ABSTRACT

Title of Dissertation: MINIMAL PERCEPTION: ENABLING AUTONOMY ON RESOURCE-CONSTRAINED ROBOTS

Chahat Deep Singh
Doctor of Philosophy, 2023

Dissertation Directed by: Professor Yiannis Aloimonos
Department of Computer Science

Mobile robots are widely used and crucial in diverse fields due to their autonomous task performance. They enhance efficiency, and safety, and enable novel applications like precision agriculture, environmental monitoring, disaster management, and inspection. Perception plays a vital role in their autonomous behavior for environmental understanding and interaction. Perception in robots refers to their ability to gather, process, and interpret environmental data, enabling autonomous interactions. It facilitates navigation, object identification, and real-time reactions. By integrating perception, robots achieve onboard autonomy, operating without constant human intervention, even in remote or hazardous areas. This enhances adaptability and scalability.

This thesis explores the challenge of developing autonomous systems for smaller robots used in precise tasks like confined space inspections and robot pollination. These robots face limitations in real-time perception due to computing, power, and sensing constraints. To address this, we draw inspiration from small organisms such as insects and hummingbirds, known for

their sophisticated perception, navigation, and survival abilities despite their minimalistic sensory and neural systems. This research aims to provide insights into designing compact, efficient, and minimal perception systems for tiny autonomous robots.

Embracing this minimalism is paramount in unlocking the full potential of tiny robots and enhancing their perception systems. By streamlining and simplifying their design and functionality, these compact robots can maximize efficiency and overcome limitations imposed by size constraints. In this work, a *Minimal Perception framework* is proposed that enables onboard autonomy in resource-constrained robots at scales (as small as a credit card) that were not possible before. Minimal perception refers to a simplified, efficient, and *selective* approach from both hardware and software perspectives to gather and process sensory information. Adopting a task-centric perspective allows for further refinement of the minimalist perception framework for tiny robots. For instance, certain animals like jumping spiders, measuring just 1/2 inch in length, demonstrate minimal perception capabilities through sparse vision facilitated by multiple eyes, enabling them to efficiently perceive their surroundings and capture prey with remarkable agility.

This thesis introduces a cutting-edge exploration of the minimal perception framework, pushing the boundaries of robot autonomy to new heights. The contributions of this work can be summarized as follows:

- Utilizing minimal quantities such as uncertainty in optical flow (Ajna Chp 2) and its untapped potential to enable autonomous navigation, static and dynamic obstacle avoidance, and the ability to fly through unknown gaps.

- By utilizing the principles of interactive perception (Chp 3), the framework proposes novel object segmentation in cluttered environments eliminating the reliance on neural network

training for object recognition.

- Introducing a generative simulator called WorldGen (Chp 4) that has the power to generate countless cities and petabytes of high-quality annotated data, designed to minimize the demanding need for laborious 3D modeling and annotations, thus unlocking unprecedented possibilities for perception and autonomy tasks.

- Proposed a method to predict metric dense depth maps (Chp 5) in never-seen or out-of-domain environments by fusing information from a traditional RGB camera and a sparse 64-pixel depth sensor.

- The autonomous capabilities of the tiny robots are demonstrated on both aerial and ground robots: (a) autonomous car with a size smaller than a credit card ($70mm$), and (b) bee drone with a length of $120mm$, showcasing navigation abilities, depth perception in all four main directions, and effective avoidance of both static and dynamic obstacles. (Chp 6)

In conclusion, the integration of the minimal perception framework in tiny mobile robots heralds a new era of possibilities, signaling a paradigm shift in unlocking their perception and autonomy potential. This thesis would serve as a transformative milestone that will reshape the landscape of mobile robot autonomy, ushering in a future where tiny robots operate synergistically in swarms, revolutionizing fields such as exploration, disaster response, and distributed sensing.

MINIMAL PERCEPTION: ENABLING AUTONOMY ON RESOURCE-CONSTRAINED ROBOTS

by

Chahat Deep Singh

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2023

Advisory Committee:
- Dr. Yiannis Aloimonos, Chair/Advisor
- Dr. Cornelia Fermüller
- Dr. Guido de Croon
- Dr. Christopher Metzler
- Dr. Nitin J. Sanket
- Dr. Inderjit Chopra, Dean's Representative

To my family, friends, mentors

and

to the people who strive to do what they love

Acknowledgments

I am sincerely humbled and overwhelmed with gratitude as I pen down this section. The process of my Ph.D. has taken me on a thrilling ride of emotions, obstacles, accomplishments, and profound personal development. It has been a remarkable journey that could not have been completed without the unflinching support and contributions of several incredible individuals.

It all started during my Master's in Robotics in 2016 at the University of Maryland. I still remember the first day I met Prof. Yiannis Aloimonos – I was awestruck by his inspiring thoughts on perception and how both small and large animals/insects solve the same day-to-day tasks but in different ways. He asked me one fundamental question – *'What is the minimum information required to solve a given task?'* Little did I know that this question would lay down the foundation of this thesis. I will be perpetually grateful to him for giving me this opportunity to work in the Perception and Robotics Group (PRG). I am also forever grateful to Dr. Cornelia Fermüller for introducing me to the field of *neuromorphic* perception and mentoring me. Thank you for treating me as your child and making PRG feel like a home away from home. The amount of freedom I have gotten from you two for creative thinking is unfathomable and has led me to this research. Thank you for allowing me to mentor students on my own and teach courses while pursuing my research – valuable skills that I would not have developed otherwise. Undoubtedly, these were the best years of my academic life!

I am eternally indebted to Nitin J. Sanket, for his unwavering unconditional support and

mentoring, especially during my masters, which has touched the depths of my heart and forever transformed my life. This journey would be impossible without you! Thank you for being a better than best friend throughout this journey. Thank you for teaching me the art of research and fostering me to think outside the box. I will always cherish the road trips and the fun sessions in the lab and in the house! Thank you for all the photography lessons among countless other things. I have thoroughly enjoyed discussing math with you. Thank you for being there at every step of the way – more than I deserved.

PRG has been like an amazing family all these years. I am thankful to everyone in the group for limitless discussions and fun experiences. I would like to thank Levi Burner with whom I had an insane amount of random discussions, especially math. Chethan Mysore Parameshwara for introducing, teaching, and involving me in neuromorphic perception. Kanishka Ganguly for helping me with all my papers, especially hardware and Linux-related stuff. To *Loonix*, our *pet* cockroach who stayed with us at 3 am during drone flight experiments. It was scary when you flew faster than the drone. Thank you Snehesh Shreshta for taking the lead with us in setting up the aerial robotics lab. A big thanks to Xiaomin Lin, Jingxi Chen, Botao He, Konstantinos Zampogiannis, Francisco Barranco, Michael Maynord, Chinmaya Devraj, Matthew Evanusa, Peter Sutor, Behzad Sadrfaridpour.

I am always indebted to all the Master's students that have helped me in my research over the years – Prateek Arora, Ashwin V. Kuruttukulam, Abhinav Modi, Yashveer Jain, Kartik Madhira, Varun Asthana, Saumil Shah and Akash Gupta. And a huge thanks to the undergraduate students for teaching me valuable mentorship skills – Rishabh Singh, Riya Kumari and Rohan Uttamsingh.

A huge thanks to Guido de Croon and Davide Scaramuzza for teaching me valuable skills

in research; Inderjit Chopra for inviting me to teach an aerial robotics course in your department; Luca Carlone and Ashok Agarwala for your insights into academia; Christopher Metzler to introducing me to computational imaging. And finally President Darryll Pines for his continuous support in the RoboBeeHive project.

To mom – Harbir Kaur and dad – Rajinder Singh Kambo, thank you for your patience over the years. I know I cannot thank you enough for believing in me even when I did not. I have deeply embedded your qualities within me without even realizing it. Completing this journey—from an underachieving student to where I am now—feels like a real-life Cinderella story. Jasmeet Singh Kambo, I have no words for you. Firstly, thank you for forcing me to enjoy my undergraduate life with fun projects and not worrying about grades. Thank you for being a life mentor! Sakshi Singh, I guess there's another doctor in the family now. Thank you for taking all the pressure off my shoulders. To the entire *Certified Siblings* – Jasmeet Singh, Sakshi Singh, Aman Kaur, Sparsh Deep Singh, Ananta Malhotra, Hersh Deep Singh, Hershita Singh, Arsh Deep Kaur, Utsav Agarwal, and Tapasi Malhotra for the family trips and endless fun discussions every weekend. Special thanks to Arhaan Agarwal and Jaiveer *'Fateh'* Singh. A big thank you to Hersh for his insightful views on mentoring, academia, and random fun math discussions.

A huge thank you to Sunaina Prabhu and Kedar Gaitonde for being there as emotional support throughout. To our family in College Park (*Ghar ek Mandir*) – Priyal Gala, Anoorag Sunkari, Vinayak Bendale, Ankita Tondwalkar, Prateek Arora, Harshvardhan Uppaluru, Shankar Ramesh, Pranay Kanagat, Devyani Gera, Kunal Mehta, Ishmeet Singh, Aprit Agarwal, Meghavi Prashnani, Nakul Garg, Pooja Guhan, Mrunal Dhaygude, and Aakriti Agrawal, thank you for your support and endless *Bakchodi*. A big thank you for my unconditional pet dogs – Bansi, Stella, and Lilly.

I would like to express my deepest gratitude to Pranshu Jhamb, Tapan Khattar, and Niharika Singh for their unwavering support, even during times when I couldn't be there. I offer my heartfelt apologies for not being able to attend your respective weddings. Finally, I extend my heartfelt gratitude to Naitri Rajyaguru for standing by my side during the last leg of this remarkable journey. Your support has been genuinely priceless and irreplaceable!

I extend my heartfelt appreciation to Tom Ventsias and Maria Herd for their invaluable assistance in promoting my research throughout the years. I am deeply grateful to BBC Earth, Voice of America, Maryland Today, and IEEE Spectrum for featuring my research. A special thank you goes to Indian Creek Elementary School for giving me the opportunity to teach the findings of my research to third-grade students. I am grateful to Ivan Pensiky and Kimberly Edwards for their support in the labs. Ania Picard, your unwavering support has meant the world to me. I would also like to express my gratitude to Janice Perrone, Tom Hurst, and Sharron Mcelroy for their patience and assistance with logistics.

I am immensely thankful to Vikram Hrishikeshavan and Derrick Yeo from the aerospace department for their invaluable help with drone hardware. I also want to express my profound gratitude to the Department of Computer Science, UMIACS, and the Maryland Robotics Center. Lastly, I extend my thanks to the Wikimedia Foundation for providing a free source of education.

I wish to express my sincere appreciation for the generous financial support received from the Office of Naval Research (ONR), Brin Family Foundation, Northrop Grumman Corporation, NVIDIA, National Science Foundation (NSF), Intel, Dean's Fellowship, Ann G. Wylie Fellowship, and the Future Faculty Fellowship.

Additionally, I am immensely grateful to the remarkable open-source platforms, including Linux, TensorFlow, ArduPilot, Raspberry Pi, NVIDIA Jetson, and PX4. Without their invaluable

contributions, this thesis would not have been achievable.

Remembering everyone is an impossible feat, and from the depths of my heart, I humbly apologize to those I may have inadvertently missed.

As I close this chapter and embark on new adventures, I carry with me the memories, lessons, and relationships forged along the way. May this acknowledgment serve as a token of my deepest appreciation and as a reminder of the indelible impact each and every one of you has had on my life. Thank you from the bottom of my heart.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

Nature has spent 3.8 billion years on research and development in genetic evolution. They have evolved over the years based on their daily operations, habitat, and surrounding. The evolution in nature has been *purposive* rather than *generic*. This evolution is largely driven by their perceptual behaviors based on their needs and environment. Over the years, these systems have learned to solve specific tasks very efficiently. These parsimonious systems or living beings carry a blueprint to develop the next generation of robots. The key to parsimony is using a minimal amount of information/cues or sensing modalities for an efficient competition of goals. In stark contrast, we have been developing robots and AI frameworks for merely 50 years, spending the most time developing independent modules. I draw inspiration from nature to formulate robot autonomy frameworks using only onboard sensing and computation at scales that were never thought possible before. The solution to robot autonomy lies at the intersection of AI, computer vision, computational imaging and robotics – resulting in *parsimonious* robots. *Parsimony* refers to thriving in a resource-constrained environment. Even with drastically varied power and sensor constraints, bees and birds are often able to perform similar tasks. To enable autonomy at such scales, my research focuses on the *robustness*, *unification* and *generalizibility* of AI frameworks for robots. *Robustness* refers to inferring prediction in the presence of noise in the input data. *Unification* refers to the formulation of a single mathematical framework that

enables solving different robotics problems. *Generalizibility* refers to the ability of an AI to work across various environmental conditions.

Inspired by nature, this thesis deals with a minimal perception framework for robots to develop the next generation of tiny, efficient, effective, and *purposive* robots with onboard autonomy. It introduces alternative methodologies to depth sensors that are essential for robot autonomy, especially navigation and obstacle avoidance.

An outline of the thesis is given below:

- Chapter 2 proposes the general formulation of uncertainty principles in optical flow and their unconventional uses in robot navigation.

- Chapter 3 utilizes the uncertainty in optical flow along with the principles of interactive perception to segment the never-seen objects in a cluttered environment by repeated nudging.

- Chapter 4 introduces a generative simulator to automatically create petabytes of high-quality annotated data of real-world cities (digital twins) without any need for manual efforts in 3D modeling by computer vision and robotics applications.

- Chapter 5 presents a method to predict metric dense depth maps by utilizing only a traditional RGB sensor with a 64-pixel super sparse depth sensor. We demonstrate its ability to navigate in unknown environments on a credit card-sized robot.

- Chapter 6 serves as the culmination of the minimal perception framework within the context of robot autonomy applications, encapsulating key findings and insights, and offering valuable reflections for future directions in this field.

1.1 Resource-constraint autonomy

Robot autonomy within a resource-constrained environment is a complex and challenging task that requires intricate strategies for optimal functionality. The basic premise revolves around designing robotic systems capable of performing tasks autonomously despite limitations in computational resources, energy supplies, or sensor capabilities. This is of particular importance in scenarios such as aerial robotics, deep-sea exploration, or extraterrestrial missions, where resource management becomes critical. Algorithms, such as those based on reinforcement learning or genetic algorithms, are often used to optimize resource allocation dynamically. These algorithms are designed to make trade-offs between resources used and the quality of the task performed, thus maximizing efficiency. Sensor fusion techniques are also crucial in managing limited sensor capabilities, merging data from multiple sources to improve understanding and accuracy. Moreover, power management strategies, including sleep-wake cycles and variable processing speed, are implemented to minimize energy consumption. Software and hardware are co-designed for these systems to leverage the unique properties of specific hardware for better resource management. As such, resource-constrained robot autonomy requires a multi-faceted approach integrating planning, learning, perception, and decision-making to facilitate successful operation.

From a perception point of view, resource-constrained robot autonomy presents intriguing challenges and necessitates innovative solutions. The perception system of an autonomous robot, including cameras, lidar, sonar, and other sensors, serves as its *'eyes'* but in a resource-constrained environment, the capabilities of these sensors may be limited. Therefore, advanced methods such as sensor fusion become crucial, which integrate data from different

sensor types to form a more comprehensive understanding of the environment and reduce perceptual uncertainty. Deep learning and computer vision techniques are also used to extract relevant features from the sensory data and to recognize objects and patterns, yet they must be implemented efficiently due to limited computational resources. Furthermore, perception-based strategies must account for energy consumption, as continuous data acquisition and processing can be power-intensive. As such, low-power modes and selective perception, where only relevant data are actively processed, can be effective strategies. Hence, in resource-constrained robot autonomy, the perception system must balance between detail and depth of environmental understanding, computational demand, and energy efficiency to ensure reliable operation.

Although, these computationally expensive perception algorithms can be outsourced using a cloud computer or a companion computer over networking. So the question the first question that comes to mind is – *'Why do we need onboard autonomy?'*. Autonomous systems that rely on either wirelessly connected companion computers in the vicinity or cloud computing are susceptible to deployment in the wild. Such systems tend to fail in GPS-denied environments as well are prone to latency issues. Onboard robot autonomy leads to secure systems that reduce the possibilities of hacking and various security threat as well as make the robots more robust. Although, we have some highly capable autonomous robots with onboard computing that are relatively large in size (more than 300mm) – both aerial and ground robots. So, *'Why do we need small robots?'* These robots are safe, and agile and can be deployed as swarms. These swarms are highly scalable and can be effectively produced at much cheaper costs. Furthermore, these autonomous swarms enable the robots to inspect confined and dangerous areas that are time constrained such as thermonuclear power plants. It is a well-known fact that robot autonomy is substantially affected by the speed and size of the memory, sensor type, and quality as well as the

power required. This directly affects the robotic system's size, area, and weight.

1.2 Learning From Nature

The field of biomimetics, or biologically inspired engineering, offers valuable insights into designing and creating robots based on the study of nature - animals, birds, insects, and even plant life. These natural entities exhibit unique abilities honed by millions of years of evolution, providing excellent models for robotic systems. For instance, the agility of a cheetah can inform the development of robots with enhanced locomotion capabilities. Similarly, the echolocation method used by bats can inspire the design of robotic sensing systems that operate effectively in low-light conditions. The collective behavior of ants and bees provides concepts for swarm robotics, where multiple robots work together to perform complex tasks more efficiently than a single robot could. Studying bird flight can contribute to advancements in aerial drone technology, providing insights into energy efficiency and aerodynamics. In the case of insects like spiders, their ability to create intricate web structures could influence the design of construction or 3D printing robots. Thus, understanding nature and its mechanisms opens up a wide array of possibilities for designing innovative, efficient, and adaptable robotic systems.

From a perception perspective, biomimetics plays a vital role in developing advanced robotic systems inspired by nature. A central premise is how animals, birds, and insects perceive and interact with their environment, offering rich insights for creating efficient robotic perception systems. For instance, the complex visual processing system of a dragonfly, capable of detecting movement and depth with extraordinary precision, can inspire the development of sophisticated machine vision algorithms for robots. The sonar system of bats, which enables

navigation and hunting in the dark, provides a model for developing robust echo-based sensing mechanisms, especially useful for robots operating in low-visibility environments. Birds, with their ability to adjust their flight based on wind conditions, offer insights for developing adaptive perception and control systems in aerial drones. Similarly, the tactile sensing of rodents' whiskers could inform the design of touch-based perception for robots operating in dark or cluttered spaces. Swarm robotics often draw inspiration from ants and bees, which communicate and coordinate effectively to perceive their environment collectively and perform complex tasks. Thus, perception research in biomimetics is about deciphering and applying nature's sophisticated sensory systems to enhance robotic perception and interaction capabilities.

Thus, in order to make these systems autonomous, let us understand what we can learn from nature to build the next generation of onboard tiny autonomous robots. What does it take from the currently existing autonomous systems to downsize by a multitude of factors while maintaining or eventually enhancing autonomous capabilities? One way is to look at nature and living beings and observe their behaviors. Let us look at Fig. 1.1. It indicates the perception capabilities which are largely driven by their perception systems as compared to their body lengths. It is important to note that perception capabilities monotonically increase with body size i.e. with the increase in body length, their perception systems become matured with some exceptions in a few living beings. These exceptions include jumping spiders, cuttlefish, and various species of frogs. Jumping spiders have a very sparse low-resolution vision system that enables them to process fast-moving objects or prey and quickly react to hunt them. Whereas cuttlefish and different species of frogs and other beings have developed their visual systems by modifying the aperture shapes. For example, a cuttlefish has a 'W'-shaped aperture while there are species of frogs that have vertical or horizontal openings. Also, note that the blue and green bubbles have real-world

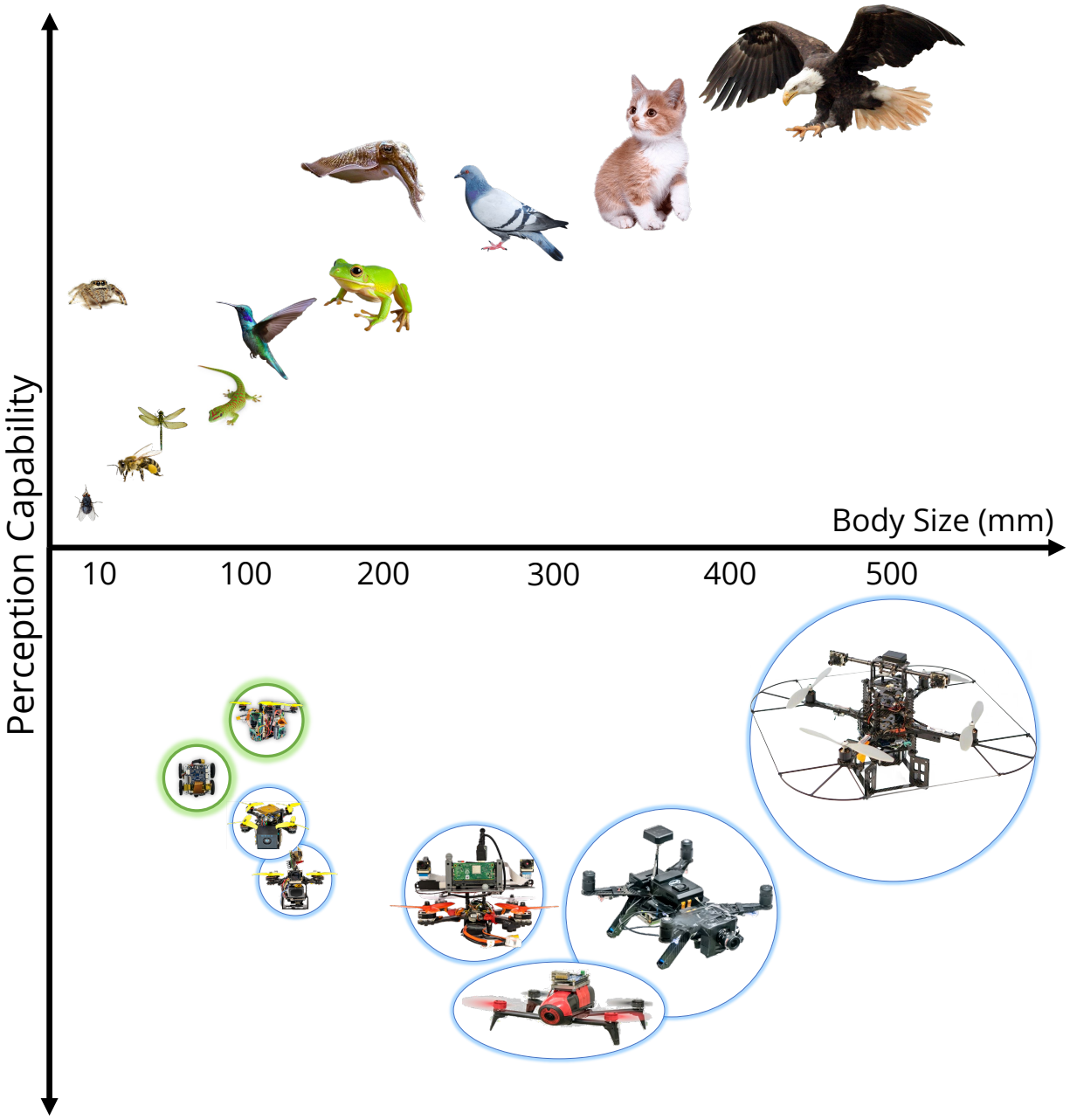

Figure 1.1: A qualitative comparison of living beings and robots in terms of perceptual capabilities with respect to their scaled body length. *Note that cat and eagle sizes are not to scale.*

tiny robots with onboard autonomy. Before this work, there were robots that existed that are as small as 120mm that are able to perform autonomous tasks such as navigation, static and dynamic obstacle avoidance as well as flying through unknown-shaped gaps [3]. These robots

are represented in a blue bubble. This work enables us to downsize and boost the autonomy performance even further at robots that are as small as a credit card (less than 3 inches in length). These robots are presented with a green bubble in Fig. 1.1. In the following sections, we will learn how bio-inspired solutions will boost resource-constraint autonomy in mobile robots.

1.3 Frugal AI

Nature's grandest architecture is built upon eons of evolution, an epitome of minimalism refined over time. Our approach to building robots should mirror this ethos - evolving complexity through simplicity, function through form, and achieving great feats not through excess but efficiency. One of the most influential theories highlighting the elegance of simplicity in language structure is Noam Chomsky's Minimalist Program [4]. Chomsky, a renowned linguist, cognitive scientist, and philosopher, proposed the Minimalist Program as a radical rethinking of syntactic theory. Its underlying principle is the idea that nature, including human language, operates in the simplest and most efficient way possible. The core of Chomsky's Minimalist Program rests on the assumption that sentences are built from a basic lexical inventory through a series of binary merges. This process minimizes computational complexity by reusing the same operation for structuring sentences, enabling an infinite array of expressions from a finite set of elements. Another significant facet of the Minimalist Program is the notion of 'economy.' Chomsky theorized that linguistic expressions follow the principle of economy, ensuring the most resource-efficient outcomes. This mirrors the concept of minimalism where 'less is more.'

Based on such theories, a new field has emerged – Frugal AI which refers to the development and deployment of artificial intelligence (AI) systems that operate effectively with

minimal resources, particularly in terms of computational power, energy consumption, and data requirements. The concept aims to create AI models that maintain high efficiency and robust performance despite these constraints. This is particularly relevant in scenarios where the availability of resources is limited, such as remote areas, edge devices, or low-income regions. Frugal AI can also contribute to sustainable AI practices, reducing the environmental impact associated with data centers and large-scale computing.

The development of frugal AI poses significant research challenges, necessitating the exploration of methods that allow model compression, efficient learning, and effective inference. Techniques such as quantization, pruning, and knowledge distillation are often used to compress deep learning models without substantial loss of accuracy. Transfer learning and few-shot learning strategies are used to enable models to learn effectively from small data sets. On-device AI and federated learning can be utilized for inference in resource-constrained environments while also preserving privacy. As such, the research and implementation of frugal AI embody a shift towards more accessible, sustainable, and decentralized AI practices, making advanced technologies more equitable and less resource-intensive.

While many forms of perception in the animal kingdom are astoundingly complex, they also often exhibit a remarkable sense of frugality. This 'minimal perception' underscores how living beings optimize their perceptual capacities in resource-constrained conditions. In the following section, we will engage in a detailed exploration of how diverse organisms employ distinct strategies to address identical challenges.

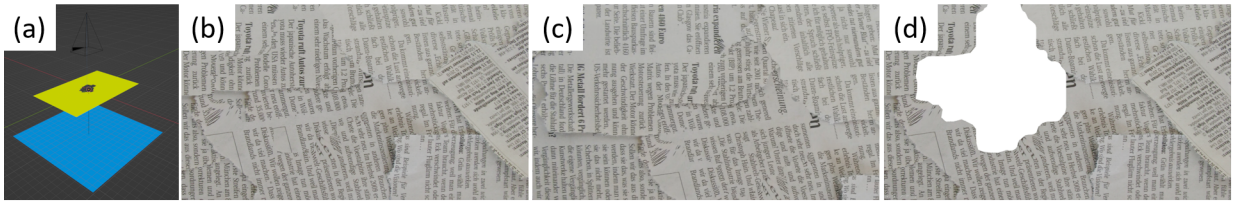

Figure 1.2: Segmentation of the gap with similar texture on the foreground and background elements.

1.4 Active Vision

Active vision [3,5–7] is a concept in computer vision and robotics that describes a system's ability to control the focus of its attention rather than passively analyzing an entire scene. This is often achieved through physical motion or manipulation of the sensor or environment. It reflects the concept of "looking around" to gather information, mimicking the way humans and animals actively observe their surroundings. Active vision systems can also dynamically modify their field of view or adjust parameters such as focus and aperture to capture the most pertinent information. This allows for more efficient data collection and processing since it enables the system to concentrate resources on the most relevant aspects of the visual scene.

Figure 1.2 illustrates a scene comprising a foreground and a background. In Figure 1.2a, the side view setup is presented, where the yellow region represents the foreground with a gap or hole, while the blue region represents the background. Notably, both foreground and background elements possess identical textures. Observing the scene from a single view, as depicted in Figure 1.2b, it is unfeasible to determine the exact location of the gap. However, by combining Figures 1.2b-c, it becomes possible to calculate the optical flow, enabling the estimation of ordinal depth. Consequently, the gap in the image can be identified, as demonstrated in Figure 1.2d.

In real-world applications, active vision strategies could significantly enhance the

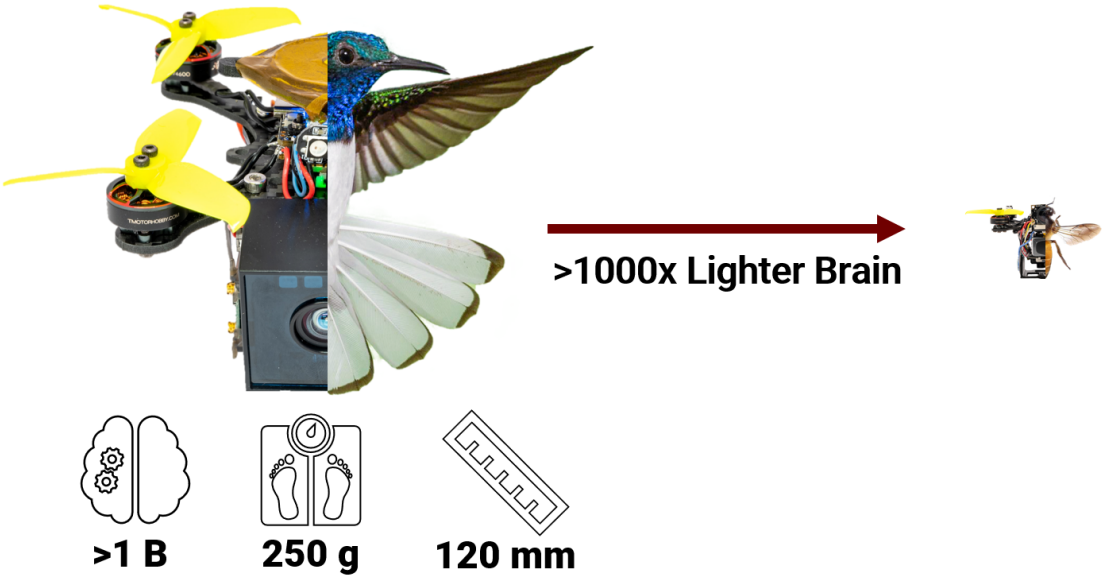

Figure 1.3: A comparison between a honey bee and a hummingbird.

capabilities of automated systems, such as autonomous vehicles, industrial robots, and surveillance systems. For instance, an autonomous vehicle equipped with an active vision system can adjust its sensors to focus on specific areas of interest like pedestrians, road signs, or other vehicles. Similarly, in surveillance applications, an active vision system can concentrate on unusual movements or behavior, making the overall system more effective and efficient. Thus, active vision forms a critical part of advanced AI systems, enabling them to interact more effectively with their environments.

Birds and bees – two different species (See Fig. 1.3 with very different resources in terms of sensing quality, number of neurons, weight, power, etc., they solve the same problem of flying through never seen unknown unstructured gaps but in a very different manner. While birds seamlessly traverse through these gaps, due to lower computation and sensing quality in bees, they tend to utilize active vision techniques. Bees wander around the gaps and observe the gaps from various positions and orientations in order to estimate the position and relative size (to the

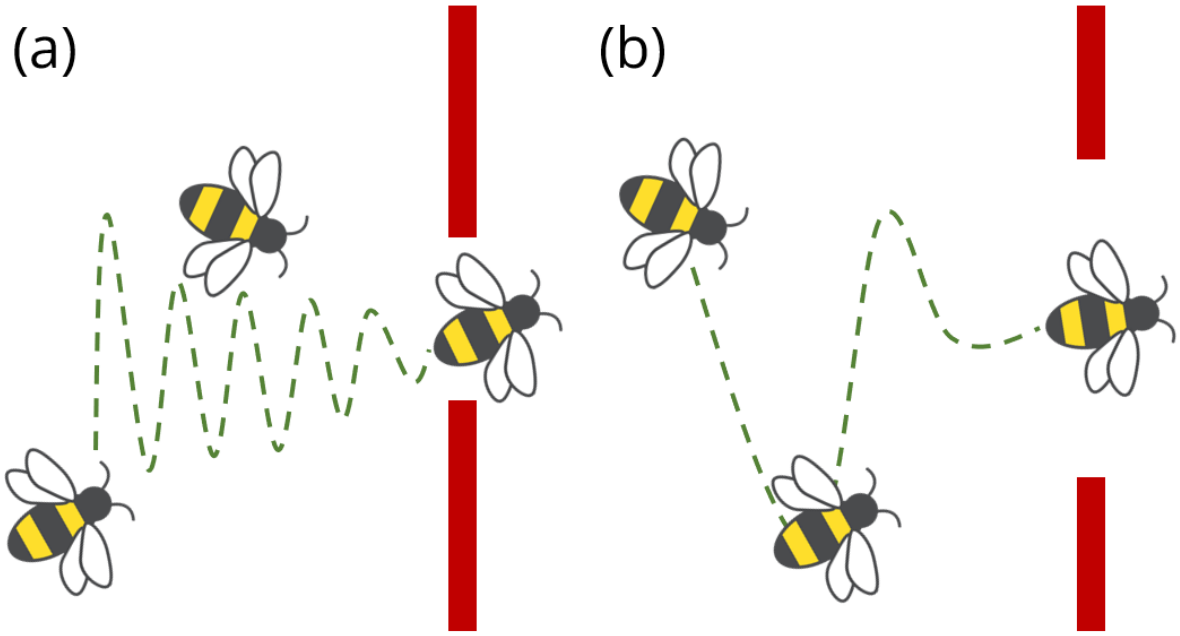

Figure 1.4: Bee Peering

background) of the gap. It is also important to note that bees do not estimate the size of the gaps but they estimate the size of the gap with respect to their body lengths. This effect the amount of *peering* the bees have to actively perform in order to have an ordinal depth perception of the gap. Fig. 1.4 demonstrates the amount of movement required for the bee for different sizes of gaps. This image has been adapted from [8].

GapFlyt [9] studies these behaviors and introduced TS^{2}P where they obtain and stack optical flow [10] from different views in order to estimate the ordinal depth. Fig. 2.5 shows the active strategy used in GapFlyt [9]. However, a significant limitation of this technique is the absence of mathematical assurances for successful gap traversal. The drone lacks the ability to estimate the metric depth of the gap or determine whether it has successfully traversed through it. However, this challenge has been addressed by the TinyDepth approach, which is discussed in detail in Section 1.9.

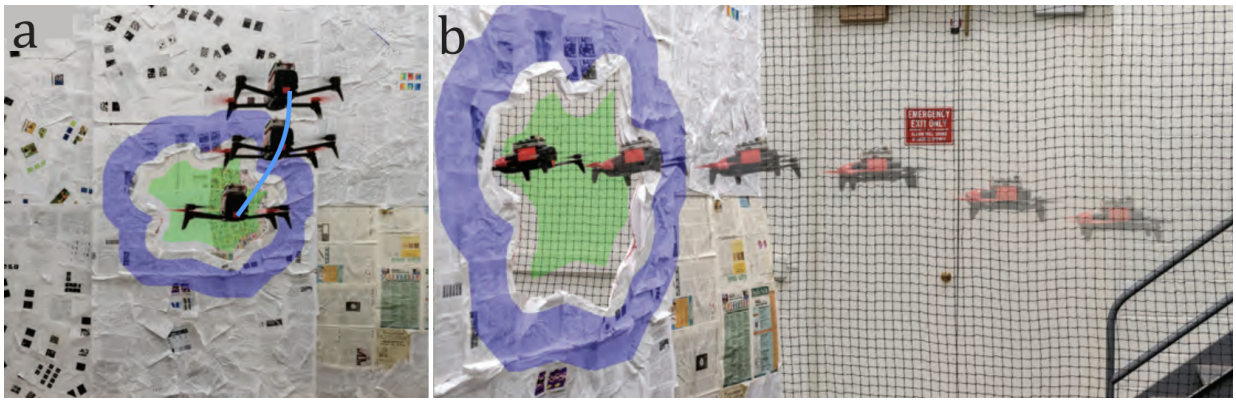

Figure 1.5: Flying Through Unknown Gaps. (a) Active strategy and (b) Visual Servoing

1.5 Principles of Minimal Perception

Minimal perception is defined by a living being's ability to extract maximal information from their environment using minimal sensory data. It reflects an organism's capacity to maintain functional performance while minimizing the cognitive and energy resources required for perception. This economization of resources is not only essential for survival but also a testament to nature's ingenuity.

Fig. 1.6 shows the classification of the minimal perception framework. The notion of *Minimal Perception* can be conceptualized at different levels – from cognitive to the sensor. This work deals with three different categories of minimalism in perception:

1. Minimalism in Information: What is the minimum information by the robot required to complete a given task? This can be carried forward by utilizing minimal quantities such as uncertainty of optical flow (Chapter 2) or by active perception (observing from multiple views to learn the structure of the scene rather than the texture. See Fig. 1.7).

2. Minimalism in Sensing Modality: What is the minimal choice of sensor required to a given size, area, weight, and power constraint to solve the tasks at hand?

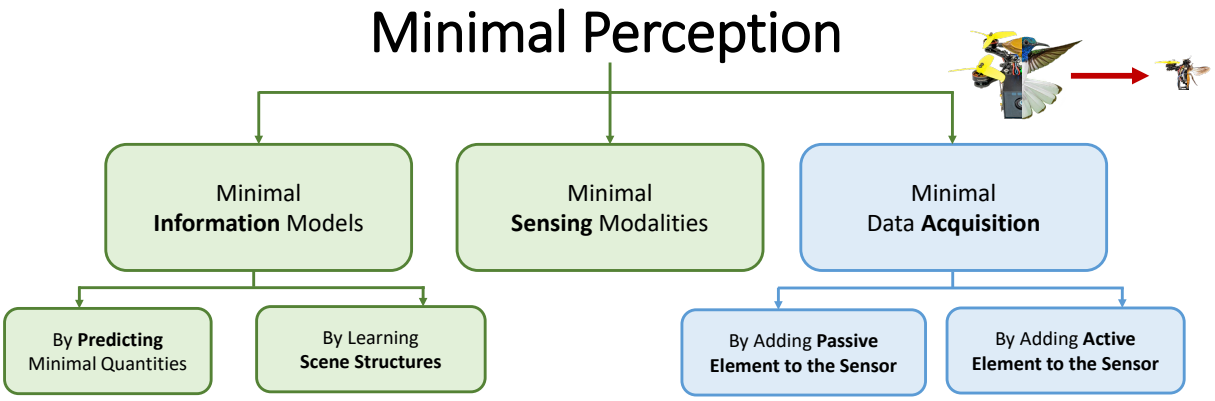

Figure 1.6: Minimal Perception Framework. Green parts are presented in this thesis and the blue parts are on going work that are presented in the future work.

3. Minimalism in data acquisition: By adding an active or passive element in front of the sensor, can we extract the minimal information from the environment that is required to solve a set of tasks?

The underlying principles of minimal perception are rooted in the goal of extracting essential information while optimizing resource utilization in autonomous systems. The principles governing minimal perception can be defined as follows:

1. Selective Information Extraction: Minimal perception involves the selective extraction of

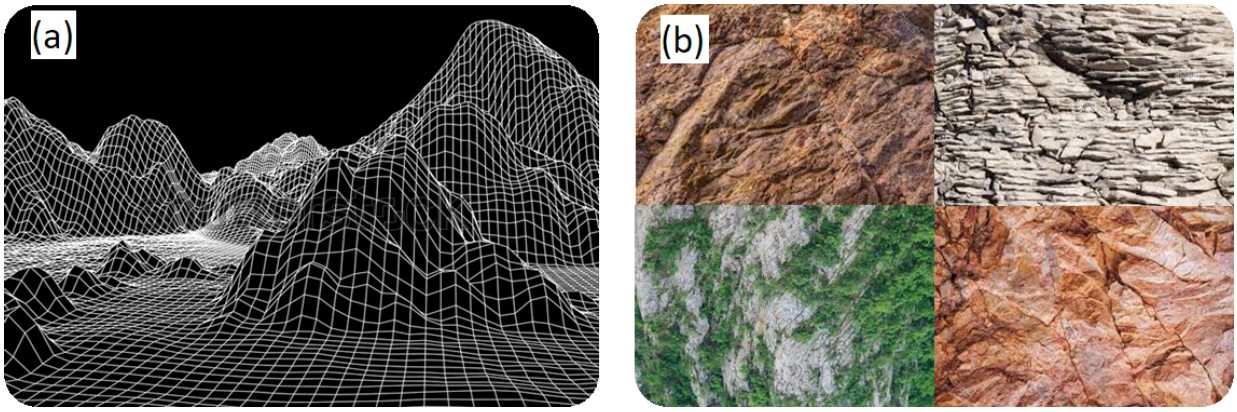

Figure 1.7: Learning to estimate the structure of the unknown scene (a) by observing it from multiple views can reduce the neural network model size in the estimation of the structures like mountains with various textures (b).

pertinent information from the environment. This principle focuses on identifying and prioritizing relevant data while disregarding non-essential or redundant information. By employing techniques such as feature selection, saliency analysis, or attention mechanisms, minimal perception aims to minimize the computational burden associated with processing large amounts of data.

2. Minimal Prior Knowledge: *'What is the information $\mathcal{I}$ required to solve $\mathbb{N}$ set of tasks $\mathcal{T}_{\mathbb{N}}$ in a given amount of time?'* This question addresses the possibility of solving a given task with absolutely minimal prior knowledge or information. Active and interactive Perception [11–13] strategies play a key role in solving these tasks with minimal prior knowledge.

3. Adaptive Sensing: Minimal perception incorporates adaptive sensing strategies to optimize data collection based on the specific context and task requirements. Adaptive sensing techniques dynamically adjust sensor parameters such as sensing modality, changes in aperture shapes, etc. to effectively capture the necessary information. By adopting the sensing process to the prevailing conditions, minimal perception reduces resource consumption and maximizes the efficiency of data acquisition.

4. Attention Mechanism: Attention mechanisms play a crucial role in minimal perception by directing computational resources toward salient stimuli. Inspired by nature's visual attention, these mechanisms allocate processing power and sensory focus to the most relevant parts of the input data. By selectively attending to significant features or regions, minimal perception optimizes computational efficiency and facilitates real-time responsiveness in resource-constrained systems. In Chapter 5, we exemplify this principle by showcasing the robot's capability to estimate dense depth in all spatial directions.

However, it selectively focuses its computational resources solely on the direction associated with the highest potential risk for effective obstacle avoidance and navigation during tasks.

5. Hardware-Software Optimization: Hardware-software co-design for mobile robots plays a vital role in maximizing the capabilities and performance of these autonomous systems. It involves the seamless integration and optimization of hardware components and software algorithms specifically tailored to the requirements of mobile robotic applications. The co-design process aims to strike a balance between computational efficiency, power consumption, real-time responsiveness, and physical constraints to enable mobile robots to navigate their environments, perform complex tasks, interact with humans, and adapt to changing conditions.

1.6 Predicting Minimal Quantities Using Uncertainty Principles

The widely successful classical theory of visual perception utilizes a single image that is tailor-made for static scenes. However, this theory is destined to fail on robots due to the dynamic nature of real-world environments, limiting robot autonomy. Combing motion or *Temporal Information* (TI) along with the sensor characteristics enabled us to unlock hidden potentials of perception that were not possible before. The use of TI empowers us to solve common robotics problems such as navigation and segmentation without any need for depth (or range) sensors. To accomplish such tasks, it is crucial for the robot to learn the geometry of the environment and the physics of the robot (*how things move*), rather than learning only the scene characteristics (*how the environment looks like*).

The presence of motion or temporal information (TI) introduces uncertainty in network predictions. Roboticists and computer vision scientists currently exploit this uncertainty to improve predictions. However, these uncertainties contain untapped concealed information with significant potential for addressing various robotics problems. Aleatoric uncertainty specifically characterizes the inherent bias in data collection by sensors, such as cameras' limited ability to perceive obstructed objects. To demonstrate the potential of these additional cues, aleatoric uncertainty prediction was exclusively employed on TI, specifically optical flow, for diverse robotics applications. The primary advantage of relying solely on uncertainty, rather than traditional predictions, is a substantial reduction in computational costs ranging from 10 to 100 times. Works such as Ajna (Chapter 2) and NudgeSeg [14] (Chapter 3)) showcase real-time robotics tasks utilizing uncertainty, including navigation static and dynamic obstacle avoidance, traversing unknown gaps, and segmentation tasks. Significant uncertainties were observed in areas where optical flow presented challenges, such as occlusions and motion blur, and this information was effectively utilized to detect obstacles within the scene. Additionally, a novel class of sensors known as neuromorphic sensors or event cameras, capable of extracting TI at the sensor level itself, can be employed to further enhance robot efficiency.

Quoting the philosopher Socrates, who famously said, "The only true wisdom is in knowing you know nothing," it is crucial to recognize when an agent lacks certainty, just as it is important to evaluate the accuracy of predictions. In the field of Robotics, we often rely blindly on neural network predictions for quantities such as Depth, Optical Flow, and Surface Normal, without quantifying the reliability of these predictions. This reliance has prompted Roboticists to acknowledge the need for incorporating uncertainties, leading to the adoption of the gold-standard approach in robotics: combining multiple measurements using Bayesian

formulations and propagating distribution statistics.

While uncertainties prove valuable for merging multiple measurements, we believe that their potential in robotics remains underexploited. This is mainly because uncertainties offer contextual information beyond their combined capabilities. Before delving into specific examples, let us introduce two common types of uncertainties: Aleatoric, also known as observational data uncertainty, and Epistemic, which pertains to model uncertainty. Aleatoric uncertainty captures the inherent bias in data collection by a sensor, while epistemic uncertainty captures the inherent bias arising from the scenarios used to train the model. For example, aleatoric uncertainty would be high in transparent or dark regions when using RGB-D data, whereas a network trained indoors would exhibit high epistemic uncertainty when tested on outdoor data.

The contextual information provided by an epistemic uncertainty model is the need for additional data samples to improve accuracy for a particular sample. This information is valuable for determining if the agent is operating in an "out of domain" situation and whether online learning is necessary to achieve desirable performance. On the other hand, a more careful examination of the contextual information from aleatoric uncertainty reveals intriguing insights about the scene based on sensor characteristics. For instance, cameras cannot see through objects, so high aleatoric uncertainty at the depth boundaries of an object can serve as a powerful cue for various robotics tasks.

Estimating epistemic uncertainty requires variational inference and multiple runs of the neural network, making it impractical for real-time applications unless multiple neural network accelerators are employed. Conversely, aleatoric uncertainty is well-suited for real-time applications as it only requires a minor increase in the number of parameters and a single pass

of the network to predict uncertainty. In this study, the focus is on estimating heteroscedastic aleatoric uncertainty, which refers to the observational uncertainty specific to the input data.

We address the following questions in Chapter 2: *How can we estimate heteroscedastic aleatoric uncertainty in a neural network? What informational cues does it provide for various robotic tasks?* This work proposes a novel generalized formulation for heteroscedastic aleatoric uncertainty in neural networks.

1.7 Minimal Prior Knowledge – Interactive Perception

Perception and interaction constitute a synergistic pair that exhibits complementary properties in the field of robotics. Despite the inherent capabilities of most robots to engage in movement or body manipulation for the purpose of acquiring additional information, the utilization of this combined perception-interaction paradigm remains limited. Nature's creations, even at the most rudimentary biological level, exploit this active-interactive synergy to effectively address complex problem domains [15]. Consequently, the foundational principles of robotics have encompassed formal frameworks that capture the elegance of action-interaction-perception loops. By augmenting the computational requirements of a specific task through exploration and interaction, valuable information can be obtained in a manner that simplifies the underlying perception challenges.

In recent years, deep neural networks have made significant strides in object segmentation, effectively delineating objects within color and depth images for specific classes [11, 13, 16]. However, the performance of these networks relies heavily on the availability of training data encompassing diverse classes and objects. This limitation restricts their ability to generalize well

to previously unseen objects or zero-shot samples. Resource constraints further exacerbate the issue as robots can only be trained on a limited number of samples.

Furthermore, object segmentation based solely on image frames depends on the recognition and pattern-matching cues. To address these challenges more efficiently, our proposed approach leverages the active nature of robots and their capacity to interact with the environment. By engaging in interactions with objects, robots can induce additional geometric constraints to facilitate the segmentation of zero-shot samples. Our framework introduces a process where the robot repeatedly nudges or pokes at objects, leveraging the resulting motion cues to generate and refine segmentation masks at each step. The fundamental concept underlying our approach is that each rigid body exhibits a unique motion signature (optical flow) during each nudge. We exploit this characteristic to provide an initial estimation for the robot to learn about new objects through interaction, analogous to how infants acquire knowledge about their surroundings.

Since the method only relies on optical flow for segmenting these objects, it only utilizes a monocular monochrome camera. The method is evaluated on zero-shot samples (GrassMoss and Rocks) and the YCB dataset [17], and compared with state-of-the-art methods such as Mask-RCNN [18], PointRend [19], and 0-MMS [2]. It is observed that NudgeSeg outperforms previous state-of-the-art passive approaches on zero-shot samples. Chapter 3 will extensively explore the realm of interactive perception and provide an in-depth analysis of the NudgeSeg [14] framework.

1.8 Learning Structure via a Generative Simulator – Minimizing Annotations and Modeling

In the field of computer vision, a significant challenge involves transferring learned quantities such as 'depth' and 'optical flow' from one environment to another. Unlike living beings, current robotic systems lack the ability to infer depth in new surroundings and struggle with the cross-domain inference of acquired knowledge. For instance, a depth prediction model trained on outdoor data fails to accurately predict depth in indoor environments. In the research presented in Chapters 2, 3, 5, the approach addresses this issue by training models in a simulated environment and evaluating their performance on real-world scenes. Notably, this approach eliminates the need for fine-tuning the models with real-world data, which is contrary to the prevailing literature. This strategy effectively mitigates the problem of overfitting in neural networks, which commonly arises when fine-tuning is performed using testing data. Moreover, it improves the scalability of the networks, enabling their deployment across a variety of robot sizes and scales.

Neural network predictions are often constrained by simulated data generated using imperfect camera models that lack photorealism and accurate camera physics. Conversely, the collection and annotation of real-world data can be prohibitively expensive. In recent research, the open-source framework WorldGen [20] was employed to autonomously generate diverse structured and unstructured 3D photorealistic scenes, such as city views, object collections, and object fragmentation. This data generation process relies on existing open-source object models, world maps, and semantic information. WorldGen, a perception-centric generative simulator,

enables the modification of textures, object structures, motion, camera properties, and lens properties using photorealistic camera models, thereby reducing data bias in neural networks. Significant improvements in optical flow predictions were demonstrated using the WorldGen data. Furthermore, the capabilities of the WorldGen simulator were extended to include human motion data with various textures and structural characteristics. Remarkable advancements in human pose estimation on event data in the real world were achieved solely through training on the WorldGen simulation environment [21]. Additionally, simple and effective methods for learning to generate training data tailored to specific robotics applications were explored.

WorldGen serves as a high-level open-source Python library for generating an unlimited amount of synthetic data. This library provides a platform for generating visual data to simulate various scenarios, including self-driving cars, autonomous drones, object segmentation, active vision, motion segmentation, tracking, and computational photography. Its key contribution lies in the API that enables the construction of generative environments and streamlines the process of generating synthetic data, thereby lowering the difficulty barrier for researchers and practitioners. WorldGen is built around Blender™, a free and open-source 3D creation suite, allowing the generation of synthetic data such as city maps, collections of moving objects, and object fragmentation. The design of WorldGen emphasizes scalability and speed. Chapter 4 provides a comprehensive discussion of the various components and details employed in building WorldGen.

1.9 Minimal Sensing Modality

Minimal sensing modality in robots refers to the implementation of a simplified sensory system that enables the robot to perceive and *understand* the environment using a limited set of sensors. The goal is to design a sensing framework that optimizes resource utilization while still providing sufficient information for the robot to perform its intended tasks effectively. This approach involves carefully selecting a subset of sensors that capture key aspects of the robot's surroundings, such as proximity, orientation, or object detection, based on the specific requirements of the application. By minimizing the number and complexity of sensors, the robot can reduce cost, power consumption, and computational overhead while maintaining a practical level of situational awareness. The challenge lies in finding the right balance between sensor richness and system constraints to ensure reliable and efficient operation in real-world scenarios.

Accurate measurement of distances and depth cues is a fundamental requirement for autonomous robots to comprehend the geometric properties of a 3D scene. When it comes to navigation, agents heavily rely on depth maps to effectively traverse intricate and dynamic environments. However, conventional depth estimation algorithms, whether monocular or stereo-based, often involve computationally expensive operations or necessitate high-quality sensors. Consequently, their implementation becomes challenging in resource-constrained settings. To address this, leveraging motion cues like parallax, as observed in pigeons, can expedite depth computation. Previous endeavors have aimed to mitigate computational burdens by lowering the resolution or capitalizing on known environmental cues. Nonetheless, these approaches fall short in terms of accuracy for obstacle avoidance or fail to generalize to unfamiliar scenes when applied in real-world scenarios.

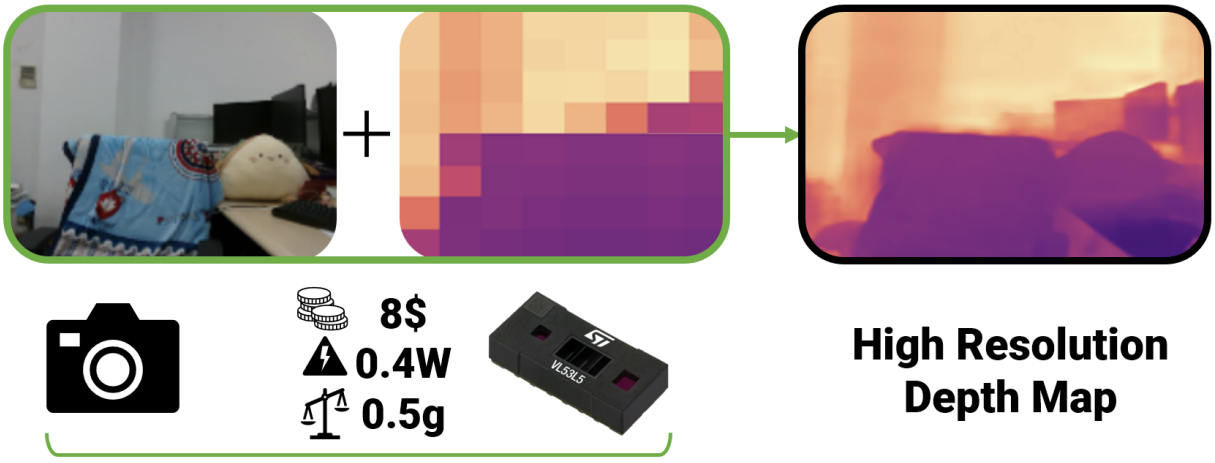

Figure 1.8: Combining RGB with a tiny sparse sensor leads to high-resolution depth maps.

Chapter 5 introduces TinyDepth, a compact neural network architecture that leverages a sparse depth sensor with low resolution and low power consumption (64 depth values). The proposed method achieves dense depth estimation by combining this sensor with a high-resolution monocular RGB camera. To enhance the training process, information from multiple viewpoints is exploited, incorporating motion parallax cues. This approach enables the model to generalize effectively to previously unseen or zero-shot scenes without the need for fine-tuning or retraining. Remarkably, the network achieves a processing rate of 4.3Hz on the Raspberry Pi CPU, providing accuracy comparable to larger networks while surpassing them significantly in terms of speed. Due to its lightweight computational demands and sensor requirements, this method is highly suitable for deployment on small-sized robots, including palm-sized and even hummingbird-sized aerial platforms.

Fig. 1.8 shows the conventional depth camera on the left and the contrasting setup of RGB and sparse depth sensor in order to estimate a high-resolution depth map. The work presented in this study is closely related in spirit to [22], emphasizing notable differences. In contrast, the model proposed here is much smaller, reaching sizes up to $126\times$ smaller as compared to

Intel Realsense D435i – an industrial defacto depth sensor. Moreover, our approach incorporates cues from multiple views and demonstrates the ability to generalize to zero-shot or unseen environments following simulation-based training. The efficacy of this approach is validated through real-world robotics experiments, explicitly focusing on navigation in complex static scenes involving both ground and aerial robots.

1.10 Minimal Data Acquisition

Minimal data acquisition in robot perception refers to the efficient and judicious collection of sensory information required for effective perception tasks in robotics. By carefully selecting and prioritizing the relevant data, robots can optimize their computational resources and improve real-time decision-making capabilities. This approach focuses on acquiring only the essential information needed to perceive the environment accurately while disregarding redundant or irrelevant data. Techniques such as active perception and sensor fusion play a crucial role in minimizing data acquisition. Active perception involves intelligent control strategies that guide the robot's sensors to actively gather information from specific areas of interest, maximizing the utility of acquired data. Sensor fusion combines data from multiple sensors to create a comprehensive and reliable representation of the environment. By adopting minimal data acquisition strategies, robots can enhance their perceptual capabilities while reducing computational complexity and achieving efficient and streamlined operations in various domains, including navigation, object recognition, and scene understanding.

Two ways to modify the data without computing are by adding either a passive element (such as custom apertures) or an active element (such as a rotating prism) in front of the

Figure 1.9: Illustration of a tiny robotic bee pollinating the flowers.

camera/sensor plane in order to filter out the data at a hardware level. This is discussed in the Chapter 7.

1.11 Applications of Minimal Perception

The future of tiny mobile robots and drones holds immense potential for various industrial, research, and consumer applications. With advancements in miniaturization and robotics technology, these miniature devices can perform intricate tasks in constrained environments with great precision and agility. The integration of artificial intelligence and minimal perception thinking is expected to play a crucial role in shaping the next generation of these robots. By employing minimal perception thinking, tiny mobile robots and drones can navigate complex environments, avoid obstacles, and carry out specific tasks with improved adaptability and autonomy. Furthermore, the integration of AI techniques, such as machine learning and computer vision, can enhance their perception capabilities, enabling them to interpret and respond to dynamic environments effectively. This convergence of minimal perception thinking and AI

Figure 1.10: Weight comparison between a 330ml of Pepsi can with the tiny autonomous car.

holds significant promise in unlocking the full potential of tiny mobile robots and drones across industries ranging from healthcare and agriculture to manufacturing and surveillance.

The thesis findings have led to significant practical applications in two areas. Firstly, the utilization of tiny robot bee drones equipped with comprehensive metric depth perception capabilities in all directions enables efficient pollination processes. Secondly, an onboard credit card mobile robot weighing a mere 100g demonstrates autonomous navigation capabilities. Visual representations of the weight comparison and the RoboBee can be found in Fig. 1.10 and Fig. 1.9, respectively. Subsequent chapters of this thesis will delve into the modeling of the necessary frameworks and showcase real-world robotic applications, particularly focusing on navigating in unfamiliar environments.

This page intentionally left blank.

Chapter 2: Generalized Deep Uncertainty For Parsimonious Robots

Robots are proactive entities functioning within fluctuating environments using imperfect sensors. These variable sensor readings often result in predictive inaccuracies and can prove untrustworthy. As a solution, robotic researchers employ fusion methods involving multiple observations. Recently, neural networks have emerged as leaders in terms of accuracy for perception-oriented predictions for robotic decision-making, although they frequently lack associated uncertainty measurements with the predictions. This chapter will introduce a mathematical model for determining heteroscedastic aleatoric uncertainty in any random distribution, without requiring preliminary data knowledge. This model doesn't make any assumptions about prediction labels and is impartial to network design.

A specific category of networks proposed in this work, known as *Ajna*, involves a minimal computational addition and necessitates only a slight alteration to the loss function during neural network training to capture uncertainty in predictions. This facilitates real-time operation even in robots under severe computational limitations, such as small drones. It will also explore the informative indicators found in the uncertainties of predicted values and their use in consolidating common robotics challenges. Specifically, this work proposes a strategy to avoid dynamic obstacles, traverse cluttered scenes, pass through unknown gaps, and segment an object pile. This is achieved not by computing depth but by utilizing the uncertainties of optical flow acquired

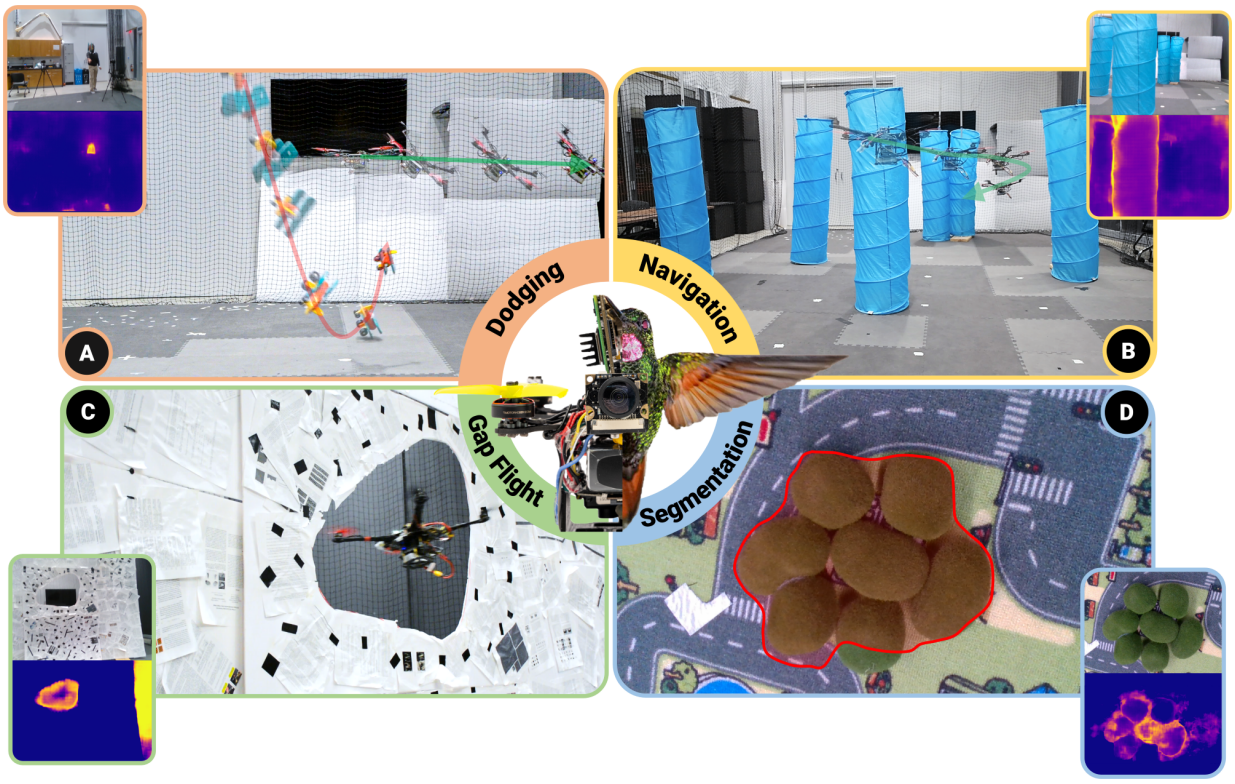

Figure 2.1: Unification of common robotics problems using the novel generalized heteroscedastic aleatoric uncertainty formulation for neural networks – *Ajna*. This chapter experimentally demonstrates the efficacy of using uncertainty for the following robotics tasks: (A) Dodging dynamic obstacles, (B) Navigating through cluttered scenes, (C) Flying through unknown gaps, and (D) Segmentation of unknown object piles. This chapter shows that such an algorithmic approach would enable autonomy at scales not thought possible before such as the drone the size of a hummingbird as shown in the center. *All the images in this chapter are best viewed in color and on a computer screen at 200% zoom.*

from a monocular camera with onboard sensing and computation.

This chapter will effectively assess and exhibit the proposed *Ajna* network on four aforementioned typical robotics and computer vision tasks, showing results comparable to methods that directly use depth.

2.1 Background

As an old saying goes – *"If knowledge is power, knowing what you don't know is wisdom"*. It is as important to know when the agent is unsure as much as the correctness of the prediction.

Especially in the case of neural network predictions, estimating the uncertainty associated with these predictions aid in taking better decisions rather than blindly relying on these predictions based on the assumption that they are correct. Roboticists have remarked on this observation and this led to the approach of combining multiple measurements using uncertainties which have become the gold-standard approach in robotics. Fundamentally, these measurements are combined using Bayesian formulations and propagating the distribution statistics.

Although uncertainties are very useful for combining multiple measurements, they are underutilized in robotics. This is due to the fact that uncertainties also provide contextual cues/information. Before this chapter provides examples of the previous statement, let us talk about two kinds of common uncertainties: Aleatoric or observational data uncertainty and Epistemic or model uncertainty.

The aleatoric uncertainty models the inherent bias in the way a sensor collects data and epistemic uncertainty models the inherent bias in the scenarios used to collect the training data. For e.g., the aleatoric uncertainty would be high for transparent or dark regions for RGB-D data and the epistemic uncertainty of a network trained indoors would be high when tested on outdoor data.

The contextual information that an epistemic uncertainty model provides is that the trained model requires more data to improve accuracy for the particular input sample. Such information is useful to know if one is operating 'out of domain' and if online learning is required for a

desirable operation. On the contrary, contextual information from Aleatoric uncertainty when studied more carefully is more intriguing as it helps unravel information about the scene based on the sensor characteristics. For e.g., cameras cannot see through objects, hence one would expect high aleatoric uncertainty at the object's depth boundaries which can act as a powerful cue for performing various robotics tasks.

Furthermore, from a pragmatic viewpoint, estimating epistemic uncertainty requires variational inference and multiple runs of the neural network leading it ineffectual for real-time applications unless multiple neural network accelerators are used. On the contrary, aleatoric uncertainty is highly suited for real-time applications since it requires a minor increase in the number of parameters and requires a single pass of the network to predict the uncertainty. In this work, we focus on estimating the heteroscedastic aleatoric uncertainty, i.e., observational uncertainty with respect to the input data.

In particular, this work proposes a generalized loss function formulation to estimate the heteroscedastic aleatoric uncertainty that can be used to model various probability distributions and relate it to the works in the past decade. This demonstrates that previous works are special cases of our generalized formulation. Furthermore, this work presents a theoretical analysis of what information/cues this uncertainty formulation provides for various prediction modalities. Finally, this work applies the predicted uncertainty to perform various robotic tasks and demonstrates the unification such a methodology can bring to various classes of robotics problems. The class of networks as *Ajna* which is named after the third eye of Lord Shiva from Hindu Mythology and refers to the eye of wisdom/consciousness/intuition since our networks can "see" (predict) where they might not work well. The uncertainty of predicted values is denoted as Υ as it represents the Greek letter for u standing for uncertainty and resembles the shrug emoji

🤷. We formally define the problem statement and a list of our contributions next.

The following questions are addressed: *How to estimate the heteroscedastic aleatoric uncertainty of a neural network? What informational cues does it provide for various robotic tasks?* Given an input x, label $\hat{y}$, and prediction $\tilde{y}$, the heteroscedastic aleatoric uncertainty Υ is predicted by minimizing the proposed generalized loss function. The loss function reduces to classical statistical properties of variance for common distributions such as Gaussian or Laplacian. Additionally, the uncertainty of optical flow is learned using this loss function, which is then applied to four example robotic tasks: (a) Navigating through a scene with static obstacles, (b) Dodging unknown dynamic obstacles, (c) Detecting and Flying through unknown shaped gaps, and (d) Segmenting an unknown object pile (See Fig. 4.2). A summary of the contributions in this chapter is provided below:

- A generalized heteroscedastic aleatoric uncertainty formulation for neural networks
- Analysis of informational cues provided by heteroscedastic aleatoric uncertainty for robotic tasks
- Extensive real-world experiments demonstrating how such uncertainty can be used for various robotic tasks
- Discussion of how uncertainty can act as a unifying parsimonious framework for various robotics applications

Uncertainties and error statistics have been widely utilized in robotics for several decades. In the subsequent sections, the works concerning the estimation of uncertainties in neural

networks and the applications of deep uncertainty in computer vision and robotics will be presented.

2.1.1 Estimating Uncertainties in Neural Networks

As previously mentioned, two types of uncertainties exist: (a) Aleatoric or observational uncertainty and (b) Epistemic or model uncertainty. Previous studies focused on estimating either Aleatoric or Epistemic uncertainty individually. Approaches such as [23–25] solely estimated Epistemic uncertainty by assuming a Gaussian prior distribution over weights. These models are known as Bayesian Neural Networks (BNN). Although the mathematical formulations of BNNs are straightforward, their inference requires complex computations as marginal distributions across all neurons need to be computed. Additionally, [26] introduced dropout variational inference to make Epistemic uncertainty estimation tractable through stochastic Monte Carlo dropout. In contrast, [27] presented a method specifically for Aleatoric uncertainty estimation, which was later combined with Epistemic uncertainty in [28] to obtain the concept of "total uncertainty." However, these methods were either computationally slow for robotic applications or lacked sufficient accuracy. To address this, [29] introduced Lightweight Probabilistic Deep Networks, which propagate uncertainties using assumed density filtering. An even faster variant was proposed, which directly predicts uncertainties only in the final layer. The approach was further extended in [30] to be agnostic to the network architecture and loss function. For a comprehensive overview of related works, please refer to [31], which provides a detailed summary of prior research.

2.1.2 Applications of Deep Uncertainty in Robotics and Computer Vision

In the field of robotics, the fusion of uncertainties and their statistical analysis has been widely employed to combine multiple measurements obtained from either a single sensor or multiple sensors. Recent research has witnessed a shift in focus towards incorporating uncertainty fusion techniques within neural networks, owing to the dominance of deep learning approaches in terms of accuracy metrics. For instance, TLIO [32] proposed a methodology that fuses multiple inertial measurements, leveraging predicted uncertainties in conjunction with an Extended Kalman Filter, to estimate odometry. KFNet [33] introduced a neural network-based fusion approach that combines measurement and process models, drawing inspiration from the classical Kalman Filter formulation [34], which was specifically applied to the problem of camera relocalization. In the pursuit of robust performance, IVOA [35] incorporates predicted uncertainties into the navigation stack. Moreover, a general framework for uncertainty estimation, encompassing both aleatoric and epistemic uncertainties, was presented in [30]. This framework was successfully applied to three tasks: (a) End-to-End Steering Angle Prediction, (b) Object Future Motion Prediction, and (c) Closed-Loop Control of a Quadrotor.

In the field of computer vision, the utilization of deep uncertainty predictions to enhance performance has gained significant attention in recent years. Various applications, including object detection, optical flow estimation, visual odometry, monocular depth estimation, stereo depth/disparity, and surface normals estimation, have leveraged uncertainties as a regularizer to improve robustness. To address noisy samples in 3D object detection using LiDAR data, Feng et al. [36] proposed a method that learns to ignore such samples. Several works, such as Lee et al. [37], Kang et al. [38], Ilg et al. [39], Gast et al. [29], and Li et al. [40], employ

either a Generative Adversarial model or an aleatoric uncertainty model to estimate uncertainties. These uncertainties are then used as regularizers to train optical flow models, leading to improved performance as observed empirically. In our work, we provide theoretical reasoning to explain this phenomenon, specifically attributing it to loss attenuation at optical flow discontinuities.

Methods presented by Yuan et al. [41], Bae et al. [42], Roessle et al. [43], and Bhatt et al. [44] focus on estimating dense depth from stereo or monocular views. They aim to improve accuracy at the boundaries by incorporating an uncertainty metric. Martin-Brualla et al. [45] utilize the same aleatoric uncertainty formulation to enhance volumetric color rendering in a NeRF (Neural Radiance Fields) model. Their approach involves rejecting dynamic objects based on uncertainty estimates. Eldesokey et al. [46] exploit uncertainty for self-supervised depth completion, achieving state-of-the-art performance. Similarly, Poggi et al. [47] utilize uncertainty obtained through image flipping to enhance monocular depth estimation results. Costante et al. [48] propose a method to estimate and incorporate total uncertainty into a deep visual odometry pipeline. Furthermore, Kawashima et al. [49] present an alternative approach for aleatoric uncertainty estimation, employing virtual residuals to address overfitting and demonstrating state-of-the-art results in age and monocular depth estimation. Alternatively, uncertainty has been indirectly learned as the probability of outlier/inlier in SFMLearner [50].

2.2 Method – Ajna

2.2.1 General Heteroscedastic Aleatoric Uncertainty Formulation

Consider an input x provided to a neural network $\mathbb{N}$, which has weights W. Let $\tilde{y}$ represent the estimated output of the neural network $\mathbb{N}$ (Eq. 2.1), while the ground truth prediction is

denoted by $\hat{y}$.

$$\tilde{y} = \mathbb{N}\left(x|W\right) \tag{2.1}$$

The objective is to learn weights W in order to optimize the following problem:

$$\underset{W,\Upsilon}{\operatorname{argmin}} f\left(\hat{y}, \tilde{y}\right) \text{ s.t. } \Upsilon = k\left(f\left(\hat{y}, \tilde{y}\right), x\right) \tag{2.2}$$

In this context, the symbol f represents a distance metric between the predicted value $\tilde{y}$ and the ground truth value $\hat{y}$. The symbol Υ corresponds to a monotone function k that depends on the heteroscedastic aleatoric uncertainty of the underlying probability distribution $p(x, \tilde{y}|W)$. This uncertainty is positively correlated with the expected error or risk. The correlation between two random variables X and Y is formally expressed as the Pearson correlation $\rho_{X,Y}$ in Equation 2.3, where the symbol $\mathbb{E}$ denotes the expectation operator.

$$\rho_{X,Y} = \frac{\mathbb{E}\left(XY\right) - \mathbb{E}\left(X\right)\mathbb{E}\left(Y\right)}{\sqrt{\mathbb{E}\left(X^2\right) - \mathbb{E}\left(X\right)^2}\sqrt{\mathbb{E}\left(Y^2\right) - \mathbb{E}\left(Y\right)^2}} \tag{2.3}$$

To reiterate, the function Υ is dependent on the input x and exhibits correlation with the estimated error between $\tilde{y}$ and $\hat{y}$. Its formal definition is presented below:

$$\Upsilon\left(x|W\right) \coloneqq h\left(\mathbb{E}\left(d\left(\hat{y}, \tilde{y}\right)\right)\right) \;\; s.t. \;\; \rho_{\Upsilon, f(\hat{y},\tilde{y})} > 0 \tag{2.4}$$

In this context, let d and f denote distance metrics on a set X, such that $f, d : X \times X \rightarrow [0, \infty)$, satisfying the properties of identity, symmetry, and the triangle inequality. It is important to note that Υ does not necessarily correspond to the variance of the distribution $p\left(x, \tilde{y}|W\right)$, but

it must fulfill the condition $\rho_{\Upsilon,\nu} > 0$, where ν represents the variance (which may be challenging to compute for arbitrary distributions). Intuitively, Υ represents the anticipated error, risk, or lack of confidence in the predicted output. To obtain Υ, which will be referred to as "uncertainty" for easier comprehension, a self-supervised optimization of the following function needs to be performed.

$$\underset{\tilde{y},\Upsilon}{\operatorname{argmin}}\, h\left(\Upsilon\right) f\left(\hat{y},\tilde{y}\right) + \lambda g\left(\Upsilon\right) \tag{2.5}$$

In the above optimization function, the function g represents a monotone function of the uncertainty, ensuring preservation of domain order and convexity. On the other hand, the function h is responsible for inverting the monotonicity of g, satisfying $\rho_{h,g} < 0$ (where h could also be a function of g). The rationale behind this formulation is to establish a two-way coupling between Υ and $\tilde{y}$ in order to prevent trivial solutions and appropriately scale the values. The term $h\left(\Upsilon\right) f\left(\hat{y},\tilde{y}\right)$ scales the value of $f\left(\hat{y},\tilde{y}\right)$ based on the uncertainty per input dimension, simulating "outlier rejection" by weighing different noisy observations. It can be considered as a loss attenuator. However, this approach can lead to trivial solutions where $\Upsilon \to \infty$ (if unbounded) to minimize the loss. To mitigate this issue, a simple penalty term $\lambda g\left(\Upsilon\right)$ is added to counteract the occurrence of exploding values for Υ. This formulation extends the work presented in [28]. The selection of the functions g, h, and f is at the discretion of the user and can be tailored based on domain-specific knowledge. The relationship between f, g, and h has been established in previous studies, as shown in Table 2.1. It is important to note that our formulation is derived by summarizing a substantial amount of prior work from various domains that estimate uncertainty, risk, and/or learned robustness parameters. We identified a common trend in these previous

works and developed a blueprint function that can be employed to design novel loss functions. In summary, we unify previous approaches into a single generalized function, and specific functional parameters from our formulation (Eq. 2.5) can be substituted to obtain the previously proposed works (Table 2.1).

Note that in the formulation presented, Υ can represent either uncertainty (similar to co-variance) or lack of confidence (risk) of any arbitrary distribution. For complex distributions, Υ can be a complex function of the variance ν, resulting in qualitative rather than quantitative uncertainty. However, by carefully selecting functions f, g, h, and λ, Υ can be transformed into a quantitative function of ν with straightforward closed-form solutions. In such cases, it is also possible to work towards certifying the robustness of neural networks within a limited domain of training/operating data.

Formally, a network is considered certifiably robust when the error in predicting perturbed inputs is bounded by a value τ. If x is the input and x' is the perturbed input, the l_p distance between their respective outputs should be constrained to τ, expressed as $\|\mathbb{N}(x|W) - \mathbb{N}(x'|W)\|_p \leq \tau$. We hypothesize that this definition of robustness should also incorporate the network's confidence as an additional constraint. In essence, the network would "inform" us when it is speculating a failure. However, such a formulation requires comprehensive mathematical treatment and falls beyond the scope of this chapter. Moreover, we consider it a promising direction for future research endeavors.

By employing Eq. 2.5 in a self-supervised manner, Υ is learned in conjunction with $\tilde{y}$, with both being dense and exhibiting variations across pixel locations $\mathbf{x}$. In practical terms, the minimized loss function can be expressed as follows:

$$\underset{\tilde{y},\Upsilon}{\operatorname{argmin}} \mathbb{E}\left(h\left(\Upsilon_{\mathbf{x}}\right) f\left(\hat{y}_{\mathbf{x}}, \tilde{y}_{\mathbf{x}}\right)\right) + \lambda \mathbb{E}\left(g\left(\Upsilon_{\mathbf{x}}\right)\right) \forall \mathbf{x} \tag{2.6}$$

The above equation models the distribution given below (assuming the minimization of the Negative Log-Likelihood (NLL) in Eq. 2.6).

$$p\left(\hat{y}|\tilde{y}, \Upsilon\right) \propto \prod_{\forall \mathbf{x}} \exp\left(-h\left(\Upsilon_{\mathbf{x}}\right) f\left(\hat{y}_{\mathbf{x}}, \tilde{y}_{\mathbf{x}}\right)\right) \exp\left(-g\left(\Upsilon_{\mathbf{x}}\right)\right) \tag{2.7}$$

It is important to note that, throughout the remainder of this chapter, the symbol Υ represents heteroscedastic aleatoric uncertainty, unless otherwise specified. For the sake of simplicity, we will use the term '*uncertainty*' to refer to heteroscedastic aleatoric uncertainty, unless stated otherwise.

2.2.2 Informational Cues from Uncertainty $\boldsymbol{\Upsilon}$

The answer to the question "What informational cues does the uncertainty hold?" can be found by considering the uncertainty of specific quantities. In the context of robot autonomy, the estimation of the following quantities is typically of interest: 1. Optical flow, 2. Monocular/Stereo depth, 3. Surface normals, and 4. Semantic segmentation. Therefore, in the subsequent subsections, we will concentrate on these four quantities.

2.2.3 Uncertainty of Optical Flow

Optical flow at a pixel located at $\mathbf{x} = \begin{bmatrix} x & y \end{bmatrix}^T$ is denoted as $\dot{\mathbf{p}}_{\mathbf{x}}$ and is given by

$$\dot{\mathbf{p}}_{\mathbf{x}} = \frac{1}{Z_{\mathbf{x}}} \begin{bmatrix} xV_z - V_x \\ yV_z - V_y \end{bmatrix} + \begin{bmatrix} xy & -(1+x^2) & y \\ (1+y^2) & -xy & -x \end{bmatrix} \Omega \tag{2.8}$$

Here, $V = \begin{bmatrix} V_x & V_y & V_z \end{bmatrix}^T$, $\Omega = \begin{bmatrix} \Omega_x & \Omega_y & \Omega_z \end{bmatrix}^T$ denotes the 3D linear and angular velocities of the camera respectively. $Z_{\mathbf{x}}$ denotes the depth at a pixel $\mathbf{x}$. To gather insight into when a high uncertainty would be obtained we need to revisit the mathematical definition of optical flow. The optical flow $\dot{\mathbf{p}_{\mathbf{x}}}$ essentially is giving us the matching of each pixel $\mathbf{x}$ between two images and is obtained by solving the brightness constancy equation

$$\underset{\dot{\mathbf{p}_{\mathbf{x}}}}{\operatorname{argmin}} \mathcal{I}_t(\mathbf{x}) - \mathcal{I}_{t+\delta t}(\mathbf{x} + \dot{\mathbf{p}}_{\mathbf{x}}) \tag{2.9}$$

Here, $\mathcal{I}_t(\mathbf{x})$ and δt denote the brightness of the point at $\mathbf{x}$ at time t and small time increment respectively. In practice, since a single point sample at $\mathbf{x}$ would be too noisy to match, a small brightness over a small patch neighborhood $\mathcal{N}$ is matched using some function of the difference in patch brightness denoted by f (this could be as simple as the sum) and is given as

$$\underset{\dot{\mathbf{p}_{\mathbf{x}}}}{\operatorname{argmin}} f\left(\mathcal{I}_t(\mathbf{x}) - \mathcal{I}_{t+\delta t}(\mathbf{x} + \dot{\mathbf{p}}_{\mathbf{x}})\right) | \forall \mathbf{x} \in \mathcal{N} \tag{2.10}$$

The Lipschitzness of the estimated optical flow $\dot{\mathbf{p}_{\mathbf{x}}}$ to noise is defined in terms of the estimate's robustness to perturbations and is expressed as:

$$d\left(f(\dot{\mathbf{p}}), f(\dot{\mathbf{p}} + \nu)\right) \leq K d(\dot{\mathbf{p}}, \dot{\mathbf{p}} + \nu) \tag{2.11}$$

Here, ν represents the noise in the estimate, K denotes a positive constant, and d stands

for a distance metric. A smaller value of K indicates a lower sensitivity. Now, let us examine the condition under which a small error in the estimated flow significantly affects the brightness matching score. To comprehend this phenomenon mathematically, we will consider the Mutual Information (Eq. 2.12) between the distributions of optical flow with and without noise. As $\nu \to 0$, the mutual information $I\left(f\left(\dot{\mathbf{p}}_{\mathbf{x}}\right); f\left(\dot{\mathbf{p}}_{\mathbf{x}}+\nu\right)\right)$ should be maximized.

$$I\left(f\left(\dot{\mathbf{p}}_{\mathbf{x}}\right); f\left(\dot{\mathbf{p}}_{\mathbf{x}}+\nu\right)\right) = D_{\mathrm{KL}}\left(P_{f(\dot{\mathbf{p}}_{\mathbf{x}}),f(\dot{\mathbf{p}}_{\mathbf{x}}+\nu)} \middle\| P_{f(\dot{\mathbf{p}}_{\mathbf{x}})} \otimes P_{f(\dot{\mathbf{p}}_{\mathbf{x}}+\nu)}\right) \tag{2.12}$$

Here, $f\left(\dot{\mathbf{p}}_{\mathbf{x}}\right)$ denotes the matching score from Eq. 2.10, P_a denotes the marginal distribution of a and $\otimes$ is the operator that multiplies two marginal distributions. Now, since the neighborhood $\mathcal{N}$ is small, the deviation of x, y inside the neighborhood is small. V, Ω are intrinsic camera properties and do not depend on the scene. The only geometric variable left that can affect the distributional shift of $f\left(\dot{\mathbf{p}}_{\mathbf{x}}+\nu\right)$ away from $f\left(\dot{\mathbf{p}}_{\mathbf{x}}\right)$ even when ν is small is large changes to $Z_{\mathbf{x}}$. This means that the depth varies a lot spatially with small changes to location, i.e., $\nabla_{\mathbf{x}} Z$ is large. These are depth discontinuities or edges. They are generally dominated by object boundaries. Hence, a large uncertainty is correlated with object boundaries.

The keen reader might also think about the fact that the Z is a latent variable that implicitly influences Υ, but the appearance of the image should also be affecting it directly. Intuitively, should not Υ have high values for areas that cannot be matched? Indeed this is true, this is the other case where large "flat" (uniform color) regions will lead to large uncertainty due to the absence of non-distinct features. Here a feature would be called distinct based on the structure

tensor M

$$M = \begin{bmatrix} \nabla_x \mathcal{I}^2 & \nabla_x \mathcal{I} \nabla_y \mathcal{I} \\ \nabla_x \mathcal{I} \nabla_y \mathcal{I} & \nabla_y \mathcal{I}^2 \end{bmatrix} \tag{2.13}$$

Let λ_1 and λ_2 be the Eigenvalues of M, then if $\lambda_1 \approx \lambda_2 \to 0$, the region is flat. Note that there is a minimum size of the neighborhood $\mathcal{N}$ that is flat for uncertainty to be high and this neighborhood size is directly related to the receptive field.

To summarize, a high uncertainty Υ on estimated optical flow $\dot{\mathbf{p}}_{\mathbf{x}}$ is either due to depth boundaries or flat regions in the image larger than the receptive field of the network or when severe local illumination changes are encountered. From a slightly different perspective, depth boundaries give rise to occlusions (parts of the scene are covered by other parts of the scene that are closer) or accretions (parts of the scene are revealed as the closer part occluding it has now moved away) and in-turn leads to high Υ since there is no mapping from $\mathcal{I}_t(\mathbf{x})$ to $\mathcal{I}_{t+\delta t}(\mathbf{x} + \dot{\mathbf{p}}_{\mathbf{x}})$. Formally,

$$\nexists \dot{\mathbf{p}}_{\mathbf{x}} \text{ s.t. } f\left(\mathcal{I}_t(\mathbf{x}) - \mathcal{I}_{t+\delta t}(\mathbf{x} + \dot{\mathbf{p}}_{\mathbf{x}})\right) | \forall \mathbf{x} \in \mathcal{N} \to 0 \tag{2.14}$$

In the case of a flat patch, the failure is due to a non-unique mapping from $\mathcal{I}_t(\mathbf{x})$ to $\mathcal{I}_{t+\delta t}(\mathbf{x} + \dot{\mathbf{p}}_{\mathbf{x}})$. Formally,

$$\exists \{\dot{\mathbf{p}}^i_{\mathbf{x}} | i > 1\} \text{ s.t. } f\left(\mathcal{I}_t(\mathbf{x}) - \mathcal{I}_{t+\delta t}\left(\mathbf{x} + \dot{\mathbf{p}}^i_{\mathbf{x}}\right)\right) | \forall \mathbf{x} \in \mathcal{N} \to 0 \tag{2.15}$$

In other words, the map from $\mathcal{I}_t(\mathbf{x})$ to $\mathcal{I}_{t+\delta t}(\mathbf{x} + \dot{\mathbf{p}}_{\mathbf{x}})$ is surjective.

$$\mathcal{I}_t(\mathbf{x}) \twoheadrightarrow \mathcal{I}_{t+\delta t}(\mathbf{x} + \dot{\mathbf{p}}_{\mathbf{x}}) \text{ s.t. } f\left(\mathcal{I}_t(\mathbf{x}) - \mathcal{I}_{t+\delta t}(\mathbf{x} + \dot{\mathbf{p}}_{\mathbf{x}})\right) | \forall \mathbf{x} \in \mathcal{N} \to 0 \tag{2.16}$$

We experimentally observe that uncertainties are generally relatively higher for depth

discontinuities as compared to other factors.

2.2.4 Uncertainty of Monocular/Stereo Depth

The focus will now be on stereo depth. Stereo depth is calculated using the disparity, which represents the displacement between corresponding pixels in two stereo images. This disparity calculation is a specific case of the optical flow discussed previously. To simplify the discussion, let's assume a horizontal stereo camera system where $V_z = V_y = 0$ since the cameras are perfectly aligned or calibrated to be aligned, and $V_x = b$ (in focal lengths) represents the baseline of the camera system. Additionally, $\Omega = 0$ since there is no rotation between the cameras. It should be noted that in this case, the optical flow (disparity) is computed between two stereo images, rather than different frames from a monocular camera. The equation for the disparity D (or flow) can be expressed as follows:

$$D_{\mathbf{x}} = \frac{bf}{Z_{\mathbf{x}}} \tag{2.17}$$

Here, the focal length f is expressed in pixels, while both b and $Z_{\mathbf{x}}$ are measured in physical units. It is evident that, based on the previous subsection's discussion, the same rationale remains valid.

$$\nexists D_{\mathbf{x}} \text{ s.t. } f\left(\mathcal{I}_L\left(\mathbf{x}\right) - \mathcal{I}_R\left(\mathbf{x} + D_{\mathbf{x}}\right)\right) | \forall\, \mathbf{x} \in \mathcal{N} \rightarrow 0 \tag{2.18}$$

Here, $\mathcal{I}_L$ and $\mathcal{I}_R$ are left and right camera images. To summarize, the depth boundaries between stereo pairs give us a high Υ along with large flat regions and severe illumination changes.

In monocular depth estimation, a deeper understanding of the construction of loss functions

is necessary for analysis. Given the ill-posed nature of estimating depth from a single view without prior knowledge, many studies incorporate a penalty on $\nabla \tilde{Z}\mathbf{x}$ to discourage significant changes in the estimated depth $\tilde{Z}\mathbf{x}$ within a small region. This penalty is justified by the observation that most surfaces exhibit smoothness, except at the boundaries of objects. By following a similar line of reasoning as previously discussed, it can be inferred that even when employing a monocular depth estimation network, object boundaries will typically result in higher values of Υ.

2.2.5 Uncertainty of Surface Normals

Imagine a surface $\mathcal{S}$ being imaged onto an image plane as follows

$$\mathbf{x} = K \left[R, T \right] \mathbf{X} \tag{2.19}$$

Here, the camera intrinsic matrix is denoted as K, and $\left[R, T \right]$ represents the relative pose of the camera in relation to the surface. Without loss of generality, we can assume $R = \mathbb{I}3 \times 4$. The points on the image plane are represented as $\mathbf{x}$, while $\mathbf{X}$ represents the real-world points on the surface $\mathcal{S}$. The surface normal $\mathbf{nX}$ at a point $\mathbf{X}$ in the real world is defined as $\mathbf{nX} = \nabla \mathcal{S}\mathbf{X}$. The presence of significant local variations in surface normals or changes in viewing direction will result in large values of $\boldsymbol{\Upsilon}$. This can occur in two main scenarios: (a) when there are drastic changes in surface normals, and (b) when there are substantial local variations in illumination.

In the first case, the image capture process involves projecting a three-dimensional scene onto a two-dimensional plane, leading to an inverse relationship between the depth dimension $Z_{\mathbf{x}}$ and the spatial representation of x and y. Consequently, even small changes in x and y can cause

significant changes in $\mathbf{n_x}$ (the normal on the image plane) only if there are large variations in $Z_\mathbf{x}$. It is important to note that the gradient becomes ill-defined at the intersection of two surfaces with different normal directions, which intuitively explains this phenomenon.

In the second case, the reflective properties of the material and/or drastic movement of the light source can impact the uncertainty value. Specifically, when observing a reflective surface, it is anticipated that high uncertainties will arise due to the specularities caused by point sources of light. In this situation, the incorporation of prior knowledge regarding the estimated quantity (surface normals) can be employed to determine the values of f, g, and h as described in [42].

2.2.6 Uncertainty of Semantic Segmentation

Given that semantic segmentation relies on the local appearance within the limited locality size determined by the network's receptive field to predict class labels for each pixel, a high value of Υ indicates situations where the appearance lacks distinctiveness to be accurately classified into a single class. For instance, a white barrel may resemble a lane line, or pavement may resemble a road due to changes in illumination. In such cases, Υ can be employed to temporally filter semantic labels for purposes such as odometry [51] or enhancing robustness. As this study specifically focuses on exploring the nontraditional applications of uncertainty, the investigation of this aspect is left as a potential avenue for future research.

In summary, the heteroscedastic aleatoric uncertainty serves as a loss attenuator during the learning process. Consequently, the uncertainty is high when encountering additions or deletions in the scene, typically observed at object boundaries and depth transitions. Therefore, uncertainty can be considered as an attention mechanism when estimating a quantity that exhibits

discrepancies at object edges and depth boundaries.

Astute readers may question why epistemic uncertainty cannot be employed for the same purpose. Epistemic uncertainty models factors beyond the learned data distribution during training. While it may be effective in detecting zero-shot obstacles, it necessitates careful construction of the training set, which is often challenging. Moreover, compared to aleatoric uncertainty, epistemic uncertainty requires conducting N passes of the network, further complicating the process.

2.2.7 Uncertainty and its relationship to Confidence and Inlier ratio

The concept of loss function attenuation based on a criterion has been extensively explored in prior research. This attenuation has primarily been investigated through two formulations: one involving an inlier ratio and the other involving a robustness parameter. The first formulation is employed when the goal is to learn a simplified model from data in an unsupervised or self-supervised manner. For instance, when regressing a six-degree-of-freedom camera pose from a pair of images containing numerous moving objects, it becomes necessary to focus solely on the background regions to obtain a robust estimation. This approach models a subset (a special case) of regions with high uncertainty. The second formulation is utilized when robustness is desired, particularly in the presence of erroneous labels. This formulation typically involves an optimization problem of the following nature: $\operatorname{argmin}_{\tilde{y},\alpha} f\left(\tilde{y}, \hat{y}, \alpha\right)$, where α represents a robustness parameter (per-pixel in the case of images), and f denotes a distance function. This approach estimates the *importance* of a pixel in predicting $\tilde{y}$ as closely as possible to $\hat{y}$. In this context, the measure of importance is inversely correlated with the uncertainty measure.

Table 2.1: Relation to existing works (Chronological Order).

$f(\tilde{y}, \hat{y})$	$h(a)$	$g(a)$	λ	y	Reference
$\|\tilde{y} - \hat{y}\|_2^2$	a^{-2}	$\log(a^2)$	1.0	Semantic Segmentation	[28]
$\|\tilde{y} - \hat{y}\|_1$	a	$-\log(a)$	0.2	Monocular Depth	[50]
$(\tilde{y} - \hat{y})^2$	a^{-1}	$\log(a)$	1.0	Optical Flow	[29]
$\|\tilde{y} - \hat{y}\|_1$	a^{-1}	$\log(a)$	1.0	Optical Flow	[39]
$\begin{cases} 0.5(\tilde{y} - \hat{y})^2, \\ \text{if } \|\tilde{y} - \hat{y}\| < 1 \\ \|\tilde{y} - \hat{y}\| - 0.5, \\ \text{otherwise} \end{cases}$	a^{-2}	$\log(a^2)$	2.0	3D Bounding Box	[36]
$\|\tilde{y} - \hat{y}\|_2^2$	a^{-2}	$\log(a)$	6.0	Optical Flow	[33]
$\|\tilde{y} - \hat{y}\|_2^2$	a^{-1}	$\log(a)$	1.0	Visual Odometry	[48]
$\|\tilde{y} - \hat{y}\|_2^2$	a^{-1}	$\log(a)$	1.0	Dense Depth	[46]
$\|\tilde{y} - \hat{y}\|_1$	a^{-1}	$\log(a)$	1.0	Monocular Depth	[47]
$\|\tilde{y} - \hat{y}\|_1$	$\frac{1}{\log(1+e^{a+\epsilon})}$	$\log(1 + e^a)$	1.0	Optical Flow	[12]
$\|\tilde{y} - \hat{y}\|_1$	a^{-1}	$\log(a)$	$\frac{1}{\sqrt{2}}$	Stereo Disparity	[41]
$\|\tilde{y} - \hat{y}\|_1$	a^{-1}	$\log(a)$	1.0	Monocular Depth	[40]
$\cos^{-1}(\tilde{y}^T \hat{y})$	$-a$	$\log\left(\frac{1+e^{\pi a}}{1+a^2}\right)$	1.0	Surface Normals	[42]
$(\tilde{y} - \hat{y})^2$	a^{-2}	$\log(a^2)$	2.0	Optical Flow	[38]
$(\tilde{y} - \hat{y})^2$	a^{-2}	$\log(a^2)$	1.0	Monocular Depth	[44]
$\|\tilde{y} - \hat{y}\|_2^2$	a^{-2}	$\log(a^2)$	1.0	Pixel Color	[45]
$(\tilde{y} - \hat{y})^2$	a^{-2}	$\log(a^2)$	1.0	Monocular Depth	[43]

2.3 Results

2.3.1 Quadrotor Platform

The experiments utilize a custom-built quadrotor platform called PRGLabrador500 [52]. The platform features an X-shaped frame with dimensions of 500 mm (motor to motor). T-Motor F80 Pro 2500KV motors are used in conjunction with 6042×3 propellers. The ArduPilot 4.1.4 firmware, operating on the Holybro Kakute F7 flight controller, is responsible for the position-hold and lower-level control. Additionally, the platform is equipped with a

GL9306 optical flow sensor and a Benewake TFMini-S LIDAR serving as the altimeter source. Navigational commands at a higher level are processed and transmitted by the companion computer NVIDIA Jetson TX2 [53] using RC-Override, which interacts with the flight controller operating in Loiter mode via MAVROS. On the Jetson TX2, vision and planning algorithms are executed onboard at an approximate frequency of 8 Hz using Python 3.6. The quadrotor, including a 1800mAh 3S LiPo battery, has a take-off weight of 1110 g, a thrust-to-weight ratio of 4.9:1, and a flight time of approximately 10 minutes. All flight experiments are conducted within the Brin Family Aerial Robotics Lab at the University of Maryland, which offers a flying volume measuring $7.3 \times 5.5 \times 5$ m^3.

2.3.2 Perception Pipeline

In this section, the overall procedure for achieving the proposed tasks will be described. It involves two key steps: (a) Perception and (b) Control. The perception pipeline, which is common for every task, will be described next.

The perception pipeline executes on consecutive RGB color frames with an image size of 320×240 px at a frame rate of 30 Hz. These sequential image frames are inputted into the neural network, specifically the EVPropNet architecture [54], with a modification in the number of output channels. In our case, the network has four output channels instead of one [54]. The network, named *Ajna*, consists of 2.72M parameters and requires approximately 6.3GFLOPs for a forward pass. The model size of *Ajna* is 10.40MB, and each inference takes around 49ms (20.4 Hz) for a batch size of one. Training of *Ajna* involves the utilization of the loss function described in Eq. 2.5, which employs self-supervision for uncertainty learning and supervised

labels for target prediction learning. Specifically, the network is trained to predict dense optical flow $\widetilde{\dot{\mathbf{p}}_{\mathbf{x}}}$ and its corresponding dense heteroscedastic aleatoric uncertainty $\boldsymbol{\Upsilon}_{\mathbf{x}}$. Training of *Ajna* is performed on the Flying Chairs 2 dataset [55, 56] for 400 epochs with a learning rate of 10^{-4}. Subsequently, it is further trained for 50 additional epochs on the FlyingThings3D dataset [57] with a learning rate of 10^{-5}. A batch size of 32 is employed for training. The loss function used for training our networks is based on [12].

$$\underset{\widetilde{\dot{\mathbf{p}}_{\mathbf{x}}}, \boldsymbol{\Upsilon}_{\mathbf{x}}}{\operatorname{argmin}} \mathbb{E}\left(\frac{\widetilde{\dot{\mathbf{p}}_{\mathbf{x}}} - \widehat{\dot{\mathbf{p}}_{\mathbf{x}1}}}{\log\left(1 + e^{\boldsymbol{\Upsilon}_{\mathbf{x}} + \epsilon}\right)} + \lambda \log\left(1 + e^{\boldsymbol{\Upsilon}_{\mathbf{x}}}\right)\right) \tag{2.20}$$

In our mathematical formulation in Eq. 2.5, this is equivalent to using $f\left(\tilde{y}, \hat{y}\right) = \tilde{y} - \hat{y}_1$, $h\left(a\right) = {}^{1}\!/_{\log\left(1+e^{a+\epsilon}\right)}$, $g\left(a\right) = \log\left(1 + e^{a}\right)$ with $\epsilon = 10^{-3}$, $\lambda = 1.0$ and $\mathbb{E}$ is the expectation/averaging operator. All the hyperparameters are obtained through cross-validation.

To summarize, the networks receive consecutive image frames as input and produce four output channels: two channels for optical flow in the x and y directions, and two channels for the uncertainty of the optical flow in the x and y directions. The optical flow vector at pixel location $\mathbf{x}$ is represented as $\widetilde{\dot{\mathbf{p}}_{\mathbf{x}}} \in \mathbb{R}^{2\times 1}$, and its aleatoric heteroscedastic uncertainty is denoted as $\boldsymbol{\Upsilon}_{\mathbf{x}} \in \mathbb{R}^{2\times 1}$. Subsequently, the predicted optical flow uncertainty $\boldsymbol{\Upsilon}$ is used to compute a point on the image through morphological operations. This point is then utilized to determine the control strategy based on the specific task, as described in the subsequent sections.

It is important to emphasize that the purpose of this work is to demonstrate the utilization of uncertainty in various robotics applications and how such a formulation can unify different classes of robotics problems. Therefore, no additional information such as color, optical flow, or depth is utilized in our experiments, except for comparative purposes in this chapter. Moreover,

no prior knowledge regarding the placement or type of structures employed in our experiments is assumed. Our control actions rely solely on the Υ obtained from the current image pairs, without employing temporal smoothing or filtering techniques. All the perception, planning, and control algorithms are implemented on the NVIDIA Jetson TX2 and the flight controller of the aerial robot, enabling effortless portability to a palm-sized aerial robot [58] [59] We describe the specific experiment and its environmental setup along with the control policies for four applications in the following sections: (a) Dodging dynamic obstacles (Sec. 2.3.3), (b) Navigating through unstructured environments (Sec. 2.3.4), (c) Flying through an unknown gap (Sec. 2.3.5) and (d) Finding the object pile (Sec. 2.3.6).

2.3.3 Application 1: Dodging Dynamic Obstacles

In this experiment, a method is presented for detecting and dodging unknown (zero-shot) dynamic obstacles using only a monocular camera. The procedure of dodging dynamic obstacles involves three key steps: (a) Detection of the obstacle or independently moving object(s), (b) Prediction of the obstacle trajectory on the image plane, and (c) Invoking a dodging maneuver to avoid getting hit by the obstacle(s). The fact that a dynamic obstacle will have the maximum amount of occlusions and accretions on the consecutive frames from a hovering quadrotor is utilized, resulting in high uncertainty of optical flow. The dynamic obstacle is detected by performing simple morphological operations on the obtained uncertainty map. Furthermore, the obstacle is tracked over three frames by detection (segmentation) to compute the direction the obstacle would hit on the image plane. Subsequently, a safe direction is computed, and a control command is executed to move in that direction for best-effort dodging as proposed in [60]. The

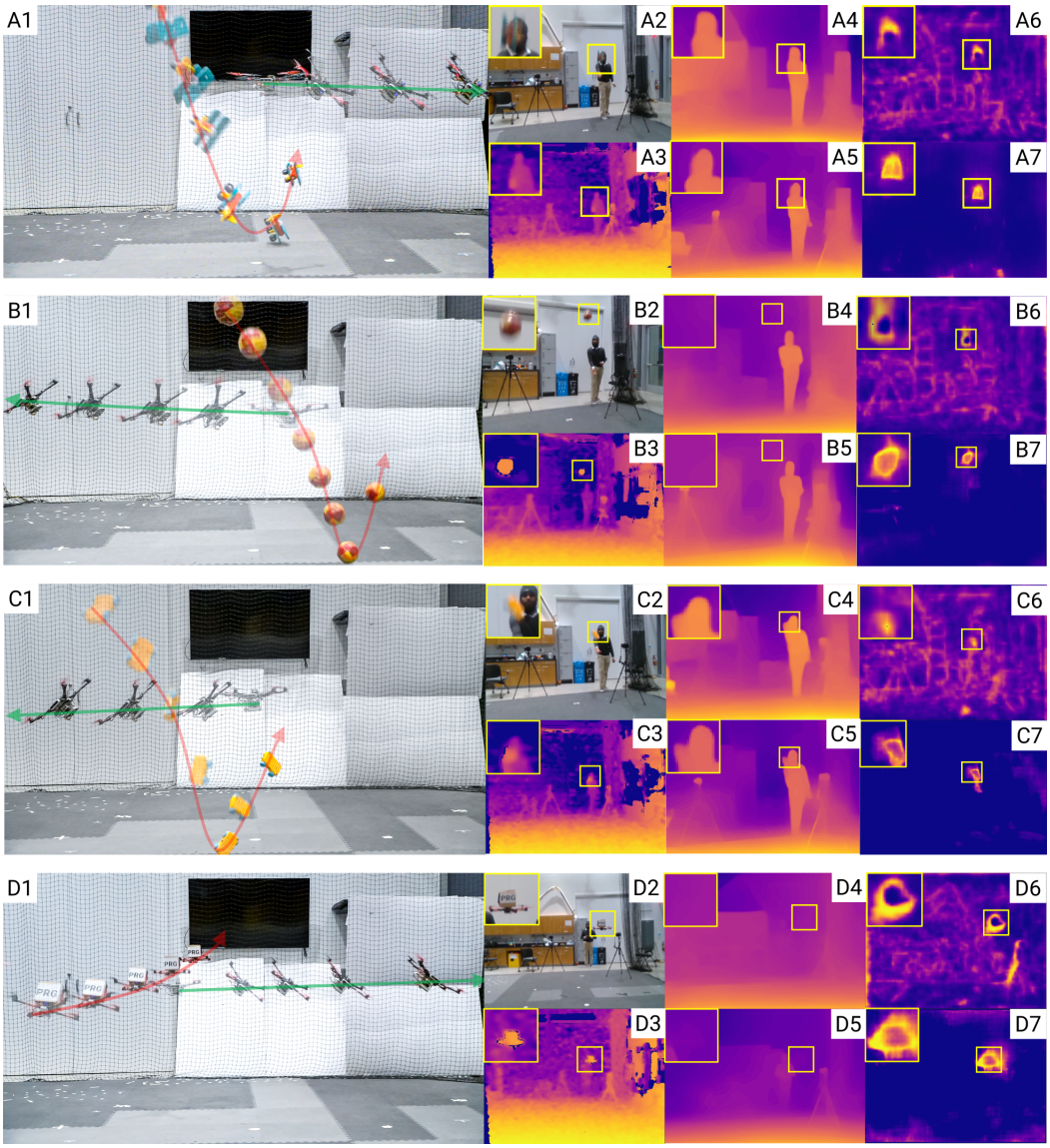

Figure 2.2: A sequence of images of quadrotor dodging objects. Dodging (A) Airplane, (B) Ball, (C) Cart and (D) Drone. Here, the object and quadrotor transparency show the progression over time. Red and green arrows indicate object and quadrotor directions, respectively. In each sub-figure, the outputs are shown in the following order (taking example as sub-figures of A): (A1) Image sequence of dodging, (A2) RGB image as seen by the quadrotor, (A3) D435i depth image, (A4) MiDaS-S output, (A5) MiDaS output, (A6) OccMask, (A7) *Ajna*. The color map used in all the depth images is `plasma`, where blue color represents far and yellow is close. The colormap for occlusion and uncertainty map is inverse `plasma`, where blue color represents lower uncertainty/occlusions and yellow represents higher uncertainty/occlusions. The yellow boxes show the zoomed-in view of the object. *The colormap is consistent across all figures in this chapter.*

experimental setup and results are described in the following sections.

The experimental setup contains a hovering quadrotor at which the obstacles are thrown or flown into such that a collision would definitely occur if the quadrotor were to hold its position and would not invoke a dodging maneuver. We experiment with four different obstacles, varying in shape, size, color, texture and trajectory: (a) a Spherical ball of diameter 140mm, (b) Toy car of size $185 \times 95 \times 45$mm, (c) Toy airplane of size $270 \times 250 \times 160$mm and (d) PRGHusky360 quadrotor of size $440 \times 370 \times 160$mm. Note that no prior information about the objects is used in any of the experiments. Objects (a) to (c) are thrown at the quadrotor and follow a parabolic trajectory under the influence of gravity and object (d) follows a linear trajectory. The objects are thrown or flown at speeds of 4.5 to 8.0 ms^{-1} from a distance ranging from 4.8 m to 6.0 m. We achieve an overall success rate of 83.3% over 60 trials. We compare our results with depth-based methods, event-based methods and occlusion-based methods (See Fig. 2.2 and Table 2.2). In the depth-based methods, D435i gives true scale depth whereas the MiDaS and MiDaS-S [61] [62] only output relative scale depth. Here, MiDaS denotes the MiDaS v3.0 DPT-Large [62] pre-trained model and MiDaS-S denotes the MiDaS v2.1 small pre-trained model directly obtained from the original work without any fine-tuning or retraining. For comparison with occlusion-based methods, we utilize the predicted occlusion mask from MaskFlowNet [63]. We call this prediction OccMask. In all the depth-based methods, we threshold the depth value as an obstacle when it is closer than a particular depth value to dodge them. Clearly, we observe that Intel RealSense D435i (one of the best depth sensors on the market) is not able to obtain depth on moving objects accurately when they are far, hence necessitating an alternative formulation for dynamic obstacle dodging like the one presented in this work. In the event-based method, the approach is adapted from [64], where the output is the probability of each pixel being an obstacle. Alternatively, a stereo pair of event cameras can provide the guarantee of dodging obstacles [65].

Finally, in the occlusion-based method, we threshold the large values as belonging to dynamic obstacles followed by morphological operations similar to our method Ajna. We compare our results with the aforementioned methods on metrics such as Detection Rate (DR), Run time, FLOPs and number of parameters in Table 2.2(a). Here, we define DR as

$$\text{DR} = \frac{\text{Num. Success}}{\text{Num. Trials}}; \quad \text{Success} := IoU \geq 0.5 \tag{2.21}$$

where $IoU = (\mathcal{D} \cap \mathcal{G})/(\mathcal{D} \cup \mathcal{G})$. Here, $\mathcal{D}$ is the predicted mask and $\mathcal{G}$ is the ground truth mask.

2.3.4 Application 2: Navigating through unstructured environments

In this experiment, we present a method to navigate a quadrotor towards a goal direction through different unstructured environments: (a) Indoor forest, (b) Boxes and (c) Photo-Realistic Simulated Forest. We identify the safe region in the current image to avoid obstacles while also moving towards the goal by dynamically weighing the contributions of the local planner (avoiding obstacles) and global planner (going towards the goal). Once the weighted intermediate goal direction is obtained, a control policy is deployed on the quadrotor to change the current heading direction using a Proportional-Integrative-Derivative (PID) controller to reach the goal whilst avoiding collisions. The current desired direction $\tilde{\mathbf{v}}_g$ is obtained as the weighted sum of the goal direction $\mathbf{v}_g$ and free path direction $\mathbf{v}_{\text{free}}$. $\tilde{\mathbf{v}}_g$ acts as the global planner and $\mathbf{v}_{\text{free}}$ acts as the local planner. This policy is based on the policy from [66] with minor modifications explained next.

Consider a small neighborhood $\mathcal{N}$ on the image plane centered around the intersection of the goal direction vector and image plane. Let $\mathbf{v}_{\text{free}}$ be the geometric center of the largest free space in the neighborhood $\mathcal{N}$. We consider higher values of uncertainty in optical flow to

be closer to the camera since the rotation-compensated optical flow is inversely proportional to depth [9]. Now, let Z_{close} be the closest depth value in $\mathcal{N}$. Let $Z_{o,i}$ denote the depth value in $\mathcal{N}$ in different directions i. We chose the second most 'unsafe' region such that $Z_o = \min(Z_{o,i} > Z_{\text{close}} + \delta)$ where δ is a user-defined heuristic. This formulation is inspired by the classical receding horizon planner [67]. The final control policy is given by

$$\tilde{\mathbf{v}}_g = (1-w)\mathbf{v}_g + w\mathbf{v}_{\text{free}};\ w \in [0,1] \tag{2.22}$$

$$\mathbf{e}(t) = \tilde{\mathbf{v}}_g(t) - \tilde{\mathbf{v}}_{\text{curr}}(t) \tag{2.23}$$

$$\mathbf{u}(t) = K_p\mathbf{e}(t) + K_i\int_0^\tau \mathbf{e}(\tau)d\tau + K_d\frac{d\mathbf{e}(t)}{dt} \tag{2.24}$$

$$w = \frac{1}{1+e^{(-Z_o/Z_{\text{close}})}};\ Z_{\text{close}} = \min Z(x,y)\ \forall\ \{x,y\} \subset \mathcal{N} \tag{2.25}$$

where $\tilde{\mathbf{v}}_{\text{curr}}$ is the estimated current heading direction. Note that the goal vector is dominated by $\mathbf{v}_{\text{free}}$ when the foreground element is very close i.e., $w \to 1$ and by global goal vector $\mathbf{v}_g$ when there are no obstacles nearby i.e., $w \to 0$. We deploy this policy in all three aforementioned environments. We control yaw velocity and net thrust vector to achieve the desired direction. In addition to [66] where both 'safe' and 'unsafe' regions are utilized, our method also weighs in the second most 'unsafe' region.

2.3.4.1 Indoor Forest Environment

An indoor forest environment is constructed using cylindrical play tunnels[1] of diameter 0.48m and height 1.83m (Fig. 2.3A). These tunnels are scattered over the space of 7m $\times$

[1] https://www.target.com/p/antsy-pants-play-tunnel/-/A-51735999

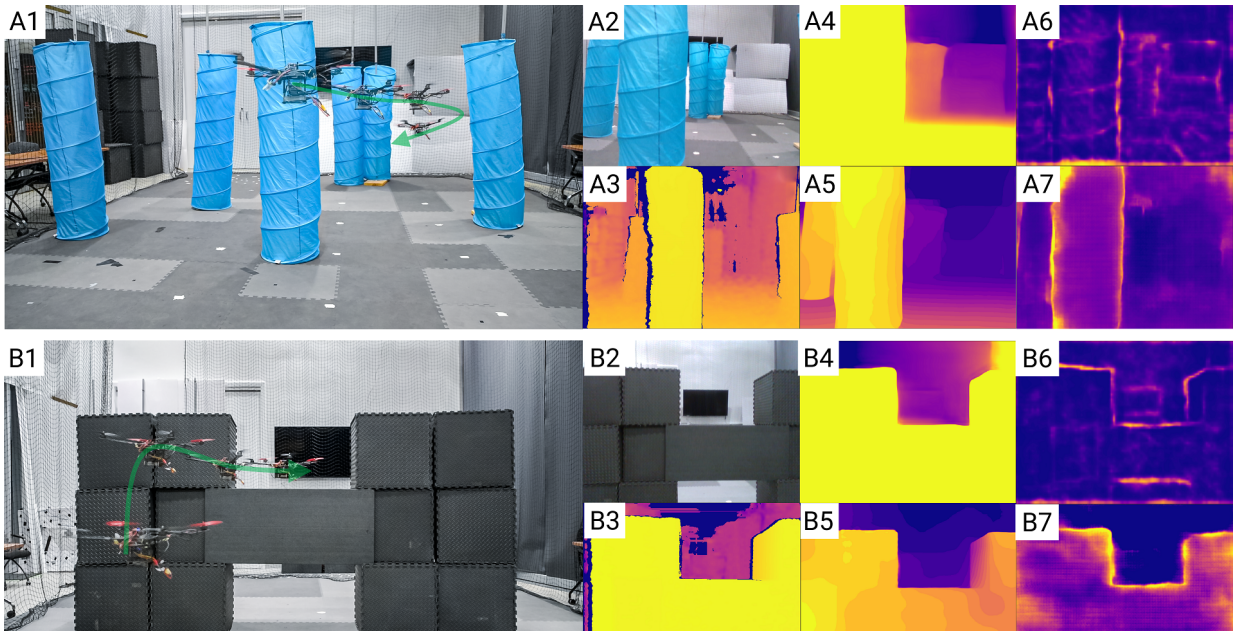

Figure 2.3: A sequence of images of the quadrotor navigating through cluttered environments. (A) Indoor forest, (B) Boxes. Here, the object and quadrotor transparency show the progression of time. Green arrow indicates the quadrotor direction. In each sub-figure, the outputs are shown in the following order (taking example as sub-figures of A): (A1) Image sequence of navigation, (A2) RGB image as seen by the quadrotor, (A3) D435i depth image, (A4) MiDaS-S output, (A5) MiDaS output, (A6) OccMask, (A7) *Ajna*.

5m in an unstructured manner. The quadrotor successfully navigates through the indoor forest environment at an average speed of 3.2m sec^{-1} with a success rate of 86% over 50 trails in different configurations.

2.3.4.2 Boxes Environment

We construct cubes or 'boxes' using yoga mats of dimensions 0.6m×0.6m for our unstructured environment (Fig. 2.3B). The rectangular matt in the middle is of the dimension 1.37m×0.6m. This environment was constructed in the same area as mentioned in 2.3.4.1. The quadrotor successfully navigates through the box environment on an average speed of 1.6m sec^{-1} with a success rate of 82% over 50 trails. Fig. 2.3 shows a comparison of our results with Intel

D435i, MiDaS-S, MiDaS and OccMask.

2.3.4.3 Simulated Forest Environment

RGB images are rendered in Blender®[2] 3D software within a photo-realistic forest scene (Fig. 2.4B). Fig. 2.4A illustrates a simplified top view of the forest. A quadrotor equipped with a camera follows a differential flatness model for its controller. The simulation employs the Cycles rendering engine, which utilizes a ray-tracing algorithm to generate images seen by the quadrotor. The rendering is performed at a resolution of 640×480 px, with a frame rate of 30 frames per second. The forest scene consists of static trees with varying sizes, shapes, and textures. The perception and control algorithms are tested within the simulation. Comparisons are made between *Ajna*, MorphEyes [66], MiDaS-S, MiDaS, and OccMask in terms of Success Rate (SR), average safe point error, run time, and average path length increase compared to the ground truth depth (Table 2.2b and Fig. 2.4). The success rate is defined as the ratio of successful drone navigation trials without collisions to the total number of trials. The average safe point error is the difference between the ideal safe point obtained from the ground truth depth map and the safe point computed by each respective method. The safe point represents the safest point on the image plane that the drone should aim for, as defined in [66]. Note that the safe point error is measured in pixels on the image plane. Average path length increase

2.3.5 Application 3: Flying Through An Unknown Gap

In this experiment, we present a method to detect and navigate through a gap of unknown shape and location using only a monocular camera. The procedure of flying through a gap

[2] https://www.blender.org/

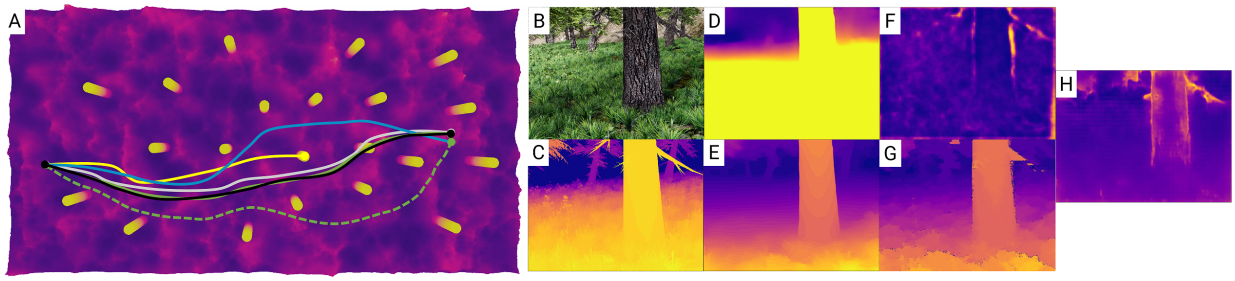

Figure 2.4: Comparison of various methods to navigate through a simulated realistic forest scene. (A) The scene from the top view with paths overlaid (direction of travel is left to right). The legend is as follows: white – ground truth depth, green – MiDaS, dashed green – MiDaS-S, yellow – OccMask, black – MorphEyes, blue – *Ajna* (ours), (B) Sample RGB Image as seen by the quadrotor, (C) ground truth depth, (D) MiDaS-S output, (E) MiDaS output, (F) OccMask output, (G) MorphEyes output and (H) *Ajna* output.

Table 2.2: Quantitative Evaluation for various applications.

(a) Dodging dynamic obstacles

Method	DR (%) ↑	Run Time (ms) ↓	FLOPS (G) ↓	Num. Params (M) ↓
D435i* [68]	100.0	1.0	–	–
MiDaS-S [61]	0.0	12.0	43.7	21.1
MiDaS [61]	3.1	137.8	1052.9	344.6
EVDodgeNet† [64] (SegNet)	40.4	2.5	0.2	0.03
EVDodgeNet† [64] (DB+H+SegFlowNet)	90.7	11.0	5.2	3.6
OccMask [63]	74.8	31.4	62.7	20.6
***Ajna* (Ours)**	89.2	10.1	6.3	2.7

(b) Navigating through unstructured environments

Method	SR (%) ↑	Avg. Path Len. Inc. (%) ↓	Avg. Safe Pt. Error (px.) ↓	Run Time (ms) ↓	FLOPs (G) ↓	Num. Params (M) ↓
MorphEyes‡ [66]	99.0	0.6	1.1	2.5	–	–
MiDaS-S [61]	32.0	7.8	6.8	13.4	43.7	21.1
MiDaS [61]	97.0	1.0	1.1	139.2	1052.9	344.6
OccMask [63]	0.0	–	40.0	35.3	62.7	20.6
***Ajna* (Ours)**	92.0	2.7	4.1	11.6	6.3	2.7

(c) Flying through unknown shaped gaps

Method	DR (%)	Run Time (ms) ↓	FLOPs (G) ↓	Num. Params (M) ↓
GapFlyt (PWC-Net) [9, 69]	94.2	91.0	90.8	8.75
GapFlyt (FlowNet2) [9, 70]	93.0	124.0	24836.4	162.5
GapFlyt (SpyNet) [9, 71]	74.0	70.0	149.8	1.2
D435i* [68]	100.0	1.0	–	–
MiDaS-S [61]	0.0	12.0	43.7	21.1
MiDaS [61]	30.1	137.8	1052.9	344.6
OccMask [63]	56.2	31.4	62.7	20.6
***Ajna* (Ours)**	91.0	10.1	6.3	2.7

* Uses depth images. † Uses event sensor images and results are taken from [64]. ‡ Uses stereo camera images. All other methods above use RGB image(s) as their input.

involves two key steps: (a) Detection of the gap, and (b) Aligning and flying through the gap. In the first step, we utilize the active vision [72,73] philosophy to perform an 'exploratory' maneuver just like in [9]. We then utilize the fact that the uncertainty of optical flow inside the gap would be much higher than outside. This is intuitively true because the number of occlusions and accretions caused inside the gap would be much larger due to the parallax effect. This disparity

in uncertainty enables us to detect the gap in an effortless manner using basic morphological operations. Once the gap is detected, we track the contour indirectly by tracking the foreground (the part outside the gap) and background regions (the part inside the gap) separately. We then propagate the gap contour shape using Focus of Expansion (FOE) constraints as in [9] to fly through the safe point $\mathbf{x_s}$. Further, we track the gap and visually servo towards the safe point by actively switching between background and foreground as necessary [9]. We describe the experimental setup and results next.

The experimental setup contains a rigid scene with two near-planar 'wall' surfaces, namely, the foreground and the background (Fig. 2.5. The foreground contains an unknown-shaped gap. The foreground wall has newspaper stuck on it to add texture to the scene. The background wall has real-life features and hence are not augmented with additional features. The average distance from the initial position of the quadrotor to the foreground and background is 3.0m and 7.2m, respectively. For the detection of the gap, we bump up the proportional gain of the position controller [74, 75] momentarily to invoke random 'exploratory' maneuvers. This could easily be replaced by a fixed diagonal line trajectory like in [9]. Over this maneuver, a number of images are captured. The uncertainty in optical flow between these set of images is used to detect the gap. Similar to [9], once the gap is detected, the foreground and background regions are tracked using Kanade–Lucas–Tomasi tracker [76] and the quadrotor servo towards the gap. We compare our results using optical flow methods, depth-based methods and occlusion-based methods on metrics of Detection Rate (DR), run time, FLOPs and number of parameters in Table 2.2c. The DR is the same as defined in section 2.3.3. We obtained a DR accuracy of 91% over 100 trails across four different unknown-shaped gaps with a minimum tolerance of just 8cm. The optical flow methods for detecting the gap is based on GapFlyt [9] with the input to the algorithm

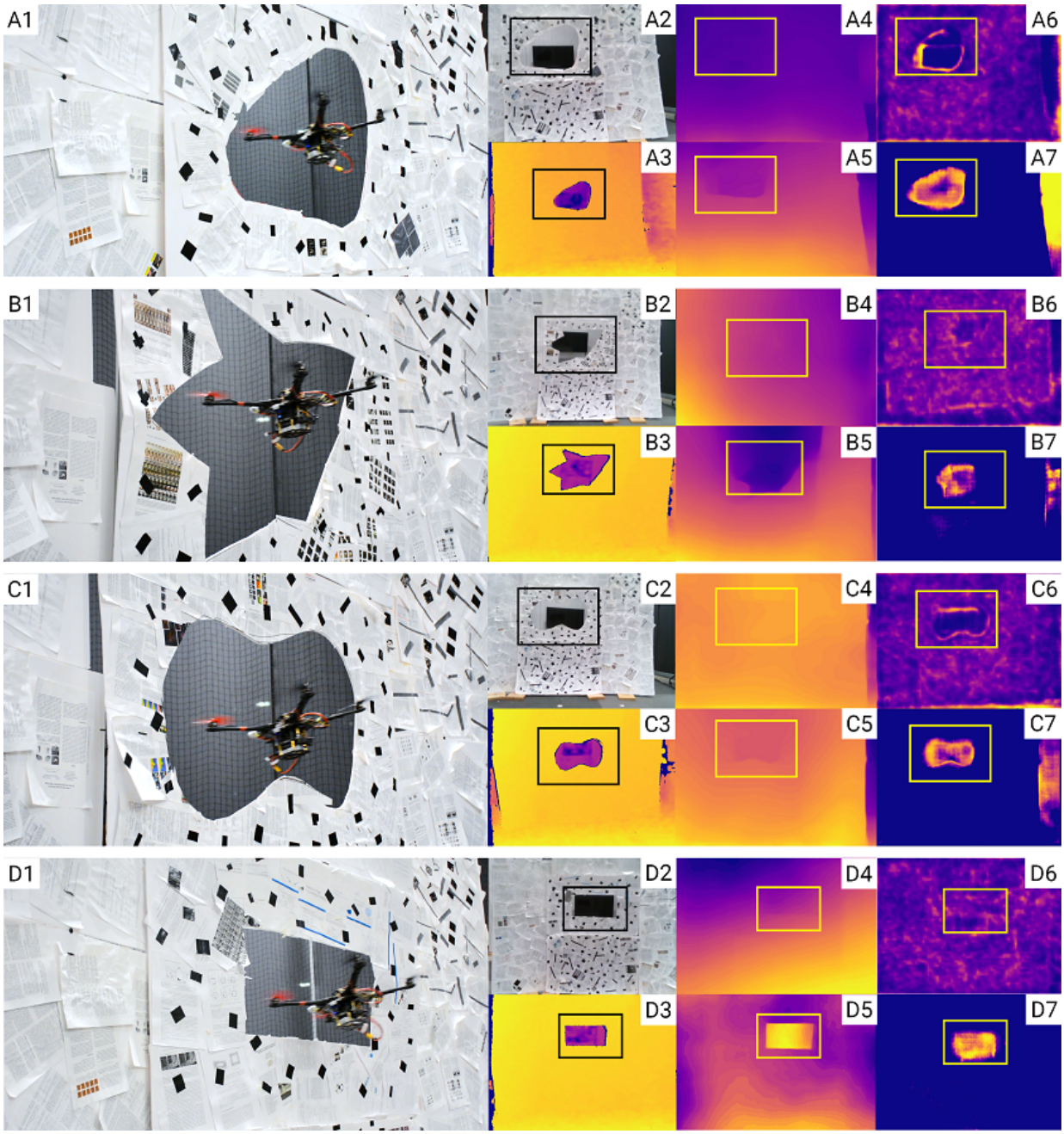

Figure 2.5: Image of quadrotor flying through unknown gaps. (A) `Egg`, (B) `Goku`, (C) `Infinity`, (D) `Rectangle`. In each sub-figure, the outputs are shown in the following order (taking example as sub-figures of A): (A1) Image of flight through the gap, (A2) RGB image as seen by the quadrotor, (A3) D435i depth image, (A4) MiDaS-S output, (A5) MiDaS output, (A6) OccMask, (A7) *Ajna*. The black or yellow boxes on the images show the window location.

coming from different optical flow networks such as PWC-Net, FlowNet2 and SpyNet. In the depth-based methods (D435i, MiDaS, MiDaS-S), the gap is obtained by clustering depth values

into foreground and background, followed by obtaining the region of largest disparity between the values. For the occlusion-based method OccMask, we utilize simple morphological operations after thresholding the values to obtain the gap.

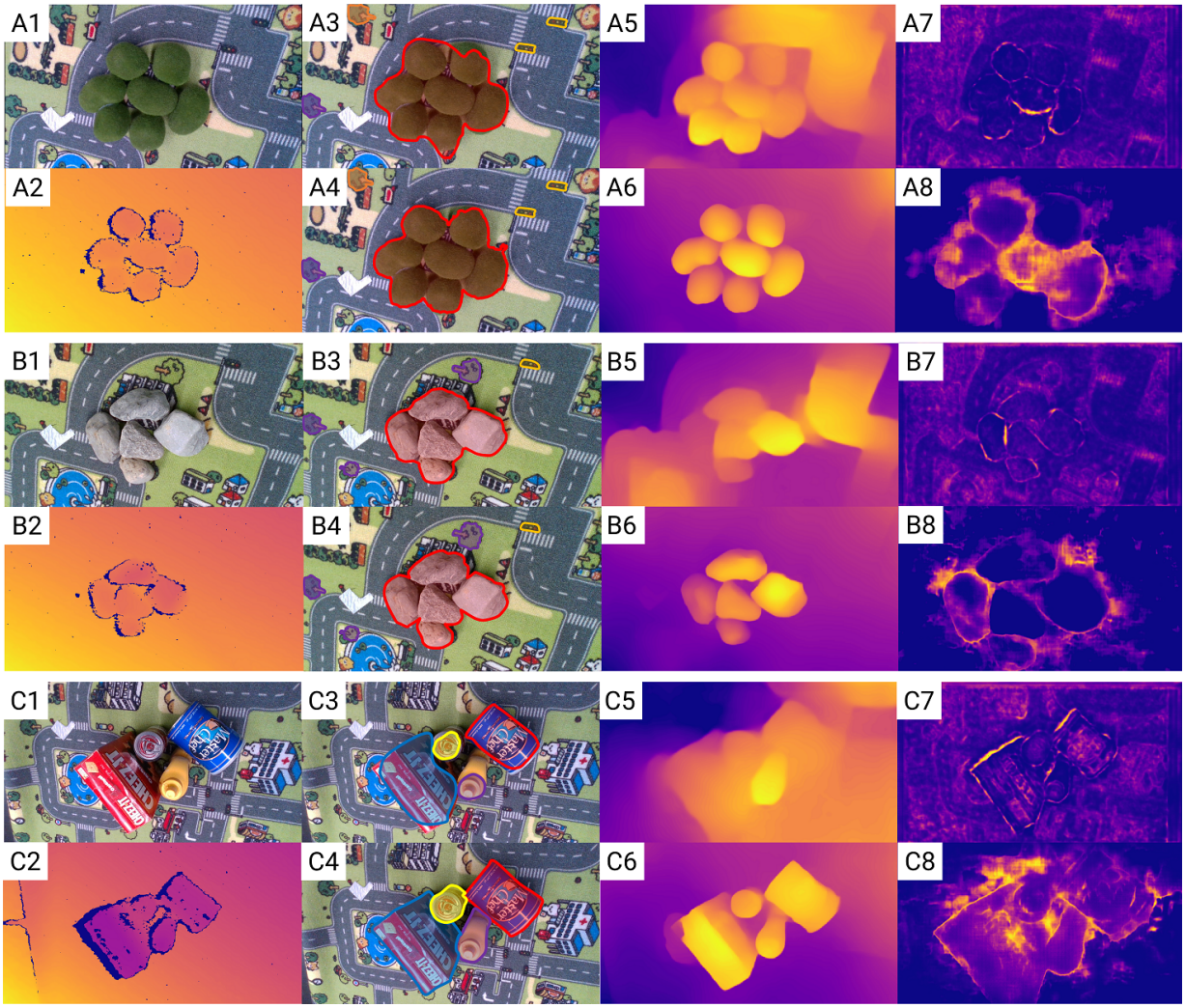

Figure 2.6: Outputs for segmentation experiments using various methods on different datasets: (A) `GrassMoss`, (B) `Rocks`, (C) `YCB`. In each sub-figure, the outputs are shown in the following order (taking example as sub-figures of A): (A1) RGB image as seen by the robot, (A2) D435i depth image, (A3) Mask R-CNN output, (A4) PointRend output, (A5) MiDaS-S output, (A6) MiDaS output, (A7) OccMask, (A8) *Ajna* output. Different colors in A3 and A4 show different objects with different labels being detected by the instance segmentation.

2.3.6 Application 4: Segmentation of Object Pile

We study the possibility of using *Ajna* for foreground-background segmentation tasks. In this experiment, we assume that the object set is placed on a planar surface. We utilize the activeness of the robot to obtain uncertainty due to occlusions. The camera moves actively to take two snapshots from different views of the same scene to compute the uncertainty Υ. The boundary between the foreground and background is correlated to the occlusion between two frames. These occluded regions give rise to high uncertainty. We conduct our experiments on `GrassMoss` (Fig. 2.6A), `Rocks` (Fig. 2.6B) and `YCB` (Fig. 2.6C) dataset as mentioned in [12] on 30 unique configurations per dataset. We qualitatively compare Ajna's output with Intel D435i, Mask-RCNN [18], PointRend [19], MiDaS-S, MiDaS and OccMask. Note that Ajna and OccMask are active approaches and rely on two images to segment the foreground and background. For segmenting in the D435i image, we utilize a plane fitting algorithm on the depth map and remove it for the background subtraction. Object segmentation methods like Mask-RCNN and PointRend rely on features and texture maps. They segment the object cluster but also segment some objects on the background texture (see traffic lights and trees being segmented in Fig. 2.6B3). In MiDaS-S and MiDaS, we segment the background by thresholding the depth value on the predicted output image. In OccMask, the occlusions are visible on the objects but it also results in artifacts in the background textures as well (see Fig. 2.6A7). In Fig. 2.6A8), Ajna gives an estimate of where the object pile is located. It is important to point out that just using $\boldsymbol{\Upsilon}$ directly does not provide segmentation of the object pile. It acts as an attention mechanism to 'show' where the object pile is but is seldom sufficient to provide segmentation of the pile. This can however be used as an initialization step for interactive segmentation as

proposed in [12] for segmenting out zero-shot or unknown objects/samples. If one is segmenting known objects, one can utilize instance segmentation-based methods such as Mask-RCNN or PointRend that identify objects from a learned database and one can extract the required objects directly. An alternative approach is to utilize a method that provides depth like the D435i or infers depth using a neural network like MiDaS or MiDaS-S and one can segment the objects that 'stand out' of the plane of the table. None of the later methods generalizes to novel/zero-shot objects since they rely on learning to predict outputs based on textures rather than geometry. *Ajna* when combined with NudgeSeg can be used as a method to learn novel objects to train methods like Mask-RCNN, PointRend or MiDaS. This can further enable the operation of robots in the wild. Furthermore, it is simple to know if an object is zero-shot by 'looking' at the epistemic uncertainty of predictions and this presents an interesting avenue for future work.

2.3.7 Network Speed on Different Hardware

In this section, we test the *Ajna* network's inference speed on various computing platforms such that they can be deployed on various sized quadrotors (as low as 120 mm sized as shown in Fig. 4.2). The inference time for *Ajna* on an Intel i9 CPU, NVIDIA Titan Xp GPU, NVIDIA Jetson TX2, Intel NCS2 and Google Coral TPU are 140.7ms, 9.1ms, 49.0ms, 268.9ms and 34.4ms respectively. For a detailed comparison of various network architectures for related tasks, we refer the readers to [77]. Here, the time on the Intel i9 CPU and NVIDIA Titan Xp GPU are presented to act as a baseline and cannot be deployed on small robots. The NVIDIA TX2 used in our experiments performs well for small aerial robots. Specialized neural network accelerators such as the Intel NCS2 and the Google Coral TPU are tailor-made for tiny aerial robots. Our

work and its unifying conceptualization will enable a multitude of tasks on tiny aerial robots when coupled with such neural network accelerators.

2.4 Discussion

Perception on robots for various autonomous operations is generally centered around building a 3D representation of the scene using mature Simultaneous Localization And Mapping (SLAM) or Odometry algorithms. Various sensor suites have been utilized to accomplish the aforementioned tasks. Central to such methods are fusing multiple noisy measurements either from single or multiple sensors to obtain more accurate results. This has facilitated the modeling and utilization of uncertainty of various sensing and processing modalities. However, these uncertainties also have latent informational cues which are rarely utilized in robots. Furthermore, when one is building a parsimonious solution to various robotics problems to comply with the Size, Weight, Area and Power (SWAP) constraints to perform all sensing and computation on-board, one has to utilize every bit of informational cue available. This will further lead to unifying various robotic tasks on a parsimonious agent.

In our work, we present one such unifying framework for four common robotic tasks on a resource-constrained aerial robot: (a) dodging dynamic obstacles, (b) navigating through a cluttered environment, (c) flying through gaps, and (d) segmenting of object pile. All these methods rely on the activeness of the agent, i.e., its ability to obtain images from different views. Based on the heteroscedastic aleatoric uncertainty Υ of optical flow, we obtain object boundaries due to accretions and deletions which are central to performing 'depth-based segmentation' that are used in solving the aforementioned four problems using a simple perception stack. Note that,

Υ of optical flow has additional informational cues that aid the detection of dynamic obstacles and navigation problems: motion blur which leads to optical flow being ill-defined. In the dynamic obstacle case, when the obstacle is moving much faster than the robot, the amount of motion blur leads to an ill-posed estimation of optical flow, which gives rise to high Υ. This has the same effect as the properties of event cameras, dynamic obstacles 'stand out' due to the virtue of producing a large number of events which is mimicked by high values of Υ.

In the navigation case, the closer the object, the more amount of motion blur it will have due to slight movement while the image/frame is being acquired, coupled with the fact that the optical flow of the closer objects is higher, the inherent value of Υ will be high. In a qualitative sense, Υ has information from both worlds: depth and motion boundaries. This is exactly what we observe from the results from Fig. 2.4 and Table 2.2(b). Results using *Ajna* are in the middle (in terms of SR, path length and safe point error) of utilizing depth-based methods and occlusion methods. In the occlusion-based method OccMask, there is no trivial way to solve the boundary assignment problem, i.e., which part of the high occlusion region belongs on the tree. This is due to the informational cue about depth being missing in this representation. However, in depth-based methods, the boundary assignment is trivial but the computational cost to obtain the depth map is much higher (at least $\sim 7\times$ as compared to Ajna). Furthermore, works such as [78] avoid processing holes in optical flow, especially near the focus of expansion which leads to a very inefficient solution. Our method can be easily incorporated into such methods to avoid holes in optical flow as shown in Fig. 2.7A. Particularly, in the cases where optical flow values are small near the focus of expansion even though the obstacle is present, our method can provide robustness. Notice how Υ is high even though the optical flow magnitude is low near the focus of expansion because it is on the tree trunk. Such approaches can lead to minimal representations

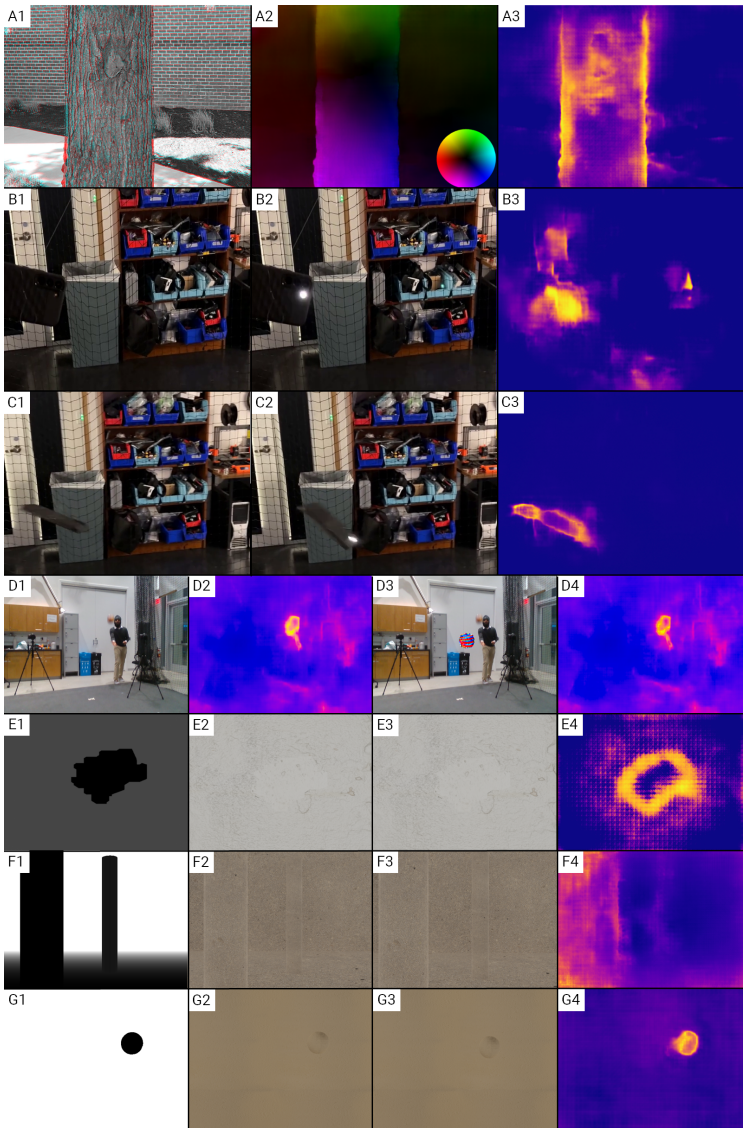

Figure 2.7: (A1) Input image pair as an anaglyph, (A2) Optical flow with colormap shown as inset, (A3) *Ajna*'s predicted uncertainty. Despite low $\dot{\mathbf{p}}$ in the highlighted white region, the quadrotor needs to dodge this area. This is correctly predicted as high $\boldsymbol{\Upsilon}$. This is a common example where $\boldsymbol{\Upsilon}$ provides additional information over $\dot{\mathbf{p}}$. (B1 to B3) input image frames and $\dot{\mathbf{p}}$ under blinking LED without motion. (C1 to C3) input image frames and $\dot{\mathbf{p}}$ under blinking LED with motion. (D1 to D4): Image input, predicted $\boldsymbol{\Upsilon}$, image input with flow attack, predicted $\boldsymbol{\Upsilon}$ under attack. (E), (F) and (G) are experiments of flying through gaps, flying through a forest and detecting dynamic obstacles. Left to right: ground truth depth (white is 4 m and black is 0 m), input images 1 and 2, predicted $\boldsymbol{\Upsilon}$.

for high-speed agile flight through the forest and can advance the work of [79, 80].

We obtain and analyze uncertainty in various different settings (see Fig. 2.7A-G). Fig. 2.7B

shows how uncertainty behaves in the case when only light illumination changes (blinking LED), whereas 2.7C represents uncertainty in the case of both light changes as well object under motion. Fig. 2.7D shows that our method is invariant to black-box adversarial attacks [81]. We observe that in Fig. 2.7D3 the input image has an adversarial patch superimposed, yet this does not affect our uncertainty predictions. Fig. 2.7E-G illustrates three different experiments – flying through unknown gaps, navigating in static environments and dodging dynamic objects. For Fig. 2.7E-G experiments, the first column represents the depth map of the scene (black represents 0 m, white represents 4 m), second and third columns are the consecutive input images to our neural network and last column is the predicted uncertainty. We observed that uncertainty at depth boundaries is almost always far greater than the uncertainty at low-texture regions. This study was performed on a variety of real-world textures with varying the number of texture components (variations in the smoothness of the scene including color-flat or low texture regions) in over 7000 images and 100 textures in three scenarios of detection of unknown gaps (50 different shaped gaps, one such example is shown in Fig. 2.7E) static obstacle environments (one such example is shown in Fig. 2.7F) and dynamic obstacles (one such example is shown in Fig. 2.7G).

We can also analyze how neural networks see depth using a single image [82] versus two or more images. Let us shed some light on the simulation forest experiment. Fig. 2.4B represents the input RGB image of a simulated forest and Fig. 2.4C represents the ground truth depth map. If we look closely, in MiDaS output (Fig. 2.4E), the branch of the tree is missing. Also, notice that the tree branch is present in OccMask and Ajna output (Figs. 2.4F and 2.4H respectively). Note that MiDaS uses a single image to predict depth whereas OccMask and Ajna require two images to predict its output. From Table 2.2, we can conclude that MiDaS has about 128$\times$ more parameters as compared to our Ajna and yet it fails to capture small details like the tree branch. Neural

networks often fail to capture this subtle information from a single image, irrespective of the size of the pre-trained model. For robotics applications such as navigation, this subtle information is crucial in order to avoid these obstacles. Thus, the utilization of multiple images (rather than a single image) to predict quantities like depth can be more beneficial for such applications.

It is important to know that the uncertainty is scene dependent, however, we can reduce the effect of illumination changes by training with a dataset with large illumination changes. We present the results of uncertainty under changing illumination conditions in this document. It is also important to note that our networks are not re-trained or fine-tuned for changing illumination conditions.

Furthermore, we also show results of how uncertainty looks like for flat regions, blinking LED (Fig. 2.12), repetitive patterns (Fig. 2.8) and even a black-box attack on optical flow (Fig. 2.11) [81]. We notice that the optical flow uncertainty is high in all these cases which highlights that this part of the image needs more processing/attention. Furthermore, on examining Figs. 2.8, 2.10, we observe the highest uncertainty is at the accretion and deletion regions.

We analyze the uncertainty of optical flow in various environmental conditions in Fig. 2.10, showcasing that the uncertainty is majorly dominated by depth discontinuities (or accretions/deletions). The various environmental conditions we consider include both real and synthetic textures with different amounts of contrast and texture resolution and different lighting conditions. Specifically, in Fig. 2.8b-f, our approach uses images (experimental setup shown in 2.9) with the same textures on the foreground and the background. In Fig. 2.8d uses an optical illusion scene with a checkboard pattern. Fig. 2.8g-i uses different foreground and background textures with varying texture resolution and 2.8j shows an image with no texture and low contrast difference in foreground and background. It is important to note that, they consistently showcase

the highest uncertainty at the depth discontinuity. From Fig. 2.10, we show the accuracy of the *gap dectection* with different variations in textures.

In the next scenario, we will talk about the results of illumination changes (blinking LED). Referring to Fig. 2.12, we can see two different scene conditions: (a) Both motion and illumination changes and (b) only illumination changes. Similar to previous observations, in Fig. 2.12a, we observe that accretion/deletions dominate over illumination changes in the uncertainty estimation. In the case of both illumination change and motion occurring (Fig. 2.12a), we observe high uncertainty due to both but it is governed by motion primarily. In the case of only illumination (no or minimum motion), we do observe high uncertainty as one would expect (Fig. 2.12b). In this scenario, our control policy would consider this as an obstacle for further processing.

We also evaluate our uncertainty estimation in the case of black-box attacks from [81] (Refer Fig. 2.11). We notice that the attack 'patches' do not hinder the performance of the uncertainty estimation, clearly showing that our networks trained with uncertainty are robust to black-box attacks.

From a logical perspective, high uncertainty and the solution of a task are not necessary and sufficient for each other. As an example, let's take the task of detecting an independently moving object (IMO). If P is the proposition, where P = 'There is a high uncertainty in this region' and Q is the proposition such that Q = 'There is an IMO there', then $Q \implies P$ but $P \not\Longrightarrow Q$. We could have high uncertainty because of a blinking light, for example, or because there is an object nearby. So, in principle, a system based on uncertainty estimation will never miss an IMO – it may think, however, in some instances, that there is an IMO while there isn't one. This is the price that qualitative minimal perception has to pay. Thinking that there is an

IMO at a close-by location is not necessarily a bad thing for a minimal system, since close-by objects could be potential obstacles and should be avoided.

2.4.1 Limitations and Future Work

It is important for us to also discuss a few limitations of using *just* optical flow uncertainty for the tasks described before. Uncertainty can be high due to a multitude of reasons such as depth boundaries, color-flat regions, extreme brightness changes, and blinking lights. Generally, we observed that (more examples can be found in the supplementary material), depth boundaries or dynamic obstacles have higher uncertainties than other factors and can be utilized for navigation. Nevertheless, uncertainty from multiple sources (such as optical flow and surface normals) can be utilized to disambiguate amongst various scenarios and can be an interesting avenue for future work. Furthermore, using uncertainty might not present a complete solution but can act as a safety/attention mechanism that highlights the part of the image which needs more processing/attention. From a logical perspective, high uncertainty and the solution of a task are not necessary and sufficient for each other. As an example, let's take the task of detecting an independently moving object (IMO). If P is the proposition, where P = 'There is a high uncertainty in this region' and Q is the proposition such that Q = 'There is an IMO there', then $Q \implies P$ but $P \not\Longrightarrow Q$. We could have high uncertainty because of a blinking light, for example, or because there is an object nearby. So, in principle, a system based on uncertainty estimation will never miss an IMO – it may think, however, in some instances, that there is an IMO while there isn't one. This is the price that qualitative minimal perception has to pay.

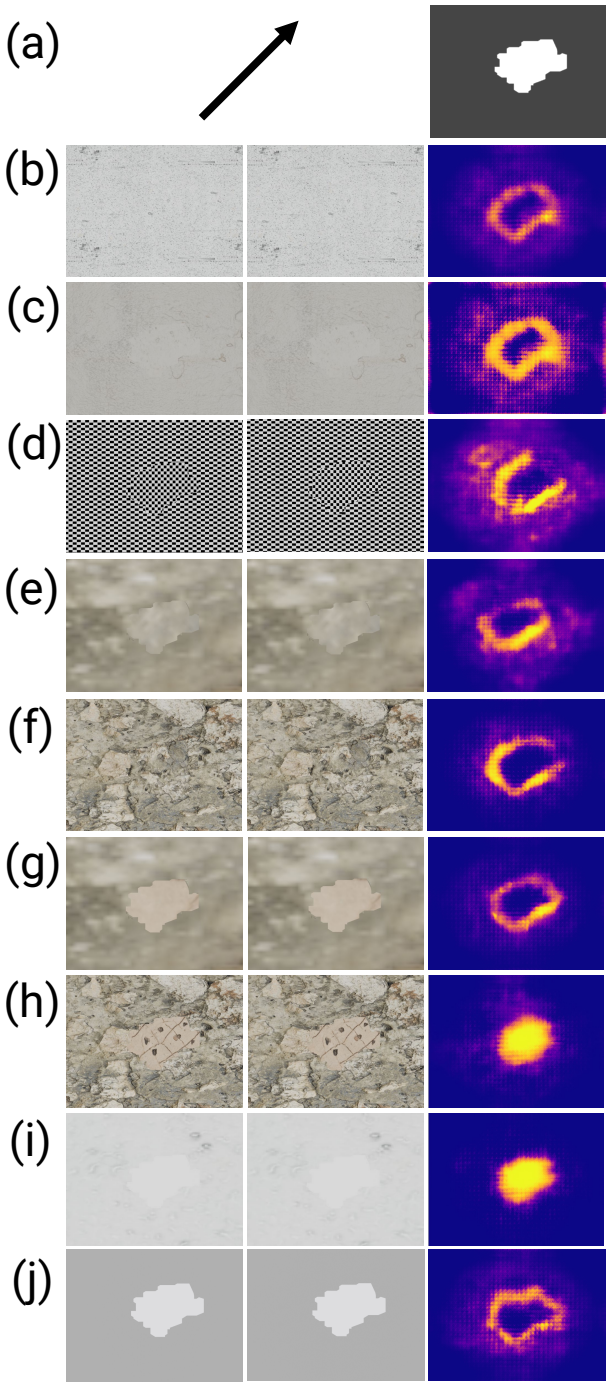

Figure 2.8: Uncertainty Estimation from a moving camera looking at an unknown-shaped gap. Fig. 2.9 shows the environmental setup. (a) shows the direction of camera motion and the ground truth mask: white is the background and grey is the foreground. (b)-(j) shows the pair of images and uncertainty (from left to right). Note that (d) shows uncertainty in a challenging/illusion scene with a checkerboard pattern. (b)-(f) scenes with the same texture in foreground and background; (g)-(i) scenes with different textures and (j) no texture in both foreground and background.

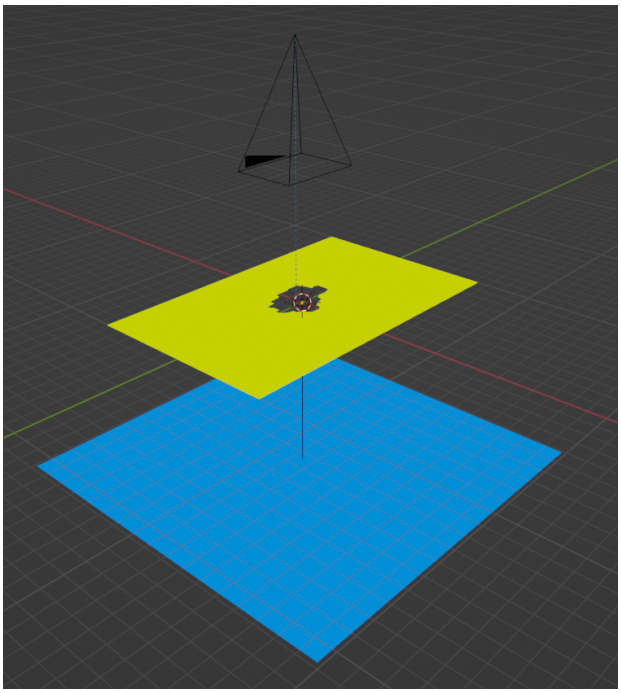

Figure 2.9: GapFlyt experiment setup in 3D for uncertainty estimation in different texture environments. The top pyramid represents the camera, the yellow plane represents the foreground with a gap and the blue plane represents the background.

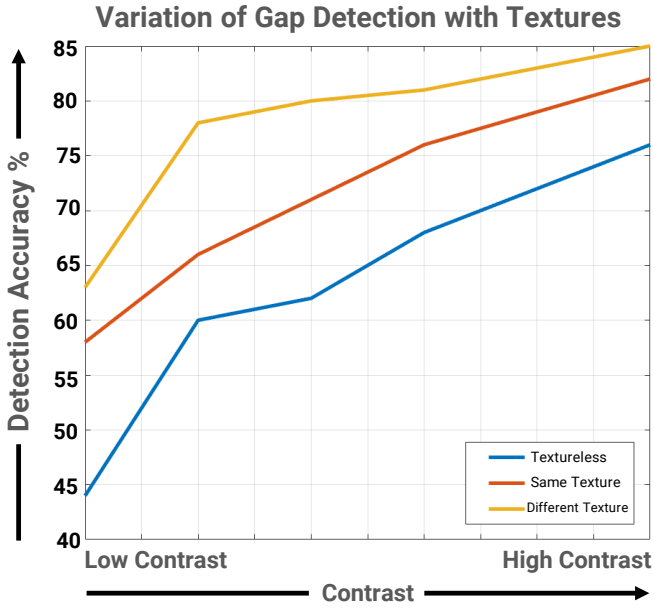

Figure 2.10: The plot represents how the detection accuracy of the gap varies with the texture resolution and contrast.

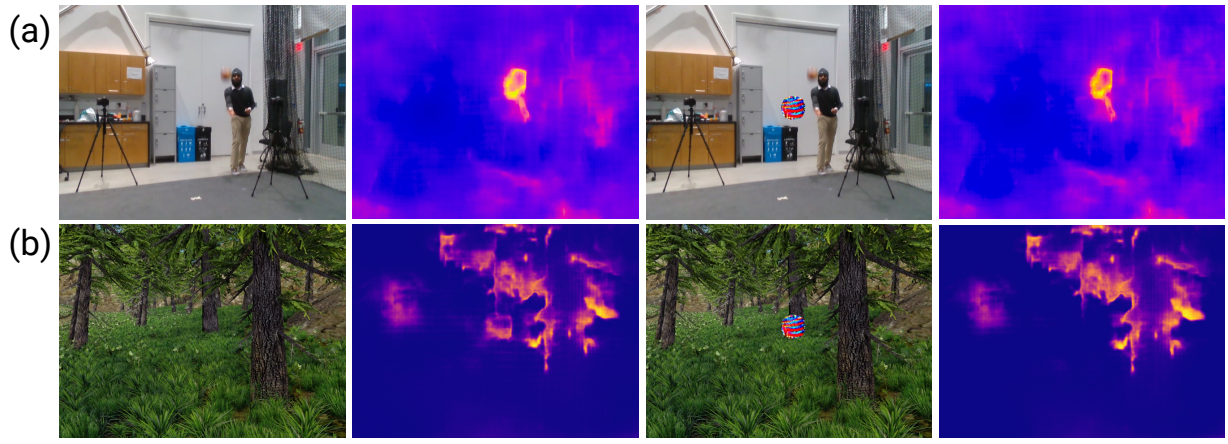

Figure 2.11: From left to right: Input Image, Uncertainty Estimation, Input Image with Black-Box patch, Uncertainty on the patched image. We show that uncertainty behavior is not affected by the optical flow attack patch in both cases of (a) obstacle avoidance and (b) navigation.

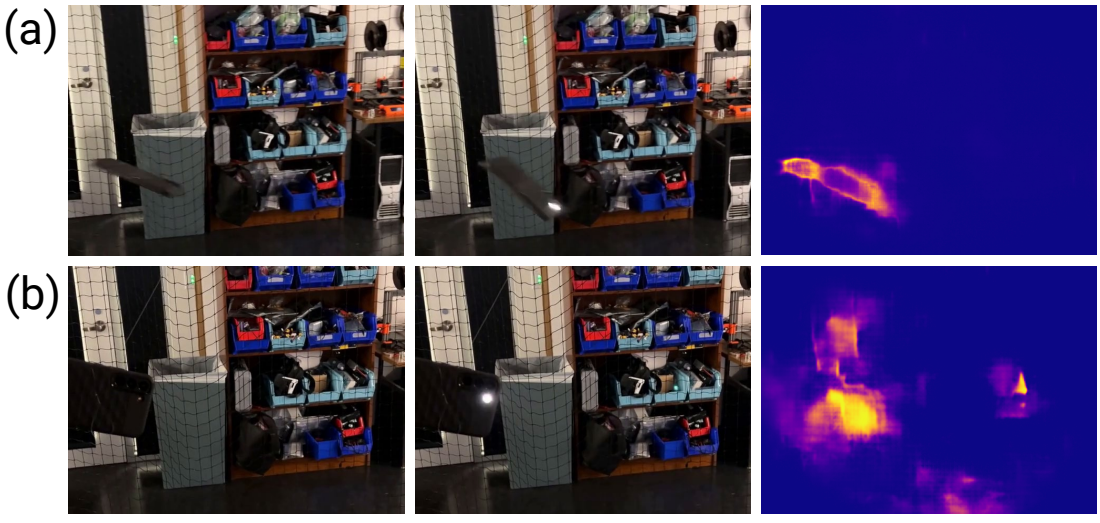

Figure 2.12: From left to right: Pair of consecutive input images and uncertainty. (a) shows uncertainty with both motion and illumination changes and (b) shows only illumination changes.

Thinking that there is an IMO at a close-by location is not necessarily a bad thing for a minimal system, since close-by objects could be potential obstacles and should be avoided.

2.5 Conclusion

Several interesting avenues for future work arise in light of the nascent stage and boundless potential of this research. First and foremost, an intriguing direction would involve analyzing the aleatoric uncertainty of commonly used robotics sensors, such as inertial sensors, LIDARs, SONARs, and others. This analysis could potentially offer a novel approach to sensor fusion by leveraging informational cues, as opposed to relying solely on raw uncertainty values through Bayesian formulations and related techniques. Additionally, uncovering latent informational cues may lead to the discovery of efficient and non-trivial solutions for addressing common robotics problems in a unified manner.

While these analysis and experimental results have demonstrated the applicability of Υ in solving typical robotics problems, challenges remain in the development of a task planner capable of seamlessly switching between the aforementioned tasks and effectively disambiguating different scenarios in real-world deployments. Moreover, employing uncertainty principles with traditional RGB cameras in low-light environments would significantly impact obstacle detection performance. Exploring uncertainty principles tailored for High Dynamic Range (HDR) or event cameras represents an intriguing avenue for future investigation.

Overall, the method presented adopts a baby-steps approach through the introduction of a generalized uncertainty formulation. This formulation enables the unconventional resolution of

common robotics problems. It is anticipated that this approach will expand the horizons of robot autonomy, enabling advancements in sizes previously deemed unattainable. The utilization of Υ is expected to facilitate parsimonious and minimal solutions in this regard.

This page intentionally left blank.

Chapter 3: Novel Object Segmentation With Minimum Knowledge

Recent advances in object segmentation have shown that deep neural networks achieve excellent results in segmenting objects of specific classes in color and depth images. However, the performance of these networks is dependent on the training data, particularly the number of classes and objects used, which limits their ability to generalize to unseen objects or *zero-shot samples*. Moreover, traditional object segmentation approaches using image frames heavily rely on recognition and pattern-matching cues. In contrast, our approach leverages the inherent *active* nature of robots and their capacity to *interact* with the environment to introduce additional geometric constraints that aid in segmenting zero-shot samples.

This chapter introduces a novel framework called '*NudgeSeg*' for segmenting unknown objects in cluttered scenes using a monochrome monocular camera. The framework achieves segmentation by iteratively nudging the objects and capturing additional motion cues at each step. The obtained motion cues are utilized to improve the accuracy of segmentation masks. The effectiveness of the proposed approach is demonstrated through the successful segmentation of novel objects in diverse cluttered scenes. A comprehensive evaluation is conducted, comparing the performance of the NudgeSeg framework with existing image and motion segmentation methods. The results showcase an impressive average detection rate exceeding 86% for zero-shot samples.

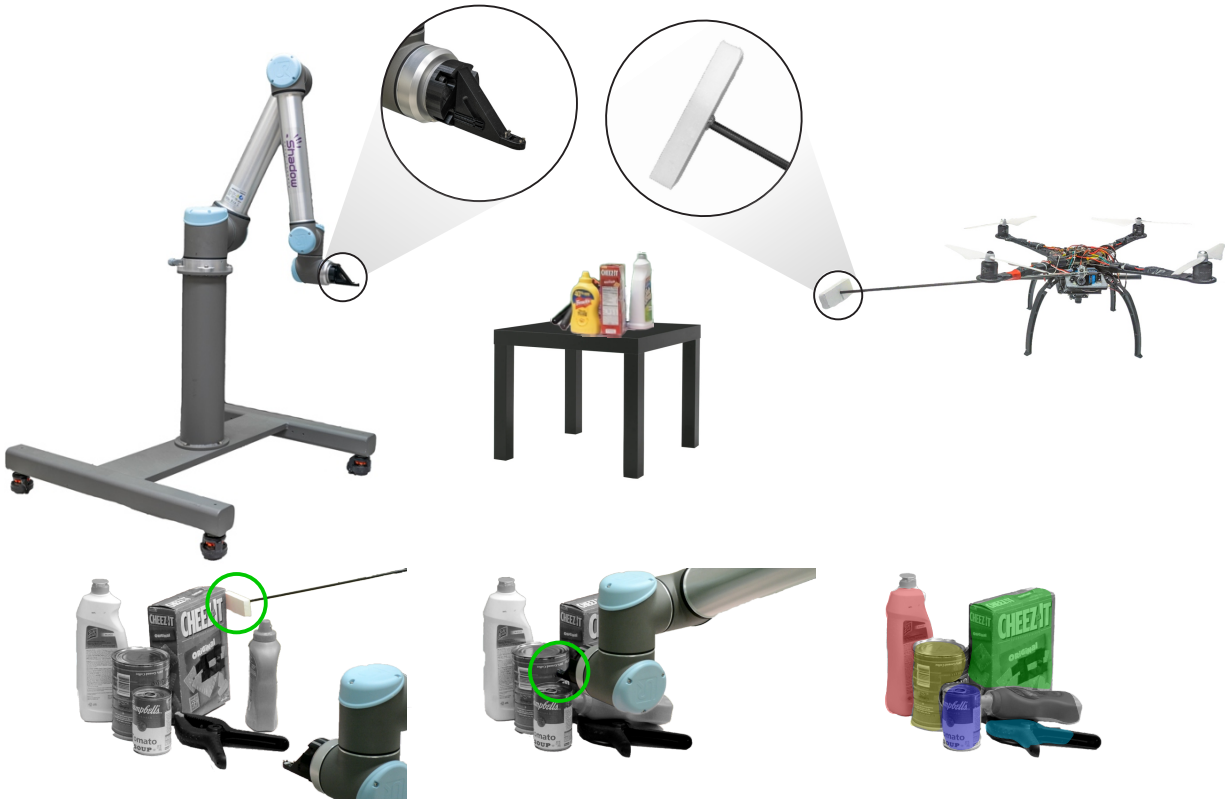

Figure 3.1: Top row: Robots (UR-10 and a quadrotor) used to physically interact (or nudge) with the objects to get motion cues for segmenting objects in a clutter. Bottom row (left to right): Initial Configuration of a cluttered scene and the first nudge being invoked, final nudge is invoked, final Segmentation of the cluttered scene. Green circles show the nudge operation. *All the images in this chapter are best viewed in color at 200% zoom on a computer screen.*

The supplementary video is available at `http://prg.cs.umd.edu/NudgeSeg`.

3.1 Background

The synergy between *Perception* and *Interaction* remains underutilized in robotics, despite the ability of most robots to either move or manipulate their bodies to gather more information. Effectively capturing this information in a "smart" manner can simplify the underlying problem, a strategy frequently employed by nature's creations. Even the simplest biological organisms rely on this active-interactive synergy to tackle complex problems [15]. The pioneers of robotics established formal foundations that encompassed the elegance of action-interaction-perception

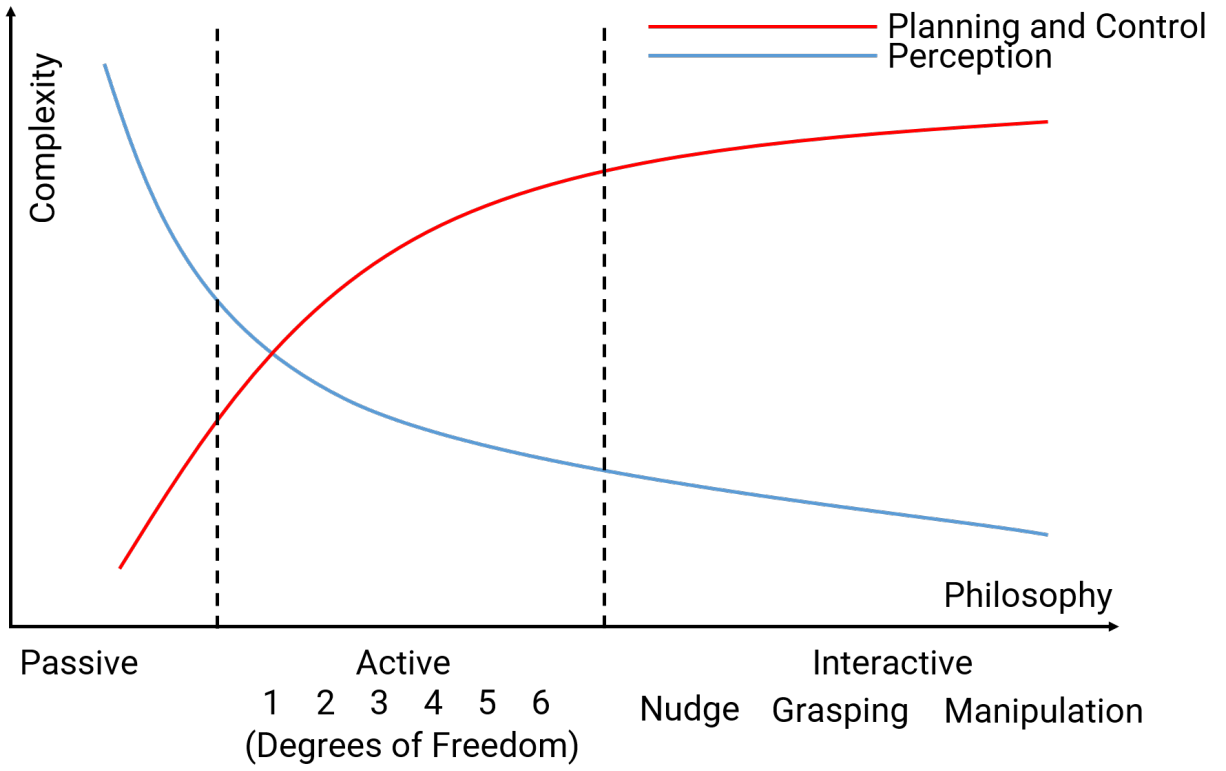

Figure 3.2: A conceptual graph of variation of complexity in perception, planning, and control with task philosophy. As a keen observation, the algorithmic complexity decreases with an increase in the manipulator motion.

loops. By incorporating exploration and/or interaction, the computational requirements for a given task can be supplemented, obtaining information that simplifies the perception problem. Fig. 3.2 presents a representative plot illustrating how the complexity of perception, planning, and control varies with different design philosophies. It is evident that the degree of activeness, as indicated by the number of available degrees of freedom, affects the level of interactivity. The range of interactivity can span from nudging or grasping (pick-up) to complex manipulation tasks (e.g., screwing a lid). The choice of task philosophy allows for a trade-off between the complexity of planning and control and that of perception.

In this chapter, a framework is presented that captures the elegance necessary to address the problem of segmentation of objects of unknown shape and type (also referred to as *zero-shot*

objects). The proposed method serves as an initial guess for learning new objects on a robot through interaction, resembling the process by which infants learn about novel objects. To the best of our knowledge, this work represents the first attempt to tackle the challenge of zero-shot object segmentation from clutter through iterative interaction using a grayscale camera. The problem statement is formally described, followed by a list of key contributions.

3.1.1 Problem Formulation and Contributions

The question we address is as follows: *How can we segment objects of unknown shape and type (zero-shot samples) from a cluttered scene by interacting (nudging) with the objects using a monochrome monocular camera?*

The key contributions of this work are given below:

- We propose an active-interactive nudging framework called *NudgeSeg* for segmenting zero-shot objects from clutter using a monochrome monocular camera.
- Conceptualization of uncertainty in optical flow to find the object clutter pile.
- Extensive real-world experiments on different robots including a quadrotor and a robotic arm to show that our framework is agnostic to the robot's structure.

We make the following explicit assumptions in our work:

- The surface on which the cluttered object pile is located is planar.
- All the individual objects in the clutter are non-deformable.

Instance and Semantic Segmentation: In the past few years, there has been a resurgence of interest in instance and semantic segmentation in the context of deep learning. The initial

approach proposed by Long *et al.* [83] was the first to utilize fully Convolutional Neural Networks (CNNs) for semantic segmentation. Subsequently, CNN meta-architecture approaches [18] achieved top rankings in recent object segmentation challenges. Another notable method, TensorMask [84], employed a sliding-window network design to generate precise high-resolution segmentation masks. More recently, a region-based segmentation model was introduced in [19], which further improved the accuracy of region-based approaches by producing masks with intricate details.

Motion Segmentation: The problem of segmenting motion into clusters that exhibit similar 3D motion has been extensively investigated in previous studies [85, 86]. Subsequently, several methods employing an expectation maximization approach for segmentation [87,88] have been introduced to further enhance the results for sequences captured in diverse environments. Recent works [89–91] have addressed more intricate scenarios by incorporating optical flow inputs, where the motion angle serves as the primary motion information for segmentation. Moreover, significant advancements have been made in the field of motion segmentation using event cameras over the past decade. These approaches based on event cameras commonly employ event alignment techniques and expectation-maximization schemes [2,92] to derive segmentation masks for independently moving objects.

Active and Interactive Approaches: The first conceptualization of Active vision was presented in the works of Aloimonos *et al.* [5] and Bajcsy *et al.* [6], where the key concept involved moving the robot in an exploratory manner to gather more information for the task at hand. An interactive approach, which involves interaction between the agent and its environment, was formally defined in [93] and serves as a complementary counterpart to active approaches. For robots with limited computational resources, prioritizing activeness and interactiveness becomes

crucial to gather more information and simplifying the perception problem. In other words, by exploring more, it is possible to trade off the number of required sensors or the computational complexity needed to solve a given task. The effectiveness of such an approach has been demonstrated in clearing or segmenting piles of unknown objects using manipulators [94–97] with the utilization of depth and/or stereo cameras. Similar concepts of interaction have also been employed to infer object properties, such as shape and weight, using only haptic feedback [98]. Recently, the concept of learning to segment by randomly grasping objects in a self-supervised manner using color images was introduced by [99].

This contribution is derived from utilizing iterative nudging of objects to generate motion cues for performing motion segmentation on previously unseen objects.

The proposed *NudgeSeg* segmentation framework is described in detail in Sec. 3.2, with subsections providing an explanation for each step of the process. The evaluation methodology is presented in Sec. 3.3, including comprehensive comparisons with other state-of-the-art segmentation methods and detailed analysis. Discussion and potential future directions are provided in Sec.

3.2 NudgeSeg Framework

The framework employed in this study incorporates active and interactive perception concepts for its various components. The initial phase applies the principles of active perception, wherein the camera is repositioned to generate a hypothesis regarding the location of the "object pile" through foreground-background segmentation. Subsequently, the interactive perception approach is employed to iteratively manipulate objects in order to acquire additional information

for refining the segmentation hypothesis. A comprehensive explanation of both components follows.

3.2.1 Active perception in NudgeSeg

As previously discussed, the preliminary step for object segmentation involves the task of locating the pile of objects. Since the robot(s) does not possess a depth sensor, obtaining the segmentation of the object pile is not straightforward. Consequently, the approach employed involves leveraging the robot's mobility to generate a function that is correlated with depth by moving the camera.

The image captured at time index i is denoted as $\mathcal{I}i$. The dense pixel association matrix, also known as optical flow [100], is represented as $\dot{p}ij$ between frames i and j. It should be noted that these optical flow equations employ the pinhole camera projection model.

The correlation between the foreground and background boundaries is associated with the level of occlusion experienced by a pixel between two frames. This occlusion is inversely proportional to the condition number κ of the optical flow estimation problem. While obtaining κ directly is not feasible, we can derive its dual quantity, namely the estimated uncertainty ρ. Specifically, the utilization of the *Heteroscedastic Aleatoric uncertainty* [29] allows us to obtain ρ without the significant additional computational overhead. The process of obtaining ρ from a network is explained in detail in Section 3.2.4.

Once ρ is obtained between two frames, the next step involves determining the *first nudge direction*. This is achieved by performing morphological operations on ρ to extract the blobs that constitute the object pile $\mathcal{O}_k$ (where k represents the blob index). Subsequently, the convex hull

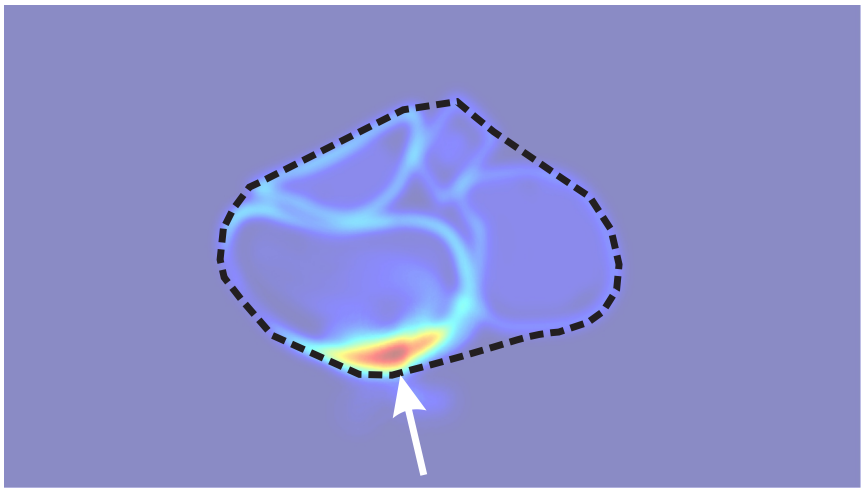

Figure 3.3: First nudge policy using uncertainty in optical flow. Hotter colors represent higher uncertainty. The dashed line represents the convex hull of the cluttered scene and the arrow represents the direction of the first nudge at point $\mathcal{N}_1$.

of this set $\mathcal{C}(\mathcal{O})$ is computed. The first poke direction is obtained through the following two-step process:

Firstly, the blob with the highest average uncertainty value is identified, denoted as $\text{argmax}_k \mathbb{E}(\rho(\mathcal{O}_k))$. This blob is chosen due to its relatively lower noise in ρ.

Secondly, the first nudge point $\mathcal{N}_1$ is calculated as the closest point to the centroid of the previously identified blob k'.$\mathcal{N}_1 = \text{argmax}_{\mathbf{x}} \|\mathcal{C}(\mathcal{O}_\kappa)_{\mathbf{x}} - \overline{(\mathcal{O}_\kappa)}\|_2$(3.1)

Here, $\mathbf{x}$ is used to index each point in $\mathcal{C}(\mathcal{O}_k)$, and $\overline{A}$ represents the centroid of A. The objects at $\mathcal{N}_1$ are then adjusted using the *nudging tool* (See Fig. 5.1). Fig. 3.4(a) provides a summary of the active segmentation part.

3.2.2 Interactive perception in NudgeSeg

The interactive perception part is based on the clustering of rigid parts of the scene using optical flow $\dot{p}$ (refer to Sec. 3.2.4 for details on obtaining $\dot{p}$). A complete segmentation of the scene is not typically achieved by clustering optical flow from a single nudge. To address this,

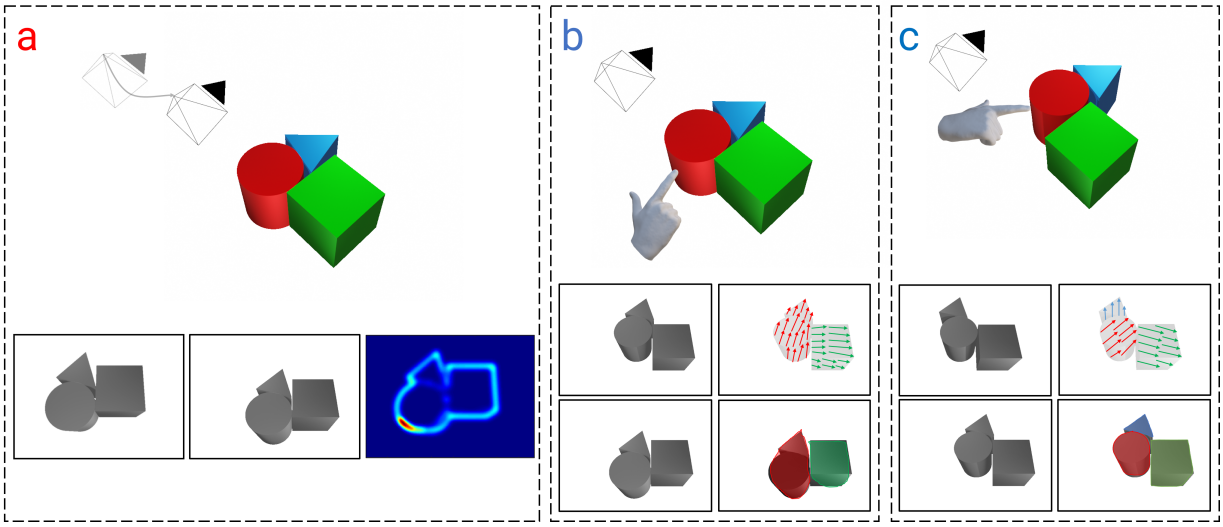

Figure 3.4: (a) *Active* perception in *NudgeSeg* framework. The top row shows the movement of the camera. The bottom row shows the image inputs and uncertainty ρ. (b) and (c) *Interactive* perception in *NudgeSeg* framework. The top row shows the object nudging. The bottom row shows the input images (before and after the nudge), optical flow representation, and segmentation hypothesis where colors indicate cluster membership.

an iterative nudging strategy is proposed, which relies on the statistics of the current cluster hypothesis (see Sec. 3.2.2.1 for more details).

To split the current scene based on optical flow (before and after the first nudge) into multiple segments, Density-based Spatial Clustering of Applications with Noise (DBSCAN) [101] is employed, as it does not depend on the input of the number of clusters. The following criteria are utilized to assign two points $\mathbf{X}$ and $\mathbf{Y}$ to the same cluster:

$$\mathbf{X} - \mathbf{Y}_2 < \tau_d \tag{3.2}$$

$$\mathcal{M}_{\mathbf{X}} - \mathcal{M}_{\mathbf{Y}2} < \tau_{\mathcal{M}} \tag{3.3}$$

$$\min\left(|\mathcal{A}_{\mathbf{X}} - \mathcal{A}_{\mathbf{Y}}|,\, 2\pi - |\mathcal{A}_{\mathbf{X}} - \mathcal{A}_{\mathbf{Y}}|\right) \leq \tau_{\mathcal{A}} \tag{3.4}$$

Here, $\tau_d, \tau_{\mathcal{M}}$ and $\tau_{\mathcal{A}}$ represent user-defined thresholds. The input to DBSCAN consists of 4-dimensional data comprising image coordinates, optical flow magnitude, and direction: $\begin{bmatrix} \mathbf{x} & \mathcal{M} & \mathcal{A} \end{bmatrix}^T$. Points that do not belong to any cluster are eliminated as noise points due to their low density. The iterative removal of these points continues until the termination criterion, as described in Section 3.2.3, is reached. The output of the segmentation hypothesis at each iteration (referred to as time index i) is denoted as $\mathcal{H}_i$.

3.2.2.1 Nudging Policy

Firstly, the covariance matrix Σ_i^k of the optical flow $\dot{p}$ is computed for all clusters of the hypothesis $\mathcal{H}_i^k$ (k being the cluster index), and it is given by

$$\Sigma_i^k = \mathbb{E}\left(\left(\dot{p}_i^k - \mathbb{E}\left(\dot{p}_i^k\right)\right)\left(\dot{p}_i^k - \mathbb{E}\left(\dot{p}_i^k\right)\right)^T\right) \tag{3.5}$$

The Eigenvectors and Eigenvalues for each cluster are determined using Σ_i^k, where $v_{\text{min},i}^k$, $v_{\text{max},i}^k$, $\lambda_{\text{min},i}^k$, and $\lambda_{\text{max},i}^k$ are obtained. Subsequently, the cluster k with the second largest condition number $\kappa_* = {|\lambda_{\text{max}}|}/{|\lambda_{\text{min}}|}$ is selected (excluding the cluster associated with the highest motion information, i.e., best κ, to avoid nudging the object excessively). The nudging direction is determined based on the Eigenvector direction v_{min} of the chosen cluster. However, if $\kappa_* \leq \tau$, the cluster with the largest κ is nudged instead, as the motion cue quality corresponding to κ_* is considered noisy. The nudge is performed using a simple Proportional-Integrative-Differential (PID) controller, which serves as the nudge point. Following each nudge i, the mask is propagated

to the new frame j by warping it using optical flow. The resulting propagated mask is denoted as $\hat{\mathcal{H}}$.

3.2.2.2 Mask Refinement

Finally, for each nudge step, once the masks are propagated, the robot aligns itself to initiate the next nudge for new motion cues. After invoking the next nudge, a new $\mathcal{H}$ is generated, which may or may not overlap with the previously propagated masks. To determine the updated masks, the union of ${}^{k}\mathcal{H}_j$ and ${}^{k}\hat{\mathcal{H}}_j$ is computed. Please refer to Algorithm 1 for detailed information on mask refinement. Figure 3.4(b) and (c) illustrate a single step (a single nudge) of the *interactive* segmentation part.

3.2.3 Verification and Termination

If the mean Intersection over Union (IoU) between $\mathcal{H}i$ and $\mathcal{H}i + n$ has not exceeded a predefined threshold, the verification step is executed. During this step, each segment is independently nudged towards its geometric center to evaluate if the object further splits. A single nudge is performed for each cluster, and if any segment splits into multiple parts, the procedure described in Section 3.2.2 is applied again for those clusters. If no further splitting occurs, the process is terminated.

3.2.4 Network Details

We obtain optical flow $\dot{p}$ using a Convolutional Neural Network (CNN) based on PWC-Net [69]. We use the multi-scale (L scales) training loss given below:

Algorithm 1: Segment Propagation And Mask Refinement

Data: Optical Flow $\dot{p}_i^k$, Segmentation Hypothesis of k masks $\{\mathcal{H}_i\}$ in i^{th} frame.
Result: Updated Segmentation Hypothesis $\{\mathcal{H}_j\}$ in j^{th} frame.

if ${}^k\mathcal{H}_j \cap {}^k\hat{\mathcal{H}}_j > \tau_{\mathcal{H}}$ **then**
  ${}^k\hat{\mathcal{H}}_j \rightarrow {}^{k+1}\mathcal{H}_j$ `Update Masks;` ${}^k\mathcal{H}_j \rightarrow \max({}^k\hat{\mathcal{H}}_j, {}^k\mathcal{H}_j) - ({}^k\hat{\mathcal{H}}_j \cap {}^k\mathcal{H}_j)$;
else
  ${}^k\hat{\mathcal{H}}_j \rightarrow {}^{k+1}\mathcal{H}_j$ `Create New` $\mathcal{H}$`;`
end

$$\mathcal{D}\left(\hat{\dot{p}}_l, \tilde{\dot{p}}_l\right) = \sum_{\forall l} \alpha_l \mathbb{E}_{\mathbf{x}} \left(\| \hat{\dot{p}}_l(\mathbf{x}) - \tilde{\dot{p}}_l(\mathbf{x}) \|_1 \right) \tag{3.6}$$

The variables used above are defined as follows: l represents the pyramid level, $\hat{\dot{p}}_l$ represents the ground truth optical flow at level l, and $\tilde{\dot{p}}_l$ represents the predicted optical flow at level l. The parameter α_l corresponds to the weighting parameter for the l-th level. For more detailed information regarding the network, please refer to [69].

Finally, in order to obtain the *Heteroscedastic Aleatoric uncertainty* [29], the network's output channels are modified from two to four, denoted as $\tilde{\hat{p}}$ (two channels). The predicted uncertainty ρ is obtained in a self-supervised manner, with two channels corresponding to each channel of $\tilde{\hat{p}}$. The training process involves utilizing the following final loss function.

$$\mathcal{L} = \sum_{\forall l} \alpha_l \mathbb{E}_{\mathbf{x}} \left(\frac{\| \hat{\dot{p}}_l(\mathbf{x}) - \tilde{\dot{p}}_l(\mathbf{x}) \|_1}{\log_e\left(1 + e^{(\rho_l(\mathbf{x}) + \epsilon)}\right)} \right) + \sum_{\forall l} \alpha_l \mathbb{E}_{\mathbf{x}} \left(\log_e\left(1 + e^{\rho_l(\mathbf{x})}\right) \right) \tag{3.7}$$

In this context, ϵ represents a regularization value utilized to prevent numerical instability and is specifically set to 10^{-3}. It is important to note that the uncertainty is acquired at each level l; however, the average values across levels, scaled by α_l, are employed as input for the framework. Please refer to the first row and second column of Fig. 3.6 to observe the uncertainty

outputs obtained from the network. The model is trained using the *ChairThingsMix* dataset [102]. It should be emphasized that the PWC-Net used in our experiments is neither retrained nor fine-tuned on any of the utilized data.

3.3 Experimental Results and Discussion

3.3.1 Description of robot platforms – Aerial Robot and UR10

The hardware setup comprises a Universal Robot UR10 and a custom-built quadrotor platform called PRGLabrador500α[1] (refer to Fig. 5.1). Both robots are equipped with a 3D-printer end-effector for performing nudging motions. PRGLabrador500α is constructed using an S500 frame, DJI F2312 960KV motors, and 9450$\times$2 propellers. The lower-level attitude and position hold controller utilizes the ArduCopter 4.0.6 firmware installed on the Holybro Kakute F7 flight controller. This setup is augmented with an optical flow sensor and a one-beam LIDAR serving as the altimeter source. Both robots are outfitted with a Leopard Imaging M021 camera featuring a 3mm lens, which captures monochrome image frames at a resolution of 800×600 px and a frame rate of 30 frames/sec. These frames are used in our segmentation framework. All higher-level navigational commands are transmitted by the companion computer (NVIDIA Jetson TX2 running Linux for Tegra®) to control both robots.

[1] https://github.com/prgumd/PRGFlyt/wiki

3.3.2 Quantitative Evaluation

3.3.2.1 Evaluation Sequences

The *NudgeSeg* framework is tested on four sequences of objects. For the evaluation of the framework, the results are averaged over 25 trial runs for each sequence.

For each iteration of an evaluation sequence, N_i objects are randomly selected from a given range of objects N ($N_i \leq N$) and positioned on a table in a randomly generated configuration space. Let $\mathcal{N}$ represent the total number of nudges required to solve the segmentation task for a specific configuration, and $\mathcal{N}_{\text{avg}}$ denote the average number of nudges across all trials in a sequence. Please refer to Table 3.1 for detailed information regarding the sequences, and Fig. 3.5 for a visual representation of the sequences. It is important to note that the objects in the `GrassMoss` and `YCB-attached` sequences (Fig. 3.5 (a) and (d)) consist of adversarial samples where objects have been permanently "glued" together.

3.3.2.2 State-of-the-art Methods

The results are compared with three state-of-the-art methods: 0-MMS [2], Mask-RCNN [18], and PointRend [19].

The 0-MMS approach (0-MMS [2]) employs an event camera, which differs from a classical camera, to segment moving objects. Event cameras provide asynchronous log intensity differences resulting from relative motion, with a latency in the microsecond range. This type of sensor generates sparse events with high temporal resolution and relies on significant object displacement between time intervals. To manipulate the event data, inducing large motion in the

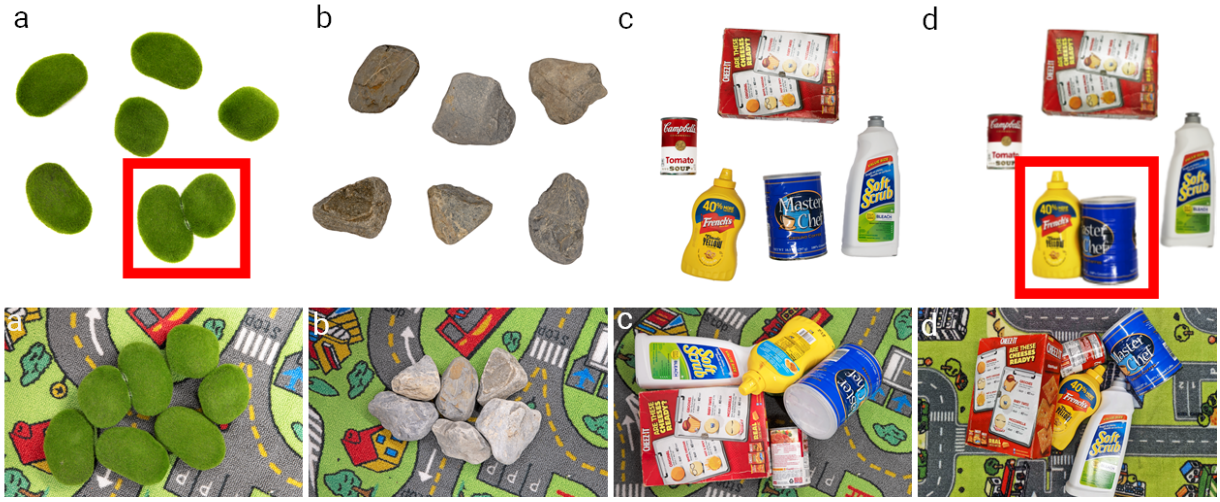

Figure 3.5: Top Row: Sample objects used in Table 3.1 as the evaluation sequences. Bottom Row: Sample cluttered scene for each sequence.

Table 3.1: Description of Evaluation Sequences.

Sequence Name	N	$\mathcal{N}_{\text{avg}}$	Reference Fig.
`GrassMoss`	5-8	5.7	3.5a
`Rocks`	5-7	5.2	3.5b
`YCB`	5-9	6.3	3.5c
`YCB-attached`	4-8	4.8	3.5d

objects was achieved by a single large nudge, as described in the original work.

Furthermore, a comparison is made with single-frame passive segmentation methods, namely Mask-RCNN and PointRend, which were trained on the MS-COCO dataset's `train2017` subset (approximately 123K images). It should be noted that the YCB object sequences [17] shown in Figs. 3.5c and 3.5d are a subset of the classes present in the MS-COCO dataset. However, the sequences depicted in Figs. 3.5a and 3.5b consist of zero-shot samples, representing classes that are not included in the MS-COCO dataset.

3.3.2.3 Evaluation Metrics

Before evaluating and comparing the performance of *NudgeSeg* with the aforementioned methods, it is necessary to define the evaluation metrics used. Intersection over Union (IoU) is a commonly used and standardized evaluation criterion for comparing segmentation methods. The IoU is calculated as follows:

$$IoU = (\mathcal{D} \cap \mathcal{G})/(\mathcal{D} \cup \mathcal{G}) \tag{3.8}$$

where predicted mask is denoted as $\mathcal{D}$ and the ground truth mask $\mathcal{G}$. The Detection Rate (DR) is defined based on the Intersection over Union (IoU) for each cluster as follows:

$$\text{Success} := IoU \geq \tau;\ \text{DR} = \frac{\text{Num. Success}}{\text{Num. Trials}} \tag{3.9}$$

DR is defined at two accuracy levels with τ of 0.5 and 0.75, denoted as DR50 and DR75 respectively. For passive segmentation methods, IoU_i, DR$50, i$, and DR$75, i$ are computed for an image after each nudge (indexed as i) along with the initial configuration. The final IoU, DR50, and DR75 for each trial (which includes multiple nudges) are determined as the highest accuracy among all time indexes for a fair comparison. Finally, the average results across trials are computed for each scenario as previously mentioned.

3.3.2.4 Observations

Now, the comparison of *NudgeSeg* with other state-of-the-art segmentation methods using the aforementioned evaluation metrics will be presented. The performance of the method on

different sequences and its comparison with different methods are shown in Table 3.2 (Refer 3.3.2.1). The segmentation methods of PointRend and Mask-RCNN are compared on both RGB and monochrome images. As *NudgeSeg* incorporates more motion cues through the iterative nudging of the cluttered scene, it outperforms 0-MMS, which relies on motion cues from a single large nudge. Moreover, the active segmentation approaches, namely the *NudgeSeg* framework, and 0-MMS [2], demonstrate significantly better performance than the passive segmentation methods, PointRend and Mask-RCNN, particularly in zero-shot samples. This emphasizes the effectiveness of active methods, attributed to the iterative segmentation propagation and the utilization of sensor-motor loops for targeted information gathering. The active method can be employed to train a passive method like Mask-RCNN or PointRend in a self-supervised manner, thereby emulating the memory capabilities observed in animals.

The generalization performance of most passive segmentation methods is inferior to that of active methods, although their masks exhibit sharper boundaries during correct segmentation due to the inclusion of memory information. To capture this characteristic, a new metric named IoU_s is introduced, which represents the Intersection over Union for successfully detected objects.

As mentioned previously, segmentation results are obtained after every nudge, and the highest accuracy results are selected. Subsequently, the average results across trials are computed for each scenario, as previously mentioned.

Based on the results shown in Table 3.2(d), it can be observed that the masks generated by Mask-RCNN and PointRend exhibit sharper boundaries (better alignment with the ground truth) when the adversarial objects are omitted. Moreover, these masks gradually approach the performance of the active methods as the IoU values, which are averaged to determine success, increase (or decrease). The incorporation of memory information is clearly seen to have a

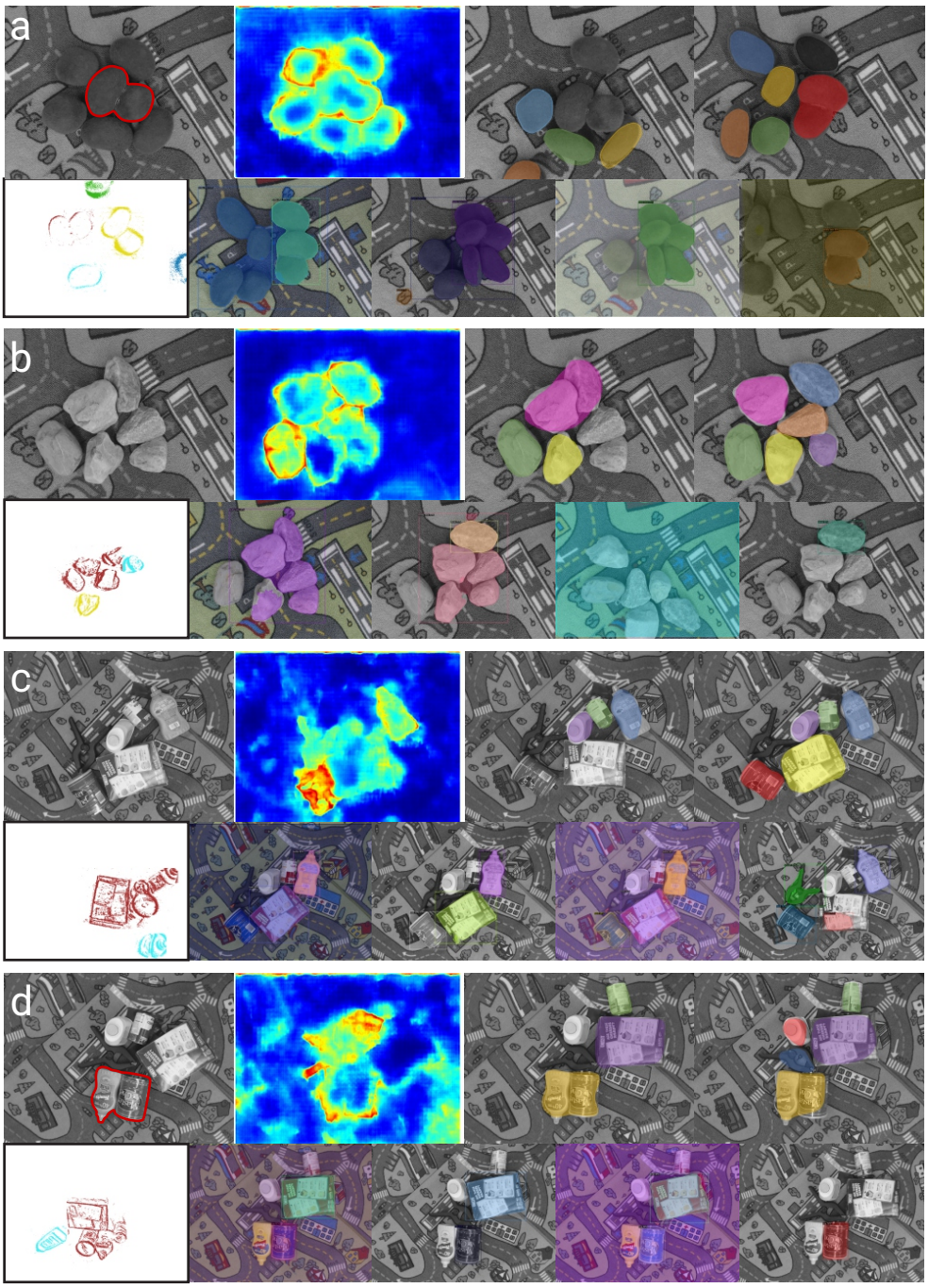

Figure 3.6: For each sub-figure: First row (From left to right): Sample monochrome input image, Uncertainty in optical flow ρ, Segmentation hypothesis after first nudge, Final segmentation masks. Second row: (From left to right): Outputs of 0-MMS [2], PointRend (color input), PointRend (mono input), Mask-RCNN (color input), Mask-RCNN (mono input). Note that in (a) and (d), the objects highlighted with a red boundary in the top left image of the respective sequences are 'glued' together and are considered to be adversarial samples. *This image is viewed best in color at 400% zoom on a computer screen.*

Table 3.2: Evaluation with different segmentation methods for multiple sequences.

Sequence	*NudgeSeg*	0-MMS [2]	PointRend [19]		Mask-RCNN [18]	
Sensor Used	Mono	Event	Color	Mono	Color	Mono
			(a) IoU ↑			
GrassMoss	0.82	0.77	0.14	0.14	0.12	0.10
Rocks	0.87	0.63	0.16	0.17	0.11	0.13
YCB	0.68	0.58	0.44	0.42	0.36	0.39
YCB-attached	0.70	0.61	0.38	0.35	0.32	0.32
			(b) $DR_{50}(\%)$ ↑			
GrassMoss	89.6	64.3	11.1	9.4	8.2	6.7
Rocks	94.1	30.2	14.7	14.1	9.5	7.7
YCB	80.9	28.4	42.4	40.6	36.1	39.3
YCB-attached	82.0	32.2	37.7	31.4	32.1	34.9
			(c) $DR_{75}(\%)$ ↑			
GrassMoss	86.3	64.4	10.1	9.9	7.4	6.3
Rocks	91.1	30.9	13.5	13.2	7.2	6.1
YCB	74.5	42.3	38.8	35.1	32.9	34.2
YCB-attached	76.2	32.4	33.1	27.0	27.2	29.3
			(d) IoU_s ↑ (for DR_{50})			
GrassMoss	**0.88**	0.84	0.83 (**0.91**)	0.80 (0.87)	0.81 (0.90)	0.79 (0.88)
Rocks	0.89	0.80	**0.97**	0.95	0.91	0.88
YCB	0.77	0.70	**0.96**	0.88	0.92	0.85
YCB-attached	0.75	0.72	**0.79**	0.77	0.75	0.72

Note: $(\cdot)$ represents the IoU_s after the removal of adversarial examples in `GrassMoss` sequence.

significant positive impact on the segmentation performance, particularly along the edges, which are crucial for grasping. Additionally, it should be noted that in cases where no color information is provided as input, the results obtained by Mask-RCNN and PointRend are slightly inferior to those of the active methods, indicating that these networks rely on color boundaries for accurate segmentation.

Figure 3.7: Qualitative Results with (a) no error, $\epsilon_{\mathcal{A}} = 0$, $\epsilon_{\mathcal{M}} =$ (b) $\pm 5\%$, (c) $\pm 10\%$, (d) $\pm 20\%$, $\epsilon_{\mathcal{M}} = 0$, $\epsilon_{\mathcal{A}} =$ (e) $\pm 10^{\circ}$, (f) $\pm 20^{\circ}$, (g) $\pm 30^{\circ}$.

Table 3.3: Evaluation of `GrassMoss` sequence with different amount of errors in $\mathcal{A}$ and $\mathcal{M}$.

Err. Metric	No Error	$\epsilon_M = 5$	$\epsilon_M = 10$	$\epsilon_M = 20$	$\epsilon_A = 10^{\circ}$	$\epsilon_A = 20^{\circ}$	$\epsilon_A = 30^{\circ}$
$IoU \uparrow$	0.82	0.77	0.70	0.64	0.62	0.41	0.23
$\text{DR}_{50}(\%) \uparrow$	89	82	74	68	61	60	49
$\text{DR}_{75}(\%) \uparrow$	86	77	69	60	46	37	17

If the injected error is not stated explicitly, it is taken to be zero.

3.3.2.5 Analysis

For most robotics applications, the performance of the algorithm is also influenced by the quality of the camera sensor. Erroneous sensor behavior, such as missing data, false colors, poor contrast, motion blur, and low image resolution, can often cause robotics algorithms to fail. Such sensor errors can result in significant errors in fundamental properties like optical flow. To assess the limitations of the framework, error is artificially introduced in optical flow, separately in angle and magnitude. The framework's performance is then evaluated under these conditions. Uniform noise $\mathcal{U}(\epsilon)$ is added independently to both $\mathcal{A}$ and $\mathcal{M}$ (refer to 3.2.2 for definitions) at each pixel, where $[-\epsilon, \epsilon]$ represents the noise bound. The perturbed (noise-induced) magnitude and angle of the flow vectors are denoted as $\widetilde{\mathcal{M}}$ and $\widetilde{\mathcal{A}}$, respectively, and can be expressed as:

$$\widetilde{\mathcal{M}}_{\mathbf{x}} = \left(1 + \frac{\mathcal{U}(\epsilon_{\mathcal{M}})}{100}\right) \mathcal{M}_{\mathbf{x}} \tag{3.10}$$

$$\widetilde{\mathcal{A}}_{\mathbf{x}} = (1 + \mathcal{U}(\epsilon_{\mathcal{A}}))\, \mathcal{A}_{\mathbf{x}} \tag{3.11}$$

The framework is evaluated on different values of $\epsilon_{\mathcal{M}}$ and $\epsilon_{\mathcal{A}}$, specifically $0, 5, 10, 20\%$ and $0, 10, 20, 30^{\circ}$ respectively. The qualitative result in Fig. 3.7 demonstrates how the error in optical flow affects the first segmentation hypothesis $\mathcal{H}_1$. It should be noted that the flow error is introduced to the output of PWC-Net, which already has an angle error of approximately 5% compared to the ground truth optical flow. The performance in terms of intersection over union (IoU) in $\mathcal{H}_1$ shows a significant decrease when noise is added to $\mathcal{A}$ rather than $\mathcal{M}$. This indicates that the optical flow angle is more crucial for active segmentation methods compared to magnitude, and this distinction in evaluation is often overlooked in most optical flow studies.

The performance of the `GrassMoss` sequence with varying errors in $\mathcal{A}$ and $\mathcal{M}$ is presented in Table 3.3. A significant drop in DR_{75} is observed, decreasing from 86% to approximately 17% when there is a $\pm 30^{\circ}$ error in $\mathcal{A}$ (which is about 6 times the error compared to PWC-Net). It is evident that accurate computation of optical flow is crucial for active and interactive segmentation approaches. Therefore, the performance of interactive segmentation methods will be influenced by the optimization of neural networks through quantization or reduction of parameter count.

Passive segmentation methods are evaluated on the sample non-cluttered images depicted in Fig. 3.5 (top row). PointRend and Mask-RCNN exhibit similar performance in the `GrassMoss` and `Rocks` sequences, but they demonstrate significantly better results in the `YCB` sequence, achieving approximately 94% DR_{50} for both methods. This indicates that background clutter and object occlusions can have a substantial impact on segmentation performance.

3.4 Discussions

Considering the Size, Weight, Area, and Power (SWAP) constraints, the robot's sensor suite may include various sensors, including a depth camera, which plays a vital role in enabling three-dimensional nudging. In such scenarios, the determination of the appropriate nudge location can be computed using a collision-avoidance path planner.

However, the *where to nudge?* problem can be modeled using reinforcement learning to achieve a refined solution. Specifically, a reward function based on the success criterion of the *NudgeSeg* framework for delayed rewards can be envisioned. This approach facilitates the learning of more generalized optical flow in conjunction with segmentation, resulting in improved performance for zero-shot objects in a genuine robotics environment through the utilization of sensorimotor loops. In this context, the reinforcement agent predicts the optimal nudge location and its corresponding estimated reward, without consideration for the specific object class.

To further enhance the framework, a soft suction gripper can be employed, allowing for adaptation to diverse object morphologies. Specifically, the suction gripper can be utilized subsequent to each nudging step, enabling the retrieval of the unknown object from the cluttered pile, thereby facilitating the perception problem.

Through the integration of the aforementioned approaches, the active-interactive segmentation model is deemed as a promising direction aligning with the life-long learning paradigm [103].

3.5 Conclusions

An active-interactive philosophy is presented for segmenting unknown objects from a cluttered scene. The methodology involves the iterative process of 'nudging' the objects and relocating them to acquire additional motion cues. These motion cues, obtained as optical flow, are utilized to identify and refine the segmentation hypothesis at each step. The proposed approach solely relies on a monochrome monocular camera and exhibits superior performance compared to the current state-of-the-art object segmentation methods, particularly for zero-shot samples. The effectiveness of the approach is demonstrated through the successful segmentation of novel objects in diverse cluttered scenes, with comprehensive comparisons against passive and motion segmentation methods on two distinct mobile robots: a quadrotor and a robotic arm. The results showcase an impressive average detection rate exceeding 86% for zero-shot samples. It is firmly believed that this methodology can serve as an initial stride towards learning novel objects and facilitating the development of a genuine lifelong learning system.

This page intentionally left blank.

Chapter 4: WorldGen: Generative Simulator for Minimal Perception

In the era of deep learning, the performance of neural network models heavily relies on data. Generating large datasets poses challenges related to scalability, cost efficiency, and photorealism. To overcome the limitations of expensive and labor-intensive dataset collection and annotations, researchers have turned to computer-generated datasets. However, the lack of photorealism and limited availability of computer-aided data have constrained the accuracy of network predictions.

In this context, the WorldGen framework is introduced as an open-source solution for automatically generating numerous structured and unstructured 3D photorealistic scenes, such as city views, object collections, and object fragmentation. Along with scene generation, WorldGen provides rich ground truth annotation data. By functioning as a generative model, WorldGen grants users complete access and control over features like texture, object structure, motion, camera, and lens properties. This level of control enhances generalizability and mitigates data bias in neural networks.

The effectiveness of WorldGen is demonstrated through its evaluation in the context of deep optical flow. This tool holds the potential to facilitate future research in diverse domains related to robotics and computer vision by reducing the need for manual labor and the cost associated with acquiring rich and high-quality data.

4.1 Background

High-quality image data plays a crucial role in achieving accuracy in deep-learning models. In certain tasks, the quality of data outweighs the significance of neural architecture and hyperparameters. However, data collection, cleaning, and annotation have presented significant challenges, resulting in substantial expenses reaching millions of dollars. These challenges arise from issues related to data diversity, image quality, meticulous manual annotations, licensing concerns [104], and security considerations. Moreover, providing clear and concise instructions to human labelers for data labeling is a nontrivial task, as the requirements cannot always be easily explained. To address these issues, a universal framework is proposed to generate a vast number of 3D scenes representing different environments, accompanied by annotated images for various autonomous car, drone, and indoor robot tasks. This framework aims to enhance the efficiency of deep learning frameworks in terms of cost, speed, labor, and data variety by enabling them to gather photorealistic data more effectively.

On the other hand, synthetic data and simulators are commonly employed in research and practice but suffer from limitations in terms of scalability and photorealism. Although extensively used as benchmarks for evaluating quantities such as depth, optical flow, and segmentation due to their "perfect" ground truth annotations, most of these simulators lack support for realistic camera models or advanced rendering techniques for generating high-quality photorealistic images. Although a few approaches have demonstrated some success in reducing the generalization gap using synthetic data for geometric-based quantities like depth and optical flow compared to texture-centric tasks such as semantic segmentation [105, 106].

To further narrow the sim-to-real gap, a significant amount of synthetic scene generation

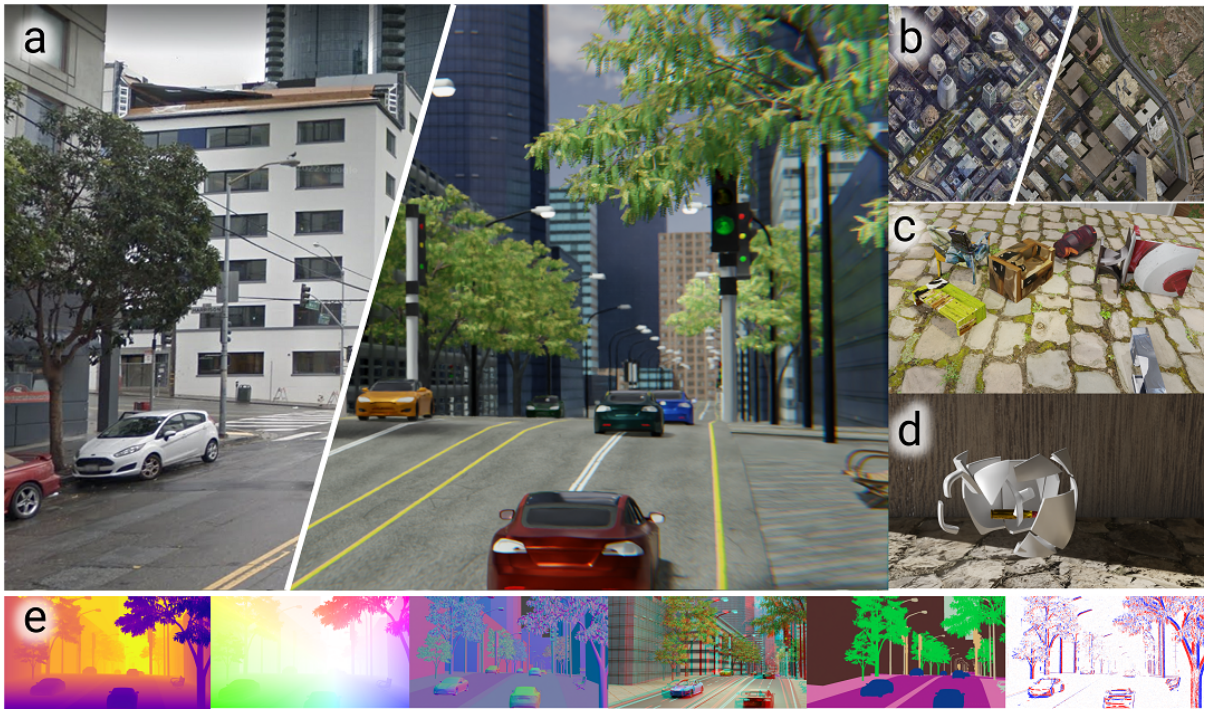

Figure 4.1: Generative ability of WorldGen: (a) Comparison between Google Street View (left) and the same street in WorldGen (right), (b) Comparison of Google Maps satellite image vs. WorldGen top view, (c) Collection of 3D objects in motion, (d) Object fragmentation,(e) Annotation from left to right: depth, optical flow, surface normals, stereo anaglyph, image segmentation, event frame. *All the images in this chapter are best viewed in color on a computer screen at 200% zoom.*

Table 4.1: Comparison of the different simulation environments.

Name	Rendering	GI	Physics	Scaling
UnrealCV [107]	UE4	×	UE4	×
iGibson [108]	PyRender	×	PyBullet	✓
Omnidata [109]	Blender	✓	–	×
Blenderproc [110]	Blender	✓	Bullet	×
AirSim [111]	UE4	×	AirSim	×
CARLA [112]	UE4	×	UE4	×
Kubric [113]	Blender	✓	PyBullet	✓
WorldGen *(Ours)*	Blender	✓	Bullet	✓

* Only the Blender rendering engine supports ray-tracing with OptiX denoiser for photorealistic images; others use rasterization. GI: Full Global Illumination Support; Physics Engine Used; Scaling: Scalable to generate large datasets.

with extensive variability is necessary. This entails either the manual creation of high-quality 3D assets or the organized assembly of these assets to construct a scene. It becomes evident that the implementation of a scalable simulation environment is imperative for enhancing the predictive models' quality [108] and [113] (see Table 4.1) showcased the effectiveness of scalable data in their training-based prediction analysis. This study introduces a scalable framework for the automated generation of structured and unstructured environments tailored for perception and robotics applications, eliminating the need for manual intervention. Fig. 5.1 illustrates the capability of WorldGen to automatically generate structured cities, object datasets, and object fragments through code.

In the past five years, self-driving car simulators for perception, planning, and control [111, 112, 114] have experienced significant growth. However, a notable limitation of these simulators is their lack of scalability in data generation. Due to the labor-intensive nature of creating 3D environments, simulators are typically equipped with a limited number of scenes or towns, hindering scalability and generalizability. Although these simulators offer scalability for traffic simulations, they often fall short in addressing perception tasks. For instance, the CARLA [112] library includes a mere 40 building models. To overcome this limitation, our method leverages the 3D structures of buildings extracted from satellite semantics and 3D maps, enabling the creation of an extensive range of building models.

Learning geometric annotations such as an optical flow that relies substantially on motion parameters (rather than textures) have been studied extensively in the past decade. Unlike monocular depth estimation and instance segmentation, the prediction of such geometric quantities through convolutional neural networks generalizes well across different domains. For such predictions, synthetic datasets have widely been used [105, 106, 115, 116] with a limited

number of objects and textures. Recent advancements in scalable scene generation environments like Kubric [113] have shown improvements in optical flow prediction due to more variability in data generation.

Another common problem in vision and robotics is segmenting and tracking independently moving objects [117–119]. It is a key process to understanding the dynamics of the scene structure for navigation tasks. Applications such as dodging any malicious objects on drones [60, 120], tracking objects using RGB and event cameras in motion [121, 122], and flying through gaps [123] have been well examined. In this chapter, we provide a framework to generate countless 3D scenes with texture, structure and lighting variability for the aforementioned applications. We summarize our key contributions next.

4.1.1 Key Contributions

- WorldGen is introduced as a generative simulator for creating diverse environments commonly encountered in real-world robotics and computer vision tasks.
- WorldGen's scalability enables the generation of an unlimited quantity of photorealistic data with variations in object placements, as well as object shape, texture, lighting, camera, and motion properties.
- Realistic camera distortions, including barrel distortion, chromatic aberration, and camera aperture, among other computational photographic properties, are supported by WorldGen.
- The utility of WorldGen is demonstrated in the improvement of optical flow accuracy through its enhanced data generation capabilities.

This chapter is organized as follows: Sec. 4.2 presents the construction, design principles, and details of the WorldGen generative simulator. Next, Sec. 4.3 discusses the usefulness of WorldGen in various robotics and computer vision applications. Finally, the chapter concludes in Sec. 4.4 with considerations for future work.

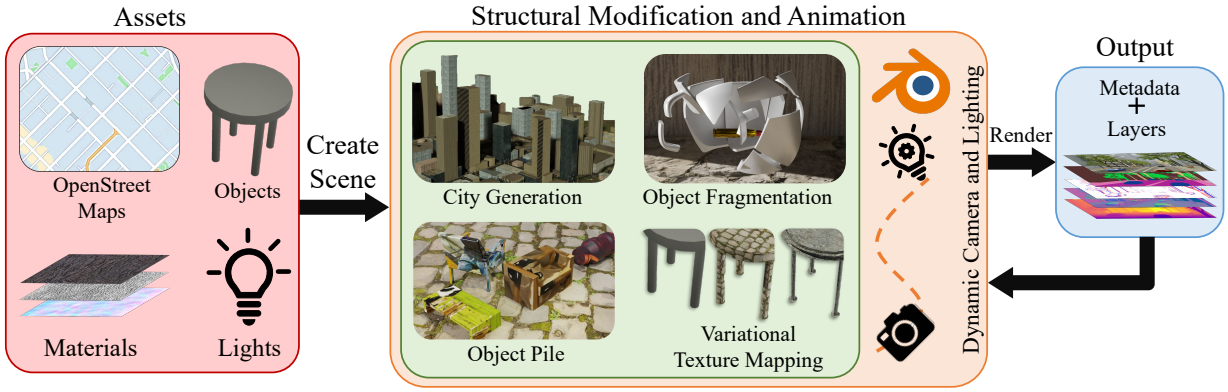

Figure 4.2: An overview of WorldGen Framework: (a) Assets: Loads the assets such as maps, objects, materials etc. into WorldGen environment, (b) Structural Modification and Animation: Modifying the texture maps and applying physics and motion models on different objects in the scene, (c) Rendering: Generates rich ground truth data with the desired metadata (time, frame number, camera intrinsic and extrinsic properties).

4.2 WorldGen Generative Simulator

WorldGen is a high-level open-source python library designed to generate synthetic data in large quantities. This library serves as a platform for generating visual data used in simulations for self-driving cars, autonomous drones, object segmentation, active vision, motion segmentation, tracking, computational photography, and other applications. The main contribution of WorldGen is an API that enables the creation of generative environments and simplifies the process of generating synthetic data, making it more accessible to researchers and practitioners. WorldGen utilizes Blender™, a free and open-source 3D creation suite, as its core tool for generating synthetic data, including city maps, collections of moving objects, and object

fragmentation. Currently, the framework incorporates the `Bullet` physics engine to handle object collisions, gravity, friction, and other force fields. An overview of WorldGen is depicted in Fig. 4.2. The design of WorldGen revolves around the central principles of scalability and speed. The subsequent sections discuss the various components and specific details involved in building WorldGen.

4.2.1 Environment

The framework is structured in three different stages: (a) Loader: asset loading of objects, materials, textures, lights, and/or map information to generate a structured 3D environment, (b) Structural Modification and Animation: generation of structurally different objects through variation of texture maps (explained in Sec. 4.2.2), application of physics and motion models to objects, camera, and lights, and (c) Rendering: generation of rendered frames, annotation, and desired metadata (such as time, frame number, camera intrinsic and extrinsic parameters) used for building WorldGen next.

Rendering: The framework employs Blender's *Cycles*, a ray-trace based production render engine, for the rendering process. To optimize render speed, a lower number of samples (or path-traced iterations per pixel) is typically chosen, resulting in a grainy appearance in the output renders. NVIDIA OptiX™ [124], a recurrent denoising autoencoder, is utilized to reduce this graininess (noise) without requiring additional rendering iterations. Additionally, the framework leverages both full Global Illumination (GI) for achieving photo-realistic renders and fast GI approximation for faster rendering. These rendering techniques serve as the foundation for constructing WorldGen in the subsequent stages.

4.2.2 Texture Mapping

WorldGen utilizes UV mapping [125], a well-known process in computer graphics, which projects a 2D image onto the surface of a 3D model. In addition to RGB images, open-source textures commonly include displacement and normal maps to simulate lighting effects and create the illusion of bumps on the 3D model. These mapping techniques enable the enhancement of a simplified or low-poly mesh without increasing the vertex count, thereby reducing rendering time. By adjusting the strength of the displacement and normal maps, a structurally distinct variant of the same object can be rendered without modifying the 3D object itself (See Fig. 4.3). Consequently, these maps undergo modification with additive white Gaussian noise before being applied to the 3D objects. This variational texture mapping technique facilitates the generation of numerous "similar" object renders from a limited set of object meshes. Such a capability can be leveraged to create extensive datasets for predicting geometric quantities (or annotations), such as optical flow, thereby improving generalizability across different domains. For further details, please refer to Sec. 4.3.1, which elaborates on the construction of WorldGen.

4.2.3 Generative Models

At the time of writing, WorldGen supports three different generative models: *(1)* City Maps *(2)* Object Pile, and *(3)* Object Fragmentation.

Figure 4.3: Mapping textures to a round table. Top row: Rendered Output, Bottom row: Sample textures projected on a sphere. (a) Barebone 3D model, (b)-(d) Different Textures applied on (a). *Note: Variational mapping models change the structure of the 3D objects in different renders (notice the legs on the chair). Here, the Gaussian noise in* $(d) > (c) > (b)$.

Figure 4.4: (a) OpenStreetView, (b) Depth Map, (c) 3D Model View Generated by WorldGen and (d) Final Rendered View

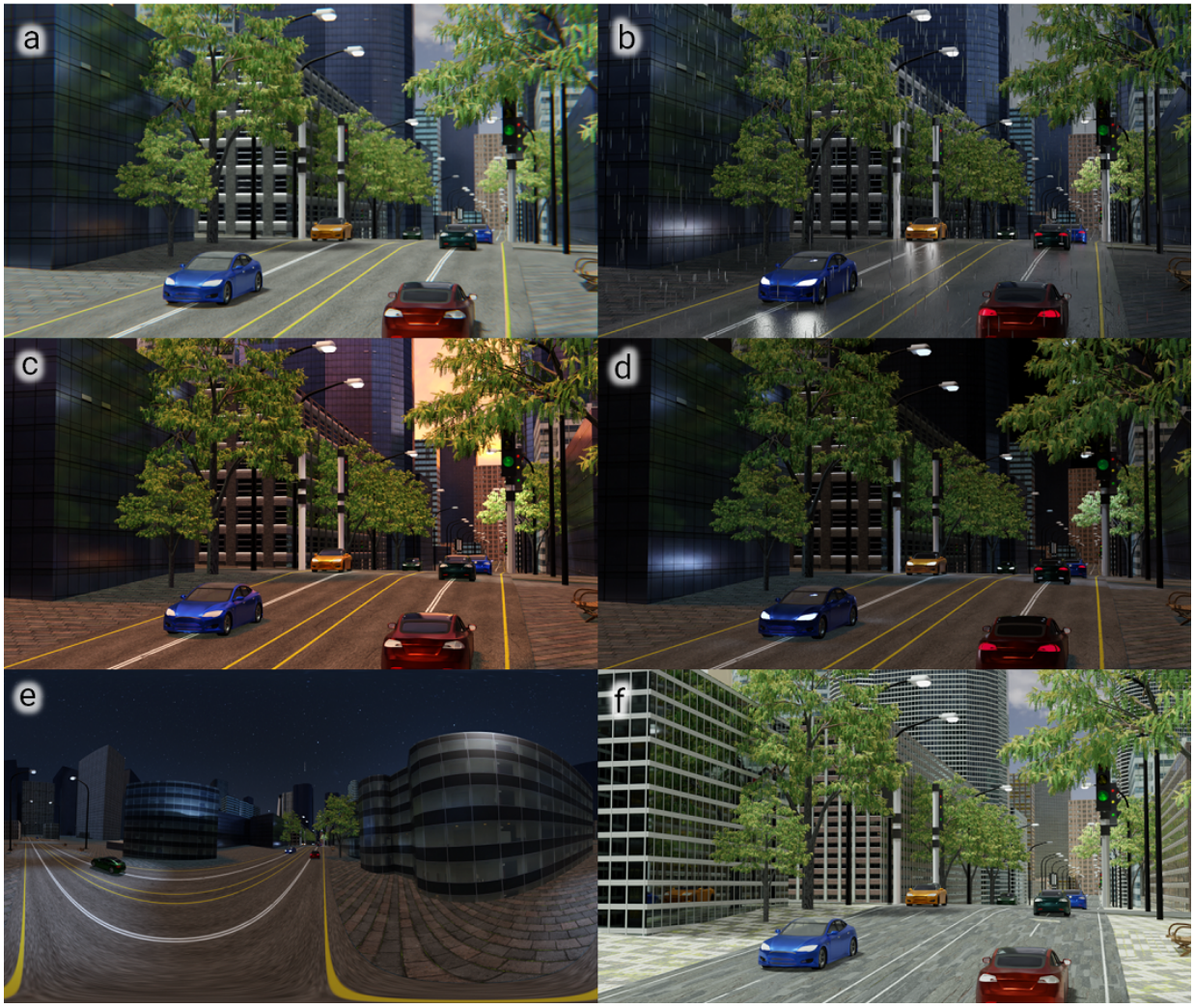

Figure 4.5: City environment in different weather and time of the day: (a) Day, (b) Night with rain, (c) Dawn and (d) Night without rain, (e) panoramic view of the city and (f) demonstrates the generative ability of WorldGen by changing the textures of the entire scene while keeping the same structure.

4.2.3.1 City Models

The framework utilizes OpenStreetMaps (OSM), a crowd-sourced project containing semantic labels and 3D terrain maps from the satellite perspective. The input of latitude and longitude is used to import semantic and 3D terrain information into the framework using the tool mentioned in [126]. Relevant assets, based on the semantic information such as buildings, water

bodies, forests, vegetation, roads, highways, pedestrian pathways, and railways, are imported from open source libraries (refer to Sec. 4.2.6). The appropriate materials are also assigned to these assets (refer to Fig. 4.4). Different roof structures, such as flat or gable, can be applied to various buildings. The building roof information obtained from the OSM semantic data enables the deployment of realistic roof structures (flat or gable) on the respective buildings. Additional elements like radio antennas, air-conditioning vents, and chimneys are randomly distributed on the roof surface. For other solid objects such as roads, 3D meshes are created in the environment using the vertices extracted from the semantic data. Lighting is also incorporated into the scene, providing global illumination (GI) support for different weather and daytime conditions to achieve a photo-realistic visualization (see Fig. 4.5). By employing basic morphological and vector operations, road intersections are estimated to deploy traffic lights and stop signs near these intersections. Common street objects such as trees, benches, and street lights are structurally distributed near pedestrian pathways. The final rendered output of the city environment, including realistic distortions found in camera images like chromatic aberration, is illustrated in Fig. 5.1.

4.2.3.2 Object Pile

The ability of the WorldGen framework to generate a collection of moving objects in space can be compared to Kubric [113] and BlenderProc [127] in terms of scope. The framework offers a key advantage over Kubric in terms of flexibility for generating large-scale datasets. This advantage is achieved through variational texture mapping, which enables the generation of different structural variations of the same 3D models. The output of an object pile environment is shown in Fig. 5.1b. WorldGen provides support for various physics controls, including

collision shapes such as convex hulls, 3D meshes, boxes, cylinders, etc. These controls can be customized with different collision margins, friction, and coefficient of restitution, allowing for greater control in the data generation process. Additionally, our framework incorporates force fields beyond the standard gravitational force, such as wind and drag, to influence the motion of objects. This inclusion of additional force fields reflects real-world scenarios more accurately and can be beneficial for learning perception modules in realistic settings.

4.2.3.3 Object Fragmentation

Another generative model within the framework involves breaking or *fragmenting* a 3D mesh into a user-defined number of Voronoi cell fractures. The [128] method is employed to generate the specified number of 3D meshes derived from the parent mesh. Fragmentation occurs when the object comes into contact with another rigid body (active or passive) or experiences a non-uniform force field. The resulting output is a video sequence captured by camera sensors, providing semantic information for each individual fragment. Fig. 5.1c illustrates the simulation environment depicting a scenario where a bullet collides with a cup, causing it to shatter into multiple pieces. This approach holds the potential for assessing damage resulting from collisions.

4.2.4 Sensors

WorldGen renders RGB images along with annotation data, including depth map, optical flow, stereo images, semantic map, and surface normal map (See Fig. 5.1). These renderings are performed directly through the Blender Cycles engine. In the case of event camera renders, a

simple event camera model [129, 130] is employed to generate events at a location $\mathbf{x}$ when

$$\| \log\left(\mathcal{I}_t\left(\mathbf{x}\right)\right) - \log\left(\mathcal{I}_{t+\delta t}\left(\mathbf{x}\right)\right) \|_1 \geq \tau \tag{4.1}$$

In the given context, let τ denote a threshold defined by the user. $\mathbf{x}$ represents the pixel location, and $\mathcal{I}_t$ denotes a grayscale image captured at time t. An *event-frame* is produced using the aforementioned model, incorporating additive Gaussian noise. Alternatively, a more realistic event model [131, 132] can be adapted to the framework. To achieve a slow-motion effect and simulate a visual high-speed camera generating events, the animation timeline is remapped to include $\mathbb{N}$ times more frames, considering a smaller integration time for event frame generation.

Multiple sensors can be instantiated in the environment to generate outputs from different camera poses simultaneously. WorldGen utilizes 6-degree-of-freedom camera extrinsic parameters to generate outputs from various cameras. This configuration allows for the simulation of a sensor suite akin to those commonly found in self-driving cars or autonomous drones.

Camera Properties: WorldGen accommodates both dynamic camera trajectories and variations in camera intrinsic properties, including dynamic focal length, depth of field, camera aperture radius, or variable camera baseline in the case of stereo output, to simulate and generate data similar to previous works such as [133, 134]. These properties can be temporally modified using linear interpolation between defined keyframes or Bézier curves. The same method employed in WorldGen is utilized to dynamically move solid objects within the scene. Additionally, our framework supports fisheye and equirectangular projections to render full

360-panorama images (Fig. 4.5e), owing to variability in camera focal length. Furthermore, WorldGen incorporates real-world camera distortions, such as lens glare, chromatic aberrations, and barrel distortion, for a more realistic rendering. This feature is particularly advantageous for handling corner case scenarios, potentially contributing to improved simulation-to-reality generalization.

4.2.5 Lighting and Climate Conditions

Currently, the lighting conditions in WorldGen encompass three settings: midday, sunset, and night. Additionally, there are four weather conditions available: rain, cloudy, clear, and fog (Figs. 4.5a to 4.5d). WorldGen utilizes Global Illumination (GI) to achieve realistic sunlight diffusion and reflection. In addition to GI, it offers fast GI approximation to reduce rendering time while maintaining a satisfactory level of photorealism. Various aspects related to lighting, such as diffused sky radiation, sun angle, power, color temperature, and light sources like street lights, can be temporally adjusted.

4.2.6 Assets

The WorldGen generative simulator leverages existing 3D meshes and materials to create new 3D environments, allowing for modifications as needed. When generating object piles and fragmentation scenes, WorldGen imports objects and materials using a text file that contains paths to wavefront `.obj` and material `.mtl` files. This streamlined approach simplifies the importation of 3D object meshes into the WorldGen environment. A wide range of possibilities is made available through the incorporation of diverse 3D object databases and image textures.

While custom objects and textures can also be imported, the following resources are utilized:

ShapeNetCore.v2 [135]: ShapeNetCore.v2 consists of approximately 51,300 unique 3D models distributed across 55 object categories. These models have been manually verified for classification and alignment annotations. Kubric's cleaned version of ShapeNet objects is utilized, addressing issues related to auto-smoothing and backface culling found in the original ShapeNetCore objects.

Google Scanned Objects (GSO) [136]: GSO, licensed under CC-BY 4.0, encompasses 1030 scanned objects accompanied by their associated metadata, resulting in a total size of approximately 13 GB. These models reflect real object properties faithfully, as they are derived from scans rather than being manually created in a 3D modeling tool.

MS-COCO [137]: Microsoft's COCO dataset contains over 328k images, showcasing complex everyday scenes with common objects in their natural context. With 91 object classes represented, these lower-resolution images are employed for warping around objects, modifying their natural appearance.

AmbientCG [138]: AmbientCG is a vast public domain resource that operates under the CC0 license. It provides Physically Based Rendering (PRB) support, offering 3D objects, materials, and High Dynamic Range Images (HDRI). The collection includes photo-scanned materials and displacement maps with an accuracy of up to approximately 1 pixel, encompassing around 1760 assets with texture resolutions up to 8K.

Polyhaven [139]: Polyhaven is a public library featuring over 500 pre-processed HDRI images, available under the CC0 license. These images are utilized as resources for backgrounds and lighting within WorldGen.

cgbookcase [140]: cgbookcase, operating under the CC0 license, offers over 540

high-resolution PBR textures. These textures cover common city structures such as various walls, roads, buildings, concrete, and more.

In addition to the aforementioned assets, WorldGen incorporates open-source libraries and Blender add-ons for pre-processing and rendering operations, including `Blender-OSM`,

4.2.7 Design Principles

4.2.7.1 Modularity

The framework of WorldGen exhibits high modularity, with three distinct elements discussed in Sec. 4.2.1. Each module can be treated independently and replaced with a third-party module. For example, the framework can be extended to generate diverse human motions on custom backgrounds using assets from 3D character rigging resources like Adobe's Mixamo [141] to obtain ground truth data annotations for human pose estimation. The future plan includes incorporating character-rigged motion scene generation and data annotation with varying character motion into WorldGen.

4.2.7.2 Ease of use

The design philosophy of WorldGen focuses on simplifying the process of generating new scenes for researchers, educators, and practitioners, eliminating the need for extensive knowledge of computer graphics concepts, modeling, animating, and rendering tools. WorldGen achieves this through a user-friendly, high-level object-oriented Python API, while leveraging Blender (and Bullet) in the background.

4.2.7.3 Open Source

Upon acceptance, the scene and data generation codes will be released as open source and freely available for academic use. The utilization of third-party assets in WorldGen, governed by Apache2.0, MIT, and CC0 licenses, enables researchers to generate diverse environments, render annotations, and share the data with the community, promoting reproducibility.

4.2.7.4 Scalability

WorldGen enables the generation of new structured environments without manual labor, offering scalability from small towns to large cities, with consideration for GPU and access memory requirements.

4.3 Applications

WorldGen is specifically designed for generating 3D environments and annotated data for robotics and computer vision applications. It serves as a comprehensive framework for zero-shot generalization, offering variability in shapes, textures, and dynamic lighting while minimizing the sim-to-real transfer gap. Additionally, it can function as a simulator for the development and testing of planning and control algorithms, thanks to its flexibility in utilizing the current state and annotations as input for the subsequent time step. To demonstrate the versatility and adaptability of WorldGen, a range of common and challenging problem statements in robotics and computer vision are discussed.

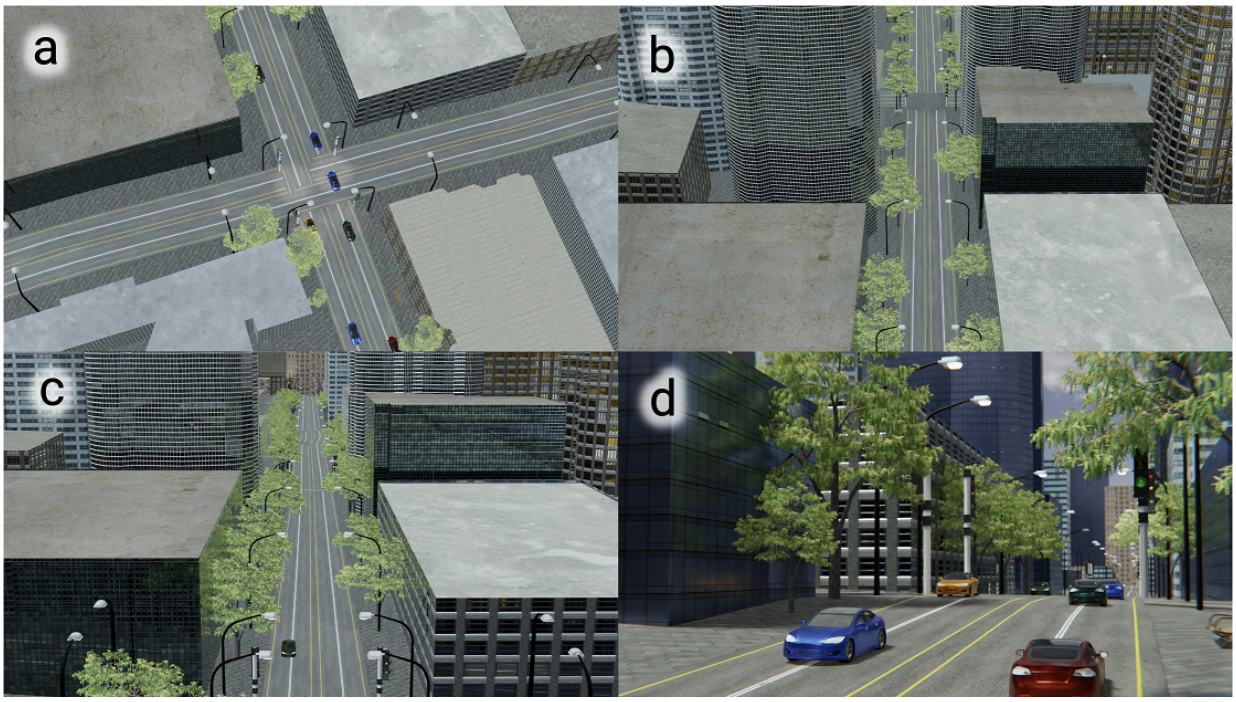

Figure 4.6: High-resolution views generated by WorldGen from views at different altitudes with dynamic lighting, camera intrinsic, and extrinsic.

4.3.1 Improvements in Optical Flow

Optical flow, a fundamental quantity in computer vision and robotics, quantifies the 2D motion of each pixel between consecutive frames. In the context of real-world data with human annotation, obtaining reliable ground truth is challenging, particularly for high-level computer vision applications such as instance segmentation. Recent advancements in optical flow methods, including PWC-Net [69], RAFT [1], and GMFlow [142], rely on synthetic datasets like MPI Sintel [116], FlyingChairs [105], and FlyingThings3D [106] for pre-training. These datasets feature synthetic chairs, lacking photorealism and realistic 3D motion. Dataset generation techniques, such as AutoFlow [143], aim to learn hyperparameters for rendering synthetic flow datasets, reducing the End Point Error (EPE). However, AutoFlow's utilization of a simple 2D layered model limits its representation of 3D motion and photorealism. Kubric [113], which

leverages photorealistic rendering using Blender, demonstrates superior performance compared to AutoFlow but specifically on Sintel's *Clean* samples.

This is the rendered output that contains shading but no image degradation whereas the *Final* pass includes motion blur, defocus blur and atmospheric effects. Since Kubric lacks both volumetric features (like fog, mist, and rain) and advanced camera features (such as depth of field and motion blur), AutoFlow performs better even without photorealism as images are generated with motion blur and fog models as well as data augmentation for visual effects.

We compare and analyze optical flow predictions using RAFT on Sintel *Clean* and *Final* pass which are trained on different datasets. Table 4.2 shows the EPE in optical flow. Note that WorldGen causes a significant increase in optical flow accuracy as compared to FlyingChairs. Furthermore, WorldGen outperforms AutoFlow due to the difference in photorealism. Note that WorldGen slightly improves over Kubric. We speculate one of the reasons to be WorldGen rendering multiple instances of the same object configuration by warping different textures on the objects, ensuring the network avoids over-fitting to object textures, and thereby learning the geometrical/motion properties of the scene. Since WorldGen supports dynamic lighting, depth of field, and motion blur modeling, we see a significant improvement over Kubric in the *Final* pass. Note that AutoFlow performs better in *Final* pass (although only slightly) because the hyperparameters of AutoFlow have been learned to optimize the performance on the Sintel dataset which gives AutoFlow an unfair advantage.

Table 4.2: Optical Flow EPE Comparison of Training RAFT [1] On Different Datasets. Lower Is better.

Dataset	Dimensionality	Parameters	Sintel Clean	Sintel Final
FlyingChairs [105]	2D	Manual	2.27	3.76
Kubric [113]	3D	Manual	1.89	3.02
AutoFlow [143]	2D	Learned	2.08	**2.75**
WorldGen (Ours)	3D	Manual	**1.86**	2.87

4.3.2 Computational Photography

Computational photography encompasses various tasks such as depth map prediction and image denoising, which rely on a substantial collection of image data. Existing synthetic data generators in robotics and computer vision often overlook the rich features provided by Blender, Unity, or Unreal Engines, such as volumetric effects and variations in sensor sizes, camera lenses, and rolling shutter sensors. WorldGen, capable of generating data for diverse lighting conditions, different *bokeh* blur effects resulting from depth of field, motion blur, variable focal lens and sensor sizes, as well as *Albedo*, *Clean*, and *Final* renders, can be employed to produce data for computational photography applications. These applications include HDR+ datasets [144], depth estimation from defocus [145, 146], image recovery from rolling shutter blur [147, 148], video sequence extraction from a single blurred frame [149], and motion blur synthesis learning from image pairs [150]. Additionally, this work can facilitate the generation of datasets for training depth estimators using a coded aperture [151, 152] camera. A toolbox for generating coded *blurred* frames within the WorldGen environment is planned for future development.

4.3.3 View Synthesis using Neural Radiance Fields

Neural volume rendering has witnessed significant growth in the past two years [153]. One of the primary challenges discussed in [153] involves synthesizing dynamic relighting scenarios. Existing methods struggle to perform well when dealing with diverse views, ranging from satellite-level observations that capture entire cities to ground-level perspectives, mainly due to substantial camera displacement and dynamic lighting variations. Recent advancements in multi-scale scene rendering [154] and dynamic irradiance view synthesis [155] have been developed to address these challenging conditions. In [155], the authors leverage multi-date images that exhibit significant changes in appearance caused by varying shadows and transient objects like cars and vegetation, utilizing the WorldView-3 dataset. To achieve more robust view synthesis in satellite images with larger displacements under different lighting and weather conditions, WorldGen can generate thousands of 3D satellite views with dynamic lighting at a higher resolution than the original satellite images, while also providing known camera intrinsic and extrinsic parameters (refer to Fig. 4.6). Furthermore, this approach can be extended to facilitate the learning of high-level planners for aerial cinematography [156].

4.3.4 Active and Interactive Perception

WorldGen functions as an automated 3D scene and dataset generation tool and additionally operates as a framework for active and interactive perception applications. Quadrotor robots, for instance, often utilize the present prediction and annotations as input for the subsequent state in their solution and testing processes. Currently, no test bench exists for the validation of robotics algorithms reliant on active and interactive perception methodologies [9, 14]. WorldGen

will serve as a benchmark pseudo-simulator to ascertain the performance of such algorithms, enhancing reproducibility. The aspiration is for WorldGen to establish itself as the OpenAI Gym [157] for Active and Interactive perception.

4.3.5 Generating Real World Traffic

Generating *realistic* traffic scenes automatically poses a challenging task. Existing methods often rely on planning and reinforcement algorithms that necessitate the availability of current and previous traffic states, traffic signal conditions, and road/pathway vertices [158]. As WorldGen facilitates the exportation of these parameters, neural autoregressive models like [159] can be employed to achieve realistic traffic distribution. The recent advancements in neural fields, such as Panoptic Neural Fields [160], have extended the capabilities to detect traffic in a 3D representation, enabling 3D scene editing. Leveraging these techniques, WorldGen can also incorporate real-life traffic data into its model to generate photorealistic city traffic scenarios.

4.3.6 Human Pose Estimation

In recent years, significant advancements have been made in 3D human pose estimation, primarily driven by large-scale datasets. However, the accuracy of these datasets in capturing precise poses is limited due to human annotation from open-world datasets [161], which are susceptible to labeling inaccuracies caused by occlusions. Conversely, motion capture systems provide accurate human pose data but are constrained by limited space, preventing the creation of open-world environments [162]. Additionally, the existing simulation environments used for dataset generation lack photorealism [163]. To address these limitations, a new generative

environment will be developed using WorldGen in conjunction with Mixamo [141], an open platform offering a wide range of character rig models and animations. By warping various 3D materials, this environment will enable the generation of a large dataset of photorealistic character animations, including diverse backgrounds like small towns and cities. This endeavor aims to advance monocular human pose estimation and explore novel possibilities for specialized sensors such as event cameras.

4.4 Conclusion

WorldGen is introduced as a modular, generative, open-source Python API for creating a large-scale generative simulator with a diverse range of scenes. Accurate ground truth labels can be generated using WorldGen for optical flow, depth, surface normals, depth cameras, semantic maps, and event data in various scenarios such as driving scenes, moving object piles, and object fragmentation. WorldGen also incorporates simulation of camera properties and motion properties, providing a perception-centric simulation environment for generating data with photorealistic quality. The dynamic manipulation of objects, lighting, and scenes in WorldGen facilitates the generation of data that aids in improving the generalization capabilities of neural networks, including optical flow, as demonstrated in the conducted experiments. Currently, WorldGen lacks the inclusion of sensors such as SONAR and LIDAR, as well as behavior modeling for self-driving cars. Additionally, the implementation of human character rigging is absent, highlighting potential directions for future research and development.

This page intentionally left blank.

Chapter 5: Generalized Neural Metric Depth Estimation

Depth estimation assumes a crucial role in the navigation of robots. However, the autonomous capabilities of small mobile robots are significantly affected by constraints such as power and weight, making it impractical to incorporate high-grade depth sensors. To address this challenge and enable autonomy in small robots, we propose TinyDepth, a generalized metric depth prediction technique that leverages two distinct perspectives using an RGB sensor and a sparse Time-of-Flight (ToF) sensor. The proposed approach facilitates autonomy in tiny robots with a width as small as three inches by inferring dense depth maps at a rate of up to 4.4Hz on single-core embedded devices. In our evaluation, we compare TinyDepth against existing metric depth estimation methods using unseen samples. Furthermore, we validate the effectiveness of TinyDepth in real-world robotic environments, demonstrating its potential for enhancing the autonomy of drones and cars. Our method achieves a combined accuracy of approximately $\sim 87\%$ across various environments, without requiring retraining or fine-tuning on experimental data.

5.1 Background

Measuring distances or depth cues is a fundamental competence required by autonomous robots to understand 3D scene geometry. In the context of navigation, agents can readily use

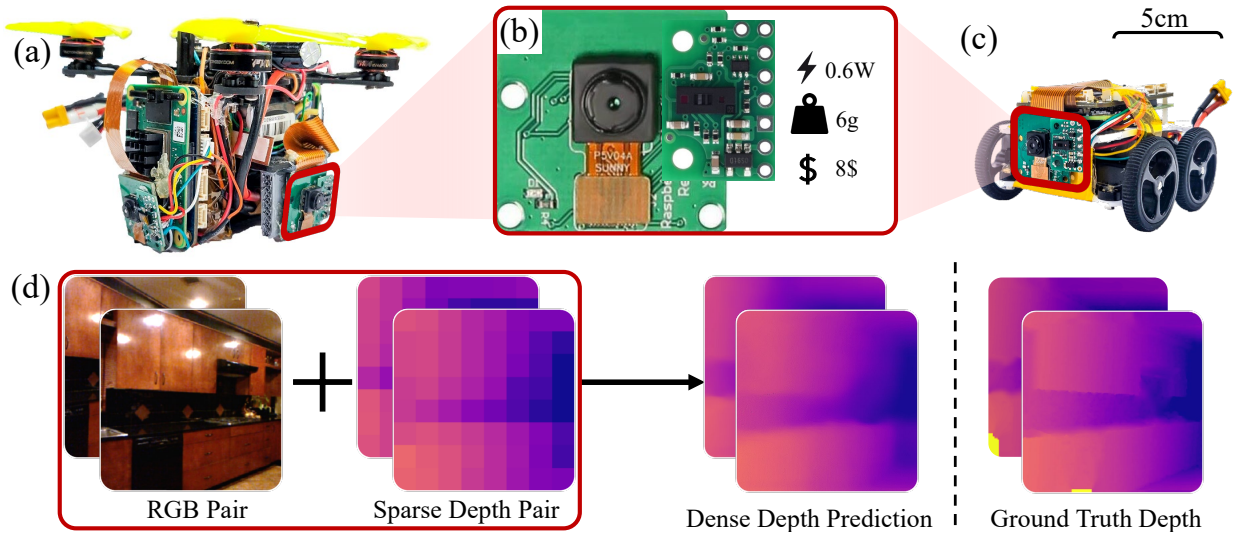

Figure 5.1: Depth Estimation for Tiny Autonomous Robots: (a) Bee drone of size 92cm in the largest dimension, (b) Lightweight sensor suite – RGB and sparse time-of-flight sensor used on Bee drone and Tiny car, (c) Tiny car of size 70cm largest dimension and (d) illustrates the pair of sensor inputs along with our metric dense depth prediction with the ground truth on the right.

the depth map to navigate in complex and dynamic environments. However, traditional depth estimation algorithms (monocular or stereo) rely on expensive computations or high-quality sensors, thereby inhibiting the usage in resource-constrained settings. Furthermore, depth computation can be sped up by using motion cues such as parallax as done by pigeons. Previous works have attempted to reduce the computational footprint by reducing resolution or using known environmental cues. Although these work well in theory, in practice, they are not accurate for obstacle avoidance or do not generalize to novel scenes.

In this chapter, TinyDepth is proposed, a tiny neural network that uses a combination of a low-resolution and low-power sparse depth sensor (64 depth values) and combines it with a monocular high-resolution RGB camera to obtain dense depth. Information from different views is utilized to add motion parallax cues to train the model, enabling better generalization to zero-shot or unseen scenes without any fine-tuning or re-training. The network runs at 4.3Hz on the Raspberry Pi CPU and is similar in accuracy to larger models and better in terms of speed by

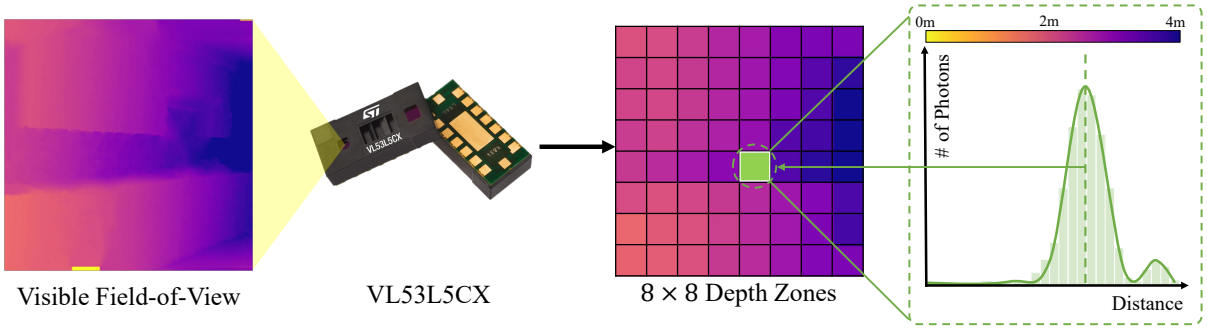

Figure 5.2: Sensing principle of VL53L5CX sensor that results in super sparse 8×8 depth resolution.

at least a factor of magnitude. The method can be used on palm-sized robots with ease and even hummingbird-sized aerial robots due to the lightweight computational and sensor footprint.

The work presented here shares a close affinity with [22], although notable distinctions exist. Notably, our model exhibits significantly smaller size (up to 126$\times$ smaller), incorporates cues from multiple perspectives, and possesses the ability to generalize to zero-shot or previously unseen environments following simulation-based training. Moreover, the effectiveness of the proposed methodology is demonstrated through real-world robotics experiments involving navigation in intricate static scenes utilizing both ground and aerial robots.

In this chapter:

- A metric dense depth prediction model is proposed by observing RGB and sparse ToF sensors from two different views.
- Real-time performance is demonstrated on single-core embedded devices for robot navigation without fine-tuning or retraining on the experimental data.

5.1.1 Monocular Depth Estimation

Monocular depth estimation has been extensively studied in the last decade, employing both supervised and unsupervised approaches. Supervised methods utilize ground truth depth labels from Lidar or RGBD cameras, along with RGB monocular images, for training. However, these methods have limitations such as high-hardware cost and a limited depth range. The application of Convolutional Neural Networks (CNNs) in a supervised fashion for monocular depth estimation was first introduced by Eigen *et al.* [164]. Subsequent improvements included the incorporation of multi-scale features/loss [165, 166] and conditional random fields [167] into CNNs. Recent advancements in transformer models [168] and ViT [169] have elevated the accuracy of monocular depth estimation to new standards [170–172]. The accuracy has been further enhanced by leveraging semantic information as a constraint for finer depth boundaries, as demonstrated by [173, 174].

5.1.2 Estimating Depth using Multiple Frames

While most monocular depth models utilize single images as inputs, the study of unsupervised depth estimation from multiple views has significantly reduced the cost of depth annotation. These methods commonly employ photometric loss between consecutive frames for training purposes. Left-right consistency for unsupervised monocular depth estimation is introduced by Godard et al. [175]. The estimation of monocular depth jointly learned with camera poses and/or optical flow is introduced by Ummenhofer et al. [176], Zhou et al. [50], and Yin et al. [177]. To predict depth maps, DeepMVS [178] constructs a set of cost volumes through plane sweep [179] using a random number of image patches. Chen et al. [180] utilize uncertainty to

enhance depth estimation accuracy from two views. Cheng et al. [181] further enhance accuracy by extracting the correlation volume pyramid to represent pixel-wise similarities between the two views. Greene et al. [182] demonstrates the recovery of depth maps from known images captured from unconstrained views.

5.1.3 Using Sparse Depth Supervision

The industry standard for metric depth estimation in industrial, researcher, and consumer applications has been established with the introduction of Microsoft KinectV2, Intel Realsense, and Apple iPhone X LiDAR sensors. Over the past five years, researchers have showcased dense depth predictions by combining RGB data with sparse metric depth data obtained through sampling methods, such as Time-of-Flight (ToF) and structured light depth sensors. The Sparse-to-Dense approach [183] employs 100 spatially random samples from LiDAR data along with RGB information to generate dense depth maps. Another approach by [184] combines RGB data with extremely sparse data, even as low as a single pixel per image, to predict dense depth. The work by [185] utilizes a dynamic spatial propagation network [186] for depth completion, while [187] leverages vision transformers for the same task. In contrast, [188] combines a 8×8 ToF depth sensor with single RGB images for depth estimation. Lastly, [189] relies solely on an 8×8 ToF depth sensor for obstacle avoidance and drone navigation.

This chapter is organized into the following sections. Sec. 5.2 provides an explanation of the sensor setup and data generation, along with the system layout and neural network details. Sec. 5.3 demonstrates the experimental setup and the utilization of the TinyDepth model in small aerial and ground robots for autonomous navigation in unknown environments. A qualitative

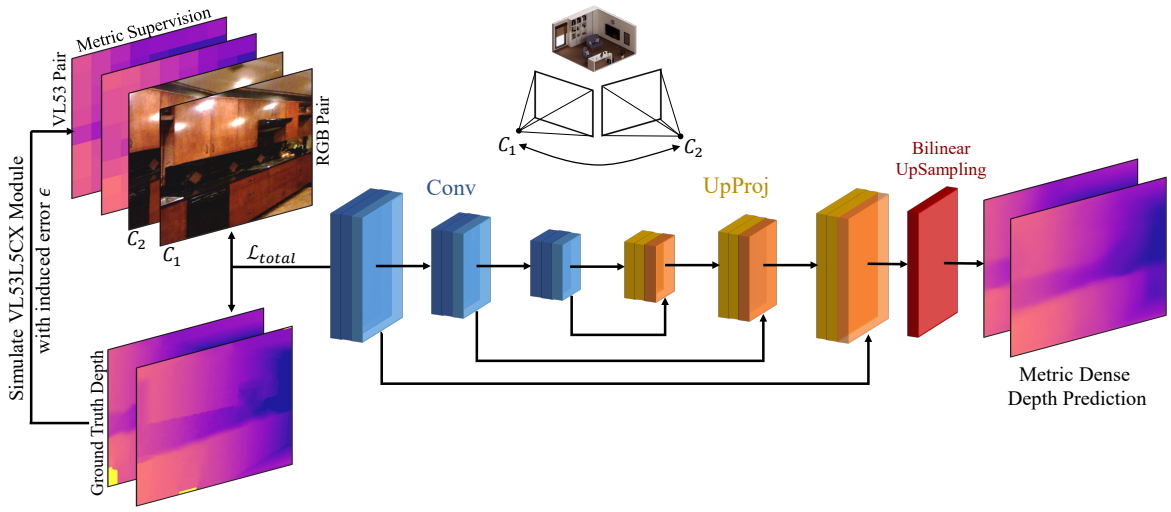

Figure 5.3: System overview: An architecture of TinyDepth encoder-decoder model that utilizes an eight-channel input by combining RGB and L5 data from two views to predict metric dense depth. *Refer Fig. 5.2 for the color scheme.*

and quantitative comparison with state-of-the-art methods is presented in Sec. 5.3.3. Finally, the findings, limitations, and prospects for future work are discussed in Secs. 5.4 and 5.5.

5.2 TinyDepth Model

The proposed method outlined in this section is designed to estimate a high-resolution metric depth map by observing pairs of traditional color images and sparse ToF depth data (8×8 pixels) from two views. Before delving into the details of the neural network, it is important to comprehend the sensor setup.

5.2.1 Sensor Setup

The setup comprises an RGB camera and a VL53L5CX sensor that are rigidly attached to each other (See Fig. 5.1). To ensure successful data alignment between the two sensors, precise knowledge of the extrinsic calibration between them is crucial. The VL53L5CX sensor consists

of 64 depth zones (8×8 px.) with the capability to estimate depth up to 400cm. It operates at a rate of 15Hz and has 4cm bins, covering a diagonal Field-of-View (FoV) of $63°$. On the other hand, the RGB camera has a diagonal FoV of $67°$.

In order to simulate the VL53 sensor (or L5 signals) for generating training data, it is necessary to understand the model of how the sensor collects the depth data. The model of L5 relies on a histogram-based algorithm for computing the depth (See Fig. 5.2). Each zone of L5 outputs the mean distance (or depth) value of the distribution obtained from all the photons projecting onto that sensor zone. The ToF L5 sensor consists of 100 bins in each zone, providing a depth range of up to 400cm, with a single bin corresponding to 4cm. This characteristic makes the sensor low-power and low-bandwidth. Additionally, L5 also provides information about the status, indicating if there are too few samples or unstable results. In such cases, the zone values are not used for any inference.

When aligning RGB data to L5's zones, relying solely on extrinsic parameters is insufficient [22], as the precise depth and pixel-wise alignment is unknown. To establish spatial correspondences between the two sensor data, we employ alignment and rectification methods proposed by Li et al. [22]. It should be noted that unstable zones that receive very few photons are discarded in the process.

5.2.2 Pre-Processing and Data Generation

To train depth maps, the sparse L5 data is simulated using ground truth depth maps. The model is trained on the NYUv2 [190] dataset. The aligned RGB and depth images are cropped to a resolution of 320×320 px. To simulate L5 signals, the ground truth depth image is divided into

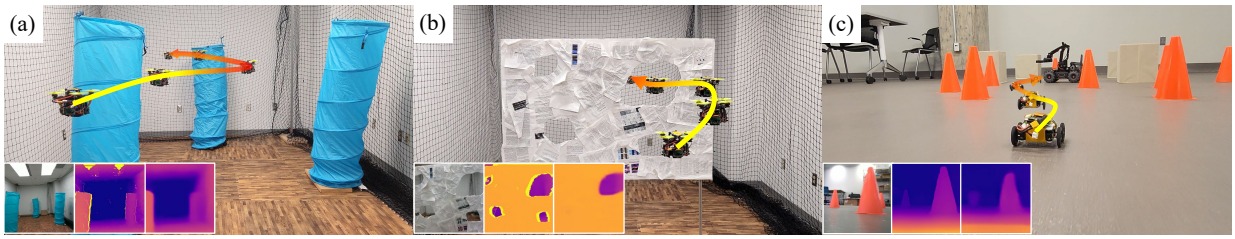

Figure 5.4: Real-world robot experiments: (a) Drone Navigation in an unstructured indoor forest scene, (b) Flying through unknown gaps, and (c) Tiny Car navigation through an obstacle course. The bottom left insets in each section of the image represent input RGB image, ground truth data, and depth prediction (left to right). *Note the gradient yellow to red line shows the traversal of the robots in time where red represents temporally later stage.*

8×8 zones, with each zone having a resolution of 40×40 px. It is important to note that both the RGB and L5 data maintain the same resolution of 320×320 px. This allows for faster learning of pixel-to-pixel correspondences between the two sensor data compared to using 8×8 px. L5 data for training. Within each zone, the mean of the normal distribution of the depth data is computed and approximated to the closest binned L5 value (multiple of 4cm), which represents the L5 signal of that zone. Gaussian noise is also introduced to each L5 signal to prevent overfitting of the TinyDepth model to the L5 data. Additionally, a few L5 signals are randomly removed to simulate the unstable zones of the VL53. It is worth mentioning that the depth data is not normalized and is maintained in metric units between 0m and 4m in order to learn the scale. The L5 simulator for NYUv2 [190] and WorldGen [20] is also made available.

For the generation of training data online, the following data augmentation methods are employed:

- $\mathbb{R}^2$ *crop* [191]: Patches are cropped from images at random locations, and the size of crop patches is randomly adjusted, followed by rescaling to 320×320 px.

- *Scale:* The RGB and depth data are augmented by applying a scale factor $s \in [1, 1.2]$.

- *Color Jitter:* The brightness, contrast, and saturation of the color images are varied within the range of $j \in [0.8, 1.2]$ of their original values.
- *Flip:* Both the RGB and depth data are horizontally flipped with a probability of 0.5.
- *Noise on L5 Signals:* Gaussian Noise is added to the L5 signals, with the standard deviation randomly chosen from 0.4, 0.8, 1.2m for the training data. Additionally, there is a 0.1 probability of no L5 signal to prevent overfitting.

It is important to note that the same data augmentation is applied to a pair of input data, specifically two RGB and two L5 data.

5.2.3 Flow-Guided Depth Estimation

To ensure the learning of geometry rather than regressing or overfitting on the training data, an eight-channel input strategy is employed, consisting of a six-channel RGB and two-channel simulated L5 data observed from two different views, denoted as C_1 and C_2 (see Fig. 5.3). This approach introduces motion cues into the network. The assumption is made that the surfaces in the field of view are Lambertian, based on relying on the ToF sensor for metric depth. A convolutional encoder-decoder architecture with skip connections is utilized, incorporating residual learning. Each encoder/convolutional block starts with a bottleneck block, followed by traditional residual blocks [192]. The output of each block is used for skip connection in the decoder (see Fig. 5.3). Instead of using larger kernel size layers for upsampling, the UpProj and FastUpProj modules proposed by Laina *et al.* [193] are employed to achieve the same level of prediction accuracy with fewer parameters and less training data compared to deconvolutional or up-convolutional layers. The depth prediction is obtained by passing it through a bilinear

upsampling layer.

For the depth prediction, loss functions inspired by those conventionally used to train optical flow networks are utilized, as they have demonstrated better generalization [69, 194]. The predicted depth is regressed with supervised ground truth data using various loss functions $\mathcal{L}_r$ explained further ahead. Mean-squared error ($\mathcal{L}_2$) is the de facto standard for depth estimation, despite its susceptibility to outliers during training as it heavily penalizes larger errors. Moreover, $\mathcal{L}_2$ tends to over smooth the surface, causing the dissolution of depth edges and rendering the prediction unsuitable for robotics tasks. To address this, Charbonnier (Pseudo Huber) [195] and Reversed Huber (berHu) [196] loss functions are adopted and evaluated, as given by

$$\mathcal{L}_{\text{Char}}(\alpha) = \begin{cases} |\alpha|, & if|\alpha| \leq c \\ \dfrac{\alpha^2 + c^2}{2c}, & \text{otherwise} \end{cases} \tag{5.1}$$

$$\mathcal{L}_{\text{berHu}}(\alpha) = \sqrt{1 + (\alpha/c)^2} - 1 \tag{5.2}$$

where c is a batch-dependent parameter. We choose c as 2.0 for $\mathcal{L}_{\text{Char}}$ and 20% of the maximum absolute error over all pixels in a batch for $\mathcal{L}_{\text{berHu}}$. For training, we test with four different models that use $\mathcal{L}_r$ as either $\mathcal{L}_1$ or $\mathcal{L}_2$ or $\mathcal{L}_{\text{Char}}$ or $\mathcal{L}_{\text{berHu}}$ and evaluate the prediction accuracies as reported in Sec. 5.3.3.

In order to force the learning of scene structure prediction from two different views rather than relying solely on texture information, the photometric loss ($\mathcal{L}_{\text{SSIM}}$) [197], commonly used for optical flow estimation, is utilized. Given that TinyDepth generates two depth maps $(\tilde{\mathcal{D}}_1, \tilde{\mathcal{D}}_2)$ from two input frames and two L5 signals observed from $(\mathcal{C}_1, \mathcal{C}_2)$, it is necessary for the

consistency of the depth maps with respect to the input order. Let the depth prediction from the input pair $(\mathcal{C}_2, \mathcal{C}_1)$ be denoted as $(\tilde{\mathcal{D}}'_2, \tilde{\mathcal{D}}'_1)$. Consequently, a forward-backward consistency loss is employed to ensure that the depth prediction remains invariant regardless of the input pair order.

$$\mathcal{L}_{\text{fb}} = \sum_{i \in N} \left(\left| \tilde{\mathcal{D}}_{\mathcal{C}_1} - \tilde{\mathcal{D}}'_{\mathcal{C}_1} \right| + \left| \tilde{\mathcal{D}}_{\mathcal{C}_2} - \tilde{\mathcal{D}}'_{\mathcal{C}_2} \right| \right) \tag{5.3}$$

Furthermore, an RGB edge-aware smoothing regularization loss is incorporated to enforce local smoothness in depth prediction while preserving sharp depth boundaries. This regularization promotes the learning of similar depth values for adjacent pixels, except in the presence of image edges.

$$\mathcal{L}_{\text{egd}} = \sum_{i \in 1,2} \sum_{\mathbf{x} \in x,y} \left(\left| \partial_{\mathbf{x}} \tilde{D}_{C_i} \right| (1 - \partial_{\mathbf{x}} I_{C_i}) \right) \tag{5.4}$$

The edge map in the x-direction on the RGB image observed from $\mathcal{C}1$ is represented by $\partial_x IC_1$. Lastly, the network is trained by combining these loss functions with the total depth loss:

$$\mathcal{L}_{\text{total}} = \gamma_1 \mathcal{L}_r + \gamma_2 \mathcal{L}_{\text{SSIM}} + \gamma_3 \mathcal{L}_{\text{fb}} + \gamma_4 \mathcal{L}_{\text{egd}} \tag{5.5}$$

The values of $\gamma_1 = 0.8, \gamma_2 = 0.2, \gamma_3 = 0.2$, and $\gamma_4 = 0.01$ are determined based on different training regimes and cross-validations. The network is trained using AdamW [198] optimizer, with an initial learning rate of 0.001, β_1 set to 0.9, β_2 set to 0.999, and the training is performed on an Nvidia GeForce RTX 2070 SUPER GPU.

5.2.4 Evaluation Metrics

The performance of the depth estimation methods is evaluated by comparing them with existing approaches. The evaluation is based on error metrics, including Root Mean Square Error (RMSE), absolute relative error (REL), and the percentage of predicted pixels where the relative error is within a threshold (δ). The metrics can be defined as follows:

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i \in N} ||\tilde{\mathcal{D}}_i - \hat{\mathcal{D}}_i||} \tag{5.6}$$

$$\text{REL} = \frac{1}{N}\sum_{i \in N} \frac{|\tilde{\mathcal{D}}_i - \hat{\mathcal{D}}_i|}{\hat{\mathcal{D}}_i} \tag{5.7}$$

$$\delta_i = \frac{1}{N}\sum_{i \in N} \left(\max\left(\frac{\hat{\mathcal{D}}_i}{\tilde{\mathcal{D}}_i}, \frac{\tilde{\mathcal{D}}_i}{\hat{\mathcal{D}}_i} \right) < 1.25^i \right) \quad \forall i \in \mathbb{N} \tag{5.8}$$

In this context, the depth prediction $\tilde{\mathcal{D}}$ and the ground truth $\hat{\mathcal{D}}$ represent the respective values for pixel i. The total number of pixels in the $\tilde{\mathcal{D}}$ image is denoted by N. It should be noted that the values in $\tilde{\mathcal{D}}$ are in metric units and are not normalized. The evaluation and comparison of different methods based on these metrics are conducted in Section 5.3.3.

5.3 Experiments and Applications

In this section, the working of the TinyDepth and TinyDepth-S (smaller) models is discussed, and they are tested on out-of-domain test samples and real-world robotics experiments. The approach is demonstrated in comparison with hardware and software

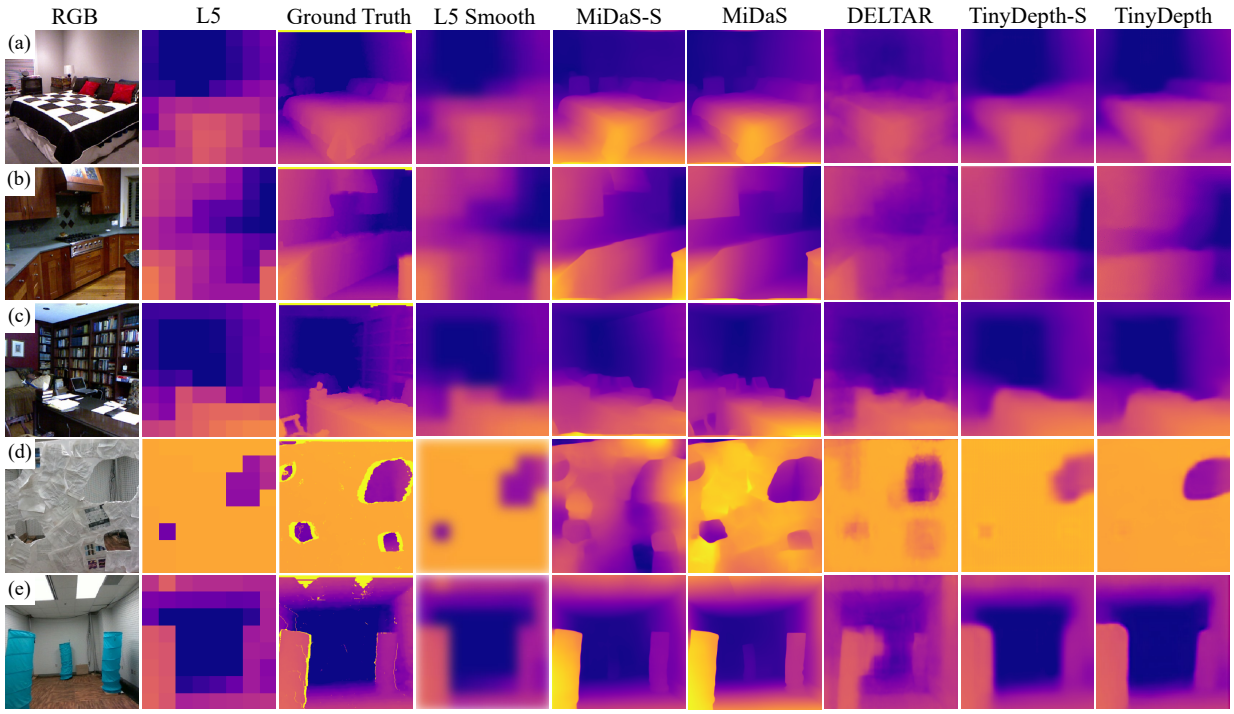

Figure 5.5: Quantitative evaluation on out-of-domain dataset: (a)-(c) NYUv2 out-of-domain samples, (d) GapFlyt data: Flying through unstructured gaps and (e) Indoor forest data for drone navigation. *Note that MiDaS/MiDaS-S use a single RGB image, DELTAR uses a single RGB + single L5 and TinyDepth uses two RGB and two L5 consecutive image pairs for depth predictions.*

industrial/research-grade standards such as Intel Realsense D435i[1] and Intel MiDaS [199], with the aim of reducing power consumption, weight, size, and cost significantly.

5.3.1 Experimental Setup

The proposed method is tested on two tiny custom robots designed for onboard autonomous navigation: (a) A Tiny Car equipped with front-facing depth sensors and (b) A Bee drone with sensing capabilities on all four faces of the drone (front, back, left, right), enabling obstacle avoidance in all directions.

[1] https://www.intelrealsense.com/depth-camera-d435i/

Table 5.1: Quantitative evaluation of different methods for metric depth estimation on out-of-domain datasets.

Method	Out-of-Domain† Testing				Runtime ↓	nParams ↓	ModelSize ↓
	RMSE ↓	REL↓	δ_1 ↑	δ_2 ↑	(ms)	(M)	(MB)
MiDaS `v3 DPT_Large 384`	–	–	–	–	314.1	344.055	1,300
MiDaS `v2.1 Small 256`	0.996	0.613	0.461	0.726	36.4	21.321	97
Comparisons with Metric Depth Methods							
VL53L5CX (smooth)	0.896	0.461	0.674	0.846	–	–	–
DELTAR	0.366	0.103	0.902	0.982	27.6	19.641	76
Sparse-to-Dense (64 samples)	0.784	0.184	0.771	0.924	104.9	63.563	636
TinyDepth (Ours) (l_2)	**0.363**	0.056	0.928	**0.983**	17.9	15.782	61
TinyDepth (Ours) (l_1)	0.534	**0.045**	**0.936**	0.961	17.9	15.782	61
TinyDepth (Ours) (l_{Char})	0.543	0.070	0.920	0.959	17.7	15.782	61
TinyDepth (Ours) (l_{berHu})	0.541	0.047	0.932	0.960	17.9	15.782	61
TinyDepth-S‡ (Ours) (l_1)	0.632	0.134	0.794	0.950	**4.3**	**0.156**	**0.644**

†Trained on NYUv2 [190] and tested on out-of-domain NYUv2 samples, WorldGen [20], SUN3D [200] and novel real-world robotics environments. ‡Quantized using TensorFlow 2 Dynamic Range Quantization.

5.3.1.1 Tiny Car

The credit card-sized autonomous car employed in this setup has dimensions of $70 \times 58 \times 32$ cm and a weight of merely 128g. For the execution of all depth prediction inferences and control policy, a single Raspberry Pi Compute Module 4 is utilized in conjunction with the Ochin Tiny Carrier Board for RPi CM4. The car incorporates an ST Microelectronics VL53L5CX Time-of-Flight (ToF) 8×8 multizone ranging sensor along with a Raspberry Pi camera v1.3. The combination of these sensors enables a typical field of view (FoV) of 45° in both horizontal and vertical directions (63° diagonal) for depth estimation. Additionally, the car is powered by a Tattu 2S 300mAh battery, which drives four 6V Pololu motors (with a 15:1 gear ratio) utilizing a DRV8835 motor driver. Each motor is connected to a 30mm diameter wheel.

5.3.1.2 Bee Drone

The Bee quadrotor has dimensions of $92 \times 92 \times 84$ cm (motor center to motor center) or a diagonal length of 118 cm. It has a weight of 278g, including a Tattu R-Line 750mAh 3S battery. To enable depth prediction and control policy, the Bee drone is equipped with two pairs of Raspberry Pi Compute Module 4 and the Ochin Tiny carrier board, which are mounted on the left and right faces of the drone. Each Raspberry Pi Module is connected to two pairs of RPi cameras and a VL53 sensor, allowing for depth perception in all four directions. The propulsion system consists of T-Motor F1404 4600KV motors paired with Gemfan Flash 2540x3 propellers. The lower-level attitude and position hold control is managed by the ArduPilot v4.3.6 firmware, which runs on the Holybro Kakute H7 Mini flight controller. This flight controller is equipped with a downward-facing GL9306 optical flow sensor for attitude estimation and a Benewake TFMini-S LIDAR as the altitude measurement source. The higher-level control and navigation processes are handled by a Raspberry Pi 4, which communicates with the flight controller using RC-Override in Loiter mode via MAVLink messages.

The Bee drone is employed to conduct two distinct navigation experiments: (a) GapFlyt [123], involving flying through unknown-shaped gaps, and (b) navigating through an indoor forest. In the GapFlyt experiment, gaps (or windows) with a maximum length of 2.5cm are placed within the environment. The average initial distance between the Bee drone and the gap is 2.2m. In the indoor forest environment, cylindrical play tunnels[2] with a diameter of 0.48m and height of 1.83m are utilized as obstacles for the Bee drone in an unstructured manner within a space measuring 4.5m by 2.3m. All experiments are conducted within the Perception and Robotics

[2] https://www.target.com/p/antsy-pants-play-tunnel/-/A-51735999

Group Lab and Robotics and Autonomy Lab at the University of Maryland, College Park.

5.3.2 Experimental Results

This section presents the demonstration of utilizing predicted depth TinyDepth models for autonomous navigation.

5.3.2.1 Autonomous Car Navigation

The autonomous car operates within an unstructured unknown scenario, as depicted in Fig. 5.4c. The objective of this experiment is to navigate towards a goal direction represented by the vector $\mathbf{v}_g$, which functions as a global vector. Moreover, a local vector $\mathbf{v}_l$ is derived through the segmentation of the depth map into depth-based binary maps that classify regions as either *safe* or *unsafe*. Only the top half of the image is utilized, as the bottom half is obscured by the ground or floor. Subsequently, $\mathbf{v}_l$ is defined as the geometric center of the largest blob within the *safe* region mask. By combining both the global and local direction vectors, the desired location $\mathbf{v}_d$ is determined through a weighted sum of the two vectors. The rationale behind this approach is that $\mathbf{v}_l$ assumes a greater influence on $\mathbf{v}_d$ when obstacles are in close proximity. This control policy draws inspiration from [133].

$\mathrm{v}_d = \gamma \mathbf{v}_l + (1 - \gamma \mathbf{v}_g)$(5.9) where γ is weighed by the closest obstacle distance Z_{min} in the depth map and is defined as

$$\gamma = \frac{1}{1 + e^{-\frac{1}{Z_{\text{min}}}}}; \gamma \in [0, 1] \tag{5.10}$$

Finally, a Proportional-Integrational-Derivative (PID) controller is employed to navigate

through the unstructured scene, and the equation is given by:

$$\mathbf{u}(t) = K_p \mathbf{e}(t) + K_i \int_0^{\tau} \mathbf{e}(t) d\tau + K_d \frac{d\mathbf{e}(t)}{dt} \tag{5.11}$$

where $\mathbf{e}(t) = \mathbf{v}d(t) - \mathbf{v}\text{curr}(t)$ and where $\mathbf{v}_{\text{curr}}(t)$ represents the current heading direction. The control policy remains in effect until the car reaches its designated goal. The autonomous driving control policy is showcased by achieving a success rate of 90

5.3.2.2 Flying Through Unknown Gaps

This scenario resembles the one described in GapFlyt [123]. In this experiment, a drone is positioned in front of a prominent foreground element containing one or more gaps or holes that it must navigate through (refer to Fig. 5.4b). The per-pixel depth, estimated by the TinyDepth network, can be represented as a univariate bimodal Gaussian distribution. Similar to [123], the contour $\mathcal{C}$ is defined as the outline of the largest gap or opening on the image plane. Depth calculation involves capturing the environment from two different viewpoints through a small-baseline translation motion. The resulting depth map is then used to segment $\mathcal{C}$. Subsequently, the foreground $\mathcal{F}$ and background $\mathcal{B}$ are defined as the pixels outside and inside $\mathcal{C}$, respectively. Similar to [123], simultaneous tracking of both $\mathcal{F}$ and $\mathcal{B}$ is performed. The control policy aims to align the center of the image with the safest point $\mathbf{x}_s$, which is defined as follows:

$$\mathbf{x}_s = \begin{cases} \mathbb{M}(\mathcal{F}) & \text{if } \overline{\overline{\mathcal{F}}} \geq \overline{\overline{\mathcal{B}}} \\ \mathbb{M}(\mathcal{B}) & \text{otherwise} \end{cases} \tag{5.12}$$

where $\mathbb{M}$ denotes median operator and $\overline{\overline{\mathcal{F}}}$ denotes the number of pixels in the foreground mask. Unlike [123], our method can know if the traversal through unknown gaps is feasible before going through the gap since TinyDepth returns metric depth as opposed to a scaled or ordinal depth map. Furthermore, our approach also provides information if the quadrotor has successfully traversed through the gap that solves the major limitation of [123]. Furthermore, we downsized the quadrotor by more than $5\times$ in the net area and $3\times$ in the weight in flying through unknown gaps experiments as compared to [123, 133]. Note that in Fig. 5.4b, there are multiple gaps on $\mathcal{F}$. Our method selects the largest gap possible for maximizing traversal safety. We obtain a success rate of 84% over 25 trials with different-shaped window configurations with both single and multiple windows with a minimum tolerance of 24mm.

5.3.2.3 Flying in Unstructured Environments

In this scenario, the quadrotor tackles the problem of navigation without colliding in an unknown forest. In this setup, the drone is equipped with the TinyDepth sensor suite in all four directions as mentioned in Sec. 5.3.1.2. Let the sparse depth from each L5 sensor in metric units be denoted as $\mathbf{L}^f, \mathbf{L}^b, \mathbf{L}^l$, and $\mathbf{L}^r$ assigned in the forward, backward, left, and right directions, respectively. Also, $\mathbf{L}_{\text{min}}$ represents the minimum depth on the sparse depth map. A similar strategy as mentioned in Sec. 5.3.2.1 is deployed with modifications in the control policy.

In contrast to the *Tiny Car* navigation approach that relies on yaw motion due to its differential drive setup, the bee drone employs roll, pitch, and throttle to maneuver and avoid obstacles in all directions. To optimize power consumption and velocity, depth inference $\tilde{\mathcal{D}}^i$ is performed solely in the direction i where

$$\min\{i \mid \mathbf{L}^i_{\text{min}} \; \forall i \in \{f, b, l, r\}\} \tag{5.13}$$

Now, $\mathbf{v}_l$ is computed using $\tilde{\mathcal{D}}^i$ similar to Section 5.3.2.1 and employs the same PID controller and weighing schemes for global and local/safe direction. It is noteworthy that the drone demonstrates the ability to navigate through the scene as depicted in Figure 5.4a, even when the right cylinder lies outside the Field of View (FoV) of the front-facing sensor suite. The bee drone successfully traverses various unknown gaps at an average speed of 2.7m sec^{-1}, achieving a success rate of 88% across 25 trials conducted with different configurations.

5.3.3 Evaluations

Given that the only other existing work that employs L5 signals and color images for dense depth prediction is DELTAR [22], our approach is also evaluated alongside monocular depth estimation methods and Sparse-to-Dense [201], which uses RGB images and 64 spatially random depth samples from Kinect data to infer depth. In this section, the evaluation of TinyDepth with these methods will be conducted, encompassing both qualitative and quantitative assessments, while also comparing the outcomes obtained using different loss functions employed by TinyDepth.

The model is trained using the NYU-Depth v2 dataset [190] for training and evaluation on out-of-domain samples from NYUv2, SUN3D [200], WorldGen [20], and real-world robotics environments. The evaluation is reported on the metrics described in Sec. 5.2.4. It is important to note that the method is not retrained or fine-tuned for evaluating on out-of-domain samples. Fig. 5.5 provides a qualitative comparison of the method with other existing methods. The method

demonstrates the most accurate depth maps with metric precision. Notably, in the samples shown in Fig. 5.5b and e, cleaner depth edges are inferred. Additionally, monocular depth estimation methods (such as MiDaS [199]) can generate sharp object boundaries but are susceptible to errors in never-before-seen examples, as shown in Fig. 5.5d, due to their reliance on a single view without depth prediction constraints. Furthermore, these methods predict the scaled depth and are less suitable for robotics applications compared to TinyDepth and DELTAR [22].

TinyDepth-S is a variant of the TinyDepth model with a reduced number of encoder-decoder blocks, quantized using TensorFlow's dynamic range quantization to achieve a runtime of 4.5Hz on a RaspberryPi 4 CPU. Table 5.1 provides a comprehensive quantitative evaluation of the aforementioned methods, including their runtime (in ms) on NVIDIA RTX 2070 SUPER, number of parameters, and model size. Notably, the larger depth prediction model used in this work is $21\times$ and $10\times$ smaller than the MiDaSv3 `DPT_Large 384` model and sparse-to-dense [201] model, respectively. Furthermore, our method's model size for depth prediction is $120\times$ smaller compared to DELTAR [22], which utilizes the same sensor suite. This size reduction enables the execution of our method on single-core embedded devices.

5.4 Discussions and Limitations

The method presented demonstrates an alternative to industrial-grade sensors such as Intel Realsense D435i for resource-constrained robots. The sensor suite employed consumes only a fraction of power, approximately $\sim 12\times$ and $\sim 7\times$ less compared to D435i and Apple iPhone LiDAR, respectively. It also has a reduced extension by a factor of $3\times$ in the longest dimension and occupies a significantly smaller volume (by $3-4$ orders of magnitude) when compared to

D435i. Furthermore, the chosen sensor is approximately $37\times$ cheaper than D435i, making it suitable for small, energy-efficient, and cost-effective robots. However, it should be noted that one drawback of the VL53L5CX sensors is their limited operating frequency, working only at 15Hz for 8×8 and 60Hz for 4×4 zone resolution. This limitation hampers the TinyDepth performance for dynamic obstacle avoidance (e.g., [130]) on smaller drones. Another significant limitation of using ToF supervision for depth prediction is its performance degradation when dealing with non-Lambertian surfaces compared to monocular depth estimation methods. We observe that a majority of the L5 signals are either invalid or result in largely incorrect metric scales when encountering non-Lambertian surfaces. Additionally, our method is trained on a relatively small baseline between two views to ensure a low model size and inference time for stable depth prediction. However, the method does not achieve very sharp boundaries compared to MiDaS [199]. We aim to address this limitation by incorporating convex upsampling techniques introduced by [1], even though it would significantly increase the inference time.

5.5 Conclusion

In this work, a resource-constrained depth prediction solution is successfully demonstrated, enabling autonomy in palm-sized robots by combining information from RGB and lightweight sparse ToF sensors. The results are compared and contrasted on never-before-seen samples and tested on real-world robotics experiments, achieving a combined accuracy of 87% for sizes less than 3 inches, where industrial-grade depth sensors like Intel Realsense are not feasible. It is strongly believed that these solutions will create new opportunities for mobile robot autonomy.

This page intentionally left blank.

Chapter 6: Conclusions

The field of robotics has undergone extensive research over the past 4-5 decades, encompassing areas such as Artificial Intelligence, Computer Vision, Embedded Systems, Robot Modelling, Dynamics, Planning, Cognitive Systems, and Collaborative Homogeneous and Heterogeneous Systems, among others. Particularly for small and resource-constrained robots, there arises a necessity to reevaluate mobile robots starting from the foundational level, spanning from sensor-level considerations to the cognitive level. In order to address robotics challenges

Figure 6.1: Onboard Autonomy on Tiny Robots: An Outcome of Minimal Perception

Figure 6.2: Flower Detection of a Downfacing Camera on the RoboBee

more effectively, it is crucial to perceive robots as embodied agents. This entails fostering strong synergy not only in hardware and software co-design but also across various disciplines or subfields within the realm of robotics.

This thesis focuses on utilizing minimalism as the fundamental approach to tackle the challenge of achieving onboard autonomy in robots of unprecedented sizes and scales. The thesis culminates in the realization of onboard autonomy in robots as small as credit cards, showcasing their autonomous functionalities in previously unexplored and unstructured environments (See Fig. 6.1).

Specifically, the importance of minimal cues required to solve navigation tasks, such as flying through forests, indoor spaces, or small gaps, is demonstrated. The notion of uncertainty principles in quantities such as optical flow is demonstrated to be sufficient for navigation tasks, eliminating the need for complete optical flow computation. The utilization of optical flow uncertainty for various tasks, including navigation, flying through unknown gaps, dynamic obstacle avoidance, and segmentation, is showcased.

Furthermore, an extensive study is conducted on the importance of learning to predict the

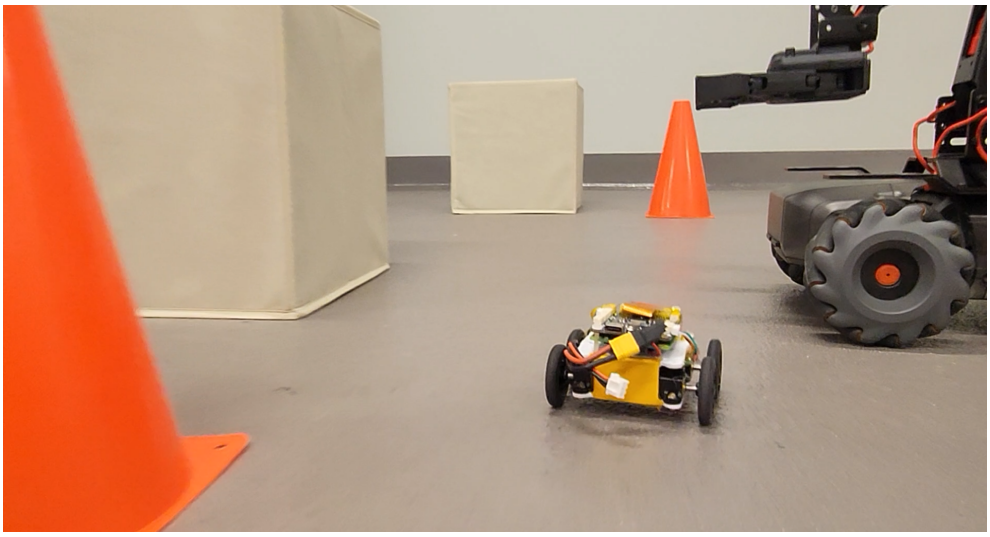

Figure 6.3: Autonomous onboard obstacle avoidance on a credit-card size robot

structures of the scene by observing the environment from two different viewpoints instead of relying solely on a single image for prediction. This is particularly crucial when dealing with out-of-domain samples or novel scenes, as it avoids overfitting on the textures of the testing data and emphasizes the need to learn a model that can accurately predict the underlying structure.

To achieve generalizability in these prediction models, a significant amount of high-quality automatic data generation is required. The presented work in the WorldGen simulator demonstrates how such data can be utilized to improve the learning of optical flow models solely in simulation, with direct applicability to the real world. This approach eliminates the need for fine-tuning or retraining the neural network models on new datasets.

Additionally, the study explores TinyDepth models that leverage a traditional RGB camera alongside super sparse 64 pixels depth sensors. These models predict dense depth maps of the scene by observing the scene from two different images, even in novel scenes. This model plays a pivotal role in enabling the autonomy framework for robots that are less than 3 inches in length, operating at approximately 4.2Hz (See Fig. 6.3). Furthermore, the discussion includes large-scale

swarm research, such as the RoboBeeHive (Fig. 6.2), which facilitates heterogeneous autonomy behaviors on hummingbird-size robots with depth estimation in all four directions.

The discussion includes the utilization of a moving element positioned ahead of an active sensor, such as neuromorphic sensors (or event cameras), to attain performance surpassing what can be achieved with either a standard RGB camera or an event camera alone. Moreover, the shapes and sizes of the apertures hold significance in effectively filtering the incoming visual data. The creation of tailored apertures for specific tasks plays a crucial role in realizing resource-efficient robots based on the principles of minimal perception.

As a parting thought, rethinking robots as embodied agents from the ground up – from cognition to the sensor level and developing an autonomy framework for specific tasks based on their size, power, weight, and compute constraints by utilizing the minimal software and hardware capabilities holds the key to push the boundaries of onboard autonomy on tiny size robots.

This page intentionally left blank.

Chapter 7: Future Directions

This chapter discusses the *blue* part of the framework as presented in Fig. 1.6.

7.1 Passive Computing – Modifying Camera Apertures

Over the course of 3.8 billion years, nature has undergone extensive research and development to optimize the sensory capabilities of animals for their specific tasks and survival. Different species, such as dragonflies and hummingbirds, employ distinct mechanisms and techniques to perceive their environments, yet both are capable of performing similar tasks in the wild. Nature has provided a diverse range of sensing paradigms, including circular eyes in humans, vertical and horizontal aperture eyes in frogs and bobcats, and compound eyes in jumping spiders and dragonflies. Fig. 7.1a shows different kinds of aperture found in nature. The selection of sensor design and cognitive architecture becomes crucial for the development of efficient and resource-conscious systems.

It may be feasible to build 3D maps, and deploy planning and control algorithms on larger systems but smaller agents are highly constrained. One may argue – "*Why not adapt or downscale existing algorithms from larger autonomous systems?*" The problem is not as simple as it sounds. There is no trivial way of downscaling perception and planning algorithms. Nature has taken ages to solve the optimization problem between hardware (sensing) and software

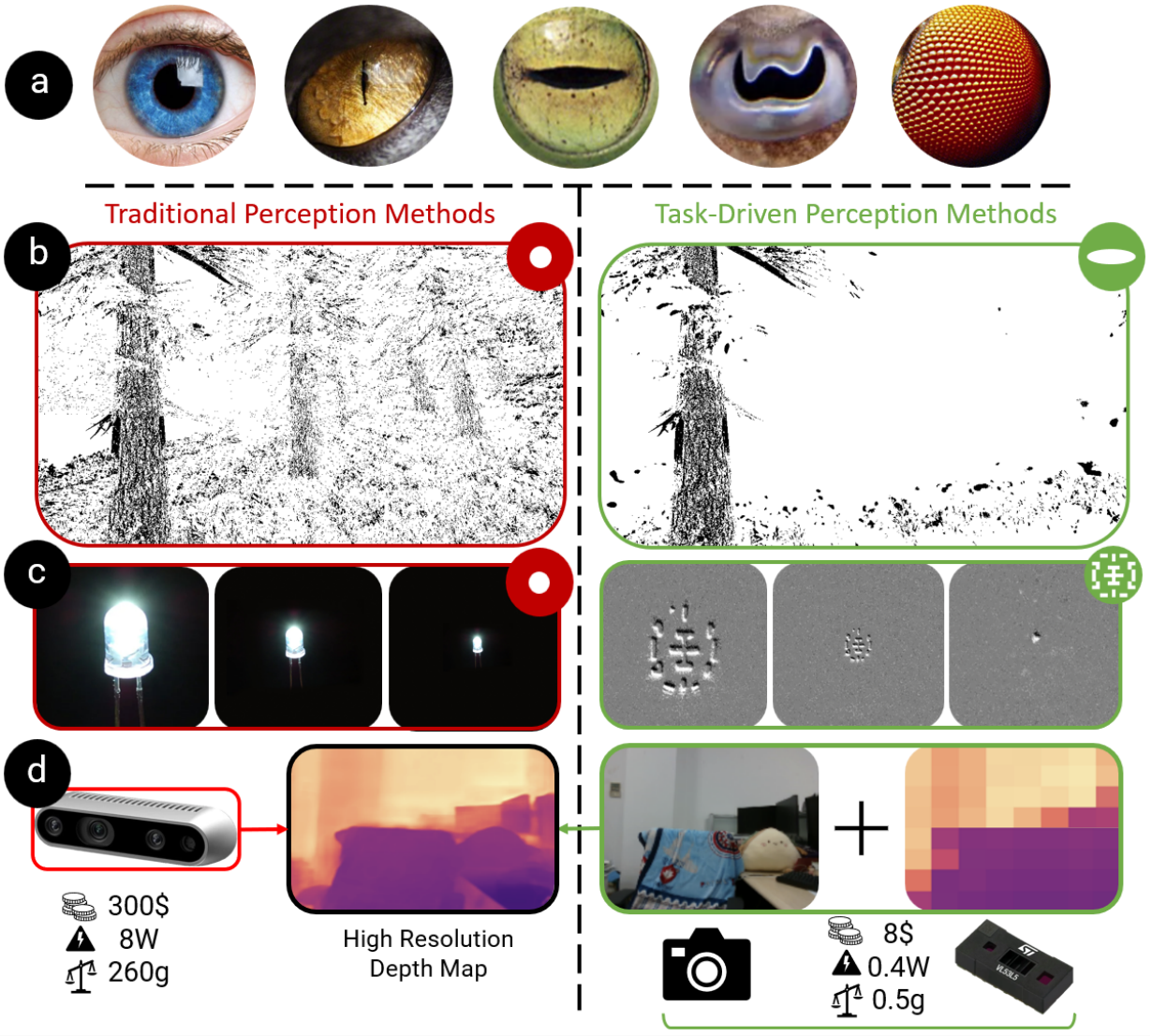

Figure 7.1: Various Apertures in the wild

(neural architecture). This *genetic* evolution took more than half a million years to evolve from stage 1 eye (non-directional photoreceptors) to stage 4 eye (high-resolution vision – lens formation) [202]. To exacerbate this problem further, the perception algorithm needs to keep up with the control loops in terms of speed for the autonomous systems to react, especially for smaller systems due to high agility. Thus, there is a need to re-imagine the entire perceptual system – both sensing and computing from the ground up.

Active or traditional computing methods require electric power to process information.

Breaking away from this idea by introducing a novel approach – *passive computing* that utilizes passive components for modulation of the input signal to simplify tasks. It is known as *passive* because these components do not require any electric energy to operate. The fundamental idea behind this approach is to offload a significant percentage of computation at the sensing level. These passive physical structures are optimally designed for a particular sensor to perform a specific task. For instance, for visual sensors, utilizing a wider aperture ensures a very shallow depth of field.

A conventional camera captures blurred versions of scene information away from the plane of focus. Similarly, an event camera produces no events in blurred regions due to a lack of contrast in those areas. The physics model of the *blur pattern* utilizes a Point Spread Function (PSF) to gather more information about the scene. Over the past two decades, extensive research has been conducted on predicting depth from RGB images using coded apertures [151, 203, 204]. Additionally, [205] proposes depth estimation from a single event camera, although its generalization across datasets is limited. Another approach presented in [206] introduces an active light sensor to aid depth estimation with event cameras. A simple passive modification can be made for the camera aperture to recover depth information from a single event camera. This involves inserting a known patterned camera aperture inside the event camera. The method aims to predict high-speed depth estimation using coded aperture event cameras.

To downscale robotic systems by a factor of 10 while maintaining the same level of autonomy, a fundamental reevaluation of the entire perception stack is required. By altering camera apertures, it is possible to extract previously inaccessible scene information that traditional camera designs cannot capture. The use of asymmetrical sensor apertures can enable the development of parsimonious and task-driven robotic systems. These passive asymmetrical

apertures allow for the segmentation of information prior to its capture by the sensor, thereby reducing computational load. In the case of drone navigation, a reduction in computational load can be achieved by leveraging horizontal or vertical or any other custom apertures. In Fig. 7.1b, the output from a neuromorphic camera is depicted, observed through both a circular aperture (on the left) and a horizontal aperture (on the right). It is noteworthy that the horizontal aperture effectively filters out background noise resulting from directional geometrical blur caused by the aperture. This reduction is accomplished by leveraging the principles of light wave behavior and its interference with the camera aperture. Additionally, specific aperture shapes are encoded in event cameras to enable the detection and tracking of moving robots, such as drones, in three-dimensional space using a single monocular-coded aperture event camera. Fig. 7.1c illustrates how the size of a light source varies with the distance of an object in the event space. The estimation of metric depth from a single camera, whether it is an RGB or neuromorphic camera, can be achieved using coded apertures from a single image [203].

7.1.1 Non Visible Spectral Sensing

The ability of bees to see ultraviolet (UV) light is a remarkable example of biological spectral perception. Unlike humans, who perceive light across a spectral range of roughly 400-700 nm, bees have evolved a vision system that extends into the ultraviolet range, which spans approximately 10-400 nm.

This difference in visual perception is a result of differing photoreceptor sensitivities. Humans possess three types of photoreceptors (or cone cells) sensitive to short (S), medium (M), and long (L) wavelength light, corresponding roughly to blue, green, and red. Bees, on the other

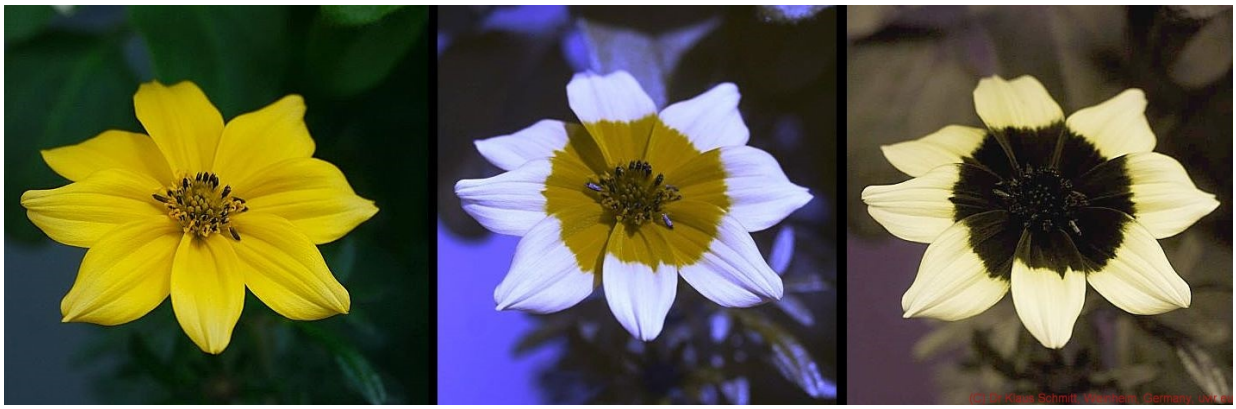

Figure 7.2: Images of Flower: (a) RGB, (b) Simulated Bee Vision and (c) UV Space

hand, possess photoreceptors sensitive to ultraviolet, blue, and green wavelengths. The bees' ultraviolet photoreceptor can detect wavelengths in the range of 300-400 nm, beyond human perceptual capabilities.

The ability to see UV light plays a crucial role in how bees detect and segment flowers. Many flowering plants have evolved patterns called 'nectar guides' or 'honey guides' that are only visible in UV light. These patterns are essentially targets that guide bees to the nectar and pollen sources within the flower. By visualizing these UV patterns, bees can efficiently locate, segment, and forage from flowers in a field. The UV vision thus enhances the bees' foraging efficiency and the plant's reproductive success through effective pollination.

Fig. 7.2 shows an image of a flower in broad daylight in RGB, simulated bee vision and UV spectrum (from left to right). (Source)[1]

Ultraviolet (UV) sensing proves essential for the functionality and efficiency of tiny robots, or microbots, by enriching their vision capabilities and aiding in object recognition similar to bees identifying and segmenting flowers. UV markers, invisible to human sight but perceptible to robots equipped with UV sensors, can guide these microbots along specified

[1] http://forum.mflenses.com/bidens-flower-in-human-vs-bee-vision-t48996.html

paths, enhancing their navigation and localization abilities. In the realm of healthcare, UV sensing allows detection of certain microscopic organisms or substances that fluoresce under UV light, presenting significant potential for monitoring harmful bacteria or body fluid analyses. Furthermore, in swarm robotics, UV light can facilitate undetectable communication amongst the microbots, allowing covert information exchange. Additionally, the use of UV light minimizes visual disturbances in human-populated environments due to its invisibility to the human eye. Lastly, the energy efficiency of UV sensors, especially those made of materials like zinc oxide or titanium dioxide, is a critical benefit for microbots that typically operate with limited power resources. This unique combination of advantages highlights the pivotal role of UV sensing in enhancing microbot capabilities, consequently expanding their applications and improving their overall effectiveness.

UV sensors, particularly those based on certain materials like zinc oxide or titanium dioxide, can be highly energy-efficient. This is a critical advantage for tiny robots, which have limited power resources. One of the main reasons UV or in general multispectral sensing is important in robotics is that it allows for better object recognition and scene understanding. Just like bees use UV sensing to discriminate flowers, multispectral sensing allows robots to segment and differentiate objects based on their spectral signatures. This is especially useful in applications like precision agriculture, where multispectral imaging can be used to identify crop diseases, estimate soil moisture levels, and even classify different species of plants.

Another crucial reason for the importance of multispectral sensing in robotics is its ability to function under varied lighting conditions. Robots operating outdoors must be able to adjust to changes in light conditions due to weather, time of day, and shadows. With multispectral sensing, robots can continue to identify and interact with their environment effectively, even when lighting

conditions change dramatically.

All in all, UV sensors will make the perceptions systems much cheaper, both algorithmically and in terms of power for purposes such as robot pollination.

7.2 Leveraging From Active Elements In Front Of The Sensor

Neuromorphic vision sensors, also known as event cameras, enable visual perception with extremely low reaction time in microseconds, thereby facilitating high-dynamic applications in robotics. However, these sensors face challenges where their outputs are influenced by both motion and texture. Specifically, they are unable to detect edges parallel to image motion and textures gradually fade when motion is minimal. This inherent problem in event cameras poses difficulties in finding algorithmic solutions.

In contrast, human vision utilizes small involuntary eye movements, known as microsaccades, to overcome perceptual fading during fixation. By constantly and subtly moving the eyes, microsaccades effectively maintain texture stability and persistence. Inspired by this mechanism, an event-based perception system has been developed that combines low reaction time with stable texture preservation.

In the design, an active rotating wedge prism is incorporated in front of the aperture of an event camera [21]. This prism redirects light and triggers events. Leveraging the principles of geometrical optics, the rotational motion introduced by the prism is algorithmically compensated, resulting in a stable texture appearance and high informational output irrespective of external motion. This hardware and software integration forms the foundation of the Artificial Microsaccade-enhanced Event Camera (AMI-EV) system.

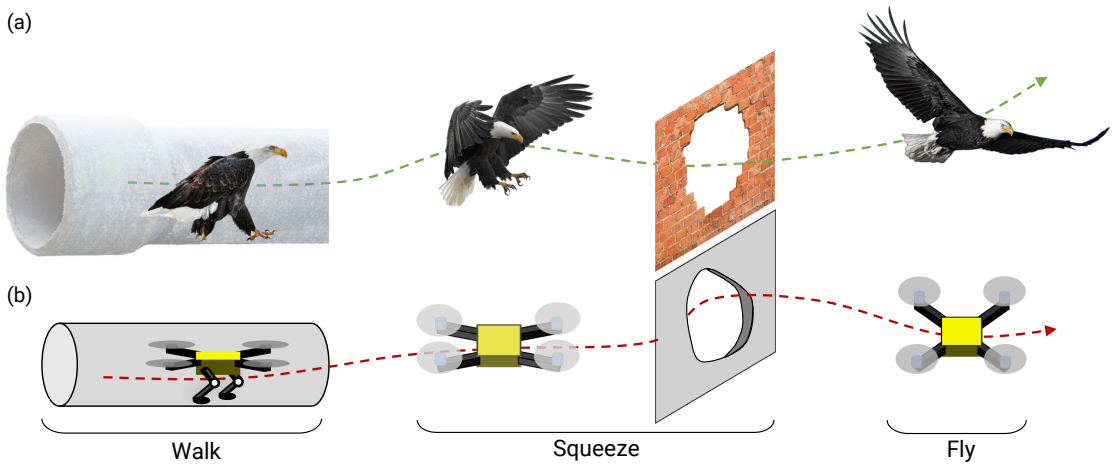

Figure 7.3: (a) Bald eagle seamlessly changing its state from walking to squeezing to flying. (Left to Right). (b) Our Morphing Quadrotor Prototype walking, squeezing and switching to flight mode. (Left to Right).

7.3 Towards Robot Morphology Design

In order to maximize the efficiency, a parsimonious robot system must be well thought based on what task needs to be completed.

The fatalities that humanity has faced due to natural disasters, building collapses and mining accidents have increased in the last decade. Deaths and mayhem can be avoided if the location of the trapped people and safe routes are estimated prior to the rescue phase. It is a race against time! A dearth of oxygen levels and blood loss can be fatal. Furthermore, the traditional method of searching for the trapped people puts other lives in jeopardy. To ameliorate these problems, researchers and safety groups have deployed aerial robots, ground vehicles and snake robots. Although, these robots are not able to reach the majority of areas due to different physical constraints. So, to overcome these barriers, there is a need to develop an autonomous drone that can walk, squeeze and fly to navigate through these challenging unknown environments.

The human fascination of nature's masters of flight have led researchers across the world

to pursue the development of morphable drones. Birds seamlessly modify their wing-span while flying and transit into a walking phase whenever necessary (Fig. 7.3a). They do this by not only sensing the environment (*exteroception*) but also by sensing their own body limbs' position and orientation (*proprioception*). Inspired by this, a morphing drone can change its shape, size or switch into a walking phase in order to navigate through narrow openings or flight-denied space (Fig. 7.3b). In the past few years, there has been advances [207–212] in morphology of a drone. Although, they have been restricted to a control problem but can be coupled with a perception problem by estimating the location and the shape of an unknown tunnel using a front-facing camera. The choice of navigation ontology: walk, morph (squeeze) or fly is governed by three critical factors – risk, time and energy.

Specifically, the problem that needs to be address is: "*What is the minimum Cost of Transportation (CoT) required for a morphing drone to navigate through an unknown environment?*" To enable the traversal through unstructured or collapsed buildings, there is a need to develop an environment-aware drone with flying, walking and morphing capabilities. Morphing is desired for traversal of small gaps through flight for its speed and agility whereas walking is desired where flying is not possible in order to save *time*, *energy* and reduce the *risk factor*. For each critical factor, CoT is computed. These CoTs will be computed for exploration (to gather more information about the environment), morphing, switching (between states), computation and sensing tasks in terms of the aforementioned critical factors. The drone perceives the visual information and then classifies the environment into a set of motifs, for example – open space, gap, tunnel etc. For each of these motifs, CoT for all the navigation ontologies are pre-computed. Although, this pre-computation of CoT is an arduous task. To robustly estimate CoT for each for the motif for all navigation ontologies, one would require an

enormous amount of data. So, to overcome this challenge, estimatation CoTs by generating a large number of random scenarios in a simulation environment. Based on the estimates from simulation, one can formulate a artificial intelligence model that can infer CoTs for an unknown real world scene. Specifically, developing a mathematical formulation that minimizes the overall CoT by predicting the future states of the robot depending on the perception, planning and control constraints. To exacerbate this scenario further, all the computation and sensing has to be performed on the drone itself as the communication between the drones and the servers are often compromised in disaster scenarios.

This page intentionally left blank.

Bibliography

[1] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *European conference on computer vision*, pages 402–419. Springer, 2020.

[2] Chethan M Parameshwara, Nitin J Sanket, Chahat Deep Singh, Cornelia Fermüller, and Yiannis Aloimonos. 0-mms: Zero-shot multi-motion segmentation with a monocular event camera.

[3] Nitin Jagannatha Sanket. *Active Vision Based Embodied-AI Design For Nano-UAV Autonomy*. PhD thesis, 2021.

[4] Noam Chomsky. A minimalist program for linguistic theory. 1993.

[5] John Aloimonos, Isaac Weiss, and Amit Bandyopadhyay. Active vision. *International journal of computer vision*, 1(4):333–356, 1988.

[6] Ruzena Bajcsy, Yiannis Aloimonos, and John K Tsotsos. Revisiting active perception. *Autonomous Robots*, 42(2):177–196, 2018.

[7] Jeannette Bohg, Karol Hausman, Bharath Sankaran, Oliver Brock, Danica Kragic, Stefan Schaal, and Gaurav S Sukhatme. Interactive perception: Leveraging action in perception and perception in action. *IEEE Transactions on Robotics*, 33(6):1273–1291, 2017.

[8] Lars Chittka. *The mind of a bee*. Princeton University Press, 2022.

[9] Nitin J Sanket, Chahat Deep Singh, Kanishka Ganguly, Cornelia Fermüller, and Yiannis Aloimonos. Gapflyt: Active vision based minimalist structure-less gap detection for quadrotor flight. *IEEE Robotics and Automation Letters*, 3(4):2799–2806, 2018.

[10] E. Ilg et al. Flownet 2.0: Evolution of optical flow estimation with deep networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, 2017.

[11] Jeannette Bohg, Karol Hausman, Bharath Sankaran, Oliver Brock, Danica Kragic, Stefan Schaal, and Gaurav S Sukhatme. Interactive perception: Leveraging action in perception and perception in action. *IEEE Transactions on Robotics*, 33(6):1273–1291, 2017.

[12] Chahat Deep Singh, Nitin J. Sanket, Chethan M. Parameshwara, Cornelia Fermüller, and Yiannis Aloimonos. Nudgeseg: Zero-shot object segmentation by repeated physical interaction. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2714–2712, 2021.

[13] Danica Kragic, Joakim Gustafson, Hakan Karaoguz, Patric Jensfelt, and Robert Krug. Interactive, collaborative robots: Challenges and opportunities. In *IJCAI*, pages 18–25, 2018.

[14] Chahat Deep Singh, Nitin J Sanket, Chethan M Parameshwara, Cornelia Fermüller, and Yiannis Aloimonos. Nudgeseg: Zero-shot object segmentation by repeated physical interaction. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2714–2712. IEEE, 2021.

[15] Martin Egelhaaf, Norbert Boeddeker, Roland Kern, Rafael Kurtz, and Jens Peter Lindemann. Spatial vision in insects is facilitated by shaping the dynamics of visual input through behavioral action. *Frontiers in neural circuits*, 6:108, 2012.

[16] Tonci Novkovic, Remi Pautrat, Fadri Furrer, Michel Breyer, Roland Siegwart, and Juan Nieto. Object finding in cluttered scenes using interactive perception. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 8338–8344. IEEE, 2020.

[17] Berk Calli, Arjun Singh, Aaron Walsman, Siddhartha Srinivasa, Pieter Abbeel, and Aaron M Dollar. The ycb object and model set: Towards common benchmarks for manipulation research. In *2015 international conference on advanced robotics (ICAR)*, pages 510–517. IEEE, 2015.

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

[19] Alexander Kirillov, Yuxin Wu, Kaiming He, and Ross Girshick. Pointrend: Image segmentation as rendering. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9799–9808, 2020.

[20] Chahat Deep Singh, Riya Kumari, Cornelia Fermüller, Nitin J Sanket, and Yiannis Aloimonos. Worldgen: A large scale generative simulator. *arXiv preprint arXiv:2210.00715*, 2022.

[21] Botao He, Ze Wang, Yuan Zhou, Jingxi Chen, Chahat Deep Singh, and Fei Gao. A texture-enhanced event camera using rotating wedge prism.

[22] Yijin Li, Xinyang Liu, Wenqi Dong, Han Zhou, Hujun Bao, Guofeng Zhang, Yinda Zhang, and Zhaopeng Cui. Deltar: Depth estimation from a light-weight tof sensor and rgb image. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part I*, pages 619–636. Springer, 2022.

[23] John Denker and Yann LeCun. Transforming neural-net output levels to probability distributions. *Advances in neural information processing systems*, 3, 1990.

[24] David JC MacKay. A practical bayesian framework for backpropagation networks. *Neural computation*, 4(3):448–472, 1992.

[25] Radford M Neal. *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media, 2012.

[26] Yarin Gal and Zoubin Ghahramani. Bayesian convolutional neural networks with Bernoulli approximate variational inference. In *4th International Conference on Learning Representations (ICLR) workshop track*, 2016.

[27] Yarin Gal et al. Uncertainty in deep learning. 2016.

[28] Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 5580–5590, Red Hook, NY, USA, 2017. Curran Associates Inc.

[29] Jochen Gast and Stefan Roth. Lightweight probabilistic deep networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3369–3378, 2018.

[30] Antonio Loquercio, Mattia Segu, and Davide Scaramuzza. A general framework for uncertainty estimation in deep learning. *IEEE Robotics and Automation Letters*, 5(2):3153–3160, 2020.

[31] Moloud Abdar, Farhad Pourpanah, Sadiq Hussain, Dana Rezazadegan, Li Liu, Mohammad Ghavamzadeh, Paul Fieguth, Xiaochun Cao, Abbas Khosravi, U. Rajendra Acharya, Vladimir Makarenkov, and Saeid Nahavandi. A review of uncertainty quantification in deep learning: Techniques, applications and challenges. *Information Fusion*, 76:243–297, 2021.

[32] Wenxin Liu, David Caruso, Eddy Ilg, Jing Dong, Anastasios I Mourikis, Kostas Daniilidis, Vijay Kumar, and Jakob Engel. Tlio: Tight learned inertial odometry. *IEEE Robotics and Automation Letters*, 5(4):5653–5660, 2020.

[33] Lei Zhou, Zixin Luo, Tianwei Shen, Jiahui Zhang, Mingmin Zhen, Yao Yao, Tian Fang, and Long Quan. Kfnet: Learning temporal camera relocalization using kalman filtering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4919–4928, 2020.

[34] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

[35] Sadegh Rabiee and Joydeep Biswas. Ivoa: Introspective vision for obstacle avoidance. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1230–1235. IEEE, 2019.

[36] Di Feng, Lars Rosenbaum, Fabian Timm, and Klaus Dietmayer. Leveraging heteroscedastic aleatoric uncertainties for robust real-time lidar 3d object detection. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, pages 1280–1287. IEEE, 2019.

[37] Serin Lee, Vincenzo Capuano, Alexei Harvard, and Soon-Jo Chung. Fast uncertainty estimation for deep learning based optical flow. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 10138–10144. IEEE, 2020.

[38] Jun-Gu Kang, Si-Dong Roh, and Ki-Seok Chung. Flownetu: Accurate uncertainty estimation of optical flow for video object detection. In *2021 4th International Conference on Artificial Intelligence and Pattern Recognition*, pages 36–41, 2021.

[39] Eddy Ilg, Ozgun Cicek, Silvio Galesso, Aaron Klein, Osama Makansi, Frank Hutter, and Thomas Brox. Uncertainty estimates and multi-hypotheses networks for optical flow. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 652–667, 2018.

[40] Shunkai Li, Xin Wu, Yingdian Cao, and Hongbin Zha. Generalizing to the open world: Deep visual odometry with online adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13184–13193, 2021.

[41] Weihao Yuan, Yazhan Zhang, Bingkun Wu, Siyu Zhu, Ping Tan, Michael Yu Wang, and Qifeng Chen. Stereo matching by self-supervision of multiscopic vision. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5702–5709. IEEE, 2021.

[42] Gwangbin Bae, Ignas Budvytis, and Roberto Cipolla. Estimating and exploiting the aleatoric uncertainty in surface normal estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 13137–13146, 2021.

[43] Barbara Roessle, Jonathan T Barron, Ben Mildenhall, Pratul P Srinivasan, and Matthias Nießner. Dense depth priors for neural radiance fields from sparse input views. *arXiv preprint arXiv:2112.03288*, 2021.

[44] Dhaivat Bhatt, Kaustubh Mani, Dishank Bansal, Krishna Murthy, Hanju Lee, and Liam Paull. f-cal: Calibrated aleatoric uncertainty estimation from neural networks for robot perception. *arXiv preprint arXiv:2109.13913*, 2021.

[45] Ricardo Martin-Brualla, Noha Radwan, Mehdi SM Sajjadi, Jonathan T Barron, Alexey Dosovitskiy, and Daniel Duckworth. Nerf in the wild: Neural radiance fields for unconstrained photo collections. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7210–7219, 2021.

[46] Abdelrahman Eldesokey, Michael Felsberg, Karl Holmquist, and Michael Persson. Uncertainty-aware cnns for depth completion: Uncertainty from beginning to end. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12014–12023, 2020.

[47] Matteo Poggi, Filippo Aleotti, Fabio Tosi, and Stefano Mattoccia. On the uncertainty of self-supervised monocular depth estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3227–3237, 2020.

[48] Gabriele Costante and Michele Mancini. Uncertainty estimation for data-driven visual odometry. *IEEE Transactions on Robotics*, 36(6):1738–1757, 2020.

[49] Takumi Kawashima, Qina Yu, Akari Asai, Daiki Ikami, and Kiyoharu Aizawa. The aleatoric uncertainty estimation using a separate formulation with virtual residuals. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 1438–1445. IEEE, 2021.

[50] Tinghui Zhou, Matthew Brown, Noah Snavely, and David G Lowe. Unsupervised learning of depth and ego-motion from video. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1851–1858, 2017.

[51] Huai-Jen Liang, Nitin J Sanket, Cornelia Fermüller, and Yiannis Aloimonos. Salientdso: Bringing attention to direct sparse odometry. *IEEE Transactions on Automation Science and Engineering*, 16(4):1619–1626, 2019.

[52] Nitin J. Sanket, Chahat Deep Singh, Cornelia Fermüller, and Yiannis Aloimonos. PRGFlyt: AI based indoor quadrotor autonomy framework for navigation and interaction tasks. https://github.com/prgumd/PRGFlyt/wiki, 2019.

[53] NVIDIA Jetson TX2. https://developer.nvidia.com/embedded/jetson-tx2.

[54] Nitin J. Sanket, Chahat Deep Singh, Chethan M. Parameshwara, Cornelia Fermüller, Guido C. H. E. de Croon, and Yiannis Aloimonos. Evpropnet: Detecting drones by finding propellers for mid-air landing and following. *CoRR*, abs/2106.15045, 2021.

[55] A. Dosovitskiy, P. Fischer, E. Ilg, P. Häusser, C. Hazırbaş, V. Golkov, P. v.d. Smagt, D. Cremers, and T. Brox. Flownet: Learning optical flow with convolutional networks. In *IEEE International Conference on Computer Vision (ICCV)*, 2015.

[56] E. Ilg, T. Saikia, M. Keuper, and T. Brox. Occlusions, motion and depth boundaries with a generic network for disparity, optical flow or scene flow estimation. In *European Conference on Computer Vision (ECCV)*, 2018.

[57] N. Mayer, E. Ilg, P. Häusser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. arXiv:1512.02134.

[58] Xin Zhou, Xiangyong Wen, Zhepei Wang, Yuman Gao, Haojia Li, Qianhao Wang, Tiankai Yang, Haojian Lu, Yanjun Cao, Chao Xu, and Fei Gao. Swarm of micro flying robots in the wild. *Science Robotics*, 7(66):eabm5954, 2022.

[59] G. C. H. E. de Croon, J. J. G. Dupeyroux, S. B. Fuller, and J. A. R. Marshall. Insect-inspired ai for autonomous robots. *Science Robotics*, 7(67):eabl6334, 2022.

[60] Nitin J Sanket, Chethan M Parameshwara, Chahat Deep Singh, Ashwin V Kuruttukulam, C Fermuller, Davide Scaramuzza, and Yiannis Aloimonos. Evdodge: Embodied ai for high-speed dodging on a quadrotor using event cameras. *arXiv preprint arXiv:1906.02919*, pages 31–45, 2019.

[61] René Ranftl, Katrin Lasinger, David Hafner, Konrad Schindler, and Vladlen Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *IEEE transactions on pattern analysis and machine intelligence*, 2020.

[62] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision transformers for dense prediction. *ArXiv preprint*, 2021.

[63] Shengyu Zhao, Yilun Sheng, Yue Dong, Eric I Chang, Yan Xu, et al. Maskflownet: Asymmetric feature matching with learnable occlusion mask. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6278–6287, 2020.

[64] Nitin J Sanket, Chethan M Parameshwara, Chahat Deep Singh, Ashwin V Kuruttukulam, Cornelia Fermüller, Davide Scaramuzza, and Yiannis Aloimonos. Evdodgenet: Deep dynamic obstacle dodging with event cameras. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10651–10657. IEEE, 2020.

[65] Davide Falanga, Kevin Kleber, and Davide Scaramuzza. Dynamic obstacle avoidance for quadrotors with event cameras. *Science Robotics*, 5(40):eaaz9712, 2020.

[66] Nitin J Sanket, Chahat Deep Singh, Varun Asthana, Cornelia Fermüller, and Yiannis Aloimonos. Morpheyes: Variable baseline stereo for quadrotor navigation. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 413–419. IEEE, 2021.

[67] Andreas Bircher, Mina Kamel, Kostas Alexis, Helen Oleynikova, and Roland Siegwart. Receding horizon "next-best-view" planner for 3d exploration. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1462–1468, 2016.

[68] Intel RealSense Depth Camera D435i. https://www.intelrealsense.com/depth-camera-d435i/.

[69] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8934–8943, 2018.

[70] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2462–2470, 2017.

[71] Anurag Ranjan and Michael J Black. Optical flow estimation using a spatial pyramid network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4161–4170, 2017.

[72] Ruzena Bajcsy, Yiannis Aloimonos, and John K Tsotsos. Revisiting active perception. *Autonomous Robots*, 42(2):177–196, 2018.

[73] John K Tsotsos. On the relative complexity of active vs. passive visual search. *International journal of computer vision*, 7(2):127–141, 1992.

[74] Seong Hun Lee and Guido de Croon. Stability-based scale estimation for monocular slam. *IEEE Robotics and Automation Letters*, 3(2):780–787, 2018.

[75] Guido CHE de Croon, Christophe De Wagter, and Tobias Seidl. Enhancing optical-flow-based control by learning visual appearance cues for flying robots. *Nature Machine Intelligence*, 3(1):33–41, 2021.

[76] Carlo Tomasi and Takeo Kanade. Detection and tracking of point. *Int J Comput Vis*, 9:137–154, 1991.

[77] Nitin J Sanket, Chahat Deep Singh, Cornelia Fermüller, and Yiannis Aloimonos. Prgflow: Unified swap-aware deep global optical flow for aerial robot navigation. *Electronics letters*, 57(16), 2021.

[78] Jean-Luc Stevens and Robert Mahony. Vision based forward sensitive reactive control for a quadrotor vtol. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5232–5238, 2018.

[79] Antonio Loquercio, Elia Kaufmann, René Ranftl, Matthias Müller, Vladlen Koltun, and Davide Scaramuzza. Learning high-speed flight in the wild. *Science Robotics*, 6(59):eabg5810, 2021.

[80] K. N. McGuire, C. De Wagter, K. Tuyls, H. J. Kappen, and G. C. H. E. de Croon. Minimal navigation solution for a swarm of tiny flying robots to explore an unknown environment. *Science Robotics*, 4(35):eaaw9710, 2019.

[81] Anurag Ranjan, Joel Janai, Andreas Geiger, and Michael J Black. Attacking optical flow. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2404–2413, 2019.

[82] Tom van Dijk and Guido de Croon. How do neural networks see depth in single images? In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2183–2191, 2019.

[83] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[84] Xinlei Chen, Ross Girshick, Kaiming He, and Piotr Dollár. Tensormask: A foundation for dense object segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2061–2069, 2019.

[85] Philip HS Torr. Geometric motion segmentation and model selection. *Philosophical Transactions of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, 356(1740):1321–1340, 1998.

[86] Roberto Tron and René Vidal. A benchmark for the comparison of 3-d motion segmentation algorithms. In *2007 IEEE conference on computer vision and pattern recognition*, pages 1–8. IEEE, 2007.

[87] Y. Weiss and E. H. Adelson. A unified mixture framework for motion segmentation: incorporating spatial coherence and estimating the number of models. In *Proceedings CVPR IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 321–326, 1996.

[88] Allan Jepson and Michael J Black. Mixture models for optical flow computation. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 760–761. IEEE, 1993.

[89] Pia Bideau, Aruni RoyChowdhury, Rakesh R Menon, and Erik Learned-Miller. The best of both worlds: Combining cnns and geometric constraints for hierarchical motion segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 508–517, 2018.

[90] Pia Bideau and Erik Learned-Miller. It's moving! a probabilistic model for causal motion segmentation in moving camera videos. In *European Conference on Computer Vision*, pages 433–449. Springer, 2016.

[91] Jonas Wulff, Laura Sevilla-Lara, and Michael J Black. Optical flow in mostly rigid scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4671–4680, 2017.

[92] T. Stoffregen et al. Event-based motion segmentation by motion compensation. In *International Conference on Computer Vision (ICCV)*, 2019.

[93] J. Bohg, K. Hausman, B. Sankaran, O. Brock, D. Kragic, S. Schaal, and G. S. Sukhatme. Interactive perception: Leveraging action in perception and perception in action. *IEEE Transactions on Robotics*, 33(6):1273–1291, 2017.

[94] Dov Katz, Moslem Kazemi, J Andrew Bagnell, and Anthony Stentz. Clearing a pile of unknown objects using interactive perception. In *2013 IEEE International Conference on Robotics and Automation*, pages 154–161. IEEE, 2013.

[95] Karol Hausman, Christian Bersch, Dejan Pangercic, Sarah Osentoski, Zoltan-Csaba Marton, and Michael Beetz. Segmentation of cluttered scenes through interactive perception. In *ICRA Workshop on Semantic Perception and Mapping for Knowledge-enabled Service Robotics, St. Paul, MN, USA*, 2012.

[96] Pulkit Agrawal, Ashvin Nair, Pieter Abbeel, Jitendra Malik, and Sergey Levine. Learning to poke by poking: Experiential learning of intuitive physics. *arXiv preprint arXiv:1606.07419*, 2016.

[97] D. Schiebener, J. Schill, and T. Asfour. Discovery, segmentation and reactive grasping of unknown objects. In *2012 12th IEEE-RAS International Conference on Humanoid Robots (Humanoids 2012)*, pages 71–77, 2012.

[98] Lorenzo Natale, Giorgio Metta, and Giulio Sandini. Learning haptic representation of objects. In *International Conference on Intelligent Manipulation and Grasping*, page 43, 2004.

[99] Deepak Pathak, Yide Shentu, Dian Chen, Pulkit Agrawal, Trevor Darrell, Sergey Levine, and Jitendra Malik. Learning instance segmentation by interaction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 2042–2045, 2018.

[100] Berthold KP Horn and Brian G Schunck. Determining optical flow. *Artificial intelligence*, 17(1-3):185–203, 1981.

[101] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231, 1996.

[102] Ruoteng Li, Robby T Tan, Loong-Fah Cheong, Angelica I Aviles-Rivero, Qingnan Fan, and Carola-Bibiane Schonlieb. Rainflow: Optical flow under rain streaks and rain veiling effect. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7304–7313, 2019.

[103] German I Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 113:54–71, 2019.

[104] Yuki M Asano, Christian Rupprecht, Andrew Zisserman, and Andrea Vedaldi. Pass: An imagenet replacement for self-supervised pretraining without humans. *arXiv preprint arXiv:2109.13228*, 2021.

[105] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick Van Der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2758–2766, 2015.

[106] Nikolaus Mayer, Eddy Ilg, Philip Hausser, Philipp Fischer, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4040–4048, 2016.

[107] Weichao Qiu and Alan Yuille. Unrealcv: Connecting computer vision to unreal engine. In *European Conference on Computer Vision*, pages 909–916. Springer, 2016.

[108] Chengshu Li, Fei Xia, Roberto Martín-Martín, Michael Lingelbach, Sanjana Srivastava, Bokui Shen, Kent Vainio, Cem Gokmen, Gokul Dharan, Tanish Jain, et al. igibson 2.0: Object-centric simulation for robot learning of everyday household tasks. *arXiv preprint arXiv:2108.03272*, 2021.

[109] Ainaz Eftekhar, Alexander Sax, Jitendra Malik, and Amir Zamir. Omnidata: A scalable pipeline for making multi-task mid-level vision datasets from 3d scans. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10786–10796, 2021.

[110] Maximilian Denninger, Martin Sundermeyer, Dominik Winkelbauer, Youssef Zidan, Dmitry Olefir, Mohamad Elbadrawy, Ahsan Lodhi, and Harinandan Katam. Blenderproc. *arXiv preprint arXiv:1911.01911*, 2019.

[111] Shital Shah, Debadeepta Dey, Chris Lovett, and Ashish Kapoor. Airsim: High-fidelity visual and physical simulation for autonomous vehicles. In *Field and service robotics*, pages 621–635. Springer, 2018.

[112] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. Carla: An open urban driving simulator. In *Conference on robot learning*, pages 1–16. PMLR, 2017.

[113] Klaus Greff, Francois Belletti, Lucas Beyer, Carl Doersch, Yilun Du, Daniel Duckworth, David J Fleet, Dan Gnanapragasam, Florian Golemo, Charles Herrmann, et al. Kubric: A scalable dataset generator. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3749–3761, 2022.

[114] Panpan Cai, Yiyuan Lee, Yuanfu Luo, and David Hsu. Summit: A simulator for urban driving in massive mixed traffic. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4023–4029. IEEE, 2020.

[115] Daniel Scharstein and Richard Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International journal of computer vision*, 47(1):7–42, 2002.

[116] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black. A naturalistic open source movie for optical flow evaluation. In A. Fitzgibbon et al. (Eds.), editor, *European Conf. on Computer Vision (ECCV)*, Part IV, LNCS 7577, pages 611–625. Springer-Verlag, October 2012.

[117] Anton Mitrokhin, Cornelia Fermüller, Chethan Parameshwara, and Yiannis Aloimonos. Event-based moving object detection and tracking. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–9. IEEE, 2018.

[118] Anton Mitrokhin, Chengxi Ye, Cornelia Fermüller, Yiannis Aloimonos, and Tobi Delbruck. Ev-imo: Motion segmentation dataset and learning pipeline for event cameras. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6105–6112. IEEE, 2019.

[119] Francisco Barranco, Cornelia Fermüller, and Eduardo Ros. Real-time clustering and multi-target tracking using event-based sensors. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5764–5769. IEEE, 2018.

[120] Davide Falanga, Kevin Kleber, and Davide Scaramuzza. Dynamic obstacle avoidance for quadrotors with event cameras. *Science Robotics*, 5(40):eaaz9712, 2020.

[121] Chethan M Parameshwara, Nitin J Sanket, Chahat Deep Singh, Cornelia Fermüller, and Yiannis Aloimonos. 0-mms: Zero-shot multi-motion segmentation with a monocular event camera. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9594–9600. IEEE, 2021.

[122] Andreas Wedel, Annemarie Meißner, Clemens Rabe, Uwe Franke, and Daniel Cremers. Detection and segmentation of independently moving objects from dense scene flow. In *International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition*, pages 14–27. Springer, 2009.

[123] Nitin J Sanket, Chahat Deep Singh, Kanishka Ganguly, Cornelia Fermüller, and Yiannis Aloimonos. Gapflyt: Active vision based minimalist structure-less gap detection for quadrotor flight. *IEEE Robotics and Automation Letters*, 3(4):2799–2806, 2018.

[124] Chakravarty R Alla Chaitanya, Anton S Kaplanyan, Christoph Schied, Marco Salvi, Aaron Lefohn, Derek Nowrouzezahrai, and Timo Aila. Interactive reconstruction of monte carlo image sequences using a recurrent denoising autoencoder. *ACM Transactions on Graphics (TOG)*, 36(4):1–12, 2017.

[125] Bruno Lévy, Sylvain Petitjean, Nicolas Ray, and Jérome Maillot. Least squares conformal maps for automatic texture atlas generation. *ACM transactions on graphics (TOG)*, 21(3):362–371, 2002.

[126] Prochitecture. Blender-osm: Openstreetmap and terrain for blender.

[127] Maximilian Denninger, Martin Sundermeyer, Dominik Winkelbauer, Youssef Zidan, Dmitry Olefir, Mohamad Elbadrawy, Ahsan Lodhi, and Harinandan Katam. Blenderproc. *arXiv preprint arXiv:1911.01911*, 2019.

[128] Blender Cell Fracture. https://docs.blender.org/manual/en/latest/addons/object/cell_fracture.html.

[129] Nitin J Sanket, Chahat Deep Singh, Chethan M Parameshwara, Cornelia Fermüller, Guido CHE de Croon, and Yiannis Aloimonos. EVPropNet: Detecting Drones By Finding Propellers For Mid-Air Landing And Following. In *Robotics: Science and systems (RSS) conference 2021*. Robotics: Science and Systems, 2021.

[130] Nitin J Sanket, Chethan M Parameshwara, Chahat Deep Singh, Ashwin V Kuruttukulam, Cornelia Fermüller, Davide Scaramuzza, and Yiannis Aloimonos. Evdodgenet: Deep dynamic obstacle dodging with event cameras. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10651–10657. IEEE, 2020.

[131] Y Hu, S C Liu, and T Delbruck. v2e: From video frames to realistic DVS events. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. IEEE, 2021.

[132] Daniel Gehrig, Mathias Gehrig, Javier Hidalgo-Carrió, and Davide Scaramuzza. Video to events: Recycling video datasets for event cameras. In *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, June 2020.

[133] Nitin J Sanket, Chahat Deep Singh, Varun Asthana, Cornelia Fermüller, and Yiannis Aloimonos. Morpheyes: Variable baseline stereo for quadrotor navigation. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 413–419. IEEE, 2021.

[134] Pablo Pueyo, Eduardo Montijano, Ana C Murillo, and Mac Schwager. Cinempc: Controlling camera intrinsics and extrinsics for autonomous cinematography. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 4058–4064. IEEE, 2022.

[135] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015.

[136] Laura Downs, Anthony Francis, Nate Koenig, Brandon Kinman, Ryan Hickman, Krista Reymann, Thomas B McHugh, and Vincent Vanhoucke. Google scanned objects: A high-quality dataset of 3d scanned household items. *arXiv preprint arXiv:2204.11918*, 2022.

[137] 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*, pages 740–755. Springer, 2014.

[138] ambientCG. https://ambientcg.com.

[139] Polyhaven. https://polyhaven.com.

[140] cgbookcase. https://cgbookcase.com.

[141] Sue Blackman. Rigging with mixamo. In *Unity for Absolute Beginners*, pages 565–573. Springer, 2014.

[142] Haofei Xu, Jing Zhang, Jianfei Cai, Hamid Rezatofighi, and Dacheng Tao. Gmflow: Learning optical flow via global matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8121–8130, 2022.

[143] Deqing Sun, Daniel Vlasic, Charles Herrmann, Varun Jampani, Michael Krainin, Huiwen Chang, Ramin Zabih, William T Freeman, and Ce Liu. Autoflow: Learning a better training set for optical flow. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10093–10102, 2021.

[144] Michaël Gharbi, Jiawen Chen, Jonathan T Barron, Samuel W Hasinoff, and Frédo Durand. Deep bilateral learning for real-time image enhancement. *ACM Transactions on Graphics (TOG)*, 36(4):118, 2017.

[145] Marcela Carvalho, Bertrand Le Saux, Pauline Trouvé-Peloux, Andrés Almansa, and Frédéric Champagnat. Deep depth from defocus: how can defocus blur improve 3d estimation using dense neural networks? In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, pages 0–0, 2018.

[146] Shir Gur and Lior Wolf. Single image depth estimation trained via depth from defocus cues. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7683–7692, 2019.

[147] Mingdeng Cao, Zhihang Zhong, Jiahao Wang, Yinqiang Zheng, and Yujiu Yang. Learning adaptive warping for real-world rolling shutter correction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17785–17793, 2022.

[148] Zhixiang Wang, Xiang Ji, Jia-Bin Huang, Shin'ichi Satoh, Xiao Zhou, and Yinqiang Zheng. Neural global shutter: Learn to restore video from a rolling shutter camera with global reset feature. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17794–17803, 2022.

[149] Meiguang Jin, Givi Meishvili, and Paolo Favaro. Learning to extract a video sequence from a single motion-blurred image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6334–6342, 2018.

[150] Tim Brooks and Jonathan T Barron. Learning to synthesize motion blur. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6840–6848, 2019.

[151] Anat Levin, Rob Fergus, Frédo Durand, and William T Freeman. Image and depth from a conventional camera with a coded aperture. *ACM transactions on graphics (TOG)*, 26(3):70–es, 2007.

[152] Jorge Bacca, Tatiana Gelvez-Barrera, and Henry Arguello. Deep coded aperture design: An end-to-end approach for computational imaging tasks. *IEEE Transactions on Computational Imaging*, 7:1148–1160, 2021.

[153] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021.

[154] Yuanbo Xiangli, Linning Xu, Xingang Pan, Nanxuan Zhao, Anyi Rao, Christian Theobalt, Bo Dai, and Dahua Lin. Bungeenerf: Progressive neural radiance field for extreme multi-scale scene rendering. In *The European Conference on Computer Vision (ECCV)*, 2022.

[155] Roger Marí, Gabriele Facciolo, and Thibaud Ehret. Sat-NeRF: Learning multi-view satellite photogrammetry with transient objects and shadow modeling using RPC cameras. *arXiv preprint arXiv:2203.08896*, 2022.

[156] Ioannis Mademlis, Vasileios Mygdalis, Nikos Nikolaidis, Maurizio Montagnuolo, Fulvio Negro, Alberto Messina, and Ioannis Pitas. High-level multiple-uav cinematography tools for covering outdoor events. *IEEE Transactions on Broadcasting*, 65(3):627–635, 2019.

[157] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[158] Haitao Yuan and Guoliang Li. A survey of traffic prediction: from spatio-temporal data to intelligent transportation. *Data Science and Engineering*, 6(1):63–85, 2021.

[159] Shuhan Tan, Kelvin Wong, Shenlong Wang, Sivabalan Manivasagam, Mengye Ren, and Raquel Urtasun. Scenegen: Learning to generate realistic traffic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 892–901, 2021.

[160] Abhijit Kundu, Kyle Genova, Xiaoqi Yin, Alireza Fathi, Caroline Pantofaru, Leonidas Guibas, Andrea Tagliasacchi, Frank Dellaert, and Thomas Funkhouser. Panoptic Neural Fields: A Semantic Object-Aware Neural Scene Representation. In *CVPR*, 2022.

[161] Mykhaylo Andriluka, Leonid Pishchulin, Peter Gehler, and Bernt Schiele. 2d human pose estimation: New benchmark and state of the art analysis. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.

[162] Gregory Rogez and Cordelia Schmid. Mocap-guided data augmentation for 3d pose estimation in the wild. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

[163] Salehe Erfanian Ebadi, You-Cyuan Jhang, Alex Zook, Saurav Dhakad, Adam Crespi, Pete Parisi, Steven Borkman, Jonathan Hogins, and Sujoy Ganguly. Peoplesanspeople: A synthetic data generator for human-centric computer vision. *arXiv preprint arXiv:2112.09290*, 2021.

[164] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. *Advances in neural information processing systems*, 27, 2014.

[165] David Eigen and Rob Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *Proceedings of the IEEE international conference on computer vision*, pages 2650–2658, 2015.

[166] Dan Xu, Wei Wang, Hao Tang, Hong Liu, Nicu Sebe, and Elisa Ricci. Structured attention guided convolutional neural fields for monocular depth estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3917–3925, 2018.

[167] Elisa Ricci, Wanli Ouyang, Xiaogang Wang, Nicu Sebe, et al. Monocular depth estimation using multi-scale continuous crfs as sequential deep networks. *IEEE transactions on pattern analysis and machine intelligence*, 41(6):1426–1440, 2018.

[168] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[169] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[170] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision transformers for dense prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12179–12188, 2021.

[171] Shariq Farooq Bhat, Ibraheem Alhashim, and Peter Wonka. Adabins: Depth estimation using adaptive bins. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4009–4018, 2021.

[172] Doyeon Kim, Woonghyun Ka, Pyungwhan Ahn, Donggyu Joo, Sehwan Chun, and Junmo Kim. Global-local path networks for monocular depth estimation with vertical cutdepth. *arXiv preprint arXiv:2201.07436*, 2022.

[173] Bin Cheng, Inderjot Singh Saggu, Raunak Shah, Gaurav Bansal, and Dinesh Bharadia. S 3 net: Semantic-aware self-supervised depth estimation with monocular videos and synthetic data. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXX*, pages 52–69. Springer, 2020.

[174] Shengjie Zhu, Garrick Brazil, and Xiaoming Liu. The edge of depth: Explicit constraints between segmentation and depth. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 13116–13125, 2020.

[175] Clément Godard, Oisin Mac Aodha, and Gabriel J Brostow. Unsupervised monocular depth estimation with left-right consistency. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 270–279, 2017.

[176] Benjamin Ummenhofer, Huizhong Zhou, Jonas Uhrig, Nikolaus Mayer, Eddy Ilg, Alexey Dosovitskiy, and Thomas Brox. Demon: Depth and motion network for learning monocular stereo. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5038–5047, 2017.

[177] Zhichao Yin and Jianping Shi. Geonet: Unsupervised learning of dense depth, optical flow and camera pose. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1983–1992, 2018.

[178] Po-Han Huang, Kevin Matzen, Johannes Kopf, Narendra Ahuja, and Jia-Bin Huang. Deepmvs: Learning multi-view stereopsis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2821–2830, 2018.

[179] Robert T Collins. A space-sweep approach to true multi-image matching. In *Proceedings CVPR IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 358–363. Ieee, 1996.

[180] Weirong Chen, Suryansh Kumar, and Fisher Yu. Uncertainty-driven dense two-view structure from motion. *IEEE Robotics and Automation Letters*, 8(3):1763–1770, 2023.

[181] Kai Cheng, Hao Chen, Wei Yin, Guangkai Xu, and Xuejin Chen. Exploiting correspondences with all-pairs correlations for multi-view depth estimation. *arXiv preprint arXiv:2205.02481*, 2022.

[182] W Nicholas Greene and Nicholas Roy. Multiviewstereonet: Fast multi-view stereo depth estimation using incremental viewpoint-compensated feature extraction. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9242–9248. IEEE, 2021.

[183] Fangchang Ma and Sertac Karaman. Sparse-to-dense: Depth prediction from sparse depth samples and a single image. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 4796–4803. IEEE, 2018.

[184] Antonio Loquercio, Alexey Dosovitskiy, and Davide Scaramuzza. Learning depth with very sparse supervision. *IEEE Robotics and Automation Letters*, 5(4):5542–5549, 2020.

[185] Yuankai Lin, Tao Cheng, Qi Zhong, Wending Zhou, and Hua Yang. Dynamic spatial propagation network for depth completion. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 1638–1646, 2022.

[186] Sifei Liu, Shalini De Mello, Jinwei Gu, Guangyu Zhong, Ming-Hsuan Yang, and Jan Kautz. Learning affinity via spatial propagation networks. *Advances in Neural Information Processing Systems*, 30, 2017.

[187] Zhang Youmin, Guo Xianda, Poggi Matteo, Zhu Zheng, Huang Guan, and Mattoccia Stefano. Completionformer: Depth completion with convolutions and vision transformers. *arXiv preprint arXiv:2304.13030*, 2023.

[188] Yijin Li, Xinyang Liu, Wenqi Dong, Han Zhou, Hujun Bao, Guofeng Zhang, Yinda Zhang, and Zhaopeng Cui. Deltar: Depth estimation from a light-weight tof sensor and rgb image. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part I*, pages 619–636. Springer, 2022.

[189] Hanna Müller, Vlad Niculescu, Tommaso Polonelli, Michele Magno, and Luca Benini. Robust and efficient depth-based obstacle avoidance for autonomous miniaturized uavs. *arXiv preprint arXiv:2208.12624*, 2022.

[190] Pushmeet Kohli Nathan Silberman, Derek Hoiem and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012.

[191] Zhenyu Li, Zehui Chen, Jialei Xu, Xianming Liu, and Junjun Jiang. Litedepth: digging into fast and accurate depth estimation on mobile devices. In *Computer Vision–ECCV 2022 Workshops: Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part II*, pages 507–523. Springer, 2023.

[192] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[193] Iro Laina, Christian Rupprecht, Vasileios Belagiannis, Federico Tombari, and Nassir Navab. Deeper depth prediction with fully convolutional residual networks. In *2016 Fourth international conference on 3D vision (3DV)*, pages 239–248. IEEE, 2016.

[194] Nitin J. Sanket, Chahat Deep Singh, Cornelia Fermüller, and Yiannis Aloimonos. PRGFlow: Benchmarking SWAP-Aware Unified Deep Visual Inertial Odometry, 2020.

[195] Pierre Charbonnier, Laure Blanc-Feraud, Gilles Aubert, and Michel Barlaud. Two deterministic half-quadratic regularization algorithms for computed imaging. In *Proceedings of 1st international conference on image processing*, volume 2, pages 168–172. IEEE, 1994.

[196] Art B Owen. A robust hybrid of lasso and ridge regression. *Contemporary Mathematics*, 443(7):59–72, 2007.

[197] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

[198] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[199] René Ranftl, Katrin Lasinger, David Hafner, Konrad Schindler, and Vladlen Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(3), 2022.

[200] Jianxiong Xiao, Andrew Owens, and Antonio Torralba. Sun3d: A database of big spaces reconstructed using sfm and object labels. In *Proceedings of the IEEE international conference on computer vision*, pages 1625–1632, 2013.

[201] Fangchang Ma and Sertac Karaman. Sparse-to-dense: Depth prediction from sparse depth samples and a single image. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 4796–4803. IEEE, 2018.

[202] Russell D Fernald. Evolution of eyes. *Current opinion in neurobiology*, 10(4):444–450, 2000.

[203] Max Grosse, Gordon Wetzstein, Anselm Grundhöfer, and Oliver Bimber. Coded aperture projection. *ACM Transactions on Graphics (TOG)*, 29(3):1–12, 2010.

[204] Hayato Ikoma, Cindy M Nguyen, Christopher A Metzler, Yifan Peng, and Gordon Wetzstein. Depth from defocus with learned optics for imaging and occlusion-aware depth estimation. In *2021 IEEE International Conference on Computational Photography (ICCP)*, pages 1–12. IEEE, 2021.

[205] Javier Hidalgo-Carrió, Daniel Gehrig, and Davide Scaramuzza. Learning monocular dense depth from events. In *2020 International Conference on 3D Vision (3DV)*, pages 534–542. IEEE, 2020.

[206] Manasi Muglikar, Diederik Paul Moeys, and Davide Scaramuzza. Event guided depth sensing. In *2021 International Conference on 3D Vision (3DV)*, pages 385–393. IEEE, 2021.

[207] Amedeo Fabris, Kevin Kleber, Davide Falanga, and Davide Scaramuzza. Geometry-aware compensation scheme for morphing drones. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 592–598. IEEE, 2021.

[208] Davide Falanga, Kevin Kleber, Stefano Mintchev, Dario Floreano, and Davide Scaramuzza. The foldable drone: A morphing quadrotor that can squeeze and fly. *IEEE Robotics and Automation Letters*, 4(2):209–216, 2018.

[209] Nathan Bucki and Mark W Mueller. Design and control of a passively morphing quadcopter. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 9116–9122. IEEE, 2019.

[210] Karishma Patnaik, Shatadal Mishra, Seyed Mostafa Rezayat Sorkhabadi, and Wenlong Zhang. Design and control of squeeze: A spring-augmented quadrotor for interactions with the environment to squeeze-and-fly. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1364–1370. IEEE, 2020.

[211] Andreas Papadimitriou, Sina Sharif Mansouri, Christoforos Kanellakis, and George Nikolakopoulos. Geometry aware nmpc scheme for morphing quadrotor navigation in restricted entrances. In *2021 European Control Conference (ECC)*, pages 1597–1603. IEEE, 2021.

[212] Enrico Ajanic, Mir Feroskhan, Stefano Mintchev, Flavio Noca, and Dario Floreano. Bioinspired wing and tail morphing extends drone flight capabilities. *Science Robotics*, 5(47):eabc2897, 2020.