pushkal Katara, Abhinav Gupta, Harit Pandya, Y V S Harish, AadilMehdi Sanchawala, Gourav Kumar, Brojeshwar Bhowmick, and K. Madhava Krishna. Rtvs: A lightweight differentiable mpc framework for real-time visual servoing. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3798–3805, 2021.

[3] Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. `http://pybullet.org`, 2016–2021.

[4] Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, Devi Parikh, and Dhruv Batra. Habitat: A Platform for Embodied AI Research. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

[5] Stephen James, Zicong Ma, David Rovick Arrojo, and Andrew J. Davison. Rlbench: The robot learning benchmark and learning environment. *IEEE Robotics and Automation Letters*, 5(2):3019–3026, 2020.

[6] Unity real-time development platform — 3d, 2d vr ar engine, 2021. Accessed: 13-09-2021.

[7] Yan Wang, Wei-Lun Chao, Divyansh Garg, Bharath Hariharan, Mark Campbell, and Kilian Q Weinberger. Pseudo-lidar from visual depth estimation: Bridging the gap in 3d object detection for autonomous driving. In *CVPR*, 2019.

[8] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. Pointrcnn: 3d object proposal generation and detection from point cloud. In *CVPR*, 2019.

[9] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Maskrcnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[10] Peter Buxbaum. Many warehouses don't have wms, Aug 2018.

[11] Trends in warehouse automation-2021, Oct 2020.

[12] Olaf Ronneberger and Fischer. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, 2015.

[13] Thomas Roddick, Alex Kendall, and Roberto Cipolla. Orthographic feature transform for monocular 3d object detection. *arXiv preprint*, 2018.

[14] Pranjali Pathre, Anurag Sahu, Ashwin Rao, Avinash Prabhu, Meher Shashwat Nigam, Tanvi Karandikar, Harit Pandya, and K. Madhava Krishna. Mvracklay: Monocular multi-view layout estimation for warehouse racks and shelves. In *2022 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 1476–1482, 2022.

[15] S. Hutchinson, G.D. Hager, and P.I. Corke. A tutorial on visual servo control. *IEEE Transactions on Robotics and Automation*, 12(5):651–670, 1996.

[16] Pushkal Katara, Y V S Harish, Harit Pandya, Abhinav Gupta, Aadil Mehdi Sanchawala, Gourav Kumar, Brojeshwar Bhowmick, and Madhava Krishna K. Deepmpcvs: Deep model predictive control for visual servoing, 2021.

[17] Max Argus, Lukas Hermann, Jon Long, and Thomas Brox. Flowcontrol: Optical flow based visual servoing, 2020.

[18] Y V S Harish, Harit Pandya, Ayush Gaud, Shreya Terupally, Sai Shankar, and K. Madhava Krishna. Dfvs: Deep flow guided scene agnostic image based visual servoing, 2020.

[19] Zetao Yang, Xungao Zhong, Chunxiao Miao, Jing Zhao, and Xunyu Zhong. Optical-flow-based visual servoing for robotic moving control using closed-loop joints. In *2021 40th Chinese Control Conference (CCC)*, pages 2145–2149, 2021.

[20] Corinna Cortes and Vladimir N Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[21] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[22] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[23] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.

[24] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*. Springer, 2016.

[25] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*. Springer, 2014.

[26] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.

[27] Jason Ku, Melissa Mozifian, Jungwook Lee, Ali Harakeh, and Steven L Waslander. Joint 3d proposal generation and object detection from view aggregation. In *IROS*, 2018.

[28] Ming Liang, Bin Yang, Shenlong Wang, and Raquel Urtasun. Deep continuous fusion for multi-sensor 3d object detection. In *ECCV*, 2018.

[29] Samuel Schulter, Menghua Zhai, Nathan Jacobs, and Manmohan Chandraker. Learning to look around objects for top-view representations of outdoor scenes. In *ECCV*, 2018.

[30] Ziyan Wang, Buyu Liu, Samuel Schulter, and Manmohan Chandraker. A parametric top-view representation of complex road scenes. In *CVPR*, 2019.

[31] Kaustubh Mani, Swapnil Daga, Shubhika Garg, Sai Shankar, J. Krishna Murthy, and K. Madhava Krishna. Monolayout: Amodal layout estimation from a single image. *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2020.

[32] C. Mayershofer, D. M. Holm, B. Molter, and J. Fottner. Loco: Logistics objects in context. In *2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 612–617, 2020.

[33] NVIDIA Isaac SDK, 2019.

[34] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models, 2020.

[35] Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics, 2015.

[36] Prafulla Dhariwal and Alex Nichol. Diffusion models beat gans on image synthesis, 2021.

[37] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models, 2022.

[38] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents, 2022.

[39] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour, Burcu Karagol Ayan, S. Sara Mahdavi, Rapha Gontijo Lopes, Tim Salimans, Jonathan Ho, David J Fleet, and Mohammad Norouzi. Photorealistic text-to-image diffusion models with deep language understanding, 2022.

[40] Saurabh Saxena, Abhishek Kar, Mohammad Norouzi, and David J. Fleet. Monocular depth estimation using diffusion models, 2023.

[41] Bingxin Ke, Anton Obukhov, Shengyu Huang, Nando Metzger, Rodrigo Caye Daudt, and Konrad Schindler. Repurposing diffusion-based image generators for monocular depth estimation, 2023.

[42] Andreas Lugmayr, Martin Danelljan, Andres Romero, Fisher Yu, Radu Timofte, and Luc Van Gool. Repaint: Inpainting using denoising diffusion probabilistic models, 2022.

[43] Eyal Molad, Eliahu Horwitz, Dani Valevski, Alex Rav Acha, Yossi Matias, Yael Pritch, Yaniv Leviathan, and Yedid Hoshen. Dreamix: Video diffusion models are general video editors, 2023.

[44] Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion, 2023.

[45] Anurag Ajay, Yilun Du, Abhi Gupta, Joshua Tenenbaum, Tommi Jaakkola, and Pulkit Agrawal. Is conditional generative modeling all you need for decision-making?, 2023.

[46] Michael Janner, Yilun Du, Joshua B. Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis, 2022.

[47] Yanjie Ze, Gu Zhang, Kangning Zhang, Chenyuan Hu, Muhan Wang, and Huazhe Xu. 3d diffusion policy, 2024.

[48] Tim Brooks, Aleksander Holynski, and Alexei A. Efros. Instructpix2pix: Learning to follow image editing instructions, 2023.

[49] Bahjat Kawar, Shiran Zada, Oran Lang, Omer Tov, Huiwen Chang, Tali Dekel, Inbar Mosseri, and Michal Irani. Imagic: Text-based real image editing with diffusion models. In *Conference on Computer Vision and Pattern Recognition 2023*, 2023.

[50] Chenlin Meng, Yutong He, Yang Song, Jiaming Song, Jiajun Wu, Jun-Yan Zhu, and Stefano Ermon. Sdedit: Guided image synthesis and editing with stochastic differential equations, 2022.

[51] Francois Chaumette and Seth Hutchinson. *Visual servo control, part I: Basic approaches*, volume 13, pages 82–90. IEEE, 2006.

[52] Ezio Malis, François Chaumette, and Samuel Boudet. Visual servoing without explicit image jacobian: general 2d features. In *Proceedings 2001 IEEE/RSJ International Conference on Intelligent*

Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millennium (Cat. No. 01CH37180), volume 1, pages 298–303. IEEE, 2001.

[53] Ezio Malis, François Chaumette, and Samuel Boudet. Visual servoing using the self-calibration of a camera. In *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No. 99CH36288C)*, volume 3, pages 2182–2187. IEEE, 1999.

[54] Seth Hutchinson, Gregory D Hager, and Peter I Corke. A tutorial on visual servo control. *IEEE Transactions on Robotics and Automation*, 12(5):651–670, 1996.

[55] Yang Chen, Haibo Wang, Shuyang Yang, Zhongkai Cai, and Shengjin Tang. Deep learning-based visual servoing: A survey. *Robotics and Autonomous Systems*, 131:103606, 2020.

[56] Gonzalo Ferrer and François Chaumette. Adaptive image-based visual servo control under uncertainties. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5297–5302. IEEE, 2016.

[57] Lianru Liu, Zhen Ma, Guoqing Sun, Tao Zhang, and Jie Zhang. Real-time implementation of image-based visual servoing for uav autonomous landing. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 10167–10172. IEEE, 2019.

[58] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow, 2020.

[59] Philipp Fischer, Alexey Dosovitskiy, Eddy Ilg, Philip Häusser, Caner Hazırbaş, Vladimir Golkov, Patrick van der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks, 2015.

[60] Harshit K. Sankhla, M. Nomaan Qureshi, Shankara Narayanan V., Vedansh Mittal, Gunjan Gupta, Harit Pandya, and K. Madhava Krishna. Flow synthesis based visual servoing frameworks for monocular obstacle avoidance amidst high-rises, 2022.

[61] Gunjan Gupta, Vedansh Mittal, and K. Madhava Krishna. Dyngraspvs: Servoing aided grasping for dynamic environments. In *2023 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 1–8, 2023.

[62] Kevin Black, Mitsuhiko Nakamoto, Pranav Atreya, Homer Walke, Chelsea Finn, Aviral Kumar, and Sergey Levine. Zero-shot robotic manipulation with pretrained image-editing diffusion models, 2023.

[63] Mel Vecerik, Carl Doersch, Yi Yang, Todor Davchev, Yusuf Aytar, Guangyao Zhou, Raia Hadsell, Lourdes Agapito, and Jon Scholz. Robotap: Tracking arbitrary points for few-shot visual imitation, 2023.

[64] Chelsea Finn and Sergey Levine. Deep visual foresight for planning robot motion, 2017.

[65] Frederik Ebert, Chelsea Finn, Sudeep Dasari, Annie Xie, Alex Lee, and Sergey Levine. Visual foresight: Model-based deep reinforcement learning for vision-based robotic control, 2018.

[66] Ajay Sridhar, Dhruv Shah, Catherine Glossop, and Sergey Levine. NoMaD: Goal Masked Diffusion Policies for Navigation and Exploration. *arXiv pre-print*, 2023.

[67] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

[68] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems 27*. 2014.

[69] Kaichun Mo, Shilin Zhu, Angel X. Chang, Li Yi, Subarna Tripathi, Leonidas J. Guibas, and Hao Su. PartNet: A large-scale benchmark for fine-grained and hierarchical part-level 3D object understanding. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[70] Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Proceedings of Robotics: Science and Systems (RSS)*, 2023.

[71] Santhosh Kumar Ramakrishnan, Aaron Gokaslan, Erik Wijmans, Oleksandr Maksymets, Alexander Clegg, John M Turner, Eric Undersander, Wojciech Galuba, Andrew Westbury, Angel X Chang, Manolis Savva, Yili Zhao, and Dhruv Batra. Habitat-matterport 3d dataset (HM3d): 1000 large-scale 3d environments for embodied AI. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2021.