BatMobility: Towards Flying Without Seeing for Autonomous Drones

Emerson Sie, Zikun Liu, and Deepak Vasisht
University of Illinois Urbana-Champaign

Abstract

Unmanned aerial vehicles (UAVs) rely on optical sensors such as cameras and lidar for autonomous operation. However, such optical sensors are error-prone in bad lighting, inclement weather conditions including fog and smoke, and around textureless or transparent surfaces. In this paper, we ask: is it possible to fly UAVs without relying on optical sensors, i.e., can UAVs fly without seeing? We present BatMobility, a lightweight mmWave radar-only perception system for UAVs that eliminates the need for optical sensors. BatMobility enables two core functionalities for UAVs – radio flow estimation (a novel FMCW radar-based alternative for optical flow based on surface-parallel doppler shift) and radar-based collision avoidance. We build BatMobility using commodity sensors and deploy it as a real-time system on a small off-the-shelf quadcopter running an unmodified flight controller. Our evaluation[1] shows that BatMobility achieves comparable or better performance than commercial-grade optical sensors across a wide range of scenarios.

CCS Concepts

• **Computer systems organization** → **Robotics**; **Sensors and actuators**; • **Computing methodologies** → **Machine learning**.

Keywords

Machine Learning, Quadrotor, Radar, Egomotion, RF Sensing

ACM Reference Format:
Emerson Sie, Zikun Liu, and Deepak Vasisht. 2023. BatMobility: Towards Flying Without Seeing for Autonomous Drones. In *The 27th Annual International Conference on Mobile Computing and Networking (ACM MobiCom '23), October 2–6, 2023, Madrid, Spain.* ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/3570361.3592532

ACM MobiCom '23, October 2–6, 2023, Madrid, Spain

ACM ISBN 978-1-4503-9990-6/23/10…$15.00
https://doi.org/10.1145/3570361.3592532

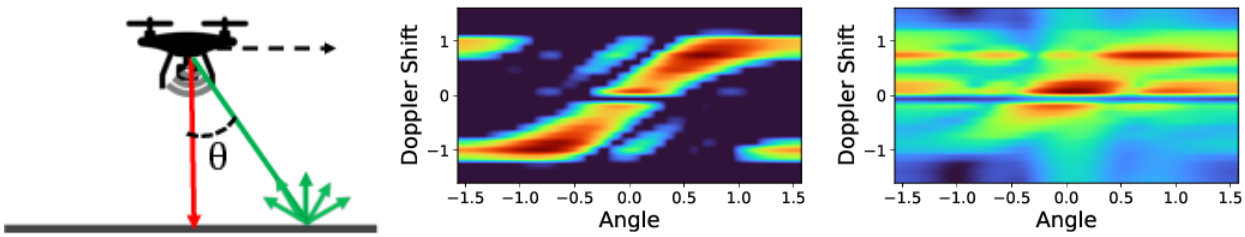
Figure 1: A downward facing radar performs altimetry and ego-velocity estimation (bottom) and a forward facing radar performs collision avoidance (top), removing the need for other exteroceptive sensors such as cameras or lidar.

1 Introduction

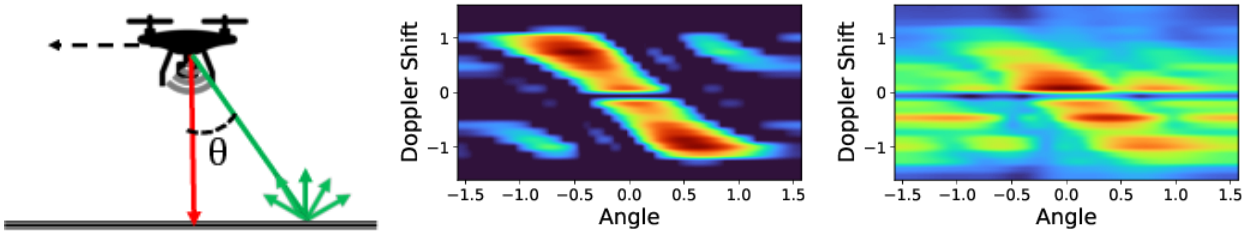
In recent years, we have experienced a proliferation of small UAVs meant for many diverse applications like inventory management in warehouses [44], cargo delivery [67, 68], building inspections [3, 55, 61], mapping & surveying [49, 63], home surveillance [53], and search & rescue operations [60]. Due to their wide ranging applications, the market size for small UAVs is expected to increase to nearly 50 billion US dollars by 2028. However, most UAVs still rely on manual teleoperation by humans. This has led to much research in sensing [49, 55, 63], automation [3, 8, 17], and control [35, 45, 62] for UAVs.

Current approaches for autonomous UAVs mainly rely on optical sensors such as cameras or lidars to perceive the surrounding environment. Systems based on optical sensing readily benefit from decades of computer vision research, but suffer from multiple shortcomings. First, optical sensors are prone to failures in visually degraded conditions due to, e.g., bad lighting conditions, inclement weather conditions like fog, and occlusions like dirt. Second, optical sensors fail to identify relative motion in untextured or featureless environments, e.g., large warehouse floors due to the lack of visual or geometric features. Similarly, optical sensors fail to register transparent objects like glass windows and panels. Finally, and perhaps less obviously, optical sensors

[1]Videos are available at the project website: https://batmobility.github.io



(a) (left) Motion of the UAV, (center) Simulated doppler-angle plot, (right) Observed doppler-angle plot.



(b) (left) Motion of the UAV, (center) Simulated doppler-angle plot, (right) Observed doppler-angle plot.

Figure 2: Estimating Surface-Parallel Doppler Shifts: At mmWave frequencies, even seemingly flat surfaces scatter the signal (green arrows) leading to continuous signatures in our doppler-angle plots. Observed signals are impacted by additional factors: clutter, multipath, and noise.

raise privacy concerns, such as in the case of outdoor delivery UAVs flying over residential areas [69].

Radio-frequency (RF) sensing has emerged as an alternative paradigm to optical sensing in recent years [1, 2, 7, 18, 70]. Given the progress in RF sensing, we ask a simple yet seemingly radical question: *is it possible for UAVs to operate autonomously without using optical sensors at all?* In other words, *can UAVs autonomously fly without seeing?*

To answer this question, we explore the use of commodity single-chip mmWave radars as the sole perception tool on autonomous UAVs. Such radars offer several advantages not only over optical sensors, but over all other types of sensors for UAVs (i.e., ultrasound) as well. First, they are invariant to light levels allowing for 24/7 operation, are robust to weather-related precipitates like fog, and can operate over long range. Second, they only collect coarse-grained information about the surrounding environment and alleviate privacy concerns. Third, they are compact in terms of size and power consumption, contain no moving parts, and can furthermore be hidden behind a facade to prevent the need for frequent cleaning due to dirt and grime. In spite of these advantages, radar is used for limited operations on UAVs (e.g., estimating height from ground) and not as a first-class perception tool.

We introduce BatMobility, a new perception system that relies purely on inexpensive radar, does not have any optical sensors, and can run on small and computationally constrained UAVs. Our contribution in this paper is two-fold. First, we use radar to enable two core navigation primitives that have traditionally been implemented using optical sensors: (a) estimating an UAV's velocity and altitude with respect to the ground to stabilize a UAV and to enable feedback-based control, and (b) obstacle detection to avoid potential collisions. Second, we demonstrate a working system that can run all the RF processing on a embedded single-board computer (SBC) on the UAV and operate the UAV in real-time. In doing so, we identify and exploit trade-offs between sensing accuracy and computational overheads. The design of BatMobility requires us to solve three challenges:

Extracting Surface-Parallel Motion Insights: UAVs commonly use downward facing optical flow sensors (i.e. video cameras) to estimate ground-parallel motion. Optical flow sensors estimate the relative motion of the UAV by measuring the motion-induced visual shift between two consecutive image frames. In BatMobility, we replace this sensor with a downward facing mmWave radar which captures the reflections of the radar signal from the environmental reflectors (primarily the ground). Post-processing on these reflected signals reveals the range, angle, and doppler shift of each reflector. Recall that doppler shift is a frequency shift introduced by the motion of a reflector, and is therefore, a natural candidate for estimating UAV velocity. However, there is a key challenge. For a downward facing radar, the ground/floor moves parallel to the radar and any motion parallel to the radar does not introduce any doppler shift. So, how does one estimate the doppler shift for a surface-parallel motion?

Our first insight is – at mmWave frequencies, most surfaces act as scatterers, not pure reflectors. Therefore, even

seemingly flat surfaces, such as textureless floors have multiple weak reflectors. When the UAV moves, they exhibit unique patterns that reflect the direction of motion in the doppler-angle plane. We illustrate these patterns through an example in Fig. 2. The left figure plots a UAV flying towards right with velocity v. The doppler shift depends on the motion along the line joining each reflection point and the radar on the UAV. Therefore, a reflector at angle θ has a doppler shift $\frac{v \sin\theta}{\lambda}$, where λ is the signal wavelength. This pattern is apparent in the center figure (built using simulation). This shows that the patterns in the doppler-angle plane correspond to the amplitude and direction of the UAV's velocity.

Multipath and Clutter: The second challenge for designing radar-based perception stems from multipath and noise induced by objects in the environment. Objects in the environment introduce additional reflections, which appear to move in different directions and at varied velocities (similar to observing objects in a mirror). These reflections confound any RF-based perception system. This problem is escalated in cluttered indoor environments, which have many reflectors. This effect is demonstrated in Fig. 2(right) collected in a real-world setup. The figure exhibits the expected pattern but has other reflections and patterns embedded as well, largely due to clutter and multipath.

The effect of clutter and multipath makes it challenging to explicitly model the relationship between the doppler-angle plot and the velocity of the UAV. To extract insights from such cluttered data, we implicitly model this effect by using a data-driven approach. Specifically, we build a machine learning model that takes the doppler-angle heatmaps as inputs and outputs the estimated velocity of the UAV. We train this model using data from multiple environments and observe that it can learn an environment-invariant mechanism to translate the heatmaps to velocity, in spite of clutter.

Reducing System Overhead: Finally, BatMobility must be able run on small UAVs in real-time. The limited compute available to small UAVs introduces several practical constraints. For example, we cannot arbitrarily increase the size of doppler-angle heatmaps as it incurs too much data to process. We observe that the design space of RF-based perception is quite large – we can tune chirp parameters, frame rates, heatmap resolutions, model architectures, etc. Each parameter has consequences for the range of operation, sensing accuracy, and computational overhead. Our work explores this design space and identifies several optimizations to enable low compute overhead and can generate velocity estimates up to 50Hz, sufficient to use existing motion control pipelines on UAVs. This ensures BatMobility is reverse compatible with existing flight controllers, as we can simply replace optical sensor inputs with RF-based ones generated by BatMobility.

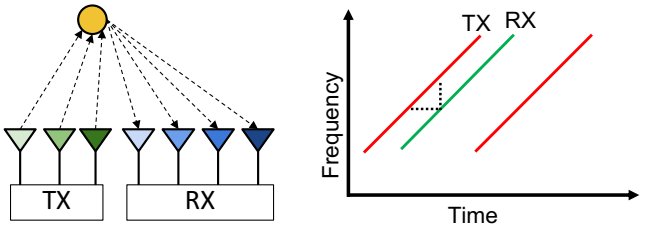


Figure 3: *Left.* A TDM MIMO scheme containing 3×4 TX-RX pairs. This allows for $N = 12$ virtual antenna array for accurate angle estimation of a reflector (yellow). *Right.* Sequence of transmitted FMCW chirps (red) and received echoes (green) for range and velocity estimation of reflectors.

We implement our end-to-end design on a small quadrotor built from commodity off-the-shelf parts. We use two mmWave radars on this UAV as shown in Fig. 1. We use a downward-facing radar to mimic a typical lidar altimeter and optical flow sensor used for state estimates (velocity and altitude), and use a forward-facing radar to mimic a typical stereo camera used for collision detection. Our design can stably hover in place and fly given paths, while running all the compute on a low-power single-board computer on the UAV. Demos are available at the project website[1]. To summarize, our contributions are as follows.

- We identify and characterize the surface-parallel doppler-shift phenomenon at mmWave frequencies for a wide variety of surfaces, including those that are highly smooth, and show how this phenomenon can be seen in doppler-angle heatmaps.
- We train a convolutional neural network (CNN) to uncover the motion of the UAV with respect to the ground from these doppler-angle heatmaps, and show superior performance to state-of-the-art commercial optical flow sensors such as the PMW3901 in a wide variety of scenarios.
- We explore various system design tradeoffs and optimizations needed to make BatMobility compatible with existing flight controllers in a plug-and-play manner. We demonstrate the end-to-end system by deploying it on a real quadcopter.
- We evaluate the use of radar as a collision avoidance sensor, and show that it can be more effective than optical sensors in a wide range of realistic scenarios.

2 Background

We present a high-level background of key principles in optical flow and radar below.

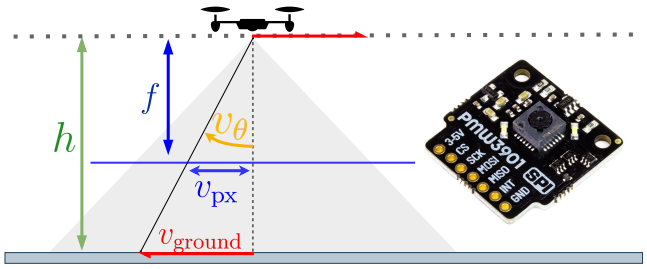


Figure 4: An optical flow sensor first converts averaged optical flow (units in px) to an angular value v_θ using a known focal length f. This is combined with altitude h to determine ground velocity.

2.1 mmWave Radar

Recent years have seen a proliferation of single-chip mmWave radar sensors [14, 39]. A typical sensor leverages Frequency Modulated Carrier Wave (FMCW) chirps, wherein the frequency of the signal varies linearly with time (Fig. 3 right). The sensor emits a sequence of FMCW chirps into an environment and captures the reflected signal at a set of receive antennas. The distance of the reflectors can be inferred from the frequency shift between the transmitted and received chirps, and their bearing angle and relative velocity can be discerned from the apparent phase changes across consecutive receive antennas and chirps respectively. For brevity, we refer the reader to [25] for a thorough treatment of FMCW radar signal processing fundamentals. For the rest of the paper, we will assume the convention that the unit of radar data for post-processing (i.e. radar cube) consists of a 3-D matrix of complex samples of dimension $N_c \times N_a \times N_s$, where N_c is the number of chirps, N_a is the number of receive antennas, and N_s is the number of time samples taken per chirp interval at each receive antenna.

As explained in [25], the angular resolution of a linear antenna array for a fixed inter-antenna spacing improves with the number of antennas, N_a. Most inexpensive single-chip radars have only a small number of physical receive antenna in any given direction (i.e. 2-4 in Fig 5 left), leading to poor angular resolution. A commonly used technique to improve angular resolution is to use time-division-multiplexing multiple-input multiple-output (TDM MIMO). Under TDM MIMO, each transmit antenna takes turns transmitting chirps into the environment. Hence, each distinct transmit-receive pair can be considered a separate (virtual) receive antenna (Fig. 3 left). This enables one to extend the number of antennas in a given direction, or to create a 2-D antenna array for simultaneous azimuth and elevation angle estimation (Fig 5 right).

2.2 Optical Flow Sensors

Optical flow refers to the apparent motion of visual features between two consecutive images. Optical flow sensors estimate the shift in camera view over time and use it to estimate the motion pattern of the UAV. For example, if the UAV moves right, it sees the view of its optical flow sensor moving left. This feedback is provided to the flight controller on the UAV and is useful for two functions: (a) hovering – the ability of the UAV to hover in-place when not required to move; and (b) motion feedback – e.g., if the drone is trying to move forward, is it really moving forward? The operation of such sensors on a UAV is depicted in Fig. 4. Algorithms for estimating optical flow have been studied in computer vision for decades [21, 43]. It is now common to find commercial optical flow sensors that integrate a small camera and an optical flow ASIC in a single package. It remains the most common and inexpensive option to achieve self-stabilization on small UAVs in GPS-denied environments. However, optical flow sensors fail in the dark, or when there is a lack of trackable visual features (e.g. smooth concrete floor). These failure modes are shared by more complex and expensive classes of camera-based techniques for estimating egomotion in GPS-denied environments, i.e. visual odometry (VO) and visual SLAM (VSLAM).

3 Motivation and Goals

Our high level goal is to enable collision-free flight through general unknown environments that may contain static or dynamic obstacles using only onboard radar perception. We envision that this would enable a plethora of use cases that conventional optical sensor + GPS reliant systems struggle with or find impossible. For example, being able to fly in foggy conditions at dawn, navigate in urban environments without crashing into the glass exteriors of modern buildings or power lines, and enabling drone deliveries over residences while respecting privacy. Our strategy involves two steps:

(1) We first achieve simultaneous altimetry and horizontal velocity estimation using a single downward facing radar. This is key since unlike ground vehicles, small quadcopters are innately unstable and will constantly drift unless given motion feedback from the sensing the environment. Without being able to measure its own velocity and altitude, a quadcopter instructed to merely loiter in place will drift off within seconds. Similarly, the quadcopter will be unable to follow more complex commands without motion feedback.
(2) Once self-stabilization is achieved and velocity-based control becomes possible, we fix the drone's altitude and use a forward facing radar to sense the environment and navigate the drone in the 2-D plane corresponding to this fixed altitude. This simplifying assumption is sufficient to test the capabilities and limitations of the system.

3.1 Design Choices

Why not use the front radar for odometry? Drone primarily rely on their downward-facing optical flow sensor at low altitudes where GPS is unavailable. At low altitudes, drones can encounter dynamic obstacles (e.g. moving humans, cars) in front of or besides the drone. These dynamic objects interfere with visual odometry pipelines which rely on a static world assumption [46]. Such dynamic objects will likely introduce errors in radar perception as well. Conversely, a forward facing radar might not encounter any reflectors at all in the environment (e.g. in an open field), leading to a lack of features to perform odometry. On the other hand, at low altitudes a strong ground reflector is always available. Furthermore, even if a forward facing radar is used to obtain odometry estimates, an auxiliary downward facing sensor is still necessary for altimetry. Hence, by using a single downward facing radar to estimate both altitude and horizontal velocity estimates, we prevent any redundancy.

Why not rely on other tracking solutions (e.g. UWB)? Such schemes assume that there is pre-existing infrastructure deployed in the environment, such as UWB anchors and IR cameras. This is infeasible for the vast majority of real world environments that one would encounter in the wild. As such, we do not consider infrastructure-assisted approaches to be fully autonomous.

Is velocity-based control sufficient for navigation? There are several ways of navigating a drone. In classical approaches [19, 36], there is a preexisting map and a desired end point is specified on this map. A path planning algorithm outputs a trajectory (i.e. sequence of coordinates on the map) to the end point. The drone constantly localizes itself on the map using some technique (e.g. SLAM, GPS). The positional error to the next waypoint is repeatedly measured and minimized using a low-level controller i.e. proportional–integral–derivative (PID) control [29]. This paradigm is infeasible for our case since we assume our environment is unknown (we do not have a preexisting map). Indeed, it would be impossible to extensively map out all the environments our drone could possibly fly in.

However, there exists a map-free paradigm for navigation in the form of velocity-based control, which has recently become popular [9, 16, 30, 56] . In this paradigm, commands are relative to the drone's current position (e.g. move forward) rather than being based on an absolute global coordinate system. Hence, localization in any global reference frame is not necessary. However, the drone needs velocity-feedback to ascertain that it is following the commands correctly. To navigate, all the drone must do is to constantly react to its sensory input (i.e., front radar) in a sensible manner. This is sufficient for certain key tasks such as lane or tunnel following. BatMobility provides velocity-based feedback to the drone, but is agnostic to how these commands are generated.

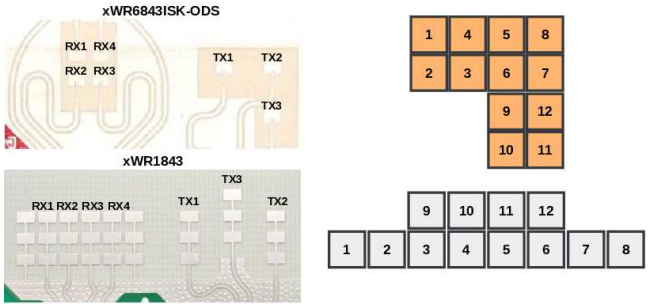


Figure 5: *Left.* Physical antenna array layouts on single-chip mmWave radar boards. *Right.* Corresponding numbered virtual antenna array under TDM MIMO.

4 Surface-Parallel Doppler Shift

Our first goal is to estimate the relative motion of the UAV with respect to its environment using a downward facing radar. A naive solution is to directly adapt a scan matching approach based on point clouds as is done with lidar. This would involve detecting individual reflectors and tracking their relative motion across a sequence of measurements. However, mmWave point clouds are extremely sparse and unstable compared to lidar point clouds, making scan matching extremely difficult [42]. This issue is further exacerbated when pointing radar to a geometrically featureless flat surface (i.e. the ground). However, FMCW radar offers us a better option: doppler shift. Doppler shift directly captures the velocity of reflectors with respect to the UAV. Intuitively, if the radar is moving away from a reflector, the doppler shift is negative. If the radar is moving towards the reflector, the doppler shift is positive.

Seeing Surface Parallel Motion: The key challenge for our design is that the motion of the UAV is parallel to the ground. Specifically, for a reflector placed at angle θ with respect to the radar (see Fig. 2), the doppler shift corresponds to a velocity of $v \sin\theta$, where v is the radial velocity of the object. Therefore, when a radar is facing downwards, the primary reflector, i.e. the floor, is at angle $\theta = 0^\circ$ and induces a zero doppler shift.

Our first insight is that due to the high frequency of mmWave signals, large reflectors like floors induce scattering, as shown in Fig. 2. Specifically, they can feature many weak reflectors at oblique angles with respect to the radar. If we densely aggregate the doppler shift for these reflectors in a doppler-angle heatmap, it should appear as a continuous curve that varies as $v \sin\theta$, where v is the velocity of the drone.

Dealing with Low Resolution: But what does this pattern look like under realistic hardware constraints? Note that the antenna arrays on single-chip radars (illustrated in Fig. 5) usually falls under one of two categories: (a) having a 1D antenna array with 8 virtual antennas in the azimuth direction [22], or (b) having a 2D antenna array but only having 4 virtual antennas in the azimuth and elevation direction [23]. We need to choose option (b) since we must estimate motion along two degrees of freedom in the ground-parallel plane. As explained in Sec. 2, this leads to an extremely low angular resolution of nearly $1/2$ radians ($30°$) at best.

Our second insight is that although resolution in the angle axis is indeed limited, we can compensate for this by arbitrarily increasing the doppler axis. We demonstrate this effect by simulating a length 4 uniform antenna array in motion in Fig. 2 (center). As shown, the boundaries of the pattern become well resolved with a sufficiently large doppler resolution. Specifically, we find that the expected doppler-angle heatmap takes the form of a fat sinusoidal shape with smooth edges. This shows that the doppler-angle heatmap representation is both evocative and informative in spite of low angular resolution.

5 Learning-based Flow Estimation and Collision Avoidance

As mentioned in Sec. 1, real-world measurements of radar signals are prone to multipath effects and clutter. Intuitively, multipath creates reflections in the environment that appear to be at different locations and move at different velocities than real objects. This is analogous to looking at a mirror. If you observe an object approaching the mirror, the object and the reflection appear to be moving in opposite directions. Similarly, clutter leads to overlap of signal from multiple reflectors in the environment at the radar, making it harder to disentangle individual patterns. Instead of modelling these effects explicitly, we learn an implicit model of these effects by leveraging a data-driven approach. This is consistent with recent work in this space. Specifically, [7, 41, 42] use a data-driven learning-based approach for different radar sensing applications.

Our approach differs in the following way: such past work uses a sparse point cloud input representation, where each point corresponds to a potential reflector in the environment. As this format mimics lidar-like outputs but is orders of magnitudes more sparse, applying existing lidar point cloud processing techniques to such data remains exceedingly challenging. On the other hand, our work leverages dense doppler-angle heatmaps. Since such heatmaps are images, they are directly compatible with mature and highly optimizable convolutional architectures commonly used in computer vision. We opt for an all heatmap and CNN-based approach, leveraging doppler-angle heatmaps and range-angle heatmaps for velocity/flow estimation and collision detection respectively. We show that this heatmap-based approach is a uniquely advantageous, as unlike past work, we can deploy our models on a small UAV and operate at high update rates (up to 50 Hz).

5.1 Radar Selection

As noted before, BatMobility uses two mmWave radars – one forward facing and one downward facing. There are multiple off-the-shelf radars available with different antenna layouts, frequency bands, etc. We choose TI xWR1843 [22] for the forward facing radar. This is because this radar has a large linear virtual antenna array. It has an antenna array comprised of 3 Tx antennas and 4 Rx antennas, i.e., 8 effective antennas arranged horizontally in a line. The large horizontal aperture provides very fine-grained azimuth resolution. This is essential for the UAV as it needs to move left or right depending on potential for collision along a given direction. The high angular resolution for this radar allows the UAV to make subtle shifts in its heading direction and effectively avoid collisions.

On the other hand, the downward facing radar needs to identify UAV motion parallel to the ground for motion flow estimation. This motion can happen in any direction parallel to the ground, which means we require a 2-D antenna array. This functionality is provided by the TI xWR6843ISK-ODS [23]. The antenna pattern is shown in Fig. 5. Since each antenna array is much shorter, we sacrifice angular resolution for the ability to measure both azimuth and elevation angle with respect to the radar simultaneously.

We run independent compute pipelines on each of these radars on a low-end compute board present on the UAV. This board feeds the outputs to an unmodified flight controller.

5.2 Estimating Radio Flow

First, we describe our radio flow estimation pipeline (outlined in Fig. 6). The flow estimation pipeline consists of three stages: (a) a preprocessing stage, where signal processing functions like doppler-angle heatmaps and altitude estimates are extracted from the radar samples, (b) an inference stage, where the heatmaps are given to a learned model to infer the current ground velocity, and (c) where the outputs of the inference and the altitude estimates are combined using trigonometry to obtain the final angular flow values. We note that, in principle, the UAV just needs to estimate its velocity. However, we perform the third step in order to integrate our system into an unmodified flight controller, which is tuned to receive optical flow values and range estimates independently.

Preprocessing: Our preprocessing stage receives a 3-D matrix (i.e. radar cube) $\mathbf{S} \in \mathbb{C}^{N_c \times N_a \times N_s}$ of samples from the radar,

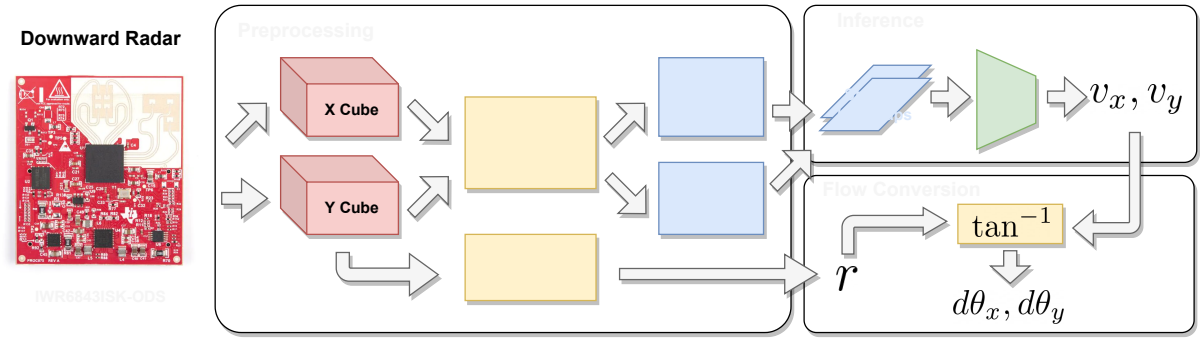


Figure 6: Flow prediction pipeline. ***Left.*** **Preprocessing stage where an incoming radar cube is turned into stacked doppler-angle heatmaps and the altitude above ground is estimated.** ***Top right.*** **Velocity prediction using a neural network.** ***Bottom right.*** **Converting velocity predictions to angular flow rate using the altitude estimate and trigonometry.**

Algorithm 1: Doppler-Angle Preprocessing

Input: Radar sample cubes $\mathbf{S}_x, \mathbf{S}_y \in \mathbb{C}^{N_c \times 4 \times N_s}$
Parameters: Angular range θ and resolution $d\theta$, range subsampling factor K, resize shape M, N
Output: Stacked heatmap $\mathbf{H} \in \mathbb{R}^{H \times W \times 2}$

$\mathbf{S}_x \leftarrow$ `SubsampleRange` $(\mathbf{S}_x, K)$;
$\mathbf{S}_y \leftarrow$ `SubsampleRange` $(\mathbf{S}_y, K)$;
$\mathbf{S'}_x \leftarrow$ `DopplerAngleFFT` $(\mathbf{S}_x, d\theta, \theta)$;
$\mathbf{S'}_y \leftarrow$ `DopplerAngleFFT` $(\mathbf{S}_y, d\theta, \theta)$;
$\mathbf{H}_x, \mathbf{H}_y \leftarrow$ `SumOverRange` $(\mathbf{S'}_x)$, `SumOverRange` $(\mathbf{S'}_y)$;
$\mathbf{H}_x, \mathbf{H}_y \leftarrow$ `Normalize` $(\mathbf{H}_x)$, `Normalize` $(\mathbf{H}_y)$;
$\mathbf{H} \leftarrow$ `Stack` $(\mathbf{H}_x, \mathbf{H}_y)$;
$\mathbf{H} \leftarrow$ `Resize` $(\mathbf{H}, M, N)$;
return $\mathbf{H}$

where N_c is the number of chirps, N_a is the number of (virtual) receive antennas, and N_s is the number of time samples taken per chirp interval. Since we are using TDM MIMO on the xWR6843ISK-ODS, $N_a = 12$ as shown in Fig. 5. On the other hand, N_c and N_s are user configurable parameters. Increasing N_c and N_s increases accuracy at the expense of computational overhead (Sec. 6 and 7).

We start by computing altitude from $\mathbf{S}$. Then we split $\mathbf{S}$ into orthogonal 1D X and Y ground-parallel components. We do this by choosing and combining samples from specific numbered RX antennas (Fig. 5).Specifically, we mimic a 1D antenna array in the X direction by stacking and adding the samples received at antennas 1, 4, 5, 8 and 2, 3, 6, 7 in those specific orders. We do the same thing for the Y direction using antennas 8, 7, 12, 11 and 5, 6, 9, 10. This gives us $\mathbf{S}_x, \mathbf{S}_y \in \mathbb{C}^{N_c \times 4 \times N_s}$.

We feed $\mathbf{S}_x, \mathbf{S}_y$ into the remainder of the preprocessing stage (Algorithm 1). First, the $\mathbf{S}_x$ and $\mathbf{S}_y$ radar cubes are downsampled along the range axis by a tunable factor K. By reducing the amount of samples, we reduce the computational burden of the following step. Next, we compute doppler-angle heatmaps at each of the N_s/K range bins of $\mathbf{S}_x, \mathbf{S}_y$ using a 2-D FFT. After this, summing over the range bins gives us doppler-angle heatmaps. Then we normalize the heatmaps such that all pixels lie within $[0.0, 1.0]$. Finally, the normalized X and Y heatmaps are stacked into a 2 channel image $\mathbf{H}$, which is then downsampled to a smaller size via interpolation. $\mathbf{H}$ is then fed to a feedforward model for inference.

Inference: Our goal is to extract 2-D velocity information from the stacked heatmaps $\mathbf{H}$, i.e. velocity along X and Y axes. From Sec. 4, we know that the velocity information exhibits as the amplitude of a sinusoidal curve in these maps. To extract this information, we opt to use a CNN.

For the model architecture, we consider several different architectures based on paring down ResNet18 in order to control the computational cost of an inference. We choose ResNet18 as the base model as it is exhibits good accuracy while still respecting the computational constraints of SBCs on small UAVs. We call the pared-down models ResNet18Mini (4 layers), ResNet18Micro (2 layers), and ResNet18Nano (1 layer). Aside from removing layers, our previous choice of downsampling $\mathbf{H}$ also plays a vital role in reducing the computational cost.

Angular Flow Conversion: A standard flight controller takes in optical flow values, which are measured as angular values (since absolute height information is unavailable to an optical sensor alone). We plan to convert the velocity measurements into these angular values for compatibility with off-the-shelf controllers. One might wonder if it is possible for the model to regress to the angular flow values directly. However, this is not possible because the doppler-angle heatmaps correspond to absolute velocities. We include a angular flow conversion step which normalizes the output

Figure 7: Data collection apparatus. *Left:* We attach a RealSense T265 tracking camera, a D435 stereo depth camera, and a DCA1000 capture card to a 3D-printed frame. We use both an IWR1843 and an IWR6843ISK-ODS (not shown) with the DCA1000. *Right:* Moving the sensor suite around by hand.

of the model into angular flow values using the current estimated altitude above ground (in preprocessing) and a basic trigonometry identity.

Dataset Collection: We aim to make the training data collection process seamless and low overhead. Therefore, we design the sensor suite depicted in Figure 7, which consists of an Intel RealSense T265 [27] tracking camera, a D435 [26] stereo depth camera, and a DCA1000 data capture card [24] attached to a 3D-printed frame. We configure the T265, D435, and DCA1000 to stream 6DoF pose readings, images, and radar cubes at a rate of 200 Hz, 30 Hz, and up to 50 Hz respectively.

In order to collect a dataset for our model, we point the sensor setup to random admissible flat surfaces in a wide variety of indoor and outdoor environments. Once a surface is chosen, we record a sequence of timestamped poses, depth images, and radar samples while moving the sensor for a fixed length of time (typically 100 seconds). The motion undertaken can be arbitrary as long as the sensor suite is continuously pointed towards the surface, however the goal is to cover as wide of a range of motions in the plane parallel to the surface as possible and to mimic the range of motion that a drone might take during flight. Then, we compute the ground truth angular flow at each time using the pose estimates from the tracking camera as well as the range readings from the depth camera, and use nearest-neighbor interpolation to assign a ground-truth for each radar cube. We ensure that our dataset consists of traces of a wide variety of surfaces, such as carpets, grass, concrete, drywall, metal, and so on. To train each model, we collect at least 25 traces of surfaces of 100 seconds each. For a radar capturing at a rate of 50Hz, this translates to 125000 datapoints in total per dataset.

Model Training: To train a model, we first split our dataset into training and validation sets of surfaces in a 60/40 ratio.

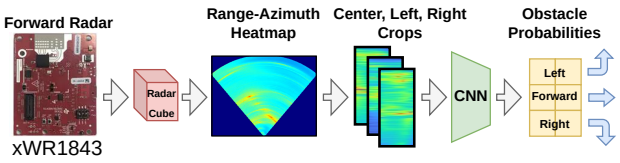


Figure 8: Collision detection. We use a CNN that classifies a the probability of an obstacle in the left, center, and right part of a color image, depth image, or radar heatmap.

For our loss function, we use the RMSE (L2) loss function, which is common for regression tasks, and use Adam optimizer [33]. We find that using a batch size of 128 and a learning rate of $1e-4$ works well across all model architectures. At the end of each epoch, we log the RMSE of the model's prediction on the validation set, and keep the model with the best performance. We typically do not need to train for any more than 200 epochs to saturate the performance on the validation set. The above is done for one set of preprocessing parameters and choice of model architecture - to find the best set of preprocessing parameters and choice of model architecture, we run a grid search on a range of possible parameter choices and choose the model with the best validation performance.

5.3 Detecting and Avoiding Collisions

With the velocity estimation from the downward facing radar, we can instruct the UAV to follow velocity setpoints. For example, we can have the UAV move forward at a constant speed, stop, turn, and move in another direction. But how do we use the forward facing radar to instruct the UAV to avoid and navigate around obstacles in the environment?

Intuition: Intuitively, collision detection relies on finding obstacles in close proximity. For collision avoidance task, we compress our radar data into a range-angle plot. At a high level, our goal is to identify if there is an obstacle directly in front of the radar, i.e. at a short range. If a possible collision is detected, we want to identify the direction to steer to (i.e., what direction is collision-free?). We note that we expect radar sensors to fare better than visual sensors in scenarios like glass windows or mirrors, which fail to register as obstacles for visual sensors.

Design: Our learning approach is inspired from [16], which trains a deep neural network that takes a color image from the forward facing camera, crops the image into a center, left, and right component, and outputs the probability of there being an obstacle in a small distance ($\leq\sim$ 1m) in the direction of each component. The idea is to move in the direction with the lowest probability of containing an obstacle. For example, if the center crop has an obstacle probability below a certain threshold, the UAV moves forwards. However, if it is blocked, it stops and turns in the direction of the crop with the lowest obstacle probability.

We adapt this basic design by modifying the input to take in 2D range-angle heatmaps from the forward facing radar instead of RGB images from a forward facing camera. We first derive the range-angle heatmaps from the radar samples using Capon beamforming [10] for an angle range of ±45 degrees from boresight and an angular resolution of 1 degree. Then, we split the resulting heatmap into a left, center, and right crop each spanning 30 degrees. Finally, the crops are given to a a neural network as a single batch to predict the probability of an obstacle in each crop.

Model Architecture: Our model architecture is again based on ResNet18 [20]. Specifically, we modify the first layer to take into account that our range-azimuth heatmaps are single-channel images, and modify the final fully connected layer to output two logits representing the probability of there being an obstacle and not being an obstacle respectively. We train the network using the standard cross-entropy loss function for classification tasks.

Dataset Collection: To collect the dataset, we use the same sensor setup as depicted in Figure 7. We move the sensors around by hand in a large variety of real world environments and conditions. But how do we label the ground-truth for each crop of each range-azimuth heatmap? To find range-azimuth crops containing obstacles, we collect trajectories where the radar is pointed at random obstacles (i.e. going down the side of a building, towards the wall of a hallway while walking through it) and label the crops of all collected range-azimuth heatmaps as containing obstacles. Conversely, to find obstacle-free range-azimuth heatmap crops, we collect trajectories keeping the radar far away from any obstacles (i.e. wandering in an open atrium). To prevent bias in the dataset, we collect an equal number of obstacle-filled and obstacle-free range-azimuth heatmap crops.

Model Training: Similar to the flow prediction model, we split our dataset into a training and validation set of image crops in a ratio of 80/20. We train the models using the standard cross-entropy loss function for classification tasks, with Adam as the optimizer, a learning rate of $1e-4$, and a batch size of 128. To prevent overfitting, we do not train the model for more than 20 epochs, and keep the model with the best performance on the validation set.

6 System Design

We discuss our experimental UAV platform and the design decisions involved.

6.1 Assembling the Quadcopter

We assembled our experimental UAV platform shown in Fig. 10 from commercial off-the-shelf parts. We use a frame with a relatively small wingspan of 300mm, which allows us to fly in constrained environments. For the flight controller, we use the Pixhawk 4 Mini [4], which runs the open source PX4 Autopilot flight stack firmware [6].

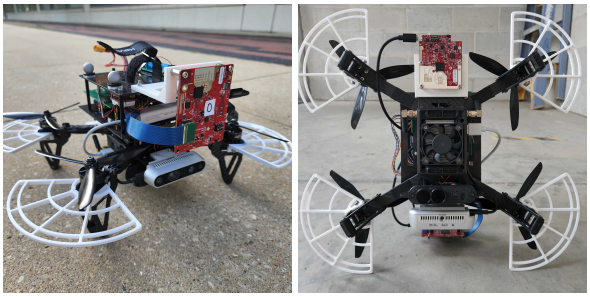


Figure 9: Our experimental UAV platform. *Left:* Frontal view, featuring the IWR1843 and the D435i camera. *Right:* Bottom view, featuring the optical flow and rangefinder module (bottom) and the AWR6843ISK (top).

Sensors: For comparison, the UAV features two independent sensor suites, a typical optical sensor suite and our own radar-based sensor suite. Only one sensor suite is used at a time throughout our experiments.

The baseline optical sensor suite consists of a downward-facing PSK-CM8JL65-CC5 unidirectional infrared rangefinder [58] and PMW3901 optical flow sensor [5] for state-estimation and a forward-facing Intel RealSense D435i [26] stereo depth camera for collision avoidance.

The radar sensor suite consists of a downward-facing 60GHz TI AWR6843ISK single-chip radar [23] and a forward-facing 77GHz TI IWR1843 single chip radar [22] for state-estimation and collision avoidance respectively. We pair each radar with a DCA1000EVM [24] board in order to extract raw radar samples rather than preprocessed point clouds from the radars.

Onboard Compute: For compute, we use an Up Board [28] single-board computer (SBC) running Ubuntu 18.04, featuring a quad-core Intel Atom x5-z8350 CPU and 4 GB of RAM. The SBC is connected to the two DCA1000 boards via a 1Gb Ethernet interface, which allows for continuous streaming of radar samples from the boards without dropped packets. It is also connected to the Pixhawk 4 flight controller through a UART interface. We use this to send the flight controller angular flow estimates from our radar flow pipeline as well as velocity setpoints from our collision avoidance module.

6.2 Flow Update Rate vs Accuracy

Recall that we use a sequence of N_c chirps to compute the doppler-angle plots. This, in addition to the chirp time T_c, determines the maximum unambiguous velocity $v_{\max}$ and velocity resolution v_{res}. Specifically,

$$v_{\max} = \frac{\lambda}{4T_c}, \quad v_{\text{res}} = \frac{\lambda}{2N_cT_c} \tag{1}$$

Figure 10: Indoor flight arena used for experiments. VICON is used to measure the UAV's motion.

Note that $v_{\max}$ and v_{res} are inversely proportional to T_c and $N_c T_c$ respectively. In practice, T_c is limited by the capabilities of the radar frontend - for most single chip radars it is possible to choose $v_{\max}$ of anywhere between 1 to 20 m/s. However, we can arbitrarily increase the velocity resolution in principle by simply increasing N_c (i.e. accumulating more chirps).

We call the time needed to collect these chirps the integration time, i.e. $T_{\text{int}} = N_c T_c$. Furthermore, data arrives from the DCA1000 in batches of 20-30 chirps at a time, which is typically smaller than N_c. Whenever a new batch of chirp data arrives, we accumulate the chirp data into a FIFO circular buffer of size N_c and then run the flow estimation pipeline using all the chirps accumulated in the buffer. Hence, the rate of arrival of new chirp data determines the flow update rate f_{flow}, and the total latency of the system comprises the integration time well as the flow estimation time.

Clearly, the flow estimation pipeline must complete before the next batch of chirp data arrives, otherwise some data will be dropped. Hence, higher update rates impose stricter computational requirements. This leads to a tradeoff between flow update rate and estimation accuracy. The higher the value of N_c, the better the accuracy of our velocity estimates. However, it also increases the system latency. This means the reaction time of the UAV is increased. By contrast, lowering N_c allows for faster and lower latency updates at the expense of accuracy.

We explore tradeoff empirically in Sec. 7.3. Specifically, we tune the radar processing pipeline and the ML models using the parameters in Tab. 1 to achieve update rates of 15Hz to 50 Hz, and compare their performance.

7 Evaluation

We conduct our experiments in an 6×6 m indoor flight arena, which features a textureless soft foam floor interspersed with textured hard foam tiles which function as landing pads. We obtain the ground truth motion of the UAV throughout its flight using a VICON capture system.

f_{flow}	T_{int}	θ_{res}	$\theta_{\max}$	K	$M \times N$	**Model**
15Hz	140ms	6°	60°	3	24x24	Mini
30Hz	100ms	3°	60°	4	24x24	Micro
40Hz	75ms	6°	60°	6	24x24	Micro
50Hz	60ms	6°	60°	12	12x24	Nano

Table 1: Preprocessing parameters and model architectures for our 15Hz, 30Hz, 40Hz, and 50Hz models.

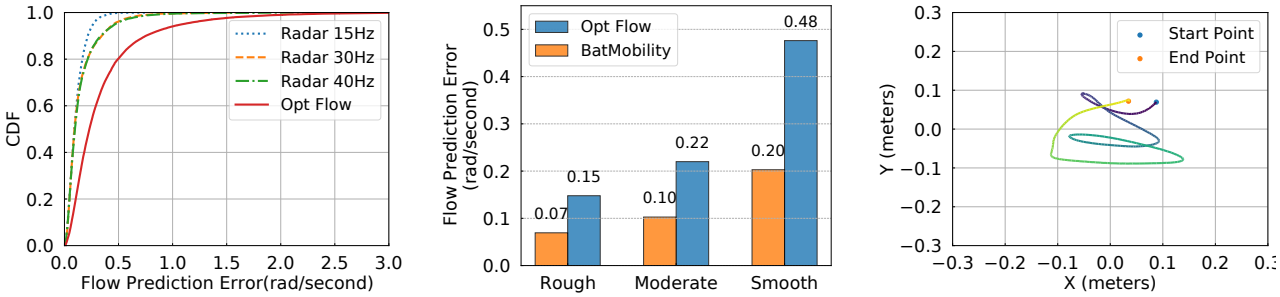
7.1 Flow Estimation Accuracy

First, we evaluate BatMobility's flow estimation accuracy under different radar update rate settings and compare with an optical flow sensor as a baseline. For each setting, we collect 12 test traces in different lighting conditions (dark, and bright), and different textures (rough, moderate, and smooth). These test traces are taken from environments which do not appear in the train/validation dataset. We extract the flow prediction results from these trajectories and show the overall flow estimation accuracy of different hardware settings in Fig. 11(a). It shows that radar settings with 15, 30, 40Hz always give better predictions than optical flow, and they achieve the median prediction error of 0.091, 0.093 and 0.097 rad/seconds, compared to the optical flow error of 0.230 rad/seconds. This is because optical flow generally does worse on transparent, featureless and dark environments while our radar design is robust to these factors.

To analyze how surface textures affect the flow prediction for both optical flow and BatMobility, we show comparison results in Fig. 11(b) on three different kinds of surfaces: rough (carpet), moderate (floor and table), and smooth (whiteboard and metal). The results show that BatMobility achieves 0.08, 0.12, 0.28 rad/second lower estimation error compared to optical flow in the rough, moderate, and smooth surface respectively. This shows that BatMobility equipped with downward facing radar is capable of achieving robust performance in variety of surfaces and this ability is essential to stabilize the UAV during real flight. Note that the performance of BatMobility worsens with the smoothness of the surface because extremely smooth surfaces (like whiteboard) are closer to pure reflectors as opposed to scatterers. Since BatMobility uses the scattering effect for doppler-estimation, its performance worsens on these polished surfaces.

7.2 Position Hold Performance

We compare the position holding (hovering) performance of the UAV in a VICON room when using the optical flow sensor and BatMobility. These systems are needed to provide feedback to the UAV to correct drifts in its position. This is a challenging test because BatMobility must capture very low velocity motion to avoid drift in position. Futhermore, note that the flight room flooring (foam) is a completely novel surface that was not seen during BatMobility's training.



(a) Flow prediction accuracy (b) Accuracy variation across surfaces (c) Qualitative position hold result

Figure 11: (a) BatMobility outperforms an off-the-shelf optical flow sensor in accurate flow prediction (2.5X) improvement. (b) BatMobility outperforms the sensor across surface types, but worsens due to low scattering on smooth surfaces like whiteboards and metal. (c) A UAV can hold its position by responding to motion feedback from BatMobility.

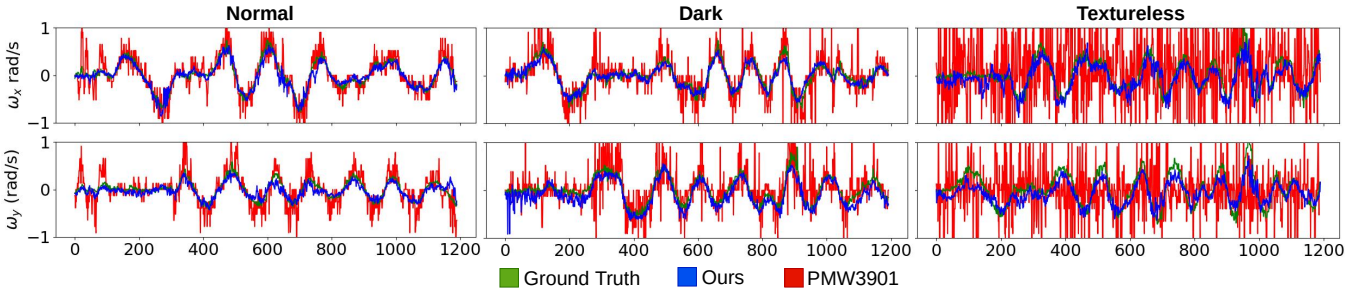


Figure 12: Sample test traces for our 40 Hz model under three operating conditions on a rough surface. Note that the performance of BatMobility is relatively invariant, whereas optical flow struggles under visually adverse conditions.

We show qualitative results in Fig. 11(c). The figure shows a 20 seconds trajectory of the UAV equipped with BatMobility when we set it to be in the position holding mode. It shows that BatMobility manages to stabilize the UAV within the radius of 0.1m by solely using radar. Any drifts in its motion are estimated by BatMobility accurately and leads to course correction by the UAV.

Next, we present quantitative results. We test the optical flow sensor performance in three different environment conditions: the textured surface with sufficient lighting, the dark environment, and the textureless surface. For each setting, we collect 5 trajectories and each trajectory has a duration of 10 seconds. We compare the optical flow sensor with the 40Hz radar pipeline and the results are shown in Figure 13(a). As shown, the UAV equipped with optical flow sensor can hold the position well in the normal environment with the average deviation of 0.07 meters. Note that, optical flow performs better than BatMobility's deviation of 0.19 meters in such settings (despite a worse flow accuracy than BatMobility). This is because the optical flow sensor operates at a higher update rate (at least 50 Hz) and provides more frequent feedback. However, when it comes to the dark and textureless environment, we notice that the optical flow sensor quickly loses the ability to give correct flow prediction and the UAV crashes in less than 10 seconds.

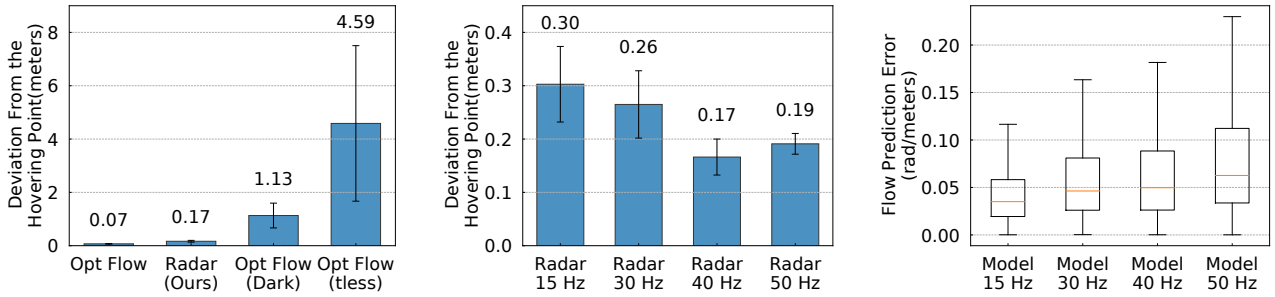
7.3 Update Rate vs Performance

We investigate the tradeoff between the update rate and hovering performance. We plot the median deviation from intended position in Fig. 13(b) for different update rates. It shows that when update rate goes up, the deviation from the hovering point gets smaller. This is because more frequent updates allow the flight controller to react sooner to any drift from its position.

However, at 50 Hz, the deviation increases. This is because of two reasons: (a) As shown in Fig. 13(c), the flow prediction accuracy is lower at higher update rates, and (b) we observe that our software platform is unable to consistently output updates at 50 Hz, thereby dropping some updates. This is a limitation of our current design and is fixable by a more hardware focused design (e.g., using FPGAs). We also note that we compare to a hardware implementation of a commercial optical flow sensor which outputs 50 Hz consistently. Therefore, we stick to 40 Hz as our final design.

7.4 Collision Detection

We test the collision detection ability when UAV is facing different materials including whiteboard, glass, metal, wall, and net in our flight room. We compare the collision detection accuracy with two other baseline modalities: depth from an infrared stereo camera, and RGB. For each surface, we collect 2 minute traces of data and results are shown in Fig. 14. The result shows that BatMobility achieves 93%, 78%, 77%,



(a) Median Deviation from Position (b) Deviation vs Update Rate (c) Flow accuracy vs update rate

Figure 13: Loiter Test. (a) UAV equipped with BatMobility holds its position, but optical flow fails in dark and textureless conditions. (b) Higher update rates support better hovering performance, in spite of higher flow prediction errors shown in (c).

(a) Accuracy on different materials (b) Examples of RGB collision (c) Examples of Radar collision

Figure 14: Collision detection. (a) We test the collision detection accuracy on different materials. BatMobility achieves highest accuracy in vast majorities of cases. (b) Two examples where RGB system misses the obstacle but BatMobility succeeds. (c) Two examples of BatMobility failure cases on specular surfaces.

95%, and 84% collision prediction accuracy on the aforementioned surfaces respectively. We outperform the baselines. Vision-based methods suffer against glass (transparency) and metal (reflective surface). BatMobility's performance is also reduced against glass and metal due to their near specular reflection as opposed to scattering. Note that, these are frame-level collision prediction results. A flight controller can aggregate multiple frames before it decides to stop or move forward.

We show two examples in Fig. 14(b) where the RGB-aided system misses the obstacle. This is because nets and glass are inconspicuous to RGB cameras, especially when the UAV is moving. We also show 2 cases in Fig. 14(c) where BatMobility fails to make right decisions. This is because metal and board surfaces are smooth and there is a chance that the transmitted signal undergoes specular reflection and doesn't reflect back to the receive antenna.

7.5 Flow Ablation Study

What effect does varying the preprocessing parameters and choice of model architecture have on the flow prediction performance? We retrain the 15 Hz model with different parameters and report the validation accuracy in Table 2. We find that the two biggest determinants are the choice of model architecture (i.e. ResNet18, ResNet18Mini) and the integration time T_{int}. By contrast, the effect of varying preprocessing parameters (θ_{res}, θ_{max}, K, M, N) on the flow accuracy is usually marginal, yet it can greatly reduce the pipeline runtime.

Parameters					ResNet18	Mini
T_{int}	θ_{res}	θ_{max}	K	M, N		
70ms	4°	90°	3	32, 32	0.071	0.076
140ms	4°	90°	3	32, 32	0.069	0.071
140ms	6°	60°	3	32, 32	0.071	0.072
140ms	6°	60°	4	32, 32	0.069	0.073
140ms	6°	60°	4	24, 24	0.071	0.074

Table 2: Effect of preprocessing parameters and model architecture on flow validation RMSE (rad/s). Lower is better.

8 Related Work

RF-Based Odometry: There is a large body of work on the use of spinning radar for ego-motion estimation of cars [11, 32, 47, 50, 52]. Although such radars have exceptional range and angular resolution, their bulkiness, power consumption, and mechanical nature make them unsuitable for small UAVs.

Recent work has focused on bringing radar-based odometry techniques to more compact single-chip mmWave radar sensors. milliEgo [42] is a learning-based method that fuses

radar point clouds with IMU data. [34] fuses doppler shift from radar point clouds with IMU readings using a factor graph. [51] extends this technique to 3D by leveraging two orthogonally placed radars in addition to an IMU. All such approaches use outward facing radars and thus rely on the presence of static reflectors in the surrounding environment. By contrast, BatMobility's odometry relies solely on a single ground facing radar and does not require an auxiliary IMU. Furthermore, such works are tested only on ground robots, whereas BatMobility addresses the challenges of integrating radar-based odometry in a plug-and-play fashion on a real UAV.

Finally, another body of work uses wireless localization techniques to provide odometry estimates to UAVs [12, 37, 64, 71]. These approaches require pre-existing wireless anchor points in the environment. BatMobility does not require such infrastructure and instead relies purely on on-board sensing, thus enabling true autonomy.

RF-Based Obstacle Detection: In an automotive context, mmWave radar is often used to detect and image surrounding cars [7, 18, 48]. Analogously, various works propose the use of mmWave radar on UAVs to detect various obstacles that it might crash into [13, 57, 59, 66]. We differ from prior works in two ways. Firstly, we do not assume prior knowledge on the kind of obstacle we wish to detect (human, car, another UAV) and instead opt for a generic notion of an obstacle as being anything which a UAV can physically collide. Secondly, unlike prior approaches which operate on preprocessed point clouds, we mindfully opt for an end-to-end approach based on radar heatmaps to prevent any task-relevant cues from being lost during the point cloud conversion process.

Control Policies for UAVs: Strategies for collision-free flight for UAVs fall under two main categories. The first comprises classical approaches based on planning paths in metric maps of the environment. These works are highly reliant on accurate geometric maps of the environment, which are obtained through relatively expensive optical depth sensors (i.e. stereo depth cameras, lidars). Such approaches include [40, 65, 72]. The alternative paradigm is learning-based approaches that map raw sensor data to actions [38]. For example, vision-based approaches have become popular in recent years due to the ubiquity of cameras and better means of generating image data at scale. Work in this space includes [9, 16, 30, 54, 56]. We note that BatMobility is designed with this paradigm in mind.

9 Concluding Discussion

We present BatMobility, a learning-based pure radar perception system that enables autonomous drones to fly in unstructured and unknown environments. We highlight the advantages and disadvantages of an RF-only approach when compared to a conventional optical sensor based system. The core of our approach is a novel radio flow primitive that can be broadly useful in other contexts involving motion sensing, As such, we hope this work opens up new frontiers in the use of RF sensing for robotics, self-driving vehicles, VR/AR, and beyond. We conclude with a brief discussion of limitations and future work:

Deploying in External Environments: We test BatMobility in a controlled flight room environment to measure its performance with VICON cameras. However, our evidence suggests that the system would work just as well or even better outdoors. This is because our experiments were performed on a foam and smooth concrete floor, which appear as highly smooth surfaces to the radar. As our results show, BatMobility works better on rough surfaces, which are ubiquitous in outdoor environments (e.g. grass, asphalt roads).

System Constraints: Our system is primarily constrained by the maximum operating range and maximum unambiguous velocity of the radar. These are determined by the radar chirp parameters as explained in Sec. 2 and 6. For our evaluation, we choose chirps with maximum range within 3-5 m and maximum velocity of 1.5-2 m/s. We believe this is appropriate for BatMobilityś ideal use case, i.e. flying cautiously at low altitudes in order to prevent collisions with surrounding objects. One potential avenue of future exploration is to adaptively tune the radar's chirp parameters depending on the current scenario (i.e. increasing maximum range at higher altitudes, increasing maximum velocity when moving faster, etc).

Achieving Agile Flight: We focus only on 2-D action primitives in BatMobility (i.e. moving forward, turning, etc). Although such an action set is practical for navigation purposes, it does not encompass the full set of a quadcopter's capabilities. This prevents us from, for example, performing drone acrobatics [31], or following time-optimal trajectories through environments [15]. We leave such exploration to future work.

Acknowledgments — We thank the reviewers and our anonymous shepherd for their insightful comments and suggestions on improving this paper. This work was supported in part by NSF RINGS Award 2148583. This work was carried out in part in the Intelligent Robotics Laboratory, University of Illinois Urbana-Champaign. We thank John M. Hart for help regarding the flying arena. We thank Shahab Nikkhoo for his guidance and suggestions. We thank Kris Hauser for letting us use his 3D printer. We are grateful to Jayanth Shenoy, Bill Tao, Maleeha Masood, Ishani Janveja, and Om Chabra for their feedback on initial drafts.

References

[1] Fadel Adib, Zachary Kabelac, Dina Katabi, and Robert C. Miller. 2014. 3D Tracking via Body Radio Reflections. In Proceedings of the 11th USENIX Conference on Networked Systems Design and Implementation (Seattle, WA) (NSDI'14). USENIX Association, USA, 317–329.

[2] Fadel Adib and Dina Katabi. 2013. See through walls with WiFi!. In Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM (Hong Kong China, 2013-08-27). ACM, 75–86. https://doi.org/10.1145/2486001.2486039

[3] Naveed Anwar, Muhammad Amir Izhar, and Fawad Ahmed Najam. 2018. Construction monitoring and reporting using drones and unmanned aerial vehicles (UAVs). In The Tenth International Conference on Construction in the 21st Century (CITC-10). 2–4.

[4] PX4 Autopilot. [n. d.]. Pixhawk 4 Mini. http://docs.px4.io/main/en/flight_controller/pixhawk4_mini.html.

[5] PX4 Autopilot. [n. d.]. PMW3901-Based Flow Sensors. https://docs.px4.io/main/en/sensor/pmw3901.html.

[6] PX4 Autopilot. [n. d.]. PX4 Open Source Autopilot. https://px4.io/.

[7] Kshitiz Bansal, Keshav Rungta, Siyuan Zhu, and Dinesh Bharadia. [n. d.]. Pointillism: accurate 3D bounding box estimation with multi-radars. In Proceedings of the 18th Conference on Embedded Networked Sensor Systems (Virtual Event Japan, 2020-11-16). ACM, 340–353. https://doi.org/10.1145/3384419.3430783

[8] Lorenzo Bertizzolo, Salvatore D'oro, Ludovico Ferranti, Leonardo Bonati, Emrecan Demirors, Zhangyu Guan, Tommaso Melodia, and Scott Pudlewski. 2020. SwarmControl: An automated distributed control framework for self-optimizing drone networks. In IEEE INFOCOM 2020-IEEE Conference on Computer Communications. IEEE, 1768–1777.

[9] Rogerio Bonatti, Ratnesh Madaan, Vibhav Vineet, Sebastian Scherer, and Ashish Kapoor. 2020. Learning Visuomotor Policies for Aerial Navigation Using Cross-Modal Representations. In 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (Las Vegas, NV, USA, 2020-10-24). IEEE, 1637–1644. https://doi.org/10.1109/IROS45743.2020.9341049

[10] Jack Capon. 1969. High-resolution frequency-wavenumber spectrum analysis. Proc. IEEE 57, 8 (1969), 1408–1418.

[11] Sarah H. Cen and Paul Newman. 2018. Precise Ego-Motion Estimation with Millimeter-Wave Radar Under Diverse and Challenging Conditions. In 2018 IEEE International Conference on Robotics and Automation (ICRA). IEEE, Brisbane, QLD, 1–8. https://doi.org/10.1109/ICRA.2018.8460687

[12] Guoxuan Chi, Zheng Yang, Jingao Xu, Chenshu Wu, Jialin Zhang, Jianzhe Liang, and Yunhao Liu. 2022. Wi-drone: wi-fi-based 6-DoF tracking for indoor drone flight control. In Proceedings of the 20th Annual International Conference on Mobile Systems, Applications and Services (MobiSys '22). Association for Computing Machinery, New York, NY, USA, 56–68. https://doi.org/10.1145/3498361.3538936

[13] Sedat Dogru and Lino Marques. 2020. Pursuing Drones With Drones Using Millimeter Wave Radar. 5, 3 (2020), 4156–4163. https://doi.org/10.1109/LRA.2020.2990605

[14] G Klaric Felic, RJ Evans, Hoa Thai Duong, Hoang Viet Le, J Li, and E Skafidas. 2015. Single-chip millimeter wave radar. Microwave J 58 (2015), 108–116.

[15] Philipp Foehn, Dario Brescianini, Elia Kaufmann, Titus Cieslewski, Mathias Gehrig, Manasi Muglikar, and Davide Scaramuzza. [n. d.]. AlphaPilot: Autonomous Drone Racing. In Robotics: Science and Systems XVI (2020-07-12). Robotics: Science and Systems Foundation. https://doi.org/10.15607/RSS.2020.XVI.081

[16] Dhiraj Gandhi, Lerrel Pinto, and Abhinav Gupta. [n. d.]. Learning to fly by crashing. In 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (2017-09). 3948–3955. https://doi.org/10.1109/IROS.2017.8206247 ISSN: 2153-0866.

[17] Sahil Garg, Gagangeet Singh Aujla, Aiman Erbad, Joel JPC Rodrigues, Min Chen, and Xianbin Wang. 2021. Guest editorial: Blockchain envisioned drones: Realizing 5G-enabled flying automation. IEEE Network 35, 1 (2021), 16–19.

[18] Junfeng Guan, Sohrab Madani, Suraj Jog, Saurabh Gupta, and Haitham Hassanieh. [n. d.]. Through Fog High-Resolution Imaging Using Millimeter Wave Radar. In 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (Seattle, WA, USA, 2020-06). IEEE, 11461–11470. https://doi.org/10.1109/CVPR42600.2020.01148

[19] Peter Hart, Nils Nilsson, and Bertram Raphael. [n. d.]. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. 4, 2 ([n. d.]), 100–107. https://doi.org/10.1109/TSSC.1968.300136

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

[21] Berthold K.P. Horn and Brian G. Schunck. [n. d.]. Determining optical flow. 17, 1 ([n. d.]), 185–203. https://doi.org/10.1016/0004-3702(81)90024-2

[22] Texas Instruments. [n. d.]. IWR1843 single-chip 76-GHz to 81-GHz industrial radar sensor evaluation module. https://www.ti.com/tool/IWR1843BOOST.

[23] Texas Instruments. [n. d.]. IWR6843 intelligent mmWave overhead detection sensor (ODS) antenna plug-in module. https://www.ti.com/tool/IWR6843ISK-ODS.

[24] Texas Instruments. [n. d.]. Real-time data-capture adapter for radar sensing evaluation module. https://www.ti.com/tool/DCA1000EVM.

[25] Texas Instruments. 2020. The fundamentals of millimeter wave radar sensors. https://www.tij.co.jp/lit/wp/spyy005a/spyy005a.pdf.

[26] Intel. [n. d.]. Depth Camera D435. https://www.intelrealsense.com/depth-camera-d435/.

[27] Intel. [n. d.]. Intel® RealSense™ Tracking Camera T265. https://www.intelrealsense.com/tracking-camera-t265/.

[28] Intel. [n. d.]. Up Board Series. https://up-shop.org/up-board-series.html.

[29] Michael A Johnson and Mohammad H Moradi. 2005. PID control. Springer.

[30] Elia Kaufmann, Antonio Loquercio, Rene Ranftl, Alexey Dosovitskiy, Vladlen Koltun, and Davide Scaramuzza. [n. d.]. Deep Drone Racing: Learning Agile Flight in Dynamic Environments. In Proceedings of The 2nd Conference on Robot Learning (2018-10-23). PMLR, 133–145. https://proceedings.mlr.press/v87/kaufmann18a.html ISSN: 2640-3498.

[31] Elia Kaufmann, Antonio Loquercio, René Ranftl, Matthias Müller, Vladlen Koltun, and Davide Scaramuzza. 2020. Deep Drone Acrobatics. (2020). https://doi.org/10.48550/ARXIV.2006.05768 Publisher: arXiv Version Number: 2.

[32] Dominik Kellner, Michael Barjenbruch, Jens Klappstein, Jürgen Dickmann, and Klaus Dietmayer. 2014. Instantaneous ego-motion estimation using multiple Doppler radars. In 2014 IEEE International Conference on Robotics and Automation (ICRA). 1592–1597. https://doi.org/10.1109/ICRA.2014.6907064 ISSN: 1050-4729.

[33] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014).

[34] Andrew Kramer, Carl Stahoviak, Angel Santamaria-Navarro, Aliakbar Agha-mohammadi, and Christoffer Heckman. 2020. Radar-Inertial Ego-Velocity Estimation for Visually Degraded Environments. In 2020 IEEE International Conference on Robotics and Automation (ICRA). IEEE, Paris, France, 5739–5746. https://doi.org/10.1109/

ICRA40945.2020.9196666

[35] Megan Landau and Sebastian Van Delden. 2017. A system architecture for hands-free UAV drone control using intuitive voice commands. In Proceedings of the companion of the 2017 acm/ieee international conference on human-robot interaction. 181–182.

[36] Steven M. LaValle. 1998. Rapidly-exploring random trees : a new tool for path planning. The annual research report (1998).

[37] Anton Ledergerber, Michael Hamer, and Raffaello D'Andrea. [n. d.]. A robot self-localization system using one-way ultra-wideband communication. In 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (Hamburg, Germany, 2015-09). IEEE, 3131–3137. https://doi.org/10.1109/IROS.2015.7353810

[38] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. 2016. End-to-end training of deep visuomotor policies. 17, 1 (2016), 1334–1373. Publisher: JMLR. org.

[39] Jaime Lien, Nicholas Gillian, M Emre Karagozler, Patrick Amihood, Carsten Schwesig, Erik Olson, Hakim Raja, and Ivan Poupyrev. 2016. Soli: Ubiquitous gesture sensing with millimeter wave radar. ACM Transactions on Graphics (TOG) 35, 4 (2016), 1–19.

[40] Jiahao Lin, Hai Zhu, and Javier Alonso-Mora. [n. d.]. Robust Vision-based Obstacle Avoidance for Micro Aerial Vehicles in Dynamic Environments. In 2020 IEEE International Conference on Robotics and Automation (ICRA) (Paris, France, 2020-05). IEEE, 2682–2688. https://doi.org/10.1109/ICRA40945.2020.9197481

[41] Chris Xiaoxuan Lu, Stefano Rosa, Peijun Zhao, Bing Wang, Changhao Chen, John A. Stankovic, Niki Trigoni, and Andrew Markham. [n. d.]. See through smoke: robust indoor mapping with low-cost mmWave radar. In Proceedings of the 18th International Conference on Mobile Systems, Applications, and Services (Toronto Ontario Canada, 2020-06-15). ACM, 14–27. https://doi.org/10.1145/3386901.3388945

[42] Chris Xiaoxuan Lu, Muhamad Risqi U. Saputra, Peijun Zhao, Yasin Almalioglu, Pedro P. B. de Gusmao, Changhao Chen, Ke Sun, Niki Trigoni, and Andrew Markham. 2020. milliEgo: single-chip mmWave radar aided egomotion estimation via deep sensor fusion. In Proceedings of the 18th Conference on Embedded Networked Sensor Systems. ACM, Virtual Event Japan, 109–122. https://doi.org/10.1145/3384419.3430776

[43] Bruce D. Lucas and Takeo Kanade. [n. d.]. An iterative image registration technique with an application to stereo vision. In Proceedings of the 7th international joint conference on Artificial intelligence - Volume 2 (San Francisco, CA, USA, 1981-08-24) (IJCAI'81). Morgan Kaufmann Publishers Inc., 674–679.

[44] Yunfei Ma, Nicholas Selby, and Fadel Adib. [n. d.]. Drone Relays for Battery-Free Networks. In Proceedings of the Conference of the ACM Special Interest Group on Data Communication (Los Angeles CA USA, 2017-08-07). ACM, 335–347. https://doi.org/10.1145/3098822.3098847

[45] Thi Thoa Mac, Cosmin Copot, Trung Tran Duc, and Robin De Keyser. 2016. AR. Drone UAV control parameters tuning based on particle swarm optimization algorithm. In 2016 IEEE International Conference on Automation, Quality and Testing, Robotics (AQTR). IEEE, 1–6.

[46] Koji Minoda, Fabian Schilling, Valentin Wüest, Dario Floreano, and Takehisa Yairi. 2021. VIODE: A Simulated Dataset to Address the Challenges of Visual-Inertial Odometry in Dynamic Environments. IEEE Robotics and Automation Letters 6, 2 (April 2021), 1343–1350. https://doi.org/10.1109/LRA.2021.3058073 arXiv: 2102.05965.

[47] Chris D. Monaco and Sean N. Brennan. 2020. RADARODO: Ego-Motion Estimation From Doppler and Spatial Data in RADAR Images. IEEE Transactions on Intelligent Vehicles 5, 3 (Sept. 2020), 475–484. https://doi.org/10.1109/TIV.2020.2973536 Conference Name: IEEE Transactions on Intelligent Vehicles.

[48] Ramin Nabati and Hairong Qi. [n. d.]. RRPN: Radar Region Proposal Network for Object Detection in Autonomous Vehicles. In 2019 IEEE International Conference on Image Processing (ICIP) (Taipei, Taiwan, 2019-09). IEEE, 3093–3097. https://doi.org/10.1109/ICIP.2019.8803392

[49] Norzailawati Mohd Noor, Alias Abdullah, and Mazlan Hashim. 2018. Remote sensing UAV/drones and its applications for urban areas: A review. In IOP conference series: Earth and environmental science, Vol. 169. IOP Publishing, 012003.

[50] Yeong Sang Park, Young-Sik Shin, and Ayoung Kim. 2020. PhaRaO: Direct Radar Odometry using Phase Correlation. In 2020 IEEE International Conference on Robotics and Automation (ICRA). 2617–2623. https://doi.org/10.1109/ICRA40945.2020.9197231 ISSN: 2577-087X.

[51] Yeong Sang Park, Young-Sik Shin, Joowan Kim, and Ayoung Kim. 2021. 3D ego-Motion Estimation Using low-Cost mmWave Radars via Radar Velocity Factor for Pose-Graph SLAM. IEEE Robotics and Automation Letters 6, 4 (Oct. 2021), 7691–7698. https://doi.org/10.1109/LRA.2021.3099365 Conference Name: IEEE Robotics and Automation Letters.

[52] M. Rapp, M. Barjenbruch, M. Hahn, J. Dickmann, and K. Dietmayer. 2017. Probabilistic ego-motion estimation using multiple automotive radar sensors. Robotics and Autonomous Systems 89 (March 2017), 136–146. https://doi.org/10.1016/j.robot.2016.11.009

[53] Ring. [n. d.]. Ring Always Home Cam | Indoor Flying Camera, Home Security Camera. https://ring.com/always-home-cam-flying-camera.

[54] Stephane Ross, Narek Melik-Barkhudarov, Kumar Shaurya Shankar, Andreas Wendel, Debadeepta Dey, J. Andrew Bagnell, and Martial Hebert. [n. d.]. Learning monocular reactive UAV control in cluttered natural environments. In 2013 IEEE International Conference on Robotics and Automation (Karlsruhe, Germany, 2013-05). IEEE, 1765–1772. https://doi.org/10.1109/ICRA.2013.6630809

[55] Maurizio Rossi, Davide Brunelli, Andrea Adami, Leandro Lorenzelli, Fabio Menna, and Fabio Remondino. 2014. Gas-drone: Portable gas sensing system on UAVs for gas leakage localization. In SENSORS, 2014 IEEE. IEEE, 1431–1434.

[56] Fereshteh Sadeghi and Sergey Levine. [n. d.]. CAD2RL: Real Single-Image Flight Without a Single Real Image. In Robotics: Science and Systems XIII (2017-07-12). Robotics: Science and Systems Foundation. https://doi.org/10.15607/RSS.2017.XIII.034

[57] Ali Safa, Tim Verbelen, Lars Keuninckx, Ilja Ocket, Matthias Hartmann, André Bourdoux, Franky Catthoor, and Georges Gielen. 2021. A Low-Complexity Radar Detector Outperforming OS-CFAR for Indoor Drone Obstacle Avoidance. (2021). https://doi.org/10.48550/ARXIV.2107.07250 Publisher: arXiv Version Number: 1.

[58] seeed studio. [n. d.]. PSK-CM8JL65-CC5 Infrared Distance Measuring Sensor. https://www.seeedstudio.com/PSK-CM8JL65-CC5-Infrared-Distance-Measuring-Sensor-p-4028.html.

[59] Lei Shi, Christopher Allen, Mark Ewing, Shahriar Keshmiri, Mikhail Zakharov, Francisco Florencio, Nahal Niakan, and Robert Knight. [n. d.]. Multichannel sense-and-avoid radar for small UAVs. In 2013 IEEE/AIAA 32nd Digital Avionics Systems Conference (DASC) (2013-10). 6E2–1–6E2–10. https://doi.org/10.1109/DASC.2013.6712628 ISSN: 2155-7209.

[60] Skydio. 2022. Autonomous Drones for Public Safety. https://www.skydio.com/public-safety.

[61] Skydio. 2022. Drones for Construction. https://www.skydio.com/construction.

[62] Paweł Smyczyński, Łukasz Starzec, and Grzegorz Granosik. 2017. Autonomous drone control system for object tracking: Flexible system design with implementation example. In 2017 22nd International Conference on Methods and Models in Automation and Robotics (MMAR). IEEE, 734–738.

[63] Lina Tang and Guofan Shao. 2015. Drone remote sensing for forestry research and practices. Journal of Forestry Research 26, 4 (2015), 791–797.

[64] Janis Tiemann and Christian Wietfeld. [n. d.]. Scalable and precise multi-UAV indoor navigation using TDOA-based UWB localization. In 2017 International Conference on Indoor Positioning and Indoor Navigation (IPIN) (Sapporo, 2017-09). IEEE, 1–7. https://doi.org/10.1109/IPIN.2017.8115937

[65] Vladyslav Usenko, Lukas von Stumberg, Andrej Pangercic, and Daniel Cremers. [n. d.]. Real-time trajectory replanning for MAVs using uniform B-splines and a 3D circular buffer. In 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (Vancouver, BC, 2017-09). IEEE, 215–222. https://doi.org/10.1109/IROS.2017.8202160

[66] Nikhil Wessendorp, Raoul Dinaux, Julien Dupeyroux, and Guido C. H. E. de Croon. [n. d.]. Obstacle Avoidance onboard MAVs using a FMCW Radar. In 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (Prague, Czech Republic, 2021-09-27). IEEE, 117–122. https://doi.org/10.1109/IROS51168.2021.9635901

[67] Wired. 2022. Drones Have Transformed Blood Delivery in Rwanda. https://www.wired.com/story/drones-have-transformed-blood-delivery-in-rwanda.

[68] Wired. 2022. Small drones are giving Ukraine an unprecedented edge. https://arstechnica.com/information-technology/2022/05/small-drones-are-giving-ukraine-an-unprecedented-edge/.

[69] The Zebra. 2021. Could Delivery Drones Be the Next Tech Privacy Violation? 88% of Americans Think So. https://www.thezebra.com/resources/home/delivery-drones-survey/.

[70] Mingmin Zhao, Yonglong Tian, Hang Zhao, Mohammad Abu Alsheikh, Tianhong Li, Rumen Hristov, Zachary Kabelac, Dina Katabi, and Antonio Torralba. [n. d.]. RF-based 3D skeletons. In Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication (Budapest Hungary, 2018-08-07). ACM, 267–281. https://doi.org/10.1145/3230543.3230579

[71] Peijun Zhao, Chris Xiaoxuan Lu, Bing Wang, Niki Trigoni, and Andrew Markham. [n. d.]. 3D Motion Capture of an Unmodified Drone with Single-chip Millimeter Wave Radar. In 2021 IEEE International Conference on Robotics and Automation (ICRA) (2021-05). 5186–5192. https://doi.org/10.1109/ICRA48506.2021.9561738 ISSN: 2577-087X.

[72] Xin Zhou, Zhepei Wang, Hongkai Ye, Chao Xu, and Fei Gao. [n. d.]. EGO-Planner: An ESDF-Free Gradient-Based Local Planner for Quadrotors. 6, 2 ([n. d.]), 478–485. https://doi.org/10.1109/LRA.2020.3047728