

Delft University of Technology
Master of Science Thesis in Computer and Embedded Systems Engineering

Drawn to the Light: Autonomous Optical Navigation for Micro-Drones

Bart van Maris



Drawn to the Light: Autonomous Optical Navigation for Micro-Drones

Master of Science Thesis in Computer and Embedded Systems
Engineering

Embedded Systems Group
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology
Van Mourik Broekmanweg 6, 2628 XE Delft, The Netherlands

Bart van Maris

1st July 2026

Author

Bart van Maris

Title

Drawn to the Light: Autonomous Optical Navigation for Micro-Drones

MSc Presentation Date

9th of July 2026

Graduation Committee

Marco Zuñiga Zamalloa Delft University of Technology

Gosia Migut Delft University of Technology

Harry Huang Delft University of Technology

Abstract

Small drones increasingly operate indoors, where GPS is unavailable and the heavier sensors normally used for indoor positioning exceed a micro-drone’s weight, size, and power budget. Visible light is a practical alternative, because the spaces a drone operates in are already lit. An ordinary lamp can be modulated to act as a navigation beacon, while the drone carries only a small light sensor to detect it. Prior work by Huang et al. navigates a micro-drone toward a single such beacon, recovering the direction of the light from a ring of photodiodes, but it keeps no record of the light field. It therefore cannot distinguish one beacon from another, nor route around an obstacle that blocks the light.

This thesis gives a weight-, size-, and power-constrained micro-drone the navigation those methods lacked, using only on-board light sensing. We design a fused controller that combines two directions. The first is the bearing toward the beacon, read from a ring of photodiodes. The second is a gradient, which the drone estimates from a map of the light field that it builds as it flies. The sensing runs on a custom light-sensing board that fits well within a micro-drone’s payload budget. A signal pipeline on the drone then recovers each beacon from the board’s noisy output through a frequency analysis, and reports a confidence measure for each reading. We validate the approach in a Webots simulation and in real flight. The drone reaches modulated beacons forming a predetermined path, and routes around an obstacle, using its stored map of the light field when the obstacle blocks the beacon.

Preface

Throughout my studies, I found myself increasingly drawn to the practical application of the theories we learned in class. This preference ultimately led me to choose the subject of this thesis. Beyond its practical aspects, the growing relevance of drones in today's geopolitical landscape further sparked my interest in the topic. The initial direction of this thesis also offered a considerable degree of freedom, for which I am grateful to my thesis advisor and supervisor. This freedom gave me the opportunity to design my own printed circuit board for the first time. Reaching that milestone made me feel that this thesis truly brought my studies full circle, coupling my computer science background with my studies in Computer and Embedded Systems Engineering.

I would first like to thank my daily supervisor, Harry Huang, not only for providing the foundation for this work, but also for his excellent guidance, support, and encouragement throughout the completion of this thesis. I would also like to thank my thesis advisor, Marco Zuñiga Zamalloa, for his guidance and the critical questions that helped shape this thesis into its final form. Finally, I would like to thank my roommates, family, colleagues, and girlfriend for patiently listening to my endless explanations and discussions about this work. Without you all serving as my rubber ducks, I would never have arrived at many of the insights that made this thesis possible.

Bart van Maris

Delft, The Netherlands
1st July 2026

Declaration on the Use of Artificial Intelligence

During the preparation of this thesis, the author made use of generative artificial-intelligence tools (large language models) to assist with language editing, rephrasing, and the structuring of text, as well as for occasional support while writing and debugging analysis and figure-generation code. All such output was reviewed, verified, and edited by the author, who takes full responsibility for the content of this thesis. The research itself—the hardware design, the controller, the experiments, and their interpretation—was conceived and carried out by the author. No generative tool was used to produce experimental data or results.

Bart van Maris

Contents

Preface	v
Declaration on the Use of Artificial Intelligence	vii
1 Introduction	1
1.1 Motivation & Problem Statement	1
1.2 Research Objectives	3
1.3 Contributions	4
1.4 Thesis Overview	4
2 Background & Related Work	6
2.1 Related Work	6
2.2 Drone Platform	9
2.3 Visible Light Communication & Sensing	9
2.3.1 The Light Signal	10
2.4 Algorithms from Huang et al.	11
2.4.1 Bearing Angle Estimation	11
2.4.2 Gradient-Based Source Seeking	12
2.4.3 Comparing the Algorithms	13
2.5 Simulation Environment	13
3 Problem & Algorithm Design	15
3.1 Why Existing Methods Fall Short	16
3.2 System Overview	18
3.3 Bearing Angle	19
3.4 Light Intensity Map	21
3.5 Gradient Estimation	22
3.5.1 Selection of the Gradient Estimator	23
3.6 Multi-Beacon Traversal	28
3.7 Controller State Machine	30
3.8 Control Output	31
4 PCB Hardware Design	33
4.1 The Original Prototype	33
4.2 Design Requirements	34
4.3 Circuit Design	35
4.4 PCB Layout	37

5	Real-World Implementation	39
5.1	From Intensity to SNR	39
5.2	Filter Validation	45
5.3	Effect of Ambient Lighting	47
6	Multi-Beacon Experiments	51
6.1	Experimental Setup	51
6.2	Separating Beacons by Frequency	53
6.3	Sequential Multi-Target Traverse	54
7	Obstacle Experiments	59
7.1	Experimental Setup	59
7.2	SNR Field Mapping	60
7.3	Obstacle Avoidance	64
	7.3.1 Simulation	64
	7.3.2 Real-World Flight	65
7.4	Generalisation to a Blimp Platform	67
	7.4.1 Motor Noise on the CrazyFlie	67
8	Conclusions and Future Work	71
8.1	Future Work	73
	8.1.1 Improving the Sensing Hardware	73
	8.1.2 Extending the Navigation	74
	8.1.3 Characterisation and Robustness	75
A	Controller Parameters	80

Chapter 1

Introduction

This chapter introduces the problem addressed in this thesis and the methods developed to solve it. We begin by examining why indoor navigation is difficult for micro-drones, whose limited capabilities tightly constrain the sensing and computation they can carry. We then discuss visible-light sensing as a promising direction for indoor navigation, highlighting how existing lighting infrastructure can provide guidance with minimal changes to the environment. We next identify the limitations that remain in related work, which motivate the research questions and contributions of this thesis. Finally, we outline the organisation of the remaining chapters.

1.1 Motivation & Problem Statement

Small unmanned aircraft are increasingly used for tasks that once required human operators, including structural inspection, environmental monitoring, and warehouse inventory tracking [10]. Many of these applications take place indoors, where demand continues to grow. The indoor inspection-drone market alone is projected to expand from USD 254 million in 2024 to USD 614 million by 2032 [16]. Yet indoor operation remains difficult. Outdoor drones navigate mainly with GPS, but its signals become unreliable once a drone moves inside. Roofs and walls block satellite signals, while surrounding structures can also create severe interference in dense urban areas. As a result, GPS-based positioning can become too weak or unstable to support reliable navigation [30].

Micro-drones offer a promising platform for many indoor applications. Their small size and low mass make them attractive in cluttered spaces such as warehouses and greenhouses, where a larger drone could damage shelves, cargo, pipes, glass panels, or plants in the event of a collision. A smaller drone reduces this risk and can operate in tighter spaces. However, the same compact form factor also creates the central design constraint of this thesis. A micro-drone cannot carry heavy sensors, large batteries, or powerful onboard computers, so its navigation system must remain lightweight, low-power, and simple to deploy. The central problem of this thesis is therefore how a micro-drone can navigate reliably in GPS-denied indoor environments, such as warehouses or greenhouses. It must do so under strict weight, size, and power constraints, and with minimal changes to the environment.

Visible light provides a promising solution for our target scenario, since indoor spaces are generally already equipped with lighting infrastructure [32]. The idea of visible-light positioning is to use ordinary lamps as signal transmitters that provide information for the drone to estimate its position. A simple switching circuit can be attached to a lamp to modulate the light on and off faster than the human eye can perceive. In this way, the lamp continues to illuminate the room steadily for people while also carrying a signal that the drone can detect [25]. The drone then needs to carry only a small light sensor, whose mass and power draw are well within a micro-drone’s budget. However, many visible-light positioning systems still require the lamp positions to be measured precisely before the first flight. For example, the work by Hernández et al. [14] requires densely installed lights with known positions, which increases the deployment effort before the drone can navigate on the light signals.

A light-based navigation system has a central trade-off between the position information it provides and the lighting infrastructure it requires. Visible-light positioning systems can recover a drone’s position with centimetre-level accuracy [19], but they typically rely on multiple modulated ceiling lights whose locations are known in advance. This turns the lighting layout itself into part of the navigation system. Deployment then becomes costly in spaces where the light infrastructure is sparse, irregular, or difficult to survey, such as a street with sparsely placed lamps or a living room with only a single light. Using a single light source places a much smaller requirement on the target environment. One light cannot determine the drone’s full position, but it can still give a direction to follow, much as a lighthouse guided ships toward harbour long before they could fix their own position. A single modulated lamp can serve the drone in the same way, as a beacon it steers toward in open space.

Instead of relying on dense lighting infrastructure, Huang et al. [15] first demonstrate a light-guided navigation paradigm that uses a single modulated beacon without requiring a pre-surveyed grid of lamp positions. Their original approach develops two navigation methods that this thesis builds on. The first estimates the bearing toward the beacon using multiple photodiodes arranged in a ring structure. The second reduces the sensing hardware to a single photodiode, estimating the beacon direction from how the measured light intensity changes as the drone moves. Both methods are designed around the same baseline setting: one modulated lamp and a drone flying in an open indoor space, as illustrated in Figure 1.1. This baseline leaves two limitations for more realistic missions. First, tracking one beacon at a time does not by itself demonstrate that the drone can distinguish and visit several beacons in sequence. Second, navigation in open space towards a light anchor does not guarantee successful flight in cluttered environments, where obstacles may block the light signal or cause the drone to collide. This thesis addresses these two limitations by extending their original methods.

This thesis aims to guide a weight-, size-, and power-constrained micro-drone to its targets using only onboard light sensing, as illustrated in Figure 1.2. It pursues two goals. The first is to reach several light beacons in succession, rather than only a single beacon. The second is to reach a beacon even when an obstacle blocks the direct path to it. Achieving this on a micro-drone is challenging because a platform weighing only a few tens of grams has very limited payload, volume, and power. It cannot rely on the heavy sensors or the computational resources available to larger drones, and must instead draw

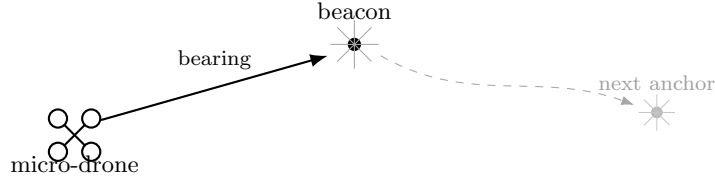


Figure 1.1: **Single-anchor light-guided navigation**, the prior approach this thesis builds on [15]. From one modulated lamp the drone recovers only a *direction*, the bearing, and follows it inward, much as a ship is guided by a lighthouse without ever being told its exact position. Chaining such anchors could in principle guide the drone along a route (grey, right), but the prior methods reach only one beacon at a time in open space: they cannot tell one anchor from another, and a blocked line of sight leaves no direction to follow. These are the two limitations this thesis addresses.

everything it needs to navigate from the small, low-power light sensing it can carry.

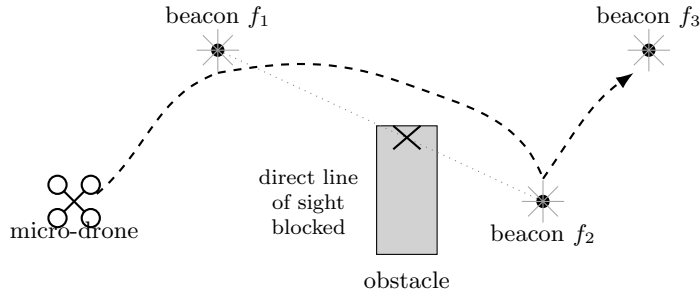


Figure 1.2: **An illustration of the navigation task this thesis addresses.** This is a schematic example, not a recorded flight path or the exact scene used in the experiments. A micro-drone must visit several light beacons in succession, distinguishing them by their modulation frequencies f_1 , f_2 , and f_3 . An obstacle blocks the direct line of sight to beacon f_2 , so steering toward the light alone is not enough to reach it. The drone must instead detour around the obstacle, which requires it to remember the light field rather than react only to the light it currently senses. The experiments in Chapters 6 and 7 test the multi-beacon and obstacle cases separately.

1.2 Research Objectives

This gap — visiting several beacons in succession, and reaching a beacon when obstacles are present — raises three questions about a weight-, size-, and power-constrained drone, which this thesis addresses in turn:

1. Can such a drone carry the necessary light sensing on board and recover a reliable beacon signal in flight?
2. Can it then navigate to several distinct light beacons in succession?
3. Can it navigate around an obstacle that obstructs its direct path to a beacon?

The first question is the foundation of this work: whether a board within the drone’s mass budget can recover a reliable beacon signal in flight at all. The second and third are the two goals that the foundation makes possible, and that the prior single-beacon methods could not reach: visiting several beacons in succession, and routing around an obstacle that blocks the path to one of them. Answering all three requires both a sensing platform light enough to fly and a controller that can act on what that platform is able to sense.

1.3 Contributions

To answer the research questions laid out in the previous section, this thesis makes four contributions.

First, it presents a custom light-sensing board, small and light enough to fly on a micro-drone, that gives the drone the directional light sensing its navigation depends on.

Second, it develops a fused navigation controller, the core technical contribution of this thesis. It combines two directions: the bearing toward a visible beacon, and a gradient that the drone follows through the surrounding light field.

Third, it delivers the two navigation capabilities the thesis set out to achieve, both made possible by this controller. The first is multi-beacon traversal: the drone tells the beacons apart and visits them in succession, rather than reaching only one. The second is obstacle avoidance: by remembering the light field it has already passed through, the drone routes around an obstacle that blocks its view of the beacon, instead of stalling or colliding when the direct light is lost.

Finally, it validates the complete system in a Webots simulation and in real flight. The drone reaches several beacons in succession, and routes around an obstacle that blocks its line of sight to a beacon.

This thesis builds directly on the work of Huang et al. [15], which provides the single-beacon bearing and gradient methods and the Webots simulator. The fused controller, the wider-field light sensing, the multi-beacon and obstacle-avoidance navigation, the sensing board, and the validation in real flight are the contributions of this thesis.

1.4 Thesis Overview

The remainder of this thesis is organised as follows. Chapter 2 reviews how drones navigate GPS-denied indoor spaces, motivates the choice of visible light, and introduces the prior algorithms of Huang et al. Chapter 3 analyses why those methods fall short under multiple beacons and occlusion, and designs the fused controller that overcomes them. Chapter 4 presents the photodiode board

and the payload, power, and size budget it must meet. Chapter 5 describes the real-world implementation, including the signal pipeline that recovers each beacon from the noisy measured light. Chapter 6 reports the multi-beacon experiments, and Chapter 7 the obstacle-avoidance experiments, in both simulation and real flight. Chapter 8 draws the conclusions and sets out the remaining limitations and the directions for future work they open.

Chapter 2

Background & Related Work

In this chapter, we review how drones navigate GPS-denied indoor environments using alternative sensors and identify the remaining challenges of these sensors. We then introduce a drone platform that weighs only 27 grams, whose compact form factor and low mass have drawn increasing research attention for operation in cluttered indoor environments. Since this work aims to improve the robustness of indoor autonomous drone navigation, we first review the light-guided navigation algorithms proposed by Huang et al. [15], which enables drone navigation using a single light anchor with minimal modification to existing lighting infrastructure. Building on their work, this thesis proposes an extension that further demonstrates the potential of visible-light sensing for reliable indoor drone navigation.

2.1 Related Work

Small unmanned aircraft are now widely used for tasks such as structural inspection, environmental monitoring, and warehouse inventory tracking [10]. Many of these applications take place indoors, where the indoor drone market alone is estimated to grow from USD 254 million in 2024 to USD 614 million by 2032 [16]. However, unlike outdoor drones that can rely on GPS for navigation, indoor drones immediately lose reliable GPS signals due to the roofs and surrounding structures that block the signal, making GPS-based positioning inaccurate [30]. Thus, indoor navigation must rely on alternative sensors, which can increase the complexity and payload of the sensors carried by drones or require additional anchor infrastructure to be installed in the environment.

Indoor navigation methods rely on different sensing modalities, and the choice strongly affects the implementation costs. We compare these approaches along two cost dimensions. One cost falls on the drone: the sensor and computation it must carry, both of which are tightly constrained by a micro-drone’s limited payload capacity and onboard processing resources. The other cost falls on the deployment of the system: the infrastructure that must be put in the environment, which determines the complexity when deploying the method in a new environment. Table 2.1 summarises the main approaches against these

two costs.

One family of methods locates the drone by receiving signals from infrastructure installed in the environment beforehand. Two approaches take advantage of electromagnetic signals, i.e., radio signals. Ultra-wideband anchors measure how long a pulse takes to travel between an anchor and the drone, providing positioning information of tens of centimetres accuracy [26]. WiFi fingerprinting instead matches the signal strengths the drone measures against a map of measurements surveyed in advance, where its accuracy is reported to be within a metre [30]. There is a clear trade-off between the two, but neither offers a promising solution. WiFi covers a building using existing hardware, yet its accuracy is limited to the metre level, whereas ultra-wideband is more accurate but requires dedicated anchors reinstalled in a new environment. Another method uses sound: acoustic positioning localises the drone from the travel time of ultrasonic pulses transmitted from fixed beacons [28, 1]. All three share one cost in our setting: they require infrastructure to be installed or surveyed before the first flight.

To address these challenges, researchers have explored vision-based approaches that remove the need for external anchors by building a map as the drone flies. A camera captures a stream of images of the surrounding environment for further processing. Visual simultaneous localisation and mapping, or visual SLAM, then tracks features across these images to estimate the drone’s motion and build a map of its surroundings [22]. This requires no infrastructure in the space. However, the cost shifts to the drone itself: visual SLAM requires a high-resolution camera and powerful on-board computation to process images in real time, which a micro-drone with limited payload and a tight power budget cannot provide.

Each of these approaches is unsuitable for a small drone in one of two ways. Radio and acoustic methods add extra infrastructure to every space, while the vision approach adds a camera and computation that the drone cannot carry. Visible-light signals, on the other hand, provide a potential solution to address these challenges. Modern indoor spaces are already well lit by existing lighting infrastructure. Thus, if we can utilise these light sources as transmitters, we can avoid installing additional anchors in the room; an ordinary lamp can already be used as a navigation beacon [25]. The drone then needs to carry only a lightweight light sensor, whose energy consumption and mass are well within a micro-drone’s budget.

Visible-light positioning is among the most accurate indoor navigation approaches based on optical signals, i.e., visible light or infrared light at wavelengths from roughly 380 nm to 1 μ m [13]. Through triangulation, such systems can localise a receiver with centimetre-level accuracy by using a photodiode to read signals from several modulated ceiling luminaires [19, 32]. However, triangulation requires the photodiode to receive signals from at least three light sources at the same time. As a result, the system needs a dense and fixed grid of light sources to achieve the claimed accuracy, which can be costly to provide. The light infrastructure must also satisfy two requirements: *1) multiple light sources must overlap densely so that the receiver is guaranteed to observe at least three different signals, and 2) each light beacon must be installed at a precise location that is measured in advance.* Many spaces cannot arrange lighting so regularly, such as street lamps or aisle lighting bulbs. Thus, although visible-light positioning can provide accurate position information, it depends on dense

infrastructure that is often nontrivial to deploy in every target environment.

A single light beacon, by contrast, places a far smaller demand on the space. With only one source in view a drone cannot recover its position through triangulation, since triangulation requires at least three light sources. It can, however, recover the direction of the light, which is sufficient for the drone to navigate toward a target light source. This makes the single beacon a deliberately minimal design rather than a compromise. What it forgoes against dense positioning is needing to determine the full position of the drone, since a direction alone is enough to reach the beacon. What it gains is a large reduction in the infrastructure the space must provide: whereas dense positioning depends on a grid of luminaires installed and surveyed to known positions, a single beacon needs only one ordinary modulated lamp. The drone meanwhile carries the same light sensor it would for any light-based method, so this saving comes at no added cost on board. Reducing the requirement to a single modulated lamp is what makes the approach practical in the many spaces that cannot host, or do not warrant, a dense positioning grid. Huang et al. [15] demonstrate such single-beacon navigation on a micro-drone.

Table 2.1: Indoor navigation approaches, set against two costs: the infrastructure that must be installed in the space, and the sensing and computation the drone must carry. The single light beacon, the approach this thesis builds on, keeps both small.

Approach	Infrastructure to install	On-board sensing and compute
Ultra-wideband	Dedicated radio anchors	Radio transceiver
WiFi fingerprinting	Prior site survey	WiFi radio
Acoustic	Ultrasonic anchors	Microphone
Visual SLAM	None	Camera and heavy computation
Dense visible-light positioning	Grid of modulated luminaires	Photodiode
Single light beacon	One modulated lamp	Photodiode

Recovering a direction, rather than a full position, from an array of sensors provides an alternative way to support navigation. Prior work has explored this idea using radio signals, where multiple antennas receive signals from different directions and estimate the transmitter’s bearing angle by comparing signal strength across the antennas [17]. Single-beacon light navigation applies the same principle, but uses photodiodes and light instead of radio signals and bulky antennas.

As demonstrated by Huang et al. [15], single-beacon light-guided navigation enables a drone to reach a single target in an unobstructed open space. However, two problems remain before this approach can support fully autonomous drone operation. *First, if an obstacle lies between the drone and the transmitter, it can block the light signal and cause the navigation mission to fail. Second, autonomous drone applications often require the drone to visit multiple targets in a predefined order, whereas the previous approach tracks only one light beacon at a time and therefore cannot directly support more complex missions.* This thesis addresses these limitations through a custom sensing board and a controller that

builds and follows a map of the light field, thereby improving the robustness of autonomous drone navigation under single-beacon guidance. We first review the methods we build on in Section 2.4.

2.2 Drone Platform

In this thesis, we build our platform on the Bitcraze CrazyFlie 2.1, an open-source research quadrotor weighing about 27 grams and small enough to be held in one hand [5]. We chose it because it is openly programmable and provides an interface that makes it easy to integrate customised sensors. However, its limited payload capacity and onboard computation constrain both the sensor choices and algorithmic complexity it can support.

The CrazyFlie can localise itself accurately with the Lighthouse positioning deck, which tracks infrared signals emitted by base stations placed around the room [3]. Yet this option, like the radio and dense visible-light methods discussed in Section 2.1, depends on infrastructure installation and calibration before the first flight. Since our goal is to enable navigation with minimal changes to the space, we set such methods aside and instead adopt a simpler setup in this thesis.

Although the Crazyflie offers only a small payload budget, it still leaves several grams for an additional sensor board. In our measurements, the Crazyflie 2.1 can lift roughly 15 grams of payload. Because this thesis uses the Flow Deck V2 for onboard odometry and ground-truth recording, part of this budget is already occupied, leaving only about 13 grams for the sensing hardware added in this work [5, 4]. The added hardware must also operate within the limited current available from the drone’s small lithium battery. Together, these payload and power constraints define the design requirements for the photodiode board presented in Chapter 4.

2.3 Visible Light Communication & Sensing

With the platform and its budget fixed, we turn our focus to the light signal the navigation system is built upon. This section sets out how an ordinary lamp becomes a beacon, and why the intensity a single photodiode reads gives only a direction, not a position.

Light suits the task because the space is already lit, as Section 2.1 set out: the lamps that illuminate a room can also serve as beacons without new infrastructure installation. Visible light communication turns a lamp into such a beacon by modulating it, switching its output faster than the eye can follow [25]. The lamp continues to light the room as before, so a person sees an ordinary, steady light, while a photodiode sampling it quickly enough reads the modulation as a signal. The lighting thus keeps its original purpose and gains a second one, carrying a beacon signal at no cost to the illumination it provides. In this work each beacon is a lamp switched on and off at a fixed frequency, which distinguishes it from steady background light and from beacons modulated at other frequencies.

How strong a beacon’s signal can be detected by the drone depends on more than distance. Two angles also matter: the angle at which the light leaves the

source, and the angle at which it meets the photodiode. Each angle contributes a cosine factor, and the intensity falls with the square of the distance. Together these form the Lambertian model [18]. Figure 2.1 shows the geometry. Writing α for the angle off the source’s axis, β for the angle of incidence at the sensor, and d for the distance, the received intensity is proportional to $\cos^m \alpha \cos \beta / d^2$, where the order m describes how narrowly the source concentrates its light. Intuitively, the light weakens as either the source or the sensor turns away from the line between them. The order m sets how sharply the source’s contribution falls off, with a larger m for a more tightly focused source.

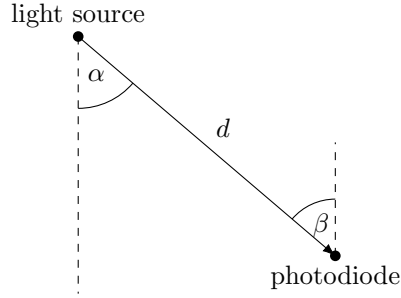


Figure 2.1: **The Lambertian geometry. The light leaves the source at an angle α off its axis, travels a distance d , and meets the photodiode at an angle β off the sensor’s normal. The received intensity is proportional to $\cos^m \alpha \cos \beta / d^2$.**

This dependence on both angle and distance makes a single intensity reading ambiguous. A weak reading can arise for two different reasons: the emitter may be far away, causing the received intensity to decrease with distance, or the light may arrive at a large angle from the photodiode’s normal direction, reducing the received signal as shown in Figure 2.2. In other words, when a photodiode measures a light intensity, it cannot determine the light incident direction from the photodiode’s reading alone. This ambiguity is why a single light source cannot determine an absolute position, and why visible-light positioning systems usually combine multiple overlapping sources. In this thesis, we avoid this requirement by treating light as a direction to follow rather than a position to solve for.

2.3.1 The Light Signal

The drone senses the target light with photodiodes. A photodiode produces a small electrical current proportional to the light striking it, and the board converts that current into a value. The navigation reads one value per photodiode, which we call the per-sensor light signal and write S_n for sensor n . It measures how strongly that sensor sees the beacon. How S_n is recovered, however, differs between the two settings this thesis uses. On the real drone it is extracted from a noisy voltage, as Chapter 5 describes, and in simulation it is read directly from a modelled sensor, as Section 2.5 describes. For the algorithms that follow, it is enough that each sensor reports one such value.

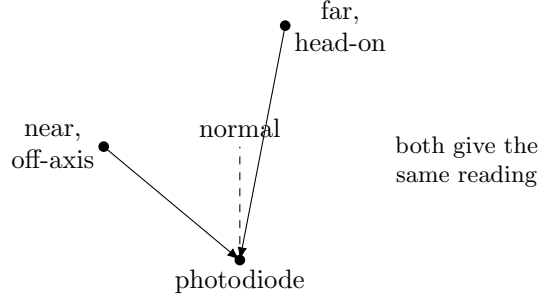


Figure 2.2: **The intensity ambiguity.** A near source seen well off the sensor’s axis and a far source seen head-on can deliver the same reading to a single photodiode, so one reading does not fix the source’s position.

2.4 Algorithms from Huang et al.

This thesis builds on the single-beacon algorithms of Huang et al. [15]. They propose three: bearing angle guidance (BAG), dither-free gradient ascent (DGA), and dither extremum seeking (DES). We build on BAG and DGA, and describe DES to explain why we do not. Section 2.4.3 compares all three.

2.4.1 Bearing Angle Estimation

Bearing-angle guidance recovers the direction of a light beacon using an array of photodiodes [15]. Each photodiode in the array faces a different known direction and receives the strongest signal when light arrives from that direction. When a beacon lies to one side of the array, the photodiodes facing the beacon measure stronger signals than those facing away. Comparing the signal strengths across the array therefore reveals the beacon’s direction.

The bearing combines these per-sensor readings into a single direction. Each photodiode contributes a vector pointing the way it faces, weighted by how strongly it sees the beacon, and we sum these vectors. Figure 2.3 shows this for a ring of eight photodiodes. Using the per-sensor light signal S_n and writing θ_n for the direction sensor n faces, the bearing ψ_B is the angle of the weighted sum,

$$\psi_B = \text{atan2}\left(\sum_n S_n \sin \theta_n, \sum_n S_n \cos \theta_n\right). \quad (2.1)$$

Sensors facing closer to the beacon contribute more strongly to the weighted sum, so the weighted sum of the light-signal strengths points toward the dominant light direction. By spacing eight photodiodes evenly around the ring, the array samples the full 360-degree horizontal plane; for any beacon direction, one or more nearby sensors receive stronger signals than the rest. The resulting bearing is therefore an estimate of the beacon direction projected onto the plane of the photodiode ring.

BAG provides the light-source direction quickly and accurately. From a single set of readings it points at the beacon, with no need to move or explore, which makes it the best solution for navigation in open space [15]. Its weakness is that

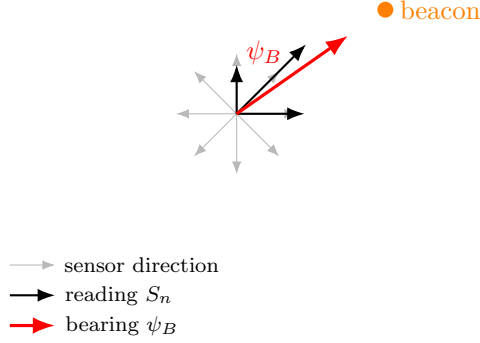


Figure 2.3: The bearing as a magnitude-weighted vector sum on a ring of eight photodiodes. The grey arrows are the directions the sensors face; the black arrows are their readings, longer where the sensor sees the beacon more strongly; their sum, in red, is the bearing ψ_B , which points toward the beacon.

it has no memory from the previously passed space. Each estimate uses only the latest readings, so the drone steers toward whatever light is strongest at that instant, even when that light is a reflection. The readings alone do not record the space the drone has crossed, so the bearing cannot tell that an obstacle stands between the drone and the beacon. Steering on the bearing alone, the drone heads directly toward the strongest light and cannot plan a path around an obstruction.

2.4.2 Gradient-Based Source Seeking

Whereas the bearing is reactive and memoryless, gradient-based source seeking lets the drone use the history of what it has sensed. It treats the light as a field that grows toward the beacon, and moves in the direction in which it estimates the field rises fastest. Following that gradient leads to the peak of the field, which for a light field is the beacon itself. Because it is built from measurements gathered across a region, the gradient still gives a direction when the beacon is briefly out of direct view.

The difficulty is measuring the gradient, and Huang et al. propose two ways to do it [15]. The first, dither extremum seeking, uses a single photodiode and adds a small oscillation, or dither, to the drone's path. As the drone oscillates, it measures how the light changes and infers the uphill direction. This is the classical extremum-seeking approach to source seeking without knowledge of the drone's own position [2, 31, 7]. Its drawback is the dither itself, the constant oscillation produces indirect, inefficient flight, and the probing motion is easily overwhelmed by disturbances. Thus, we choose not to use DES in this thesis.

The second, dither-free gradient ascent, removes the oscillation. It still uses a single photodiode, but instead of probing around it estimates the gradient from the drone's recent motion. The drone records where it has been and how strong the light was there. It then fits a plane to these recent measurements by least squares. The slope of that plane is the gradient [15]. This avoids the dither, but it still relies on a single sensor and on a position estimated by dead

reckoning. The drone accumulates its own motion with no external reference, so the estimate drifts over time. The method is therefore slow, since the drone must keep moving to gather points, and it is sensitive to that drift. This dither-free gradient ascent is the gradient method this thesis builds on. Chapter 3 shows how estimating the gradient from all eight photodiodes and a map of the field, rather than from one sensor and recent motion, removes the single-sensor and dead-reckoning weaknesses just described.

2.4.3 Comparing the Algorithms

The three methods differ in how directly and how quickly they reach the source, and in the sensing hardware this requires. The bearing flies the most direct route, in 7.3 metres and 8.4 seconds, but it requires the full photodiode array. Both single-photodiode methods wander further and are markedly slower, taking two to three times as long to reach the source. Dither-free gradient ascent takes 8.6 metres and 21.4 seconds, while the dithering extremum seeker is the slowest and least direct at 12.2 metres and 25.6 seconds, its oscillation lengthening the path. Table 2.2 sets the three side by side, with the path length and travel time Huang et al. measured for each on the same task [15]. The bearing and the dither-free gradient are the two this thesis builds on, and Chapter 3 develops how they are combined.

Table 2.2: **The three single-beacon navigation algorithms of Huang et al. [15], with the path length and travel time each took to reach the source on the same task in their evaluation. This thesis builds on the bearing and the dither-free gradient.**

	Bearing guidance (BAG)	angle	Dither extremum seeking (DES)	Dither-free gradi- ent ascent (DGA)
Sensing	Photodiode array		Single photodi- ode	Single photodi- ode
Path length (m)	7.3		12.2	8.6
Travel time (s)	8.4		25.6	21.4
Main weakness	No memory of the field		Inefficient oscil- lating flight	Slow; depends on dead reckoning

2.5 Simulation Environment

We develop the controller in simulation before flying it on real hardware. Simulation runs a scenario repeatedly under identical conditions, and lets a controller fail without risking the drone. This makes it well suited to designing and comparing the navigation controllers of this thesis.

We use Webots, an open-source robotics simulator [8], with the model that Huang et al. built for their work [15]. The model reproduces the flight of their lighter-than-air blimp and the light its sensors receive. Huang et al. validated the flight dynamics against published aerodynamic data and real flights, so we take the physical behaviour as given. The simulated airframe is therefore the

blimp of Huang et al., not the CrazyFlie this thesis flies. This does not affect the navigation we study, which acts only in the horizontal plane and so should carry over despite the different airframe. Chapter 7 shows that flying the same controller on both the CrazyFlie and the blimp gives a similar result.

We deliberately simplify light sensing in the model. Webots can render light–environment interactions through ray tracing, but doing so for a fast-modulated light source is computationally expensive and unnecessary here, since our focus is navigation given the received light intensity rather than detailed optical interactions between the light and the environment. Each photodiode is therefore modelled as a Webots light sensor that directly reads the intensity reaching it from the light source, and we use that reading as the per-sensor light signal S_n .

Chapter 3

Problem & Algorithm Design

This chapter designs the navigation controller at the core of this thesis. Huang et al. [15] showed that a weight-, size-, and power-constrained micro-drone can reach a single light source in open space, using either a bearing read from a photodiode array or a gradient climbed with a single photodiode. This thesis aims to fill a gap in the work of Huang et al.: reaching several modulated beacons in succession while passing obstacles between them, using only on-board light sensing. We first show the existing methods do not solve this problem. The design question is therefore how to obtain obstacle-aware, multi-beacon navigation from light alone, within the constrained budget of the micro-drone platform we chose.

To overcome these limitations, our controller combines the two methods of Huang et al. [15] and adds a spatial memory of the light field. The bearing points toward a beacon the photodiode array can currently see, while the gradient, estimated from a map of the light the drone makes as it flies, still gives a direction when the beacon is hidden. By fusing the two, the drone steers on the bearing while the beacon is in view and falls back on the remembered light field when an obstacle blocks it. Figure 3.1 shows how the controller derives both headings from the same photodiode array and fuses them into a single heading.

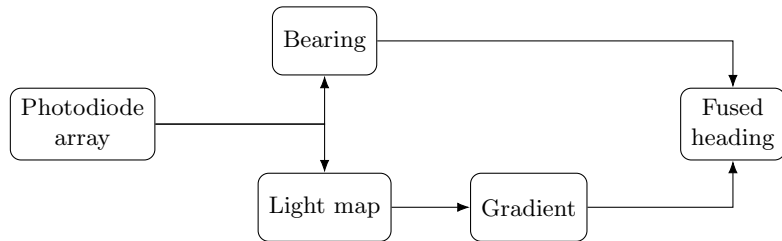


Figure 3.1: **The fused controller in concept. The same photodiode array yields two headings: a bearing read directly from the array, and a gradient estimated from a light map the drone builds as it flies. The controller fuses the two into a single heading toward the beacon.**

This spatial memory that we add rests on a second change to the sensing. The gradient method of Huang et al. [15] reads the light with a single photodiode, which sees the beacon only while it faces the source, so its estimate depends on the drone’s heading. We instead sense with a ring of eight photodiodes, so the photodiode array sees the beacon however the drone is turned. The drone can then record a heading-independent light value at every position it visits, which is what makes the light-field map possible.

The remaining sections develop the controller one component at a time: the bearing estimate, the separation of simultaneous beacons by modulation frequency, the light map, the gradient estimator and the choice among candidate estimators, the state machine that sequences a mission, and the heading and speed commands issued to the flight controller.

3.1 Why Existing Methods Fall Short

In the system proposed by Huang et al. [15], bearing-angle estimation and gradient-based estimation are first developed as single-beacon navigation primitives and evaluated in source-seeking tasks. This thesis examines two conditions that are more complex and closer to real-world applications than the baseline setting. The first is obstacle occlusion, where an object stands between the drone and the beacon and can block the light signal. The second is multi-target navigation, where the drone must visit several beacons in a predefined order. The following paragraphs explain why the original system is insufficient under these conditions, and then present simulations that demonstrate the failure cases this thesis addresses.

The bearing is an instantaneous estimate, formed from the relative magnitudes of the eight photodiode signals at a single instant as explained in Section 2.4.1. Because it requires no integration over the drone’s motion, it yields a heading immediately whenever the array receives the beacon’s light, whether directly or through a reflection. This immediacy is also its central limitation: the bearing describes only the direction of the light arriving at that instant and retains no record of the surrounding space. When an obstacle blocks the beacon, every photodiode sees only background light, so the readings no longer differ across the ring. Their weighted sum then points in no particular direction, and the bearing gives the drone neither a heading to follow nor any sign that an obstacle is in the way.

The single-photodiode gradient method, the second method of Huang et al., recovers a direction using one sensor instead of an array. Because a single sensor measures only one light value at a time, this approach is slow and fragile. It cannot sense a spatial gradient directly; instead, it infers the gradient from how the light changes between successive positions. As Figure 3.2 illustrates, this forces the drone to keep moving before it can estimate a direction, while also requiring the sensor to remain oriented toward the light. The method therefore depends strongly on the drone’s heading, because one fixed photodiode receives the direct beam only when it faces the source.

We examine these failures in simulation before testing on hardware, for two reasons. A simulated light field carries none of the ambient light or electrical noise of a real flight, so an observed failure can be attributed to the navigation method rather than to the photodiode choice limitation. The simulator also

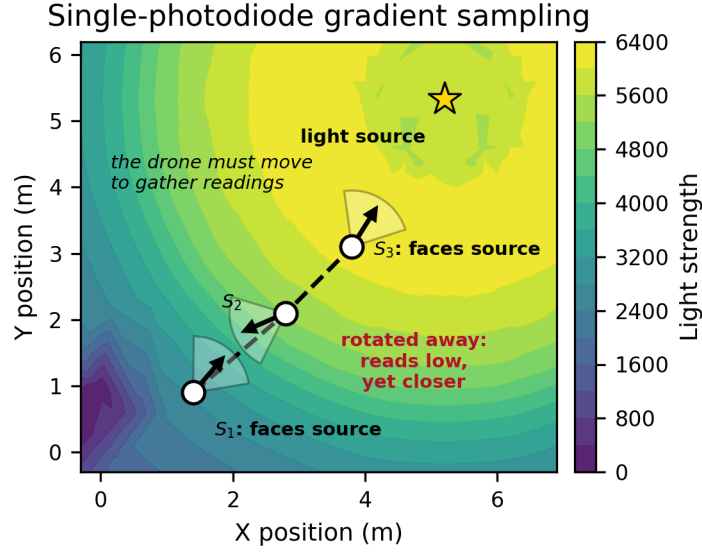


Figure 3.2: **Single-photodiode gradient estimation over the light field.** A single sensor reads only one value at a time, so the drone must keep moving to compare readings S_1 , S_2 , and S_3 along its path before it can infer which way the field rises, which makes the method slow. The reading also depends on the drone’s heading: at the middle position the drone has rotated away from the source, so it reads low even though it is closer than the first location, which corrupts the inferred gradient. Establishing a gradient therefore requires the drone both to keep moving and to keep its one sensor facing the light. The figure is an illustration drawn over real field data.

runs the same scenario repeatedly, at low cost and with no risk of damaging the aircraft, which is a critical requirement for comparing different navigation methods.

The comparison uses the Webots environment mentioned in Section 2.5, shown in Figure 3.3. A single beacon sits above the floor, a pillar stands between the drone and the beacon, and the drone starts on one side, so the pillar obstructs the direct path to the beacon. Each controller flies this same scene, from the same start position to the same beacon, with identical sensing and flight settings, so the navigation method is the only variable between runs.

The bearing-only controller steers directly along the measured light direction. Since the pillar lies slightly off the direct line to the beacon, the bearing angle direction just points past its edge, and the controller follows that direction with no indication that it leads into the obstacle. The drone therefore collides with the pillar, as Figure 3.4 shows.

The single-photodiode gradient controller also fails to reach the beacon. As Figure 3.5 shows, it follows a slow and irregular path and stalls when it reaches the pillar. This failure comes from how the gradient is estimated. The drone observes a local increase in light intensity only after it has already pressed

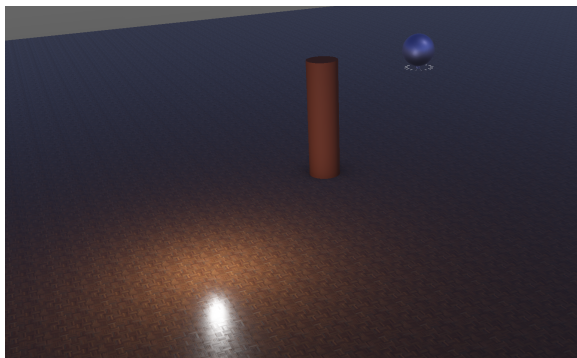


Figure 3.3: **The simulated test environment in Webots. A single beacon illuminates the floor, a pillar stands in the line of sight, and the drone starts to one side.**

against the pillar and begun moving around it. Once the sensor detects a stronger light signal after partially passing the obstacle, the gradient estimate produces a large corrective pull away from the pillar. Although this response helps the drone avoid continuous contact with the obstacle, it also pulls the drone away from the target direction, preventing it from reaching the beacon.

The two failures show the trade-off between the two approaches. The bearing is fast and accurate but has no memory beyond the current line of sight. The gradient tolerates a blocked line of sight but is slow and must read from accumulated measurements. A new type of controller is therefore developed in this thesis. We need a controller that maintains a short spatial memory. When an obstacle blocks the beacon, it uses the remembered light field to estimate a gradient and steer around the obstacle. Once past the obstacle, it follows the bearing again. The remainder of this chapter develops this controller. The results are reported in Chapter 7.

Neither baseline supports more than one beacon, so neither can perform the multi-beacon traversal this thesis targets. We do not demonstrate the multi-beacon case in simulation. As Section 2.5 explains, the simulator does not ray-trace the light but reads each sensor’s intensity directly, and it does not reproduce the on-off modulation. Without that modulation the beacons cannot be told apart, so we design the multi-beacon navigation in Chapter 6.

3.2 System Overview

The failures of the previous section define two requirements for the controller. First, the drone must move efficiently toward a visible beacon. Second, it must continue toward the beacon when the direct path is blocked. The controller satisfies the two requirements with two separate algorithms, each addressing one requirement while using data from the same photodiode array. The first is the bearing, which gives an immediate heading whenever the array receives the beacon’s light. The second is a gradient, estimated from a light map that the drone builds as it flies; because the map records where the light was stronger, the gradient still gives a direction when the beacon is not directly visible.

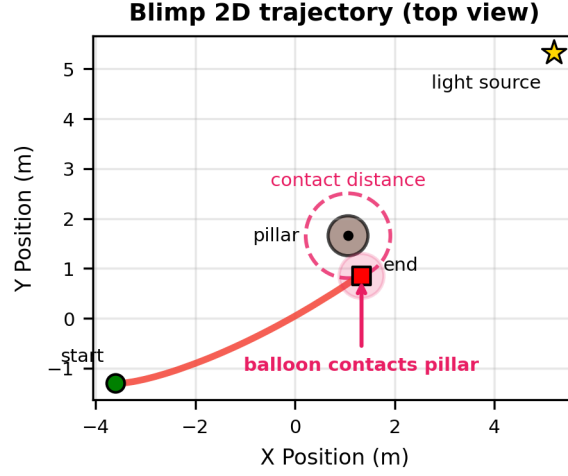


Figure 3.4: **Top-down trajectory of the bearing-only controller in simulation. The drone follows the measured light direction past the edge of the pillar and collides with it.**

Neither algorithm is always trustworthy, so the controller must decide which to use at each moment. The bearing is not reliable when its magnitude is low, which happens when the beacon is weak or blocked as explained in Section 3.3. The gradient is meaningless until the drone has measured the light over a spread of positions, as described in Section 3.5. A state machine compares the validity of the two algorithms and selects the drone’s behaviour: it follows the bearing estimate while its signal is reliable, adds the gradient once the map is dense enough and its estimate has high confidence, and follows the gradient alone when the bearing is invalid.

Each control update follows the same sequence, summarised in Figure 3.6. The per-sensor light signals S_n of Section 2.3.1 are first stored into a light map. The direction toward the light is then estimated from that map through gradient estimation. The bearing and the gradient estimate are then merged into a single target heading by the state machine.

The state machine does more than merge the two estimates. It selects the navigation strategy and forward speed for the current beacon, and it determines when the target has been reached and when the controller should advance to the next.

3.3 Bearing Angle

Before the controller can rely on the bearing, it must turn the raw bearing angle of Huang et al. [15] into a usable heading. That estimator, explained in Section 2.4.1, returns an angle from a single set of photodiode readings, but it does not smooth the angle between readings, nor does it say whether the angle can be trusted. This section adds the two steps that supply both: a validity gate that decides whether the bearing is reliable, and temporal smoothing that

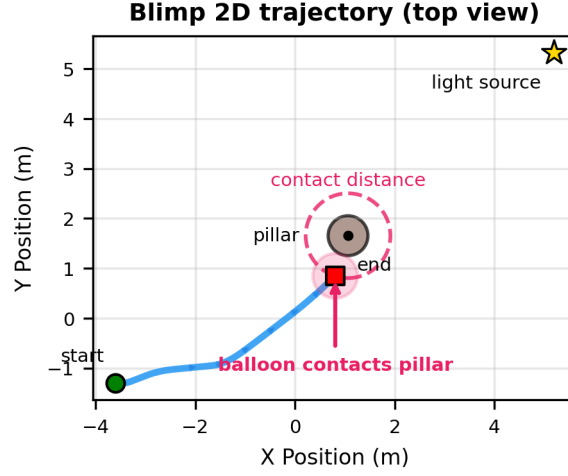


Figure 3.5: **Top-down trajectory of the single-photodiode gradient controller in simulation. The path is slow and irregular and stalls near the pillar, never reaching the beacon.**

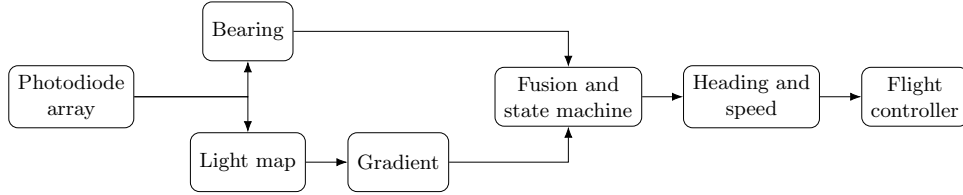


Figure 3.6: **Overview of the fused navigation controller. The photodiode array feeds both a bearing estimate and a light map; the gradient is estimated from the map; the fusion and state-machine stage merges the bearing and gradient into a target heading and forward speed, which are passed to the drone’s flight controller.**

steadies it.

The bearing is only reliable when the photodiode array actually receives the light from a beacon. With no beacon in view, every photodiode sees only ambient light, so the readings carry no directional information: their vector sum has near-zero magnitude and its angle is set by noise. The bearing cannot flag this on its own, so the controller adds a validity gate. It computes a validity score by summing the light-intensity magnitudes measured across the array, and uses the bearing only when this score exceeds a minimum threshold, which rejects headings that are really noise from a weak, distant, or absent beacon. The gradient is gated separately, by its own reliability conditions in Section 3.5.

A bearing that passes this test is still noisy between samples. Reflections, sensor noise, and small changes in attitude make the raw bearing jitter from one frame to the next, and steering on each raw value would make the drone’s heading jerk. The controller therefore smooths the bearing with an exponential

moving average, which gives steadier flight. The update is

$$\hat{\psi}_B \leftarrow \hat{\psi}_B + (1 - \beta) \Delta, \quad (3.1)$$

where Δ is an angular difference between the new bearing and the current estimate $\hat{\psi}_B$, where β controls the strength of the filter. A larger β rejects more noise but responds more slowly to a genuine change in direction.

3.4 Light Intensity Map

Gradient-based navigation needs a record of how the light varies across space, which the bearing alone does not keep. The controller builds this record as a light map: it divides the horizontal plane into a regular grid and stores one light value per cell, indexed by the drone's estimated position in a fixed world frame. The drone does not survey the area first and then navigate. It fills the map in as it flies, writing the current reading into the cell it occupies on every control update, and estimates the gradient from whatever the map holds so far. Figure 3.7 shows a survey of a flight area. A real flight fills only a small subset of this map and estimates the gradient from that partial coverage.

Each cell stores the strongest of the eight per-sensor signals S_n measured while the drone occupies it, using the per-sensor signal of Section 2.3.1. The controller indexes the cell at the drone's estimated position and updates it on every control update. Because the drone moves slowly relative to the sampling rate, it stays within one cell for multiple consecutive samples, and its path can later cross the same cell again, so each cell is measured multiple times rather than once. Rather than overwrite the stored value with each new sample, the controller blends them through an exponential moving average,

$$I_{\text{cell}} \leftarrow \alpha I_{\text{cell}} + (1 - \alpha) I_{\text{new}}, \quad (3.2)$$

which averages out the sample-to-sample fluctuation seen when the same cell is measured repeatedly. The factor α weights the stored value against the new one, set to $\alpha = 0.5$ by default. A smaller α tracks new measurements more closely at the cost of more noise, while a larger α gives a smoother but slower-adapting map.

The cell index depends on the drone's own position estimate, which is not exact. Onboard odometry drifts slowly over time, as all dead-reckoning estimators do. As it drifts, measurements are gradually stored in the wrong cells, smearing the recorded intensity across neighbouring cells and blurring the gradient estimated from the map.

Two design choices limit the effect of this drift. First, the map is reset whenever the drone switches to the next beacon, so the error accumulated while approaching one beacon is discarded with its map rather than carried into the next, and drift cannot build up across a whole mission. Second, the map is used only to estimate a heading, not the beacon's absolute position. Even as the position estimate drifts, the direction in which the light rises fastest across nearby cells stays accurate enough to steer by.

Across the visited area the map approximates a scalar field that rises toward the beacon, because the measured signal grows as the drone nears the source. This field is the spatial memory the bearing lacks, and the next section explains how we estimate a direction from it.

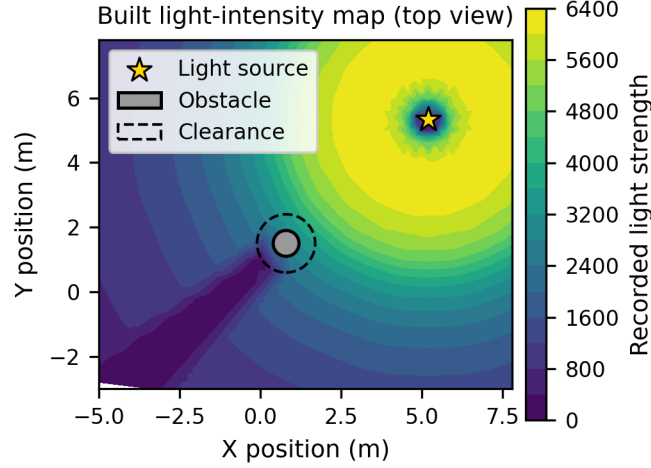


Figure 3.7: A light-intensity map of the simulation, viewed from above. Each grid cell holds the strongest light signal recorded while the drone occupied it, drawn here as a smooth field. The strength rises toward the beacon, marked by the star, and this is the scalar field whose gradient the controller follows. The grey disc is the pillar, and the dashed circle the clearance the drone has to keep around it. Since the light comes from above and the light sensors are placed on the horizontal plane there is a drop in recorded light strength when very close to or below the light source.

3.5 Gradient Estimation

The gradient gives the drone a direction to follow when the bearing is unavailable. It derives a different heading from the bearing: the bearing points along the light arriving now, whereas the gradient points in the direction in which the stored light field increases fastest, which remains defined even when the beacon is hidden. Huang et al. [15] estimated this direction from a single photodiode, by reading how that sensor’s signal changed as the drone moved. That approach is slow, because the drone must keep moving to gather readings, and fragile, because a single fixed sensor measures the source only when it faces the light. Both of these problems can be seen in Section 3.1. Our controller instead estimates the gradient from the light map described in Section 3.4. Because the map is built from the full photodiode array, it records a usable value at each visited location regardless of the drone’s heading. This removes the dependence on orientation and motion that limits the single-photodiode method.

The controller estimates the gradient from the recorded light map by fitting a plane to it. Around the drone’s current position it takes the cells $\{(x_i, y_i, I_i)\}$ and fits $I(x, y) = ax + by + c$ by weighted least squares, weighting each cell by

the inverse square of its distance to the drone so that nearer cells have greater influence on the estimate. The fitted coefficients give the gradient $\mathbf{g} = (a, b)$, and the controller steers along $\mathbf{g}$ when navigating on the gradient.

A gradient estimate, like a bearing, can be unreliable, so the controller needs a measure of its validity. The plane fit supplies one directly. If the nearby cells lie close to the fitted plane, the slope it gives is trustworthy; if they are scattered, it is not. We quantify this agreement with a weighted coefficient of determination, R^2 , which measures the fraction of the variation in the cell values that the fitted plane explains,

$$R^2 = 1 - \frac{\sum_i w_i (I_i - \hat{I}_i)^2}{\sum_i w_i (I_i - \bar{I}_w)^2}, \quad (3.3)$$

where $\hat{I}_i$ is the plane’s value at cell i , $\bar{I}_w$ is the weighted mean of the cell values, and w_i are the inverse-distance weights. An R^2 near one means the cells lie close to a single plane, so the fitted direction is trustworthy. A low R^2 means they do not, so the gradient should not be followed.

Three conditions must hold before the controller can trust a gradient estimation result. First, the fitting process must use at least a minimum number of cells, because a plane fitted through only two or three points is usually noisy and can produce a random pointing direction. Second, its R^2 must exceed a threshold, which rejects the fitting results where the cells do not lie on a consistent slope, as happens where the field is flat or where the map is still patchy. Last, its magnitude must be large enough to be meaningful, so the drone does not chase a near-flat field. The thresholds, with the controller’s other parameters, are listed in Appendix A. These three conditions define the gradient reliability used by the state machine. While they fail, the drone navigates on the bearing alone. Once they hold, the controller blends the gradient into the heading, or follows it alone when the bearing is lost.

3.5.1 Selection of the Gradient Estimator

Estimating a gradient from sparse and irregularly spaced map cells is a well-studied problem, but it is not obvious which estimator is best suited to the light maps produced during flight. Different methods make different trade-offs between robustness, accuracy, computational cost, and the ability to produce an estimate when map coverage is limited.

Since the gradient estimator runs continuously on the drone’s microcontroller, the chosen method must not only be accurate but also execute within the available computational budget. We therefore compare several candidate estimators before selecting the one used in our experiments. Five candidate methods were evaluated:

- Ordinary least-squares plane fitting (OLS) [6].
- Weighted least-squares plane fitting (WLS) [27].
- RANSAC plane fitting [9].
- Local polynomial surface fitting [6].
- Finite-difference estimation [20].

All methods estimate the local gradient from neighbouring map cells but differ in how they treat noise, sparse sampling, and outliers.

Evaluation Procedure

The methods were evaluated using a densely sampled reference map generated from the simulated light field. This reference map includes an obstacle creating a shadow. This lets us evaluate the estimators on the task they are actually used for: guiding the drone around obstacles and out of shadows. At each map location a method either produced a gradient estimate that passed the reliability criteria of Section 3.5 or returned no estimate. Three criteria were considered:

1. Success rate: how often a usable estimate is produced.
2. Directional accuracy: how closely the estimated direction matches the direction towards the light.
3. Computational cost: the processing required for a single estimate.

Success rate is the most important criterion because the gradient’s primary purpose is to remain available whenever the bearing is lost. An estimator that is slightly less accurate but available most of the time is more useful to the controller than a highly accurate estimator that frequently returns no result.

Directional accuracy was measured using the angular error:

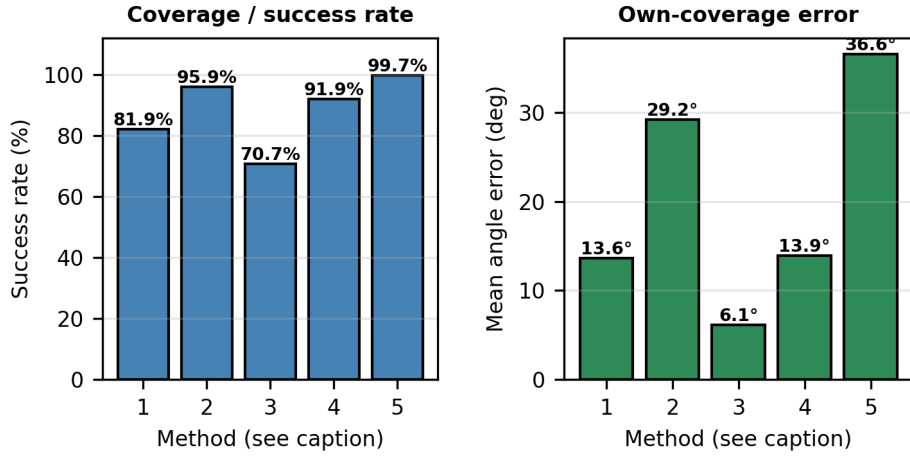
$$e_i = \left| \left(\hat{\theta}_i - \theta_i^* + 180^\circ \right) \bmod 360^\circ - 180^\circ \right|, \quad (3.4)$$

where $\hat{\theta}_i = \text{atan2}(\hat{b}_i, \hat{a}_i)$ is the estimate’s heading and $\theta_i^* = \text{atan2}(y_L - y_i, x_L - x_i)$ points from cell i to the source at (x_L, y_L) , and the wrap folds the unsigned difference into $[0^\circ, 180^\circ]$. When computing the error we discard cells within ten centimetres of the source, where the field is too steep to give a meaningful reference.

Directly comparing average angular error using the same number of data points can be misleading, because the methods estimate the gradient in different ways and therefore require different amounts of local map data before they can produce a valid estimate. Some methods return an estimate almost everywhere, whereas others withhold an estimate when the local map is sparse or poorly fitted. As a result, the average error for each method is computed only over the locations where that method produces a valid estimate, rather than over the same set of time steps for all methods. Figure 3.8(b) reports this own-coverage error for each method.

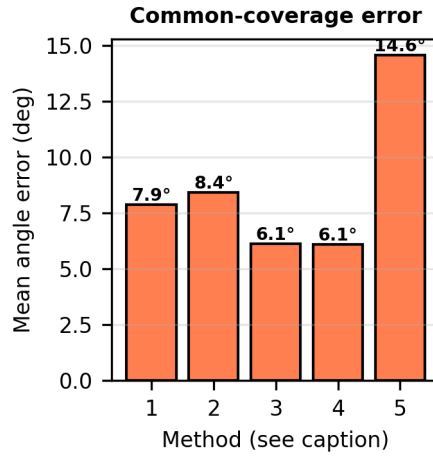
While angular accuracy is important for drone navigation, the coverage of a gradient estimator is also critical. We therefore report an additional coverage metric, defined as the estimation success rate. For each estimator, we divide the number of map cells where it produces a valid gradient estimate by the total number of cells in the map. Figure 3.8(a) reports this success rate, the ratio of the map over which each estimator can provide a usable navigation direction.

To provide a fairer comparison of angular accuracy, we also evaluate the estimators on the common set of map cells where all candidate estimators return valid results, which Figure 3.8(c) reports. In this analysis, a cell is included only if every estimator produces a valid gradient estimate at that location.



(a) Success rate: the share of map cells where the method returns a usable estimate.

(b) Own-coverage error: mean angular error over each method's own valid cells.



(c) Common-coverage error: mean angular error over the 284 cells that every method estimates.

Figure 3.8: Coverage and accuracy of the five gradient estimation methods on the reference map. The bars are numbered: (1) Ordinary Least Squares, (2) Weighted Least Squares, (3) RANSAC, (4) Polynomial Surface, (5) Finite Differences. (a) the share of cells where each method returns a usable estimate; (b) the mean angular error over each method's own valid cells; (c) the mean angular error over the common set of cells that every method estimates. Accuracy is measured as the angular error against the straight-to-source direction.

This avoids comparing methods over different parts of the map and isolates the accuracy difference among estimators by evaluating them under the same spatial conditions.

The distinction is important. Consider two methods: one returns estimates at 95% of locations, while another returns estimates at only 70%. The lower-coverage method can avoid many of the difficult locations simply by refusing to estimate there. Its average error may therefore appear lower, not because its estimates are inherently better, but because it is evaluated only on easier cases. The common success points comparison removes this effect by ensuring that every method is scored on exactly the same locations.

This behaviour is visible in the results that we will show later. Weighted least squares appears substantially less accurate when evaluated over its own coverage because it continues to provide estimates in difficult regions, including locations close to the source and behind the obstacle. When the comparison is restricted to the common subset, where all methods are evaluated on the same points, the differences between the plane-fitting methods become much smaller. This indicates that most of the apparent accuracy gap is caused by differences in coverage rather than differences in the quality of the estimated direction itself.

Results Evaluation

Figure 3.9 provides a qualitative view of the spatial behaviour of the five estimators across the reference field. Rather than displaying the exact angular error at each location, the figure shows whether each estimate satisfies the controller’s reliability criteria. This visualisation reflects each estimator’s intended role within the navigation system, where the key question is whether a given gradient is usable for the drone to navigate towards the light source and avoid the obstacle.

The figure shows that weighted least squares provides reliable estimates across most of the environment, including regions where map coverage is sparse. RANSAC rejects many locations, while finite differences remain available almost everywhere but generate noticeably noisier directions. Figure 3.8 summarises the results quantitatively.

We consider the success rate first, as it is the most important criterion. As Figure 3.8(a) shows, the finite-difference estimator is available almost everywhere, with a success rate of 99.7%, and weighted least squares follows closely at 95.9%. The polynomial surface and ordinary least squares cover 91.9% and 81.9% of the map, while RANSAC covers the least, at 72.3%. Because the gradient’s main purpose is to remain available whenever the bearing is lost, this coverage is the metric that matters most.

The own-coverage error in Figure 3.8(b) must be read against these coverage figures. On its own valid cells RANSAC has the lowest mean angular error, at 5.9° , followed by ordinary least squares and the polynomial surface at 13.6° and 13.9° , while weighted least squares and finite differences appear worse, at 29.2° and 36.6° . This ranking is misleading on its own. The high-coverage methods keep producing estimates in the difficult regions close to the source and behind the obstacle, where pointing straight at the source is no longer the useful direction, and those hard cells inflate their average. RANSAC’s low error comes partly from declining to estimate the hard cells at all.

When all methods are evaluated on the common set of 284 cells that every method estimates, shown in Figure 3.8(c), this coverage advantage is removed and the quality of the estimated direction itself is isolated. On this common set the plane-fitting methods fall within a few degrees of one another, at 8.6° ,

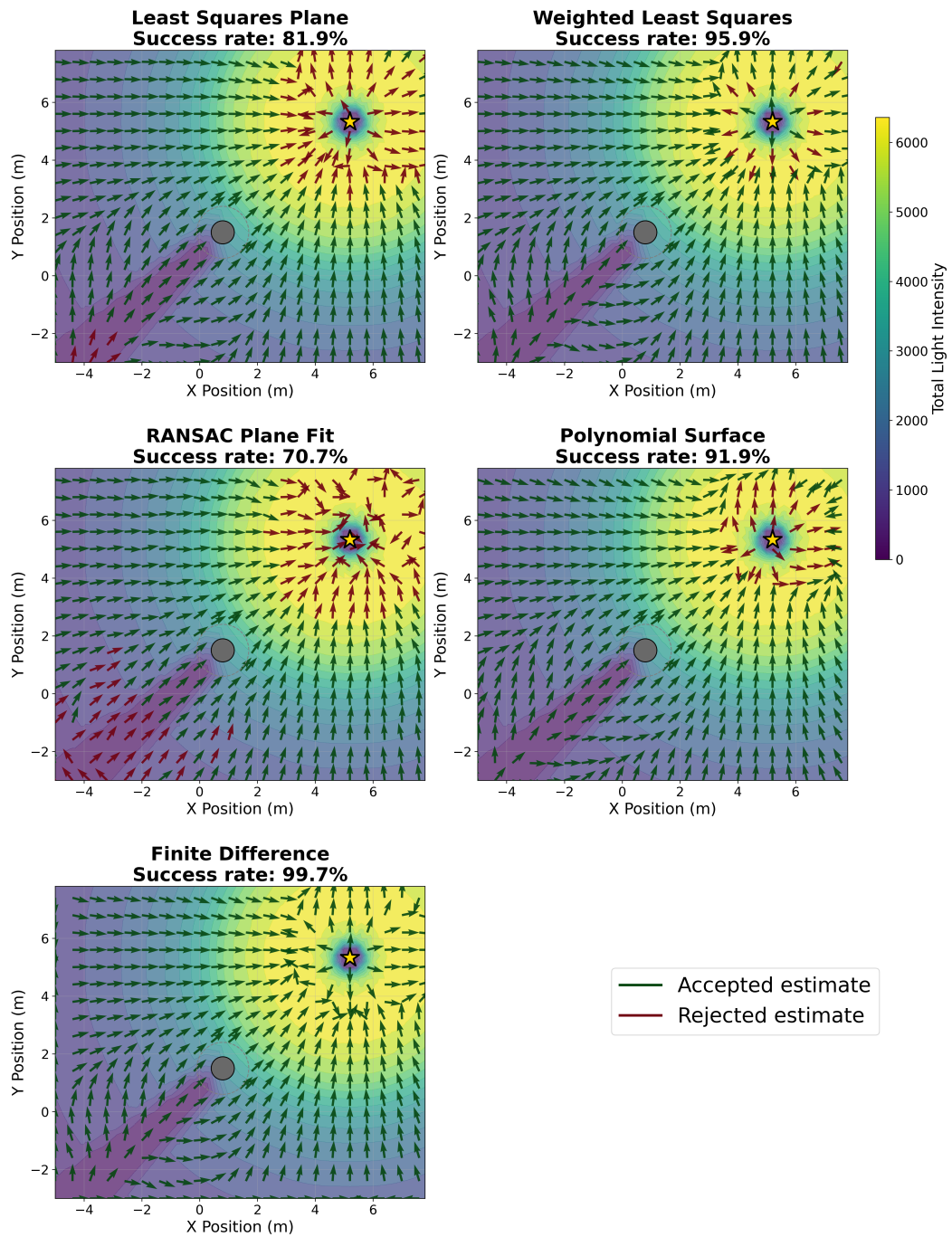


Figure 3.9: Estimated gradient directions across the field for each method, coloured by angular error against the straight-to-source direction. The grey disc is the obstacle and the dashed circle the clearance the drone keeps. The large errors cluster under the source and behind the obstacle, where the field bends and pointing straight at the source is no longer the useful direction.

9.5°, 5.9°, and 6.3° for ordinary least squares, weighted least squares, RANSAC, and the polynomial surface, while finite differences remain noticeably noisier at 15.3°. Most of the apparent accuracy gap seen on each method’s own coverage is therefore caused by which cells are estimated rather than by the intrinsic quality of the estimated direction.

Selected Method

After our evaluation, we select the weighted least squares method as our final implementation for our controller. Although several methods achieve similar directional accuracy, weighted least squares provides the best balance between the three evaluation criteria. It offers high enough coverage, low computational cost, deterministic execution time, and robustness to variations in measurement density and overall light level.

The computational cost follows from how each method uses the n cells in the local neighbourhood, which a fixed search radius keeps to a few tens of cells. Ordinary and weighted least squares, the polynomial surface, and the finite difference each build a small linear system of fixed size from the n cells and solve it once, so each costs $O(n)$ per estimate. The polynomial surface carries the largest constant factor, as it solves for six coefficients rather than three. RANSAC instead repeats a minimal-sample plane fit over up to fifty random subsets and scores all n cells each time, giving $O(kn)$ for k iterations. This is the highest cost of the five, and its iteration count makes its execution time non-deterministic.

Finite differences provide slightly higher coverage but at the cost of substantially noisier estimates. RANSAC produces accurate estimates where it succeeds but has the lowest coverage and, as its complexity shows, the highest computational cost.

Weighted least squares therefore provides the most suitable compromise for real-time deployment on the drone’s microcontroller and is used throughout the experiments in this work.

3.6 Multi-Beacon Traversal

In the real-world application, when multiple targets are present, the controller must be able to distinguish different beacons. We make the beacons distinguishable by modulating each one at a distinct frequency. Frequency analysis of a single photodiode signal then produces a separate peak for each beacon, allowing the photodiode array to sense multiple beacons at the same time. The traversal remains sequential. The controller selects one beacon as the current target, flies toward it, and selects the next beacon once the target is reached.

In a real-world environment, a photodiode reports a single mixed light-intensity value at each timestamp, where signals from all beacons are combined with steady background illumination. By collecting a sequence of intensity measurements over time, the controller can apply a Fourier transform to decompose the mixed signal into its frequency components. Because each beacon is modulated at a distinct frequency, the transformed signal contains an energy peak at each beacon’s modulation frequency, and the height of that peak reflects how strongly the corresponding beacon is observed. Figure 3.10 illustrates this process using

generated data for two beacons modulated at 150 and 200 Hz. Their signals overlap in the time domain but separate into distinct peaks in the frequency domain. The Fourier transform output is then passed to a peak-extraction step, which recovers the signal strength at each target frequency, whose implementation details are described in Chapter 5.

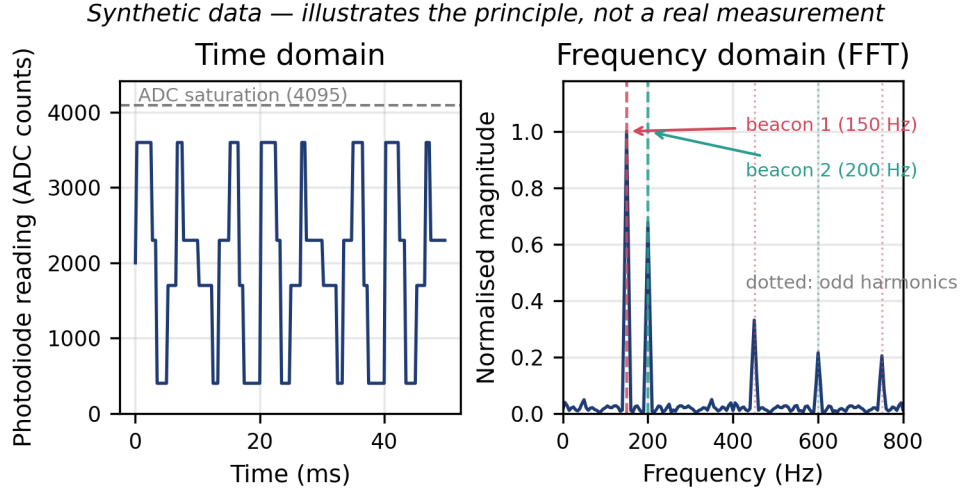


Figure 3.10: **Separating two beacons by modulation frequency.** This is generated data, shown only to illustrate the principle. Two lights are switched on and off as square waves at 150 and 200 Hz, and one photodiode sees their sum as an ADC reading between 0 and 4095. Left: a short slice of that reading in time, where the two beacons are not separable. Right: the magnitude of its Fourier transform, where each beacon appears as a peak at its own frequency. The dotted lines mark the odd harmonics that a square wave also produces.

Each photodiode reading in the array is passed through a Fourier transform to produce its own frequency-domain response. The peak heights at the modulation frequencies differ across the array according to each photodiode’s orientation relative to the light beacon. We then use these peak heights, i.e., the frequency magnitudes, as the light-signal inputs defined in Section 2.4.1. The bearing estimator processes these values and produces a bearing angle toward each beacon based on its modulation frequency.

Because real-world environments may contain noisy light sources, such as computer displays or illuminated signs, the modulation frequencies must be chosen carefully. Switching a light fully on and off produces a square wave. Although this modulates the beacon at the desired target frequency, it also introduces odd harmonics [23]. These odd harmonics can also be seen in Figure 3.10. The frequency analysis separates the fundamental target frequency from its harmonics because they fall into different frequency bins. A problem arises only when one beacon’s fundamental frequency coincides with a harmonic of another beacon. In that case, the harmonic adds energy to the wrong beacon’s bin, making the two beacons difficult to distinguish. Therefore, the target modulation frequencies must be selected to avoid harmonic collisions. For two target

beacons, we choose frequencies such as 150 Hz and 200 Hz.

3.7 Controller State Machine

The bearing and gradient methods each provide a direction estimate, but they do not by themselves decide what behaviour the drone should adopt at a given moment. Two situations must be considered before the drone acts. In the first, the drone may not have a trustworthy estimate, either because it has just started searching for the target beacon and the map is still empty, or because the light is blocked by an obstacle. In the second, both estimates are available, and the controller must decide which estimate to rely on while the drone moves. We resolve both situations with a navigation state machine, shown in Figure 3.11, which consists of five states, namely *searching*, *aligning*, *approaching*, *dwelling*, and *recovering*. The following paragraphs describe each state and its purpose.

In the *searching* state, the drone has no trustworthy bearing or gradient estimate, and its purpose is to search for a beacon. Because the photodiode array already covers the full 360-degree horizontal plane, rotating in place would reveal little additional information. The drone therefore flies a bounded figure-eight search path instead. This pattern forces the drone to explore the surrounding area, increasing the chance that it passes through regions where the beacon signal is stronger while recording light measurements into the map. The drone leaves *searching* as soon as the received signal produces a bearing whose magnitude exceeds the trust threshold. During this state, the drone does not use the gradient or bearing estimates to command its motion; it follows only the predefined search path while monitoring the signal strength.

Once the drone has collected enough light information to infer the target direction, either from the bearing estimate or from the gradient estimate, it transitions into the *aligning* state. In this state, the drone turns toward the target while holding position, allowing it to face the beacon before committing to forward motion. It transitions to *approaching* once the drone’s heading falls within the alignment tolerance of the target direction, set to 15° in this work. Appendix A lists this parameter together with the other controller parameters.

In the *approaching* state, the drone flies forward while continuously correcting its heading toward the target. However, odometry drift can gradually shift the drone’s heading away from the estimated target direction. If the heading error later exceeds a wider re-alignment threshold of 30° , the controller returns to the *aligning* state, ensuring that the drone does not continue flying forward while severely misaligned.

The heading used during *aligning* and *approaching* is not always determined by the bearing alone. At the start of the mission, when the map is still sparse, the drone follows the bearing estimate because too few cells have been recorded to support a reliable gradient. Once the gradient satisfies the reliability conditions in Section 3.5, the controller blends the bearing and gradient estimates into a single target heading. In this way, the drone begins using both estimates during normal approach, before the bearing is lost, making the later transition to gradient-only control smoother when the bearing estimate becomes invalid.

The drone must also detect arrival at a beacon. The measured light strength rises as the drone nears the source, so the drone is treated as arrived once this strength exceeds an arrival threshold. This threshold is tuned per flight, with

its value listed in Appendix A. On arrival the drone enters the *dwelling* state, holds position for a fixed dwell time of 1.0 s, and then moves to the next beacon.

The *recovering* state handles the case where the bearing becomes invalid in flight, for example when an obstacle moves between the drone and the beacon. The magnitude of the bearing estimate then drops below the trust threshold, and a controller without memory would have no heading to follow. However, if the gradient estimate meets its reliability conditions, the drone enters *recovering* and continues on the gradient direction alone, climbing the stored light field to move out of the obstacle’s shadow. It returns to *approaching* as soon as the bearing is valid again, and falls back to *searching* only if the gradient estimate also becomes unreliable.

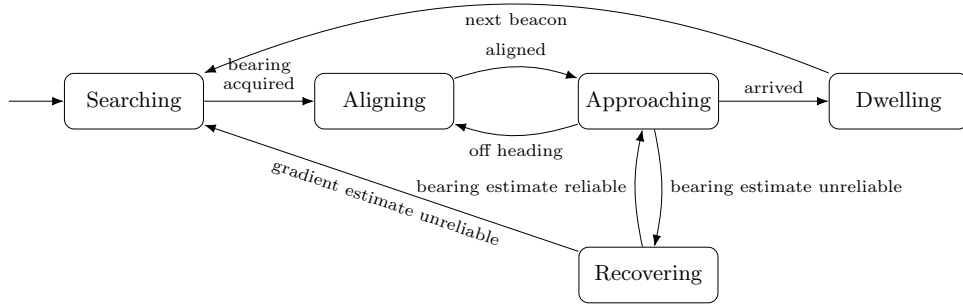


Figure 3.11: **Navigation state machine.** The drone searches for a beacon, aligns its heading, then approaches while correcting; on arrival it dwells briefly before moving to the next beacon. It enters recovering to follow the gradient alone when the bearing becomes invalid, and returns to searching if the gradient estimate also becomes unreliable.

3.8 Control Output

The controller’s primary output is a single target heading, obtained by merging the bearing direction ψ_B and the gradient direction ψ_G . Both are angles, so the controller combines them with a weighted circular mean [21],

$$\psi = \text{atan2}(w_B \sin \psi_B + w_G \sin \psi_G, w_B \cos \psi_B + w_G \cos \psi_G), \quad (3.5)$$

A plain numerical average is unsuitable here because angles wrap at 360° . Averaging 359° and 1° gives 180° , the opposite of the correct direction, whereas the circular mean returns 0° .

The weights w_B and w_G follow the validity of each input, as decided by the state machine. When both are valid, the controller sets $w_B = w_G$, so the heading is their bisector. When only one is valid, its weight is one and the other zero, and the controller follows that input alone. When neither is valid, there is no direction to command, and the drone falls back to the *searching* state rather than holding a fixed heading.

The controller also needs to set a forward speed once it has found a reliable target direction. The drone flies forward only when it is aligned with the target,

so its forward speed is set to zero while it turns during the *aligning* state. Holding the speed at zero during large heading corrections prevents the drone from drifting sideways. Once the heading falls within the alignment tolerance, the controller enters the *approaching* state and commands a fixed forward speed toward the target. The heading and forward speed are sent to the platform’s flight controller as setpoints, while the altitude is held at a fixed height throughout the experiment. The navigation controller therefore operates only in the horizontal plane. Because it communicates with the platform only through heading and speed commands, the same navigation logic is independent of altitude in our experiments. The same implementation runs in both the Webots simulation and the real Crazyflie platform.

Chapter 4

PCB Hardware Design

A central question of this thesis is whether a weight-, size-, and power-constrained micro-drone can carry the light sensing it needs on board. Thus, in this chapter we detail our on-board visible-light sensing hardware design to meet these constraints. The bearing and gradient algorithms of Chapter 3 both read from an array of photodiodes, so the platform needs a board that holds those photodiodes, converts the raw electrical voltage into light-intensity signals, and transfers the signal to the drone’s board for further processing. The board must do all of this within the mass and size budget set in Section 2.2. This chapter develops that board. We first examine an earlier Teensy-based prototype, which established the sensing principle but remained far too large and heavy to fly. We then derive the requirements the flight board must meet. Last, we present our board design with a customised circuit and physical layout that satisfy these requirements.

4.1 The Original Prototype

Meeting the size, mass, and power limits of a micro-drone requires hardware designed specifically for that platform. Our earlier prototype, built around a Teensy 4.1 microcontroller, did not meet these constraints because it was developed as a rapid algorithm testbed rather than as hardware for drone integration. We used it to develop and evaluate parts of the bearing and gradient algorithms, and to confirm the sensing principle that a ring of photodiodes can recover a bearing toward a light source. However, the prototype was never intended to be carried by a micro-drone. It consisted of three separate modules, including the Teensy that ran the algorithms, a circular carrier board that distributed power and provided sixteen connectors, and sixteen small boards that each held one photodiode and could be plugged into the carrier. This modular construction made the prototype convenient to assemble and modify, but its size, mass, and power draw were far beyond the budget of a micro-drone, while its computational capacity also exceeded what the navigation algorithms require.

The prototype’s size and mass were dominated by the photodiode modules it used. Each photodiode was a self-contained Texas Instruments OPT101 module, which integrates a photodiode and its transimpedance amplifier in a single

package. Such integrated modules are convenient to wire, but their packaging occupies far more board area than a simple photodiode would. The carrier accommodated sixteen of these modules, twice the number later required by the flight board. The Teensy 4.1, meanwhile, is a general-purpose microcontroller whose processing capacity far exceeds the requirements of our experiment, which only needs to sample and process eight light signals. The prototype board itself was roughly 90 millimetres in diameter, more than twice the diameter of the drone board. The assembled prototype weighed about 44 grams, more than three times the 13 gram budget available for the sensing hardware. Figure 4.1 shows a partially assembled prototype.



Figure 4.1: **The original Teensy 4.1-based prototype board. Only one of the OPT101 photodiode boards is fitted here, though the carrier has room for sixteen.**

4.2 Design Requirements

The redesigned board must retain the sensing principle demonstrated by the prototype while overcoming its shortcomings and fitting within the budget of the CrazyFlie 2.1, the micro-drone targeted in this work and introduced in Section 2.2. As discussed in Section 2.2, the CrazyFlie 2.1 can lift only roughly 15 grams of additional payload. Since the Flow Deck V2 odometry deck already occupies about 2 grams of this budget, only 13 grams remain for the sensing board. The board’s size must also fit within the drone’s 44 millimetre frame and draw only the current that its small battery can supply. Table 4.1 summarises these requirements together with the final hardware profile we managed to achieve.

Three design decisions follow directly from this budget. First, the board carries eight photodiodes rather than the prototype’s sixteen, the number Huang

Table 4.1: **Design requirements for the sensing board, derived from the CrazyFlie 2.1 platform introduced in Section 2.2, together with the figures the finished board achieves.**

Metric	Requirement	Old Prototype	Achieved
Mass	≤ 13 g	≈ 44 g	≈ 6 g
Diagonal footprint	≤ 44 mm	≈ 90 mm	≈ 42 mm
On-board processor	none	Teensy 4.1 board	none, offloaded to the CrazyFlie
Photodiodes	8	16	8

et al. found sufficient to recover a reliable bearing [15], which halves both the sensor count and the board area they occupy. Second, it replaces the bulky integrated modules of the prototype with discrete surface-mount photodiode and amplifier components, which provide the same function in far less area and mass than the OPT101 modules. Third, it carries no microcontroller of its own. Rather than duplicating the prototype’s Teensy, the board offloads all computation to the CrazyFlie’s existing processor, so it only has to sense and digitise the light and spends neither mass nor power on an additional processor.

A final set of considerations follows from building on the CrazyFlie. The board attaches through the CrazyFlie’s expansion deck connector, which fixes the geometry of where the board’s mounting headers must sit. The same connector supplies the board’s power and ground, and offers a choice of digital interfaces, namely I2C, UART, and the Serial Peripheral Interface, or SPI. We return the converter readings over SPI, so the connector alone carries power, ground, and data, and the board needs no wiring of its own.

4.3 Circuit Design

The circuit must convert the light received by each photodiode into a digital value that the CrazyFlie can read, and it must do so consistently across all eight photodiodes so that their readings can be compared. Figure 4.2 shows the resulting signal chain. Each photodiode output is first passed through a transimpedance amplifier and an optional filter. A shared analogue-to-digital converter then samples all eight photodiode readings and sends the measurements to the CrazyFlie. For these measurements to be meaningful, the converter also requires a precise voltage reference, and the full signal chain must be powered by a regulated supply. We describe each stage in turn, together with the component chosen for it, as summarised in Table 4.2.

The chain begins at the photodiode, which converts incoming light into a small electrical current. Four requirements must be satisfied for our navigation task. First, the sensor must respond to the visible light spectrum rather than to other wavelengths, such as infrared. Second, it must respond fast enough to capture the beacon’s few-hundred-hertz on-off modulation. Third, it must provide a directional response, measuring a beacon more strongly when it faces the source than when the source lies off to the side, since the bearing is recovered from signal differences across sensors. Fourth, it must be small enough for eight sensors to fit within the board’s size budget. To meet these requirements, we choose the Vishay VEMD4200, a PIN photodiode in a 2×1.25 millimetre

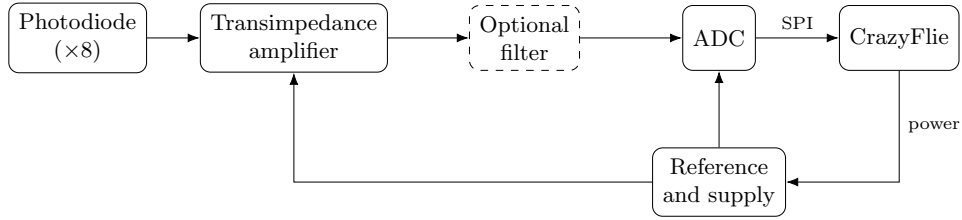


Figure 4.2: **Signal chain on the redesigned board. Each of the eight photodiodes feeds a transimpedance amplifier and an optional filter; the analogue-to-digital converter then samples all eight channels and sends them to the CrazyFlie over SPI. A precision reference and the supply, drawn from the CrazyFlie, feed both the amplifiers and the converter.**

0805 surface-mount package. Compared with phototransistors, another common light-sensing option, PIN photodiodes provide faster response and a more linear light-to-current relationship, whereas phototransistors often trade sensing speed and linearity for higher gain. The VEMD4200 has a spectral response matched to visible light, so it responds strongly to ordinary lamps. Its half-sensitivity angle of $\pm 55^\circ$ keeps each sensor directional while still providing a sufficiently wide field of view. As a result, neighbouring sensors observe different signal strengths as the beacon direction changes, preserving the contrast needed for bearing estimation.

The current a Vishay VEMD4200 produces, only a few microamperes, is far too small to measure directly. Each channel therefore converts that current to a voltage and scales it into a useful range, the standard transimpedance configuration that Graeme sets out in detail [11]. Two properties of the amplifier matter for the readings to stay comparable. First, it must add little noise of its own, so it does not obscure the faint photodiode signal. Second, its output must reach across the full input range of the converter, so the converter’s resolution is used over its whole span rather than a fraction of it. We use the Texas Instruments OPA4350, which meets both. It is a low-noise operational amplifier, so it contributes very little noise. It also has a rail-to-rail output: its output can swing almost the whole way between its two supply voltages, and so cover the converter’s full range. A general-purpose amplifier would add more noise to the faint signal and leave part of the converter’s range unused, meeting neither requirement. The OPA4350 also carries four amplifiers in one package, so two packages serve all eight channels in place of eight separate devices.

The eight amplifier outputs are then digitised and passed to the CrazyFlie. We use a single 12-bit analogue-to-digital converter, the Texas Instruments ADS7953, which scans all eight channels in sequence. This allows the board to collect one frame of eight photodiode readings and send it to the main drone board. Using one shared converter saves board area and power compared with placing one converter on each channel. It also avoids routing eight analogue signals through the drone connector, where they would be more exposed to electrical noise from the platform, including motor noise. The converter returns the readings over the SPI bus, and on the drone it repeats the scan at 500 Hz, fast enough to capture the beacons’ on-off modulation, as Chapter 5 details.

Two further parts keep these readings stable. The converter measures each voltage against a reference, so any drift in that reference would make the same light yield different readings over time and between channels. We therefore supply a dedicated precision voltage reference, the Texas Instruments REF5025A, rather than relying on the supply rail, which varies with load and temperature. For the same reason we clean the supply itself: the rail the board draws from the CrazyFlie carries switching noise from the rest of the drone, so we condition it with a low-dropout regulator, the Texas Instruments TPS73133, a linear regulator that adds no switching noise of its own.

Finally, different deployment environments introduce different noise profiles. For example, outdoor settings may contain less high-frequency artificial lighting noise but stronger low-frequency variations from sunlight and shadows. Fixing one filter choice in the hardware would therefore tie the board to a specific environment. To keep the design adaptable, each photodiode channel includes an optional first-order filter between the amplifier and the converter. This hardware filter can be configured for the target environment by changing the resistor and capacitor values. A solder bridge and unpopulated component pads allow either a low-pass or high-pass filter to be installed as required by the target environment. As shown in Chapter 5, we use a low-pass filter to attenuate power-supply noise from the drone, which otherwise corrupts the light signal.

Table 4.2: **Key components of the redesigned board, by function.**

Function	Part
Photodiode	Vishay VEMD4200
Transimpedance amplifier	Texas Instruments OPA4350
Analogue-to-digital converter	Texas Instruments ADS7953
Voltage reference	Texas Instruments REF5025A
Low-dropout regulator	Texas Instruments TPS73133

4.4 PCB Layout

With the circuit designed, the layout turns it into a physical board the drone can carry. Two concerns shape this layout: even angular coverage and tolerance of the drone’s electrical noise.

For coverage, we place the eight photodiodes at equal 45° intervals around the board so that, together, they sample the full 360° horizontal plane evenly. This spacing is important because the bearing algorithm compares the photodiode readings against one another to infer the light direction, so any large gap in coverage would leave some directions weakly sensed by the array and make the bearing estimate less reliable. The board outline extends each photodiode outward along its assigned angle, while the amplifiers and converter are grouped near the centre, as shown in Figure 4.3.

For noise tolerance, the layout must withstand the electrically noisy environment of a flying drone, where the motors draw large, rapidly switching currents that radiate electromagnetic interference into the faint photodiode signals. To reduce this coupling, the board carries a ground plane beneath the signal traces. The ground plane both shields the traces and gives the return currents a

low-impedance path back to the supply, following the standard electromagnetic-compatibility practice that Ott sets out [24].

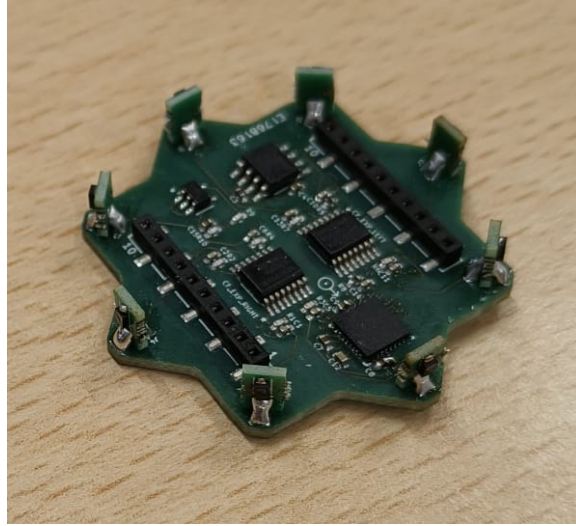


Figure 4.3: **The redesigned CrazyFlie expansion board. Eight photodiodes sit at the points of the board at 45° intervals, while the amplifiers and the analogue-to-digital converter are grouped at the centre. The two headers connect the board to the CrazyFlie.**

The final board weighs approximately 6 grams and measures roughly 42 millimetres across, comfortably within the 13 gram mass limit and the 44 millimetre footprint it is designed to meet, as Section 4.2 records. Before integrating the board with the drone, we verify it on its own. We connect the assembled board to a standalone Teensy microcontroller, used here only as a bench instrument rather than as the prototype’s flight processor, and confirm that each of the eight channels responds to light. With this basic check passed, we can attribute any fault observed later in flight to how the drone interacts with its environment or to the navigation algorithm, rather than to the board itself.

Chapter 5

Real-World Implementation

This chapter develops the signal processing pipeline that recovers each beacon from the photodiode signal on the real platform. Unlike in simulation, where the light intensity can be read directly, each photodiode here delivers a single noisy voltage in which the beacon is buried. The pipeline recovers the beacon and reports, alongside each reading, a signal-to-noise ratio that quantifies how far the reading can be trusted. We first describe the pipeline stage by stage, from the analogue anti-aliasing filter through the frequency analysis to the per-channel smoothing. We then validate it in flight, isolating the contribution of each stage to the reported signal-to-noise ratio. Finally, we measure how ambient light affects the pipeline, and show that it lowers the reported ratio.

5.1 From Intensity to SNR

In simulation the per-sensor light signal requires no recovery. As Section 2.5 describes, the drone reads S_n directly, and it represents the light intensity alone. However, the real platform cannot perform navigation simply based on the light intensity value. On the drone each photodiode reports a single voltage in which the target beacon, the ambient room lighting, sunlight through a window, and reflections off the walls and floor are summed together. This raw voltage is not meaningless, but on its own it is an unreliable basis for navigation, because it does not separate the target beacon’s light from the background noise.

To isolate each beacon from background noise, we modulate each target beacon at a distinct frequency and recover its signal in the frequency domain. Section 3.6 introduced the principle: a Fourier transform decomposes a photodiode signal into the strength carried at each frequency, and a beacon modulated at a fixed rate appears as a peak at that frequency. On the real platform, we apply the same analysis to each channel. For each photodiode, the controller measures the channel magnitude $\hat{S}_n$. This measurement is obtained by collecting a short window of samples and computing a fast Fourier transform over them. The magnitude of the peak at the beacon’s modulation frequency is then $\hat{S}_n$, which the controller uses. However, electrical noise from onboard devices can still introduce jittering noise, so we further smooth the readings over time to obtain a more stable signal. Finally, the processed signals S_n are passed to the bearing and gradient estimators, which compute the direction toward the

light source.

The height of this peak, $\hat{S}_n$, does not on its own state how reliable the reading is. A faint beacon seen across a bright, noisy room can raise a peak height that does not stand out from the fluctuations around it, whereas a strong beacon stands well clear out of them. We therefore pair every reading with a signal-to-noise ratio, or SNR. The SNR is the height of the beacon's peak divided by the noise floor. The noise floor is the magnitude at the frequencies other than the target beacon's. An SNR of one means the peak is no taller than the noise floor, so the beacon cannot be told apart from the noise and the reading cannot be trusted. A large SNR means the beacon dominates its frequency bin and the reading is reliable. Figure 5.1 shows an example, in which a single beacon stands as a peak well above the noise floor.

The controller uses the SNR in two ways. First, each SNR value is tied to a beacon: because the value comes from a peak in a specific frequency bin, we attribute it to the beacon whose modulation frequency matches that bin. Second, it gates out any reading whose SNR falls below a threshold. Chapter 6 reports flights that reach several beacons in succession, where the drone is able to identify different target beacons by their own modulation frequency.

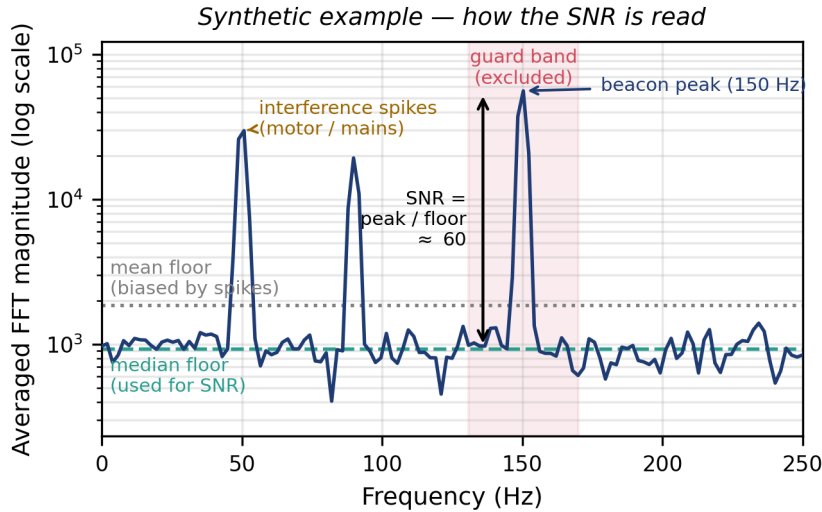


Figure 5.1: **How the signal-to-noise ratio is read from a single channel's spectrum. This is generated data, shown only to illustrate the principle. A beacon modulated at 150 Hz produces a peak well above the broadband noise floor, while motor and mains interference add narrow spikes elsewhere. The floor is estimated with the median of the bins outside a guard band around the beacon, rather than the mean, because a few tall spikes pull the mean up but barely move the median.**

Modulation does not make the signal from the beacon immune to the effects of the surrounding environment. A strong enough background light leaves little of the photodiode's sensing range for the beacon's modulation, so the peak can saturate, an effect we return to in Section 5.3. What modulation does help with

is the ability to attribute a measured signal to a specific beacon and to reject readings whose SNR is too low to act on.

The frequency analysis that recovers the beacon is only useful if noise does not contaminate its frequency bin, and flight introduces a great deal of noise. First, the spinning motors couple mechanical vibration into the board. Second, the motors also draw current that the analogue-to-digital converter for the photodiodes picks up as electrical interference. Third, light reflecting off nearby surfaces returns delayed copies of the beacon by indirect paths. Although most of this noise lies well above the beacons' modulation band, it can fold back, or alias, into the lower frequencies where the beacons lie, and corrupt the peaks the analysis depends on [23]. Figure 5.2 shows the complete pipeline that suppresses this noise, from the sampled voltage to the reported signal S_n and its SNR. It is divided between the analogue domain, before sampling, and the digital domain after it, and the following paragraphs describe each stage in turn.

To prevent high-frequency contributions induced by noise from aliasing into the sampled signal, we apply an initial analogue filtering stage. The photodiode board of Chapter 4 provides optional-filter pads, onto which we populate a first-order low-pass filter with a cut-off of $f_c = 500$ Hz, so that high-frequency content is attenuated before it can alias. Each photodiode signal is also sampled at $f_s = 500$ Hz. According to the sampling theorem, the highest frequency that can be represented without aliasing is the Nyquist frequency, equal to half the sample rate, which is 250 Hz in our system [23]. Any frequency components above this 250 Hz limit are folded back into the sampled spectrum and may therefore negatively impact our measured signals. The placed hardware filter is not intended to sharply isolate the beacon band. Instead, its purpose is to attenuate the strongest out-of-band energy before it reaches the ADC, thereby reducing the severity of aliasing. Because the filter is only first order and its cut-off frequency lies above the Nyquist limit, attenuation around and above 250 Hz is relatively weak, and some aliased content can still enter the sampled signal. The beacon modulation frequencies, however, lie well below the Nyquist frequency and are therefore sampled faithfully.

However, even with hardware filtering, some noise remains within the 0–250 Hz band that the 500 Hz sampling rate captures. We therefore filter each collected window of samples in software, in five steps. The first two prepare the window for the Fourier transform: subtracting the mean and applying a Hamming window. The remaining three reduce the noise in the result: Welch averaging, a median noise floor, and temporal smoothing.

First, we remove the steady ambient component from the signal. Ambient light introduces a constant offset in the measured intensity, and subtracting this offset leaves only the fluctuating modulated signal for further processing. Second, we multiply the sampled window by a Hamming window, a smooth taper that suppresses the abrupt discontinuity at the window boundaries. Without this taper, the discontinuity would smear energy across the spectrum as spectral leakage, a problem surveyed in detail by Harris [12]. Figure 5.3 shows both preprocessing steps on a sampled window, together with the leakage suppressed by the Hamming window. The transform has a frequency resolution of $\Delta f = f_s/N$. For an $N = 256$ point transform sampled at 500 Hz, this gives about 1.95 Hz per bin, which is fine enough to separate beacons spaced a few hertz apart.

Third, we reduce the variance of the spectrum estimate itself. A single FFT

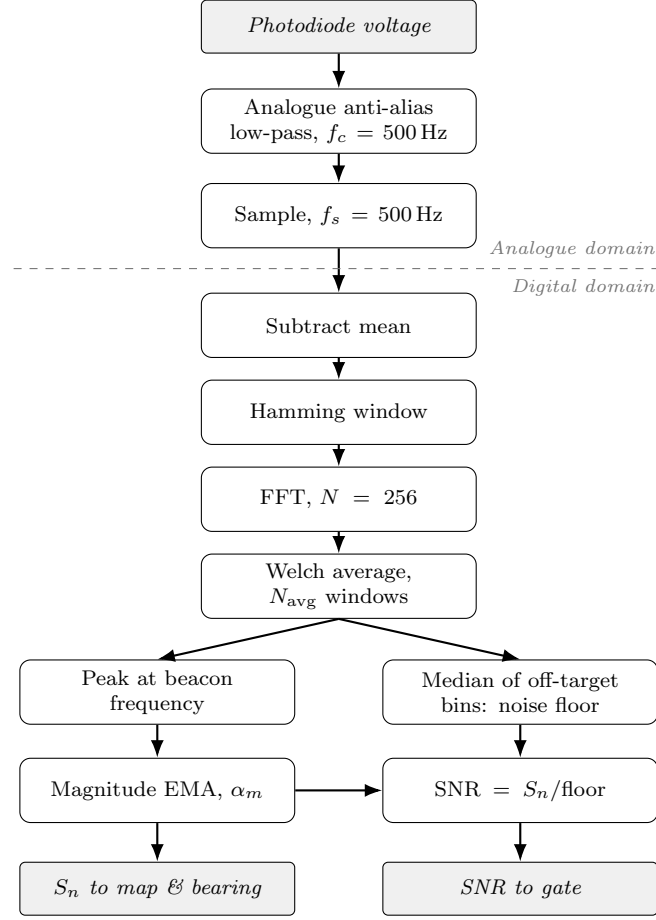


Figure 5.2: **The signal pipeline on the real platform.** Each photodiode channel passes through an analogue anti-aliasing filter before sampling, then through the digital stages that recover the beacon: mean subtraction, a Hamming window, the FFT, and Welch averaging. The averaged spectrum yields both the peak at the beacon frequency, smoothed over time into the signal S_n , and the median noise floor, from which the SNR used by the gate is formed.

gives a noisy estimate of each bin’s magnitude, and that noise raises occasional bins in the floor, which lowers the reported SNR. Rather than read the peak from one FFT, we average the magnitude spectra of several successive windows that overlap by half their length. This averaging of overlapping windows is a standard variance-reduction technique introduced by Welch [29]. For statistically independent windows it would reduce the variance of the estimate by a factor of N_{avg} . The half-window overlap leaves successive windows correlated, however, so the reduction is smaller, and with only $N_{\text{avg}} = 2$ windows it is modest. The cost of averaging is latency: each averaged spectrum spans several windows, so the reported signal responds more slowly to the drone’s motion. We therefore keep N_{avg} small, accepting only marginal smoothing in exchange

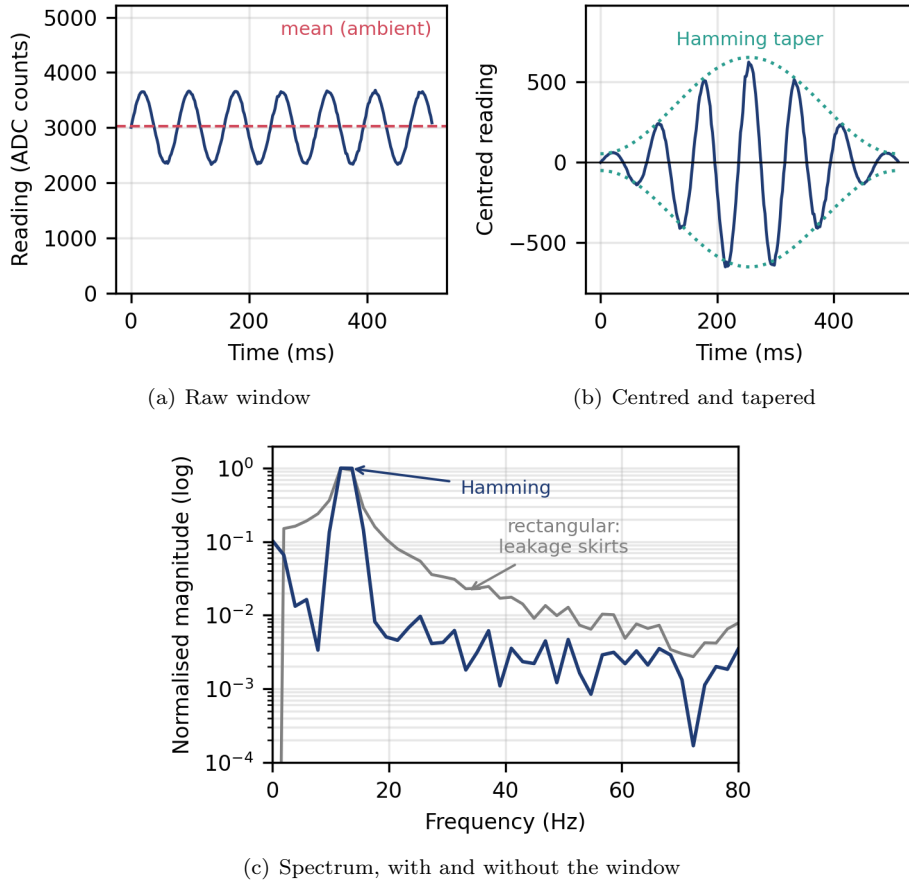


Figure 5.3: The two preprocessing steps before the transform, on generated data. (a) The raw window sits high on the ambient baseline, marked by the dashed mean. (b) Subtracting the mean centres the window on zero, and the Hamming taper smoothly forces both ends to zero. (c) On a logarithmic scale, a rectangular window, equivalent to no taper, spreads the tone's energy into wide leakage skirts, whereas the Hamming window confines it close to the true frequency.

for a responsive reading. Figure 5.4 contrasts a single noisy spectrum with the averaged one, in which the noise floor is visibly smoother while the beacon peak is unchanged.

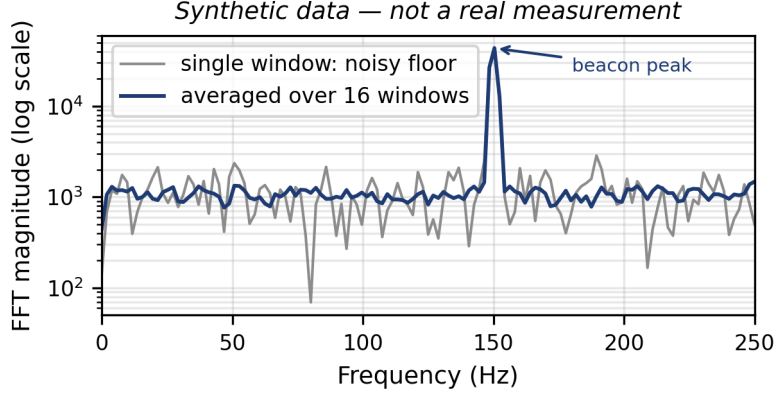


Figure 5.4: **Welch averaging, on generated data. A single window gives a jagged noise floor, so the magnitude of any one bin is an unreliable estimate. Averaging the magnitude spectra of several windows reduces this scatter and leaves a smoother, more representative floor, while the beacon peak is unaffected.**

Fourth, we make the noise floor robust to spikes. The noise floor is the representative magnitude of the spectrum away from the beacon, and it forms the denominator of the SNR. A single loud disturbance, such as a motor harmonic or a pickup at the mains frequency, appears as a tall spurious bin that would inflate a mean-based floor and make every SNR smaller. We therefore estimate the floor with the median of the off-target bins rather than their mean, because the median is barely moved by a few extreme values. We compute it over all spectrum bins except a guard band around the beacon’s frequency and its first four harmonics, which we exclude so that the beacon’s own energy is not counted as noise. The median costs only a sort of the bins rather than a sum, an expense that is negligible at this window size on the drone’s microcontroller. Figure 5.1 shows the difference the median makes: the interference spikes lift a mean-based floor noticeably, while the median floor stays on the broadband level the SNR should be measured against.

Finally, we smooth the per-channel magnitude over time, because the magnitude still fluctuates from one frame to the next. We blend each channel’s freshly measured magnitude $\hat{S}_n$ with its previous smoothed value through a first-order exponential moving average, an infinite-impulse-response (IIR) low-pass filter that weights recent measurements most heavily while retaining a fading memory of older ones [23],

$$S_n[k] = \alpha_m S_n[k - 1] + (1 - \alpha_m) \hat{S}_n[k], \quad (5.1)$$

where $\alpha_m \in [0, 1]$ sets how much of the previous value is retained. We apply the same α_m to all eight channels, so that their relative magnitudes, which encode the bearing to the beacon, are preserved while the jitter that vibration adds is

suppressed. The factor trades responsiveness against smoothness: a larger α_m gives a steadier but slower signal, a smaller one a faster but noisier signal, and we use $\alpha_m = 0.5$. Figure 5.5 shows the effect of this filtering process, where the smoothed signal is far steadier and lags only slightly where the true signal jitters.

This IIR smoothing sits on top of the Welch averaging, and both introduce some lag in the final magnitude S_n , which also delays the rest of the system. However, because we keep the drone’s flight speed slow and average only two windows, the induced delay is small enough not to affect flight performance.

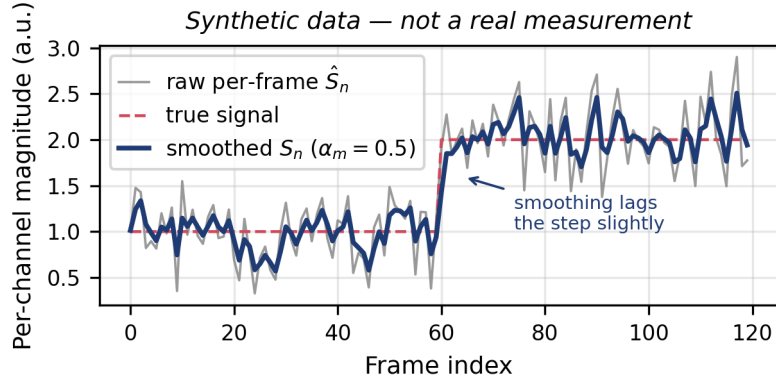


Figure 5.5: **Exponential moving average smoothing, on generated data, as the true signal steps up when the drone nears the beacon.** The raw per-frame magnitude $\hat{S}_n$ scatters widely, whereas the smoothed magnitude S_n is far steadier. The price is a short lag after the step, which a larger α_m would lengthen.

The smoothing improves the magnitude our algorithms from Chapter 3 act on, but it must not distort the SNR for the validity-check used by our controller from Chapter 3. This check decides, frame by frame, whether a reading can be trusted, so its SNR must reflect the noise present at that moment. Dividing the smoothed magnitude by an equally smoothed noise floor would make the SNR lag a sudden rise in noise, and the controller could keep trusting a reading after it has already become unreliable. We therefore reconstruct SNR each frame as the smoothed magnitude S_n divided by the current-frame noise floor. The numerator is then stable while the denominator tracks the noise present now, which keeps the validity-check calibrated to current conditions. Appendix A lists α_m , N_{avg} , the guard band, and the other pipeline parameters.

5.2 Filter Validation

We now test what each stage of the signal processing pipeline of the previous section contributes in flight, and whether each is justified. We hold the drone at a fixed position, 95 cm from a single 150 Hz beacon, so that the true signal is constant and any variation recorded over time is the noise and jitter that the filters are meant to remove. From each held run we take two numbers: the

mean SNR, which reflects the signal level, and the standard deviation of the SNR, which reflects its jitter.

We do tests with one stage of the pipeline at a time, starting with no filtering and then adding the hardware filter, Welch averaging, the median floor, and finally the magnitude smoothing. This allows the contribution of each filtering stage to be analysed separately. We measure each configuration under two conditions. In the first, the motors are off and the drone is held at a fixed altitude. In the second, the drone hovers freely with the motors running, since the spinning motors are the main source of vibration and electrical interference that the filters target. The motors-off readings provide a clean reference, while the motors-on readings show how the SNR drops once the drone begins flying and motor-induced noise is introduced.

One limitation in this experiment is due to the high degree of freedom of the drone. In free hover the drone drifts slightly, so the recorded SNR from the motors-on experiment includes a small deviation from the changing pose rather than from the environment noise alone, although unbiased drift leaves the mean unaffected.

Table 5.1: Overall SNR, the maximum across the eight channels, at the beacon’s 150 Hz modulation frequency as each stage is added to the pipeline, with the drone held 95 cm from the beacon. Each cell is the mean $\pm$ standard deviation over a steady hold, pooled across three repeats.

Configuration	Motors off	Motors on
No filtering	23.9 ± 3.9	13.9 ± 2.1
+ Hardware	17.4 ± 1.3	11.4 ± 2.8
+ Welch averaging	16.3 ± 0.9	15.1 ± 4.6
+ Median floor	74.2 ± 19.2	31.9 ± 7.5
+ Magnitude EMA	76.6 ± 17.8	32.1 ± 5.3

The result in Table 5.1 is dominated by the median noise floor stage. Introducing the hardware filter has a slight negative effect on the overall SNR. Welch averaging raises the level only slightly. The median floor then greatly improves the result, more than doubling the in-flight SNR from about 15 to 32 and more than quadrupling the motors-off SNR from about 16 to 74. The reason lies in the definition of the SNR as the peak divided by the noise floor: the median ignores the occasional spike that would increase the mean-based floor, since a mean is affected more by outliers. Because of this, the median-based floor is much smaller than the mean-based floor, so dividing the same peak by it yields a correspondingly larger SNR.

The magnitude smoothing filter increases the SNR value slightly but steadies the reading, lowering the in-flight standard deviation from 7.5 to 5.3. The standard deviation grows at the median-floor filter only because the SNR values themselves increase. This last stage therefore gives the highest SNR and the lowest standard deviation of any configuration. Even with the full pipeline the in-flight SNR of about 32 sits well below the motors-off ceiling of about 77, which confirms that the spinning motors are the main disturbance the pipeline must absorb.

Because the hardware filter initially has a negative effect on the SNR, we

ran an additional experiment to justify its presence. We tested the full pipeline once more but without the filter in place, reported in Table 5.2. Removing the filter drops the SNR from about 77 to 33 with the motors off, and from 32 to 22 in flight. This shows that the filter is not redundant. It has little effect before the median floor stage is in place, but the two are complementary. The hardware filter removes high-frequency energy, which keeps more noise out of the noise floor; combined with the median floor stage, which lowers the floor value further, this yields the highest SNR. Without the hardware filter, even the median-based noise floor can be overwhelmed by noise from high-frequency sources.

Table 5.2: **Effect of the hardware anti-alias filter within the full software pipeline, with and without the populated 500 Hz filter. Overall SNR, mean $\pm$ standard deviation.**

Configuration	Motors off	Motors on
Full pipeline	76.6 ± 17.8	32.1 ± 5.3
Full pipeline, no hardware filter	32.7 ± 3.6	21.5 ± 2.8

5.3 Effect of Ambient Lighting

The pipeline of the previous sections recovers the beacon’s signal against the background noise, but another factor limits the signal’s resolution: the level of ambient light in the room. Ambient light is not itself a dominant source of noise, unlike the motors, electrical interference, and wall reflections. Instead, it introduces a steady offset in the light intensity that each photodiode reads. A photodiode has a limited output range, so a higher nominal value leaves a smaller portion of the sensing range for the beacon’s modulation to occupy. The received signal then can only swing less between its high and low states, less energy reaches the beacon’s frequency bin, and the SNR drops. In a darkened room this static baseline light intensity value is low, the modulation swings freely, and the SNR rises. Figure 5.6 illustrates the mechanism: a low baseline leaves the full sensor range for the beacon’s swing, whereas a high baseline pushes the reading toward the upper limit and clips the swing, so less of the modulation survives to reach the beacon’s frequency bin.

To validate the influence of ambient light, we fly the same two-beacon traversal twice from the same take-off point, once in the dark and once with the room lights on. With the room lights on, the photodiodes read roughly 360 to 450 lux when facing upward and 120 to 160 lux when facing sideways at flight altitude, compared with 12 to 22 lux and 2 to 22 lux respectively in the dark. Figure 5.7 shows the two flight paths over the light maps built by the drone, and Figure 5.8 shows the overall SNR in each flight, where we show the maximum SNR among the eight channels for each timestamp. The dark flight maintains a higher SNR throughout the experiment, reaching above 30 compared with about 20 when ambient light is present. This confirms the influence of ambient light: a lower baseline intensity leaves more dynamic range for the beacon modulation, allowing more signal energy to appear in the beacon’s frequency bin. Because the drone steers itself, the two flights follow somewhat different paths, and we

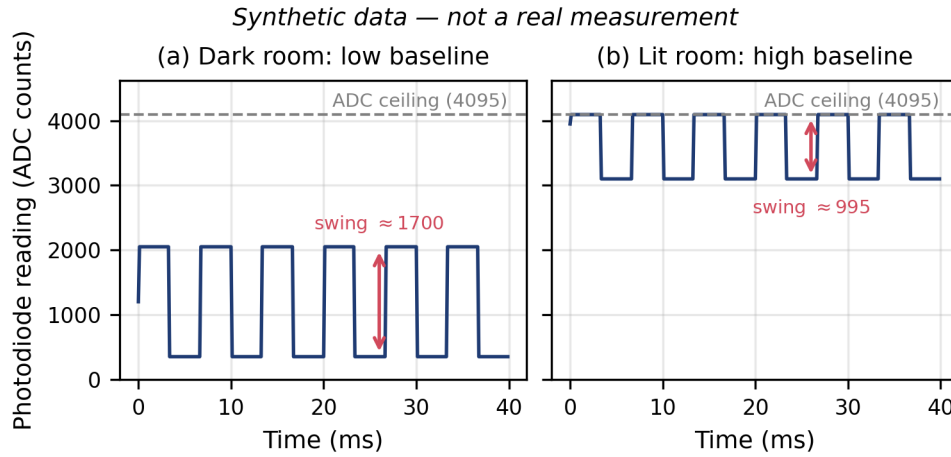
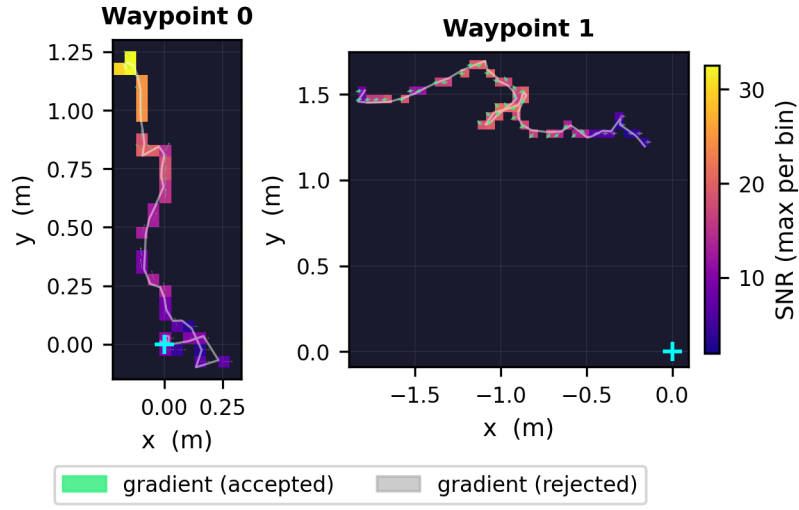
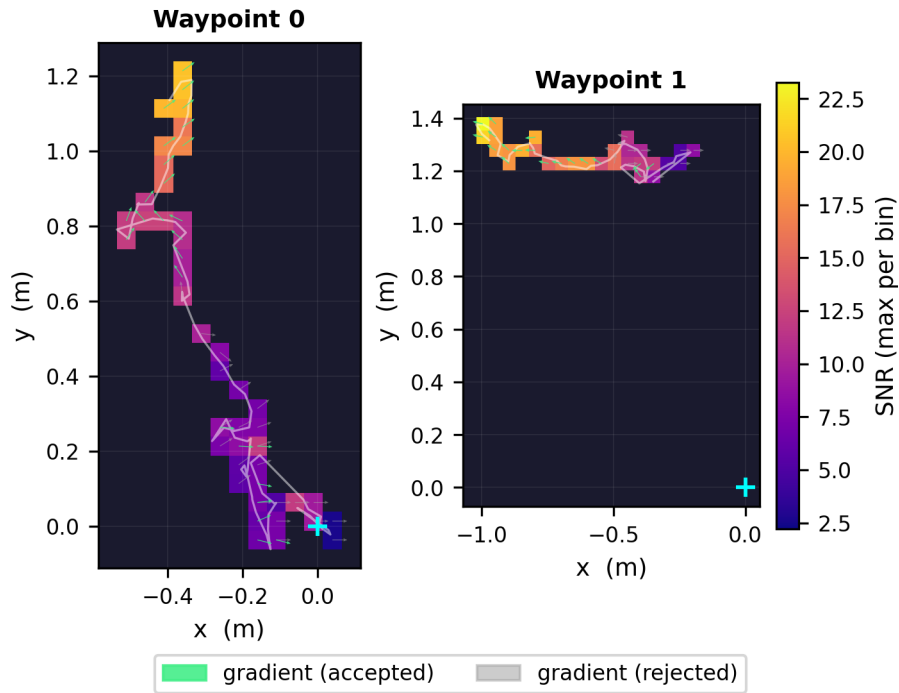


Figure 5.6: **How a higher ambient baseline shrinks the beacon’s modulation, on generated data.** The beacon switches the lamp on and off, which should produce the same swing in both rooms. (a) In the dark the low baseline leaves the whole range free, so the full swing is captured. (b) Under room lighting the high baseline sits close to the sensor’s upper limit, so the on-phase is clipped and the captured swing is much smaller, which lowers the energy at the beacon’s frequency and so the SNR.

fly only a single run per condition. These results therefore show the effect of ambient light rather than provide a perfect comparison between a room with ambient light and one without.

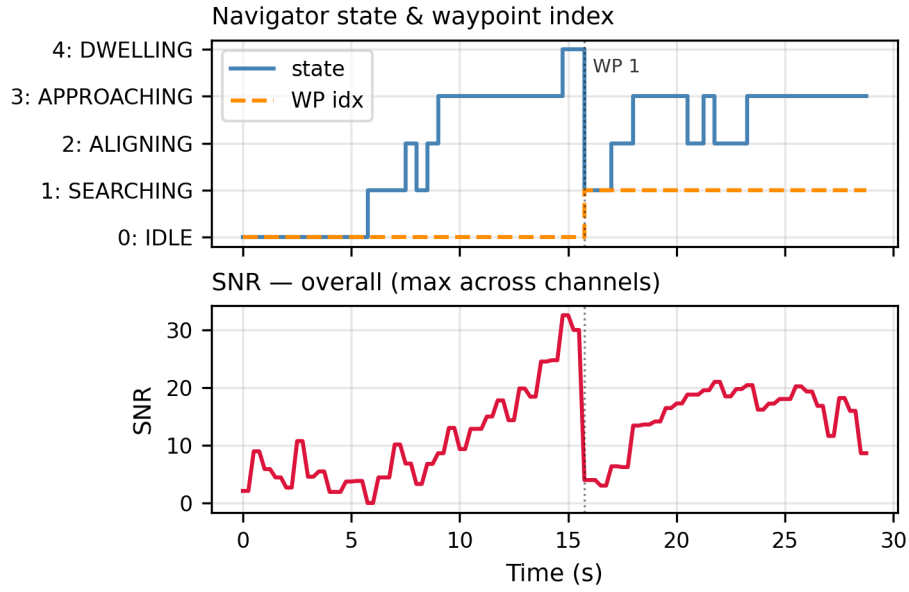


(a) Darkened room

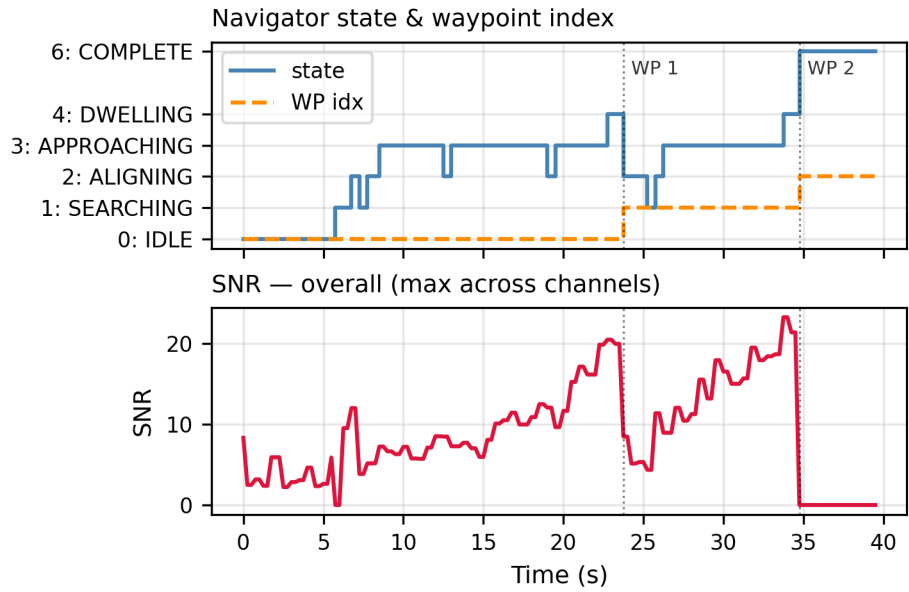


(b) Lit room

Figure 5.7: **Two-beacon traverse**, with the beacons modulating at 150 and 170 Hz, flown in a darkened room and under full room lighting. Each panel shows the flight path to each of the two beacons over the light map the drone built, with cells coloured by the SNR recorded there.



(a) Darkened room



(b) Lit room

Figure 5.8: Overall SNR over the course of the two-beacon traverse, taken as the maximum across the eight channels, in a darkened room and under full room lighting. The marked transitions separate the approach to the first beacon from the approach to the second.

Chapter 6

Multi-Beacon Experiments

One of the central aims of this thesis is to reach several modulated light beacons in succession, which the systems proposed in Huang et al. [15] do not provide. We validate this extended system in two stages, each answering a separate question. The first asks whether the photodiode array can distinguish two beacons by their modulation frequencies. We test this with a stationary bench setup, using a fixed test rig that does not fly and is therefore free from flight-induced motion noise and motor noise. The second asks whether the complete fused controller can guide the drone to multiple beacons in sequence. Testing the sensing stage first in a stationary setting ensures that any failure observed in real flight cannot be attributed to the system’s inability to separate the beacons.

6.1 Experimental Setup

In the first stage, the bench study uses the original Teensy 4.1 prototype from Chapter 4 rather than the final sensing board, because the prototype is better suited for a pure sensing test. It carries its own processor and is not constrained by the size, weight, and power limits of the micro-drone, allowing it to sample faster and log data continuously over longer sessions. Its sixteen photodiodes sample at 1600 Hz, well above the flight board’s 500 Hz, so both beacon frequencies remain well within the sampled band.

To map each beacon across space, we move the Teensy 4.1 board by hand across a regular grid with 0.2 m spacing, covering an area of roughly 0.5 m^2 , and record one measurement at each grid cell. Moving the board by hand removes the drone’s own motion and vibration from the experiment, so the recovered fields reflect the sensing performance alone. This procedure also produces a dense and repeatable static light-intensity map for each beacon, which we use to evaluate how well the sensing pipeline separates and measures the beacon signals.

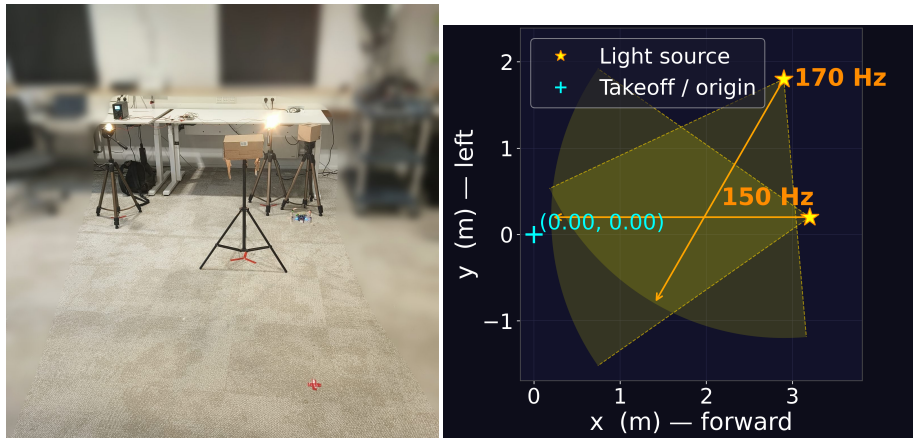
Figure 6.1 shows the arrangement. Two lamps, modulated at 150 and 200 Hz, stand at opposite ends of the experiment area. The room remains dark so only the light from computer monitors at the back of the bench introduces minor noise. The monitor runs at full brightness about a metre away and flickers at 144 Hz, an interferer we return to in Section 6.2.

The second stage flies the full system on the Crazyflie 2.1, carrying the re-



Figure 6.1: **Static two-light bench setup on the original Teensy 4.1 prototype.** Two lamps, marked by the blue and green arrows and modulated at 150 and 200 Hz, stand at opposite ends of the area, with the photodiode ring moved across it to map the signal. The room is otherwise dark.

designed eight-photodiode board described in Chapter 4 and running the fused controller and state machine introduced in Chapter 3. Two beacons, now modulated at 150 and 170 Hz, are placed in an open room. The drone takes off from a fixed origin and is given a mission that visits the light beacons in sequence. Figure 6.2 shows the layout: the two lamps are placed a few metres ahead of the take-off point, with one roughly straight ahead and the other to the left, so reaching them in sequence requires the drone to make a substantial heading change between targets. The figure provides an indicative reference for the room arrangement rather than an exact measurement, so its distances should not be interpreted as ground-truth values. We evaluate the system with two missions, one traversing two target beacons and the other traversing three, to test the platform under both short and long navigation sequences.



(a) The room, with the two beacons on tripods and the drone already airborne above its take-off point
(b) Top-down geometry, beacons at 150 and 170 Hz and the take-off origin

Figure 6.2: **Real-world flight setup for the multi-beacon traverses on the CrazyFlie 2.1.** The two beacons, modulated at 150 and 170 Hz, stand a few metres from the take-off origin, one roughly ahead and the other to the left. The top-down geometry is a reference to the layout only, not an exact, measured match to the real room. In the photograph the drone is already airborne, hovering just above its take-off point between the origin and the right-hand beacon. The photograph also still shows the obstacle box, which is not present during these open-space traverses; the same room with the obstacle in place is used only for the experiments of Chapter 7.

6.2 Separating Beacons by Frequency

The multi-beacon scheme mentioned in Chapter 3 depends on the sensors on the Teensy prototype measuring every beacon. A single frequency analysis of a photodiode signal can then detect multiple beacons in parallel rather than one at a time, and the controller can then select the current target beacon by reading the peak magnitude at that beacon's modulation frequency. We

test this directly on the static bench. At each position, we record one sample window and compute the SNR at both 150 and 200 Hz from the same data. Figures 6.3(a) and 6.3(b) visualise each beacon’s SNR across the area. Switching the controller’s attention from one beacon to another is then only a matter of choosing which frequency magnitude to use. This choice is made in software and adds no extra sensing cost, so we do not test it separately.

The result confirms that the two beacons are recovered independently and at the same time. As the top row of Figure 6.3 shows, each beacon’s SNR field rises to a clear peak at its own lamp’s position. Because the two lamps stand at opposite ends of the area, the two SNR maps peak in opposite corners, which confirms that each reading belongs to its own beacon.

The bottom row of Figure 6.3 shows why the controller should navigate using SNR rather than raw signal magnitude. The magnitude fields are much noisier than the SNR fields above them for two reasons. First, they include gain differences across the sixteen photodiodes, since each sensor produces a slightly different voltage response even under the same light exposure. Second, they include stray 144 Hz light from the monitors, which lies only 6 Hz away from the 150 Hz beacon. This is close enough for some of its energy to leak into the 150 Hz beacon bin, creating a spurious response that can be mistaken for the target beacon. Both effects explain why we use the frequency-domain SNR rather than raw light intensity: gain differences distort the relative readings across the array, while nearby interference adds signal where no beacon is present.

The SNR fields stay clean despite the same two effects, because the SNR divides a beacon’s peak by the local noise floor, the median magnitude of the spectrum away from the beacon, as defined in Chapter 5. Dividing by this floor removes both the per-sensor gain differences and the monitor interference. The gain differences cancel because a sensor’s gain scales its peak and its noise floor by the same factor, so their ratio is independent of the gain. The monitor interference is suppressed because it lifts the noise floor and the beacon’s peak together, so dividing one by the other cancels most of its effect. This is the same reasoning that motivated the SNR pipeline of Chapter 5, seen here directly in the data. The 150 Hz SNR field still carries a faint trace of the monitor interference, but far less than the magnitude field, and well within what the controller’s SNR gate tolerates.

The frequency separation demonstrated here is not specific to the prototype. The flight board reproduces it with only eight photodiodes at 500 Hz, using the different pair of 150 and 170 Hz, as the traverses of the next section show. The frequencies themselves are a free choice. The only requirement is that no beacon’s modulation frequency coincides with a harmonic of another, since a square-wave beacon also carries energy at its harmonics, as Chapter 3 explains.

6.3 Sequential Multi-Target Traverse

With the sensing confirmed, we turn to the complete system and ask whether the drone visits several beacons in succession in real flight. Sequencing the beacons is the task of the state machine of Chapter 3. It runs its searching, aligning, approaching, and dwelling phases once per beacon, and advances to the next beacon on arrival. Arrival is registered when the overall SNR rises

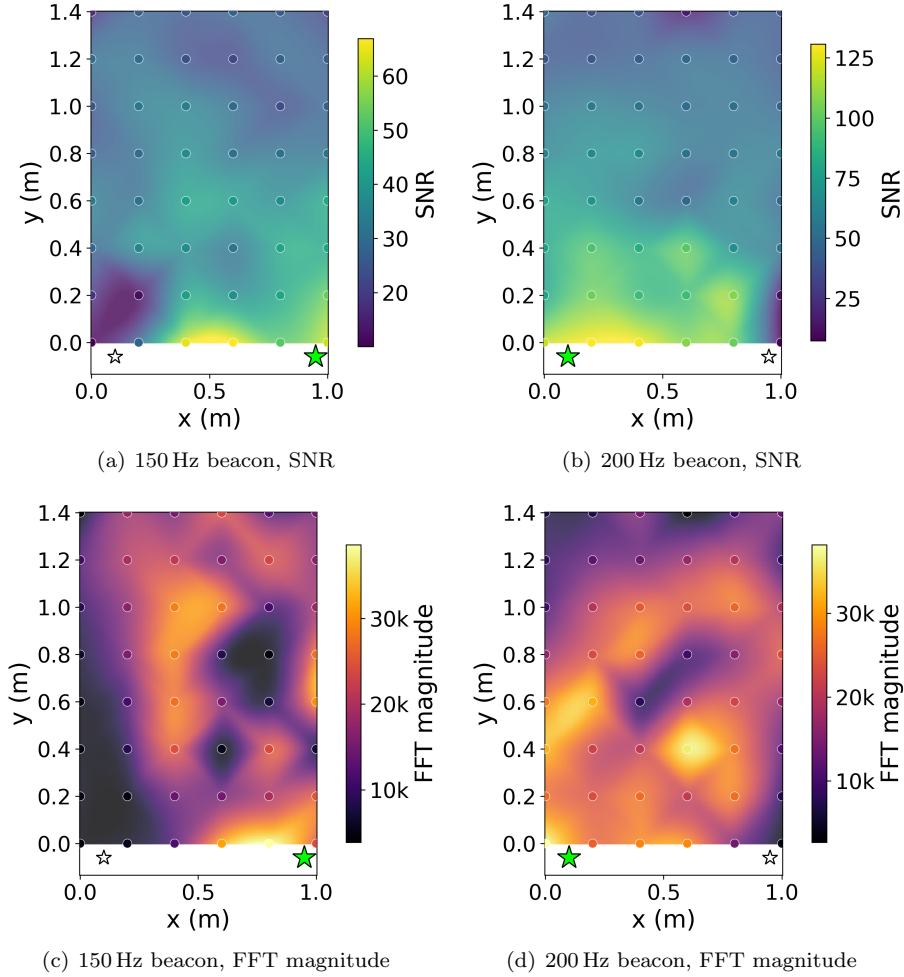
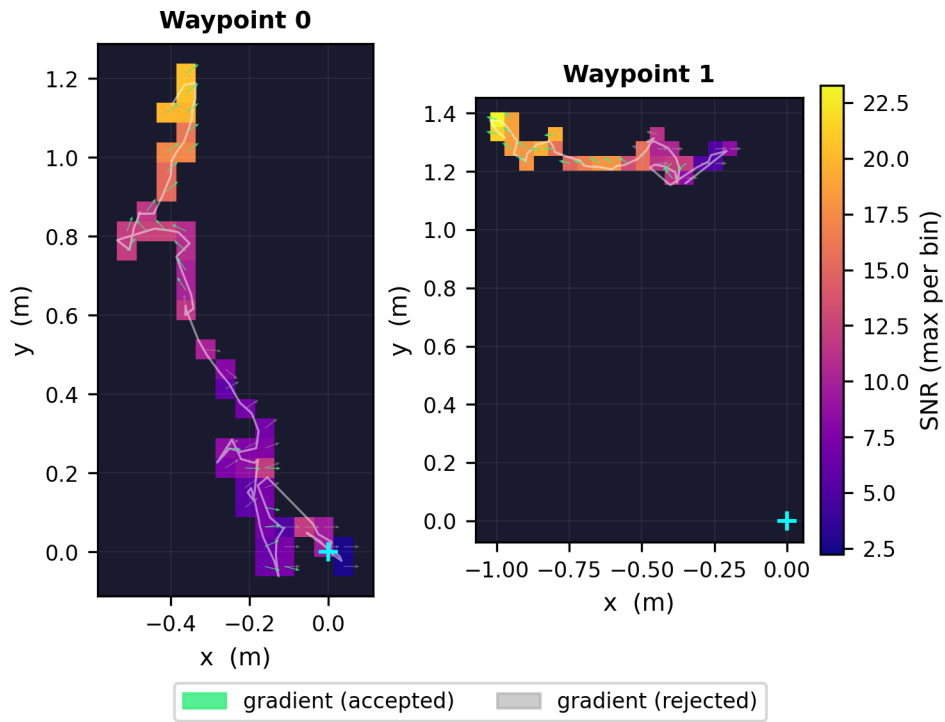


Figure 6.3: Two beacons recovered simultaneously from a single recording at each cell of a swept area on the original Teensy prototype. The columns are the two beacons, the top row is SNR and the bottom row the raw FFT magnitude, each taken as the maximum across the sixteen sensors. In every panel the filled green star marks the beacon at that panel’s frequency, the open star the other beacon, and the dots the measured cells. Each beacon’s SNR field rises to a clear peak at its own light, while the magnitude is noisier, carrying per-sensor gain differences and stray 144 Hz monitor light that the SNR removes by normalising against the local noise floor.

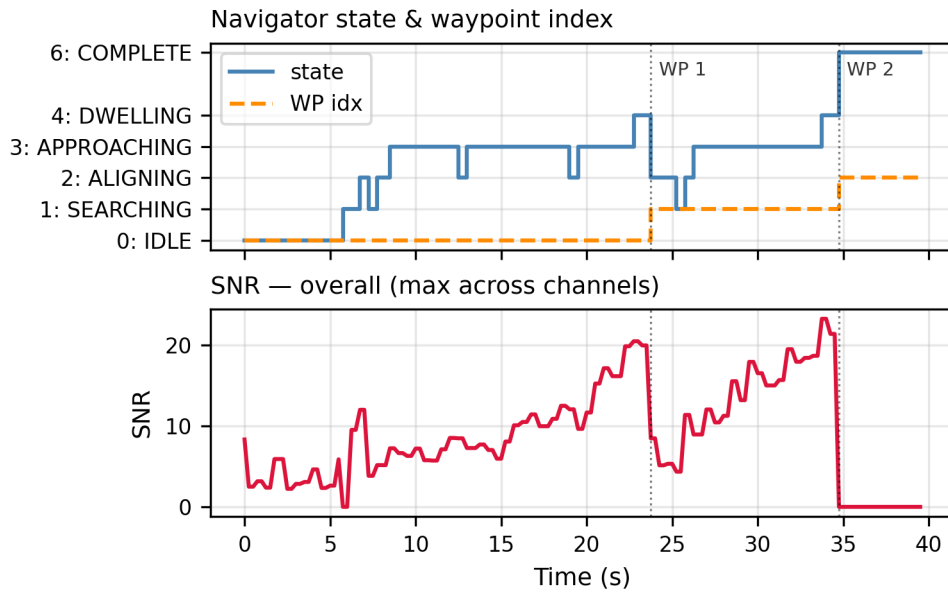
above an arrival threshold, reached only close to a beacon. We set this threshold once for the test environment, tuning it so the arrival condition triggers only near a beacon, and then hold it constant across these flights. We fly the two missions described above on the CrazyFlie, with the beacons modulated at 150 and 170 Hz, and record both the drone’s path and the controller’s internal state throughout. Each mission is a single demonstration flight, so these results show that the traversal works rather than measuring how often it succeeds over many trials, which we leave to future work.

Figure 6.4 shows the two-waypoint flight. The upper panel plots the path, which reaches the first beacon and then the second. Each beacon’s path is drawn over the light map the drone builds for that beacon as it flies, whose cells hold the SNR measured there. The map is rebuilt from scratch for each target, as Chapter 3 describes, so that the position drift accumulated while approaching one beacon is discarded with its map rather than carried into the next. This is why each panel shows only the cells gathered on the way to its own beacon. The lower panel tracks the controller’s state and the active waypoint over time. The state steps through searching, aligning, approaching, and dwelling, and the waypoint index advances by one each time the drone arrives and dwells. The overall SNR, taken as the maximum across the eight channels as in Chapter 5, climbs as the drone closes on each beacon and drops back at each handover, when the drone turns toward a target it is still far from. Together these confirm that the controller sequences cleanly from one beacon to the next.

The three-waypoint flight in Figure 6.5 extends the mission to a longer sequence. With only two physical beacons available, the third waypoint sends the drone back to the first beacon, the 150 Hz lamp, so the mission runs 150 Hz, then 170 Hz, then 150 Hz again. This longer run also exposes an implementation defect in the current state machine. The drone sequences correctly through the first two waypoints, the state machine running its full cycle for each and the waypoint index advancing as before. On the final waypoint it approaches the 150 Hz beacon and the overall SNR climbs to about 39, comfortably above the arrival threshold, so the drone does reach it. The state machine, however, fails to latch the arrival. The state trace stays in the approaching phase even as the SNR passes the threshold, which we attribute to a timing error in the transition logic. The drone therefore keeps flying instead of stopping. It drifts out of the beacon’s narrow light cone, and the SNR collapses into a region where the beacon is no longer in view. The defect lies in the arrival logic, not in the navigation approach: the sequencing is sound for the bulk of the mission, and correcting it is left to future work.

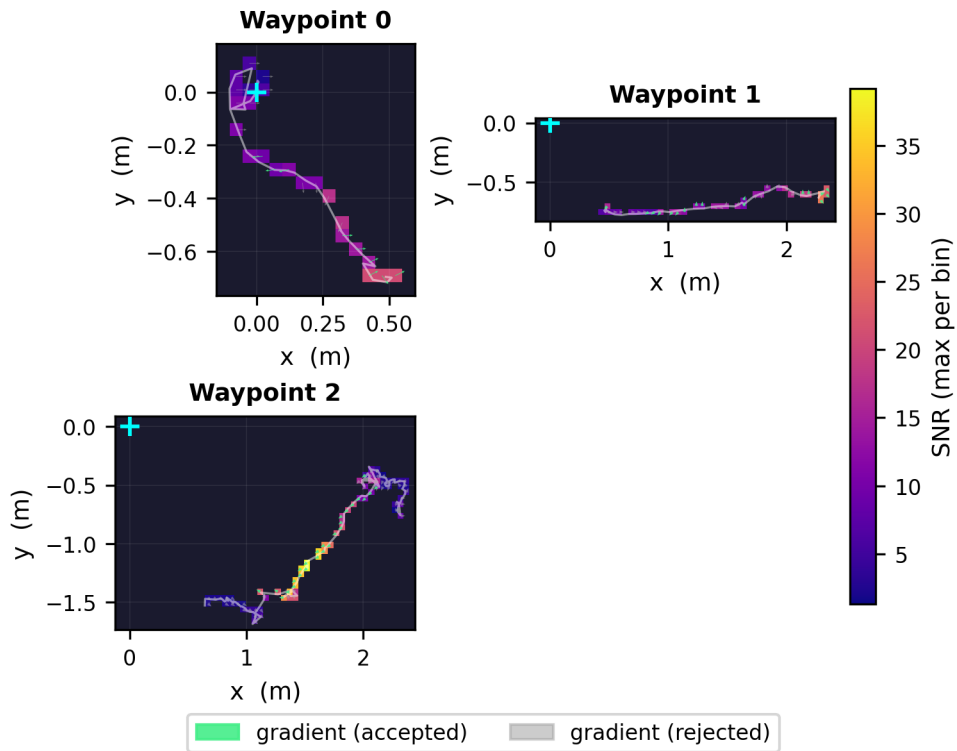


(a) Flight path to each beacon over the per-beacon light map

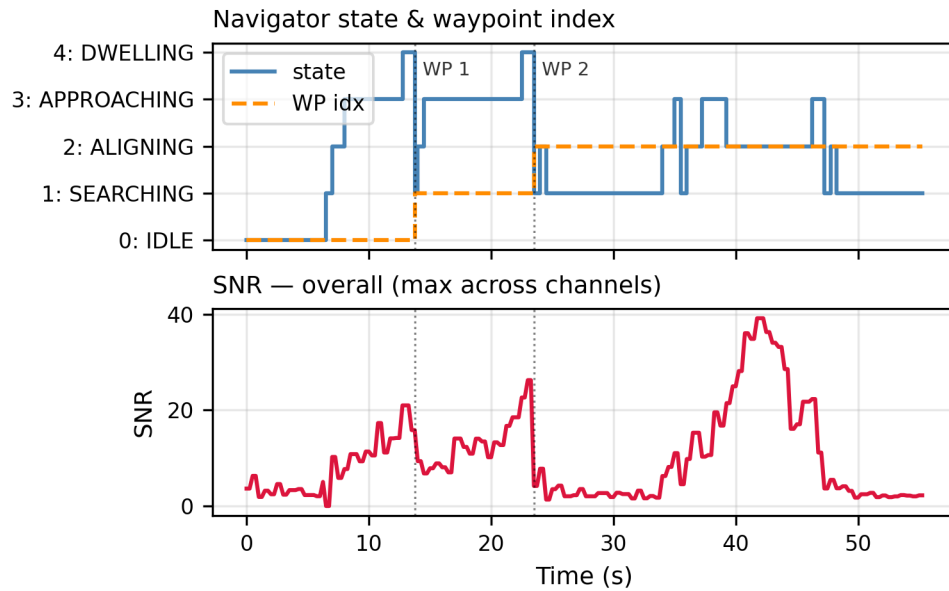


(b) Controller state, active waypoint, and overall SNR over time

Figure 6.4: **Two-waypoint traverse on the CrazyFlie 2.1, with the two beacons modulated at 150 and 170 Hz. The drone reaches each beacon in succession over the light map it builds for that beacon, while the state machine steps through its phases and advances the waypoint index on each arrival.**



(a) Flight path to each beacon over the per-beacon light map



(b) Controller state, active waypoint, and overall SNR over time

Figure 6.5: Three-waypoint traverse on the CrazyFlie 2.1, extending the sequencing to a longer mission whose third waypoint returns to the 150 Hz beacon. The drone steps correctly through the first two waypoints. On the third it reaches the beacon and the SNR climbs above the arrival threshold, but a timing error in the state machine prevents it from latching the arrival, so it overshoots, drifts out of the light cone, and the SNR collapses rather than settling into dwelling.

Chapter 7

Obstacle Experiments

This chapter tests the central task of the thesis, autonomous drone navigation towards a light beacon when an obstacle blocks the direct path. We first inspect the light field in a real room and show that the obstacle creates measurable differences in the received light signal, leaving a shadowed region that the controller can exploit for navigation. We then evaluate the fused controller in simulation and show that it can avoid the obstacle using only the light signal first. Next, we fly the same obstacle-avoidance mission on the Crazyflie in the real world. Finally, we report how the same sensing board and controller behave when transferred from the quadrotor platform to a blimp-like drone.

7.1 Experimental Setup

We study the obstacle task in two stages, each isolating a different source of difficulty. We first repeat the simulation from Chapter 3, using the same scene in which the bearing-only and gradient-only baselines failed, and show that the fused controller succeeds where they did not. We then move to the real room, flying the Crazyflie 2.1 with the photodiode board from Chapter 4 and the full signal pipeline from Chapter 5, to confirm that the controller remains effective under real-flight conditions. In both stages, the obstacle is placed between the drone’s start position and the beacon. We deliberately place the box slightly to one side of the direct line of sight of the beacon, rather than fully centred in it. A box placed directly in the middle of the line of sight would hide the beacon along the main light path, leaving the drone with little useful signal at the start and turning the experiment into a blind search. By shifting the box to one side, the obstacle occludes the beacon over only part of the path while leaving a route that the drone can still discover from the light field. This setup is therefore a more meaningful and realistic obstacle-avoidance test. The offset remains small enough that the drone is still initially drawn toward the light, but must detect the obstacle’s effect on the light field before colliding with it.

The simulated scene is the one already shown in Section 3.1, with the beacon, the obstacle pillar, and the drone approaching from one side, so we do not repeat it here. Figure 7.1 shows the real-room geometry. The room carries the same two beacons as the open-space traverses of Chapter 6, and each obstacle flight targets one of them. The photograph in Section 6.1 shows the same box already

in place. As with the open-space setup of Chapter 6, the top-down geometry of Figure 7.1 is an indicative reference to the room’s layout rather than a guide on the exact placement of the items in the setup.

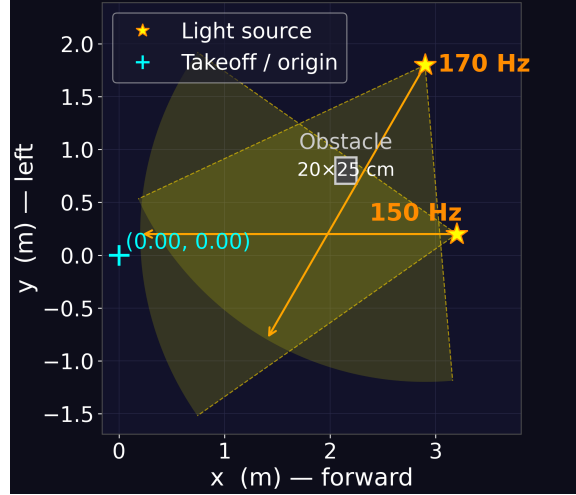


Figure 7.1: **Top-down geometry of the real-room obstacle experiment.** The room carries the same two beacons as the traverses of Chapter 6; each obstacle flight targets one of them, with the obstacle set into the route toward it and a little to one side. The layout is indicative rather than a precise survey of the room.

7.2 SNR Field Mapping

The fused controller relies on the light map when the obstacle occludes the beacon, so before flying the obstacle mission, we first verify that the real-room SNR field behaves as the design assumes. Two properties must hold. First, the light signal must increase toward the beacon, so that a gradient exists for the drone to climb. Second, the obstacle must leave a recognisable mark in the field. We check both properties by directly surveying the light field with a predefined flight path through a set of fixed points, rather than inferring the field from a navigation run.

We map the field with a survey flight that covers the area in a regular back-and-forth raster, or lawnmower pattern. The drone holds a fixed altitude of 0.8 metres, the same height used during navigation flights, so the survey samples the field through which the controller actually flies. It moves at a slow, steady speed of about 0.2 metres per second while logging the SNR at each photodiode. The map records the data in an area of about 4.5 m^2 , which is divided into 5 cm^2 cells, producing a field map much denser than any single navigation flight requires. This gives four field maps in total: the two beacons, modulated at 150 and 170 Hz, each mapped once with the obstacle in place and once without it.

These maps serve as a reference, not as a metric survey of the room. The drone’s own odometry localises their positions, and that odometry drifts over the course of the flight and depends strongly on the drone’s starting angle.

Even a small initial heading error will shift where the whole surveyed area lands, so the four maps cannot be laid directly on top of one another, and the distances within any one of them are not a true measurement of the room’s geometry. What they do show, reliably, is how the SNR field behaves relative to the beacon. Showing the field relative to the beacon is all we need, because the controller acts on a heading rather than an absolute beacon position. The navigation algorithm of Chapter 3 is built around this heading-based principle, so the odometry drift that makes the maps unsuitable as a metric survey does not impair the navigation itself.

Without the obstacle, the SNR field rises smoothly toward the beacon. As the obstacle-free panels in Figures 7.2(a) and 7.3(a) show, each beacon field increases from the far side of the room to a clear peak near the corresponding beacon. This confirms that the real-room light field provides the gradient signal required by the controller.

Placing the obstacle creates a region of low SNR behind it, as the with-obstacle panels of Figures 7.2(b) and 7.3(b) show. Where the box stands between the survey and the beacon, the modulated signal can no longer reach the photodiodes directly, so the recovered SNR drops and leaves a shadow in the field. This shadow is precisely the region in which the bearing becomes unreliable and the drone must navigate on the stored map instead, the situation the recovering phase of Chapter 3 handles.

In these survey maps we draw the obstacle larger than the real box used in the rest of the experiments. The true size of the box is as shown in Section 7.1 and is the same as in the experiments of Section 7.3.2. We enlarge the box in code because we want to guard against the drift in odometry of the drone. This larger virtual box will make sure that the drone avoids the real box even when its odometry has drifted.

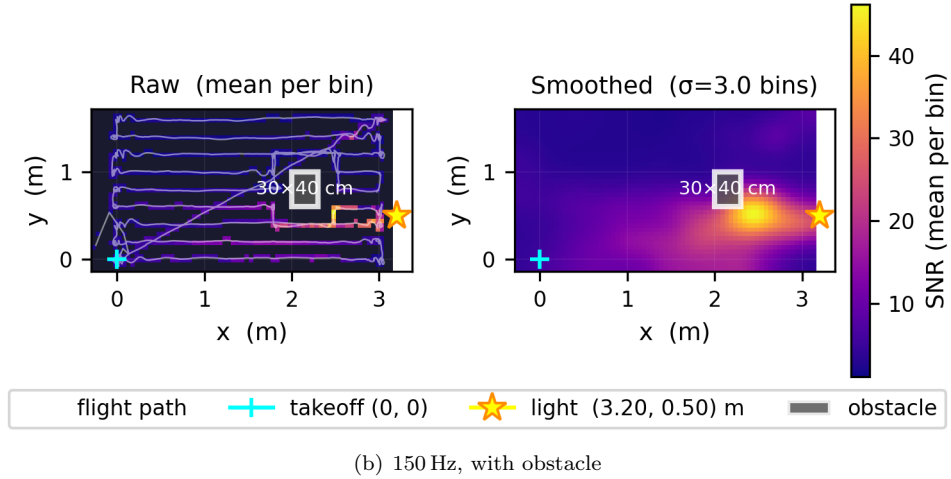
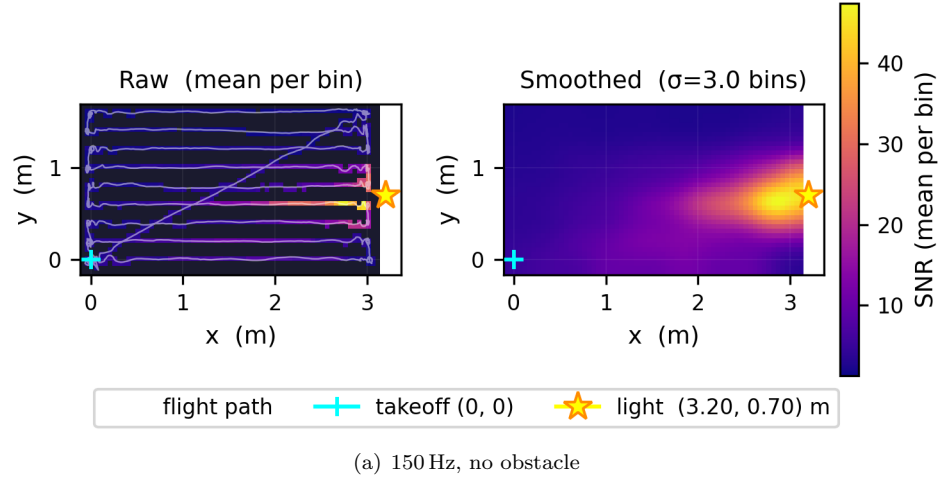
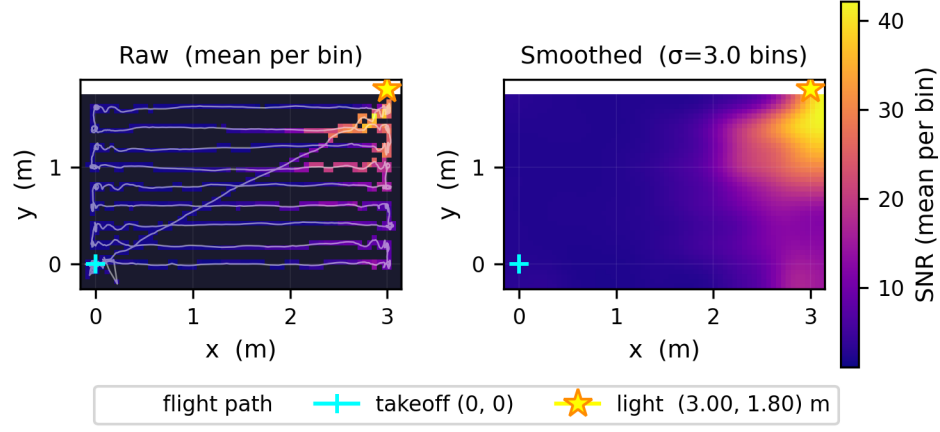
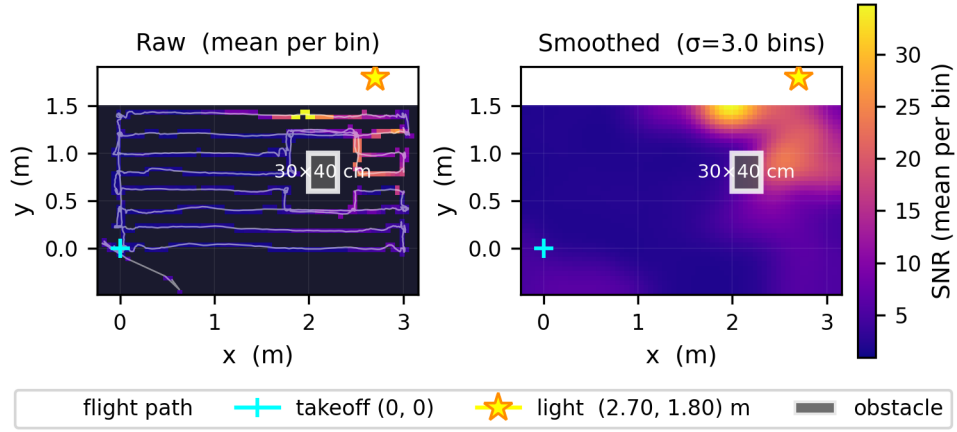


Figure 7.2: SNR field of the real room for the 150 Hz beacon, surveyed on a dense raster at a fixed altitude: Figure 7.2(a) is without the obstacle and Figure 7.2(b) with it. Each panel shows the raw per-cell mean alongside a smoothed version, with the flight path overlaid. Without the obstacle the field rises smoothly to a peak at the beacon; with the obstacle a low-SNR shadow opens behind the box. The colour encodes SNR, scaled independently in each panel between its own minimum and maximum, so it shows the shape of the field rather than absolute values comparable between panels. Positions are localised from odometry and serve as a reference, not a metric survey.



(a) 170 Hz, no obstacle



(b) 170 Hz, with obstacle

Figure 7.3: SNR field of the real room for the 170 Hz beacon, in the same format as Figure 7.2: Figure 7.3(a) is without the obstacle and Figure 7.3(b) with it. The field again rises to a peak near the beacon and shows the obstacle's shadow, with the colour again encoding SNR scaled independently per panel. A slight dip also appears near the 150 Hz beacon, which we tentatively attribute to that beacon saturating the photodiodes and lowering the recovered SNR across all frequencies.

7.3 Obstacle Avoidance

7.3.1 Simulation

After verifying that the surveyed SNR field contains the light-gradient structure required by the controller, we test the avoidance task itself. We first evaluate the fused controller in simulation, using the same scene from Chapter 3, where a pillar blocks the direct line of sight between the drone and the light source. As shown in Figure 7.4, the proposed controller succeeds in the same scenario where the bearing-only and gradient-only baselines failed in Section 3.1. The controller begins with the bearing estimate alone, because the recorded light map is still too sparse at the start to support a reliable gradient. Once the map has enough spatial coverage and the gradient estimate becomes reliable, the controller blends the gradient with the bearing estimate and steers the drone using the combined direction. This blended heading naturally curves the trajectory around the pillar, allowing the drone to avoid the obstacle and reach the beacon without stopping to search again when the direct light signal becomes weak. Importantly, this avoidance behaviour does not come from the drone falling back to the *searching* state and following a predefined search pattern. The drone remains in the *approaching* state, while the gradient information actively pulls it away from the pillar. This simulation therefore validates that the fused controller can avoid the obstacle and reach the target safely in a case where both baseline controllers fail.

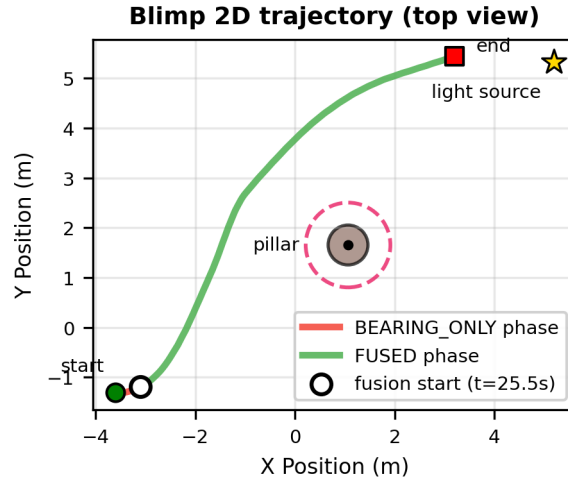


Figure 7.4: In simulation, the fused controller routes around the obstacle and reaches the beacon on the blended bearing-and-gradient heading, where the bearing-only and gradient-only controllers fail in Section 3.1.

7.3.2 Real-World Flight

However, success in simulation does not guarantee success in the real world, because real flight introduces light signal disturbances that the simulation does not model: ambient light, electrical and motor noise in the photodiode signals, and drift in the drone’s odometry. Each of these disturbances can degrade the bearing estimate and the light map on which the controller depends. We therefore fly the obstacle mission on the Crazyflie to confirm that the avoidance behaviour demonstrated in simulation also holds during real-world flight. The drone takes off from the origin and heads toward a single beacon, with the box placed slightly to one side of the straight-line path between the drone and the beacon.

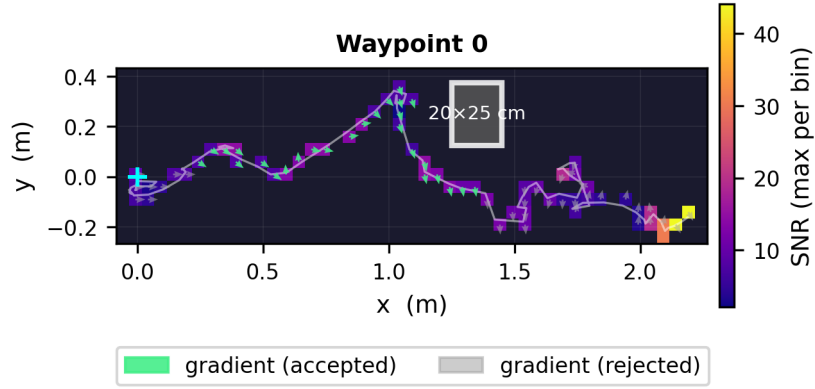
We run the experiment multiple times, but present only two representative runs in which the drone passes close enough to the obstacle to demonstrate the avoidance mechanism. Although the drone reaches the beacon in all runs, many trajectories stay well away from the box, so the obstacle does not significantly affect the flight. The two runs shown here are selected because they bring the drone near the obstacle and therefore expose the controller to the avoidance condition. They also illustrate two different ways in which the controller responds to the obstacle. In both selected flights, the drone reaches the beacon by steering around the box rather than colliding with it, and the controller correctly registers arrival before entering the dwelling state. Because we do not conduct a controlled repeatability trial, these runs demonstrate that the avoidance behaviour is feasible, but we do not quantify the success rate in this evaluation.

We first examine the flight in Figure 7.5, which shows how the controller uses the stored light map when the direct beacon signal becomes unreliable. The upper panel plots the trajectory from the origin as the drone steers past the box, drawn over the light map built during flight, where each cell stores the measured SNR. The lower panel tracks the controller state and the overall SNR over time.

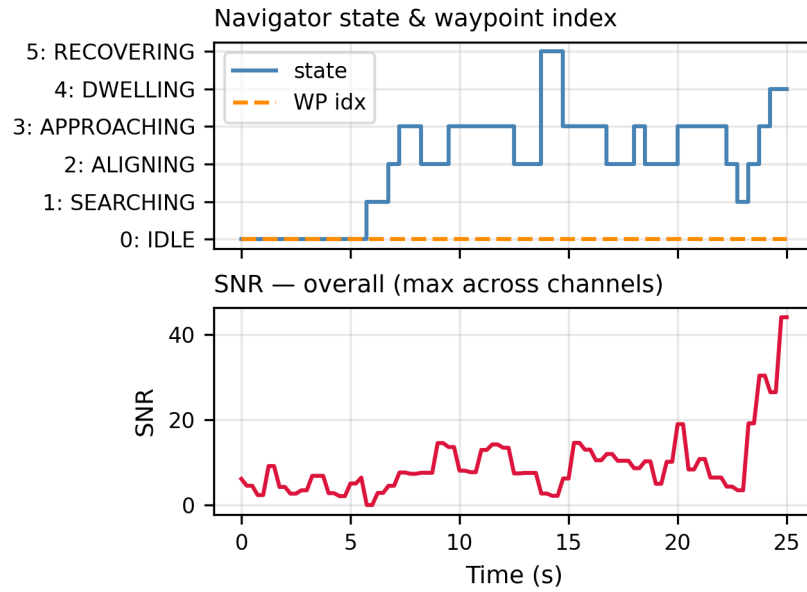
During the flight, the state machine of Chapter 3 progresses through *searching*, *aligning*, and *approaching*. As the drone moves behind the obstacle, the beacon becomes occluded and the overall SNR drops sharply, so the magnitude of the bearing estimate falls below its trust threshold and the bearing no longer provides a usable heading. Unlike in simulation, the real flight therefore loses the bearing for a short stretch as the drone passes behind the obstacle.

This is exactly the situation a bearing-only controller cannot handle: without memory of the light field, the sudden drop in SNR would leave it with no confident direction to follow. The stored map provides that direction. Because the gradient fitted to the map still satisfies its reliability conditions, the controller enters *recovering* and follows the gradient estimate alone, climbing the stored light field to move out of the obstacle’s shadow. Once the bearing’s confidence rises back above its trust threshold, the controller returns to *approaching*, moves directly towards the beacon, and enters *dwelling*.

The second flight reaches the beacon without entering the *recovering* state, representing the other operating scenario. The difference between the two flights comes from where the drone passes relative to the box’s shadow, not from any change in the controller. The first path passes through a deeper part of the shadow, where the beacon is occluded and the bearing estimate becomes unreli-



(a) Flight path over the light map the drone built



(b) Controller state and overall SNR over time

Figure 7.5: **Real-world obstacle flight on the CrazyFlie 2.1.** The drone steers around the box to reach the beacon; as it rounds the obstacle the SNR falls and the bearing becomes unreliable, so the state machine enters the recovering phase and follows the gradient map alone until the signal comes back.

able, whereas the second path skirts the shadow’s edge. In the second flight, the drone therefore keeps a usable bearing throughout the approach, so the controller remains in the *approaching* state and never needs to rely on the map alone. The blended bearing-and-gradient heading is sufficient to curve the trajectory around the obstacle, as Figure 7.6 shows. Together, the two flights show that the controller uses the light map only when the direct light signal drops significantly: it recovers using the gradient when the bearing falls below its trust threshold, and otherwise avoids the obstacle using the fused heading while the bearing remains valid.

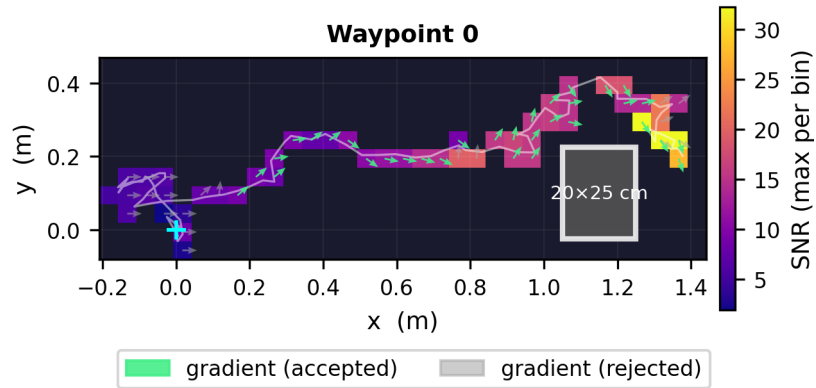
7.4 Generalisation to a Blimp Platform

Our validation spans both the Webots simulation and the Crazyflie 2.1, but the contribution is not intended to be tied to a single platform. We therefore also demonstrate that the sensing system can be deployed on a different airframe. This validation is carried out in collaboration with Huang et al. [15], who developed the lighter-than-air blimp and the Webots simulation used in this thesis. The blimp is a closely related micro-drone platform, supported by a helium balloon and driven by a different motor layout. We mount the same light-sensing board and run the same controller proposed in this thesis on their blimp platform, and confirm that it can reach several modulated beacons in sequence and avoid obstacles, as it does on the Crazyflie quadrotor. The same board and algorithm therefore operate across two different airframes, demonstrating that the method proposed in this thesis depends primarily on the light-sensing and control strategy rather than on the specific drone platform.

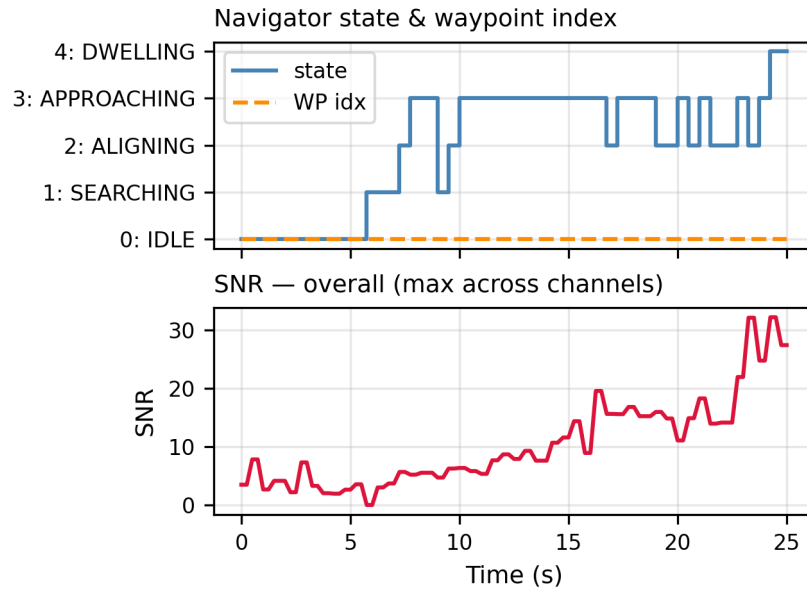
However, we observe a substantial difference in sensing performance between the blimp and the Crazyflie: the blimp flights achieve a higher SNR than the Crazyflie flights. Since the same photodiode board is mounted in the same position on both airframes, this difference is likely caused by the drone platform rather than the sensor itself. The helium balloon keeps the blimp aloft, so its motors run only when the platform moves and consume much less power. In contrast, the Crazyflie must drive its motors continuously to remain airborne. The following section examines how this platform difference affects the performance of the sensing board.

7.4.1 Motor Noise on the CrazyFlie

One major difference between the blimp and the Crazyflie is how their motors are used. To remain airborne, the Crazyflie must drive all four motors at a high duty cycle, whereas the blimp platform from the work by Huang et al. [15] relies on buoyancy and uses its motors only lightly. This suggests that the motors are likely the main source of the sensing degradation, so we measure their effect directly. We hold the board at a fixed pose and record the SNR while running an increasing number of motors, both with and without the hardware anti-aliasing filter. Table 7.1 and Figure 7.7 show the results. The SNR falls steadily as more motors spin. With the hardware filter in place, it drops from about 61 with all motors off to about 25 at full throttle, confirming that motor operation is the dominant disturbance.



(a) Flight path over the light map the drone built



(b) Controller state and overall SNR over time

Figure 7.6: **A second real-world obstacle flight in which the drone keeps a usable bearing throughout, so the controller stays in its approaching phase and steers around the box on the blended heading without ever entering the recovering phase.**

A motor introduces noise into the board in two ways. First, it draws current, which can create electrical interference. Second, it mechanically shakes the board through motor vibration. To separate these two effects, we repeat the four-motor case with the motors moved onto extension wires, which mechanically isolates the motors from the board while keeping them electrically connected. With the filter in place, this setup recovers about half of the lost SNR, increasing it from about 25 to about 43. This result shows that mechanical vibration and electrical interference both contribute substantially to the sensing degradation. The mechanical component is especially important for the Crazyflie because the airframe that transmits motor vibration is also the structure that provides lift. As a result, the sensing board cannot be fully isolated from motor vibration without changing the drone’s mechanical design.

The motors are therefore the main reason the CrazyFlie records a lower SNR than the blimp: because the blimp drives its motors far less, it avoids most of the SNR loss they cause. Motor noise is also hard to exclude since at different duty cycles the motors will introduce different frequencies of noise. Additionally, the mechanical vibrations they cause are hard to avoid, since the motors need a mechanical connection to the drone to even be able to apply the forces to move the drone. We leave both issues for future work to investigate and solve.

Table 7.1: Overall SNR on the CrazyFlie at a fixed pose while running an increasing number of its motors, with the hardware anti-alias filter left unpopulated and with it soldered. Each cell is the mean $\pm$ standard deviation. The last row repeats the four-motor case with the motors moved onto extension wires, mechanically isolating them from the board.

Configuration	No HW filter	HW filter
0 motors	26.7 ± 1.8	60.6 ± 16.3
1 motor	24.2 ± 1.8	49.2 ± 10.9
2 motors	23.2 ± 1.7	46.0 ± 9.0
3 motors	18.0 ± 2.3	28.7 ± 3.2
4 motors	16.7 ± 2.2	24.8 ± 2.6
4 motors, extension wires	19.3 ± 1.5	43.1 ± 7.3

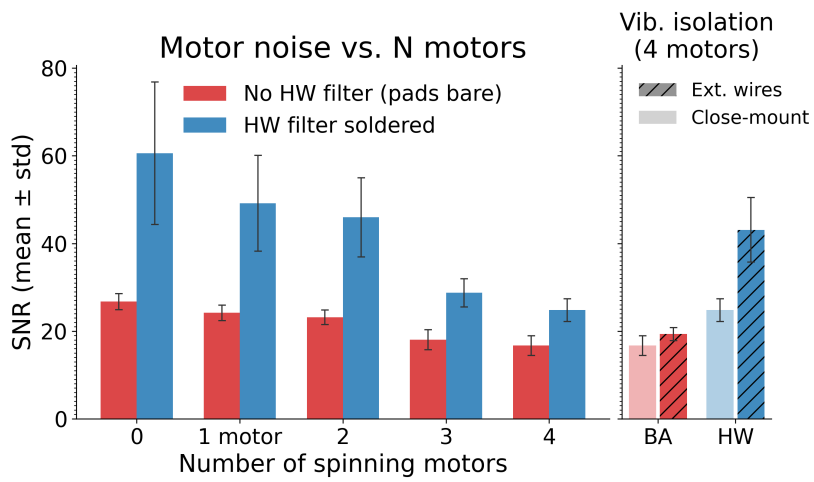


Figure 7.7: Overall SNR on the CrazyFlie at a fixed pose. The left panel shows the SNR as the number of spinning motors increases, for the board with the hardware anti-alias filter left unpopulated (bare pads) and with it soldered; the SNR falls as more motors spin. The right panel is the vibration-isolation test at four motors. Here BA denotes the bare board with no hardware filter and HW the board with the hardware filter soldered. For each, the solid bars are the close-mounted motors and the hatched bars the same motors moved onto extension wires, which isolates their vibration from the board while leaving them electrically connected.

Chapter 8

Conclusions and Future Work

In this thesis, we ask a central question for autonomous drone deployment: how can autonomous navigation be provided on a weight-, size-, and power-constrained micro-drone with minimal changes to the environment? We address this question with a visible-light navigation system that uses lightweight on-board light sensing to guide the drone toward modulated light beacons, requiring only minimal modification to existing lighting infrastructure. To make the system robust in cluttered indoor environments, we extend it so the drone can visit several beacons in succession and reach a beacon whose direct light an obstacle partly blocks.

The work builds on the light-guided navigation system proposed by Huang et al. [15], especially their bearing-angle estimation and gradient-based light-source seeking algorithms. The bearing estimate provides an immediate direction toward a visible beacon, but it keeps no memory of the space the drone has already observed. The gradient method, in contrast, can follow changes in the light field, but it is slower because it must infer direction from measurements gathered during motion. These methods are insufficient for the more realistic missions considered in this thesis. They do not by themselves provide robust sequencing across multiple beacons. A bearing-only controller can also fail when an obstacle weakens or blocks the direct light signal, even though the bearing is the most efficient method in open-space source seeking.

This thesis set out to overcome these limitations within the constraints of a micro-drone. Meeting its two goals required solving four challenges, which follow from the three research questions of Chapter 1. Table 8.1 maps the research questions onto these challenges and goals. The following paragraphs discuss each challenge and how this thesis addresses it.

The first challenge is whether the sensing is feasible at all. We find that it is not the limiting factor. The custom board of Chapter 4 provides eight-direction light sensing for under half of the drone’s payload budget. It carries no processor of its own and adds no functionality the drone does not need. Useful directional light sensing is therefore well within the reach of a micro-drone. The difficulty lies in the navigation, not in the sensor.

The second challenge is to recover a trustworthy beacon signal in flight.

Table 8.1: **How the three research questions of Chapter 1 map onto the four challenges solved in this thesis and the two navigation goals they enable. The first research question concerns feasibility and underpins both goals; the second and third are the goals themselves, each resting on one further challenge.**

Research question	Challenge	Goal
RQ1: Can the drone carry the light sensing on board and recover a reliable beacon signal in flight?	Challenge 1: Fit the light sensing within the drone’s budget. Challenge 2: Recover a reliable beacon signal in flight.	Foundation for RQ2 and RQ3
RQ2: Can it navigate to several beacons in succession?	Challenge 3: Distinguish beacons by modulation frequency.	Goal 1: Visit several beacons in sequence
RQ3: Can it navigate around an obstacle blocking the beacon?	Challenge 4: Build a spatial memory of the light field.	Goal 2: Reach a beacon despite an obstacle

Chapter 5 develops the signal pipeline that does this. The decisive factor is how the pipeline estimates the noise floor. Using a median rather than a mean more than doubles the in-flight signal-to-noise ratio, from about 15 to about 32. The hardware anti-alias filter matters too, but only in combination. On its own it barely changes the ratio, yet removing it from the full pipeline lowers the in-flight ratio from about 32 to 22. Recovering the signal in flight therefore depends less on the measurement itself than on rejecting the noise around it.

The third challenge supports the first goal: distinguishing several beacons. We modulate each beacon at its own frequency, so a single recording carries every beacon as a separate peak, and the controller chooses its target by frequency. With this, the drone reaches several beacons in succession in real flight, as Chapter 6 demonstrates. The first goal is therefore met.

The fourth challenge supports the second goal, which is the central problem of the thesis: reaching a beacon when an obstacle blocks the direct path. The enabling idea is spatial memory. The fused controller of Chapter 3 builds a map of the light field as it flies and estimates a gradient from it. The drone therefore retains a direction to follow even when the beacon is hidden. The contrast with the prior methods is sharp. As Chapter 7 shows, the bearing-only controller flies straight into the obstacle, and the single-photodiode gradient stalls against it. Both fail immediately. The fused controller instead detects when the beacon is occluded and curves around the obstacle without stopping, reaching the beacon in both simulation and real flight. The two real flights show this in two regimes. In the first, the bearing is lost behind the obstacle, and the drone navigates on the stored map alone. In the second, the blended heading alone is enough to round the obstacle, without ever falling back on the map. The second goal is therefore met: the drone keeps its efficiency while avoiding the obstacle, rather than crashing in a cluttered environment.

Underpinning both goals is the fused controller, the core technical contribution of this thesis. It adds a spatial memory of the light field to the bearing and gradient primitives of Huang et al. [15]. At each moment it decides which

of the two estimates to trust.

Taken together, the four challenges are solved, and the two goals with them, in both simulation and real flight. Huang et al. [15] reached a single beacon in open space. This thesis reaches several in succession and routes around an obstacle that hides one, using only the lamps already in the room.

The result is not limited to one airframe. In a collaboration with Huang et al., whose blimp and simulator this work builds on, the same board and controller also navigate on their blimp. The navigation therefore depends on the light sensing, not on the platform.

Light-guided navigation thus moves a step closer to a practical, low-infrastructure positioning method for the smallest indoor drones.

8.1 Future Work

Some limitations remain in the current system, and each points to a concrete direction for future work, so we treat the two together. We group them into improvements to the sensing hardware, extensions to the navigation method, and a broader characterisation and robustness study.

8.1.1 Improving the Sensing Hardware

The most immediate limitation is the system’s operating range, which the photodiode sensing circuit bounds at both ends. When the drone is far from a beacon, the modulated signal becomes weak relative to steady background light. The SNR then falls below the level needed for reliable navigation, as measured in Chapter 5. When the drone is very close to the beacon, the photodiodes instead approach saturation and the recovered signal flattens. The dip seen near a strong beacon in Section 7.2 suggests this upper bound. The usable region therefore lies between these two limits. This limitation comes from the current circuit, not from visible-light navigation itself, and a redesigned front end could widen the range at both ends.

A transimpedance amplifier with adjustable or switched gain would let the front end follow the signal as the drone moves. Near a beacon, where the photodiodes now saturate, it would back off; far from it, where the signal is weak, it would amplify, widening the clean band at both ends. Higher-order band-pass filters tuned to the beacon band would help further, passing the modulation more selectively and rejecting the ambient light that otherwise reduces the sensor’s usable range, the effect measured in Section 5.3.

The motor-noise measurements of Chapter 7, reported in Section 7.4.1, show the CrazyFlie’s motors are the main cost of flight, and that vibration and electrical interference contribute in roughly equal measure. This opens a clear hardware direction. Soft-mounting the board or the motors to damp the vibration should recover the mechanical half of the loss, while better shielding and supply filtering would address the electrical half. The same measurements explain why the blimp records a higher SNR than the CrazyFlie: as Section 7.4 shows, the blimp runs its motors far less. A controlled side-by-side flight of the two carrying the same board would then measure how much of that gap soft-mounting and shielding can close.

Photodiode geometry is a further direction, and it is also what restricts the system to fixed-altitude flight. The photodiodes sit in a horizontal ring, as shown in Section 4.4. The bearing therefore only resolves direction within that plane, and the array responds weakly to light from above or below it. The ring was chosen deliberately, to reduce the overhead ambient light entering the sensors. A more spherical arrangement would let the system sense the vertical directions too. It would admit more ambient light in exchange, which a more capable sensing circuit could manage.

Covering elevation carries a hardware cost. A single ring uses N photodiodes, with $N = 8$ in Chapter 4. Covering the upper hemisphere at the same angular spacing would need several such rings stacked over elevation, so the sensor count grows roughly with the square of the per-ring count. Full spherical coverage is not required in practice, since the drone need not sense light from directly below. This keeps the count well under that worst case.

The same redesign would also support faster, more agile flight. During aggressive manoeuvres the drone tilts, changing where each photodiode points. In this thesis we keep the flight gentle and near-level, so that the horizontal-plane bearing of Section 2.4.1 stays valid. Future work could instead measure the tilt and correct the bearing error that more aggressive motion introduces.

8.1.2 Extending the Navigation

The controller currently acts only in the horizontal plane, since the algorithms of Chapter 3 use a two-dimensional light map indexed by the drone’s horizontal position. Lifting the navigation into three dimensions would let the drone seek beacons above or below its flight height. The algorithmic extension is relatively direct. The light map of Section 3.4 would store intensity over a volume, indexed by horizontal position and altitude rather than over a flat grid. This raises it from an $O(G^2)$ grid to an $O(G^3)$ volume for G cells per axis. The gradient estimator of Section 3.5 would simply add a third coordinate. This increases computation, but not prohibitively. The harder part is sensing: the extension needs light variation along the elevation axis, which the horizontal photodiode ring cannot provide. For this reason, three-dimensional navigation should be developed together with the photodiode-board redesign above, since elevation sensing cannot come from software alone.

Future work should also test the controller in more diverse obstacle settings than the single box used in Chapter 7. Real deployments may contain pillars, tables, shelves, walls, or other objects that block and reshape the light field in different ways. Multiple obstacles, narrow corridors, and concave layouts could each stress the map-based recovery mechanism of Section 3.7 by creating shadows, detours, or local maxima that the stored gradient may not escape. Evaluating these cases would clarify when the current recovery strategy is sufficient and when additional planning or obstacle awareness is needed.

A further direction is to relax the static-environment assumption. All experiments in this thesis keep the beacons and obstacles fixed while the drone navigates. The map is rebuilt from scratch for each beacon, as Section 3.4 describes. A change between one beacon and the next is therefore absorbed when the map is reset. The harder case is a beacon or obstacle that moves *during* a single approach, once the drone has already mapped part of the light field. The affected cells then hold stale values, and the controller may follow an out-

dated gradient until it remaps the region. Handling this robustly would let the controller detect that a stored cell no longer matches the field and refresh it on the fly, rather than relying on the per-beacon reset alone. Evaluating the controller under such dynamic conditions would clarify when the approach remains effective.

Another direction is to refine how the controller fuses the bearing and gradient estimates. The current controller uses fixed, equal weights whenever both estimates are valid, as defined by the fusion rule in Equation 3.5. When only one estimate remains valid, the state machine mentioned in Section 3.7 switches to only using that estimate. Our design is simple, but it treats each input as either trusted or ignored, even though both estimates already carry information about their reliability. A more flexible approach would be for the controller to adjust the bearing and gradient weights, w_B and w_G , according to their confidence. The bearing confidence could come from the summed photodiode magnitude used as the validity score in Section 3.3, while the gradient confidence could come from the R^2 value of the plane fit in Equation 3.3. With this design, the drone would rely more on the bearing when the beacon is clearly visible and more on the gradient when the bearing weakens inside an obstacle shadow, while still allowing both estimates to contribute to the final heading. Such confidence-weighted fusion would make the transition between the two estimators smoother than the current hard-threshold hand-off.

8.1.3 Characterisation and Robustness

The real-world results in this thesis are demonstrations rather than a measure of reliability. We flew more missions than the few shown in Chapters 6 and 7, and the drone reached its beacon on most of them, but we did not run a controlled repeatability study. This matters for deployment: a system trusted to navigate on its own must be characterised by how often it succeeds, not only by whether it can. Quantifying that success rate across varied conditions is the broad goal of the study below.

The controller contains several tunable parameters, listed in Appendix A. Their values are currently selected empirically for the experiments in this thesis, so future work should quantify how sensitive navigation performance is to each setting. A formal ablation study would address this by repeating the same missions under fixed conditions while varying one parameter at a time and keeping the others unchanged. This study would also provide the controlled evaluation the present demonstrations lack. By measuring success rate, path length, travel time, and recovery behaviour under each parameter setting, such an ablation study would replace the current hand-tuned values with practical tuning guidelines for deploying the system in new environments.

The arrival condition deserves closer attention before real deployment. In the current state machine, described in Section 3.7, the controller registers arrival using a single SNR threshold. The three-waypoint flight in Section 6.3 shows the weakness of this design: although the SNR rises above the threshold near the final beacon, the state machine does not reliably latch the arrival event, and the drone continues flying. A more robust arrival mechanism should therefore avoid relying on an instantaneous SNR value alone. For example, the controller could require the SNR to stay above the threshold for several consecutive frames before confirming arrival, or estimate the drone’s position relative to the target

beacon. This would make beacon handover more reliable in longer missions.

Bibliography

- [1] A. Altena, S. Luesutthiviboon, G. de Croon, M. Snellen, and M. Voskuijl. Comparison of acoustic localisation techniques for drone position estimation using real-world experimental data. In *Proc. 29th Int. Congress on Sound and Vibration (ICSV)*, 2023.
- [2] Kartik B. Ariyur and Miroslav Krstić. *Real-Time Optimization by Extremum-Seeking Control*. Wiley, Hoboken, NJ, USA, 2003.
- [3] Bitcraze AB. Lighthouse Positioning Deck. <https://www.bitcraze.io/products/lighthouse-positioning-deck/>. Product documentation, last accessed 2026.
- [4] Bitcraze AB. Flow Deck v2. <https://www.bitcraze.io/products/flow-deck-v2/>, 2019. Product documentation, last accessed 2026.
- [5] Bitcraze AB. Crazyflie 2.1. <https://www.bitcraze.io/products/crazyflie-2-1/>, 2021. Product documentation, last accessed 2026.
- [6] Åke Björck. *Numerical Methods for Least Squares Problems*. SIAM, Philadelphia, PA, USA, 1996.
- [7] Jennie Cochran and Miroslav Krstić. Nonholonomic source seeking with tuning of angular velocity. *IEEE Transactions on Automatic Control*, 54(4):717–731, 2009.
- [8] Cyberbotics Ltd. Webots: Open-source mobile robot simulation software. <https://cyberbotics.com/>. Last accessed 2026.
- [9] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6), 1981.
- [10] Dario Floreano and Robert J. Wood. Science, technology and the future of small autonomous drones. *Nature*, 521(7553):460–466, 2015.
- [11] Jerald G. Graeme. *Photodiode Amplifiers: Op Amp Solutions*. McGraw-Hill, New York, NY, USA, 1996.
- [12] Fredric J. Harris. On the use of windows for harmonic analysis with the discrete Fourier transform. *Proceedings of the IEEE*, 66(1):51–83, 1978.
- [13] Eugene Hecht. *Optics*. Pearson, Boston, MA, USA, 5th edition, 2017.

- [14] Ricardo Ampudia Hernández, Talia Xu, Yanqui Huang, and Marco A. Zúñiga Zamalloa. Firefly: Localizing drones with visible light communication and sensor fusion. In *2023 19th International Conference on Distributed Computing in Smart Systems and the Internet of Things (DCOSS-IoT)*, 2023.
- [15] Harry Huang, Talia Xu, and Marco Zúñiga Zamalloa. Exploiting light to enhance the endurance and navigation of lighter-than-air micro-drones. <https://arxiv.org/abs/2601.13088>, 2026. arXiv:2601.13088 [cs.RO].
- [16] Intel Market Research. Indoor inspection drone market: Growth analysis, dynamics, key players and innovations, outlook and forecast 2026–2032. <https://www.intelmarketresearch.com/indoor-inspection-drone-market-21142>, 2026.
- [17] Muhammad Faisal Iqbal, Zubair Khalid, Muhammad Zahid, and Asim Abdullah. Accuracy improvement in amplitude comparison based passive direction finding systems by adaptive squint selection. *IET Radar, Sonar & Navigation*, 14(5):662–668, 2020.
- [18] Toshihiko Komine and Masao Nakagawa. Fundamental analysis for visible-light communication system using LED lights. *IEEE Transactions on Consumer Electronics*, 50(1):100–107, 2004.
- [19] Ye-Sheng Kuo, Pat Pannuto, Ko-Jen Hsiao, and Prabal Dutta. Luxapose: Indoor positioning with mobile phones and visible light. In *Proc. 20th Annual Int. Conf. on Mobile Computing and Networking (MobiCom)*, pages 447–458, 2014.
- [20] Randall J. LeVeque. *Finite Difference Methods for Ordinary and Partial Differential Equations: Steady-State and Time-Dependent Problems*. SIAM, Philadelphia, PA, USA, 2007.
- [21] Kanti V. Mardia and Peter E. Jupp. *Directional Statistics*. Wiley, Chichester, UK, 1999.
- [22] Raúl Mur-Artal and Juan D. Tardós. ORB-SLAM2: An open-source SLAM system for monocular, stereo, and RGB-D cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262, 2017.
- [23] Alan V. Oppenheim and Ronald W. Schaffer. *Discrete-Time Signal Processing*. Prentice Hall, Upper Saddle River, NJ, USA, 2nd edition, 1999.
- [24] Henry W. Ott. *Electromagnetic Compatibility Engineering*. Wiley, Hoboken, NJ, USA, 2009.
- [25] Parth H. Pathak, Xiaotao Feng, Pengfei Hu, and Prasant Mohapatra. Visible light communication, networking, and sensing: A survey, potential and challenges. *IEEE Communications Surveys & Tutorials*, 17(4):2047–2077, 2015.
- [26] Jorge Peña Queralta, Carmen Martínez Almansa, Fabrizio Schiano, Dario Floreano, and Tomi Westerlund. UWB-based system for UAV localization in GNSS-denied environments: Characterization and dataset. In *Proc.*

- IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 4521–4528, 2020.
- [27] Tilo Strutz. *Data Fitting and Uncertainty: A Practical Introduction to Weighted Least Squares and Beyond*. Springer Vieweg, Wiesbaden, Germany, 2 edition, 2016.
 - [28] Yimiao Sun, Weiguo Wang, Luca Mottola, Zhang Jia, Ruijin Wang, and Yuan He. Indoor drone localization and tracking based on acoustic inertial measurement. <https://arxiv.org/abs/2504.00445>, 2025. arXiv:2504.00445 [cs.RO].
 - [29] Peter D. Welch. The use of fast Fourier transform for the estimation of power spectra: A method based on time averaging over short, modified periodograms. *IEEE Transactions on Audio and Electroacoustics*, 15(2):70–73, 1967.
 - [30] Faheem Zafari, Athanasios Gkelias, and Kin K. Leung. A survey of indoor localization systems and technologies. *IEEE Communications Surveys & Tutorials*, 21(3):2568–2599, 2019.
 - [31] Chunlei Zhang, Daniel Arnold, Nima Ghods, Antranik Siranosian, and Miroslav Krstić. Source seeking with nonholonomic unicycle without position measurement—part I: Tuning of forward velocity. In *Proc. 45th IEEE Conf. on Decision and Control (CDC)*, pages 3040–3045, 2006.
 - [32] Yuan Zhuang, Luchi Hua, Longning Qi, Jun Yang, Pan Cao, Yue Cao, Yongpeng Wu, John Thompson, and Harald Haas. A survey of positioning systems using visible LED lights. *IEEE Communications Surveys & Tutorials*, 20(3):1963–1988, 2018.

Appendix A

Controller Parameters

This appendix lists the parameter values used by the fused navigation controller of Chapter 3 and by the real-world signal pipeline of Chapter 5. The symbols match those introduced in those chapters.

Parameter	Symbol	Value
<i>Bearing</i>		
Minimum signal strength to trust a bearing	—	5
Bearing smoothing factor	β	0.7
Frames a lost bearing is held before dropping	—	3
<i>Light map and gradient</i>		
Grid cell size	r	0.05 m
Cell smoothing factor	α	0.5
Neighbourhood radius	$d_{\min}, d_{\max}$	0.05, 3.0 m
Weight regulariser	ε	0.01
Minimum cells for a gradient	—	7
Minimum weighted R^2	—	0.5
Minimum gradient magnitude	—	0.5
<i>Fusion</i>		
Bearing weight	w_B	0.5
Gradient weight	w_G	0.5
<i>State machine and flight</i>		
Alignment tolerance	—	15°
Re-alignment threshold	—	30°
Arrival signal strength, tuned per flight	—	25–32.5
Search yaw rate	—	30°/s
Cruising forward speed	—	0.20 m/s
Maximum forward speed	—	0.30 m/s
Flight altitude	—	0.8 m
Dwell time per beacon	—	1.0 s
<i>Signal pipeline (Chapter 5)</i>		
Hardware low-pass cut-off	f_c	500 Hz
Sample rate	f_s	500 Hz
FFT size	N	256
Frequency resolution	Δf	1.95 Hz
Window overlap	—	50 %
Frames averaged	N_{avg}	2
Peak search half-window	—	1 bin (≈ 2 Hz)
Noise guard band	—	10 bins
Harmonics excluded	—	4
Magnitude smoothing factor	α_m	0.5

Table A.1: **Parameter values used by the fused navigation controller.**